การควบคุมหุ่นยนต์แขนกล

ROBOT ARM CONTROLLER

โดย

นาย ณัฐวุฒิ บัวสาย

นาย ณัฐพงศ์ พันตรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

หัวข้อปริญญานิพนธ์	**การควบคุมหุ่นยนต์แขนกล**
จัดทำโดย	**นาย ณัฐวุฒิ บัวสาย** **นาย ณัฐพงศ์ พันตรี**
อาจารย์ที่ปรึกษา	**ผศ. ประดิษฐ์ วัชรพิบูลย์** **ผศ. ไพศาล สิทธิโยภาสกุล**

บทคัดย่อ

โครงงานนี้ เป็นการควบคุมหุ่นยนต์แขนกลโดย กำหนดให้หุ่นยนต์แขนกลมีลักษณะการทำงานแบ่งออกเป็น 3 แบบ คือ แบบที่ 1 นั้นเป็นการปฏิบัติงานแบบสั่งงานโดยตรง คือการที่เราควบคุมการทำงานของแขนกลโดยผู้ปฏิบัติงานเป็นผู้บังคับหุ่นยนต์แขนกลโดยการบังคับทางคอมพิวเตอร์โดยตรง และแบบที่ 2 นั้นเป็นการปฏิบัติงานแบบอัตโนมัติ คือการที่เราควบคุมการทำงานของแขนกลโดยผู้ปฏิบัติงานเป็นผู้บังคับหุ่นยนต์แขนกลโดยการสั่งงานจากชุดควบคุม โดยการติดเซ็นเซอร์ ไว้ตามข้อพับหรือจุดเคลื่อนไหวของชุดควบคุมที่ติดตั้งไว้กับแขนของผู้ปฏิบัติงาน เมื่อผู้ปฏิบัติงานเคลื่อนไหว ค่าของเซ็นเซอร์ ก็จะเปลี่ยนแปลง โดยค่านี้จะถูกนำไปประมวลผลด้วยคอมพิวเตอร์ และบังคับให้หุ่นยนต์แขนกลเคลื่อนไหวตามการเคลื่อนไหวของผู้ปฏิบัติงาน และแบบที่ 3 นั้นเป็นการจดจำการเคลื่อนไหว โดยการควบคุมนี้ในขั้นต้นจะมีขั้นตอนการทำงานเหมือนแบบสั่งงานโดยตรง และ แบบอัตโนมัติ แต่ในขณะบังคับนั้น ส่วนประมวลผลจะบันทึกค่าของเซ็นเซอร์ และรูปแบบการทำงานทุกขั้นตอนโดยละเอียดไว้ เมื่อต้องการให้ทำงานส่วนประมวลผลจะดึงข้อมูลขั้นตอนการทำงานที่ได้บันทึกไว้มาอ้างอิงในการบังคับหุ่นยนต์แขนกล จากการทำงานเช่นนี้จะเห็นว่าหุ่นยนต์แขนกล สามารถเคลื่อนไหวเลียนแบบผู้ปฏิบัติงานและจดจำการทำงานได้

Project Report ROBOT ARM CONTROLLER

BY Mr. NATTAWUT BUASAI
Mr. NATTAPONG PANTREE

Advisor Mr. PRADIT VACHRAPIBOOL
Mr. PAISAN SITHIYOPASKUL

Abstract

Robot arm controller in this project has three characteristic functions for robot arm control. First is the Manual control in which user can control the motion of robot by computer control directly, Second is the Automatic control in which user can control robot by controller tool by attach an adhere equipment sensor at the motion point of controller tool at wear arm of user while user moving the sensor value then is varied and will be processed by computerize for driving the robot arm, Third is to recognize the operation control which memory the motion of operation for every step of movement for robot arm. When user requires to operate the motion hence the reference of position in memory will process and driving robot. Robot arm is able it is seen that the to move by imitation the operation of human movement.

หัวข้อปริญญานิพนธ์	การควบคุมหุ่นยนต์แขนกล ROBOT ARM CONTROLLER
จัดทำโดย	นาย ณัฐวุฒิ บัวสาย นาย ณัฐพงศ์ พันตรี
อาจารย์ที่ปรึกษา	ผศ. ประดิษฐ์ วัชรพิบูลย์ ผศ. ไพศาล สิทธิโยภาสกุล
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2543

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....อาจารย์ที่ปรึกษา
( )
.....อาจารย์ที่ปรึกษา
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )

กิตติกรรมประกาศ

คณะผู้จัดทำโครงงานขอกราบขอบคุณ ผศ.ประดิษฐ์ วัชรพิบูลย์ และ ผศ.ไพศาล สิทธิโยภาสกุล ที่กรุณาให้คำแนะนำในการพัฒนาโครงงานและได้ให้การสนับสนุนอุปกรณ์ และอื่นๆ อย่างมาก และกราบขอบคุณครูอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ และขอกราบขอบพระคุณ คุณพ่อนิรันดร และคุณแม่มารศรี บัวสาย ที่ให้กระผมได้มีวันนี้ขึ้นมาได้ และขอขอบคุณ อาจารย์และเจ้าหน้าที่แผนกการพัฒนาและออกแบบ ของศูนย์ สสวท. ที่ให้ความช่วยเหลือสนับสนุนอุปกรณ์ด้านกลไก และขอบคุณ คุณภานุวัฒน์ คงวัฒนสิน ที่ให้คำปรึกษาในด้านซอฟแวร์ จากความร่วมมือและการให้ความช่วยเหลือของทุกท่าน ทำให้โครงงานสามารถแก้ไขปัญหาต่างๆ จนโครงงานสำเร็จลุล่วงด้วยดี จึงขอกล่าวขอบคุณทุกท่านไว้ ณ โอกาส นี้

คณะผู้จัดทำ
นาย ณัฐวุฒิ บัวสาย
นาย ณัฐพงศ์ พันตรี

สารบัญ

หน้า

บทคัดย่อ

บทที่ 1 บทนำ 1

บทที่ 2 ทฤษฎีเบื้องต้น

2.1 มอเตอร์ไฟฟ้ากระแสตรง 3

2.2 เฟือง (Gears) 11

2.3 ทฤษฎีเกี่ยวกับไมโครคอนโทรลเลอร์ 22

2.4 บอร์ดไมโครคอนโทรลเลอร์ CP-BS2SX16 23

2.5 บอร์ดขยายพอร์ต 72IOZ80 28

2.6 การติดต่อสื่อสารผ่านพอร์ตขนาน 30

2.7 การติดต่อสื่อสารผ่านพอร์ตอนุกรม 33

บทที่ 3 การออกแบบระบบอิเล็กทรอนิกส์

3.1 ลักษณะโดยรวมของระบบควบคุมหุ่นยนต์แขนกล 38

3.2 ระบบอิเล็กทรอนิกส์ที่ใช้ในโครงงาน 39

3.2.1 วงจรควบคุมทิศทางการหมุนของมอเตอร์ 40

3.2.2 วงจรภาคอินพุทแบบอนาล็อก 46

3.2.3 วงจรลิมิตสวิตช์ 51

3.2.4. วงจรในส่วนของมือจับชิ้นงาน 53

3.2.5 วงจรเลือก Connect Parallel 55

3.3 การต่อใช้งานวงจรอินพุทอนาล็อก (ของชุดแขนควบคุม) 57

3.4 การต่อใช้งานวงจร_SW. Mode Hand 57

3.5 การต่อใช้งานวงจร Connect Parallel Port 58

3.6 การใช้งานฮาร์ดแวร์ของบอร์ด CP-BS2SX 60

3.7 การต่อระหว่างบอร์ด CP-BS2SX กับบอร์ด 8255 61

3.8 การต่อ บอร์ด 8255 กับภาค DRIVER 62

3.9 การต่อวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (ของชุดหุ่นยนต์) 62

สารบัญ (ต่อ)

	หน้า
บทที่ 4 การออกแบบระบบกลไกของหุ่นยนต์แขนกล	
4.1 ส่วนประกอบของแขนกล	65
4.2 ลักษะการเคลื่อนไหวของหุ่นยนต์แขนกล	66
4.3 ลักษณะเฉพาะของส่วนประกอบต่างๆ	68
4.4 ส่วนประกอบของชุดแขนควบคุม	74
4.5 ส่วนประกอบของชุดควบคุมการเคลื่อนที่ของหุ่นยนต์แขนกล	76
บทที่ 5 โปรแกรมที่ใช้ควบคุมหุ่นยนต์	
5.1 โปรแกรมควบคุมหลัก	78
5.1.1 โปรแกรมเมนูหลัก	78
5.1.2 โปรแกรมชุดควบคุมอัตโนมัติ	80
5.1.3 โปรแกรมชุดควบคุมแบบ Manual	82
5.1.4 โปรแกรมทดสอบการทำงาน	83
5.1.5 โปรแกรมบันทึกการทำงาน	84
5.2 โปรแกรมควบคุมภายในหุ่นยนต์แขนกล	85
บทที่ 6 สรุปและวิจารณ์	86
หนังสืออ้างอิง	
ภาคผนวก ก. วงจรรวมและการวางอุปกรณ์และลายวงจรพิมพ์	89
ข. คู่มือการใช้งานและโปรแกรมที่ใช้งานในโครงงาน	95
ค. DATA SHEET	141

สารบัญรูปภาพ

	หน้า
รูปที่ 1.1 ลักษณะการทำงานโดยรวม	2
รูปที่ 2.1.1 ส่วนประกอบต่างๆ ของมอเตอร์ไฟฟ้ากระแสตรง	6
รูปที่ 2.1.2 ชนิดของมอเตอร์ไฟฟ้ากระแสตรง	7
รูปที่ 2.1.3 วงจรสมมูลที่สถานะคงตัวของอาร์เมเจอร์	8
รูปที่ 2.1.4 กราฟแสดงความสัมพันธ์ระหว่างแรงบิด และความเร็วของมอเตอร์ไฟฟ้ากระแสตรง	10
รูปที่ 2.2.1 เฟืองชนิดต่าง ๆ	11
รูปที่ 2.2.2 เฟืองตรงธรรมดา	12
รูปที่ 2.2.3 แรงตามแนวแกน (Axial Force) ที่เกิดบนเฟืองตรงฟันเฉียงขณะส่งถ่ายกำลัง	13
รูปที่ 2.2.4 เฟืองดอกจอก	15
รูปที่ 2.2.5 เฟืองเกลียวสกรู	16
รูปที่ 2.2.6 เฟืองหนอน	17
รูปที่ 2.2.7 โซ่ลูกกลิ้งและบูช	18
รูปที่ 2.2.8 ล้อโซ่	19
รูปที่ 2.2.9 ล้อความฝืด	20
รูปที่ 2.3 โครงสร้างการทำงานของไมโครคอนโทรลเลอร์	22
รูปที่ 2.4.1 บอร์ด CP-BS2SX	23
รูปที่ 2.4.2 พอร์ตใช้งานของ CP-B2SX	27
รูปที่ 2.5 ลักษณะการเชื่อมต่อบอร์ด 72I0Z80	29
รูปที่ 2.6.1 คอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin	30
รูปที่ 2.6.2 คอนเน็คเตอร์แบบ Centronics	31
รูปที่ 2.6.3 Centronics Handshake	32
รูปที่ 2.7.1 คอนเน็คเตอร์แบบ DB-25	34
รูปที่ 2.7.2 คอนเน็คเตอร์แบบ DB-25	34
รูปที่ 2.7.3 การเชื่อมต่อสายของนัลโมเด็ม	36

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 2.7.4 รูปคลื่นของการส่งสัญญาณแบบอนุกรม
ด้วยระดับแรงดันของ TTL/CMOS 37
รูปที่ 2.7.5 รูปคลื่นของสัญญาณที่รับส่งผ่านพอร์ต RS-232 37
รูปที่ 3.1 บล็อกไดอะแกรมแสดงระบบควบคุมหุ่นยนต์แขนกล 38
รูปที่ 3.2 แสดงวงจรควบคุมมอเตอร์ 40
รูปที่ 3.3 แสดงทิศทางการหมุนของมอเตอร์ 40
รูปที่ 3.4 แสดงทิศทางการไหลของกระแสเมื่อ Q1 และ Q4 ได้รับไบอัส 41
รูปที่ 3.5 แสดงทิศทางการไหลของกระแสเมื่อ Q2 และ Q3 ได้รับไบอัส 41
รูปที่ 3.6 การควบคุมของวงจร 42
รูปที่ 3.7 การต่อไดโอดเพื่อป้องกันทรานซิสเตอร์เสียหาย 42
รูปที่ 3.8 แสดงวงจรวงจรควบคุมทิศทางการหมุนของมอเตอร์ 43
รูปที่ 3.9 วงจรป้องกันมอเตอร์จากความผิดพลาดในการทำงาน 44
รูปที่ 3.10 แสดงวงจรออปโต 45
รูปที่ 3.11 ไอซี ADC 0809 46
รูปที่ 3.12 วงจรทดลองไอซี ADC 0809 47
รูปที่ 3.13 วงจรกำเนิดความถี่ 500 กิโลเฮิร์ท 49
รูปที่ 3.14 วงจรกำเนิดความถี่ 3 กิโลเฮิร์ท 49
รูปที่ 3.15 วงจรอินพุทแบบอนาล็อกที่ใช้ในโครงงาน 50
รูปที่ 3.16 วงจร LIMIT SWITCH 51
รูปที่ 3.17 วงจร มือจับชิ้นงาน 53
รูปที่ 3.18 แสดงวงจร Connect Parallel 55
รูปที่ 3.19 การต่อใช้งานวงจรอินพุทอนาล็อก 57
รูปที่ 3.20 วงจร SW. Mode Hand 58
รูปที่ 3.21 วงจร Connect Parallel Port 59
รูปที่ 3.22 บล็อกไดอะแกรมการควบคุมหุ่นยนต์ 60
รูปที่ 3.23 ภาคไดร์ฟเวอร์ 62
รูปที่ 3.24 การต่อใช้งานวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล 64

สารบัญรูปภาพ (ต่อ)

	หน้า
รูปที่ 4.1 ส่วนประกอบของแขนกล	65
รูปที่ 4.2 ลักษณะการเคลื่อนไหวของชุดข้อบิดหมุนและชุดหัวไหล่	66
รูปที่ 4.3 ลักษณะการเคลื่อนไหวของชุดข้อศอกและชุดโครงฐาน	66
รูปที่ 4.4 การเคลื่อนไหวของส่วนพับข้อมือ	67
รูปที่ 4.5 การเคลื่อนไหวของส่วนบิดข้อมือ	67
รูปที่ 4.6 มือจับชิ้นงาน	68
รูปที่ 4.7 การประกอบของชุดมือจับชิ้นงานและชุดพับข้อมือ	69
รูปที่ 4.8 ส่วนประกอบของชุดบิดข้อมือ	70
รูปที่ 4.9 ส่วนประกอบของชุดพับข้อศอก	71
รูปที่ 4.10 ส่วนประกอบของชุดหัวไหล่แท่นหมุน	72
รูปที่ 4.11 ส่วนประกอบของชุดฐาน	73
รูปที่ 4.12 ลักษณะทางด้านหน้าของชุดแขนควบคุม	74
รูปที่ 4.13 ลักษณะทางด้านหลังของชุดแขนควบคุม	75
รูปที่ 4.14 ชุดควบคุมการเคลื่อนที่ของล้อหน้า (ลักษณะทางด้านหน้า)	76
รูปที่ 4.15 ชุดควบคุมการเคลื่อนที่ของล้อหน้า (ลักษณะทางด้านหลัง)	76
รูปที่ 4.16 ชุดควบคุมการเคลื่อนที่ของล้อหลัง	77
รูปที่ 5.1 โฟลว์ชาร์ตของโปรแกรมเมนูหลัก	78
รูปที่ 5.2 โปรแกรมเมนูหลัก	79
รูปที่ 5.3 โฟลว์ชาร์ตของโปรแกรมแบบอัตโนมัติ	80
รูปที่ 5.4 โปรแกรมการทำงานแบบอัตโนมัติ	81
รูปที่ 5.5 แสดงโฟลว์ชาร์ตการทำงานแบบ Manual	82
รูปที่ 5.6 โปรแกรมชุดควบคุมแบบ Manual	83
รูปที่ 5.7 โปรแกรมบันทึกข้อมูล	84

สารบัญตาราง

หน้า

ตารางที่ 2.4.1 หน่วยความจำชั่วคราวหลัก 25
ตารางที่ 2.4.2 หน่วยความจำชั่วคราวรอง 26
ตารางที่ 2.4.3 ตำแหน่งของหน่วยความจำถาวร 26
ตารางที่ 2.5 การกำหนดค่าของ Control Port 28
ตารางที่ 2.6 ตำแหน่งพอร์ต 32
ตารางที่ 2.7.1 ตำแหน่งขาสัญญาณของพอร์ตอนุกรม 35
ตารางที่ 2.7.2 หน้าที่ของขาสัญญาณแต่ละเส้น 35
ตารางที่ 2.7.3 ตำแหน่งมาตราฐานของพอร์ตอนุกรม 37
ตารางที่ 3.1 สัญญาณควบคุมการหมุน 44
ตารางที่ 3.2 Truth table ของวงจรป้องกันมอเตอร์ จากความผิดพลาดจากการทำงาน 45
ตารางที่ 3.3 ค่าโดยละเอียด 47
ตารางที่ 3.4 ค่าโดยประมาณ 48
ตารางที่ 3.5 แสดง CODE การรับช่องของสัญญาณ 50
ตารางที่ 3.6 ขาสัญญาณและการใช้งาน 59
ตารางที่ 3.7 แสดงการเชื่อมต่อระหว่าง CP-B2SX กับบอร์ด 8255 61
ตารางที่ 3.8 การใช้งานของมอเตอร์แต่ละตัว 63

บทที่ 1
บทนำ

ปัจจุบันนี้เทคโนโลยีหุ่นยนต์ได้พัฒนาขึ้นอย่างรวดเร็ว มีการใช้หุ่นยนต์มากขึ้นทั้งในงานอุตสาหกรรมและเริ่มเข้ามาเกี่ยวข้องกับชีวิตประจำวันมากขึ้น โดยหุ่นยนต์ส่วนใหญ่ที่ใช้ในงานอุตสาหกรรมจะเป็นหุ่นยนต์ที่ใช้ทำงานซ้ำๆ กัน ส่วนการทำงานที่ไม่แน่นอน หรือต้องปรับเปลี่ยนรูปแบบการทำงานบ่อย ๆ จะไม่นิยมใช้หุ่นยนต์เพราะต้องคอยเปลี่ยนโปรแกรมควบคุมหุ่นยนต์ตามรูปแบบการใช้งานที่เปลี่ยนไป จึงเป็นการไม่คุ้มค่าที่จะเปลี่ยนโปรแกรมเพื่อทำงานรูปแบบใดเพียงครั้งเดียวแล้วไม่ได้ทำงานนั้นอีกเลย

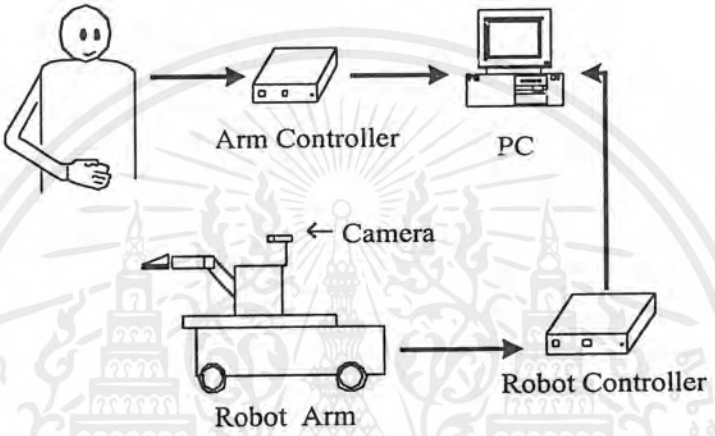
โครงงานนี้มุ่งเน้นเพื่อเพิ่มความสามารถให้กับหุ่นยนต์ เพื่อให้มีความยืดหยุ่นในการทำงาน ทำให้ใช้ประโยชน์ได้อย่างคุ้มค่า หุ่นยนต์จำนวนมากในงานอุตสาหกรรมมักจะถูกออกแบบให้ทำงานเฉพาะอย่างมีวงรอบการทำงานที่ตายตัว โดยใช้ระบบการควบคุมขนาดเล็กแต่เมื่อพิจารณาถึงความยืดหยุ่นหากมีความจำเป็นต้องปรับเปลี่ยนโปรแกรมควบคุมแล้ว จะต้องแก้ไขโดยผู้ที่มีความชำนาญทั้งด้านซอฟแวร์และฮาร์ดแวร์เท่านั้น

เนื่องจากในปัจจุบันขีดความสามารถคอมพิวเตอร์ส่วนบุคคลมีหน่วยประมวลผลที่มีการพัฒนาอย่างต่อเนื่อง ดังนั้นจึงเป็นแนวความคิดที่จะใช้ข้อดีดังกล่าวในการออกแบบโปรแกรมที่ซับซ้อน เพื่อควบคุมการทำงานของหุ่นยนต์ โดยโครงงานนี้ได้ใช้ภาษาระดับวิชวลเบสิก (Visual Basic) ในการเขียนโปรแกรมควบคุมโดยใช้เครื่องคอมพิวเตอร์ส่วนบุคคลเป็นตัวประมวลผลหลักทำงานร่วมกับไมโครคอนโทรลเลอร์ นอกจากนี้ผู้ควบคุมยังสามารถดูสภาวะการทำงานได้ทางมอนิเตอร์ ทำให้มีความคล่องตัวในการทำงานและเมื่อต้องการปรับเปลี่ยนรูปแบบการทำงานของหุ่นยนต์ ผู้ปฏิบัติงานสามารถกำหนดรูปแบบการทำงานได้เองถึงแม้ว่าผู้ปฏิบัติจะไม่มีความรู้ด้านโปรแกรมเลยก็ตาม นอกจากความสามารถในการเรียนรู้และจดจำขั้นตอนการทำงานของหุ่นยนต์แล้ว หุ่นยนต์ยังสามารถเคลื่อนที่เพื่อทำงานในตำแหน่งต่าง ๆ ได้ และเพื่อเป็นแนวทางในการศึกษาพัฒนาสำหรับผู้ที่สนใจต่อไป

ส่วนประกอบของโครงงาน

ลักษณะโดยทั่วไปของโครงงานการควบคุมหุ่นยนต์แขนกลแบ่งเป็น 4 ส่วนคือ

1. ส่วนบังคับควบคุม
2. ส่วนประมวลผล
3. ส่วนแขนกล
4. ส่วนควบคุมทิศทางการเคลื่อนที่



รูปที่ 1.1 ลักษณะการทำงานโดยรวม

1.1 ส่วนบังคับควบคุมแขนกล

ส่วนบังคับควบคุมจะเป็นชุดตรวจสอบการเคลื่อนไหวช่วงแขนของผู้ปฏิบัติงานแล้วจึงส่งสัญญาณจากการตรวจสอบเข้าส่วนประมวลผล ซึ่งมีทั้งหมด 6 ชุดดังนี้

1. ชุดตรวจสอบการพับข้อมือ
2. ชุดตรวจสอบการบิดข้อมือ
3. ชุดตรวจสอบการพับข้อศอก
4. ชุดตรวจสอบการยกหัวไหล่
5. ชุดตรวจสอบการหันหัวไหล่
6. ชุดตรวจสอบการจับชิ้นงาน

1.2 ส่วนประมวลผล

ส่วนประมวลผลจะใช้บอร์ดไมโครโปรเซสเซอร์ รุ่น CP-BS2SX เป็นตัวประมวลผลโดยการทำงานร่วมกับเครื่องคอมพิวเตอร์(PC)เพื่อควบคุมตำแหน่งของแขนกลให้สอดคล้องกับชุดควบคุม

1.3 ส่วนหุ่นยนต์แขนกล

ประกอบด้วยส่วนต่าง ๆ ดังนี้

1. ส่วนพับข้อมือ
2. ส่วนบิดข้อมือ
3. ส่วนพับข้อศอก
4. ส่วนยกหัวไหล่
5. ส่วนหันหัวไหล่
6. มือจับชิ้นงาน

1.4 ส่วนควบคุมทิศทางการเคลื่อนที่

เป็นส่วนควบคุมให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งใด ๆ ได้ โดยที่ผู้ปฏิบัติงานสามารถควบคุมผ่านทางจอภาพเพื่อเพิ่มความคล่องตัวในการปฏิบัติงาน ประกอบด้วย 2 ชุดคือ

1.ชุดขับเคลื่อนล้อ

2.กล้องตรวจสอบเส้นทาง

บทที่2
ทฤษฎีเบื้องต้น

2.1 มอเตอร์ไฟฟ้ากระแสตรง (DC MOTOR)

มอเตอร์ไฟฟ้ากับเครื่องกำเนิดไฟฟ้าต่างกันที่พลังงานป้อนเข้า (Input Power) พลังงานที่ป้อนเข้ามอเตอร์ เป็นพลังงานไฟฟ้าและพลังงานที่จ่ายออก (Output Power) เป็นพลังงานกล แต่พลังงานที่ป้อนเข้าเครื่องกำเนิดไฟฟ้าเป็นพลังงานกลและพลังงานที่จ่ายออกเป็นพลังงานไฟฟ้า นอกจากนี้ทิศทางของกระแสไฟฟ้าก็ต่างกันด้วยคือ มอเตอร์รับกระแสไฟฟ้าจากภายนอก ส่วนเครื่องกำเนิดไฟฟ้าจ่ายกระไฟฟ้าออกภายนอก

2.1.1 โครงสร้างของมอเตอร์ไฟฟ้ากระแสตรง

ส่วนประกอบที่สำคัญของมอเตอร์ไฟฟ้ากระแสตรงและเครื่องกำเนิดไฟฟ้ากระแสตรงนั้นเหมือนกันสามารถแสดงได้ดังรูป 2.1.1 โดยมีส่วนประกอบต่างๆ ดังนี้

ส่วนที่หยุดอยู่กับที่ (Stator) ประกอบด้วย

ก. เปลือกนอก (Frame หรือ Yoke) เป็นตัวยึดขั้วแม่เหล็กของส่วนที่อยู่กับที่พร้อมทั้งทำให้เส้นแรงแม่เหล็กที่เกิดจากขั้วแม่เหล็กวิ่งได้จนครบวงจร นอกจากนี้เปลือกนอกยังทำหน้าที่เป็นตัวยึดสำหรับติดตั้งเครื่องจักรรวมถึงเป็นแป้นยึดลูกปืน (Bearing) สำหรับเพลาของตัวหมุน การทำเปลือกนอกทำได้โดยการขึ้นรูป การใช้เหล็กหล่อลื่น หรือใช้วิธีม้วนเหล็กแผ่นแล้วเชื่อมเป็นวง

ข. แกนเหล็กของขั้วแม่เหล็กและขั้วแม่เหล็ก (Pole-core and Pole-shore) ทั้งสองส่วนนี้ประกอบกันขึ้นมาเพื่อทำหน้าที่เป็นแกนและขั้วแม่เหล็ก เพื่อทำให้เกิดสนามแม่เหล็กขึ้นหลังจากที่มีกระแสไฟไหลผ่านขดลวดตัวนำที่พันรอบแกนเหล็ก

ค. ขดลวดที่พันรอบแกนเหล็ก (Pole Coils) ก็คือField Coil ที่ประกอบด้วยลวดกลมหรือลวดแบนที่ทำด้วยลวดทองแดงหลายๆ เส้นนำมาทำให้เป็นรูปร่างของลวดตัวนำที่ต้องการ แล้วนำขดลวดตัวนำนี้สวมทับลงไปบนแกนเหล็กของขั้วแม่เหล็ก ดังนั้นเมื่อมีกระแสไหลผ่านขดลวดนี้ ก็ทำให้ขั้วแม่เหล็กมีอำนาจเป็นแม่เหล็กขึ้นมาโดยการผลิตเส้นแรงแม่เหล็กให้เกิดขึ้น เส้นแรงแม่เหล็กนี้ถูกตัดผ่านโดยตัวนำภายในอาร์เมเจอร์อีกทีหนึ่ง

ส่วนที่เคลื่อนที่ที่หมุนได้รอบตัว (Rotor) ประกอบด้วย

ก. แกนเหล็กของอาร์เมเจอร์ (Armature Core) ทำจากแผ่นเหล็กซิลิคอน หนาประมาณ 0.5 มิลลิเมตร ผิวทั้ง 2 ข้างจะฉาบด้วยฉนวน แล้วนามาอัดซ้อนเป็นรูปทรงกระบอกเพื่อลดการสูญเสียเนื่องจากฮีสเตอรีซีสและกระแสไหลวนในแกนเหล็ก ผิวด้านนอกของทรงกระบอกจะทำเป็นร่อง (Slot) เรียงตามแนวเส้นรอบวงรอบนอกของแกนเหล็กเพื่อใช้พันขดลวดอาร์เมเจอร์ ส่วนตรงกลางจะเจาะรูเป็นวงกลมเพื่อเอาไว้ใส่แกนเหล็ก (Shaft) แล้วก็บากเป็นช่องสี่เหลี่ยมของรูปที่เจาะนั้นร่องหนึ่งเพื่อใส่ ตัวยึด (Lock) หรือกุญแจ (Key) ทั้งนี้เพื่อไม่ให้เกิดการเคลื่อนที่ขึ้นระหว่างตัวอาร์เมเจอร์กับแกนเหล็ก นอกจากนั้นยังเจาะรูอากาศ (Air hole) เล็กๆ ทะลุผ่านอาร์เมเจอร์เพื่อระบายความร้อน

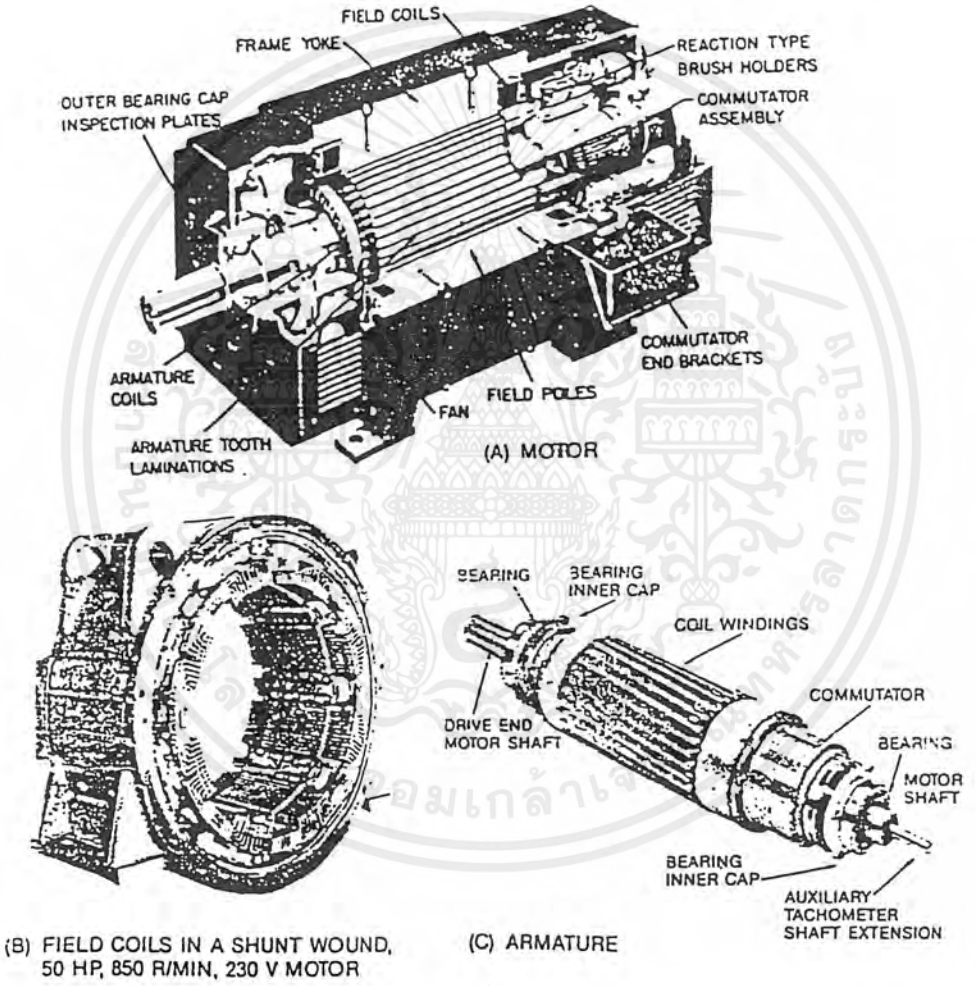
ข. ขดลวดอาร์เมเจอร์ (Armature winding) ก็คือขดลวดที่พันอยู่ในร่องของอาร์เมเจอร์

ค. คอมมิวเตเตอร์ (Commentator) มีหน้าที่คือ เป็นตัวที่เพิ่มความสะดวกในการนำกระแสออกมาจากตัวนำที่พันอยู่ในอาร์เมเจอร์ และเป็นตัวกลับกระแสไฟสลับที่เกิดขึ้นภายในอาร์เมเจอร์ให้เป็นกระแสไฟตรง หรือกระแสที่ไหลไปยังวงจรภายนอกในทิศทางเดียวกัน รูปร่างของมันเป็นรูปทรงกระบอกที่ประกอบด้วยซี่ทองแดงหลายซี่มาต่อรวมกันเป็นรูปทรงกระบอก ซี่ต่อซี่ที่ต่อชิดกันนั้นคั้นไว้ด้วยฉนวนหนาที่แข็งแรง แต่ละซี่ต่อไปยังขั้วต่อของคอมมิวเตเตอร์ เพื่อให้ตัวนำที่อยู่ในอาร์เมเจอร์ยึดเกาะได้มั่นคงอีกทีหนึ่ง การป้องกันไม่ให้ซี่คอมมิวเตเตอร์ต่างๆ เหล่านี้ต้องกระเด็นหลุดไปอันเนื่องมาจากแรงหนีศูนย์กลาง จึงใช้ห่วงวงแหวนที่ทำด้วยไมก้ายึดซี่คอมมิวเตเตอร์ทั้งหมด

ง. แกนหมุน (Shaft) เป็นตัวรองรับน้ำหนักในส่วนต่างๆ ของโรเตอร์ทั้งหมด โดยถ่ายทอดน้ำหนักไปยังลูกปืน ที่รองรับแกนนี้อีกทีหนึ่ง และมีหน้าที่เป็นตัวรับหรือถ่ายทอดการหมุนหรือการเคลื่อนที่ต่างๆ ที่เกิดขึ้นกับโรเตอร์ แกนหมุนนี้เป็นที่ยึดเกาะของคอมมิวเตเตอร์ด้วย โดยมีฉนวนที่ทำด้วยไมก้าคั้นไว้ระหว่างคอมมิวเตเตอร์กับแกนหมุน

จ. แปรงถ่านและลูกปืน (Brushes and Bearing) แท่งถ่านมีหน้าที่เป็นตัวรวบรวมกระแสที่ได้จากคอมมิวเตเตอร์เพื่อส่งไปยังวงจรภายนอก รูปร่างของแปรงถ่านเป็นแท่งสี่เหลี่ยมผืนผ้า ซึ่งทำด้วยผงถ่านอัดแน่นเป็นก้อน แท่งถ่านเหล่านี้ถูกยึดติดอยู่กับที่จับแท่งถ่าน (Brush Holder) ซึ่งเป็นกล่องสำหรับใส่แท่งถ่านลงไป ที่ยึดนี้จะยึดติดกับเปลือกนอกอีกที ฉะนั้นหน้าสัมผัสของแท่งถ่านด้านหนึ่งก็จะสัมผัสกับซี่คอมมิวเตเตอร์ ส่วนด้านตรงข้ามก็ถูกกดจากสปริงอีกทีหนึ่ง ตรงด้านที่ถูกกดของแปรงถ่านต่อเข้ากับเส้นลวดทองแดงเล็กๆ ที่ถักเป็นเปีย เพื่อส่งต่อกระแสที่ได้จากแท่งถ่านไปยังวงจรภายนอก จำนวนแท่งถ่านที่ใช้นี้จะมากหรือน้อยขึ้นอยู่กับว่า กระแสที่ได้รับจาก

คอมมิวเตเตอร์นั้นมีมากหรือน้อยเพียงใด ส่วนลูกปืนนั้นเป็นตัวที่ใช้สำหรับรับน้ำหนักทั้งหมดที่ได้รับจากตัวหมุน และยังช่วยลดแรงเสียดทานที่แกนหมุนของตัวหมุนกระทำกับลูกปืนนั้น ปกติแล้วลูกปืนนี้จะยึดติดอยู่กับฝาครอบทั้งสองด้าน ที่จะต้องยึดติดกับเปลือกนอกของเครื่องกำเนิดไปอีกทีหนึ่ง



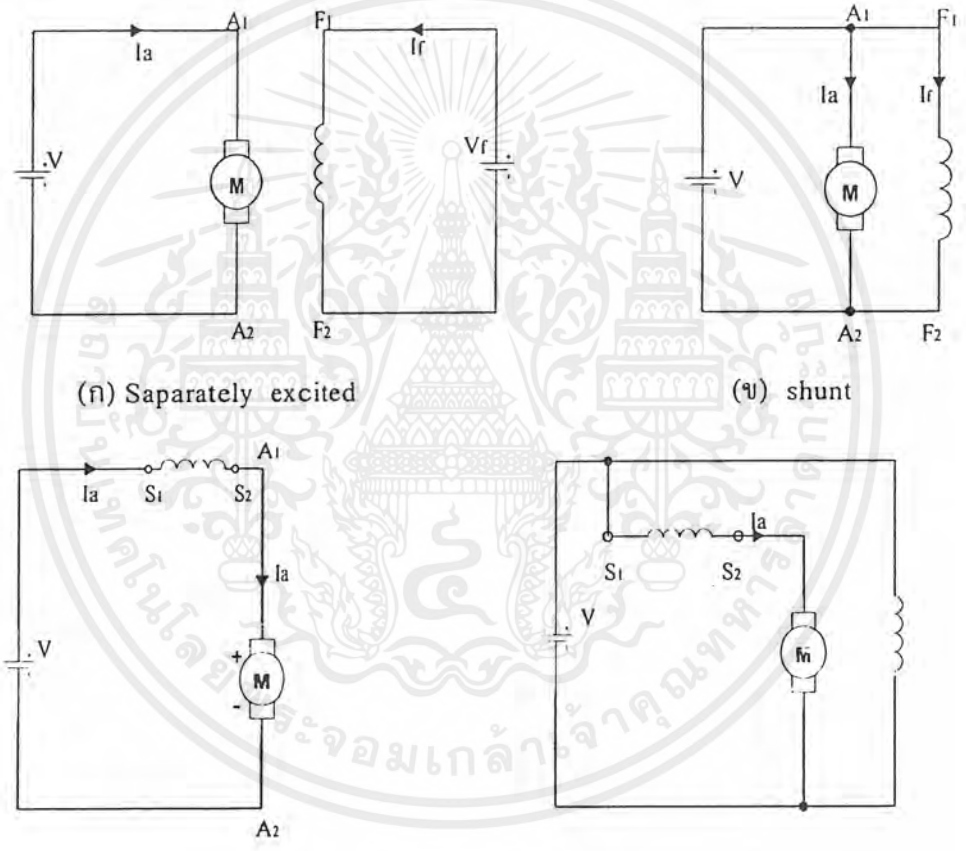
รูปที่ 2.1.1 ส่วนประกอบต่างๆ ของมอเตอร์ไฟฟ้ากระแสตรง

2.1.2 ชนิดของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรง แบ่งเป็น 4 ชนิดดังนี้

1. มอเตอร์ไฟฟ้ากระแสตรงแบบกระตุ้นแยก (Separately Excited DC Motor)
2. มอเตอร์ไฟฟ้ากระแสตรงแบบขนาน (Shunt DC Motor)
3. มอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรม (Series DC Motor)
4. มอเตอร์ไฟฟ้ากระแสตรงแบบผสม (Commulative Compound DC Motor)

ซึ่งมีวงจรสมมูลดังรูปที่ 2.1.2



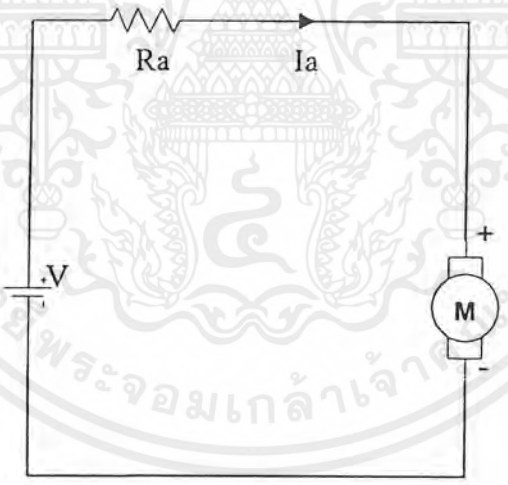
รูปที่ 2.1.2 ชนิดของมอเตอร์ไฟฟ้ากระแสตรง

ในกรณีของมอเตอร์ไฟฟ้ากระแสตรงแบบกระตุ้นแยกการควบคุมศักดาที่ตกคร่อม อาเมเจอร์และขดสนามไฟฟ้า (Field) นั้นแยกอิสระต่อกัน ส่วนมอเตอร์ไฟฟ้ากระแสตรงแบบขนาน ขดลวดสนามและอาเมเจอร์ต่อกับแหล่งจ่ายไฟเดียวกัน การที่ควบคุมแยกกันทำโดยวิธีเดียวคือการเพิ่มความต้านทาน (R) ภายในวงจร แต่เป็นการควบคุมที่ไม่มีประสิทธิภาพ

ในกรณีของมอเตอร์ไฟฟ้ากระแสตรงแบบอนุกรมกระแสที่ไหลผ่านขดสนาม (Field flux) จึงขึ้นอยู่กับกระแสอาเมเจอร์ด้วย ส่วนมอเตอร์ไฟฟ้ากระแสตรงแบบผสม แรงเคลื่อนแม่เหล็ก (Magnetomotive force: mmf) ของขดสนามที่ต่ออนุกรมกับมอเตอร์อยู่ มีผลต่อกระแสอาเมเจอร์ด้วยและมีทิศทางเดียวกับแรงเคลื่อนแม่เหล็กของขดสนามที่ต่ออนุกรมกับมอเตอร์

2.1.3 ความสัมพันธ์ระหว่างแรงบิดกับความเร็วที่สถานะคงที่

(Steady state speed torque relation)



รูปที่ 2.1.3 วงจรสมมูลที่สถานะคงตัวของอาร์เมเจอร์

จากรูปที่ 2.1.3 ความต้านทาน Ra คือความต้านทานของวงจรอาร์เมเจอร์ สำหรับมอเตอร์ไฟฟ้ากระแสตรงแบบกระตุ้นแยก คือ ความต้านทานของขดลวดอาร์เมเจอร์

สมการพื้นฐานของมอเตอร์ไฟฟ้ากระแสตรงมีดังนี้

$$E_g = K_a \phi N \qquad (2.1.1)$$

$$E_a = E_g + I_a R_a \qquad (2.1.2)$$

$$T = K_a \phi I_a \qquad (2.1.3)$$

ซึ่ง ϕ คือ จำนวนเส้นแรงแม่เหล็ก (Flux per pole), Webers

I_a คือ กระแสอาร์เมเจอร์ (Armature Current), A

E_a คือ ศักดาไฟฟ้าตกคร่อมอาร์เมเจอร์ (Armature Volage), V

R_a คือ ความต้านทานของขดลวดอาร์เมเจอร์ (Resistance of the armature circuit), Ω

N คือ ความของอาร์เมเจอร์ (Speed of armature)

T คือ แรงบิดที่มอเตอร์สร้าง (Torque developed the motor), rpm

K_a คือ ค่าคงที่ (Constant)

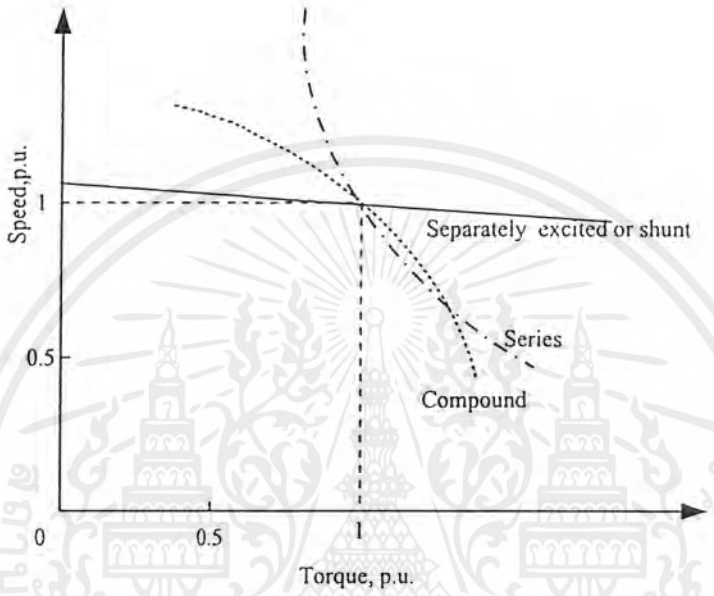
จากสมาการ (2.1.1) ถึง (2.1.3) จะได้

$$N = \frac{E_a}{K_a \phi} - \frac{R_a}{K_a \phi} I_a \qquad (2.1.4)$$

ในกรณีของมอเตอร์ไฟฟ้ากระแสตรงแบบกระตุ้นแยก ถ้าศักดาไฟฟ้าของขดสนาม คงที่ เราสามารถสมมุติให้เส้นแรงแม่เหล็กคงที่ ขณะที่แรงบิดเปลี่ยนแปลง ซึ่งจะได้ว่า

$$K_a \phi = K \text{ (Constant)} \qquad (2.1.5)$$

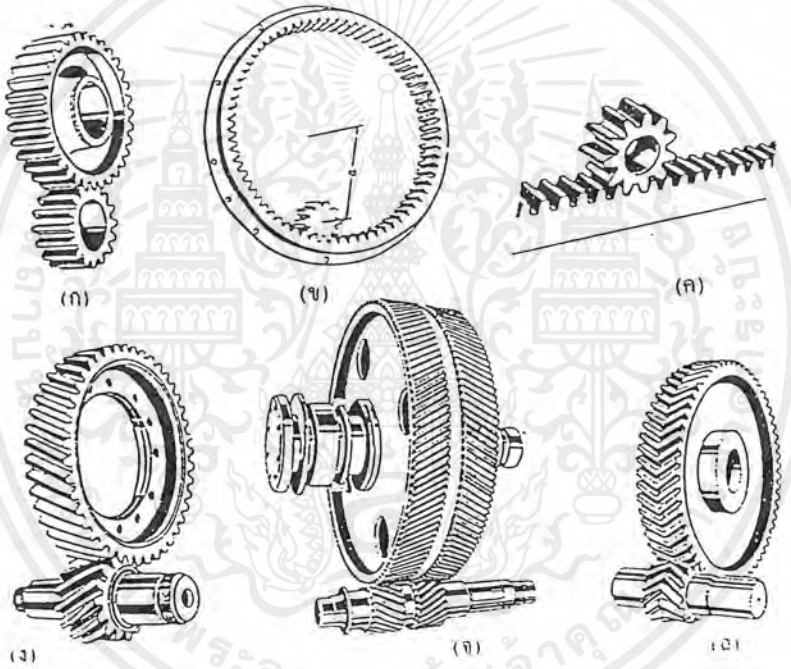
ดังนั้นจากสมการจะได้กราฟแสดงคุณลักษณะของแรงบิดและความเร็ว (Speed Torque Characteristic) ของมอเตอร์ไฟฟ้ากระแสตรงแบบกระตุ้นแยกเป็นเส้นตรง ดังรูปที่ 2.1.4



รูปที่ 2.1.4 กราฟแสดงความสัมพันธ์ระหว่างแรงบิดและความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

2.2 เฟือง (Gears)

เฟืองมีหลายประเภทแบ่งตามลักษณะของฟัน (Tooth) และตามลักษณะการวางตัวของเพลา (Shaft) ที่เฟืองสวมอยู่ เช่นฟันอาจจะตรงหรือโค้ง เพลาอาจจะตัดกันเป็นมุมฉากหรือขนานกัน

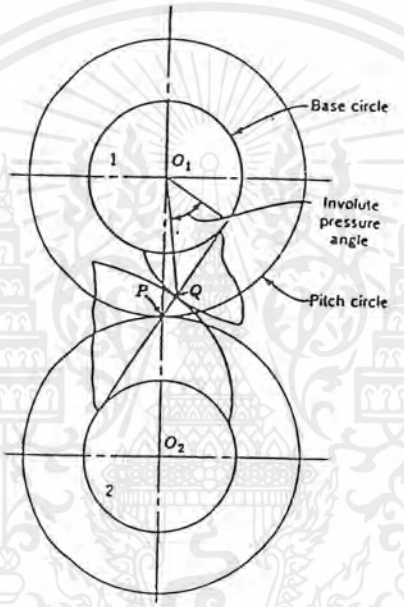


(ก) เฟืองธรรมดา (ข) เฟืองตรงและเฟืองวงแหวน (ค) เฟืองสะพาน
(ง) เฟืองตรงฟันเฉียง (จ) เฟืองตรงฟันเฉียงคู่ (ฉ) เฟืองตรงฟันเฉียงก้างปลา

รูปที่ 2.2.1 เฟืองชนิดต่าง ๆ

2.2.1 เฟืองตรงธรรมดา (Plain Spur Gear)

เฟืองตรงจะถูกนำมาใช้ในการส่งถ่ายโมเมนต์หมุนของเพลาไปยังอีกเพลาที่ขนานกัน ใช้ในงานที่ความเร็วรอบไม่เกิน 20 m/s และที่ความเร็วรอบปานกลาง นิยมใช้ในงานในกระปุกเกียร์แบบคันโยก ข้อดีของเฟืองตรงเมื่อเปรียบเทียบกับเฟืองตรงฟันเฉียง ก็คือ จะมีประสิทธิภาพดีกว่าและมีการสึกหรอน้อยกว่า เพราะว่าพื้นที่ผิวสัมผัสของฟันคู่ขบน้อยกว่า ส่วนข้อเสียเมื่อเปรียบเทียบกับเฟืองตรงฟันเฉียง คือ จะมีเสียงดังมากกว่า เมื่อทำงานที่ความเร็วรอบสูงๆ จึงมีความไวต่อการผิดพลาดของรูปร่างฟันมากกว่า มีอายุการใช้งานสั้น



รูปที่ 2.2.2 เฟืองตรงธรรมดา

ถ้า D_1 และ D_1 เป็นเส้นผ่าศูนย์กลางของ เฟือง (Pitch Circle) ตัวที่ 1 และสองตามลำดับแล้ว

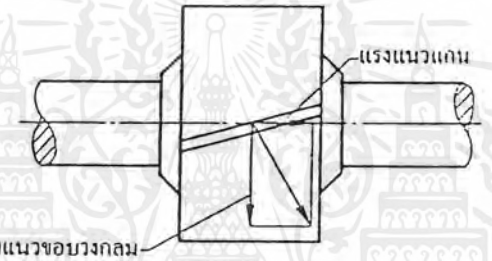
$$\omega_1 / \omega_2 = D_1 / D_2 \qquad (2.2.1)$$

และถ้าให้ N_1 แล ะ N_1 เป็นจำนวนฟันของเฟืองตัวที่ 1 และตัวที่ 2 ตามลำดับแล้ว

$$\omega_1 / \omega_2 = D_1 / D_2 = N_1 / N_2 \qquad (2.2.2)$$

2.2.1.1 เฟืองตรงฟันเฉียง (Helical Spur Gears)

เป็นเฟืองที่มีการขบของฟันหลาย ๆ ฟันในขณะเดียวกัน เนื่องจากหลายฟันไม่สามารถขบกันเต็มหน้าความกว้างในเวลาเดียวกันได้ จึงทำให้เฟืองตรงฟันเฉียงส่งถ่ายกำลังได้เงียบกว่ากันตรงธรรมดา และส่งถ่ายโมเมนต์หมุนได้มากกว่าด้วย ฟันที่เฉียงนี้จะทำให้เกิดแรงตามแนวแกนที่รองเพลาจะต้องเป็นตัวรับแรงไว้เสมอ ซึ่งจะทำให้ประสิทธิภาพลดลง เพื่อไม่ให้แรงตามแนวแกนนี้มากเกินไป จึงมีการกำหนดความเอียงของฟัน อยู่ระหว่าง 8 ถึง 20 องศา เฟืองตรงฟันเฉียงยังเหมาะกับงานที่มีความเร็วรอบสูงกว่าอีกด้วย



รูปที่ 2.2.3 แรงตามแนวแกน (Axial Force) ที่เกิดบนเฟืองตรงฟันเฉียงขณะส่งถ่ายกำลัง

เฟืองตรงที่กล่าวมาเหมาะสำหรับใช้เป็นกระปุกเฟืองทด สำหรับอัตราทดต่ำจะใช้เฟืองคู่เดียวทดส่วนการทดรอบมาก ๆ จะต้องใช้เฟืองทดหลายตัว

2.2.1.2 ขนาดพิกัดของเฟือง

ระยะพิตช์ P คือระยะเส้นโค้งจากด้านหน้าของฟันเฟืองหนึ่งไปอีกฟันหนึ่ง (ด้านเดียวกัน) โดยวัดตรงบริเวณเส้นรอบวงกลมพิตช์ ดังรูปที่ 2.2.3 ฟันเฟือง 2 ตัวที่มีระยะพิตช์แตกต่างกัน จะไม่สามารถขบกันได้ ระยะพิตช์จึงเป็นค่าที่สำคัญมากของพิกัดขนาดฟันเฟือง

เส้นรอบวงกลมพิทช์ $U = \Pi * d$ หน่วย mm (2.2.3)

เมื่อ Z = จำนวนฟัน

ขนาดเส้นผ่าศูนย์กลางที่พิกัด d

$$d = z(p/\Pi) \quad (2.2.4)$$

ผลหาร จะได้ค่าโมดูล (Module) m หน่วย mm นั่นคือ

$$d = m * z \quad (2.2.5)$$

p = ระยะพิตช์

d = เส้นผ่านศูนย์กลางวงกลมพิตช์

d_a = เส้นผ่านศูนย์กลางยอดฟัน

d_f = เส้นผ่านศูนย์กลางของโคนฟัน

a = มุมขบ

h_a = ความสูงยอดฟัน

h_a = ความสูงยอดฟัน

h_f = ความสูงโคนฟัน

c = ระยะฟรี

ค่าโมดูล m จะถูกกำหนดเป็นมาตรฐานตามแถวโมดูล DIN 780 ค่าโมดูลยิ่งมากเท่าใด ค่าระยะพิตช์และรูปร่างฟันก็ยิ่งโตมากขึ้นด้วย

สำหรับค่าที่มีความสำคัญอื่นก็คือ อัตราทด i

เมื่อให้ n_1 = ความเร็วรอบล้อเฟืองขับ

n_1 = ความเร็วรอบล้อเฟืองตาม

z_1 = จำนวนฟันล้อเฟืองขับ

z_2 = จำนวนฟันล้อเฟืองตาม

เขียนเป็นสูตรที่สัมพันธ์กันได้ดังนี้

$$n_1 * z_1 = n_2 * z_2 \quad (2.2.6)$$

หรือ $n_1 / z_1 = n_2 / z_2 = I$ (2.2.7)

และ $z_1 = d_1 / m$; และ $z_2 = d_2 / m$ (2.2.8)

จะได้ $i = d_2 / d_1$ (2.2.9)

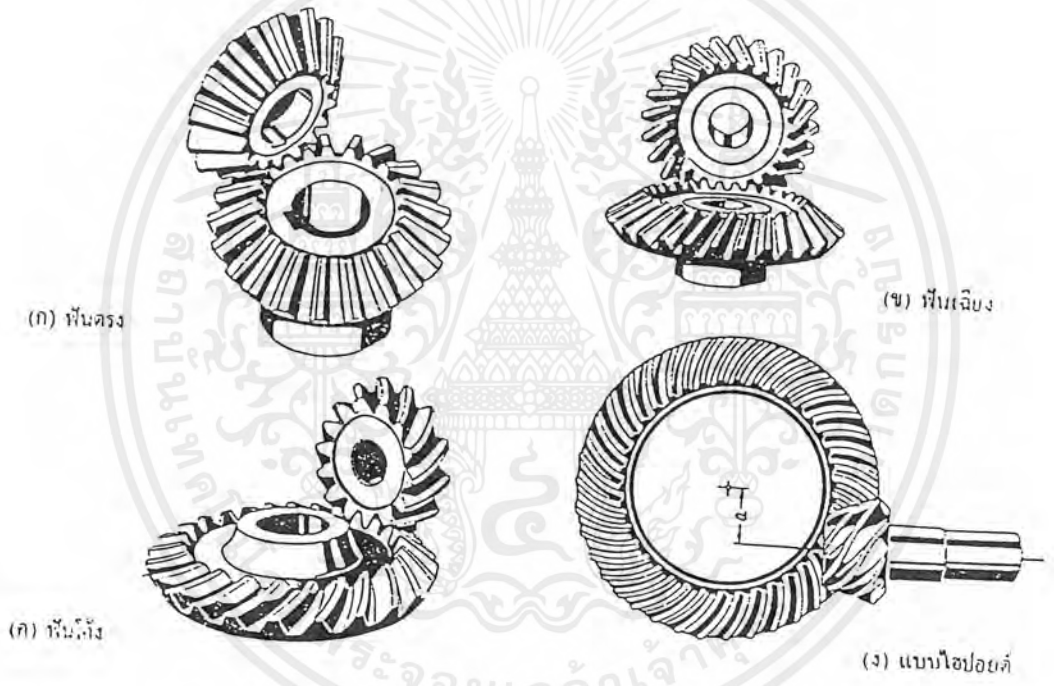
2.2.1.3 เฟืองดอกจอก (Bevel Gear)

เฟืองดอกจอกจะมีรูปร่างเป็นเรียวที่กลิ้งอยู่ด้วยกัน ฟันของเฟืองจะเรียวไปในทิศทางกึ่งกลางของเพลา โดยทั่วไปจะมีการผลิตเฟืองดอกจอกเป็นแบบฟันตรง ฟันตรงเอียง และฟันเอียงโค้ง

แรงที่เกิดจากฟันคู่ขบจะทำให้เกิดแรงตามแนวแกน จึงต้องมีรองเพลาชนิดที่รับแรงในแนวแกนไว้

ที่อัตราทดเท่ากันเฟืองดอกจอกจะเปลืองเนื้อที่น้อยกว่าเฟืองชุดตรง

ชุดเฟืองดอกจอกจะให้อัตราทดสูงสุด $i_{max} = 6$



รูปที่ 2.2.4 เฟืองดอกจอก

เฟืองดอกจอกฟันตรงจะนำมาใช้ในงานที่มีความเร็วต่ำ เช่น ชุดอุปกรณ์ถ่ายกำลังงานต่ำที่ควบคุมด้วยมือ

เฟืองดอกจอกฟันตรงเอียงจะมีเสียงดังในขณะส่งถ่ายกำลังน้อยกว่าแบบฟันตรงธรรมดา จึงนำมาใช้ในงานที่มีความเร็วรอบและกำลังงานสูงกว่า เช่น ชุดเฟืองขับในเครื่องมือกล

เฟืองดอกจอกฟันเอียงโค้ง จะนำมาใช้ในงานที่ต้องการความเงียบ เช่น ชุดเฟืองทดกำลังสูง ๆ ชุดเฟืองดิฟเฟอร์เรนเชียลของยานยนต์ เป็นต้น

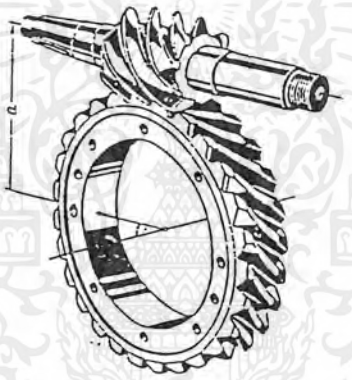
2.2.1.4 เฟืองเกลียวสกรู (Spiral Gear)

เฟืองเกลียวสกรูจะนำมาใช้ในการส่งถ่ายโมเมนต์หมุนระหว่างเพลาที่มีแนวตัดเป็นมุมต่างๆ กัน ตามที่ต้องการได้ ชุดเฟืองนี้จะส่งถ่ายกำลังได้น้อย เนื่องมาจากด้านข้างของฟันที่สัมผัสกันน้อยมาก สามารถให้อัตราทดได้ระหว่าง i = 1...5

รูปที่ 2.2.5 เฟืองเกลียวสกรู

2.2.1. 5 เฟืองหนอน (Worn Gear)

เฟืองหนอนจะนำมาใช้ในงานเมื่อเพลาขับและเพลาตามทำมุมตัดกัน 90 องศา และต้องการอัตราทดสูงมาก สามารถให้อัตราทดได้ถึง i = 50 (กรณีพิเศษได้ถึง i = 120) เมื่อเพลาเฟืองหนอนหมุนไป 1 ฟัน เฟืองหนอนล้อตามจะหมุนไป 1 ฟันถัดไป ชุดเฟืองหนอนมีเสียงดังน้อยมากและสามารถส่งถ่ายกำลังได้สูง แต่ก็มีการสึกหรอสูงบริเวณด้านข้างของฟัน ข้อเสียอีกประการหนึ่งก็คือ มีแรงที่เกิดตามแนวแกนสูงมาก ทำให้เพลาจะต้องรับภาระนี้ทั้งหมด ด้วยเหตุผลดังกล่าวมานี้ ชุดเฟืองหนอนจะต้องมีระบบหล่อลื่นที่ดี เช่น แบบอ่างน้ำมัน เป็นต้น



รูปที่ 2.2.6 เฟืองหนอน

2.2.2 โซ่กำลัง (Cnain Drives)

โซ่สามารถส่งกำลังให้ได้โมเมนต์บิด (หมุน) สูงมาก โดยให้เป็นชุดส่งกำลังที่มีขนาดเล็กได้ เป็นลักษณะการส่งกำลังด้วยรูปร่างและที่รองเพลาจะรับภาระน้อยมาก ไม่มีการลื่นไถลในขณะส่งกำลัง ในขณะส่งกำลังจะต้องมีการหล่อลื่นที่เพียงพอ โซ่กำลังจะใช้ในงานที่รับภาระดึงมาก ๆ ในที่อุณหภูมิสูง โรงงานเคมี ไอน้ำมัน บริเวณที่มีความชื้นสูง เป็นที่ซึ่งสายพานไม่สามารถนำไปใช้ได้

ข้อดีเมื่อเปรียบเทียบกับเฟือง

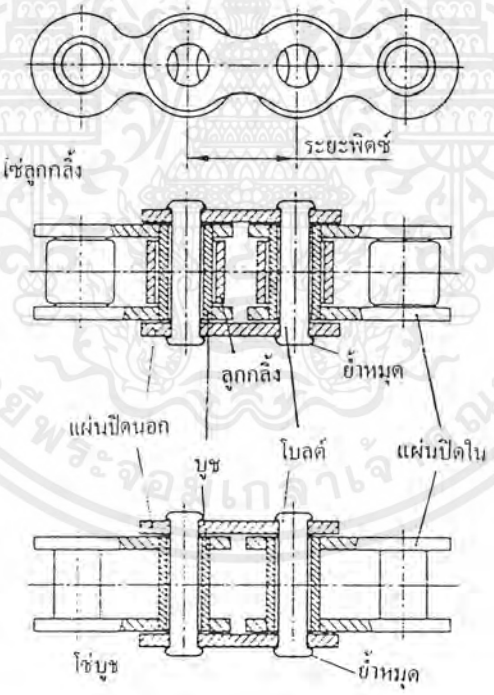
- แก้ปัญหาระยะห่างระหว่างเพลาที่ห่างกันมาก ๆ ได้
- มีความไวต่อสิ่งสกปรกน้อยกว่า

ข้อเสียเมื่อเปรียบเทียบกับเฟือง

- มีความเร็วรอบหรือมีความเร็วขอบน้อยกว่า (เนื่องจากแรงเหวี่ยงหนีศูนย์)
- ที่ความเร็วรอบสูงจะต้องใช้ตัวประกบกันการสั่นของโซ่
- เพลาต้องวางให้ขนานกันและส่วนใหญ่ต้องวางในแนวนอน

2.2.2.1 โซ่ลูกกลิ้งและโซ่บูช

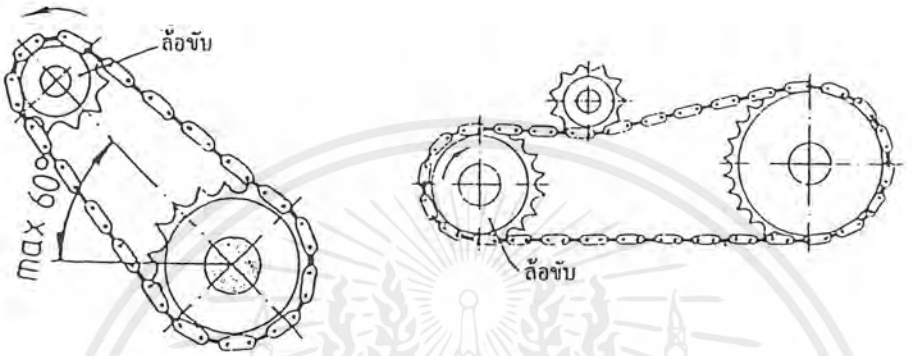
โซ่ลูกกลิ้งและโซ่บูชจะประกอบด้วยแผ่นปิดด้านข้างโซ่ด้านนอกและด้านในที่ยึดด้วยบูชและโบลท์ เข้าด้วยกัน โซ่ลูกกกลิ้งที่มีใช้งานส่วนใหญ่จะมีลูกกกลิ้งที่ชุบแข็งร้อยอยู่ในบูช ลูกกลิ้งนี้ช่วยลดความเสียดทานและการสึกหรอด้านข้างของเฟืองโซ่ในขณะที่ล้อเฟืองขับโซ่ มีเสียงดังน้อยเมื่อความเร็วโซ่สูง ในการใช้งานที่รับโมเมนท์หมุนมากๆ จะใช้โซ่ลูกกลิ้งและโซ่บูชแบบชุดหลายเส้น ดังรูป 2.2.7



รูปที่ 2.2.7 โซ่ลูกกลิ้งและบูช

2.2.2.2 ล้อโซ่ (Sprockets)

ล้อโซ่จะเป็นตัวขับเคลื่อนโซ่ จะทำจากเหล็กหล่อ เหล็กกล้าหล่อ หรือเหล็กกล้า ล้อโซ่จะมีขนาดเล็กและโตแตกต่างกันโดยจะสัมพันธ์กับภาระที่ใช้งาน ดังรูป 2.2.8



รูปที่ 2.2.8 ล้อโซ่

2.2.2.3 จำนวนฟันโซ่และความเร็วที่ใช้งาน

โดยปกติแล้วฟันล้อโซ่จะเป็นจำนวนเลขคี่ สำหรับงานส่งกำลังด้วยโซ่มีเกณฑ์กำหนดสำหรับล้อโซ่ดังนี้

ล้อโซ่ตัวเล็ก :

Z_1 = 9...11 ใช้กับความเร็วโซ่ (v) ต่ำกว่า 4 m/s

Z_1 = 11...13 ใช้กับความเร็วโซ่ (v) ถึง 4 m/s เป็นโซ่ที่มีระยะพิตช์ p ถึง 20 mm และมีความยาวโซ่เกินกว่า 40 ข้อ ใช้งานที่ไม่รับภาระมากนัก

Z_1 = 14...16 ใช้กับความเร็วโซ่ (v) ถึง 7 m/s และรับภาระปานกลาง

Z_1 = 17...25 ใช้กับความเร็วโซ่ (v) ถึง 24 m/s และรับภาระมาก

ล้อโซ่ตัวใหญ่ :

Z_1 = 30...80 มีใช้งานทั่วไป

Z_1 = 120 เป็นล้อโซ่ที่มีจำนวนฟันมากที่สุด

Z_1 = 150 ใช้งานในกรณีพิเศษ แต่มีอัตราการสึกหรอมาก ควรหลีกเลี่ยงการใช้งาน หากไม่จำเป็น

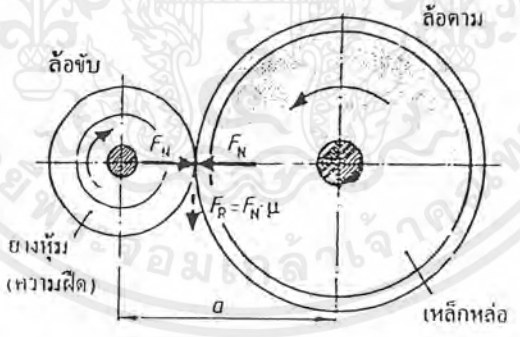
จำนวนฟันล้อโซ่มาตรฐานที่นิยมใช้งานมีดังนี้ (จำนวนฟันที่อยู่ในวงเล็บถ้าเป็นไปได้ควรหลีกเลี่ยง)

สำหรับล้อขนาดเล็ก (13) (15) 17 19 21 23 25

สำหรับล้อโซ่ขนาดโต 38 57 76 95 114

2.2.3 ล้อความฝืด

ล้อความฝืดใช้ทำหน้าที่ส่งถ่ายโมเมนต์แรงบิดด้วยความฝืด (ความเสียดทาน) ระหว่างเพลา 2 เพลาที่ขนานกันหรือเพลาที่ทำมุมตัดกันที่ระยะห่างกันตั้งแต่น้อยๆ ขึ้นไปได้ ล้อความฝืดมีความเหมาะสมสำหรับการส่งถ่ายที่ความเร็วขอบสูงได้ดี รวมทั้งการเปลี่ยนแปลงความเร็วรอบและการหมุนเปลี่ยนทิศทางได้อีกด้วย ล้อความฝืดจะหมุนได้เงียบปราศจากการกระแทรก แต่มีอัตราทดระหว่างล้อขับและล้อตามไม่คงที่ เนื่องมาจากในระหว่างการส่งถ่ายแรงขณะหมุนจะเกิดการลื่นไถลขึ้นได้



รูปที่ 2.2.9 ล้อความฝืด

2.2.3.1 การส่งถ่ายแรง

ที่ขอบล้อความฝืดจะมีแรงกด (F_N) (คือแรงปกติกระทำตั้งฉากกับแนวสัมผัส วงกลมที่จุดสัมผัส) บนแกนผิวเรียบเข้าหากัน ทำให้เกิดแรงบนล้อขับ (F_R)

$$F_R = F_N \quad \text{หน่วย N} \qquad (2.2.10)$$

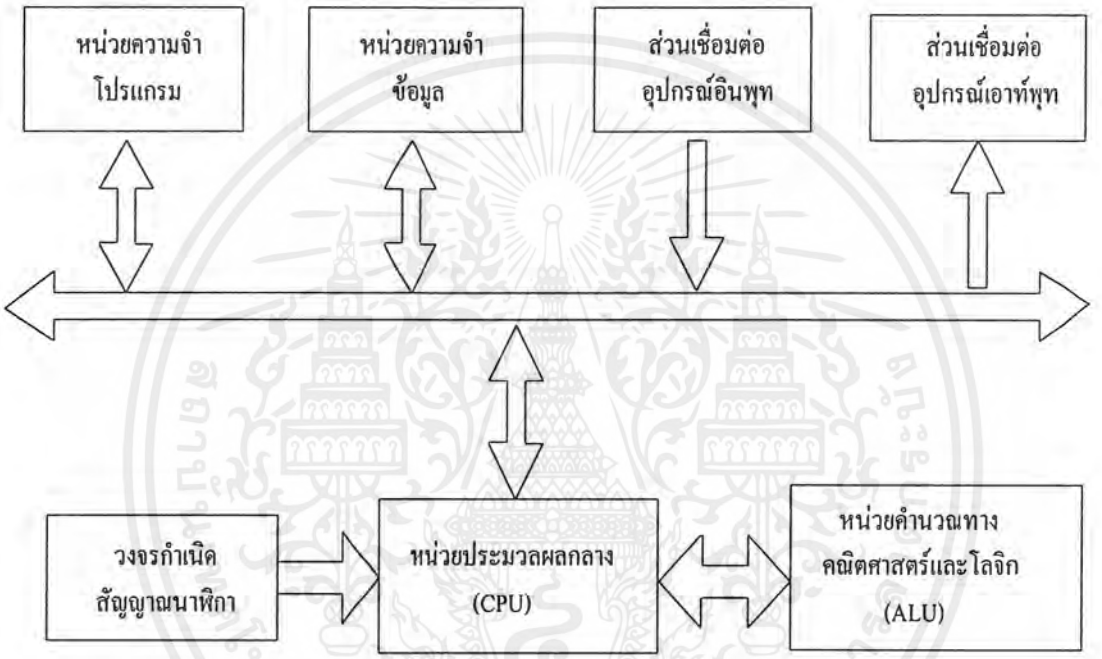
เพื่อให้ได้ความฝืดที่เหมาะสม จะมีการหุ้มขอบล้อความฝืดด้วยแผ่นพลาสติก ยางหรือหนัง ในการใช้งานรับภาระต่ำจะใช้แผ่นยางหุ้มยึดที่ขอบล้อ ข้อดีและข้อเสียของล้อความฝืดมีดังนี้

ข้อดี ให้การส่งกำลังแบบปรับความเร็วรอบอย่างอิสระได้ มีโครงสร้างแบบง่ายๆ มีระยะห่างของแกนน้อยกว่า ให้ประสิทธิภาพสูงสิ้นเปลืองค่าใช้จ่ายในการบำรุงรักษาน้อย มีเสียงดังน้อยมาก

ข้อเสีย มีการลื่นไถลขณะส่งกำลัง มีการสึกหรอบริเวณผิวเสียดทาน ส่งถ่ายพลังงานได้จำกัด มีความเร็วขอบที่จำกัด รองเพลาต้องรับภาระมากเนื่องจากแรงกดที่ล้อสูง

2.3 ทฤษฎีเกี่ยวกับไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) มาจากคำ 2 คำ คือ ไมโคร (Micro) หมายถึง ขนาดเล็ก และคำว่า คอนโทรลเลอร์ (controller) หมายถึงตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์ จึงหมายถึง อุปกรณ์ควบคุมขนาดเล็ก ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ที่รวมเอาซีพียู (CPU) หน่วยความจำ (Memory) และพอร์ต (Port) เข้าไว้ด้วยกัน โดยบรรจุรวมกันอยู่ภายในตัวถังเดียวกัน ในรูปที่ 2.3 แสดงส่วนประกอบหลักที่สำคัญของไมโครคอนโทรลเลอร์



รูปที่ 2.3 โครงสร้างการทำงานของไมโครคอนโทรลเลอร์

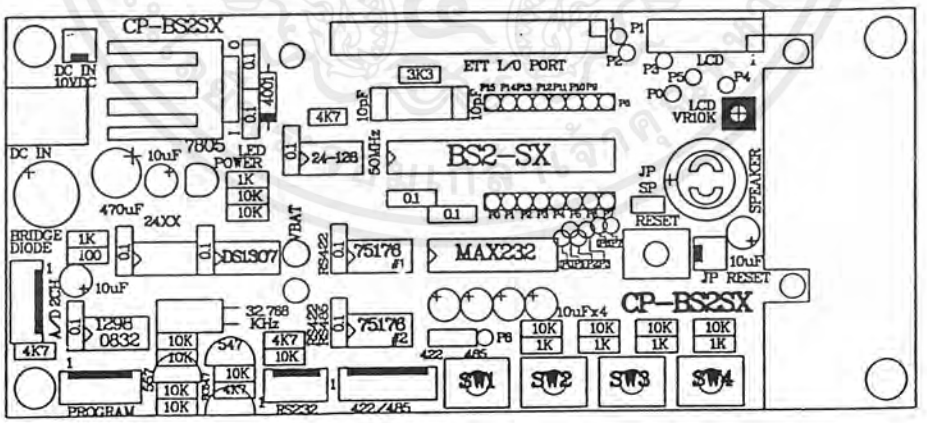
การทำงานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรเลอร์จะสามารถทำงานได้เมื่อจ่ายไฟเลี้ยงและต่อวงจรกำเนิดสัญญาณนาฬิกาให้ จากนั้นซีพียูภายในไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมเพื่ออ่านข้อมูลคำสั่งแล้วทำงานตามคำสั่งที่บรรจุอยู่ในหน่วยความจำโปรแกรม นั่นหมายความว่าต้องมีการเขียนข้อมูลลงในหน่วยความจำก่อน โดยไมโครคอนโทรลเลอร์แต่ละเบอร์จะมีรูปแบบของข้อมูลคำสั่งที่แตกต่างกัน ภาษาที่ใช้ในการเขียนโปรแกรมสามารถแบ่งได้ 2 ระดับคือ ภาษาสูง (High language) และภาษาแอสเซมบลี (Assembly language) โดยปกติไมโครคอนโทรลเลอร์ ต้องการโปรแกรมที่เขียนด้วยภาษาแอสเซมบลีเนื่องจากสามารถทำงานได้รวดเร็วผ่านกระบวนการแปลง

ข้อมูลคำสั่งเป็นข้อมูลเลขฐานสิบหก เพื่อทำงานเพียง 1 ขั้นตอนคือ แปลงจากภาษาแอสเซมบลีเป็นข้อมูลเลขฐานสิบหกที่เรียกว่า ออปโค้ด (Opcode) แต่ข้อเสียของการโปรแกรมด้วยภาษาแอสเซมบลีคือ ผู้เขียนต้องทำความเข้าใจในชุดของคำสั่งของไมโครคอนโทรลเลอร์เบอร์นั้นๆ อย่างถ่องแท้และเมื่อเปลี่ยนเบอร์ไมโครคอนโทรลเลอร์ก็ต้องทำความเข้าใจในชุดคำสั่งใหม่ ดังนั้นการเขียนโปรแกรมภาษาแอสเซมบลี ผู้เขียนต้องมีทักษะในการเขียนโปรแกรมสูงพอสมควร และเข้าใจถึงสถาปัตยกรรมของไมโครคอนโทลเลอร์เป็นอย่างดี

ในขณะที่ในการเขียนโปรแกรมด้วยภาษาสูง เช่น ภาษาซี ภาษาเบสิก ต้องผ่านกระบวนการที่เรียกว่า การคอมไพล์ (Compile) เพื่อแปลงภาษาระดับสูงเหล่านั้นเป็นภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์เบอร์นั้นๆ เสียก่อน เรียกโปรแกรมที่ใช้คอมไพล์นี้ว่า คอมไพล์เลอร์ (compiler) ซึ่งมักจะมีราคาแพงเมื่อใช้เครื่องมือทางซอฟแวร์ตัวนี้ ทำให้ผู้เขียนโปรแกรมอาจไม่จำเป็นต้องศึกษาสถาปัตยกรรมและชุดคำสั่งของไมโครคอนโทรลเลอร์เบอร์นั้นๆ อย่างลึกซึ้งเท่ากับการเขียนโปรแกรมด้วยภาษาแอสเซมบลี ทั้งนี้เพราะคอมไพเลอร์จะทำหน้าที่ส่วนนี้แทน ดังนั้นเมื่อผู้ใช้งานเปลี่ยนเบอร์ไมโครคอนโทรลเลอร์ก็เพียงแต่จัดหาโปรแกรมคอมไพล์เลอร์ที่เหมาะสมมาใช้งาน

2.4 บอร์ดไมโครคอนโทรลเลอร์ CP-BS2SX



รูปที่ 2.4.1 บอร์ด CP-BS2SX

2.4.1 คุณสมบัติของบอร์ดคอนโทรลเลอร์ CP-BS2SX

- ใช้ BS-2SX เป็นตัวประมวลผล 10,000 คำสั่งต่อวินาที ที่ 50 เมกกะเฮิร์ท
- อินพุทและเอาท์พุท 16 บิท พร้อมอินพุทแบบอนุกรม (Serial Port) 2 พอร์ทสำหรับการโปรแกรม
- ประกอบด้วยชุดคำสั่ง PBASIC 39 คำสั่ง
- หน่วยความจำข้อมูลภายใน 32 ไบต์และ Scratch Pad RAM ขนาด 64 ไบต์
- EEPROM Program 2 Kbytes x 8 program
- 12 Bit 2 channel ADC
- EEPROM 2K 2 wire interface
- 1 RS 232 Communication
- 1 RS 485 Communication

2.4.2 การต่อใช้งานของ CP-BS2SX

1. ให้ต่อสายจากเครื่องคอมพิวเตอร์ทางพอร์ทอนุกรม ที่คอนเน็คเตอร์ (Connector) 5 PIN JP PROGRAM
2. จ่ายไฟเลี้ยงวงจรให้กับบอร์ด 9-12 V กระแสตรง (DC) หรือ กระแสสลับ (AC)
3. เรียกโปรแกรม STAMP.EXE หรือ STAMP2SX.EXE ขึ้นมาทำงาน
4. เมื่อเขียนโปรแกรมเรียบร้อยแล้วให้ดาว์นโหลด (Download) โปรแกรมโดยแบ่งเป็น 2 กรณีคือ
   - 4.1 กรณีเรียกใช้โปรแกรม STAMP.EXE ให้กดปุ่ม F9 หรือเลือกที่เมนู RUN
   - 4.2 กรณีใช้โปรแกรม STAMP2SX.EXE ให้กดปุ่ม Alt+R
5. เมื่อดาว์นโหลดเสร็จแล้วโปรแกรมจะรัน (RUN) อัตโนมัติ โดยจะมีเสียงออกลำโพงอย่างต่อเนื่อง

2.4.3 การจัดการหน่วยความจำของ CP-BS2SX

ซีพียู BS-2SX มีการแบ่งหน่วยความจำออกเป็น 2 ส่วนหลักๆ คือ หน่วยความจำข้อมูล (RAM) สำหรับใช้งานทั่วไป และหน่วยความจำโปรแกรม (EEPROM) สำหรับเก็บโปรแกรมพีเบสิก (PBASIC) และเก็บข้อมูลทั่วๆ ไป โดยแรมจะแบ่งออกเป็นอีก 2 ส่วนคือ Variable RAM และ Scratch Pad RAM โดยมีการจัดการดังนี้

- VARIABLE RAM ทั้งหมดประกอบด้วย 16 เวิร์ด (Word) ซึ่งเวลาใช้งานสามารถกำหนดให้เป็น ไบต์ (Byte) นิบเบิล (Nibble) หรือบิท (Bit) ได้ด้วย โดย 3 เวิร์ดแรกถูกใช้สำหรับ I/O Registers ส่วนที่เหลือถูกใช้งานทั่วไป ดังตารางที่ 2.4.1

BASIC Stamp II sx Variable RAM (I/O) and variable space)				
Word Name	Byte Name	Nibble Names	Bit Names	Special Notes
INS	INL INH	INA,INB, INC,IND	IN0-IN7, IN8-IN15	Input pins
OUTS	OUTL OUTH	OUTA,OUTB, OUTC,OUTD	OUT0-OUT7, OUT8-OUT15	Output pins
DIRS	DIRL DIRH	DIRA,DIRB, DIRC,DIRD	DIR0-DIR7, DIR8-DIR15	I/O pin direction Control
W0	B0 B1			General Purpose
W1	B2 B3			General Purpose
W2	B4 B5			General Purpose
...(Continues W3 though W11 and B6 though B23).				
W12	B24 B25			General Purpose

ตารางที่ 2.4.1 หน่วยความจำชั่วคราวหลัก

- Scratch Pad RAM ประกอบด้วย 64 ไบต์ (0-63) สามารถเข้าถึงได้โดยใช้คำสั่ง GET และ PUT ซึ่งตำแหน่ง 0 ถึง 62 สามารถใช้งานได้ทั่วไปแต่ตำแหน่งที่ 63 สามารถอ่านได้อย่างเดียวซึ่งจะเก็บหมายเลข ID ของโปรแกรมที่รัน (RUN) ในขณะนั้น (0-7) เพราะว่า BS-2SX สามารถเขียนได้ 8 โปรแกรมหรือ 8 หน้า (Page) โดยแต่ละโปรแกรมมีขนาด 2 กิโลไบต์ ดังตารางที่ 2.4.2

BASIC Stamp 2sx Scratch Pad RAM	
RAM Location	General Purpose
0	General Purpose
1	General Purpose
...(Continues 2 though 61)...	
62	General Purpose
63	Contains ID of currently running program (0-7)

ตารางที่ 2.4.2 หน่วยความจำชั่วคราวรอง

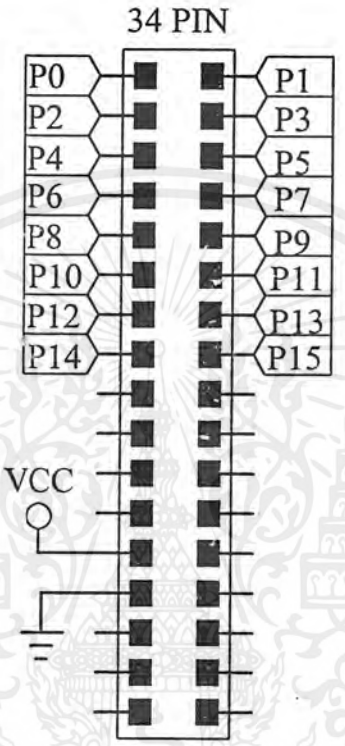
- EEPROM ประกอบด้วย 16 กิโลไบต์ สำหรับเก็บโปรแกรม PBASIC และข้อมูลทั่วไป โดยโปรแกรมๆละ 2 กิโลไบต์ โดยมีตำแหน่งต่างๆ ดังตารางที่ 2.4.3

Organization of BSX2-SX Programs	
EEPROM Locations	Contents
$0000-$07FF	Program 0 (logical locations $0000-$07FF)
$0800-$0FFF	Program 0 (logical locations $0000-$07FF)
$1000-$17FF	Program 0 (logical locations $0000-$07FF)
$2000-$1FFF	Program 0 (logical locations $0000-$07FF)
$2800-$27FF	Program 0 (logical locations $0000-$07FF)
$3000-$37FF	Program 0 (logical locations $0000-$07FF)
$3800-$3FFF	Program 0 (logical locations $0000-$07FF)

ตารางที่ 2.4.3 ตำแหน่งของหน่วยความจำถาวร

2.4.4 พอร์ต(Port)

CP-BS2SX จะมีพอร์ตใช้งานขนาด 16 บิทอยู่ 2 พอร์ต ซึ่งพอร์ตที่ต่อออกใช้งานจะเป็นคอนเน็คเตอร์ 34 PIN แต่ในบอร์ด 8255 เป็นคอนเน็คเตอร์ 40 PIN ซึ่งขาใช้งานจะไม่ตรงกัน จะต้องมีการแก้ไขเล็กน้อยดังรูปที่ 2.4.2



รูปที่ 2.4.2 พอร์ตใช้งานของ CP-B2SX

<u>หมายเหตุ</u> รายละเอียดอื่นๆ และชุดคำสั่งของ PBASIC ดูได้ในภาคผนวก

2.5 บอร์ดขยายพอร์ต 72IOZ80

บอร์ด 72IO-80 เป็นบอร์ดที่ใช้ขยายส่วนอินพุทและเอาท์พุทของระบบโปรเซสเซอร์ (Processor) โดยใช้ 8255 PIA เป็นพอร์ตอินพุทเอาท์พุทซึ่งเป็นไอซี LSI ที่นิยมนำมาใช้ในการเป็นพอร์ตมากที่สุด การทำให้เป็นพอร์ตอินพุทหรือเอาท์พุทโดยการกำหนด CONTROL CODE PORT ดังตารางที่ 2.4 โดยค่าที่กำหนดนี้ต้องส่งออกให้ตัว 8255 ก่อนการใช้งาน เช่น ถ้าต้องการกำหนดให้เป็นพอร์ทเอาท์พุททั้งพอร์ท A, B และ C จะต้องกำหนดค่า 80H ให้กับ CONTROL PORT

CONTROL CODE	PORT A	PORT B	PORT C(BIT 7-14)	PORT C (BIT3-0)
80H	OUT	OUT	OUT	OUT
81H	OUT	OUT	OUT	IN
82H	OUT	IN	OUT	OUT
83H	OUT	IN	OUT	IN
88H	OUT	OUT	IN	OUT
89H	OUT	OUT	IN	IN
8AH	OUT	IN	IN	OUT
8BH	OUT	IN	IN	IN
90H	IN	OUT	OUT	OUT
91H	IN	OUT	OUT	IN
92H	IN	IN	OUT	OUT
93H	IN	IN	OUT	IN
98H	IN	OUT	IN	OUT
99H	IN	OUT	IN	IN
9AH	IN	IN	IN	OUT
9BH	IN	IN	IN	IN

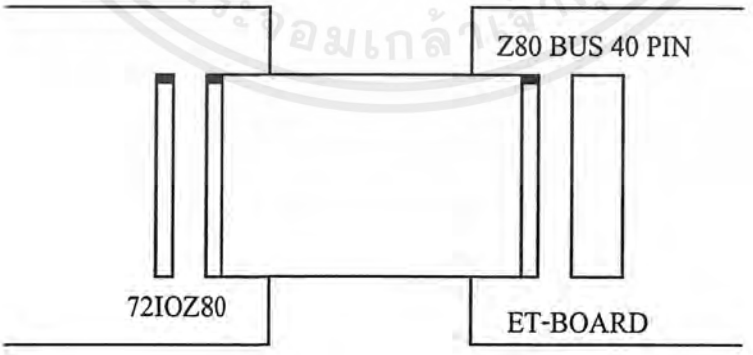
ตารางที่ 2.5 การกำหนดค่าของ Control Port

2.5.1 การดีโค้ดพอร์ต (Port Decoding)

การดีโค้ดพอร์ต 8255 โดยการใช้ไอซี TTL 74LS138 2 ตัวเป็นตัวดีโค้ดเพื่อให้ได้พอร์ตที่อยู่ในบริเวณท้ายสุดของพอร์ตซึ่งจะสามารถเข้ากับระบบต่างๆ ที่มีอยู่แล้วได้โดยไม่ต้องทับซ้อนกัน โดยการเลือก JUMPER ของพอร์ตเป็น JUMPER อยู่ที่ E0H ตำแหน่งของพอร์ตต่างๆ มีดังนี้

พอร์ต A (ตัวที่ 1) = F4H
พอร์ต B (ตัวที่ 1) = F5H
พอร์ต C (ตัวที่ 1) = F6H
Control Port (ตัวที่ 1) = F7H
พอร์ต A (ตัวที่ 2) = F8H
พอร์ต B (ตัวที่ 2) = F9H
พอร์ต C (ตัวที่ 2) = FAH
Control Port (ตัวที่ 2) = FBH
พอร์ต A (ตัวที่ 3) = FCH
พอร์ต B (ตัวที่ 3) = FDH
พอร์ต C (ตัวที่ 3) = FEH
Control Port (ตัวที่ 3) = FFH

เราสามารถต่อคอนเน็คเตอร์ 40 PINS HEADER STRIP เข้ากับ ET-BOARD หรือ Z80 CONTROL PACK ได้ดังรูป 2.5 โดยจะใช้ไฟเลี้ยง +5 V ร่วมกันทางสาย 40 PINS

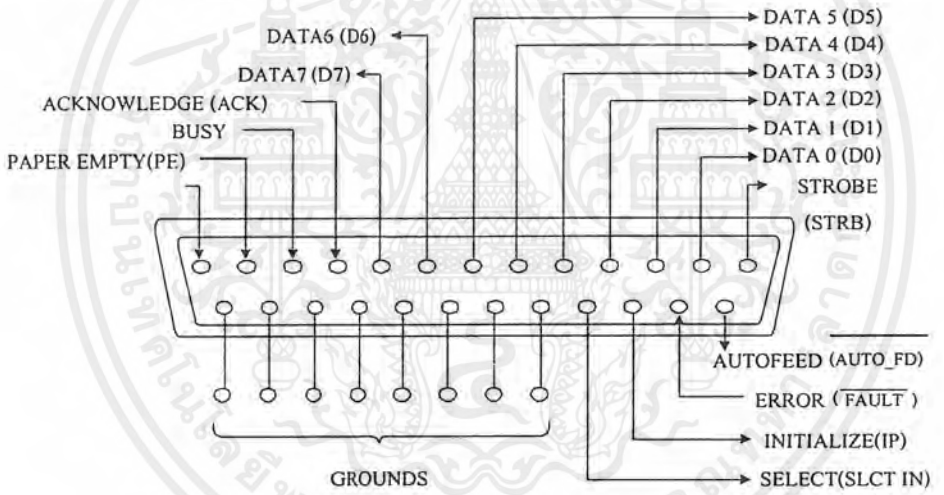


รูปที่ 2.5 ลักษณะการเชื่อมต่อบอร์ด 72I0Z80

2.6 การติดต่อสื่อสารผ่านพอร์ตขนาน

การติดต่อหรือควบคุมอุปกรณ์ภายนอก กับเครื่องคอมพิวเตอร์ผ่านทางพอร์ตขนานเป็นที่นิยมใช้กันมากเนื่องจากสามารถส่งข้อมูลได้อย่างรวดเร็วและใช้อุปกรณ์ต่อร่วมภายนอกน้อยมาก เมื่อเปรียบเทียบกับพอร์ตแบบอนุกรม พอร์ตข้อมูลแบบขนานนี้สามารถใช้เป็นอินพุทได้ 9 บิท และเป็นพอร์ตเอาท์พุทได้ 12 พอร์ต โดยพอร์ตนี้ประกอบด้วยสายควบคุม 4 เส้น สายแสดงสถานะ 5 เส้น และสายข้อมูลอีก 8 เส้น โดยปกติแล้วพอร์ตนี้มักจะเป็นแบบ D-Type ตัวเมียขนาด 25 Pin ติดอยู่ด้านหลังของเครื่องคอมพิวเตอร์ ส่วนอีกพอร์ตหนึ่งที่เป็น D-Type ตัวผู้ขนาด 25 Pin เช่นกัน นั้นเป็นพอร์ตข้อมูลแบบอนุกรมหรือ RS-232 ซึ่งแตกต่างจากพอร์ตข้อมูลแบบขนานโดยสิ้นเชิง

2.6.1 ลักษณะทางฮาร์ดแวร์(Hardware)

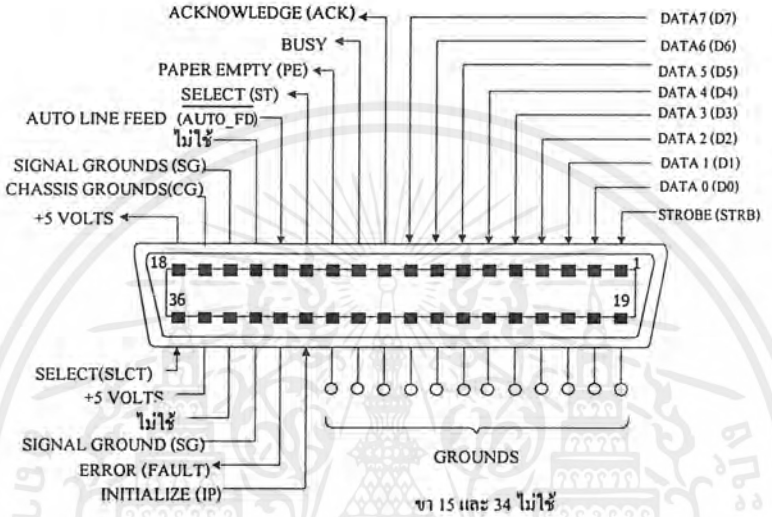


รูปที่ 2.6.1 คอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin

ลักษณะภายนอกของพอร์ตข้อมูลแบบขนานแสดงในรูปที่ 2.6.1 และ 2.6.2 ซึ่งเป็นคอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin รูป 2.6.1 และคอนเน็คเตอร์แบบ Centronics ขนาด 36 Pin รูป 2.6.2 โดยทั่วไปแล้วคอนเน็คเตอร์แบบ Centronics จะอยู่ที่เครื่องพิมพ์เป็นส่วนใหญ่ ส่วนแบบที่สามคือ 1284 Type C ซึ่งมีขนาด 36 Pin เช่นเดียวกันแต่จะมีขนาดเล็กกว่า และเพิ่มตัวล็อคสายไว้ด้วยซึ่งจะเพิ่มสายสัญญาณอีก 2 เส้นเพื่อตรวจสอบว่ามีอุปกรณ์ต่ออยู่หรือไม่

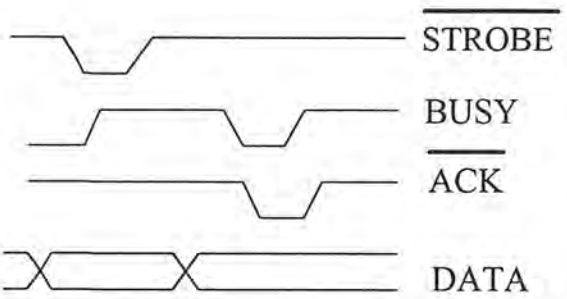
2.6.1.1 Centronics

Centronics เป็นมาตรฐานสำหรับการส่งผ่านข้อมูลจากเครื่องคอมพิวเตอร์ไปยังเครื่องพิมพ์ โดยอาศัยวิธีการแฮนเชค (Handshake) กับเครื่องพิมพ์ ซึ่งปกติแล้วจะใช้จะใช้พอร์ตมาตรฐานแบบขนานและใช้โปรแกรมควบคุมการทำงาน ลักษณะการส่งผ่านข้อมูลด้วย Centronics แสดงดังรูปที่ 2.6.2



รูปที่ 2.6.2 คอนเน็คเตอร์แบบ Centronics

จากรูปที่ 2.6.3 เมื่อต้องการส่งข้อมูลไปยังบัสข้อมูล (Data Bus) ที่ขา 2 ถึง 9 ของพอร์ตขนาน เครื่องคอมพิวเตอร์จะตรวจสอบสัญญาณ Busy ของเครื่องพิมพ์ กรณีที่เครื่องพิมพ์ไม่ว่างสัญญาณนี้จะเป็นโลจิก “0” โปรแกรมจะใส่สัญญาณ Strobe เข้าไปประมาณ 1 ไมโครวินาที และจะนำสัญญาณ Strobe ออก ข้อมูลจะถูกอ่านในช่วงขอบขาขึ้นของสัญญาณ strobe เครื่องพิมพ์จะแสดงสถานะ Busy ผ่านทาง Busy line เมื่อเครื่องพิมพ์รับข้อมูลเรียบร้อยแล้วจะตอบรับด้วยสัญญาณ $\overline{\text{ACK}}$ ซึ่งเป็นพัลซ์ (Pulse) ช่วงลบประมาณ 5 ไมโครวินาที เพื่อความรวดเร็วในการส่งข้อมูลบางครั้งจะละเลยสัญญาณ $\overline{\text{ACK}}$ เพื่อประหยัดเวลา ซึ่งจะต้องมีฮาร์ดแวร์ (Hardware) ช่วยในการทำแฮนด์เชค



รูปที่ 2.6.3 Centronics Handshake

2.6.2 ตำแหน่งพอร์ต

พอร์ตข้อมูลแบบขนานจะใช้พอร์ต 3 พอร์ตซึ่งแสดงไว้ในตาราง พอร์ตหมายเลข 3BCH เป็นพอร์ตขนานที่อยู่บนวีดีโอการ์ด (Video Card) ซึ่งสามารถตรวจสอบได้โดยการอ่านค่าจาก BIOS (Basic input/output System) ของเครื่องคอมพิวเตอร์ พอร์ตขนานพอร์ตแรกหรือ LPT1 ปกติจะกำหนดให้อยู่ที่หมายเลข 378H ขณะที่ LPT2 จะอยู่ที่ 287H ซึ่งพอร์ตทั้งสองนี้จะใช้กันมาก สำหรับพอร์ตขนาน ตำแหน่งของพอร์ตเหล่านี้จะแตกต่างตามชนิดของเครื่องคอมพิวเตอร์

Address	Notes
3BCH	Used for Parallel Ports which were incorporated in to video card and now,commonly an option for ports controlled by BIOS.Doesn't support ECP address
378H-37FH	Usual Address for LPT1
278H-27FH	Usual Address for LPT2

ตารางที่ 2.6 ตำแหน่งพอร์ต

เมื่อเปิดเครื่องคอมพิวเตอร์ BIOS จะกำหนดหมายเลขพอร์ตที่มีอยู่บนเครื่องและกำหนดชื่อเรียก LPT1, LPT2 และ LPT3 ให้กับพอร์ตเหล่านั้น ในครั้งแรกไบออสจะตรวจสอบที่ตำแหน่ง 3BCH ถ้าพบพอร์ตขนานที่ตำแหน่งนี้จะกำหนดให้เป็น LPT1 หลังจากนั้นจึงตรวจสอบที่ตำแหน่ง 378H หากพบพอร์ตขนานอีกจะกำหนดด้วยชื่อเรียกของอุปกรณ์ตัวต่อไป ในบางครั้งผู้ใช้สามารถกำหนดพอร์ตเองได้ ซึ่งตำแหน่ง 378H อาจไม่ใช่ LPT1 ก็ได้

2.7 การติดต่อสื่อสารผ่านพอร์ตอนุกรม

การติดต่อสื่อสารผ่านพอร์ตอนุกรมนั้นจะยุ่งยากมากกว่าการใช้พอร์ตขนานสำหรับอุปกรณ์ทั่วๆ ไปแล้วจะต้องการข้อมูลในการประมวลผล (Process) แบบขนาน ดังนั้นจึงเพิ่มอุปกรณ์ที่ทำหน้าที่ในการแปลงข้อมูลแบบอนุกรมมาเป็นแบบขนาน อุปกรณ์นี้คือ UART (Universal Asynchronous Receiver/Transmitter) และซอฟแวร์ที่ใช้ควบคุมการทำงานก็จะยุ่งยากกว่าการควบคุมผ่านพอร์ตมาตราฐานแบบขนาน แต่การสื่อสารผ่านพอร์ตอนุกรมก็มีข้อดีหลายอย่างดังนี้

1. การติดต่อผ่านพอร์ตอนุกรมสามารถใช้สายได้ยาวกว่าการติดต่อสื่อสารผ่านพอร์ตแบบขนาน โดยที่พอร์ตอนุกรมจะใช้ระดับแรงดันในช่วง –3 ถึง -25 V แทนโลจิก "0" และใช้แรงดันในช่วง +3 ถึง +25 V แทนโลจิก "1" ดังนั้นจะเห็นว่าช่วงการสวิง (Swing) ของแรงดันมีค่าประมาณ 50 V ส่วนพอร์ตขนานมีช่วงสวิง 5 V เท่านั้น ซึ่งจะเห็นได้อย่างชัดเจนว่าหากมีการสูญเสียในสายแล้ว การส่งข้อมูลผ่านพอร์ตอนุกรมจะสามารถส่งข้อมูลไปได้ไกลกว่าอย่างแน่นอน

2. ใช้จำนวนสายสัญญาณน้อยกว่าการส่งข้อมูลแบบขนาน ในกรณีที่อุปกรณ์อยู่ห่างจากเครื่องคอมพิวเตอร์มากๆ ย่อมจะสะดวกกว่าหากจะเดินสายเพียง 3 เส้น ซึ่งเป็นลักษณะของนัลโมเด็ม (Null Modem) เมื่อเทียบกับการเดินสายจำนวน 19 หรือ 25 เส้น ในการใช้พอร์ตขนาน

3. ปัจจุบันอุปกรณ์ที่ใช้แสงอินฟราเรด (Infrared) ได้รับความนิยมมากขึ้น ซึ่งจะเห็นได้จากอุปกรณ์ประเภทสมุดบันทึกอิเล็กทรอนิกส์ เครื่องคอมพิวเตอร์แบบกระเป๋าหิ้ว เป็นต้น จะมีการติดต่อสื่อสารโดยใช้แสงอินฟราเรดร่วมอยู่ด้วย และแน่นอนว่าการใช้แสงอินฟราเรดก็จะต้องใช้การติดต่อสื่อสารแบบอนุกรม เนื่องจากความไม่สะดวกอย่างยิ่งในการที่จะส่งข้อมูลแบบขนานด้วยอินฟาเรด

2.7.1 ลักษณะทางฮาร์ดแวร์

อุปกรณ์ที่ใช้การติดต่อสื่อสารแบบอนุกรมสามารถแยกออกได้เป็น 2 ประเภทคือ DCE (Data Communications Equipment) อุปกรณ์เหล่านี้ได้แก่ โมเด็ม (Modem) พล็อตเตอร์ (Plotter) เป็นต้นและ DTE (Data Terminal Equipment) ซึ่งก็คือคอมพิวเตอร์นั่นเอง ข้อกำหนดทางไฟฟ้าของพอร์ตอนุกรมกำหนดเป็นมาตรฐานโดย EIA (Electronics Industry Association) หรือ RS-232 ซึ่งประกอบไปด้วยสิ่งต่างๆ เหล่านี้

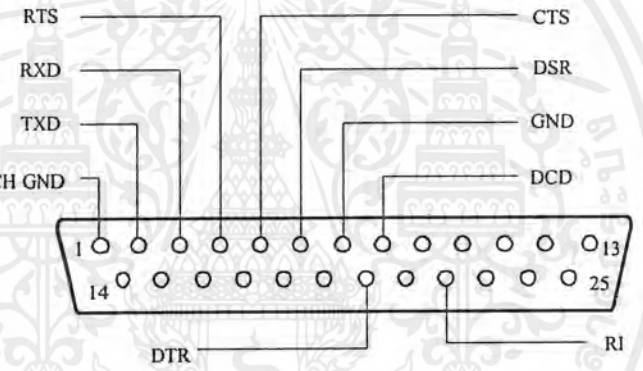
1. ช่วงไม่มีข้อมูล (space) หรือโลจิก "0" ต้องมี่แรงดันอยู่ในช่วง –3 V ถึง –25 V
2. ช่วงข้อมูล (mark) หรือ โลจิก "1" ต้องมีแรงดันอยู่ช่วง +3 ถึง +25 V
3. แรงดันในช่วง –3 ถึง +3 ไม่มีการนิยามไว้

4. แรงดันในขณะเปิดวงจรต้องมีค่าไม่เกิน 25 โวลท์

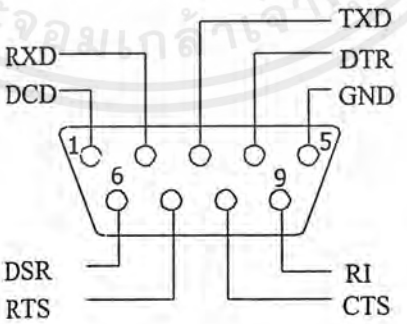
5. กระแสขณะช็อตวงจรมีค่าไม่เกิน 500 มิลลิแอมป์

มาตรฐานของ RS-232 นั้นนอกจากจะมีคุณสมบัติดังที่กล่าวมาแล้วยังจะต้องประกอบด้วยค่าความจุของสาย อัตราบอดเรต (Baud Rate) สูงสุด ซึ่งรายละเอียดต่างๆ ดูได้จากมาตรฐานของ EIA RS232-C ตามมาตรฐานของของ RS232C กำหนดอัตราบอดไว้ที่ 20,000 บิทต่อวินาทีซึ่งค่อนข้างจะช้าเกินไปสำหรับมาตรฐานในปัจจุบัน ในช่วงหลังจึงได้มีการกำหนดมาตรฐาน RS232D ขึ้นและยังคงใช้กันอยู่ในปัจจุบัน

พอร์ตอนุกรมนี้จะมีอยู่ 2 ขนาดคือ คอนเน็คเตอร์แบบ D-type ตัวผู้ขนาด 25 Pin รูปที่ 2.7.1 และคอนเน็คเตอร์แบบ D-type ตัวผู้เช่นกันขนาด 9 Pin รูปที่ 2.7.2 ซึ่งคอนเน็คเตอร์ทั้ง 2 แบบนี้จะติดอยู่ที่ด้านหลังของเครื่องคอมพิวเตอร์ ตารางที่ 2.7.1 แสดงตำแหน่งขาสัญญาณของพอร์ตอนุกรม



รูปที่ 2.7.1 คอนเน็คเตอร์แบบ DB-25



รูปที่ 2.7.2 คอนเน็คเตอร์แบบ DB-25

Full Name	D-type 25 Pin No.	D-type 9 Pin No.	Abbreviation
Transmit Data	Pin 2	Pin 3	TD
Receive Data	Pin 3	Pin 2	RD
Request To Send	Pin 4	Pin 7	RTS
Clear to Send	Pin 5	Pin 8	CTS
Data Set Ready	Pin 6	Pin 6	DSR
Signal Ground	Pin 7	Pin 5	SG
Crrier Detect	Pin 8	Pin 1	CD
Data Terminal Ready	Pin 20	Pin 4	DTR
Ring Indicator	Pin 22	Pin 9	RI

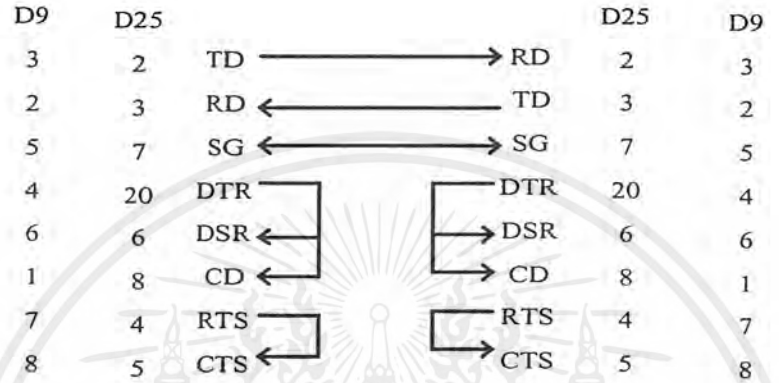
ตารางที่ 2.7.1 ตำแหน่งขาสัญญาณของพอร์ตอนุกรม

Abbreviation	Full Name	Function
TD	Transmit Data	- Serial Data Output (TXD)
RD	Reciev Data	- Serial Data Input (RXD)
CTR	Clear to Send	- Indicates that the Modem is ready to exchange data
DCD	Data Carrier Detect	- This line becomes active when the Modem detects a"Carrier" from the other Modem.
DSR	Data Set Ready	- This tells the UART that the Modem is ready to establish a link .
DTR	Data Transmit Ready	- This is the opposite to DSR .This tells the Modem that the UART is ready to link.
RTS	Request to send	- This line informs the Modem that the UART is ready to exchange data.
RI	Ring Indicator	- Goes active when Modem detects a ringing signal from the PSTN.

ตารางที่ 2.7.2 หน้าที่ของขาสัญญาณแต่ละเส้น

2.7.2 นัลโมเด็ม (Null Modem)

นัลโมเด็มใช้เพื่อเชื่อมต่อ DTE สองเครื่องเข้าด้วยกัน ซึ่งสามารถนำไปใช้ในการส่งผ่านข้อมูลระหว่างเครื่องคอมพิวเตอร์ด้วยโปรโตรคอล (Protocol) แบบ Zmodem รูปที่ 2.7.3 แสดงการเชื่อมต่อสายของนัลโมเด็ม



รูปที่ 2.7.3 การเชื่อมต่อสายของนัลโมเด็ม

ลักษณะการต่อสายของนัลโมเด็มจะใช้สายเพียง 3 เส้น (TD, RD และ SG)โดยที่สาย TD ของคอมพิวเตอร์เครื่องหนึ่งจะต่อไปยังสาย RD ของคอมพิวเตอร์อีกเครื่องหนึ่ง และในทางกลับกันสาย RD ของคอมพิวเตอร์อีกเครื่องหนึ่งจะต่อกลับไปยังสาย TD ของคอมพิวเตอร์อีกเครื่องหนึ่ง

ส่วนสาย SG ของคอมพิวเตอร์ทั้งสองเครื่องจะถูกต่อเข้าด้วยกัน การทำลูปแบ็ค (Loop back) นั้นจะนำสัญญาณ DTR ป้อนเข้ากับ DSR และ CD ของคอมพิวเตอร์ทั้งสองเครื่อง เนื่องจากเมื่อสัญญาณ DTR แอกทีฟ (Active) จะทำให้ DSR และ CD แอคทีฟตามทันทีทำให้คอมพิวเตอร์เข้าใจว่าโมเด็มพร้อมที่จะทำงานแล้ว ส่วนการต่อสาย RTS เข้ากับ CTS ของคอมพิวเตอร์ทั้งสองเครื่องนั้นเพื่อต้องการให้คอมพิวเตอร์ติดต่อสื่อสารกันด้วยความเร็วที่เท่ากัน

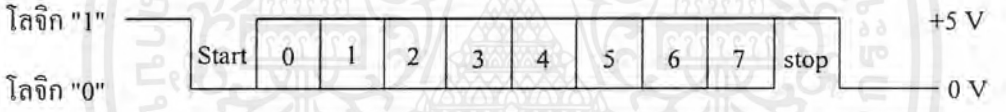
ตำแหน่งพอร์ตและ IRQ (Interrupt Request) ของพอร์ตอนุกรมนั้นมีลักษณะการกำหนดและการอ่านตำแหน่งจากไบออส (Bios) คล้ายๆ กับของพอร์ตขนาน ในตารางที่ 3.8 แสดงตำแหน่งมาตราฐานของพอร์ตอนุกรมและ IRQ ที่ใช้สำหรับตำแหน่งพอร์ตแต่ละเบอร์

Name	Address	IRQ
Com1	3F8	4
Com2	2F8	3
Com3	3E8	4
Com4	2E8	3

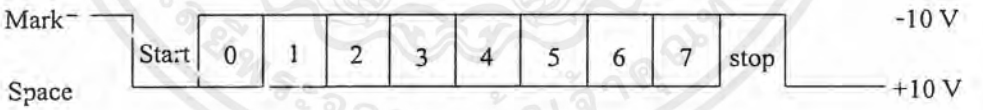
ตารางที่ 2.7.3 ตำแหน่งมาตราฐานของพอร์ตอนุกรม

2.7.3 รูปคลื่นของ RS-232

การสื่อสารโดยใช้ RS-232 นั้นเป็นการสื่อสารแบบอซิงโครนัส (Asynchronous) นั่นคือในการสื่อสารไม่จำเป็นต้องมีการส่งสัญญาณนาฬิกา (Clock) ไปกับข้อมูลด้วย ซึ่งแต่ละเวิร์ด (Word) ของข้อมูลจะใช้บิตเริ่มต้น (start bit) ในการซิงโครนัสข้อมูล โดยอาศัยสัญญาณนาฬิกาที่แต่ละด้านของการส่งข้อมูล



รูปที่ 2.7.4 รูปคลื่นของการส่งสัญญาณแบบอนุกรมด้วยระดับแรงดันของ TTL/CMOS



รูปที่ 2.7.5 รูปคลื่นของสัญญาณที่รับส่งผ่านพอร์ต RS-232

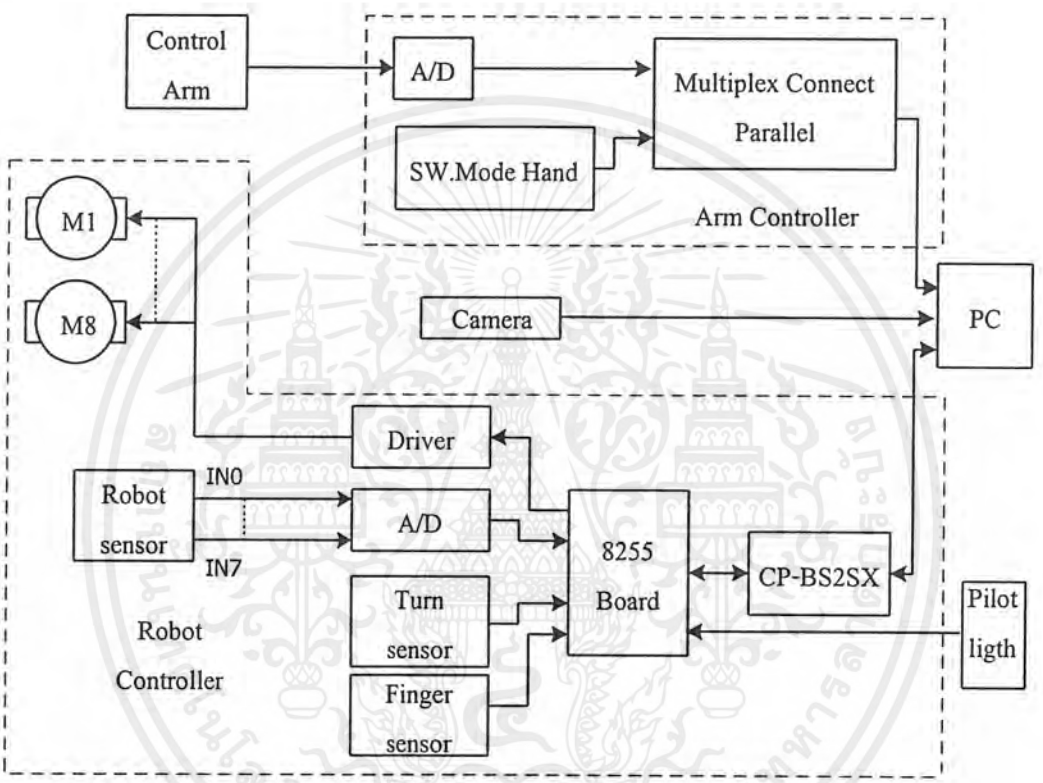
ไดอะแกรมข้างบนแสดงรูปคลื่นจาก UART เมื่อใช้รูปแบบของการส่งแบบ 8N1ซึ่ง 8N1 นี้หมายถึง ประกอบด้วยข้อมูล 8 บิท ไม่มีพาริตี้ (Parity) และ 1 บิทหยุด ในการส่งข้อมูลจะต้องส่งบิทที่มีนัยสำคัญน้อยที่สุด (LSB) ก่อนส่วนบิทที่มีนัยสำคัญมากที่สุด (MSB) จะอยู่ลำดับสุดท้าย

ในรูปที่ 2.7.5 แสดงรูปคลื่นของการรับและส่งข้อมูลผ่านพอร์ต RS-232 ซึ่งระดับของแรงดันจะแตกต่างจากในรูป 2.7.4 แต่มีลักษณะข้อมูลเหมือนกัน

บทที่ 3

การออกแบบระบบอิเล็กทรอนิกส์

3.1 ลักษณะโดยรวมของระบบควบคุมหุ่นยนต์แขนกล



รูปที่ 3.1 บล็อกไดอะแกรมแสดงระบบควบคุมหุ่นยนต์แขนกล

จากแผนภูมิแสดงขั้นตอนการทำงาน (Block diagram) สามารถแบ่งการทำงานได้ 2 ส่วนคือ

- ส่วนของแขนควบคุม (Arm Control) ซึ่งใช้การส่งข้อมูลติดต่อกับ PC (Personal Computer) ทางพอร์ตขนาน
- ส่วนของตัวหุ่นยนต์ ซึ่งใช้การส่งข้อมูลติดต่อกับ PC ทางพอร์ตอนุกรม

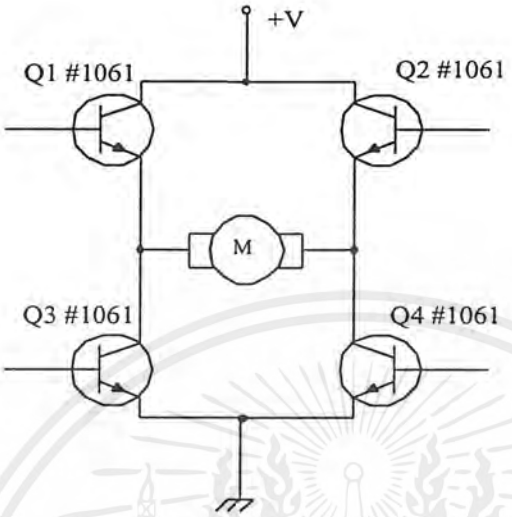
โดยทั้งสองส่วนนี้เชื่อมต่อกันโดยมีตัวกลางคือ PC และทั้งสองส่วนจะทำงานแยกอิสระต่อกันในขณะที่ทำงานโดยใช้แขนควบคุม สัญญาณที่ได้จากตัวตรวจจับการเคลื่อนไหวของชุดแขนควบคุมจะเข้าสู่วงจร A/D (Analog to Digital Converter) เพื่อแปลงสัญญาณอะนาล็อก (Analog) เป็นสัญญาณดิจิตอล (Digital) สัญญาณดิจิตอลที่ได้จะผ่านวงจร SW. Connect Parallel เพื่อเลือกว่าจะรับข้อมูลจาก Control Arm หรือจาก SW. Mode Hand ผ่านทางพอร์ตขนานเข้าสู่ PC

ในส่วนของ ROBOT จะทำงานภายใต้การควบคุมของไมโครคอนโทรลเลอร์ CP-BS2SX ซึ่งจะทำงานโดยการเชื่อมกับ PC เพื่อรับส่งข้อมูล โดยเมื่อ PC ต้องการสั่งงานก็จะส่งข้อมูลมาให้ไมโครคอนโทรลเลอร์ เมื่อไมโครคอนโทรลเลอร์ได้รับคำสั่งจาก PC แล้วก็จะทำการควบคุมบอร์ด 8255 เพื่อทำการติดต่อกับพอร์ตที่ต่อกับภาค Driver แล้วส่งข้อมูลให้ภาค Driver จากนั้นบอร์ด 8255 ก็จะรับคำสั่งจากภาค A/D ซึ่งรับสัญญาณมาจากตัวตรวจจับการเคลื่อนไหวของหุ่นยนต์ และรับค่าจาก Turn Sensor และ Finger Sensor แล้วไมโครคอนโทรลเลอร์ก็จะแปลงค่าที่ได้รับ แล้วส่งให้ PC เพื่อนำไปประมวลผลต่อไป

3.2 ระบบอิเล็กทรอนิกส์ที่ใช้ในโครงงาน

1. วงจรควบคุมทิศทางการหมุนของมอเตอร์
2. วงจรภาคอินพุทแบบอนาล็อก
3. วงจรลิมิตสวิตช์ (Limit switch)
4. วงจรในส่วนมือจับชิ้นงาน
5. วงจร Connect Parallel
6. วงจร Switch Mode Hand
7. วงจร Connect Parallel Port

3.2.1 วงจรควบคุมทิศทางการหมุนของมอเตอร์



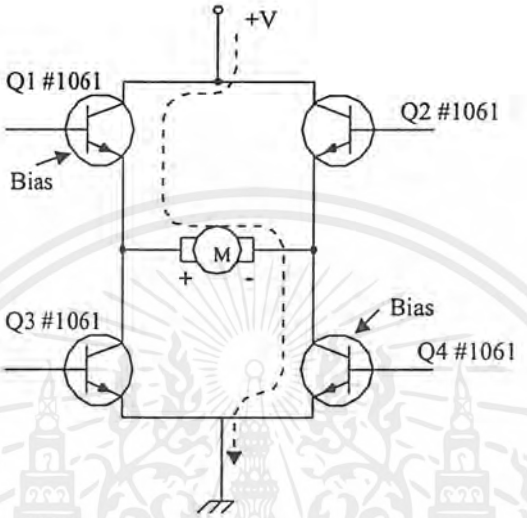
รูปที่ 3.2 แสดงวงจรควบคุมมอเตอร์

3.2.1.1 การทำงานของวงจร

การควบคุมทิศทางการหมุนของมอเตอร์ตามวงจรนี้สามารถทำได้โดยการควบคุมการไบอัส (Bias) ที่ขาเบส (base) ของทรานซิสเตอร์ทั้ง 4 ตัว ซึ่งทำงานในลักษณะของสวิตซ์ตัดต่อวงจรโดยแยกการทำงานออกเป็นคู่ เพื่อควบคุมทิศทางของกระแสที่ไหลผ่านมอเตอร์ โดยให้ไบอัส Q1 และ Q4 ทำงานพร้อมกันเพื่อควบคุมกระแสให้ไหลในทิศทางหนึ่ง และให้ไบอัส Q2 และ Q3 ทำงานพร้อมกันเพื่อควบคุมกระแสให้ไหลในอีกทิศทางหนึ่ง สมมติให้ป้อนแรงดันให้แก่มอเตอร์ดังรูปที่ 3.3 ก มอเตอร์จะมีทิศทางการหมุนตามเข็มนาฬิกา และถ้าป้อนแรงดันให้แก่มอเตอร์ดังรูปที่ 3.3 ข มอเตอร์จะมีทิศทางการหมุนทวนเข็มนาฬิกา

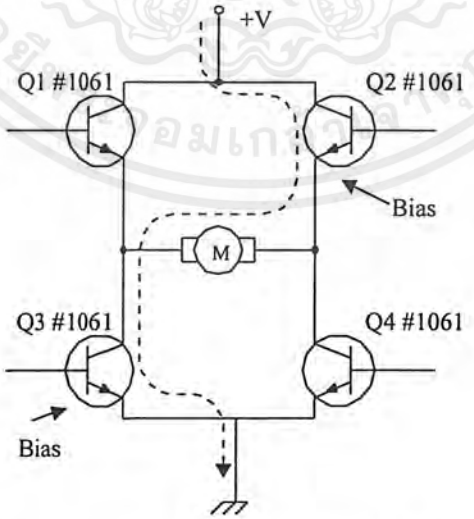
รูปที่ 3.3 แสดงทิศทางการหมุนของมอเตอร์

เมื่อเราต้องการให้มอเตอร์หมุนตามเข็มนาฬิกาเราก็ไบอัสทรานซิสเตอร์ Q1 และ Q4 พร้อมกัน ทิศทางการไหลของกระแสในวงจรจะเป็นดังรูป 3.4



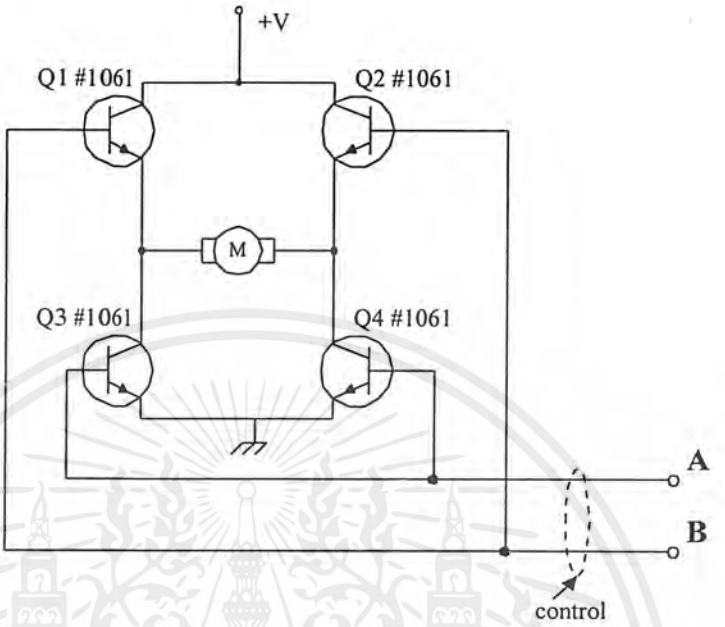
รูปที่ 3.4 แสดงทิศทางการไหลของกระแสเมื่อ Q1 และ Q4 ได้รับไบอัส

เมื่อเราต้องการให้มอเตอร์หมุนทวนเข็มนาฬิกาเราก็ไบอัสทรานซิสเตอร์ Q2 และ Q3 พร้อมกัน ทิศทางการไหลของกระแสในวงจรจะเป็นดังรูปที่ 3.5



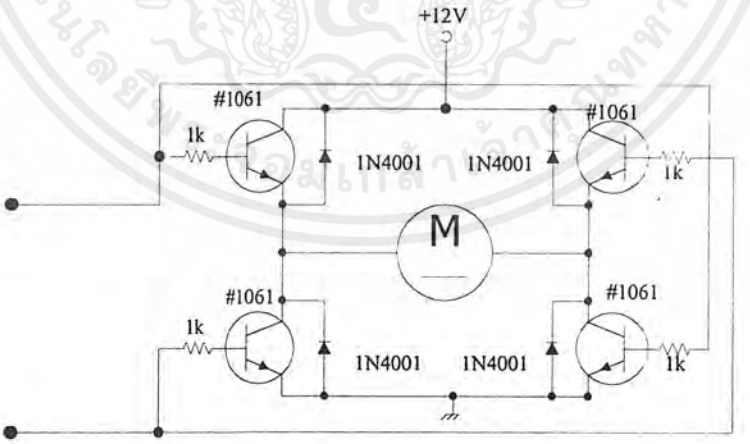
รูปที่ 3.5 แสดงทิศทางการไหลของกระแสเมื่อ Q2 และ Q3 ได้รับไบอัส

เมื่อเราไม่ต้องการให้มอเตอร์หมุนเราก็เพียงแต่หยุดป้อนไบอัสทรานซิสเตอร์ทั้ง 4 ตัว ก็จะไม่มีกระแสไหลในวงจร



รูปที่ 3.6 การควบคุมของวงจร

3.2.1.2 การต่อไดโอดเพื่อป้องกันทรานซิสเตอร์เสียหาย



รูปที่ 3.7 การต่อไดโอดเพื่อป้องกันทรานซิสเตอร์เสียหาย

จากรูปพบว่าเราต้องต่อไดโอดระหว่างขาคอลเล็คเตอร์ (Collector) และอิมิตเตอร์ (Emitter) ของทรานซิสเตอร์ทุกตัวเพื่อป้องกันกระแสไหลย้อนกลับ (Reverse bias) อันเนื่องมาจากการยุบตัวของสนามแม่เหล็กของมอเตอร์ โดยไดโอดจะเป็นตัวลัดวงจรเมื่อเกิดกระแสไฟย้อนกลับเพื่อป้องกันรอยต่อ (Junction) ของทรานซิสเตอร์ขณะเกิดกระแสย้อนกลับ

3.2.1.3 การนำไปประยุกต์ใช้งาน

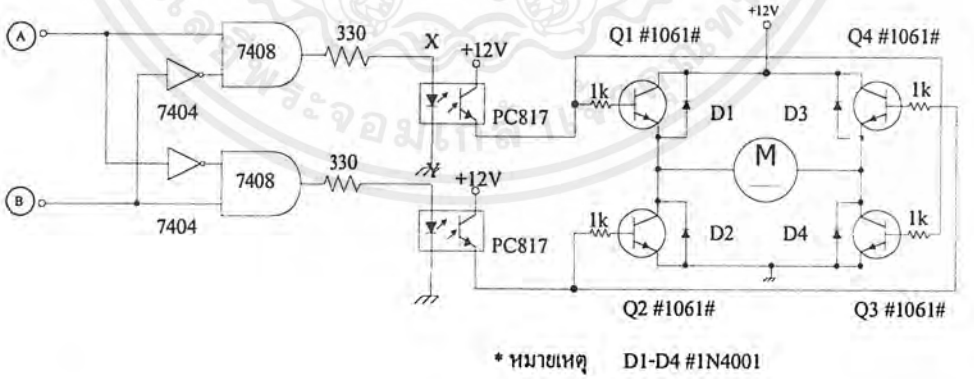
จากการทำงานของวงจรควบคุมการหมุนของมอเตอร์ที่ต้องการสภาวะการทำงาน 3 สภาวะ ดังนี้

A=0, B=0

A=0, B=1

A=1, B=0

ตัวตรวจจับการเคลื่อนไหว จะกำเนิดสัญญาณ DC เพื่อนำไปเปรียบเทียบและประมวลผลได้สัญญาณเอาต์พุตสำหรับควบคุมการหมุนของมอเตอร์ผ่านวงจรป้องกันความผิดพลาดจากการทำงานเพื่อป้องกันสัญญาณเอาท์พุททั้ง 2 ที่อาจเป็นโลจิก 1 พร้อมกัน เมื่อผ่านวงจรป้องกันความผิดพลาดจากการทำงานแล้วจะได้สัญญาณเป็นสัญญาณ A และ B เป็นสัญญาณควบคุมมอเตอร์ และการทำงานของมอเตอร์นี้จะมีผลต่อการเปลี่ยนแปลงของตัวตรวจจับการเคลื่อนไหวดังรูปที่ 3.8



รูปที่ 3.8 แสดงวงจรวงจรควบคุมทิศทางการหมุนของมอเตอร์

3.2.1.4 การทำงานของวงจร

การอธิบายการทำงานของวงจรส่วนนี้จะไม่เจาะลึกถึงการทำงานโดยละเอียดมากนักเพราะการทำงานส่วนของวงจรควบคุมการหมุนของมอเตอร์ทำงานของวงจรส่วนต่าง ๆ จะถูกควบคุมจากคอมพิวเตอร์วงจรป้องกันความผิดพลาดจากการทำงาน การทำงานของวงจรในส่วนนี้จึงจะกล่าวเพียงการเชื่อมต่อสัญญาณระหว่างวงจรทั้งหมดและการทำงานโดยรวมเท่านั้น

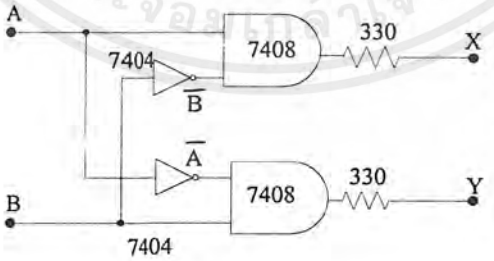
จากวงจรเราสามารถควบคุมการหมุนของมอเตอร์โดยใช้สายControl เพียง 2 เส้น กำหนดเป็น 3 สภาวะ คือ

A	B	ทิศทางการหมุนของมอเตอร์
0	0	หยุดหมุน
0	1	หมุนทวนเข็มนาฬิกา
1	0	หมุนตามเข็มนาฬิกา

ตารางที่ 3.1 สัญญาณควบคุมการหมุน

ซึ่งสภาวะ A = 1 และ B = 1 เราจะไม่ยอมให้เกิดขึ้นเพราะจะเกิดความเสียหายแก่ทรานซิสเตอร์ได้เพราะกระแสจะไหลผ่านทรานซิสเตอร์ทั้ง 4 ตัวอย่างมากมาย เนื่องจากทรานซิสเตอร์ทั้ง 4 ตัวจะมีสถานะเป็นโหลด ที่มีความต้านทานต่ำ

เราสามารถป้องกันการเกิดสภาวะ A =1 และ B=1 ได้โดยใช้วงจรป้องกันความผิดพลาดจากการทำงาน ดังรูป 3.9 โดยให้สัญญาณ control คือ A และ B



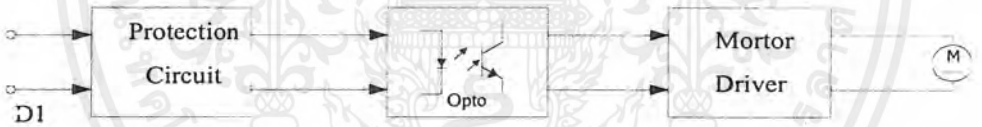
รูปที่ 3.9 วงจรป้องกันมอเตอร์จากความผิดพลาดในการทำงาน

ตารางที่ 3.2 Truth table ของวงจรป้องกันมอเตอร์จากความผิดพลาดจากการทำงาน

A	B	$\overline{A}$	$\overline{B}$	X = (A*$\overline{B}$)	Y = (B*$\overline{A}$)
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	0	0	0

จากตาราง Truth Table จะเห็นว่าวงจรป้องกันความผิดพลาดจากการทำงานจะไม่ยอมให้เกิดสภาวะ A = 1 และ B = 1 พร้อมกันไปปรากฏที่วงจรควบคุมทิศทางการหมุนของมอเตอร์ได้ เพื่อเป็นการ ป้องกันวงจรขับมอเตอร์

3.2.1.5 วงจรออปโต

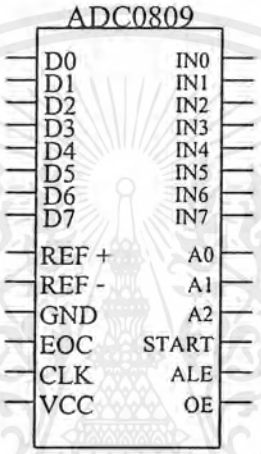


รูปที่ 3.10 แสดงวงจรออปโต

จากรูปที่ 3.10 ออปโตจะนำมาใช้สำหรับขยายกระแสที่ได้จากวงจรป้องกันความผิดพลาดจากการทำงาน แล้วป้อนเข้าสู่วงจรขับมอเตอร์อีกต่อหนึ่ง

3.2.2 วงจรภาคอินพุทแบบอนาล็อก (Analog To Digital Converter Cricuit)

วงจรภาคอินพุทแบบอนาล็อกในโครงงานใช้ไอซี เอดีซี (Analog to Digital Converter) จะทำหน้าที่เปลี่ยนสัญญาณแบบอนาล็อกให้เป็นสัญญาณทางดิจิตอล ซึ่งในโครงงานเลือกใช้ไอซี เอดีซีเบอร์ ADC0809 ซึ่งไอซีเบอร์นี้สามารถรับสัญญาณอินพุทแบบอนาล็อกได้สูงสุดถึง 5 โวลท์และยังสามารถรับได้ 8 ช่องสัญญาณอินพุทโดยสามารถเลือกได้ว่าต้องการจะแปลงสัญญาณอนาล็อกช่องใดให้เป็นสัญญาณดิจิตอลขนาด 8 บิต

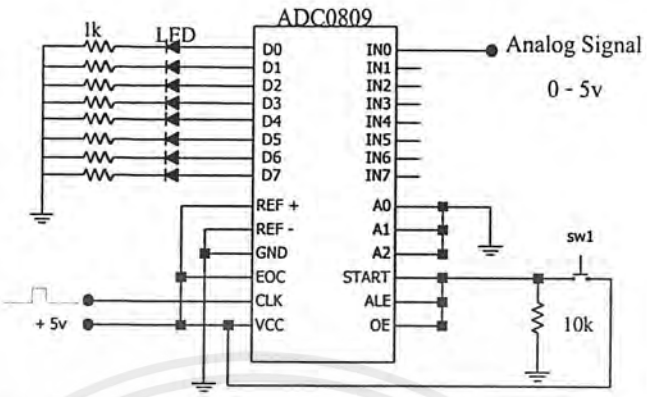


รูปที่ 3.11 ไอซี ADC 0809

คุณสมบัติของไอซี เอดีซี 0809 มีดังนี้

1. สามารถต่อใช้งานรวมกับไมโครโปรเซสเซอร์ (Microprocessors) ได้
2. มีช่องสัญญาณอินพุท 8 ช่องสัญญาณ
3. เอ้าต์พุทมีขนาด 8 บิตแบบ TTL
4. สัญญาณอินพุทมีขนาด 0 ถึง 5 โวลท์
5. การผิดพลาดในการแปลงสัญญาณบวกลบ 1 บิต
6. แรงดันไฟฟ้าที่ใช้ 5 โวลท์
7. แรงดันไฟฟ้าอ้างอิง 0 และ 5 โวลท์
8. กำลังงานที่ใช้ต่ำเพียง 15 มิลลิวัตต์
9. ใช้สัญญาณนาฬิกา 500 กิโลเฮิรท์
10. เวลาที่ใช้การแปลงสัญญาณ 100 ไมโครวินาที

3.2.2.1 วงจรทดลองการ แปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (ADC)



รูปที่ 3.12 วงจรทดลองไอซี ADC 0809

เมื่อต่อวงจรดังรูปข้างบน IC ADC 0809 จะทำงานเมื่อกด SW1 ทำให้ขา START, ALE และ OE ได้ Logig1 ADC 0809 จะรับสัญญาณอนาล็อกทางอินพุตแปลงเป็นสัญญาณดิจิตอลทางเอาต์พุต ที่ขา D0-D7 ซึ่งเอาต์พุตแต่ละขาจะมี LED ต่ออนุกรมกับความต้านทาน 220 โอห์ม ไว้เพื่อให้มองเห็นได้กรณีมีสัญญาณเข้ามาจากนั้นให้อ่านค่าที่ได้เป็น Binary แปลงเป็น HEX.Code ซึ่งได้จากผลการทดลองวงจร A/D (IC ADC 0809)

ตารางที่ 3.3 ค่าโดยละเอียด

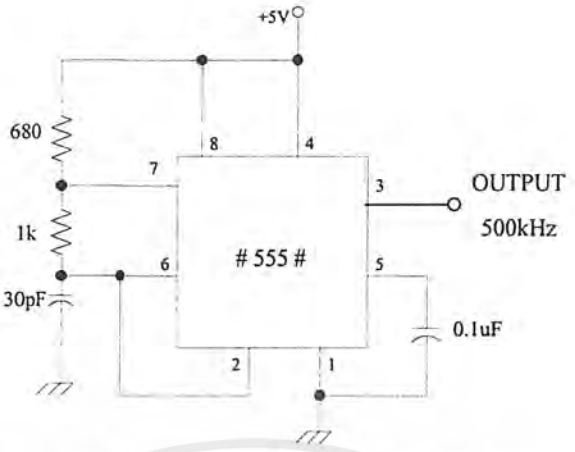
Volt (V)	Hex.Code
0.33	10H
0.34	11H
0.36	12H
0.38	13H
0.40	14H
0.45	17H
0.50	19H
0.55	1CH
0.60	1EH
0.65	21H
0.70	24H
0.75	26H
0.80	29H

Volt (V)	Hex.Code
1.0	40H
1.5	5FH
2.0	77H
2.5	94H
3.0	B7H
3.5	D7H
4.0	F6H
4.5	FFH
5.0	อินพุตค่าไม่ถึง

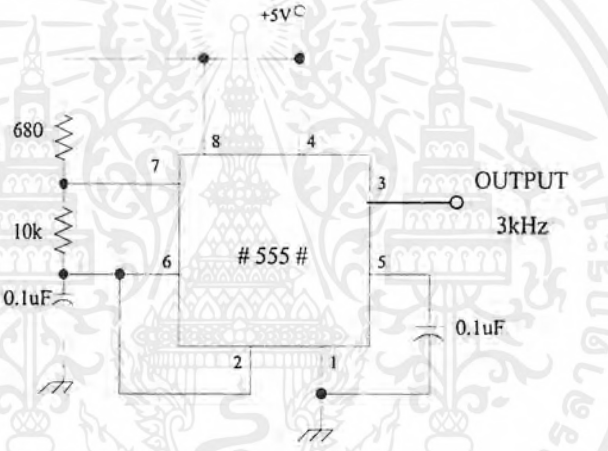
ตารางที่ 3.4 ค่าโดยประมาณ

3.2.2.2 หลักการทำงานของวงจร

ไอซี เอดีซี จะทำหน้าที่เปลี่ยนสัญญาณอนาล็อกให้เป็นสัญญาณทางดิจิตอล โดยใช้หลักการสุ่มรับสัญญาณอนาล็อกทางอินพุท (Sampling) แล้วนำขนาดของสัญญาณอนาล็อกนั้นมาเปรียบเทียบกับแรงดันอ้างอิงของวงจรแล้วจึงเปลี่ยนเป็นค่าของข้อมูลแบบดิจิตอลซึ่งความเร็วในการสุ่มขนาดสัญญาณ (Sampling Rate) ซึ่งได้จากวงจรกำเนิดความถี่ (Oscillator) ผลิตความถี่ 500 กิโลเฮิร์ทป้อนให้กับขาที่ 10 ของ ไอซี ADC0809 เป็นขาของสัญญาณนาฬิกา เนื่องจากไอซี ADC0809 ไม่มีวงจรกำเนิดสัญญาณนาฬิกาในตัวไอซีเหมือนกับไอซีเอดีซีเบอร์อื่นๆ เช่น ADC0804 เป็นต้น โดยในโครงงานนี้ได้สร้างวงจรกำเนิดสัญญาณเพื่อป้อนให้กับไอซี ADC0809 จำนวน 2 วงจรหรือ 2 ความถี่ซึ่งใช้ไอซีเบอร์ 555 เป็นตัวสร้างความถี่ โดยอาศัยการปรับค่าความต้านทาน (Resister) และค่าความเก็บประจุไฟฟ้า (Capasitor) ที่เหมาะสมเพื่อให้ได้ความถี่ตามที่ต้องการ

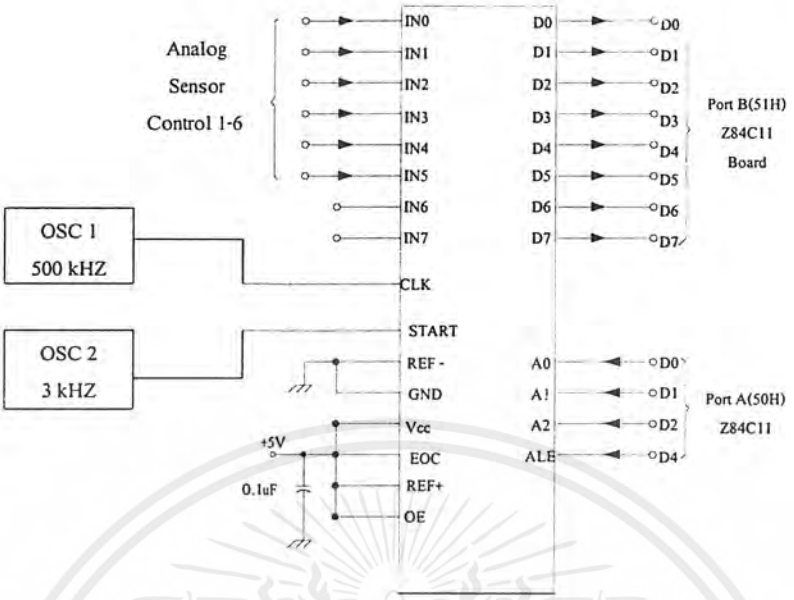


รูปที่ 3.13 วงจรกำเนิดความถี่ 500 กิโลเฮิร์ท



รูปที่ 3.14 วงจรกำเนิดความถี่ 3 กิโลเฮิร์ท

ส่วนความถี่อีกความถี่ที่สร้างขึ้นมานั้นมีค่าประมาณ 1 ถึง 3 กิโลเฮิรท์จะป้อนให้ กับขาที่ 6 ของไอซี ADC0809 ซึ่งเป็นขาของสัญญาณที่ใช้การเริ่มทำการแปลง (Start) สัญญาณแบบอนาล็อก เป็นสัญญาณทางดิจิตอล ความถี่ทั้งสองนี้จะมีความสัมพันธ์กัน โดยที่ความถี่ที่เป็นสัญญาณนาฬิกา จะมีค่ามากกว่าความถี่ที่เป็นสัญญาณเริ่มแปลงสัญญาณถ้าสัญญาญความถี่ทั้งสองมีค่าใกล้กันจะมีผลทำให้ไอซี ADC0809 ไม่สามารถที่จะทำการเปลี่ยนสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลได้ เนื่องจากเวลาที่ใช้ในการแปลงสัญญาณซึ่งยังแปลงยังไม่เสร็จก็มีสัญญาณเริ่มทำการแปลงใหม่เข้าอีกจึงทำให้ไอซี ADC0809 จะแปลงสัญญาณอนาล็อกค่าเดิมตลอดทำให้ได้ค่าสัญญาณดิจิตอลเป็นค่าเดิมด้วย



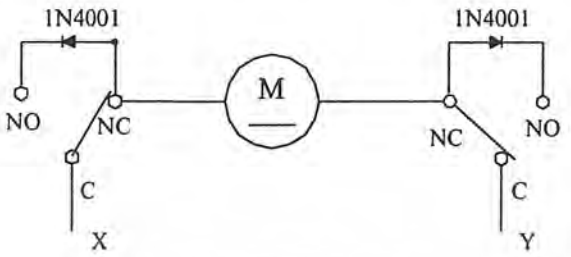
รูปที่ 3.15 วงจรอินพุทแบบอนาล็อกที่ใช้ในโครงงาน

ในการเลือกช่องสัญญาณอินพุทอนาล็อก 8 ช่องนั้นจะใช้เลือกที่ขา 23,24 และ 25 เป็นขาแอดเดรสลอจิก (Address Logic) และที่ขา 22 ใช้ในการควบคุมสัญญาณที่เข้ามาทางอินพุทดังตารางที่ 3.5

ช่องสัญญาณ	ควบคุม	แอดเดรสลอจิก		
	ALE(23)	C(24)	B(25)	A(26)
IN 0	↑	0	0	0
IN 1	↑	0	0	1
IN 2	↑	0	1	0
IN 3	↑	0	1	1
IN 4	↑	1	0	0
IN 5	↑	1	0	1
IN 6	↑	1	1	0
IN 7	↑	1	1	1

ตารางที่ 3.5 แสดง CODE การรับช่องของสัญญาณ

3.2.3 วงจรลิมิตสวิตช์ (Limit Switch)



รูปที่ 3.16 วงจร LIMIT SWITCH

วงจรลิมิตสวิทช์ใช้สำหรับป้องกันการเสียหายแก่อุปกรณ์ทางกล (Mechanics) ในกรณีที่อาจเกิดความผิดพลาดในการทำงานของวงจรในส่วนต่าง ๆ เช่น วงจรขับมอเตอร์ โปรแกรมหรือวงจรส่วนอื่น ๆ ที่เกี่ยวข้อง เมื่อเกิดความผิดพลาดในการทำงานอาจทำให้มอเตอร์หมุนไปในทิศทางใดทิศทางหนึ่งตลอดเวลา วงจรลิมิตสวิทช์จะตัดวงจรมอเตอร์ออกเพื่อหยุดการทำงาน หากไม่มีลิมิตสวิทช์แล้วส่วนที่เคลื่อนไหวจากการทำงานของมอเตอร์อาจชำรุดเสียหายได้ เนื่องจากข้อจำกัดในการเคลื่อนไหวของอุปกรณ์ทางกล ลักษณะการติดตั้งลิมิตสวิตช์นี้ จะติดตั้งในมุมที่มากกว่าสภาวะการใช้งานจริงเล็กน้อยเพื่อไม่ให้เป็นอุปสรรคในการทำงานตามปกติ แต่ถ้าเกิดการผิดพลาดในการทำงานขึ้น ลิมิตสวิทช์ต้องอยู่ในตำแหน่งที่สามารถตัดวงจรได้ทันก่อนเกิดการเสียหายแก่อุปกรณ์ทางกล

3.2.3.1 การทำงานของวงจร

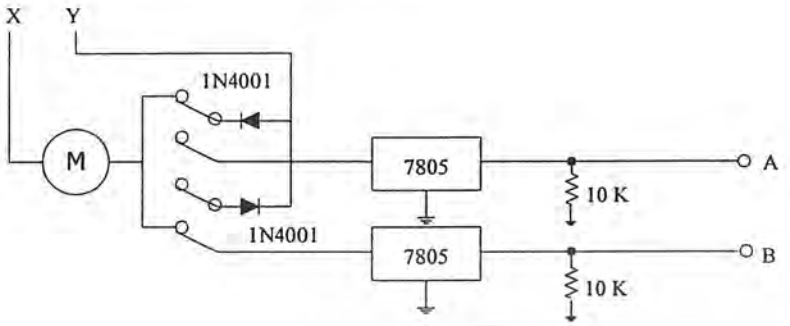
ในสภาวะปกติสวิตช์ 1 และสวิตช์ 2 จะต่ออยู่ที่ตำแหน่ง NC กล่าวคือการทำงานยังทำงานอยู่ในขอบเขตที่กำหนด เมื่อเกิดความผิดพลาดในการทำงานขึ้น สมมติให้มีกระแสไปที่จุด A เป็นบวก และที่จุด B เป็นลบ กระแสจะไหลจากจุด A ผ่านสวิตช์ 1 ที่ขา C และ NC ผ่านมอเตอร์และสวิตช์ 2 และครบวงจรที่จุด B กำหนดให้มอเตอร์หมุนไปทางซ้าย เมื่อมอเตอร์หมุนให้อุปกรณ์ทางกลเคลื่อนที่จนเลยจุดที่ทำงานตามปกติ อุปกรณ์ทางกลก็จะสัมผัสกับสวิตช์ 1 เป็นผลให้สวิตช์ 1 เปลี่ยนสภาวะมาอยู่ในตำแหน่งที่ขา C ต่อกับขา NO ทำให้กระแสไม่ครบวงจรที่ไดโอด (D1) มอเตอร์ก็จะหยุดหมุน อุปกรณ์ทางกลก็จะไม่เกิดความเสียหาย ถึงแม้ว่าสวิตช์ 1 จะตัดวงจรแล้วเราก็ยังสามารถควบคุมมอเตอร์ให้หมุนกลับมาในตำแหน่งที่ถูกต้อง (ตำแหน่งที่ทำงานตามปกติ) ได้โดยป้อนกระแสให้จุด A เป็น – และจุด B เป็น + กระแสจะไหลจากจุด B ผ่านสวิตช์ 2 ที่ขา C

ผ่านขา NC ผ่านมอเตอร์และไดโอด (D1) และสวิตช์ 1 ที่ขา NO ผ่านขา C และครบวงจรที่จุด A ซึ่งมอเตอร์จะหมุนทางขวา เมื่อมอเตอร์หมุนมาทางขวาอุปกรณ์ทางกลก็จะถอยห่างออกจากสวิตช์ 1 ทำให้สวิตช์ 1 เปลี่ยนสภาวะมาอยู่ที่ตำแหน่งที่ขา C ต่ออยู่กับขา NC แต่การเปลี่ยนสภาวะของสวิตช์ 1 นี้จะไม่มีผลต่อการเคลื่อนที่ของกระแส เพราะกระแสยังคงครบวงจรอยู่ ทำให้อุปกรณ์ทางกลเคลื่อนที่เข้ามายังตำแหน่งที่ทำงานตามปกติได้

ในทางกลับกันหากเกิดความผิดพลาดในการทำงานในทิศทางตรงข้ามกับกรณีแรกเราก็สามารถป้องกันการเสียหายแก่อุปกรณ์ทางกลและสามารถควบคุมให้อุปกรณ์ทางกลเคลื่อนที่กลับเข้ามาอยู่ในตำแหน่งที่ทำงานตามปกติได้เช่นกันดังนี้ เมื่อเกิดความผิดพลาดในการทำงานขึ้นในทิศทางตรงข้าม สมมติให้มีกระแสไปที่จุด A เป็น - และที่จุด B เป็น + กระแสจะไหลจากจุด B ผ่านสวิตช์ 2 ที่ขา C และ NC ผ่านมอเตอร์และสวิตช์ 1 และครบวงจรที่จุด A กำหนดให้มอเตอร์หมุนไปทางขวา เมื่อมอเตอร์หมุนให้อุปกรณ์ทางกลเคลื่อนที่จนเลยจุดที่ทำงานตามปกติ อุปกรณ์ทางกลก็จะสัมผัสกับสวิตช์ 2 เป็นผลให้สวิตช์ 2 เปลี่ยนสภาวะมาอยู่ในตำแหน่งที่ขา C ต่อกับขา NO ทำให้กระแสไม่ครบวงจรที่ไดโอด (D2) มอเตอร์ก็จะหยุดหมุน อุปกรณ์ทางกลก็จะไม่เกิดความเสียหาย

ถึงแม้ว่าสวิตช์ 2 จะตัดวงจรแล้วเราก็ยังสามารถควบคุมมอเตอร์ให้หมุนกลับมาในตำแหน่งที่ถูกต้อง (ตำแหน่งที่ทำงานตามปกติ) ได้โดยป้อนกระแสให้จุด A เป็นบวก และจุด B เป็นลบ กระแสจะไหลจากจุด A ผ่านสวิตช์ 1 ที่ขา C ผ่านขา NC ผ่านมอเตอร์และไดโอด (D2) และสวิตช์ 2 ที่ขา NO ผ่านขา C และครบวงจรที่จุด B ซึ่งมอเตอร์จะหมุนทางซ้าย เมื่อมอเตอร์หมุนมาทางซ้ายอุปกรณ์ทางกลก็จะถอยห่างออกจากสวิตช์ 2 ทำให้สวิตช์ 2 เปลี่ยนสภาวะมาอยู่ที่ตำแหน่งที่ขา C ต่ออยู่กับขา NC แต่การเปลี่ยนสภาวะของสวิตช์ 2 นี้จะไม่มีผลต่อการเคลื่อนที่ของกระแส เพราะกระแสยังคงครบวงจรอยู่ ทำให้อุปกรณ์ทางกลเคลื่อนที่เขามายังตำแหน่งที่ทำงานตามปกติได้ ทำให้อุปกรณ์ทางกลเคลื่อนที่เข้ามายังตำแหน่งที่ทำงานตามปกติได้ เช่นเดียวกับการเกิดความผิดพลาดในกรณีแรก

3.2.4. วงจรในส่วนของมือจับชิ้นงาน



รูปที่ 3.17 วงจร มือจับชิ้นงาน

ในส่วนของมือจับนี้มีลิมิตสวิตช์ 2 ตัวเพื่อกำหนดค่าสูงสุดและต่ำสุดในการใช้งานเช่นเดียวกับจุดเคลื่อนไหวอื่นๆ ที่กล่าวมาแล้วข้างต้น แต่ลักษณะการต่อวงจรจะแตกต่างจากจุดอื่นเพราะในส่วนของมือจับชิ้นงานนี้ ต้องต่อวงจรให้สามารถตรวจสอบสภาวะการทำงานว่าอยู่ที่สภาวะใดได้ด้วย ซึ่งสภาวะการทำงานของมือจับชิ้นงานนี้จะมีอยู่ 3 สภาวะที่ต้องตรวจสอบให้ได้คือ สภาวะจับชิ้นงาน สภาวะไม่ได้จับชิ้นงาน และสภาวะที่มือจับชิ้นงานเปิดปากจับชิ้นงานออกมากที่สุด โดยใช้ลิมิตสวิตช์ 2 ตัวเป็นตัวตรวจจับสภาวะการทำงานดังกล่าว และแสดงสภาวะนั้นต่อหน่วยประมวลผลกลางเพื่อใช้ในการตัดสินใจในการทำงาน

ในสภาวะปกติ ไม่ได้จับชิ้นงาน ลิมิตสวิตช์ทั้งสองอยู่ในสภาวะที่ขา C ต่ออยู่กับขา NC ทั้งสองตัว ซึ่งสภาวะนี้ที่จุด A, B จะมีสภาวะเป็น 0,0 ในสภาวะนี้เราสามารถควบคุมให้ปากจับชิ้นงานเคลื่อนที่ไปได้ทั้งสองทาง คือ ควบคุมให้จับชิ้นงานและเปิดปากจับชิ้นงานออกมากที่สุดได้

กำหนดให้จุด X = (+) และจุด Y = (-) จะทำให้มอเตอร์หมุนเพื่อจับชิ้นงาน

กำหนดให้จุด X = (-) และจุด Y = (+) จะทำให้มอเตอร์หมุนเพื่อเปิดปากจับชิ้นงาน

กำหนดให้ลักษณะการป้อนกระแสเป็นดังนี้ จุด X = (+) และจุด Y = (-) และสภาวะเริ่มต้นของจุด A, B คือ 0,0 ทิศทางการไหลของกระแสเป็นดังนี้

กระแสไหลจากจุด X ผ่านมอเตอร์ D2, ลิมิตสวิตช์ 1, ไดโอด D1 และครบวงจรที่จุด Y จากข้อกำหนดข้างต้นทำให้มอเตอร์หมุนเพื่อจับชิ้นงาน เมื่อมือจับชิ้นงานได้จับชิ้นงานแล้วส่วนของอุปกรณ์ทางกลจะผลักให้ลิมิตสวิทช์ 1 เปลี่ยนสภาวะจาก ขา C ต่อกับขา NC ไปเป็น ขา C ต่อกับขา NO จากการเปลี่ยนแปลงของลิมิตสวิตช์ 1 ทำให้กระแสไม่ครบวงจรมอเตอร์ก็จะหยุดหมุน และสภาวะของ A,B ก็จะเปลี่ยนไปเป็น 1,0 ทำให้หน่วยประมวลผลรู้ว่าขณะนี้มือจับได้จับชิ้นงานเรียบ

ร้อยแล้ว จากสภาวะการทำงานข้างต้นเมื่อเราต้องการให้มือจับชิ้นงานปล่อยชิ้นงานเราก็ป้อนกระแสดังนี้ จุด X = (-) และจุด Y = (+) สภาวะเริ่มต้นของจุด A,B ของสภาวะนี้คือ 0,0 ทิศทางการไหลของกระแสเป็นดังนี้

กระแสไหลจากจุด Y ผ่านไดโอด D2, ลิมิตสวิตช์ 2, มอเตอร์ และครบวงจรที่จุด X จากข้อกำหนดข้างต้นทำให้มอเตอร์หมุนเพื่อเปิดปากจับชิ้นงาน เมื่อมือจับชิ้นงานได้เริ่มเปิดปากจับชิ้นงานแล้วส่วนของอุปกรณ์ทางกลจะผลักให้ลิมิตสวิทช์ 1 เปลี่ยนสภาวะจาก ขา C ต่อกับขา NO ไปเป็น ขา C ต่อกับขา NC จากการเปลี่ยนแปลงของลิมิตสวิตช์ 1 จะไม่มีผลต่อการทำงานของวงจรเพราะกระแสเดินทางในเส้นทางของไดโอด D2 เมื่อมือจับชิ้นงานได้เปิดปากจับชิ้นงานจนสุดแล้วส่วนของอุปกรณ์ทางกลจะผลักให้ลิมิตสวิทช์ 2 เปลี่ยนสภาวะจาก ขา C ต่อกับขา NC ไปเป็น ขา C ต่อกับขา NO จากการเปลี่ยนแปลงของลิมิตสวิตช์ 2 ทำให้กระแสไม่ครบวงจรมอเตอร์ก็จะหยุดหมุน และสภาวะของ A,B ก็จะเปลี่ยนไปเป็น 0,1 ทำให้หน่วยประมวลผลรู้ว่าขณะนี้มือจับได้เปิดปากจับชิ้นงานจนสุดแล้ว จากสภาวะการทำงานข้างต้นเมื่อเราต้องการให้มือจับชิ้นงานจับชิ้นงานอีกครั้งเราก็ป้อนกระแสดังนี้ จุด X = (+) และจุด Y = (-) สภาวะเริ่มต้นของจุด A,B ของสภาวะนี้คือ 0,0 ทิศทางการไหลของกระแสเป็นดังนี้

กระแสไหลจากจุด X ผ่านมอเตอร์, ไดโอด D1, ลิมิตสวิตช์ 1 และครบวงจรที่จุด Y จากข้อกำหนดข้างต้นทำให้มอเตอร์หมุนเพื่อจับชิ้นงาน เมื่อมือจับชิ้นงานได้เริ่มเคลื่อนที่เข้าเพื่อจับชิ้นงานส่วนของอุปกรณ์ทางกลจะผลักให้ลิมิตสวิทช์ 2 เปลี่ยนสภาวะจาก ขา C ต่อกับขา NO ไปเป็น ขา C ต่อกับขา NC จากการเปลี่ยนแปลงของลิมิตสวิตช์ 2 นี้ จะไม่มีผลต่อการทำงานของวงจรเพราะกระแสเดินทางในเส้นทางของไดโอด D1 แล้วการทำงานก็จะเป็นเหมือนดังที่ได้อธิบายไปแล้วข้างต้น จากการทำงานดังกล่าวทำให้เราสามารถทราบสภาวะการทำงานของมือจับชิ้นงานผ่านทางจุด A,B ซึ่งเราสามารถนำสภาวะนี้มาใช้ประโยชน์ในการควบคุมผ่านหน่วยประมวลผลกลางได้

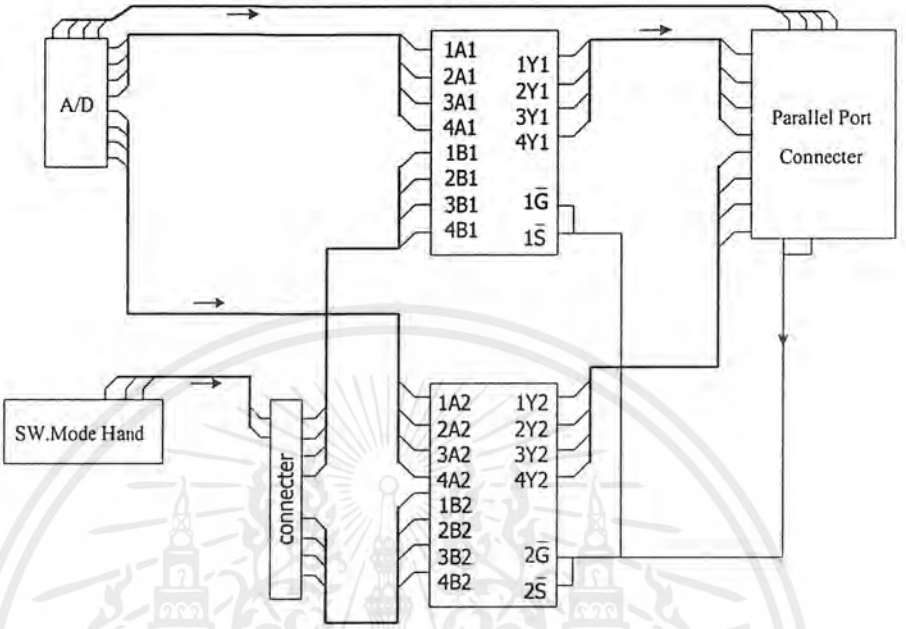
สภาวะที่ตรวจสอบได้มี 3 สภาวะดังนี้

A, B = 0,0 คือสภาวะที่ มือจับไม่ได้จับชิ้นงาน

A, B = 1,0 คือสภาวะที่ มือจับจับชิ้นงาน

A, B = 0,1 คือสภาวะที่ มือจับ เปิดปากจับชิ้นงานจนสุด

3.2.5 วงจรเลือก Connect Parallel



รูปที่ 3.18 แสดงวงจร Connect Parallel

วงจรเลือก Connect Parallel เป็นการเลือกข้อมูลที่จะติดต่อกับ PC โดยผ่านทางพอร์ตขนานดังที่ได้กล่าวไว้แล้วในเรื่องทฤษฎีการติดต่อสื่อสารทางพอร์ตขนาน จะเห็นได้ว่ามีพอร์ตใช้งานอยู่ 3 พอร์ต คือ

1. Data Port ขนาด 8 บิต (D0 – D7) ซึ่งเป็นเอาท์พุทอย่างเดียว
2. Status Port ขนาด 5 บิต ใช้เป็นอินพุทอย่างเดียว
3. Control Port ขนาด 4 บิท ซึ่งเป็นทั้งอินพุทและเอาท์พุท

จากการใช้งานเบื้องต้นของวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลซึ่งมีเอาท์พุทขนาด 8 บิท โดยข้อมูลชุดนี้จะต้องป้อนให้ PC แต่ในทางกลับกัน PC มีพอร์ตอินพุทเพียง 5 บิทเท่านั้น นั่นคือ Status Port ซึ่งยังขาดอีก 3 บิท จะเห็นได้ว่าข้อมูลที่ PC รับนั้นยังไม่ครบถ้วนจึงเป็นข้อมูลที่ผิดพลาดใช้งานไม่ได้ เพราะฉะนั้นที่เหลืออีก 3 บิท จะได้จากการใช้ Control Port ซึ่งเป็นได้ทั้งอินพุทและเอาท์พุทพอร์ตมาใช้งานเป็นอินพุทพอร์ตด้วย ก็จะได้ข้อมูลครบ 8 บิท ส่วนการควบคุมเลือกช่องสัญญาณ (channel) ของวงจร A/D นั้น ในวงจรต้องการใช้ 4 บิท แต่ใน PC มี 8 บิทจึงเพียงพอกับความต้องการใช้งานในการใช้งานของวงจร A/D ในทางปฏิบัติ จะต้องมีอินพุทของ SW Mode Hand เพิ่มเข้ามาด้วยอีกอย่างน้อย 3 บิท แต่พอร์ตอินพุทของ PC ที่ยังไม่ถูกใช้งานมีเหลืออยู่

เพียง 1 บิท ดังนั้นเราจึงได้นำเอาไอซี เบอร์ 74157 ซึ่งเป็นไอซีมัลติเพล็กซ์ (Multiplex) มาใช้งานร่วมด้วยจะทำให้สามารถรับข้อมูลเพิ่มได้อีก 8 บิท ซึ่งเพียงพอที่จะรับ 3 บิทของ Mode Hand ได้ และยังเหลืออีก 5 บิท สำหรับต่ออินพุทหรือเซนเซอร์อย่างอื่นได้อีก

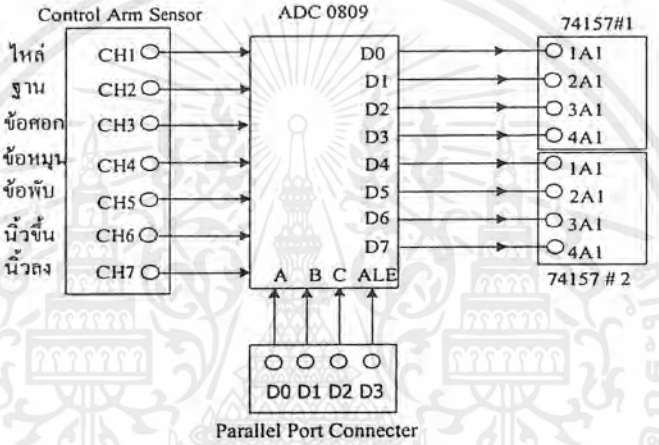
3.2.5.1 ข้อควรระวัง

ในการใช้งาน Control Port ปกติเมื่อเปิดเครื่อง ค่าเริ่มต้นของพอร์ตนี้เป็นการกำหนดให้เขียนเพียงอย่างเดียว (เอาท์พุทพอร์ท) ซึ่งจะเป็นเอาท์พุทแบบคอลเล็คเตอร์เปิด (Open Collector) ซึ่งมี 2 สถานะคือ โลจิก "0" และเปิดวงจร (High Impedance) ซึ่งสิ่งสำคัญในวงจร SW Connect Parallel จะใช้พอร์ตนี้ทำหน้าที่เป็นเป็นพอร์ตอินพุทซึ่งจะส่งค่าให้ PC แต่ใน PC จะกำหนดให้เป็นค่าเอาท์พุทและถ้าหากว่า PC ส่งค่าเอาท์พุทเป็นโลจิก "0" ในขณะเดียวกันวงจรก็ส่งค่าโลจิก "1" ไป ก็จะทำให้เกิดการลัดวงจร (Short Circuit) ขึ้นได้ ทำให้เกิดความเสียหายกับ PC และวงจรได้

โดยปกติการ์ดอินพุท-เอาท์พุท (I/O Card) จะมีตัวต้านทานพูลอัพ (pull up) แต่ก็อาจจะไม่มีทั้งหมด เพื่อให้วงจรทำงานได้อย่างถูกต้องจึงควรมีตัวต้านทานภายนอกด้วย ค่าที่ใช้คือ 4.7 กิโลโอห์ม เมื่อต่อตัวต้านทานแล้วจะทำให้สามารถใช้ 4 บิทของ Control Port เป็นตัวรับส่งข้อมูลในแบบสองทิศทางได้ และที่สำคัญพอร์ตควบคุมจะต้องกำหนดให้มีค่าเป็น xxxx0100 เพื่อให้สามารถอ่านข้อมูลได้ก่อนการใช้งานทุกครั้ง

3.3 การต่อใช้งานวงจรอินพุทอนาล็อก (ของชุดแขนควบคุม)

เป็นวงจรที่ทำหน้าที่แปลงสัญญาณอนาล็อกหรือค่าแรงดันไฟฟ้าที่ได้จากการตรวจจับการเคลื่อนไหวของชุดแขนควบคุม ให้เป็นสัญญาณดิจิตอลที่มีขนาด 8 บิท โดยสัญญาณเอาท์พุทจะต่อเข้ากับ PC ทางพอร์ตขนาน โดยผ่านทางไอซี เบอร์ 74157 เป็นตัวกลาง ส่วนสัญญาณควบคุมช่องอินพุทของสัญญาณอนาล็อกที่จะเข้ามาทำการแปลงเป็นสัญญาณดิจิตอล ซึ่งวงจรอินพุทแบบอนาล็อกจะมีการทำงานคล้ายกับการมัลติเพล็กซ์ โดยสัญญาณนี้จะควบคุมทางพอร์ตขนานของ PC ซึ่งใช้พอร์ตข้อมูลเพียง 4 บิท เท่านั้นคือ D0 – D7

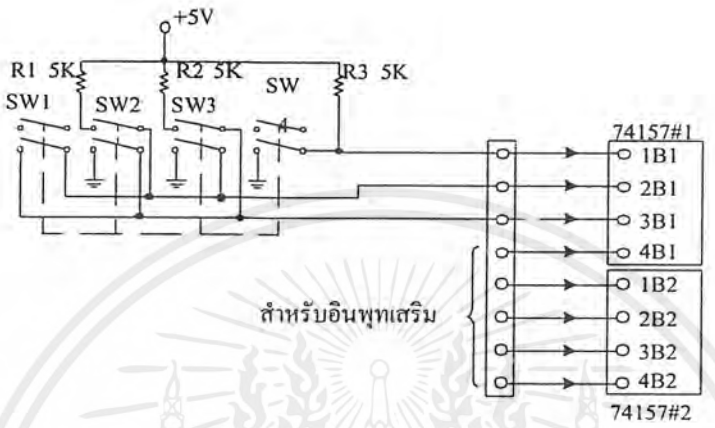


รูปที่ 3.19 การต่อใช้งานวงจรอินพุทอนาล็อก

3.4 การต่อใช้งานวงจร SW. Mode Hand

เนื่องจากที่ชุดแขนควบคุมซึ่งชุดตรวจจับการเคลื่อนไหวที่มือจับโดยปกติจะใช้สวิทช์ 2 ตัว ในการควบคุมการเคลื่อนที่ของชุดนิ้วมือ ซึ่งต่อมาได้มีการเพิ่มเติมส่วนที่เป็นฐานของหุ่นยนต์ โดยการเพิ่มล้อหน้าและล้อหลังเพื่อให้หุ่นยนต์เคลื่อนที่ได้ จึงต้องเพิ่ม SW. Mode Hand เข้าไปเพื่อให้สามารถควบคุมส่วนที่เพิ่มขึ้นได้ โดยชุด SW. Mode Hand ที่เพิ่มเข้ามานี้จะใช้สวิตช์แบบพิเศษ คือ ในสวิตช์ 4 ตัวนั้นสามารถกดให้ ON ได้เพียงตัวเดียวเท่านั้น ตัวที่เหลือจะอยู่ในสภาวะ OFF ลักษณะการทำงาน เมื่อ SW2 อยู่ในสภาวะ ON ก็จะต่อกับกราวน์ซึ่งก็คือแอคทีฟ (Active) “0” และอีกชุดของ SW2 จะต่อกับ +5V ผ่านตัวต้านทานเพื่อป้องกันการลัดวงจร ซึ่งจะไปต่อกับเอาท์พุทของ SW3 เพื่อให้ SW3 มีสัญญาณเป็น “1” และใน SW3 ก็จะทำงานในลักษณะเดียวกัน

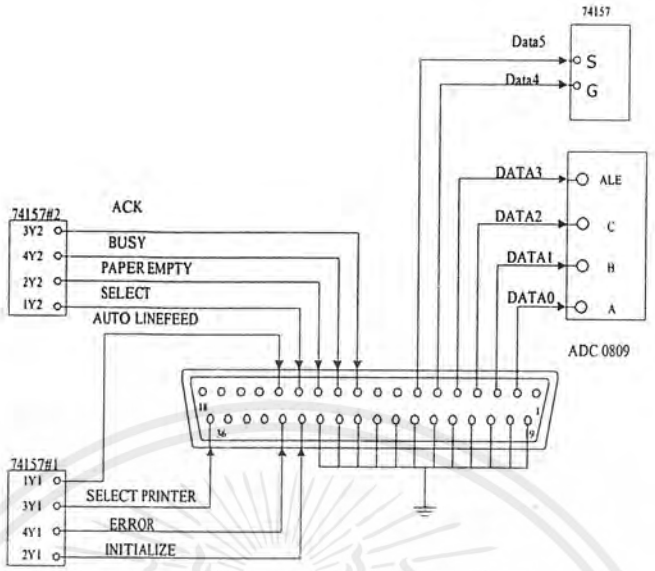
ส่วน SW1 จะทำหน้าที่ต่างจากสวิทช์ตัวอื่นๆ คือใช้เป็นตัวเลือกเมื่อใช้ต่อกับชุดควบคุมชุดเดิมซึ่งจะควบคุมด้วยฮาร์ดแวร์ทั้งหมดโดยจะทำหน้าที่บอกชุดควบคุมชุดเดิมว่าชุดควบคุมพร้อมที่จะทำงานหรือไม่



รูปที่ 3.20 วงจร SW. Mode Hand

3.5 การต่อใช้งานวงจร Connect Parallel Port

เนื่องจากการควบคุมโดยใช้ชุดแขนควบคุมนั้นไม่จำเป็นต้องมีการเคลื่อนที่เหมือนตัวหุ่นยนต์และเวลาใช้งานก็ใช้งานร่วมกับเครื่อง PC ดังนั้นจึงใช้พอร์ตขนานในการติดต่อสื่อสารเนื่องจากพอร์ตขนานนั้นการต่อวงจรด้านฮาร์ดแวร์ไม่ซับซ้อนมากเหมือนการติดต่อทางพอร์ตอนุกรมและทางด้านซอฟแวร์ก็เช่นกัน อีกทั้งความเร็วในการรับส่งข้อมูลเร็วกว่าและสามารถใช้สายเครื่องพิมพ์ได้เลย จึงเป็นการสะดวกในการใช้งาน แต่ตามที่ได้กล่าวไว้ข้างต้นแล้วว่าพอร์ตในการรับส่งข้อมูลของพอร์ตขนานนั้นมีน้อย ซึ่ง Status Port มี 5 บิท และ Control Port มี 4 บิท ซึ่งไม่สามารถใช้พอร์ตใดพอร์ตหนึ่งในการรับส่งข้อมูลได้ ดังนั้นจึงต้องใช้ทั้ง 2 พอร์ตในการรับข้อมูล และแต่ละบิทนั้นจะมีสถานะการทำงานที่ไม่เหมือนกัน เช่น บิท $\overline{\text{ACK}}$ จะแอ็คทีฟ Low ส่วนบิท Busy จะแอ็คทีฟ High เป็นต้น การเชื่อมต่อทางฮาร์ดแวร์แสดงดังรูปที่ 3.21 และตารางที่ 3.6

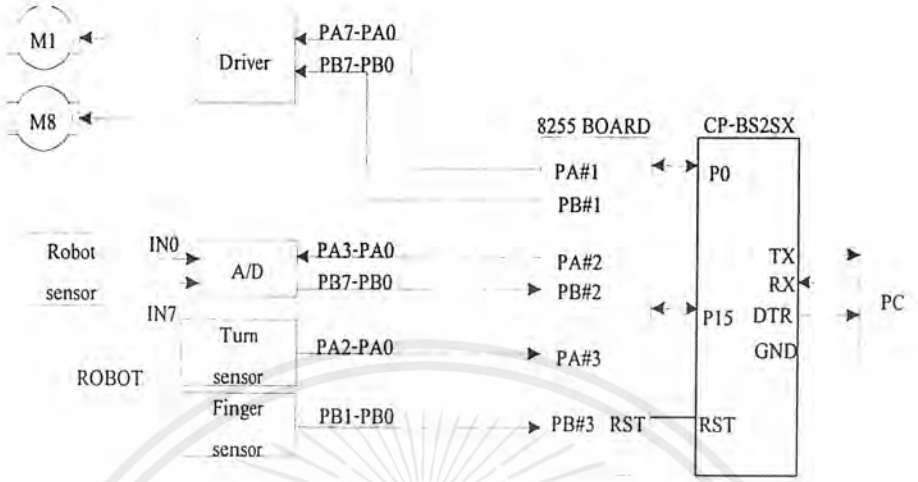


รูปที่ 3.21 วงจร Connect Parallel Port

DEVICES	PIN NO.	CENTRONICS PIN NO.	SPP SIGNAL	DIREC-TION	REGISTER	HARDWARE CONVERT
A/D	A	2	DATA0	OUT	DATA	
	B	3	DATA1	OUT	DATA	
	C	4	DATA2	OUT	DATA	
	ALE	5	DATA3	OUT	DATA	
74157	$\overline{G}$	6	DATA4	OUT	DATA	
	$\overline{S}$	7	DATA5	OUT	DATA	
74157#2	3Y2	10	$\overline{ACK}$	IN	STATUS	
	4Y2	11	BUSY	IN	STATUS	YES
	2Y2	12	PAPER EMPTY	IN	STATUS	
	1Y2	13	SELECT	IN	STATUS	
74157#1	1Y1	14	$\overline{AUTOLINE\_FD}$	IN/OUT	CONTROL	YES
	4Y1	32	ERROR	IN	STATUS	
	2Y1	31	INITIALIZE		CONTROL	
	3Y1	36	SELCT-PRINT		CONTROL	YES

ตารางที่ 3.6 ขาสัญญาณและการใช้งาน

3.6 การใช้งานฮาร์ดแวร์ของบอร์ด CP-BS2SX



รูปที่ 3.22 บล็อกไดอะแกรมการควบคุมหุ่นยนต์

เนื่องจากตัวหุ่นยนต์แขนกลนั้นต้องมีการเคลื่อนที่ ดังนั้นการติดต่อกับ PC จะไม่สะดวกเท่าที่ควรเนื่องจากสายสัญญาณที่ใช้ในการติดต่อนั้นมีด้วยกันหลายเส้น ดังนั้นจึงใช้การติดต่อสื่อสารกับ PC ทางพอร์ตอนุกรม จะเห็นได้ว่าใช้สายติดต่อเพียง 4 เส้นเท่านั้นแต่จะมีความยุ่งยากในเรื่องของวงจรควบคุมมากกว่าการติดต่อทางพอร์ตขนาน ซึ่งใน PC จะมีวงจรรองรับไว้แล้ว ในทางกลับกันคืออีกด้านหนึ่งก็จะต้องมีวงจรที่จะรองรับการสื่อสารข้อมูลเช่นกันและวงจรอาจจะแตกต่างจากใน PC แน่นอนซึ่งอาจจะใช้ไมโครคอนโทรลเลอร์ที่มีการติดต่อสื่อสารตามมา ตรฐาน RS-232 หรืออาจใช้ไอซี สำเร็จรูปที่ทำหน้าที่ควบคุมด้านนี้โดยเฉพาะ ซึ่งในโครงงานนี้ได้เลือกใช้ไมโครคอนโทรลเลอร์ตระกูล PIC รุ่น CP-BS2SX ซึ่งเอื้ออำนวยต่อการใช้งานกับอุปกรณ์อื่นๆ ด้านฮาร์ดแวร์ แต่เนื่องจาก CP-BS2SX นี้มีพอร์ตอินพุท – เอาท์พุทเพียง 2 พอร์ตหรือ 16 บิทเท่านั้น ซึ่งไม่เพียงพอต่อการใช้งานจึงต้องมีการขยายพอร์ตเพิ่มมากขึ้นโดยใช้บอร์ด 8255 รุ่น 72I0Z80 ขยายจาก 1 พอร์ตเป็น 9 พอร์ตซึ่งเพียงพอกับความต้องการ

3.7 การต่อระหว่างบอร์ด CP-BS2SX กับบอร์ด 8255

	PORT OUT H							
8255 BOARD	RD	WR	-	A4	A3	A2	A1	A0
PIN NO.	40	38	-	14	16	18	20	22
CP-B2SX	P15	P14	P13	P12	P11	P10	P9	P8
PIN NO.	16	15	14	13	12	11	10	9

(ก)

	PORT OUT L							
8255 BOARD	D7	D6	D5	D4	D3	D2	D1	D0
PIN NO.	25	19	17	13	15	23	29	27
CP-B2SX	P7	P6	P5	P4	P3	P2	P1	P0
PIN NO.	8	7	6	5	4	3	2	1

(ข)

	OTHER PORT						
8255 BOARD	A7	A6	A5	IORQ	RESET	VCC	GND
PIN NO.	8	10	12	39	30	21	24
CP-B?SX	GND	GND	GND	GND	RESET	VCC	GND
PIN NO.	27	27	27	27	ON BOARD	25	27

(ค)

ตารางที่ 3.7 แสดงการเชื่อมต่อระหว่าง CP-B2SX กับบอร์ด 8255

เนื่องจากบอร์ด 8255 นั้นถูกออกแบบมาเพื่อรองรับกับซีพียู Z-80 โดยมีดาต้าบัสขนาด 8 บิท แอดเดรสบัส 8 บิท และ คอนโทรลบัสอีก 8 บิท และขาอื่นๆ อีกรวมทั้งหมด 40 ขา แต่ไมโครคอนโทรลเลอร์ CP-B2SX นั้นมีพอร์ตอินพุทเอาท์พุทเพียง 2 พอร์ตหรือ 16 บิทเท่านั้น ซึ่งจะเห็นได้ว่าขาสัญญาณที่ใช้งานไม่เพียงพอต่อความต้องการของบอร์ด 8255 เพราะฉะนั้นจึงต้องปรับปรุงขาสัญญาณต่าง ๆ เพื่อให้ทั้งสองวงจรสามารถใช้ร่วมกันได้ดังแสดงไว้ในตารางที่ 3.7

- ขา A7 – A5 เป็นขาสัญญาณสำหรับเลือกแอดเดรสของ Z-80 ซึ่งใน CP-B2SX ไม่จำเป็นต้อง ใช้จึงต่อลงกราวนด์

- ขา $\overline{IORQ}$ เป็นขาสัญญาณที่จะแอคทีฟ "0" เมื่อ Z-80 ต้องการติดต่อกับอินพุทเอาท์พุท แต่ใน CP-B2SX ไม่จำเป็นต้องใช้จึงไม่ได้ถูกใช้งานจึงต่อลงกราวนด์

- ขา RESET ต่อกับขา RESET ของบอร์ด CP-B2SX

- ขา $\overline{RD}$ เป็นขาสัญญาณที่ Z-80 จะแอคทีฟ "0" เมื่อต้องการอ่านข้อมูลซึ่งไอซี 8255 ต้องใช้สัญญาณนี้ ดังนั้นจึงต่อเข้ากับ ขา 15 ของ CP-B2SX ซึ่งเป็นได้ทั้งอินพุทและเอาท์พุทโดยการกำหนดค่าในทางซอฟแวร์

- ขา $\overline{WR}$ จะทำงานคล้ายกับขา $\overline{RD}$ แต่จะแอคทีฟ "0" เมื่อต้องการเขียนข้อมูลออกพอร์ต 8255 ซึ่งต่อกับขา P18

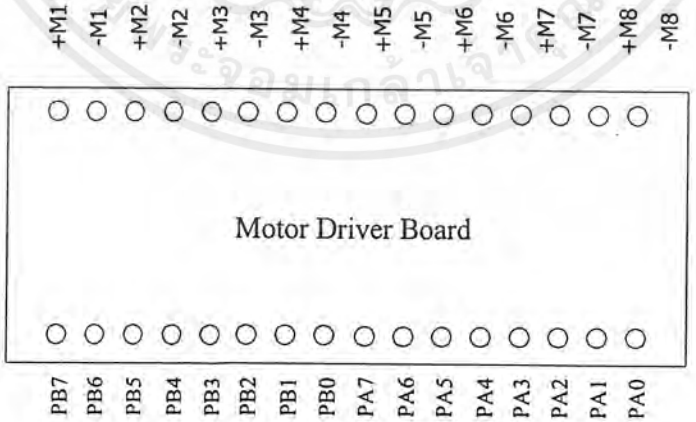
- ขา P13 ของ CP-B2SX สำหรับต่อกับลำโพง (Buzzer)

- ขา A4 – A2 เป็นขาเลือกว่าต้องการติดต่อกับ 8255 ตัวใด เพราะบอร์ด 8255 นี้มีไอซี 8255 3 ตัว ซึ่งต่อกับ P12 – P10 ของ CP-B2SX

- ขา A1 – A0 ใช้สำหรับเลือกพอร์ตของ 8255 (พอร์ต A, B, C) ซึ่งต่อกับ P9 - P8 ของ CP-B2SX

- ขา D7 – D0 เป็นขาสัญญาณข้อมูล ซึ่งต่อกับ P7 - P0 ของ CP-B2SX

3.8 การต่อ บอร์ด 8255 กับภาค DRIVER



รูปที่ 3.23 ภาคไดร์ฟเวอร์

มอเตอร์	การใช้งาน
M0	ขับล้อหลัง
M1	ขับล้อหน้า
M2	ขับฐาน
M3	ขับไหล่
M4	ขับข้อศอก
M5	ข้อพับ
M6	ข้อหมุน
M7	ขับนิ้วจับ

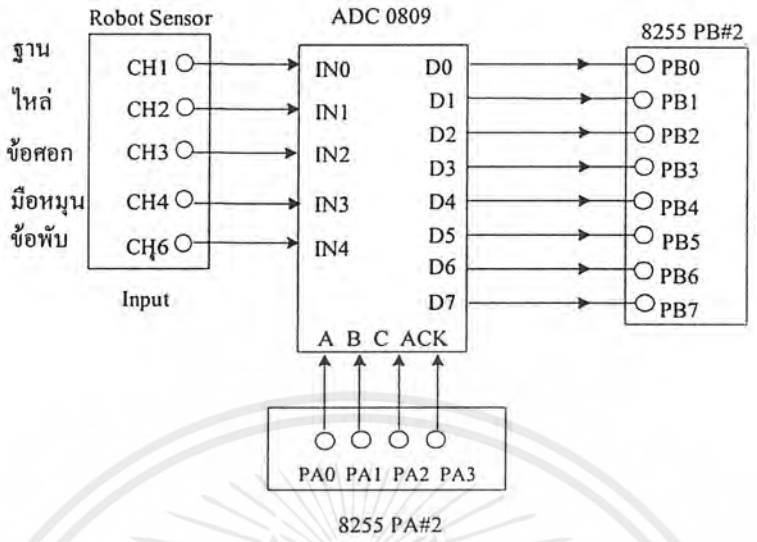
ตารางที่ 3.8 การใช้งานของมอเตอร์แต่ละตัว

ในการสั่งงานให้หุ่นยนต์ทำงานเป็นการสั่งงานโดยเมื่อ PC ประมวลผลเสร็จแล้วก็จะส่งค่าข้อมูลเป็นเลขฐานสิบหกและตรวจสอบข้อมูลที่ส่งมาว่าต้องการติดต่อกับภาค DRIVER ตัวใด เพราะว่าในการรับส่งข้อมูล 8255 สามารถรับส่งได้ครั้งละ 8 บิทเท่านั้น โดยภาค DRIVER สำหรับขับมอเตอร์นั้น DRIVER 1 ใช้พอร์ต A ของบอร์ด 8255 #1 และ DRIVER 2 ใช้พอร์ต B ของ บอร์ด 8255#1 กำหนดเป็นพอร์ตเอาท์พุท

3.9 <u>การต่อวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล</u> (ของชุดหุ่นยนต์)

พอร์ต B ของบอร์ด 8255#2 กำหนดเป็นพอร์ตอินพุทซึ่งจะใช้เก็บข้อมูลที่ได้จากการแปลงสัญญาณอนาล็อกที่ได้จากการตรวจจับความเคลื่อนไหวของตัวหุ่นยนต์ ให้เป็นสัญญาณดิจิตอลที่มีขนาด 8 บิท เพื่อที่จะรับข้อมูลและส่งให้แก่ CP-BS2SX

พอร์ต A ของบอร์ด 8255 #2 โดยจะใช้เพียงแค่ 4 บิทเท่านั้นคือบิท A0 – A3 ซึ่งจะถูกกำหนดให้เป็นพอร์ตเอาท์พุทเพื่อใช้ควบคุมช่องอินพุทของสัญญาณอนาล็อกที่จะทำการแปลงเป็นสัญญาณดิจิตอลซึ่งวงจรอินพุทแบบอนาล็อกนี้จะมีการทำงานคล้ายกับการมัลติเพล็กซ์

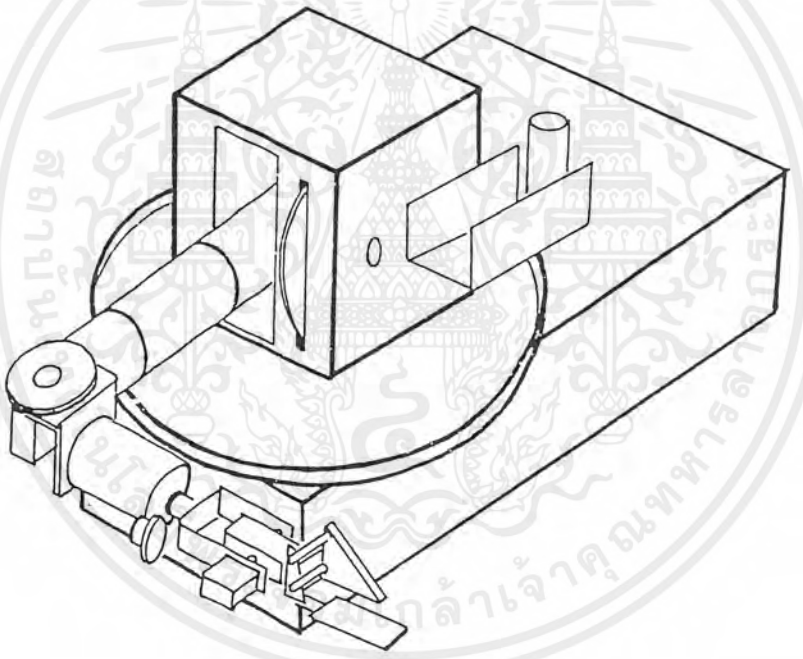


รูปที่ 3.24 การต่อใช้งานวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล

บทที่ 4
การออกแบบระบบกลไกของหุ่นยนต์แขนกล

4.1 ส่วนประกอบของแขนกล

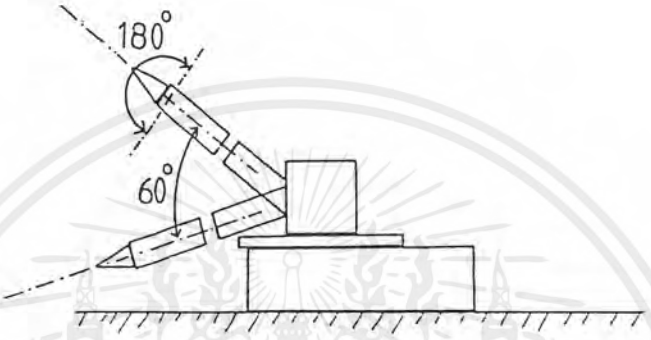
การออกแบบสร้างระบบกลไกของชุดแขนกลสำหรับ โครงงานนี้ได้วางแผนการสร้างชุดข้อมือ (มือจับ, ชุดพับขอมือ) , ชุดบิดข้อมือ, ชุดข้อศอก, ชุดหัวไหล่, โครงฐาน ซึ่งส่วนประกอบดังกล่าวมีลักษณะดังรูป 4.1



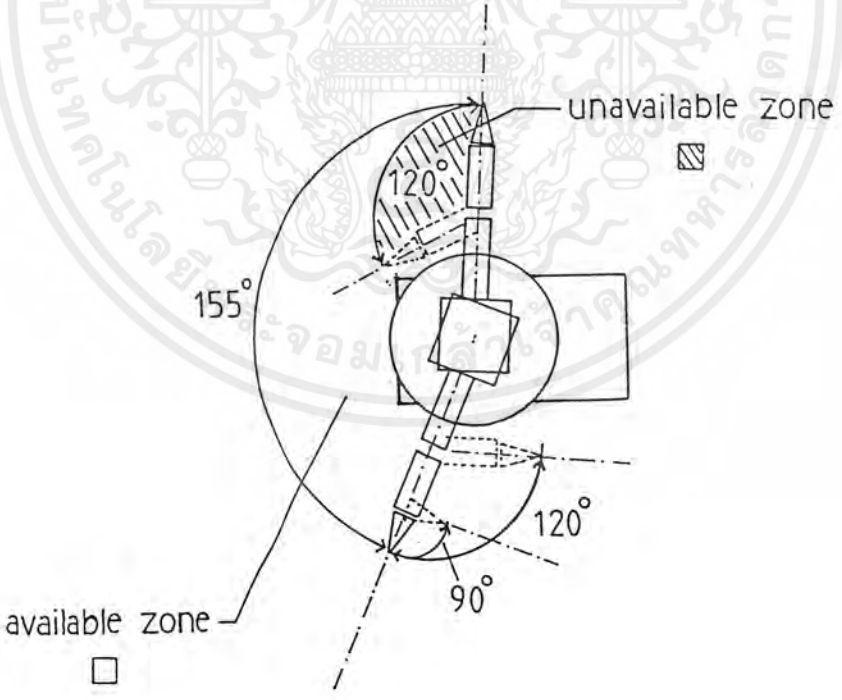
รูปที่ 4.1 ส่วนประกอบของแขนกล

4.2 ลักษณะการเคลื่อนไหวของหุ่นยนต์แขนกล

ลักษณะการเคลื่อนไหวของชุดบิดข้อมือนั้นสามารถบิดหมุนได้ถึง 180 องศา และของชุดหัวไหล่นั้นสามารถเคลื่อนที่ได้ในแนวตั้งได้ 60 องศา ดังแสดงดังรูป 4.2 ส่วนชุดของโครงฐานนั้นมีลักษณะการเคลื่อนไหวในทางแนวนอน 155 องศา และชุดข้อศอกนั้นสามารถพับและกางออกสุดได้ในช่วง 120 องศา ดังแสดงดังรูป 4.3

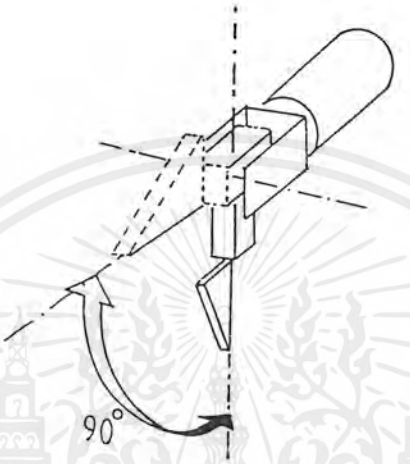


รูปที่ 4.2 ลักษณะการเคลื่อนไหวของชุดข้อบิดหมุนและชุดหัวไหล่

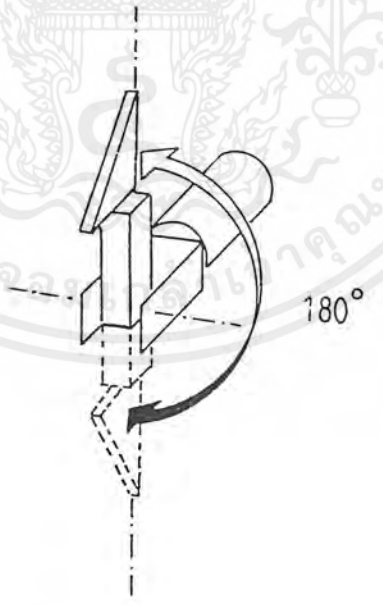


รูปที่ 4.3 ลักษณะการเคลื่อนไหวของชุดข้อศอกและชุดโครงฐาน

ส่วนลักษณะการเคลื่อนไหวของส่วนพับข้อมือและส่วนบิดข้อมือนั้น ดังแสดงดังรูป 4.4 และ รูป 4.5 นั้น ในรูป 4.4 ลักษณะการเคลื่อนไหวของส่วนพับข้อมือ นั้นมีช่วงในการเคลื่อนที่ 90 องศา ส่วนข้อบิดหมุนนั้นสามารถหมุนได้อยู่ในช่วง 180 องศา ดังแสดงในรูป 4.5



รูปที่ 4.4 การเคลื่อนไหวของส่วนพับข้อมือ

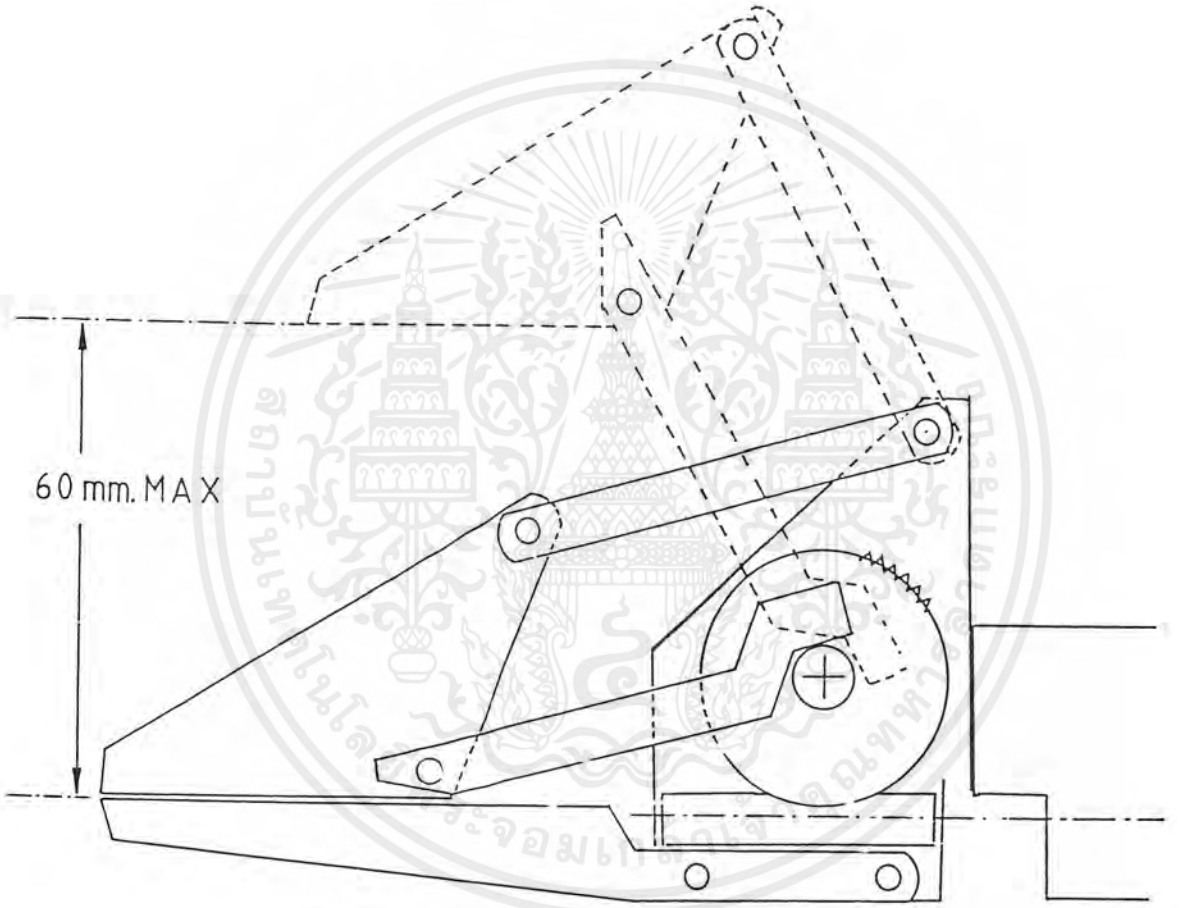


รูปที่ 4.5 การเคลื่อนไหวของส่วนบิดข้อมือ

4.3 ลักษณะเฉพาะของส่วนประกอบต่างๆ

4.3.1 ส่วนประกอบชุดมือจับชิ้นงาน

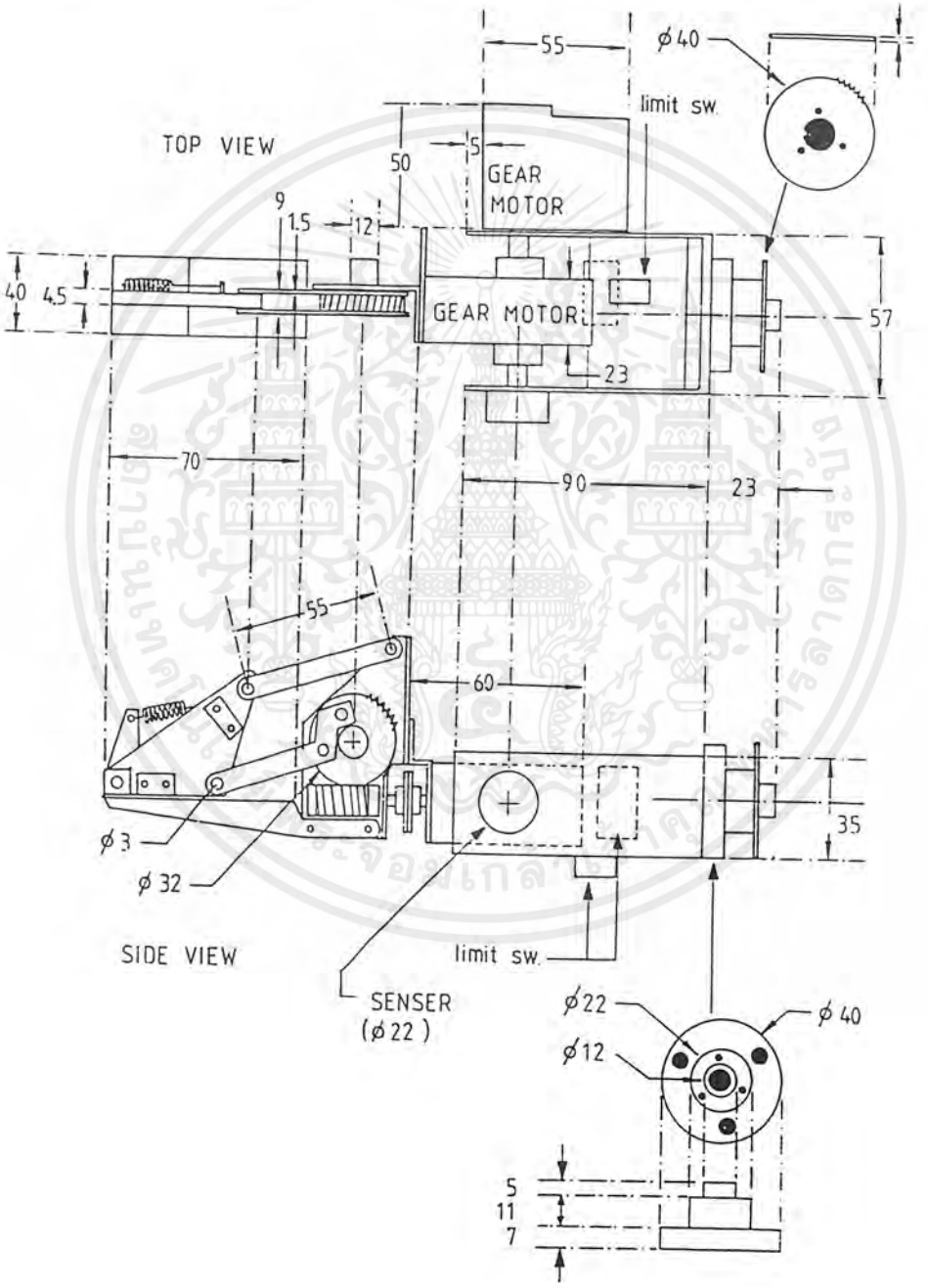
ในส่วนนี้มีการออกแบบที่มีลักษณะเป็นแบบมือหนีบ (Clamp) โดยสามารถกางนิ้ววออกได้สูงสุด 60 มิลลิเมตร ดังแสดงในรูป 4.6



รูปที่ 4.6 มือจับชิ้นงาน

4.3.2 ส่วนประกอบเฉพาะของชุดมือจับมือจับชิ้นงานและชุดพับข้อมือ

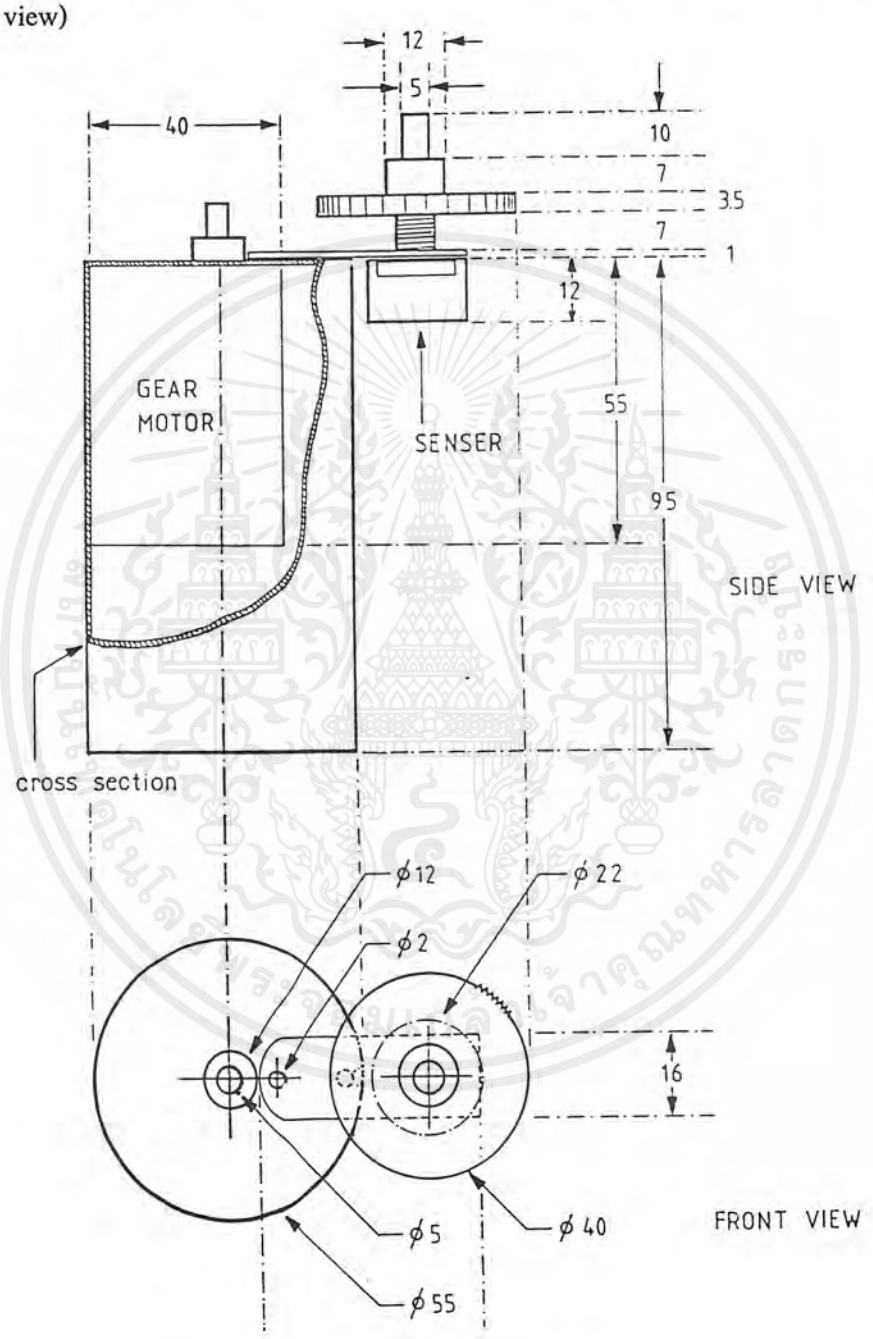
ในรูป 4.7 เป็นการแสดงถึงลักษณะการออกแบบทางกลไกและการประกอบของชุดมือจับชิ้นงานและชุดพับข้อมือ ทั้งในการสังเกตจากด้านบน (Top view) และการสังเกตจากด้านข้าง (Side view) ในโครงงานนี้จะใช้มอเตอร์ไฟฟ้ากระแสตรง ชนิดมีเฟืองทด (Motor Gear) เป็นส่วนใหญ่ ซึ่งมอเตอร์ชนิดนี้หมุนได้เฉพาะเมื่อมีการป้อนกระแสไฟฟ้าให้เท่านั้น



รูป 4.7 การประกอบของชุดมือจับชิ้นงานและชุดพับข้อมือ

4.3.3 ส่วนประกอบเฉพาะของส่วนบิดข้อมือ

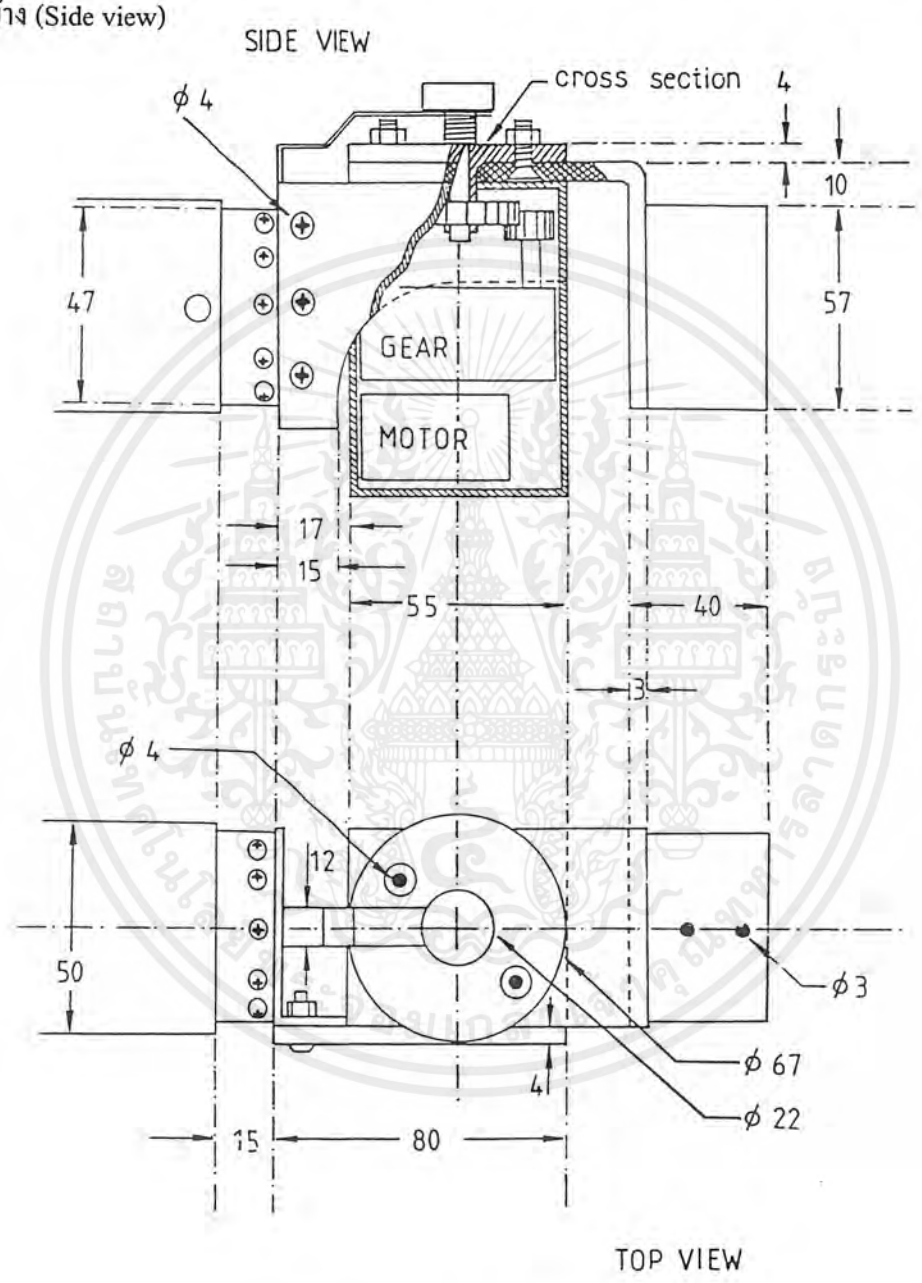
ในรูป 4.8 เป็นการแสดงถึงลักษณะการออกแบบทางกลไกและการประกอบติดตั้งมอเตอร์และเฟืองของชุดบิดข้อมือ ทั้งในการสังเกตจากด้านบน (Top view) และการสังเกตจากด้านข้าง (Side view)



รูปที่ 4.8 ส่วนประกอบของชุดบิดข้อมือ

4.3.4 ส่วนประกอบเฉพาะของชุดพับข้อศอก

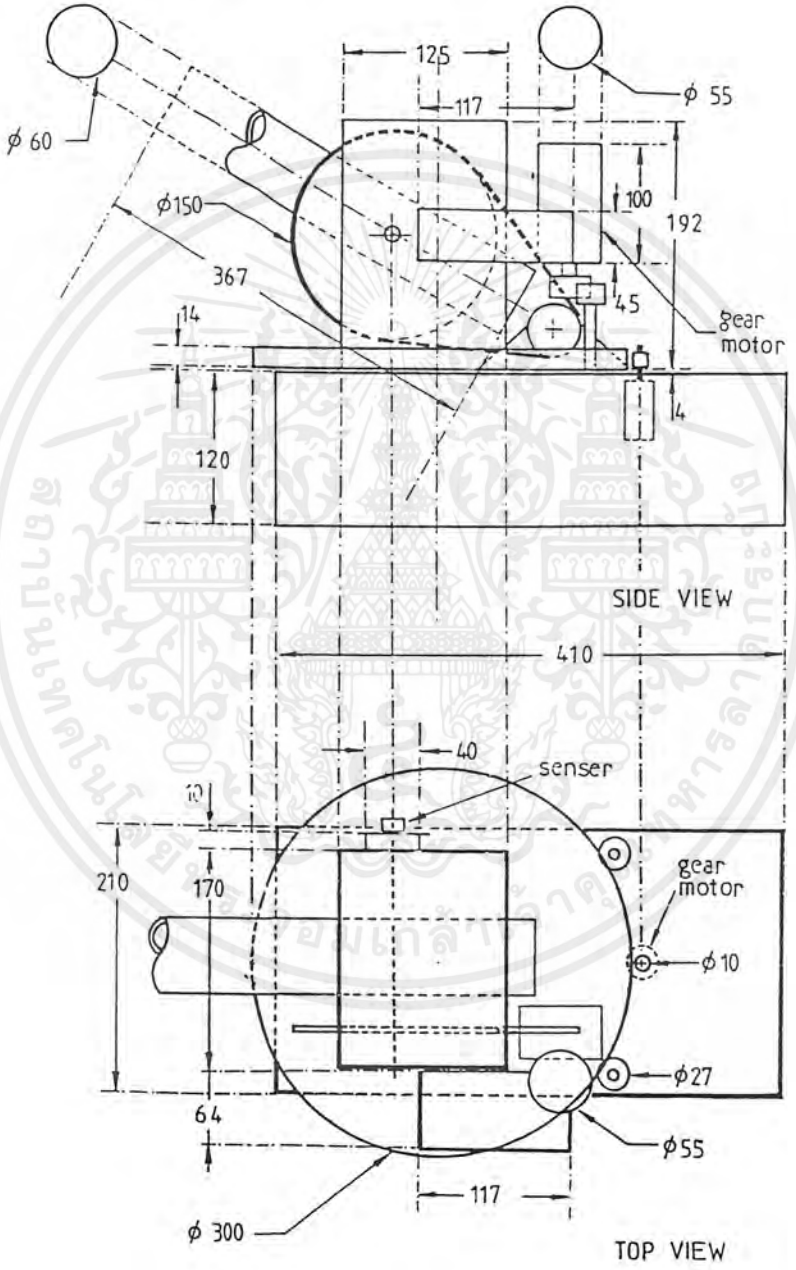
ในรูป 4.9 เป็นการแสดงถึงลักษณะการออกแบบทางกลไกและการประกอบติดตั้งมอเตอร์และเฟืองของชุดพับข้อศอก ทั้งในการสังเกตจากด้านบน (Top view) และการสังเกตจากด้านข้าง (Side view)



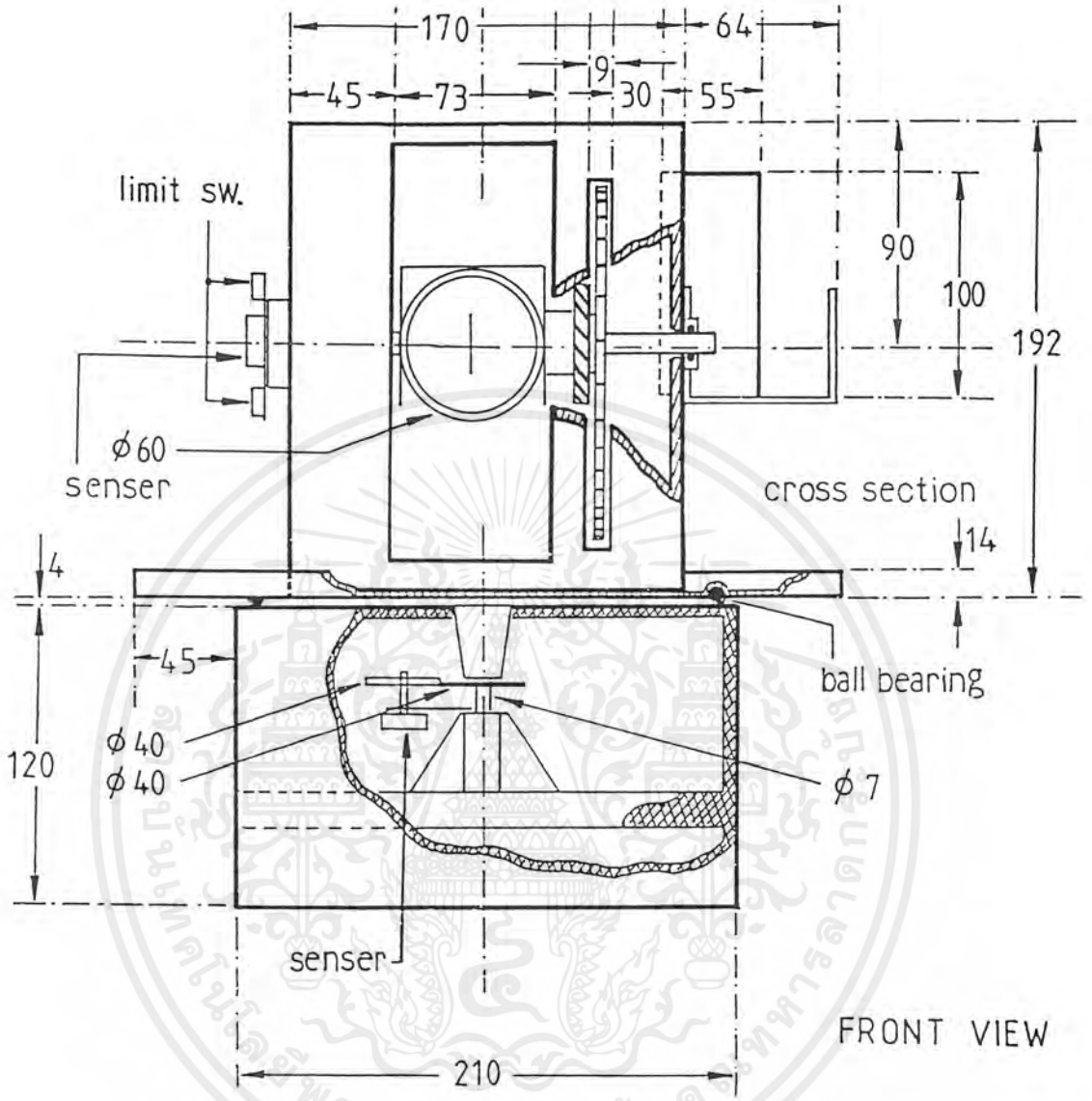
รูปที่ 4.9 ส่วนประกอบของชุดพับข้อศอก

4.3.5 ส่วนประกอบเฉพาะของชุดหัวไหล่แท่นหมุนและฐาน

ในรูป 4.10 เป็นการแสดงถึงลักษณะการออกแบบทางกลไกและการประกอบติดตั้งมอเตอร์และเฟืองของชุดหัวไหล่แท่นหมุนและฐาน ทั้งในการสังเกตจากด้านบน (Top view) และการสังเกตจากด้านข้าง (Side view)



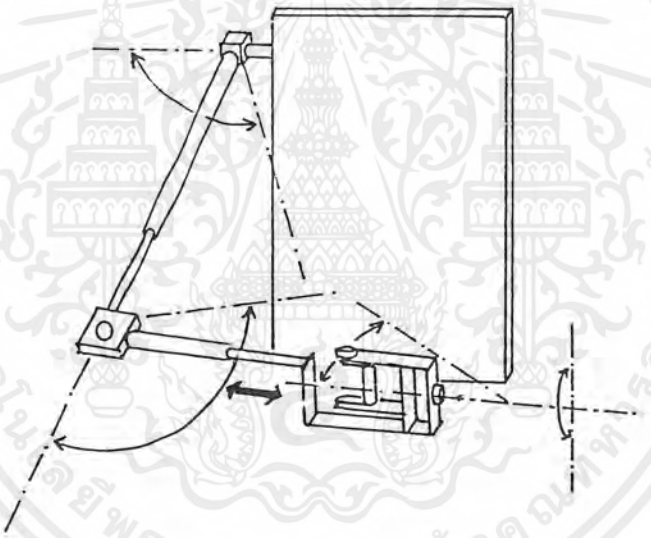
รูปที่ 4.10 ส่วนประกอบของชุดหัวไหล่แท่นหมุน



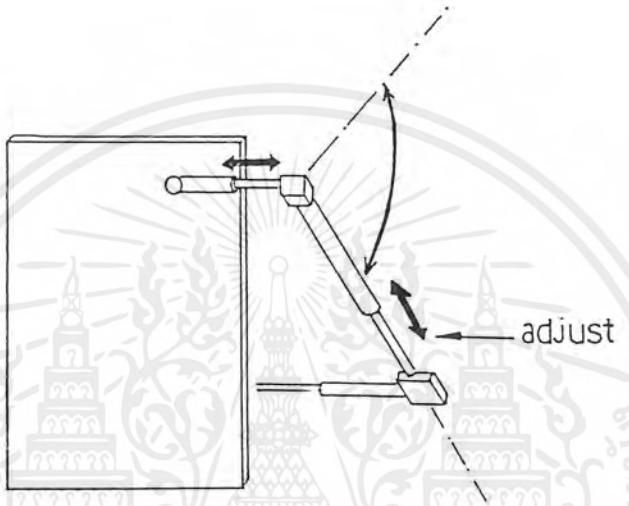
รูปที่ 4.11 ส่วนประกอบของชุดฐาน

4.4 ส่วนประกอบของชุดแขนควบคุม

เป็นส่วนประกอบของชุดแขนควบคุมที่สำหรับติดตั้งตัวตรวจจับความเคลื่อนไหวของผู้ควบคุม โดยเวลาใช้งานนั้นจะสวม (สะพาย) ไว้ด้านหลังของผู้ควบคุม ดังรูปที่ 4.12 นั้นเป็นลักษณะทางด้านหน้า และรูปที่ 4.13 นั้นเป็นลักษณะทางด้านหลัง โดยจากรูปทั้ง 2 จะเห็นได้ว่าช่วงของการเคลื่อนที่ของชุดแขนควบคุมนั้นจะมีช่วงมุมหรือองศาในการเคลื่อนที่ที่เท่ากันกับช่วงการเคลื่อนที่ของหุ่นยนต์แขนกลดังที่ได้กล่าวไว้ข้างต้น และชุดแขนควบคุมยังออกแบบมาเพื่อให้สามารถปรับ (Adjust) ช่วงขนาดความยาวของแขนและศอก และความกว้างของช่วงหัวไหล่ได้ เพื่อความสะดวกในการใช้งานและสามารถใช้กับผู้ควบคุมคนอื่นได้ด้วย



รูปที่ 4.12 ลักษณะทางด้านหน้าของชุดแขนควบคุม

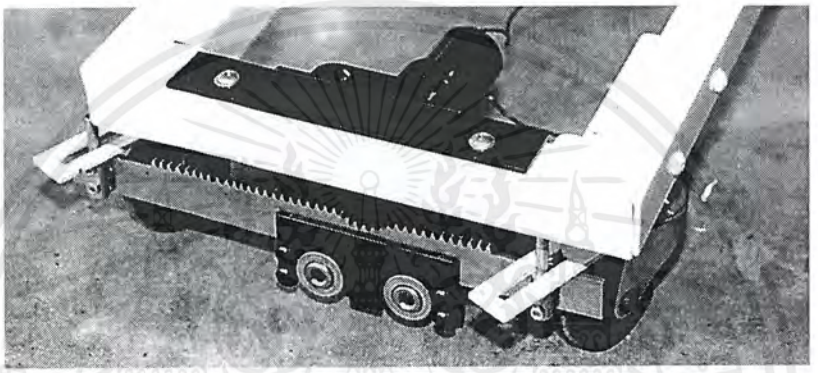


รูปที่ 4.13 ลักษณะทางด้านหลังของชุดแขนควบคุม

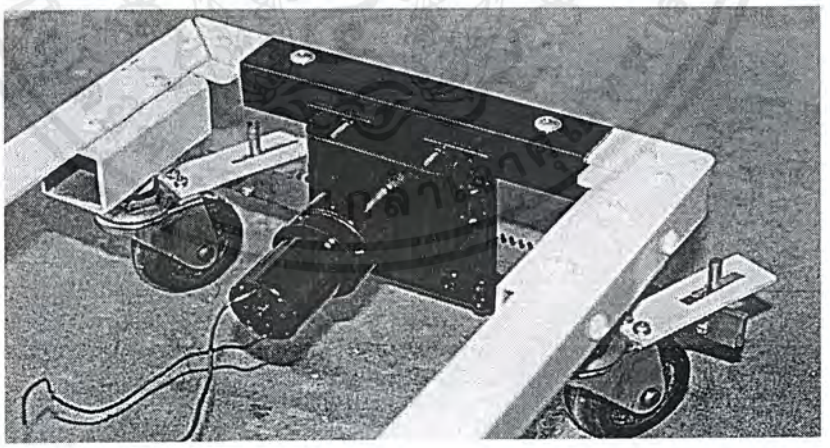
4.5 ส่วนประกอบของชุดควบคุมการเคลื่อนที่ของหุ่นยนต์แขนกล

4.5.1 ชุดควบคุมการเคลื่อนที่ของล้อหน้า

ในส่วนนี้จะเป็นการควบคุมการเลี้ยวโดยลักษณะการเลี้ยวจะคล้ายกับการเลี้ยวของรถยนต์ ซึ่งจะมีแกนเพลาหน้าเป็นตัวบังคับเลี้ยว แต่ในโครงงานนี้จะใช้เฟืองสะพานแทนในรูปที่ 4.14 และรูปที่ 4.15 เป็นการแสดงลักษณะทางด้านหน้าและหลังของชุดควบคุมการเคลื่อนที่ของล้อหน้า



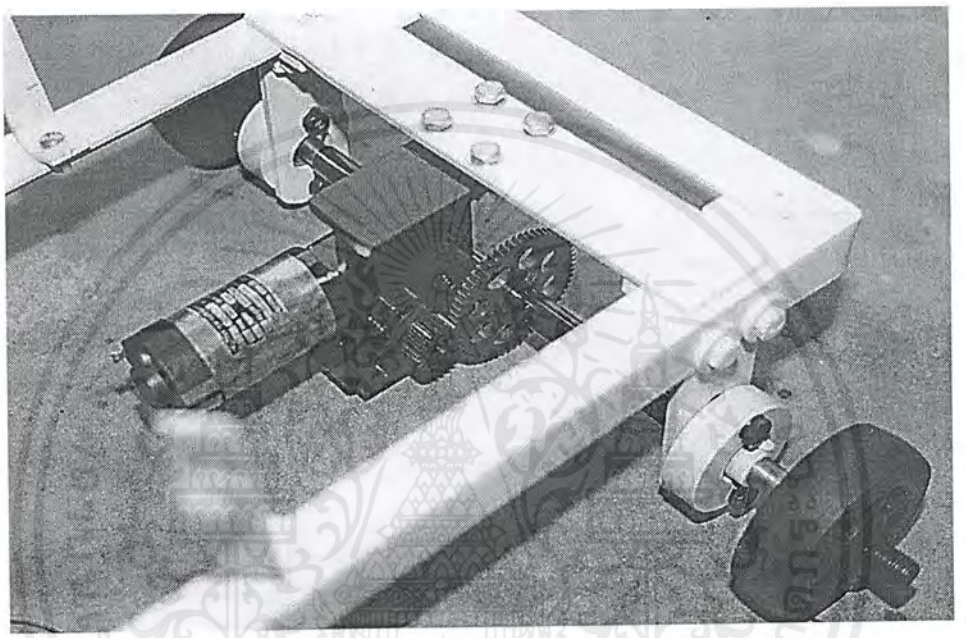
รูปที่ 4.14 ชุดควบคุมการเคลื่อนที่ของล้อหน้า (ลักษณะทางด้านหน้า)



รูปที่ 4.15 ชุดควบคุมการเคลื่อนที่ของล้อหน้า (ลักษณะทางด้านหลัง)

4.5.2 ชุดควบคุมการเคลื่อนที่ของล้อหลัง

ในส่วนนี้จะเป็นการควบคุมการเดินหน้าและถอยหลัง โดยลักษณะการทำงานจะคล้ายกับการทำงานของรถยนต์ ซึ่งจะมีแกนเพลาหลังและเฟืองท้ายที่ใช้ในการขับเคลื่อน ในรูปที่ 4.16 เป็นการแสดงลักษณะการประกอบของชุดควบคุมการเคลื่อนที่ของล้อหลัง



รูปที่ 4.16 ชุดควบคุมการเคลื่อนที่ของล้อหลัง

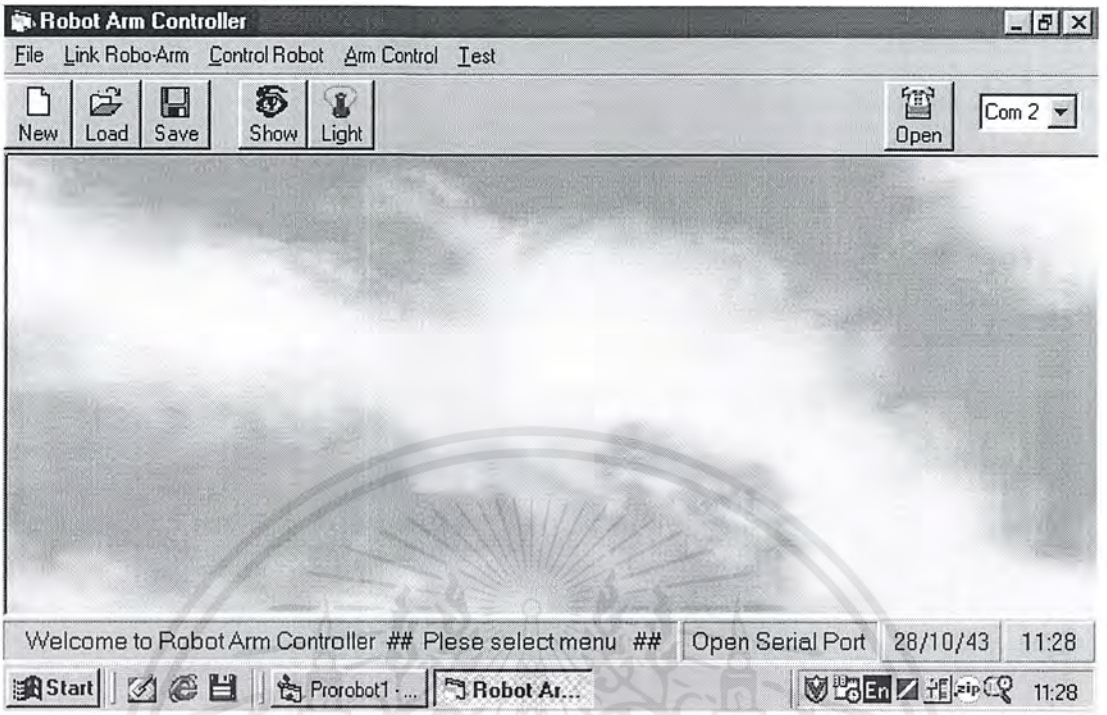
บทที่ 5
โปรแกรมที่ใช้ควบคุมหุ่นยนต์

การควบคุมการทำงานของหุ่นยนต์นี้ได้ใช้ไมโครคอนโทรลเลอร์ช่วยในการควบคุมการทำงานด้วย ดังนั้นจึงสามารถแยกโปรแกรมออกเป็น 2 ส่วนใหญ่ ๆ คือ โปรแกรมของไมโครคอนโทรลเลอร์ซึ่งเป็นซีพียู Basic Stamp 2SX และโปรแกรมสำหรับคอมพิวเตอร์ ซึ่งในโครงงานนี้ได้นำโปรแกรม Visual Basic มาใช้งาน

5.1 โปรแกรมควบคุมหลัก

5.1.1 โปรแกรมเมนูหลัก เป็นโปรแกรมที่ใช้ควบคุมโปรแกรมย่อยต่าง ๆ

รูปที่ 5.1 โฟลว์ชาร์ตของโปรแกรมเมนูหลัก



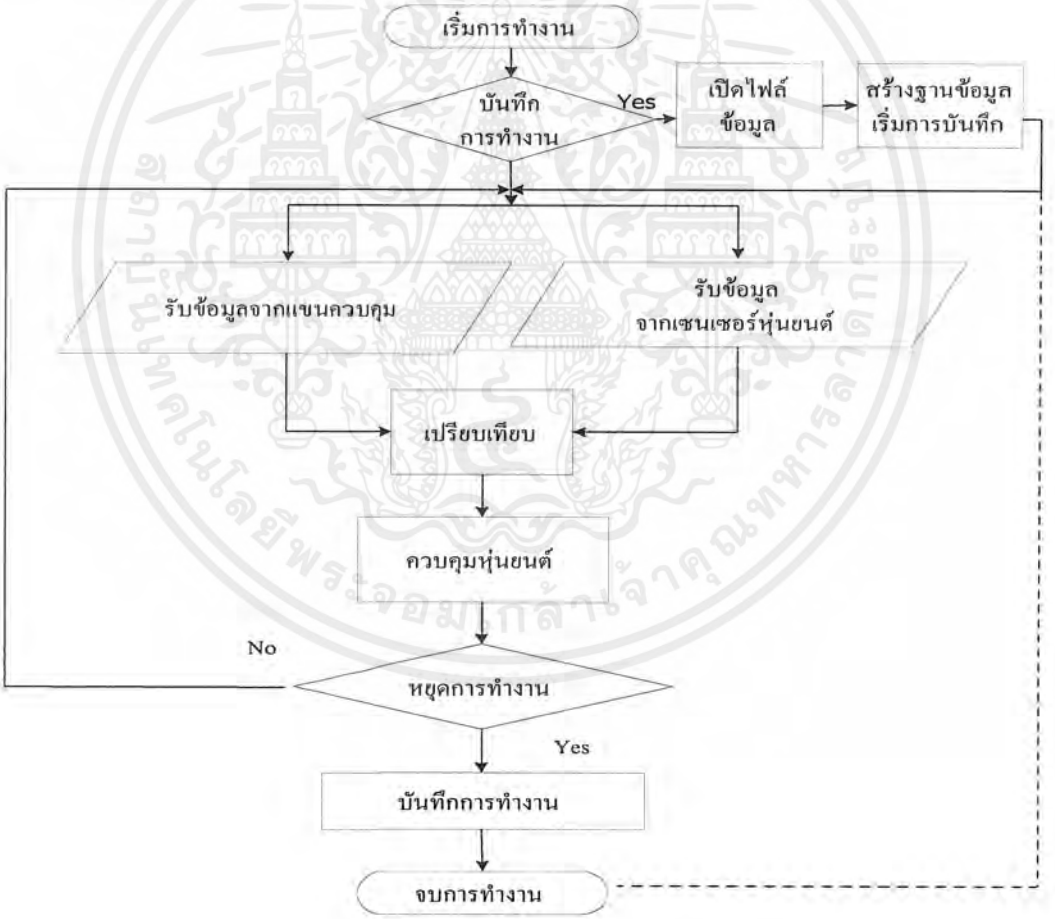
รูปที่ 5.2 โปรแกรมเมนูหลัก

จากรูปที่ 5.2 ลักษณะการเลือกการทำงานมีดังนี้

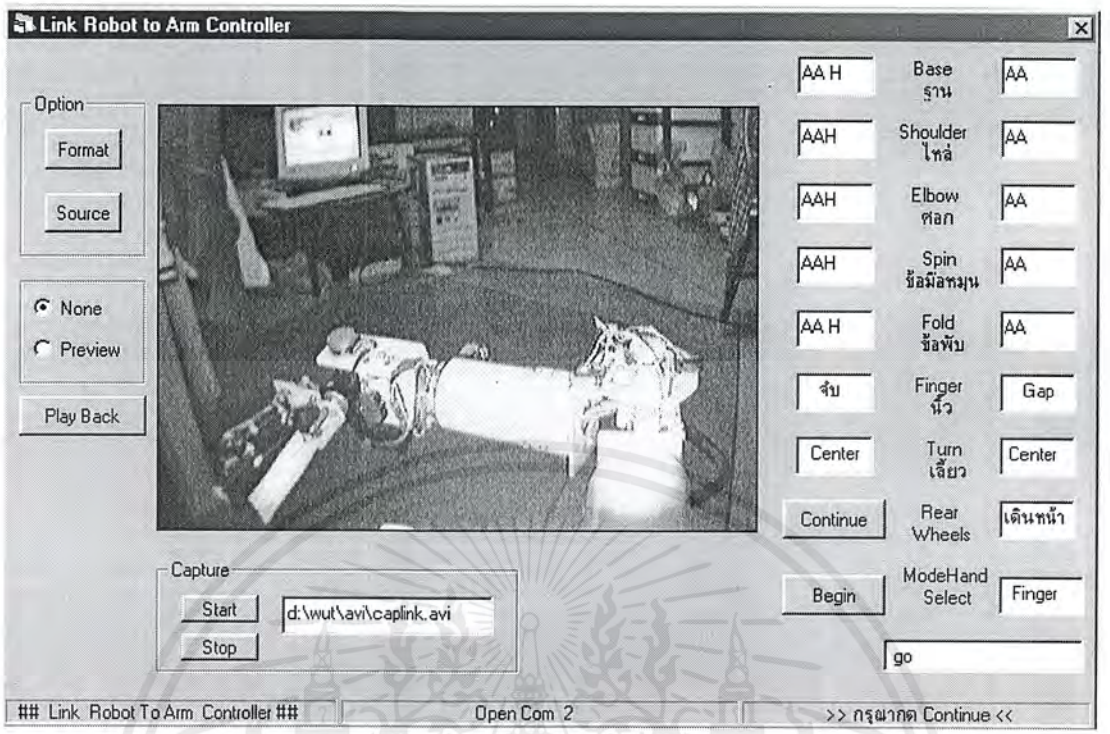
- Link Robot-Arm คือการเลือกการทำงานของโปรแกรมควบคุมการทำงานแบบอัตโนมัติ
- Control Robot คือการเลือกการทำงานของโปรแกรมชุดควบคุมแบบ Manual
- Test คือการเลือกการทำงานของโปรแกรมทดสอบการทำงานของหุ่นยนต์แขนกล
- ปุ่ม New คือ การสร้างฐานข้อมูลใหม่
- ปุ่ม Save คือ การบันทึกการทำงานของหุ่นยนต์แขนกล
- ปุ่ม Load คือ การนำข้อมูลในฐานข้อมูลมาใช้งาน
- ปุ่ม Light คือ การเปิดไฟ เมื่อกดแล้วปุ่มนี้ก็จะเปลี่ยนลักษณะการทำงานเป็นปุ่ม Off Lamp เพื่อทำการปิดไฟ
- ปุ่ม Open คือ การเปิดพอร์ตอนุกรม เมื่อกดแล้วปุ่มนี้ก็จะเปลี่ยนลักษณะการทำงานเป็นปุ่ม Close Port เพื่อทำการปิดพอร์ตเมื่อเลิกใช้งาน
- ช่องตัวเลือก เพื่อเลือกการเปิดพอร์ตว่า เปิดพอร์ตคอม 1 หรือ คอม 2

5.1.2 โปรแกรมชุดควบคุมอัตโนมัติ

เป็นโปรแกรมควบคุมการทำงานแบบอัตโนมัติ คือมีการติดต่อควบคุมหุ่นยนต์โดยสั่งงานทางแขนควบคุม เมื่อเริ่มโปรแกรมจะต้องเปิดพอร์ตอนุกรมก่อนแล้วจึงเริ่มต้นการทำงาน จากนั้นโปรแกรมได้ทำการรับค่าจากวงจร A/D ของชุดควบคุมทางพอร์ตขนานและรับค่าจาก A/D ของชุด robot sensor ทางพอร์ตอนุกรม จากนั้นจะนำเอาค่าที่ได้ทั้งสองฝั่งมาเปรียบเทียบกัน ผลคือเมื่อมีการเคลื่อนที่ของชุดควบคุมไป ตัวหุ่นยนต์ก็จะทำการเคลื่อนตามในทิศทางเดียวกัน และยังมีส่วนควบคุมอีกส่วน คือส่วนของ switch mode เนื่องจากชุดแขนควบคุมได้มี สวิตช์ควบคุมนิ้วเพียงคู่เดียว ดั้งนั้นเพื่อตัวเลือกให้สวิตช์คู่นี้ควบคุมการทำงานแขนได้ทั้งนิ้ว ล้อขับเคลื่อนและล้อเลี้ยว และการบันทึก เมื่อต้องการบันทึกก็จะทำการสร้างฐานข้อมูลใหม่ขึ้นมาและทำการบันทึกลักษณะการทำงาน



รูปที่ 5.3 โฟลว์ชาร์ตของโปรแกรมแบบอัตโนมัติ



รูปที่ 5.4 โปรแกรมการทำงานแบบอัตโนมัติ

ขั้นตอนในการทำงาน ในตอนแรกหลังจากโหลดโปรแกรมควบคุมการทำงานแบบอัตโนมัติขึ้นมาแล้วก็ต้องทำการเปิดพอร์ต โดยกดปุ่มเปิดพอร์ตตามที่ได้อธิบายไว้ในโปรแกรมเมนูหลัก จากนั้นก็กดปุ่ม Begin เพื่อส่งค่าเริ่มต้นไปบอกไมโครคอนโทรลเลอร์ว่าพร้อมที่จะเริ่มต้นทำงานจากนั้น กดปุ่ม Continue เพื่อเริ่มการทำงาน แล้วปุ่มนี้ก็จะเปลี่ยนลักษณะการทำงานเป็นปุ่ม Pause จากนี้หุ่นยนต์แขนกลก็จะเคลื่อนไหวตามการเคลื่อนไหวของชุดควบคุม เมื่อต้องการหยุดการทำงานชั่วคราวก็กดปุ่ม Pause

ส่วนของการแสดงภาพ ลักษณะการเลือกการทำงานมีดังนี้

- None และ Preview เป็นเลือกว่าจะแสดงภาพหรือไม่

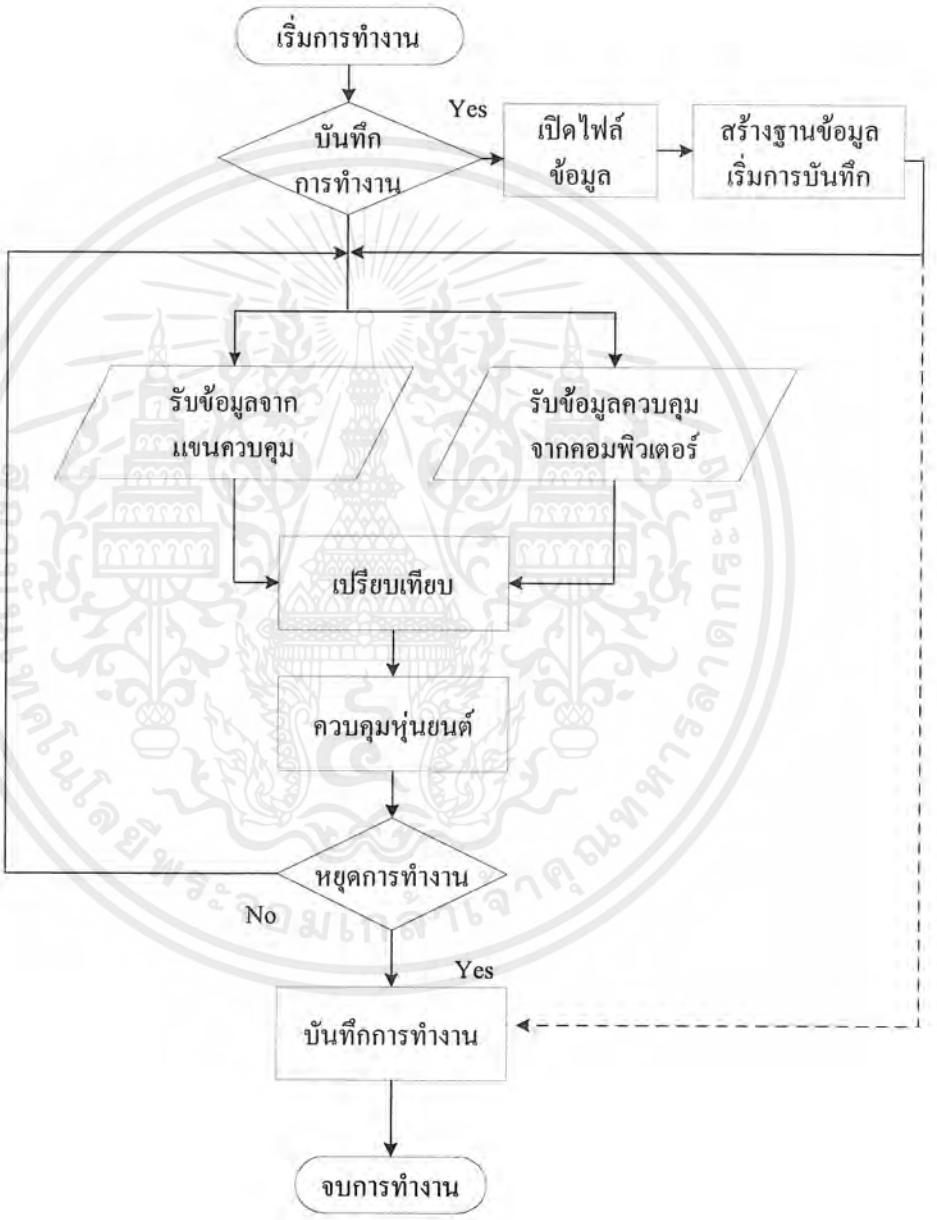
- Option เป็นการปรับแต่งภาพ โดยปุ่ม Format เป็นการปรับขนาดของภาพที่จะแสดง และใช้เลือกมาตรฐานของกล้องวีดีโอ ส่วนปุ่ม Source เป็นการปรับแสงและความคมชัดของภาพที่แสดง

- Capture เป็นการบันทึกภาพ โดยปุ่ม Start เป็นการเริ่มบันทึก และปุ่ม Stop เป็นการเริ่มหยุดการบันทึก ในการบันทึกนี้จะบันทึกเป็นไฟล์ภาพ นามสกุล AVI

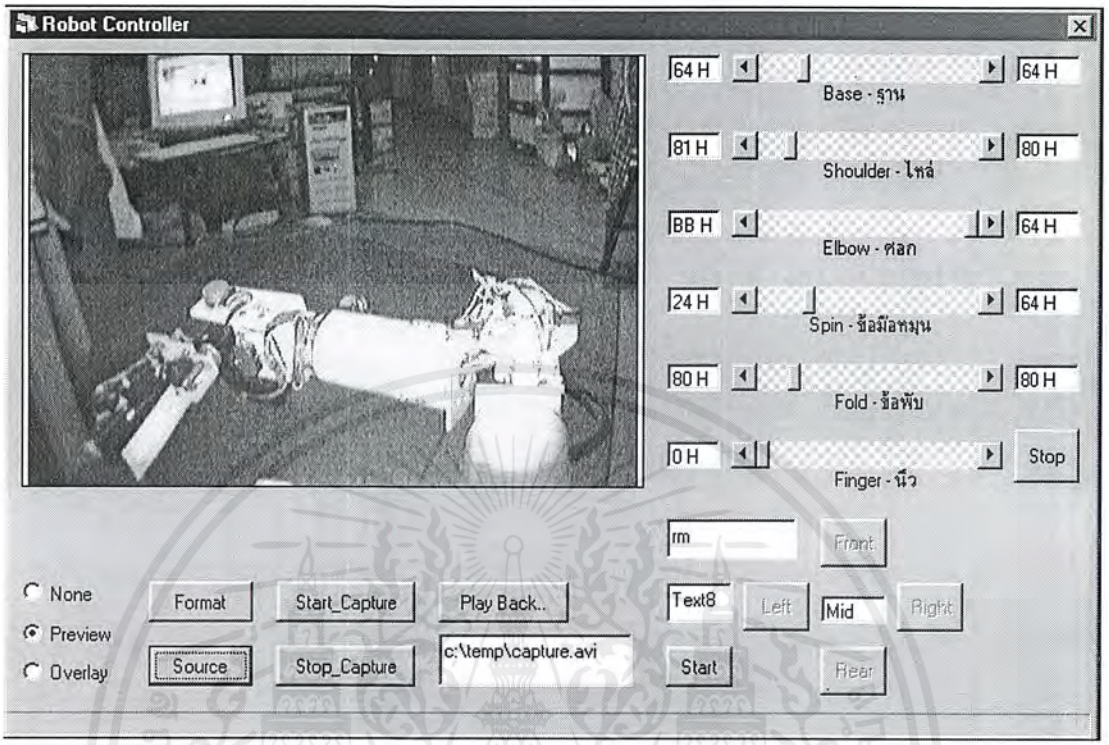
- Play Back คือการเล่นไฟล์ภาพ นามสกุล AVI ที่ได้บันทึกไว้

5.1.3 โปรแกรมชุดควบคุมแบบ Manual

โปรแกรมชุดนี้จะทำการควบคุมหุ่นยนต์โดยการควบคุมด้วยคอมพิวเตอร์โดยตรง เมื่อต้อการให้หุ่นยนต์ทำงานก็สามารถควบคุมได้จากจอมอนิเตอร์ โดยมีการแยกสั่งให้มอเตอร์แต่ละตัวเคลื่อนที่โดยอิสระจากชุดควบคุมการเคลื่อนไหว



รูปที่ 5.5 แสดงโฟลว์ชาร์ตการทำงานแบบ Manual



รูปที่ 5.6 โปรแกรมชุดควบคุมแบบ Manual

ขั้นตอนในการทำงาน ในตอนแรกหลังจากโหลดโปรแกรมควบคุมการทำงานแบบ Manual ขึ้นมาแล้วก็ต้องทำการเปิดพอร์ต โดยกดปุ่มเปิดพอร์ตตามที่ได้อธิบายไว้ในโปรแกรมเมนูหลัก จากนั้นก็กดปุ่ม Start เพื่อส่งค่าเริ่มต้นไปบอกไมโครคอนโทรลเลอร์ว่าพร้อมที่จะเริ่มต้นทำงาน จากนั้นก็สามารถทำงานได้โดยสั่งงานโดยการเลื่อนสกอร์บาร์ (Scholl bar) และควบคุมการเคลื่อนที่โดยการกดปุ่ม Front คือเดินหน้า Rear คือถอยหลัง Left คือเลี้ยวซ้าย Right คือเลี้ยวขวา ส่วนการแสดงภาพ ลักษณะการเลือกการทำงานเหมือนกันกับในโปรแกรมการทำงานแบบอัตโนมัติ

5.1.4 โปรแกรมทดสอบการทำงาน

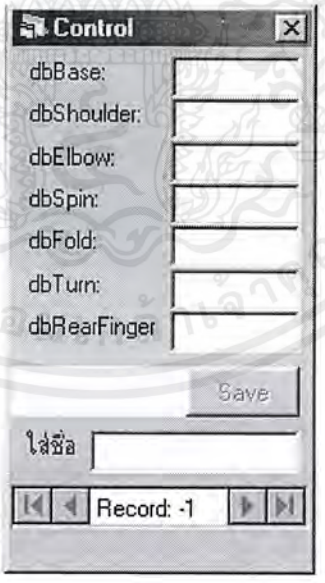
เป็นโปรแกรมเพื่อใช้ทดสอบการทำงานของชุดขับมอเตอร์ และทดสอบการทำงานของไมโครคอนโทรลเลอร์ โดยสามารถแยกสั่งมอเตอร์แต่ละตัว ให้เคลื่อนที่โดยอิสระจากชุดตรวจจับการเคลื่อนไหว

5.1.5 โปรแกรมบันทึกการทำงาน

เป็นการบันทึกการทำงานทุกๆ ลักษณะการเคลื่อนไหวของตัวหุ่นยนต์โดยละเอียด โดยจะเป็นการเก็บไว้ในรูปแบบของฐานข้อมูล (Database) ซึ่งแถวและขนาดของไฟล์ที่จะเก็บจะขึ้นอยู่เวลาที่ทำการเก็บข้อมูล ซึ่งในโครงงานนี้กำหนดให้เก็บข้อมูลทุกๆ 500 มิลลิวินาที กล่าวคือในเวลา 500 มิลลิวินาที จะเก็บข้อมูลครั้ง

ขั้นตอนการทำงาน คือหลังจากเปิดโปรแกรมชุดควบคุมแบบ Manual หรือโปรแกรมควบคุมการทำงานแบบอัตโนมัติ แล้วก็เลือกกดปุ่ม Save ที่เมนูหลัก จากนั้นโปรแกรมก็จะเปิดหน้าต่าง (Form) ขึ้นมาและเมื่อต้องการบันทึกก็ ให้ใส่ชื่อไฟล์ที่จะทำการบันทึกลงในช่อง ใส่ชื่อ จากนั้นก็กดปุ่ม Save ที่หน้าต่างของการบันทึกดังรูปที่ 5.7 แล้วปุ่มนี้ก็จะเปลี่ยนลักษณะการทำงานเป็นปุ่ม Stop จากนี้โปรมแกรมก็จะเริ่มทำการบันทึก จนกว่าจะมีการกดปุ่ม Stop

ในการโหลดไฟล์ที่ทำการบันทึกแล้วก็เช่นกัน หลังจากเปิดโปรแกรมชุดควบคุมแบบ Manual หรือโปรแกรมควบคุมการทำงานแบบอัตโนมัติ แล้วก็เลือกกดปุ่ม Load ที่เมนูหลัก จากนั้นโปรแกรมก็จะเปิดหน้าต่าง (Form) ขึ้นมา (ซึ่งในขณะนี้ปุ่ม Save ก็จะเปลี่ยนลักษณะการทำงานเป็นปุ่ม Load) และเมื่อต้องการโหลดไฟล์ ก็กดปุ่ม Load จากนี้หุ่นยนต์แขนกลก็จะทำงานตามลักษณะการเคลื่อนไหวเดิมที่ได้บันทึกไว้



รูปที่ 5.7 โปรแกรมบันทึกข้อมูล

5.2 โปรแกรมควบคุมภายในหุ่นยนต์แขนกล

เป็นโปรแกรมที่ใช้ในการรับข้อมูลจากคอมพิวเตอร์ แล้วมาประมวลผลว่าคอมพิวเตอร์ต้องการทำอะไร แล้วก็ควบคุมการทำงานของหุ่นยนต์ ควบคุมภาคขับมอเตอร์โดยผ่านทางพอร์ต (ขยาย) 8255 และควบคุมการรับข้อมูลจากชุดตรวจจับความเคลื่อนไหว และชุดเซ็นเซอร์อื่นๆ แล้วส่งข้อมูลที่ได้กลับให้แก่คอมพิวเตอร์ เพื่อที่คอมพิวเตอร์จะได้นำไปประมวลผลต่อไป

รูปที่ 5.8 โฟลว์ชาร์ตของโปรแกรมควบคุมภายในหุ่นยนต์แขนกล

บทที่ 6
สรุปและวิจารณ์

โครงงานนี้ได้แบ่งการทำงานออกเป็น 3 ส่วนหลักๆ คือ

1. การทำงานของส่วนควบคุมภายในหุ่นยนต์แขนกล ได้นำเอาชิพในตระกูล พิก (PIC) รุ่นเบสิกแสตมป์ 2 เอสเอ็กซ์ มาใช้งาน ซึ่งชิพตัวนี้ มีพอร์ตที่เป็นได้ทั้งอินพุทและเอาท์พุทและมีพอร์ตใช้งานเพียง 2 พอร์ทเท่านั้น แต่ในโครงงานนี้ต้องการพอร์ทใช้งานถึง 4 พอร์ท ฉนั้นจะต้องมีการขยายพอร์ต และในโครงงานนี้ได้นำไอซีเบอร์ 8255 มาใช้งาน ซึ่งไอซี 8255 นี้มีการใช้งานที่อิงกับชิพตระกูล Z80 เป็นส่วนใหญ่ เช่น ขาใช้งานซึ่งไอซี 8255 ต้องการ ขาสัญญาณอ่านและสัญญาณในการเขียน และสัญญาณเลือกชิพ ซึ่งซีพียู เบสิกแสตมป์ ไม่มีขาสัญญาณเหล่านี้ จึงต้องมีการปรับปรุงดัดแปลงให้สามารถทำงานร่วมกันได้ และการใช้งานร่วมกันนั้นซีพียู เบสิกแสตมป์ มีความถี่ใช้งานถึง 50 MHz ดังนั้นในการสั่งงาน ไอซี 8255 ต้องมีการชะลอความเร็ว (Delay) ด้วย

2. การทำงานของส่วนควบคุมหลัก คือการทำงานในส่วนของโปรแกรมในคอมพิวเตอร์ ซึ่งโครงงานนี้ใช้โปรแกรมวิชวลเบสิกมาใช้ แต่โปรแกรมนี้ได้ออกแบบมาเพื่อเอื้ออำนวยต่อการใช้งานด้านฐานข้อมูลเป็นส่วนใหญ่ ดังนั้นในการเขียนโปรแกรมควบคุมทางด้านไฟฟ้า เช่นการติดต่อกับพอร์ตสื่อสารนั้นโดยเฉพาะการติดต่อกับพอร์ทขนานนั้นจะไม่มีคำสั่งที่สามารถอ้างถึงบิต จึงต้องมีคอมโพเนนท์ (Component) หรือคำสั่งพิเศษเพิ่มเติมเพื่อให้สามารถใช้งานได้

3. การทำงานของชุดควบคุม ซึ่งได้ใช้พอร์ตขนานในการเชื่อมต่อ โดยต้องมีการสั่งงานจากคอมพิวเตอร์ เป็นเอาท์พุทอย่างน้อย 6 บิตและอินพุท 8 บิท แต่พอร์ตขนานของคอมพิวเตอร์นั้นมีพอร์ตดาต้า 8 บิต ซึ่งเป็นเอาท์พุท ,พอร์ตสถานะซึ่งเป็นอินพุท 5 บิท และพอร์ตคอนโทรลซึ่งเป็นทั้งอินพุทและเอาท์พุท 4 บิท จะเห็นได้ว่าไม่มีพอร์ตใดที่มีอินพุทถึง 8 บิตเลย ดังนั้นจึงต้องมีการดัดแปลงโดยเอาพอร์ตสถานะและพอร์ตคอนโทรลมาทำงานร่วมกันเพื่อให้สามารถใช้เป็นอินพุทขนาด 8 บิท

การทำงานด้านกลไก ปัญหาในการทำงานที่เกี่ยวกับกลไกก็คือ ความรู้ความชำนาญและความพร้อมในด้านเครื่องมืออุปกรณ์และเครื่องจักรต่างๆ ที่จำเป็นต้องใช้งาน เช่นเครื่องกลึง และอื่นๆ

การแปลงสัญญาณอนาล็อกเป็นดิจิตอล ซึ่งการตรวจจับการเคลื่อนไหวของชุดควบคุมและของหุ่นยนต์แขนกลนั้นจะเป็นสัญญาณอนาล็อก โดยการที่จะนำมาประมวลผลนั้นจะต้องแปลงเป็นสัญญาณดิจิตอลเสียก่อน แต่ในการแปลงสัญญาณนั้นจะมีความผิดพลาดและความไม่เที่ยงตรงของ

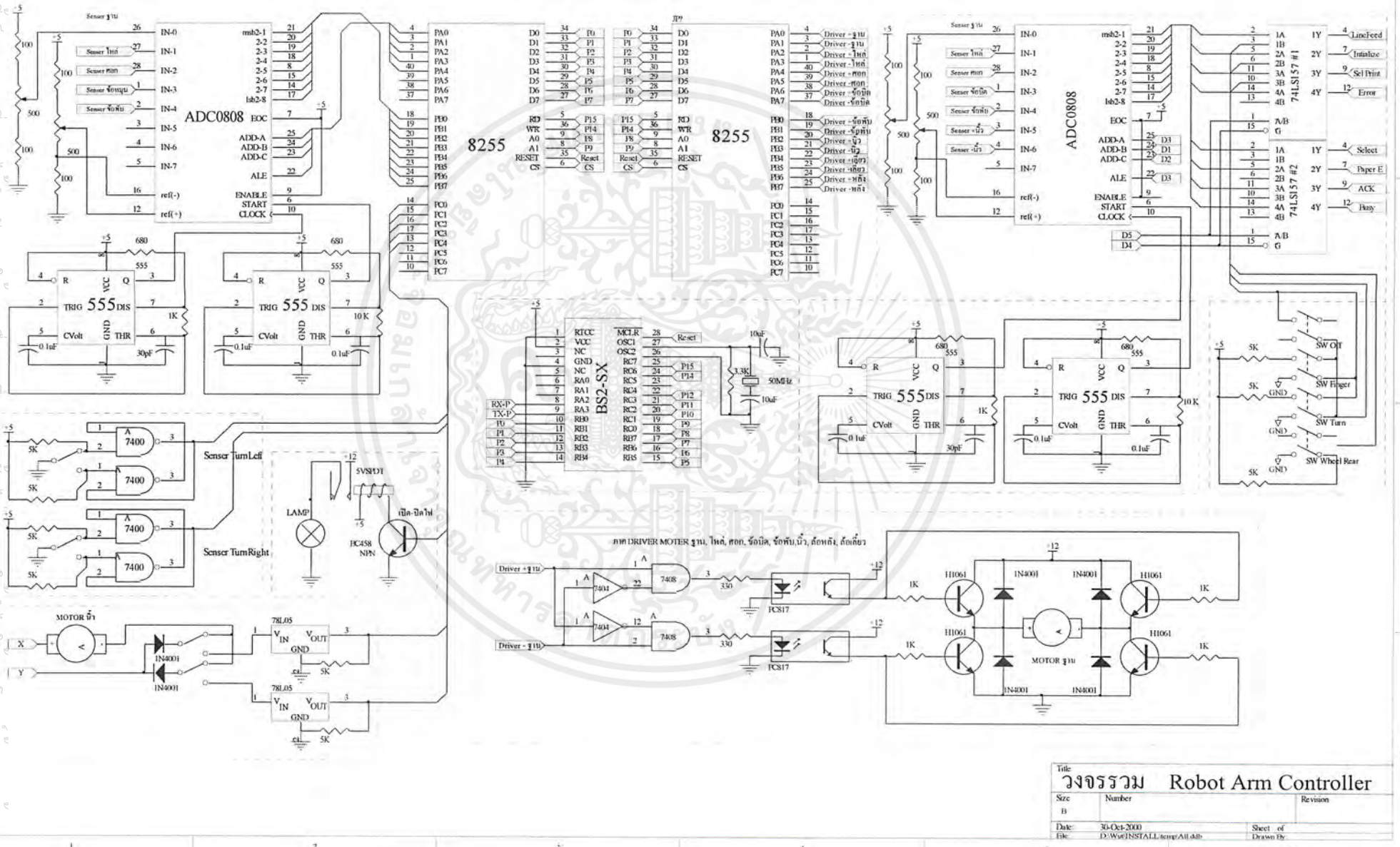
สัญญาณเกิดขึ้น เมื่อนำมาประมวลผลก็จะมีข้อผิดพลาดเกิดขึ้น ฉะนั้นจึงต้องมีการแก้ไขให้มีความผิดพลาดน้อยที่สุด จึงจะทำให้การทำงานมีประสิทธิภาพยิ่งขึ้น

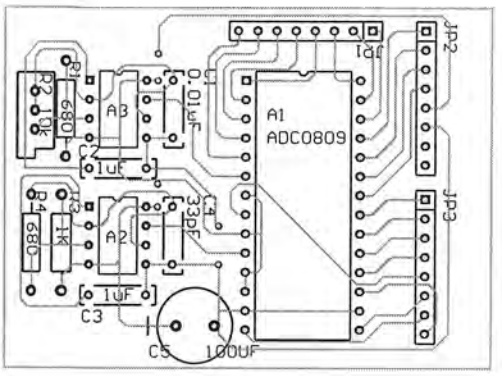
หนังสืออ้างอิง

1. วิบบูลย์ ชื่นแขก , ไมโครโปรเซสเซอร์ , สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, พิมพ์ครั้งที่ 2 พ.ศ. 2532
2. ดร.พิพัฒน์ หิรัณย์วณิชชากร , ระบบการสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์, บริษัท ซีเอ็ดยูเคชั่น จำกัด , พิมพ์ครั้งที่ 1 พ.ศ. 2542
3. กฤษดา วิศวธีรนนท์ , ไอซีดิจิตอล , ซีเอ็ดยูเคชั่น , พิมพ์ครั้งที่ 2 พ.ศ.2531
4. ธีรวัฒน์ ประกอบผล , การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ , สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) , พิมพ์ครั้งที่ 2 พ.ศ. 2541
5. วุฒิชัย กปิลกาญจน์ , กลไหลและพลศาสตร์เครื่องจักรกล , มหาวิทยาลัยเกษตรศาสตร์ ,พ.ศ. 2535
6. Walter N. Aleerich Electric Motor Control
7. กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ, Visual Basic 6 ฉบับโปรแกรมเมอร์, พิมพ์ครั้งที่3

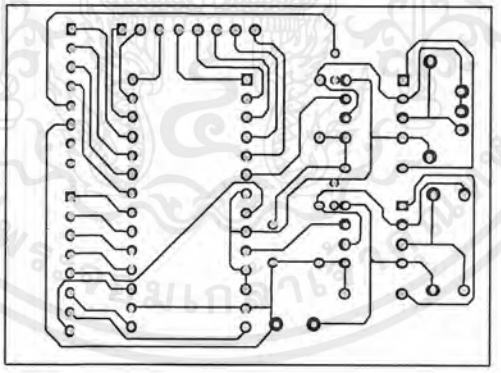
ภาคผนวก ก.

วงจรรวมและ
การวางอุปกรณ์และลายวงจรพิมพ์

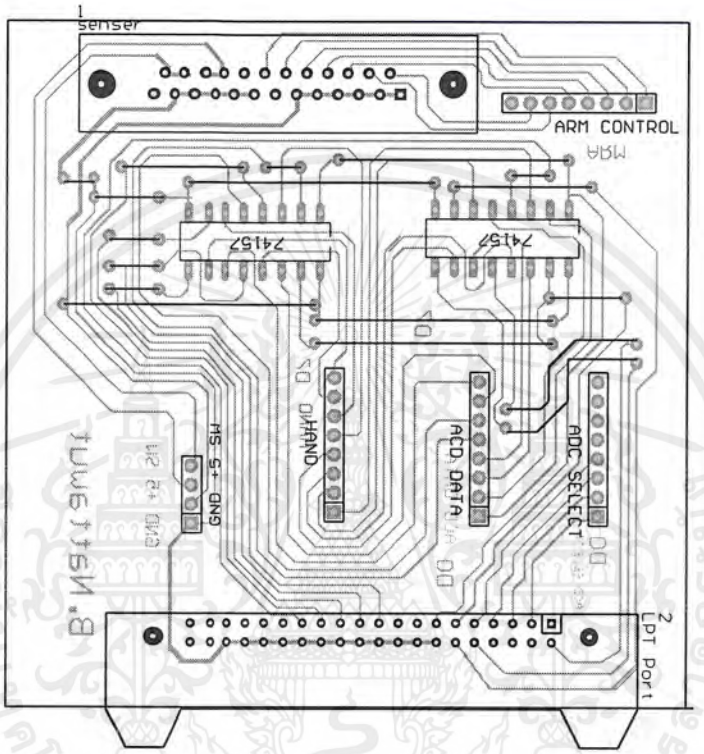

ADC0808
8255
8255
BS2-SX
74LS157 #1
74LS157 #2
555
555
555
7400
7400
7400
7400
7408
7404
7404
H1061
H1061
H1061
H1061
1N4001
1N4001
1N4001
1N4001
PC817
PC817
MOTOR จาน
MOTOR น้ำ
LAMP
BC458 NPN
5VSPDT
เปิด-ปิดไฟ
Sensor TurnLeft
Sensor TurnRight
ADC0808
Sensor จาน
Sensor ไหล่
Sensor ศอก
Sensor ข้อมือบน
Sensor ข้อมือ
Sensor ข้อพับ
Sensor นิ้ว
Sensor นิ้ว
LineFeed
Intialize
Sel Print
Error
Select
Paper E
ACK
Busy
SW OUT
SW Finger
SW Turn
SW Wheel Rear
Driver - จาน
Driver - ไหล่
Driver - ศอก
Driver - ข้อมือบน
Driver - ข้อมือ
Driver - ข้อพับ
Driver - นิ้ว
Driver - นิ้ว
Driver - ล้อหน้า
Driver - ล้อหลัง
Driver - ล้อเลี้ยว
ตาม DRIVER MOTER จาน, ไหล่, ศอก, ข้อมือ, ข้อพับ, นิ้ว, ล้อหลัง, ล้อเลี้ยว
Title
วงจรรวม Robot Arm Controller
Size
B
Number
Revision
Date: 30-Oct-2000
File: D:\Wut\INSTALL\temp\All.ddb
Sheet of
Drawn By:



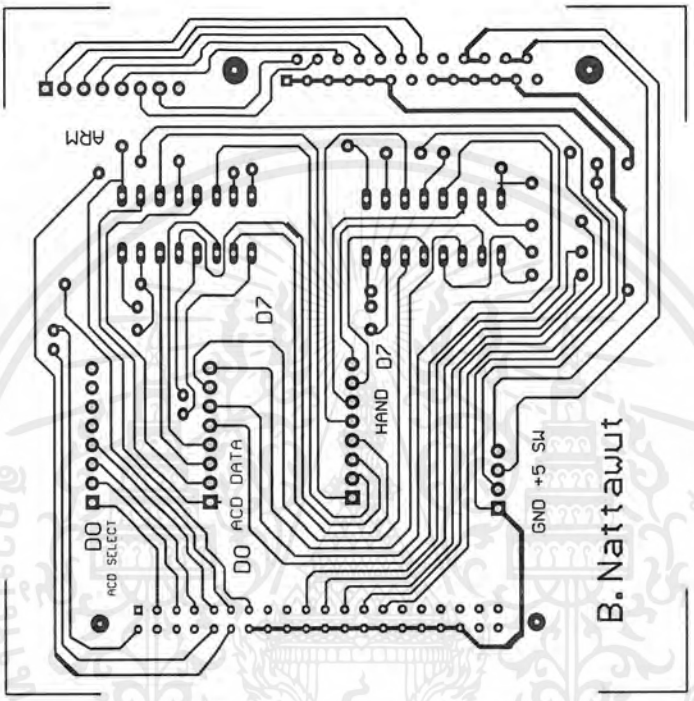
รูปที่ 7.2 การวางอุปกรณ์ของ
วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล



รูปที่ 7.3 ลายวงจรพิมพ์ของ
วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

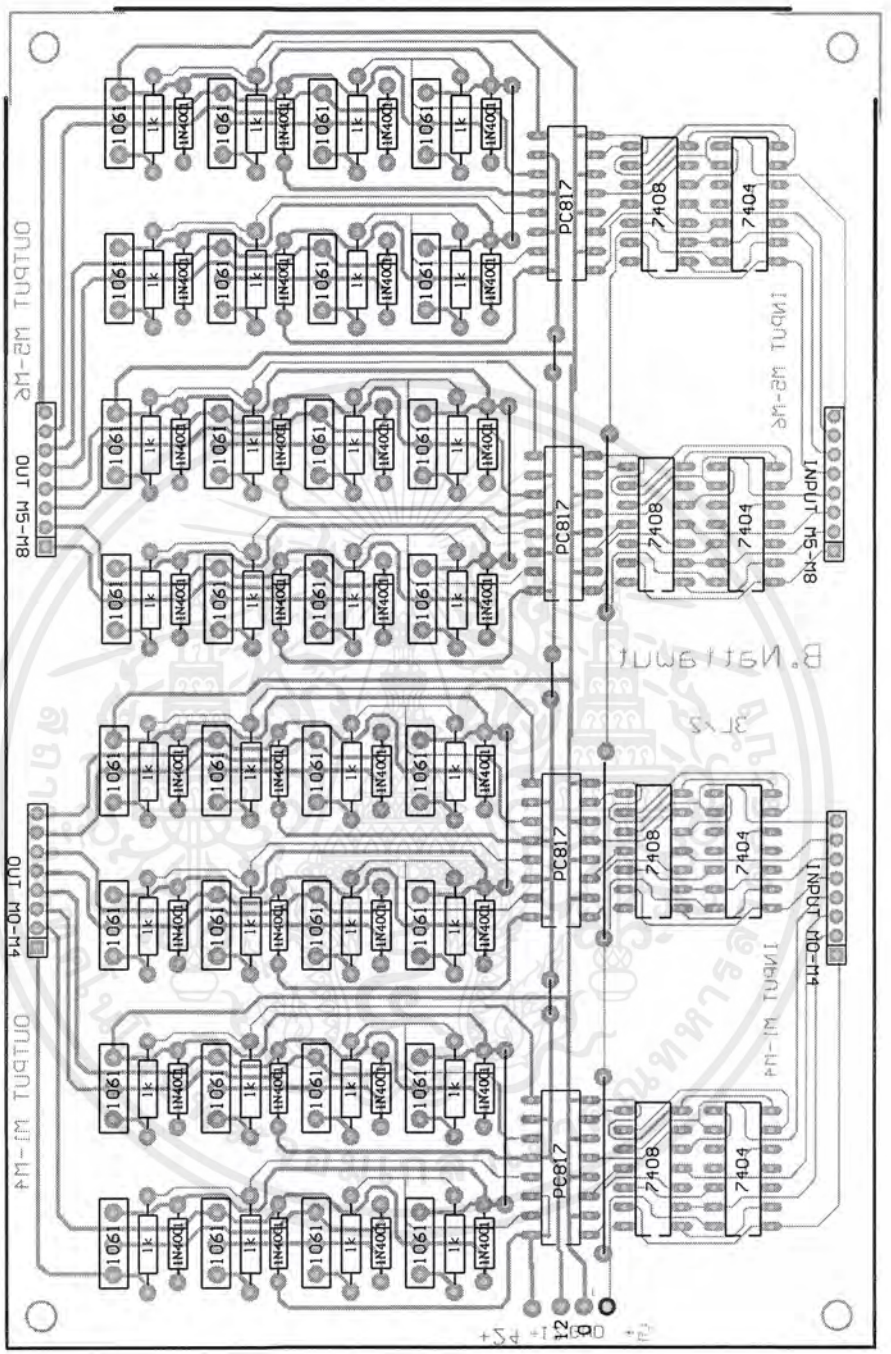


รูปที่ 7.4 การวางอุปกรณ์ของ
วงจรเลือก Connect Parallel

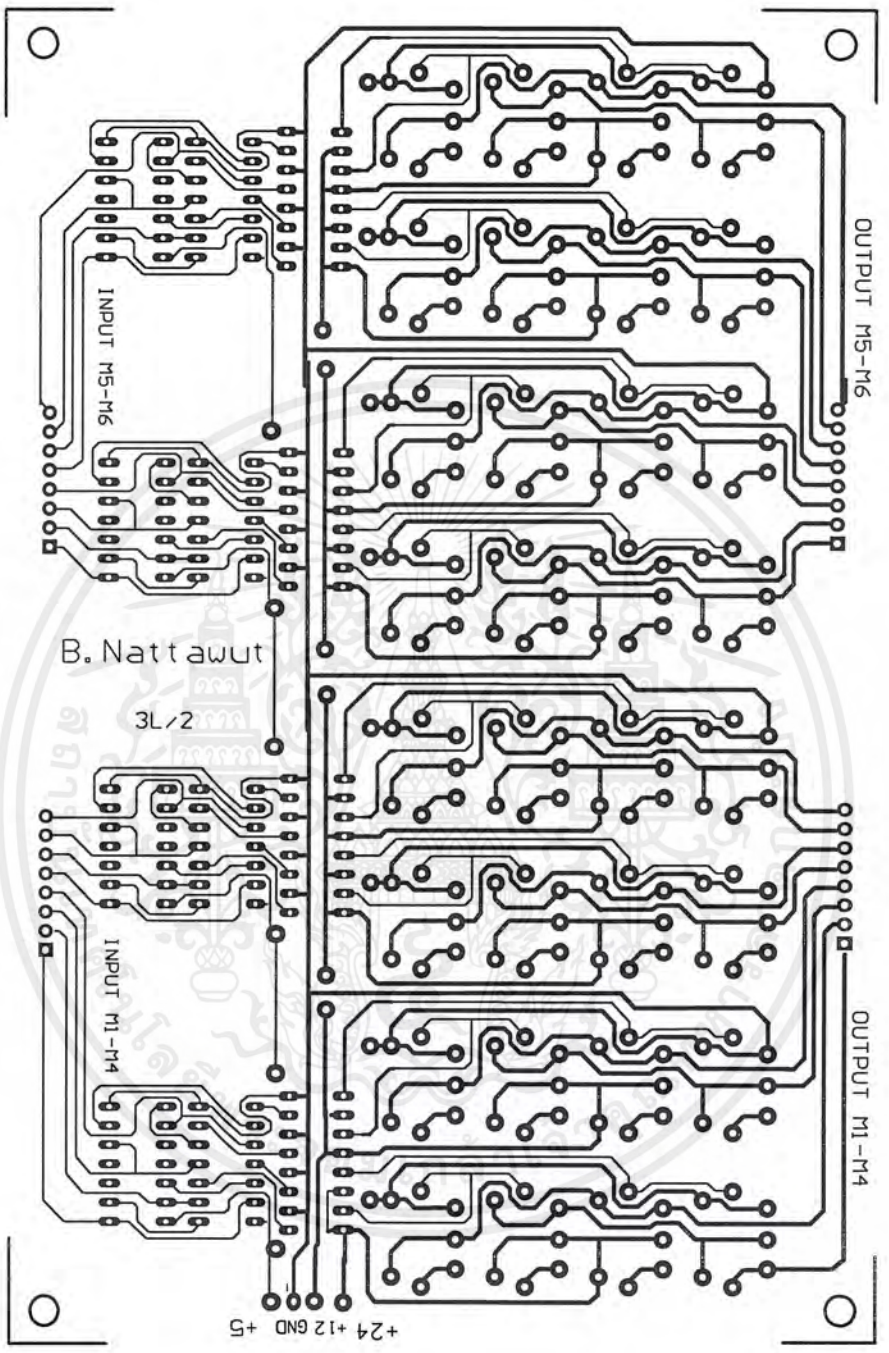


รูปที่ 7.5 ลายวงจรพิมพ์ของ

วงจรเลือก Connect Parallel



รูปที่ 7.6 การวางอุปกรณ์ของวงจรควบคุมการหมุนของ



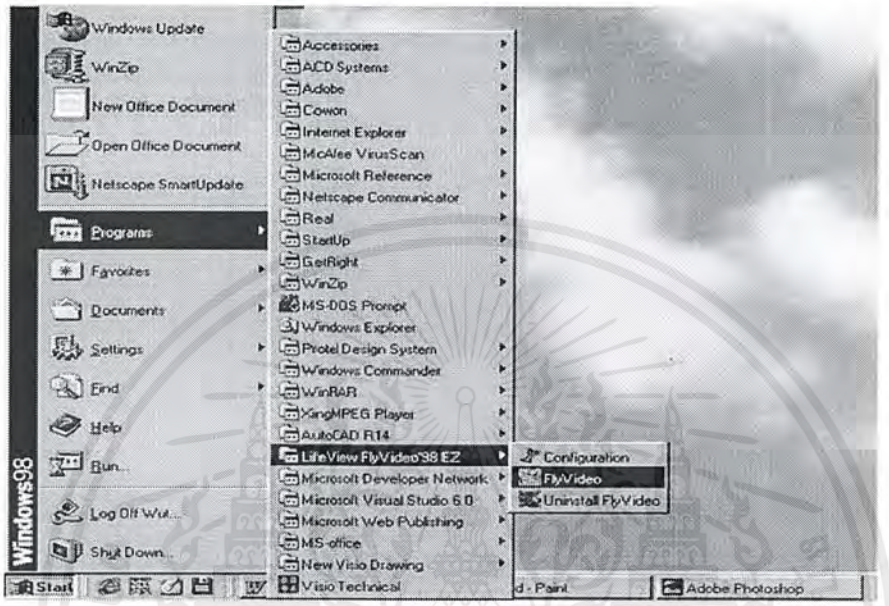
รูปที่ 7.7 ลายวงจรพิมพ์ของวงจรควบคุมการหมุนของมอเตอร์

ภาคผนวก ข.

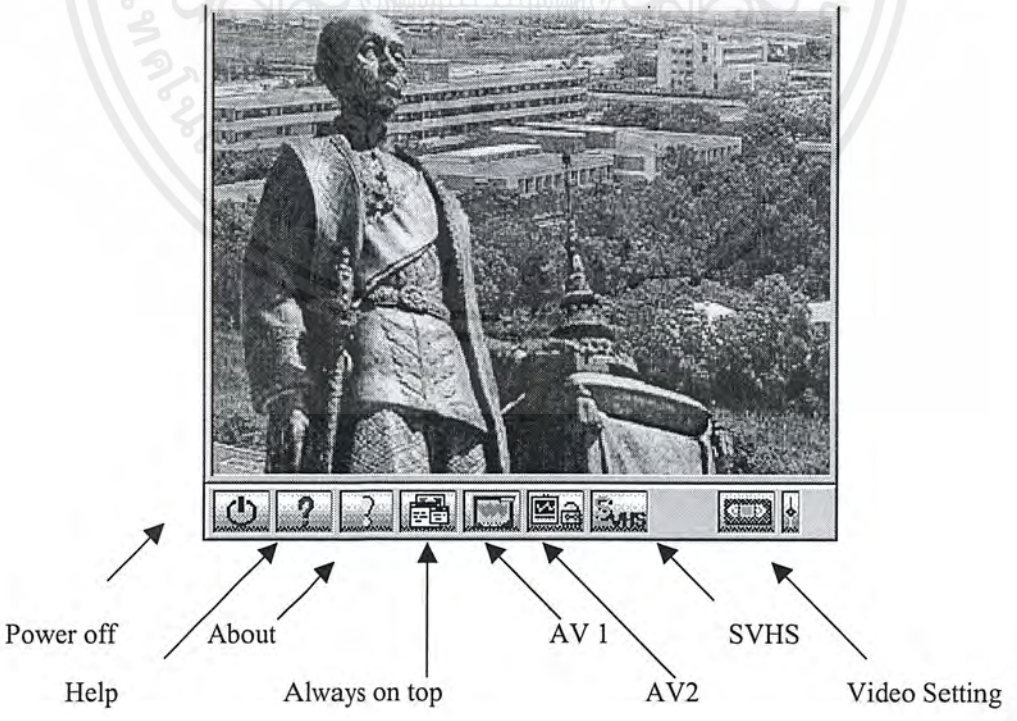
คู่มือการใช้งานและ
โปรแกรมที่ใช้งานในโครงงาน

การใช้งานโปรแกรม FlyVideo

1. คลิกปุ่ม Start เลือก Program + Life View FlyVideo ' 98 + FlyVideo เพื่อเปิด Program



2. แถบเครื่องมือหลัก



2.1 Power คลิกเมื่อต้องการเปิด Program

2.2 Help แนะนำรายละเอียดบางส่วนของ Program

2.3 About รายละเอียดของ Software

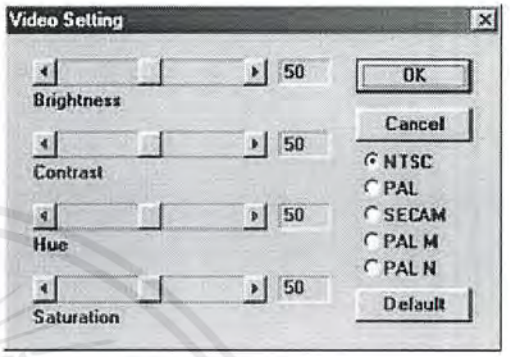
2.4 Always on top วางกรอบ Fly Video ทับไว้บน Program อื่นๆ

2.5 AV1 คลิกดูภาพจาก Video1

2.6 AV2 คลิกดูภาพจาก Video2

2.7 SVHS คลิกดูภาพจาก S-Video

2.8 Video Setting ตั้งค่าต่างๆ ของ Video →



โปรแกรมควบคุมหุ่นยนต์แขนกล

โปรแกรมควบคุมภายในหุ่นยนต์แขนกล (Basic stamp 2SX)

```
'{$STAMP BS2SX}
'**** OK**
freqout 13,500,500,800
pause 10
dtmfout 13,[9]
pause 10
freqout 13,800,650
portin   con 16
portout  con 16
baudrate        con 240
selectmode      var byte(6)
i               var byte
acddata         var byte
motor           var byte
senser          var byte
turn            var byte
in_pc           var byte

'--------------------main Program--- ----------------
main:
        dirh    = $20
        dirl    = $80
        turn    = $03
        serin   portin,baudrate,[wait("go")]      'wait go from computer
        pause 5
        freqout 13,500,700
```

```
'------------------------setup Contorl 8255 -------------
        dirs = $ffff                'all output
        outh = %11011000            'select *cs2
        pause 20
        outl = $80                  '8255 o/p all port
        outh = %10011011            'control *cs2
        pause 20
        outh    = %11010100         'select *cs1
        pause 20
        outl    = $83               '8255 pA(o),pB(i),pc3-0(i),pc7-4(o)
        outh  = %10010111           'control *cs1
        pause 20
        out14 = 1
        dirs    = $0000
'-----------------------------------------------
mode:
        dirh    = $00
        dirl    = $80
        selectmode(5) = 0
        serout  portout,baudrate,["rm"]         'sent reseve mode

        serin   portin,baudrate,[wait("m"),STR selectmode\5\"*"]
        'motor = selectmode(4)

'  ****** Driver Motor  (=>*cs2)  *****
driver:
        dirs  = $ffff
        outh    = %11011000             'select *cs2
        outl  = selectmode(0)
```

```
        outh     = %10011000    'out pA
        outh     = %11011000
        outl  = selectmode(1)
        outh     = %10011001    'out pB
        outh     = %11011001
        pause motor             'delay motor
        IF selectmode(2) = "t" THEN mor1
        goto mor2
mor1:
        gosub stop_mor
mor2:   dirs = $0000

'************************senser finger ******************
        gosub In_in_pC
        senser = in_pc & %00001100
        serout   portout,baudrate,["s",$08,senser,"#"]       'sent senser finger,turn
        pause 2
'******************* senser turn**************
        if selectmode(2) = "r" OR selectmode(2)= "q" OR selectmode(2)= "d" then right_1
        goto turn3
'-----turn right    ----
right_1:
        if selectmode(2)="r" then right1
        goto left_1
right1:
        gosub In_in_pC
        turn = in_pc & %00000011
 if turn = %00000001 OR turn =  %00000010 then turn2    '==> not direct
right2:
                dirs  = $ffff
```

```
                outh     = %11011001    'select *cs2
                outl  = $10             'turn right
                pause  1
                outh     = %10011001    'out pB
                outh     = %11011001
                pause 20
                gosub In_in_pC          'read senser turn
                turn = in_pc & %00000011
                if turn = $02 then  turn2
                        goto right2
'--------turn left--------
left_1:
        if selectmode(2)="q" then left1
        goto center_1
left1:
        gosub In_in_pC
        turn = in_pc & %00000011
                if turn = %00000001 OR turn =  %00000010 then turn2     '==> not direct
left2:
                dirs  = $ffff
                outh     = %11011001            'select *cs2
                outl  = $20                     'turn left
                pause  1
                outh     = %10011001    'out pB
                outh     = %11011001
                pause 2
                gosub In_in_pC 'read senser turn
                turn = in_pc & %00000011
                if turn = $01 then turn2
                        goto left2
```

```
'---------turn center--------
center_1:
if selectmode(2)="d" then center1
        goto turn2
center1:
        gosub In_in_pC
        turn = in_pc & %00000011

                if turn = %00000010  then center2        '==> present state is right
                goto center3
center2:
                dirs  = $ffff
                outh     = %11011001              'select *cs2
                outl  = $20                       'turn left
                pause  1
                outh     = %10011001    'out pB
                outh     = %11011001
                pause 2
                gosub In_in_pC 'read senser turn
                turn = in_pc & %00000011
                if turn = $03 then turn2
                        goto center2
center3:
                if turn = %00000001  then center4        '==> present state is left
                goto turn2
center4:
                dirs  = $ffff
                outh     = %11011001              'select *cs2
                outl  = $10                       'turn right
                pause  1
```

```
        outh    = %10011001    'out pB
        outh    = %11011001
        pause 2
        gosub In_in_pC 'read senser turn
        turn = in_pc & %00000011
        if turn = $03 then turn2
                goto center4

turn2:
        gosub stop_mor
        pause 2
        serout  portout,baudrate,["s","t",turn,"#"] 'sent senser turn
turn3:


'  ******* A to D convertor ( =>*cs1)  *****
acd:
        FOR i= $00 to $04
        dirs    = $ffff
        outh    = %11010100            'select *cs1
        outl =i                    'select channal ADC
        pause 2
        outh    = %10010100            'out pA
        outh    = %11010100
        out3    = 1                    'clk ALK
        outh    = %10010100            'out pA
        outh    = %11010100
        dirl    = $00
        outh    = %11010101
```

```
        outh     = %01010101            'in pB
        pause 2
        acddata = inl
        pause 2
        outh     = %11010101

        serout   portout,baudrate,["z",i,acddata,"#"]        'sent transeve acd
        dirs = $0000
NEXT
        pause 5
'***************** Lamp ********************
        if selectmode(2)="I" then Lamp1          'Open Lamp
        goto Lamp2
Lamp1:
        dirh = $ff
        dirA = $0
        dirB = $f
        outh = %11010110        'select *cs1
        pause 2
        outh = %11010110        'pC
        outB = %0001            'out C4
        pause 2
        outh = %10010110
        outh = %11010110
Lamp2:
        if selectmode(2)="c" then Lamp3          'Close Lamp
        goto Lamp4
Lamp3:
        dirh = $ff
        dirA = $0
```

```
        dirB = $f
        outh = %11010110        'select *cs1
        pause 2
        outh = %11010110        'pC
        outB = %0000            'out C4
        pause 2
        outh = %10010110
        outh = %11010110
Lamp4:

'************************************
dirs = $0000
goto mode
end
'****************************************
In_in_pC:
        dirh = $ff
        dirl = $00
        outh = %11010110                'select *cs1
        pause  5
        outh     = %01010110            'in pC
        pause 2
        in_pc = inA
        pause 2
        outh     = %11010110
        in_pc = in_pc & %00001111
        dirs = $0000
return
'**********************
stop_mor:
```

```
        dirs  = $ffff
        outh     = %11011000          'select *cs2
        pause 2
        outl     = $00                'stop pA
        outh     = %10011000
        outh     = %11011000
        outl     = $00                'stop pB
        outh     = %10011001
        outh     = %11011001
        dirs  = $0000
return
```

โปรแกรมควบคุมหลัก (Visual Basic)

โปรแกรมเมนูหลัก (MDIrob.Frm)

```
Public Sub Stop_m()
 Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "t", "0", Chr(0))
   End Sub
```

```
Public Sub Driver_Morter(Mode, Driv0, Driv1, Driv2, Driv3, Driv4)
If SelectMode = True Then
  MSComm1.Output = Mode
  Delay (20)
  MSComm1.Output = Driv0
  Delay (20)
  MSComm1.Output = Driv1
  Delay (20)
  MSComm1.Output = Driv2
  Delay (20)
    MSComm1.Output = "*"
    Delay (20)
SelectMode = False
   Mode = 0
    End If
End Sub
```

```
Private Sub exit_Click()
End
End Sub
```

```
Private Sub Link_rob_Click()
Unload Arm_frm01
Unload robot01
```

```
Unload test1ro
Unload test2arm
Load Link_robot1
End Sub
```

```
Private Sub Link_robo_Click()
Unload Arm_frm01
Unload robot01
Unload test1ro
Unload test2arm
Status_1 = "Link"
Load Link_robot1
End Sub
```

```
Private Sub MDIForm_Load()
StatusBar1.Panels(1) = " Welcome to Robot Arm Controller  ##  Plese select menu   ##"
StatusBar1.Panels(2) = " กรุณา Open Serial Port ครับ "
SelectMode = False
AdcMode = False
MSComm1.RThreshold = 1
ComPortNumber = 0
Status_1 = 0
End Sub
```

```
Private Sub MDIForm_Unload(Cancel As Integer)
End
End Sub
```

```
Private Sub Menu1_Click()
Unload Arm_frm01
```

```
Unload Link_robot1
Unload test1ro
Unload test2arm
Load robot01
End Sub
-----

Private Sub Menu2_Click()
Load Arm_frm01
End Sub
-----

Private Sub Menu4_Click()
Unload Arm_frm01
Unload Link_robot1
Unload test1ro
Unload test2arm
Status_1 = "robot01"
Load robot01
End Sub
-----

Private Sub Menu5_Click()
Unload Link_robot1
Unload robot01
Unload test1ro
Unload test2arm
Status_1 = "arm"
Load Arm_frm01
End Sub
-----

Public Sub MSComm1_OnComm()
If comEvReceive = 2 Then
```

```
DoEvents
 Buffer = MSComm1.Input
    Select Case Buffer
 Case "ok"
   'Text1.Text = "Running"
  Case "rm"
    SelectMode = True
       Case Else

  If Left$(Buffer, 1) = "z" Then       'ตรวจตัว "z" ตัวแรก
    AdcMode = True
       AdcSelect = Asc(Right$(Left$(Buffer, 2), 1))
      Select Case AdcSelect
        Case Is = 0
        Base = Asc(Right$(Left$(Buffer, 3), 1))
       Case Is = 1
         Shoulder = Asc(Right$(Left$(Buffer, 3), 1))
      Case Is = 2
        Elbow = Asc(Right$(Left$(Buffer, 3), 1))
     Case Is = 3
         Spin = Asc(Right$(Left$(Buffer, 3), 1))
    Case Is = 4
        Fold = Asc(Right$(Left$(Buffer, 3), 1))
     End Select
    Call Delay(10)
            End If

   If Left$(Buffer, 1) = "s" Then       'ตรวจตัว "s" ตัวแรก
    If Asc(Right$(Left$(Buffer, 2), 1)) = 8 Then    ' input senser finger
       Finger_in2 = Asc(Right$(Left$(Buffer, 3), 1))
```

```
    Select Case Finger_in2
       Case Is = Hex(0)
          Finger = "Gap"
       Case Is = Hex(4)
          Finger = "Max"
       Case Is = Hex(8)
          Finger = "Min"
       End Select
       Call Delay(10)
            End If
    If (Right$(Left$(Buffer, 2), 1)) = "t" Then     ' input senser turn
    Dim turn2 As Integer
     turn2 = Asc(Right$(Left$(Buffer, 3), 1))
    Select Case turn2
       Case Is = 1
          Turn = "Left"
       Case Is = 2
          Turn = "Right"
       Case Is = 3
          Turn = "Center"
       End Select
       Call Delay(10)
            End If
    End If
  End Select
End If
End Sub
```

```
Private Sub testarm_Click()
Unload Arm_frm01
```

```
Unload Link_robot1
Unload robot01
Unload test1ro
Status_1 = "test_r"
Load test2arm
End Sub
```

```
Private Sub testmoter_Click()
Unload Arm_frm01
Unload Link_robot1
Unload robot01
Unload test2arm
Status_1 = "test"
Load test1ro
End Sub
```

```
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
On Error Resume Next
 Select Case Button.Key
      Case "New"
       MenuNew
      Case "OpenPort"
       MenuOpenPort
      Case "Open"
       MenuOpen
      Case "lamp"
       MenuLamp_On
     Case "Save"
        MenuSave1_On
     Case "Show1"
```

```
        MenuShow1
 End Select
End Sub
-----------------------------------------------------------------------

 Private Sub MenuSave1_On()
  FileCopy App.Path & "\database\DBase.sav", App.Path & "\database\DBase.mdb"
    CHK = False
    Load frmControl
 End Sub
 Private Sub MenuOpen()
    Load frmLoad
 End Sub
 -----------------------------------------------------------------------

Private Sub MenuNew()
    FileCopy App.Path & "\database\DBase.sav", App.Path & "\database\DBase.mdb"
    CHK = False
    Load frmControl
End Sub
-----------------------------------------------------------------------

Private Sub MenuLamp_On()
If (ComPortNumber = 1) Or (ComPortNumber = 2) Then
       MDIrob.MSComm1.Output = "go"
       Call Delay(500)
 If Toolbar1.Buttons(7).Value = tbrPressed Then
        Toolbar1.Buttons(7).Image = 14
       Toolbar1.Buttons(7).ToolTipText = "Close Lamp"
       Toolbar1.Buttons(7).Caption = " Off "
         Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "l", Chr(0), Chr(0))
          ElseIf Toolbar1.Buttons(7).Value = tbrUnpressed Then
```

```
            Toolbar1.Buttons(7).Image = 13
        Toolbar1.Buttons(7).ToolTipText = "Open Lamp"
        Toolbar1.Buttons(7).Caption = " On  "
        Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "c", Chr(0), Chr(0))
            End If
Else
MsgBox "กรุณา Open Serial Port ก่อน ครับ  !!! ", vbOKOnly
End If
End Sub
-------------------------------------------------------------------------

Private Sub MenuOpenPort()
   If Toolbar1.Buttons(59).Value = tbrPressed Then
     Toolbar1.Buttons(59).Image = 7
     Toolbar1.Buttons(59).ToolTipText = "Close Serial Port"
     Toolbar1.Buttons(59).Caption = " Close "
       Combo001.Enabled = False

  Select Case Combo001.ListIndex
Case 0 To 1
    ComPortNumber = Combo001.ItemData(Combo001.ListIndex)
 Case -1
       ComPortNumber = 2
  Case Else
    StatusBar1.Panels(2) = "เกิดข้อผิดพลาดในการเชื่อมต่อ Port "
  End Select
   If MSComm1.CommPort <> ComPortNumber Then
    MSComm1.CommPort = ComPortNumber  'เลือกใช้ COM1 หรือ COM2
   '  MSComm1.Settings = "9600,N,8,1"
    MSComm1.InputLen = 0
  End If
```

```
        MSComm1.PortOpen = True
        StatusBar1.Panels(2) = " >> Serial Port Open ..Com " & Str(MSComm1.CommPort) & "<<"
    End If
        If Toolbar1.Buttons(59).Value = tbrUnpressed Then
        Toolbar1.Buttons(59).Image = 8
        Toolbar1.Buttons(59).ToolTipText = "Open Serial Port"
        Toolbar1.Buttons(59).Caption = " Open "
        StatusBar1.Panels(2) = " << Serial Port Clost >>"
        Combo001.Enabled = True
        MSComm1.PortOpen = False
    End If
End Sub
```

โปรแกรมย่อย-การทำงานแบบ Manual (robot01.frm)

```
Dim Drive1 As Integer, Drive2 As Integer
Dim AdcSelect1 As Integer, AdcData1 As Integer
Dim AdcMode1 As Boolean
Private Sub back2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H80), Chr(0), Chr(0), Chr(0))
     End Sub
```

```
Private Sub back2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub Brown_Click()
MMControl1.Visible = True
Dialog1.CancelError = True
On Error GoTo error1:
Dialog1.ShowOpen
MMControl1.FileName = Dialog1.FileName
MMControl1.Command = "Open"
Exit Sub
error1:
   MMControl1.Visible = False
   MMControl1.Command = " Close"
     End Sub
```

```
Private Sub center1_Click()
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "d", "0", Chr(0))
  End Sub
Private Sub finger_1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
```

```
Call MDIrob.Driver_Morter("m", Chr(&H1), Chr(&H0), Chr(0), Chr(0), Chr(0))
End Sub
```

```
Private Sub finger_1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub finger_2_MouseDown(Button As Integer, Shift As Integer. X As Single, Y As
Single)
Call MDIrob.Driver_Morter("m", Chr(&H2), Chr(&H0), Chr(0), Chr(0), Chr(0))
End Sub
```

```
Private Sub finger_2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub Form_Load()
SelectMode = False
AdcMode1 = False
Text1.Text = Hex$(HScroll1.Value) & " H"
Text2.Text = Hex$(HScroll2.Value) & " H"
Text3.Text = Hex$(HScroll3.Value) & " H"
Text4.Text = Hex$(HScroll4.Value) & " H"
Text5.Text = Hex$(HScroll5.Value) & " H"
Base = 100
Shoulder = 128
Elbow = 100
Spin = 100
Fold = 128
```

```
End Sub
Private Sub Stop_m()
Call MDIrob.Stop_m
End Sub
```

```
Private Sub Format1_Click()
If video.ShowDlgVideoFormat Then
 End If
End Sub
```

```
 Private Sub front_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H40), Chr(0), Chr(0), Chr(0))
   End Sub
```

```
 Private Sub front_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call MDIrob.Stop_m
End Sub
```

```
Private Sub HScroll1_Change()
Text1 Text = Hex$(HScroll1.Value) & " H"
Do While (Base + 1) < HScroll1.Value
   Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H4), Chr(0), Chr(0), Chr(0))
      DoEvents
   Loop
Do While (Base - 1) > HScroll1.Value
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H8), Chr(0), Chr(0), Chr(0))
   DoEvents
Loop
Call Delay(5000)
Call MDIrob.Stop_m
```

```
End Sub
-----------------------------------------------------------------------
Private Sub HScroll2_Change()
Text2.Text = Hex$(HScroll2.Value) & " H"
Do While (Shoulder + 1) < HScroll2.Value
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H2), Chr(0), Chr(0), Chr(0))
    DoEvents
Loop

Do While (Shoulder - 1) > HScroll2.Value
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H1), Chr(0), Chr(0), Chr(0))
  DoEvents
Loop
Call Delay(5000)
Call MDIrob.Stop_m
End Sub
-----------------------------------------------------------------------
Private Sub HScroll3_Change()
Text3.Text = Hex$(HScroll3.Value) & " H"
Do While (Elbow + 1) < HScroll3.Value
Call MDIrob.Driver_Morter("m", Chr(&H40), Chr(&H0), Chr(0), Chr(0), Chr(0))
 Call Delay(500)
  Loop
Do While (Elbow - 1) > HScroll3.Value
Call MDIrob.Driver_Morter("m", Chr(&H80), Chr(&H0), Chr(0), Chr(0), Chr(0))
 Call Delay(500)
  Loop
  Call Delay(5000)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub HScroll4_Change()
Text4.Text = Hex$(HScroll4.Value) & " H"
Do While (Spin + 1) < HScroll4.Value
Call MDIrob.Driver_Morter("m", Chr(&H10), Chr(&H0), Chr(0), Chr(0), Chr(0))
    Call Delay(300)
Loop
Do While (Spin - 1) > HScroll4.Value
Call MDIrob.Driver_Morter("m", Chr(&H20), Chr(&H0), Chr(0), Chr(0), Chr(0))
  Call Delay(300)
Loop
Call Delay(5000)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub HScroll5_Change()
Text5.Text = Hex$(HScroll5.Value) & " H"
Do While (Fold + 1) < HScroll5.Value
Call MDIrob.Driver_Morter("m", Chr(&H8), Chr(&H0), Chr(0), Chr(0), Chr(0))
   Call Delay(300)
Loop
Do While (Fold - 1) > HScroll5.Value
Call MDIrob.Driver_Morter("m", Chr(&H4), Chr(&H0), Chr(0), Chr(0), Chr(0))
  Call Delay(300)
Loop
Call Delay(5000)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub left1_Click()
      Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "q", "0", Chr(0))
    End Sub
```

```
Private Sub None_Click()
video.Preview = False
End Sub
Private Sub over_Click()
video.Preview = False
End Sub
Private Sub Prev_Click()
video.Preview = True
End Sub
Private Sub right1_Click()
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "r", 0, Chr(0))
 End Sub
Private Sub Source1_Click()
If video.ShowDlgVideoSource Then
 End If
End Sub
Private Sub Start_Capture_Click()
Text15.Enabled = False
video.CaptureFile = "D:\Wut\avi\cap.avi"
video.CaptureFile = Text15.Text
video.CaptureVideo
End Sub
Private Sub start_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If ((ComPortNumber = 1) Or (ComPortNumber = 2)) And MDIrob.MSComm1.PortOpen = True
Then
MDIrob.MSComm1.Output = "go"
Text8.Text = "go"
front.Enabled = True
back2.Enabled = True
left1.Enabled = True
```

```
right1.Enabled = True
center1.Enabled = True
StatusBar1.Panels(1) = "Open Com " & Str(MDIrob.MSComm1.CommPort)
    If Turn = "Center" Or Turn = "Right" Or Turn = "Left" Then
      Call MDIrob.Stop_m
    Else
      Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "d", 0, Chr(0))
      Call Delay(100)
    End If
Else
MsgBox "กรุณา Open Port ก่อน ครับ  !!! ", vbOKOnly
End If
End Sub
-------------------------------------------------------------------------------
Private Sub start_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call Delay(50)
HScroll1.Value = Base
HScroll2.Value = Shoulder
HScroll3.Value = Elbow
HScroll4.Value = Spin
HScroll5.Value = Fold
End Sub
-------------------------------------------------------------------------------
Private Sub Stop_Capture_Click()
Text15.Enabled = True
video.CaptureEnd
End Sub

Private Sub stop1_Click()
Call MDIrob.Stop_m
End Sub
-------------------------------------------------------------------------------
Private Sub Timer1_Timer()
```

```
Text10.Text = Hex$(Base) & " H"
Text11.Text = Hex$(Shoulder) & " H"
 Text12.Text = Hex$(Elbow) & " H"
 Text13.Text = Hex$(Spin) & " H"
 Text14.Text = Hex$(Fold) & " H"
 Text9.Text = Buffer
 Frame2.Caption = Turn
 Text6.Text = "  " & Finger
End Sub
```

โปรแกรมย่อย_ การทำงานแบบ Automatic (Link_robot1.frm)

```
Private Sub Form_Load()
StatusBar1.Panels(1) = "##  Link  Robot To Arm  Controller ##"
pause1.Caption = "Start"
Drive1 = &H0
Drive2 = &H0
Out &H378, &H0
Out &H37A, &H5
Call Delay(10)
Inp (&H37A)
Text14.Text = ""
Text15.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
End Sub
-----------------------------------------------------
Private Sub Format1_Click()
If Video2.ShowDlgVideoFormat Then
End If
End Sub
-----------------------------------------------------
Private Sub none1_Click()
Video2.Preview = False
End Sub
-----------------------------------------------------
Private Sub pause1_Click()
If pause1.Caption = "Start" Or pause1.Caption = "Continue" Then
pause1.Caption = "Pause"
Call Delay(500)
If Turn = "Center" Or Turn = "Right" Or Turn = "Left" Then
   Call MDIrob.Stop_m
Else
```

```
    Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "d", 0, Chr(0))
Call Delay(1000)
End If
Timer_arm_in.Interval = 500
Timer_link.Interval = 100
StatusBar1.Panels(3) = " >> Pause to Stop <<"
Else
pause1.Caption = "Continue"
Timer_arm_in.Interval = 0
Timer_link.Interval = 0
Call MDIrob.Stop_m
StatusBar1.Panels(3) = " >> กรุณากด Continue <<"
End If
End Sub
-----------------------------------------------------------
Private Sub playback1_Click()
MMControl1.Visible = True
Dialog1.CancelError = True
On Error GoTo error1:
Dialog1.ShowOpen
MMControl1.FileName = Dialog1.FileName
MMControl1.Command = "Open"
 Exit Sub
error1:
   MMControl1.Visible = False
   MMControl1.Command = " Close"
End Sub
-----------------------------------------------------------
Private Sub Preview1_Click()
Video2.Preview = True
End Sub
-----------------------------------------------------------
```

```
Private Sub Source1_Click()
If Video2.ShowDlgVideoSource Then
End If
End Sub
-----------------------------------------------------------
Private Sub start1_Click()
Text18.Enabled = False
Video2.CaptureFile = "d:\wut\avi\caplink.avi"
Video2.CaptureFile = Text18.Text
Video2.CaptureVideo
End Sub
-----------------------------------------------------------
Private Sub stop1_Click()
Text18.Enabled = True
Video2.CaptureEnd
End Sub
-----------------------------------------------------------
Private Sub Timer_arm_in_Timer()
Out &H378, &H0
Call Delay(10)
Out &H378, &H8
Call Delay(200)
shoulder_arm = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \
2))
Text10.Text = Hex$(shoulder_arm) & " H"
Out &H378, &H1
Call Delay(100)
Out &H378, &H9
Call Delay(200)
base_arm = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \ 2))
Text9.Text = Hex$(base_arm) & " H"
Out &H378, &H2
Call Delay(100)
```

```
Out &H378, &HA
Call Delay(200)
elbow_arm = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \ 2))
Text11.Text = Hex$(elbow_arm) & " H"

Out &H378, &H3
Call Delay(100)
Out &H378, &HB
Call Delay(200)
spin_arm = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \ 2))
Text12.Text = Hex$(spin_arm) & " H"

Out &H378, &H4
Call Delay(100)
Out &H378, &HC
Call Delay(200)
fold_arm = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \ 2))
Text13.Text = Hex$(fold_arm) & " H"

Out &H378, &H5
Call Delay(100)
Out &H378, &HD
Call Delay(200)
finger_arm_down = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor
&HB) \ 2))

Out &H378, &H6
Call Delay(100)
Out &H378, &HE
Call Delay(200)
```

```
finger_arm_up = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \
2))

Out &H378, &H20
Call Delay(100)
ModeHand = ((Inp(&H379) And &HF8 Xor &H80) Or ((Inp(&H37A) And &HF Xor &HB) \ 2))
Call Delay(200)
'*********************************************
Select Case (ModeHand And &H7)
  Case Is = 5
  Text17.Text = "Finger"
    If finger_arm_down > 200 Then
      Text14.Text = "จับ" & finger_arm_down
      Finger_arm = "in"

    ElseIf finger_arm_up > 200 Then
      Text14.Text = "ปล่อย" & finger_arm_up
      Finger_arm = "out"

    ElseIf finger_arm_down <= 5 And finger_arm_up <= 5 Then
      Text14.Text = "หยุด" & finger_arm_up & finger_arm_down
      Finger_arm = "stopf"
    End If
   Case Is = 6
    Text17.Text = "เลี้ยว"
    If finger_arm_down > 200 Then
      Text15.Text = "Left" & finger_arm_down
      TurnS = "left"
        ElseIf finger_arm_up > 200 Then
      Text15.Text = "Right" & finger_arm_up
```

```
        TurnS = "right"
          ElseIf finger_arm_down <= 5 And finger_arm_up <= 5 Then
        Text15.Text = "**" & finger_arm_up & finger_arm_down
        TurnS = "stopt"
      End If
    Case Is = 3
      Text17.Text = "Rear Wheel"
      If finger_arm_down > 200 Then
        Text8.Text = "ถอย" & finger_arm_down
        BackWheel = "rear"
          ElseIf finger_arm_up > 200 Then
        Text8.Text = "หน้า" & finger_arm_up
        BackWheel = "go"
          ElseIf finger_arm_down <= 5 And finger_arm_up <= 5 Then
        Text8.Text = "หยุด" & finger_arm_up & finger_arm_down
        BackWheel = "stopb"
        End If
  Case Is = 7
      Text17.Text = "Off"
End Select
Out &H378, &H0
Text1.Text = Hex(Base)
Text2.Text = Hex(Shoulder)
Text3.Text = Hex(Elbow)
Text4.Text = Hex(Spin)
Text5.Text = Hex(Fold)
End Sub
-------------------------------------------------------------------
Private Sub Timer_link_Timer()
DoEvents
Drive1 = Drive1 And &HCF
```

```
'*****************************Turn***********
If TurnS = "left" Then
    Select Case Turn
      Case "Center"
        Turn_arm = 1
        Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "q", 0, Chr(0))
      Case "Right"
        Turn_arm = 2
        Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "d", 0, Chr(0))
      Case "Left"
        Turn_arm = 4
         Drive1 = Drive1 And &HCF
      End Select
         Call Delay(1000)
  ElseIf TurnS = "right" Then
    Select Case Turn
      Case "Center"
        Turn_arm = 3
        Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "r", 0, Chr(0))
      Case "Left"
      Turn_arm = 2
        Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "d", 0, Chr(0))
      Case "Right"
      Turn_arm = 4
                Drive1 = Drive1 And &HCF
      End Select
      Call Delay(1000)
          Else
    Drive1 = Drive1 And &HCF
  End If
```

```
    Text7.Text = Turn
     Drive1 = Drive1 And &HCF
      '*********************************Finger*************
    If Finger_arm = "in" Then  'จับ
      RF_arm = 1
       Text6.Text = Finger
         Drive2 = (Drive2 And &HFC) Or &H1
        ElseIf Finger_arm = "out" Then 'ปล่อย
        RF_arm = 2
         Text6.Text = Finger
         Drive2 = (Drive2 And &HFC) Or &H2
      Else
      RF_arm = 5
         Drive2 = Drive2 And &HFC
    End If
      '***********************Back Wheel**************
If BackWheel = "go" Then
   RF_arm = 3
   Drive1 = (Drive1 And &H3F) Or &H80
   ElseIf BackWheel = "rear" Then
   RF_arm = 4
   Drive1 = (Drive1 And &H3F) Or &H40
   Else
   RF_arm = 5
    Drive1 = Drive1 And &H3F
End If
'***********************Base****************
If (Base - 2) > (base_arm + 2) Then
   Drive1 = (Drive1 And &HF3) Or &HFB
ElseIf (Base + 2) < (base_arm - 2) Then
```

```
    Drive1 = (Drive1 And &HF3) Or &HF7
  Else
    Drive1 = (Drive1 And &HF3)
    End If
    '*****************************Shoulder********
  If (Shoulder - 1) > (shoulder_arm + 1) Then
    Drive1 = (Drive1 And &HFC) Or &HFD
  ElseIf (Shoulder + 1) < (shoulder_arm - 1) Then
    Drive1 = (Drive1 And &HFC) Or &HFE
  Else
    Drive1 = (Drive1 And &HFC)
    End If
  '***************Elbow*************************
  If (Elbow - 2) > (elbow_arm + 2) Then
    Drive2 = (Drive2 And &H3F) Or &HBF
  ElseIf (Elbow + 2) < (elbow_arm - 2) Then
    Drive2 = (Drive2 And &H3F) Or &H7F
  Else
    Drive2 = (Drive2 And &H3F)
    End If
      '******************Spin*****************
    If (Spin - 2) > (spin_arm + 2) Then
    Drive2 = (Drive2 And &HCF) Or &HEF
  ElseIf (Spin + 2) < (spin_arm - 2) Then
    Drive2 = (Drive2 And &HCF) Or &HDF
  Else
    Drive2 = (Drive2 And &HCF)
    End If
    '********************Fold**************
     If (Fold - 2) > (fold_arm + 2) Then
```

```
    Drive2 = (Drive2 And &HF3) Or &HF7
ElseIf (Fold + 2) < (fold_arm - 2) Then
    Drive2 = (Drive2 And &HF3) Or &HFB
Else
    Drive2 = (Drive2 And &HF3)
    End If
'*****************************************************
DoEvents
    Call Delay(0)
Call MDIrob.Driver_Morter("m", Chr(Drive2), Chr(Drive1), Chr(0), Chr(0), Chr(0))
  Call Delay(1000)
End Sub
```

โปแกรมย่อย_ทดสอบการทำงาน (testro1.frm)

```
Dim AdcSelect As Integer, AdcData As Integer

Private Sub back1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
    Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H40), "0", "0", Chr(0))
  Text1.Text = " หน้า "
 End Sub
    Private Sub back1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
  Call MDIrob.Stop_m
  End Sub
    Private Sub back2_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
 Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H80), "0", "0", Chr(0))
  Text1.Text = " ถอย "
  End Sub
  Private Sub back2_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
  Call MDIrob.Stop_m
  End Sub
    Private Sub base1_Click(Index As Integer)
   Call MDIrob.Stop_m
    End Sub
    Private Sub base1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
  Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H8), "0", "0", Chr(0))
    End Sub
```

```
Private Sub base1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Stop_m
End Sub
-----------------------------------------------------------------------
Private Sub base2_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H4), "0", "0", Chr(0))
End Sub
-----------------------------------------------------------------------
Private Sub base2_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single,
Y As Single)
Call MDIrob.Stop_m
End Sub
-----------------------------------------------------------------------
Private Sub Begin_Click()
If (ComPortNumber = 1) Or (ComPortNumber = 2) And MDIrob.MSComm1.PortOpen = True
Then
Frame2.Visible = True
Else
MsgBox "กรุณา Open Serial Port ก่อน ครับ  !!! ", vbOKOnly
End If
End Sub
-----------------------------------------------------------------------
Private Sub center1_Click()
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "d", "0", Chr(0))
Text2.Text = Turn
End Sub
-----------------------------------------------------------------------
Private Sub elbow1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H80), Chr(&H0), "0", "0", Chr(0))
End Sub
-----------------------------------------------------------------------
Private Sub elbow1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub elbow2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H40), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub elbow2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub finger1_Click()
Call MDIrob.Stop_m
End Sub
Private Sub finger1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H2), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub finger1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub finger2_Click()
Call MDIrob.Stop_m
End Sub
Private Sub finger2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H1), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub finger2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub fold1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H8), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub fold1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
```

```
Private Sub fold2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H4), Chr(&H0), "0", "0", Chr(0))
End Sub
-----------------------------------------------------------------------
Private Sub fold2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
-----------------------------------------------------------------------
Private Sub Form_Load()
SelectMode = False
AdcMode = False
End Sub
-----------------------------------------------------------------------
Private Sub front1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "q", "0", Chr(0))
   Text2.Text = Turn
End Sub
-----------------------------------------------------------------------
Private Sub front2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H0), "r", "0", Chr(0))
Text2.Text = Turn
End Sub
-----------------------------------------------------------------------
Private Sub shoulder1_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H2), "0", "0", Chr(0))
End Sub
-----------------------------------------------------------------------
Private Sub shoulder1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Stop_m
End Sub
-----------------------------------------------------------------------
Private Sub shoulder2_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H0), Chr(&H1), "0", "0", Chr(0))
```

```
End Sub
Private Sub shoulder2_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As
Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub spin1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H20), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub spin1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub spin2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H10), Chr(&H0), "0", "0", Chr(0))
End Sub
Private Sub spin2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Stop_m
End Sub
Private Sub start1_Click()
MDIrob.MSComm1.Output = "go"
Text9.Text = "go"
Call Delay(100)
If Turn = "Center" Or Turn = "Right" Or Turn = "Left" Then
   Call MDIrob.Stop_m
Else
   Call MDIrob.Driver_Morter("m", Chr(0), Chr(0), "d", 0, Chr(0))
Call Delay(100)
End If
End Sub
Private Sub stop1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Call MDIrob.Driver_Morter("m", Chr(&H1), Chr(&H0), "t", "0", Chr(0))
```

```
End Sub
Private Sub Timer1testro_Timer()
Text2.Text = Turn
Text3.Text = Hex$(Base) & " H"
Text4.Text = Hex$(Shoulder) & " H"
 Text5.Text = Hex$(Elbow) & " H"
 Text6.Text = Hex$(Spin) & " H"
 Text7.Text = Hex$(Fold) & " H"
 Text8.Text = Finger & Finger_in2

 If Buffer = "rm" Then
 Text10.Text = "rm"
 Else
 Text10.Text = ""
 Text11.Text = Buffer
 End If
End Sub
```

โปรมแกรมส่วนกลาง (Module.bas)

```
Public Base, Shoulder, Elbow, Spin, Fold, Finger_in2 As Integer
Public Finger, Turn, Status_1, Mode, Driv2, NameSave, NameLoad As String
Public ComPortNumber, DbTurn, DbRF As Byte
Public SelectMode As Boolean
Public AdcMode, CHK  As Boolean
Public Delay1 As Integer
Public Buffer As String
Public Video_On As Boolean
Public Driv0, Driv1, Driv3, Driv4, i, j As Integer
Public base_arm, shoulder_arm, elbow_arm, spin_arm, fold_arm, finger_arm_up,
finger_arm_down, Turn_arm, RF_arm As Integer
'Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
-----------------------------------------------------------------
Public Sub Delay(Delay1 As Integer)
For i = 1 To Delay1
DoEvents
'Delay_add (300)
Next i
End Sub
-----------------------------------------------------------------
Public Sub Delay_add(Delay2 As Integer)
For j = 1 To Delay2
DoEvents
Next j
End Sub
```

ภาคผนวก ค.

DATASHEET

National Semiconductor

June 1999

ADC0808/ADC0809
8-Bit µP Compatible A/D Converters with 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

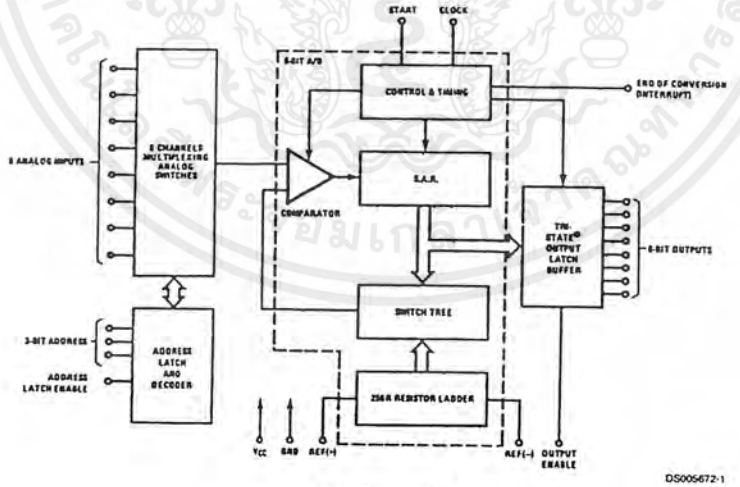
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 µs

Block Diagram

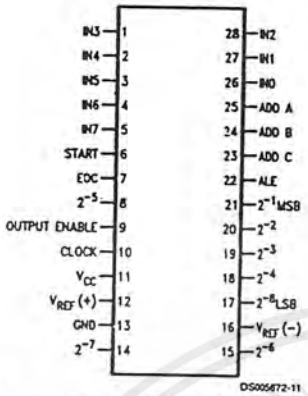


See Ordering Information

TRI-STATE® is a registered trademark of National Semiconductor Corp.

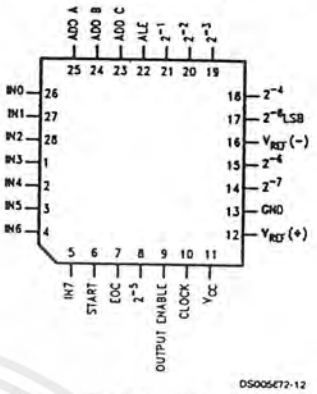
Connection Diagrams

Dual-In-Line Package



Order Number ADC0808CCN or ADC0809CCN
See NS Package J28A or N28A

Molded Chip Carrier Package



Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C			−55°C to +125°C
Error	±½ LSB Unadjusted	ADC0808CCN	ADC0808CCV	ADC0808CCJ	ADC0808CJ
	±1 LSB Unadjusted	ADC0809CCN	ADC0809CCV		
Package Outline		N28A Molded DIP	V28A Molded Chip Carrier	J28A Ceramic DIP	J28A Ceramic DIP

Absolute Maximum Ratings (Notes 2, 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage at Any Pin Except Control Inputs	−0.3V to (V_{CC}+0.3V)
Voltage at Control Inputs (START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)	−0.3V to +15V
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A=25°C	875 mW
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Molded Chip Carrier Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 8)	400V

Operating Conditions (Notes 1, 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le T_{MAX}$
ADC0808CCN,ADC0809CCN	−40°C≤T_A≤+85°C
ADC0808CCV, ADC0809CCV	−40°C ≤ T_A ≤ +85°C
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.0 V_{DC}

Electrical Characteristics

Converter Specifications: V_{CC}=5 V_{DC}=V_{REF+}, $V_{REF(-)}$=GND, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK}=640 kHz unless otherwise stated.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	ADC0808					
	Total Unadjusted Error	25°C			±½	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±¾	LSB
	ADC0809					
	Total Unadjusted Error	0°C to 70°C			±1	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±1¼	LSB
	Input Resistance	From Ref(+) to Ref(−)	1.0	2.5		kΩ
	Analog Input Voltage Range	(Note 4) V(+) or V(−)	GND−0.10		V_{CC}+0.10	V_{DC}
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	V_{CC}+0.1	V
$\frac{V_{REF(+)}+V_{REF(-)}}{2}$	Voltage, Center of Ladder		V_{CC}/2-0.1	V_{CC}/2	V_{CC}/2+0.1	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V
I_{IN}	Comparator Input Current	f_c=640 kHz, (Note 6)	−2	±0.5	2	μA

Electrical Characteristics

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, 4.75≤V_{CC}≤5.25V, −40°C≤T_A≤+85°C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
ANALOG MULTIPLEXER						
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=5V,				
		T_A=25°C		10	200	nA
		T_{MIN} to T_{MAX}			1.0	μA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=0,				
		T_A=25°C	−200	−10		nA
		T_{MIN} to T_{MAX}	−1.0			μA
CONTROL INPUTS						
$V_{IN(1)}$	Logical "1" Input Voltage		V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage				1.5	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN}=15V			1.0	μA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN}=0	−1.0			μA
I_{CC}	Supply Current	f_{CLK}=640 kHz		0.3	3.0	mA
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	I_O=−360 μA	V_{CC}−0.4			V

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40^{\circ}C \le T_A \le +85^{\circ}C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(0)}$	Logical "0" Output Voltage	I_O=1.6 mA			0.45	V
$V_{OUT(0)}$	Logical "0" Output Voltage EOC	I_O=1.2 mA			0.45	V
I_{OUT}	TRI-STATE Output Current	V_O=5V			3	µA
		V_O=0	−3			µA

Electrical Characteristics

Timing Specifications $V_{CC}=V_{REF(+)}=5V$, $V_{REF(-)}=GND$, $t_r=t_f=20$ ns and $T_A=25^{\circ}C$ unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	(*Figure 5*)		100	200	ns
t_{WALE}	Minimum ALE Pulse Width	(*Figure 5*)		100	200	ns
t_s	Minimum Address Set-Up Time	(*Figure 5*)		25	50	ns
t_H	Minimum Address Hold Time	(*Figure 5*)		25	50	ns
t_D	Analog MUX Delay Time From ALE	R_S=0Ω (*Figure 5*)		1	2.5	µs
t_{H1}, t_{H0}	OE Control to Q Logic State	C_L=50 pF, R_L=10k (*Figure 8*)		125	250	ns
t_{1H}, t_{0H}	OE Control to Hi-Z	C_L=10 pF, R_L=10k (*Figure 8*)		125	250	ns
t_c	Conversion Time	f_c=640 kHz, (*Figure 5*) (Note 7)	90	100	116	µs
f_c	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	(*Figure 5*)	0		8+2 µS	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE Output Capacitance	At TRI-STATE Outputs		10	15	pF

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: A zener diode exists, internally, from V_{CC} to GND and has a typical breakdown voltage of 7 V_{DC}.

Note 4: Two on-chip diodes are tied to each analog input which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. The spec allows 100 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 100 mV, the output code will be correct. To achieve an absolute $0V_{DC}$ to $5V_{DC}$ input voltage range will therefore require a minimum supply voltage of 4.900 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors. See *Figure 3*. None of these A/Ds requires a zero or full-scale adjust. However, if an all zero code is desired for an analog input other than 0.0V, or if a narrow full-scale span exists (for example: 0.5V to 4.5V full-scale) the reference voltages can be adjusted to achieve this. See *Figure 13*.

Note 6: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence (*Figure 6*). See paragraph 4.0.

Note 7: The outputs of the data register are updated one clock cycle before the rising edge of EOC.

Note 8: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Functional Description

Multiplexer. The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. *Table 1* shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE 1.

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach (*Figure 1*) was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +½ LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0808 as measured using the procedures outlined in AN-179.

Functional Description (Continued)

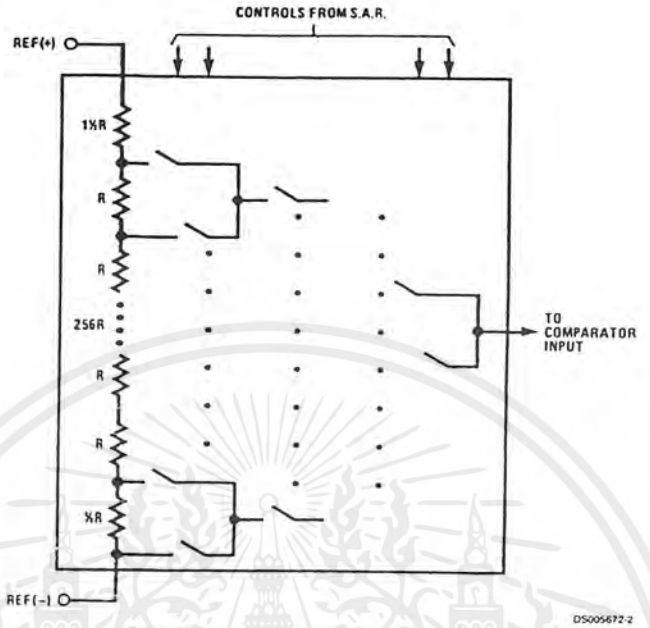


FIGURE 1. Resistor Ladder and Switch Tree

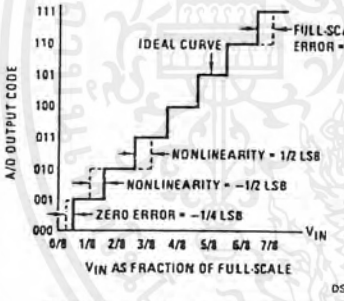


FIGURE 2. 3-Bit A/D Transfer Curve

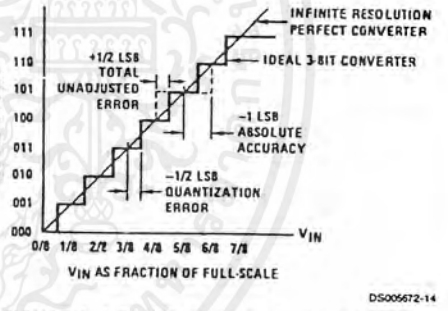


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

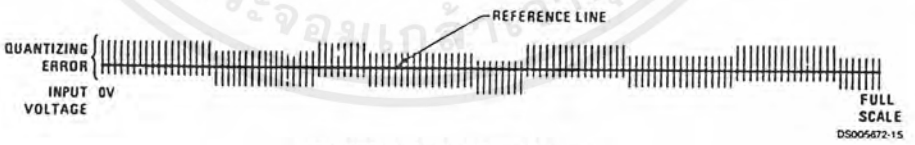


FIGURE 4. Typical Error Curve

Timing Diagram

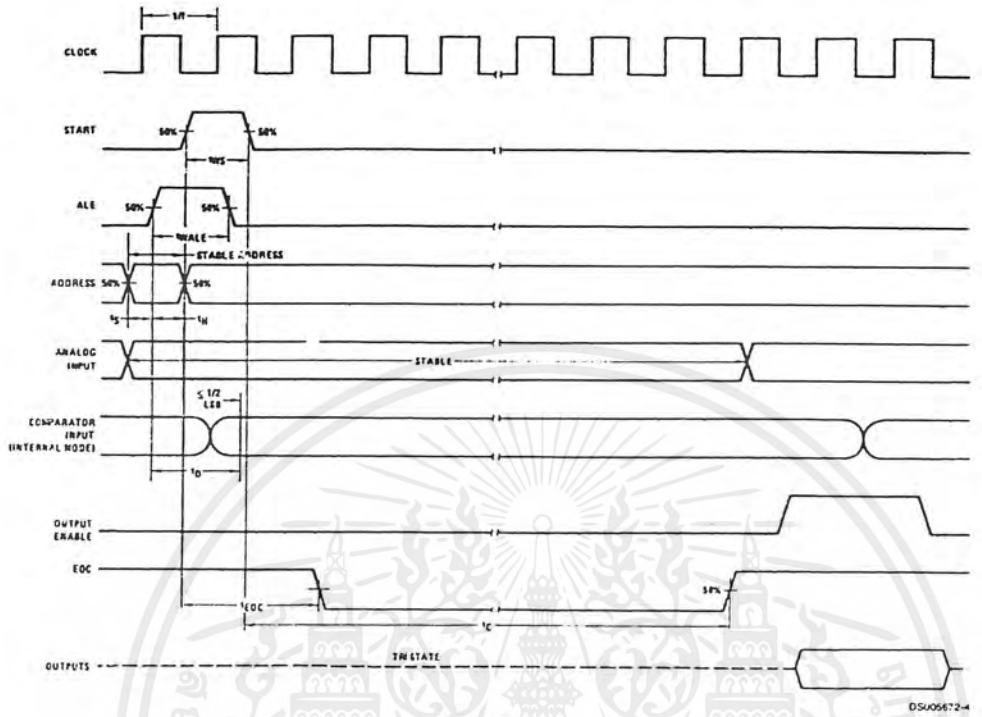


FIGURE 5.

Typical Performance Characteristics

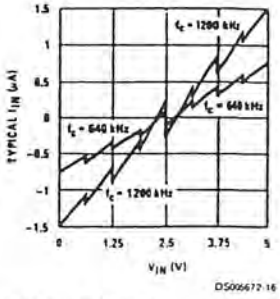


FIGURE 6. Comparator I_{IN} vs V_{IN} ($V_{CC}=V_{REF}=5V$)

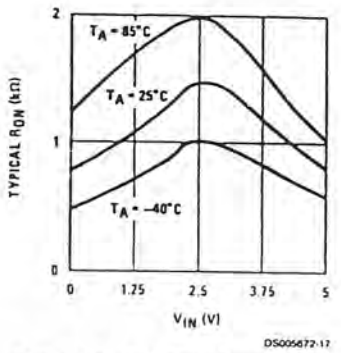


FIGURE 7. Multiplexer R_{ON} vs V_{IN} ($V_{CC}=V_{REF}=5V$)

TRI-STATE Test Circuits and Timing Diagrams

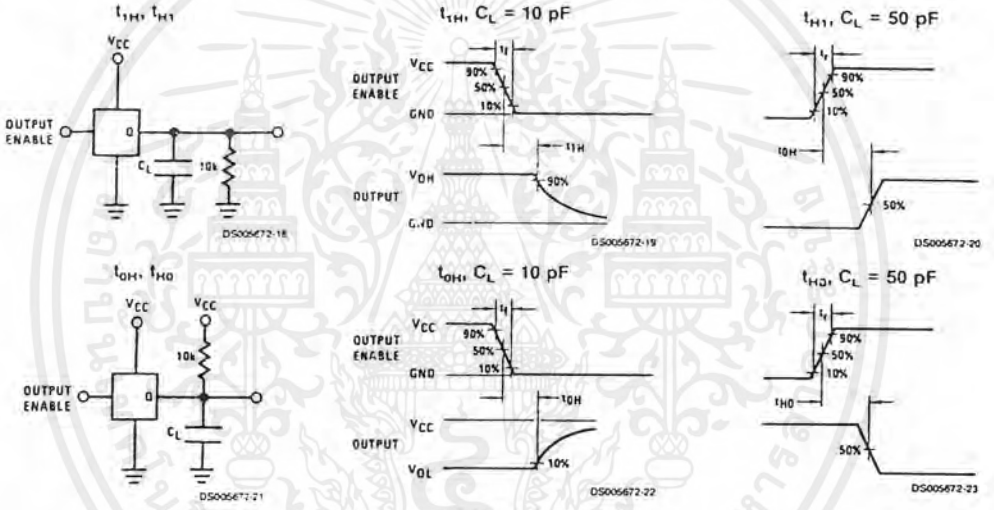


FIGURE 8.

Applications Information

OPERATION

1.0 RATIOMETRIC CONVERSION

The ADC0808, ADC0809 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0808 is expressed by the equation

$$\frac{V_{IN}}{V_{fs}-V_Z}=\frac{D_X}{D_{MAX}-D_{MIN}} \qquad (1)$$

V_{IN}=Input voltage into the ADC0808

V_{fs}=Full-scale voltage

V_Z=Zero voltage

D_X=Data point being measured

D_{MAX}=Maximum data limit

D_{MIN}=Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0808, ADC0809 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs, (*Figure 9*).

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc., are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a sys-

Applications Information (Continued)

tem reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC}=V_{REF}=5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

2.0 RESISTOR LADDER LIMITATIONS

The voltages from the resistor ladder are compared to the selected into 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder, Ref(−), should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 10 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V is used, the supply should be adjusted to the same voltage within 0.1V.

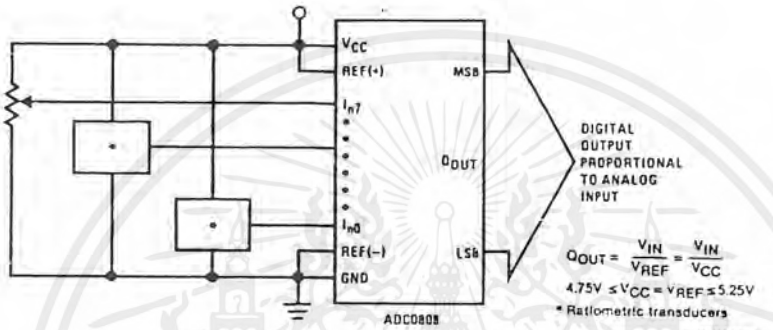
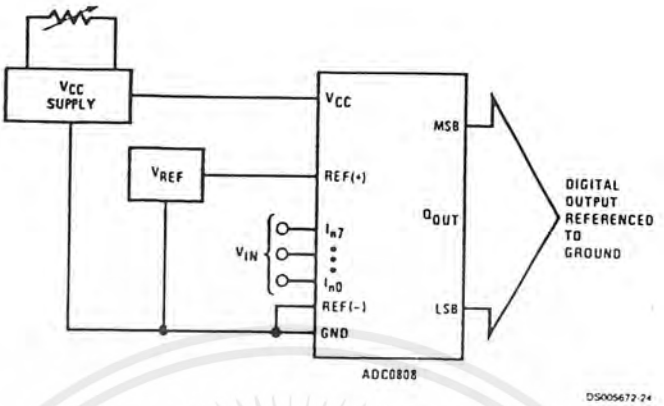


FIGURE 9. Ratiometric Conversion System

The ADC0808 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 11* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 12*. The LM301 is overcompensated to insure stability when loaded by the 10 µF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 13*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB bit to be half the size of a 5V reference system.

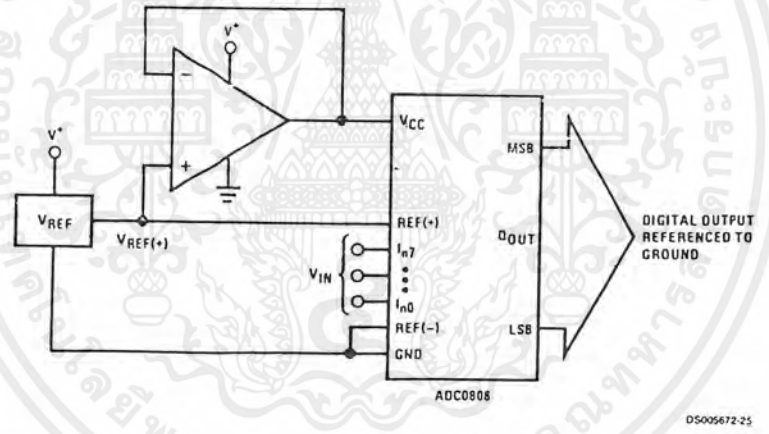
Applications information (Continued)



$$D_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 10. Ground Referenced Conversion System Using Trimmed Supply



$$D_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 11. Ground Referenced Conversion System with Reference Generating V_{CC} Supply

Applications Information (Continued)

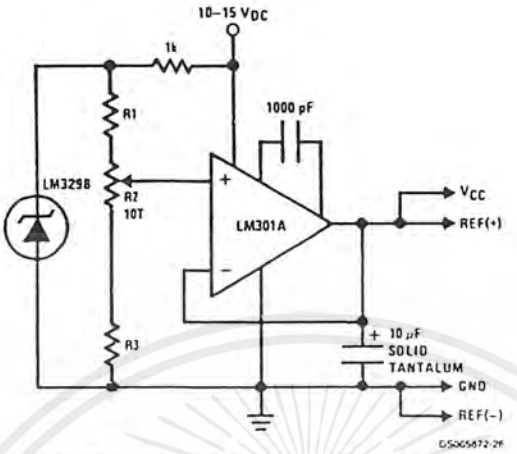
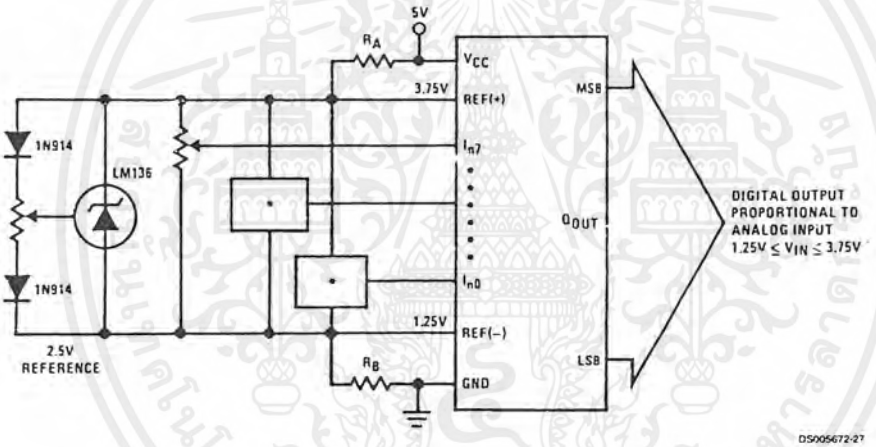


FIGURE 12. Typical Reference and Supply Circuit



$R_A = R_B$

*Ratiometric transducers

FIGURE 13. Symmetrically Centered Reference

3.0 CONVERTER EQUATIONS

The transition between adjacent codes N and N+1 is given by:

$$V_{IN} = \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} + \frac{1}{512} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

Where: V_{IN}=Voltage at comparator input
$V_{REF(+)}$=Voltage at Ref(+)
$V_{REF(-)}$=Voltage at Ref(–)
V_{TUE}=Total unadjusted error voltage (typically $V_{REF(+)} \div 512$)

Applications Information (Continued)

4.0 ANALOG COMPARATOR INPUTS

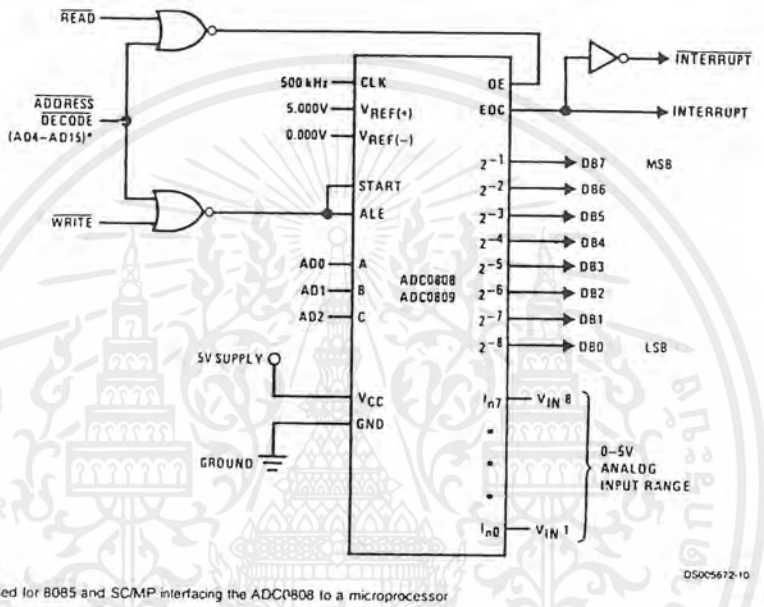
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6*.

If no filter capacitors are used at the analog inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally.

Typical Application



*Address latches needed for 8085 and SC/MP interfacing the ADC0808 to a microprocessor

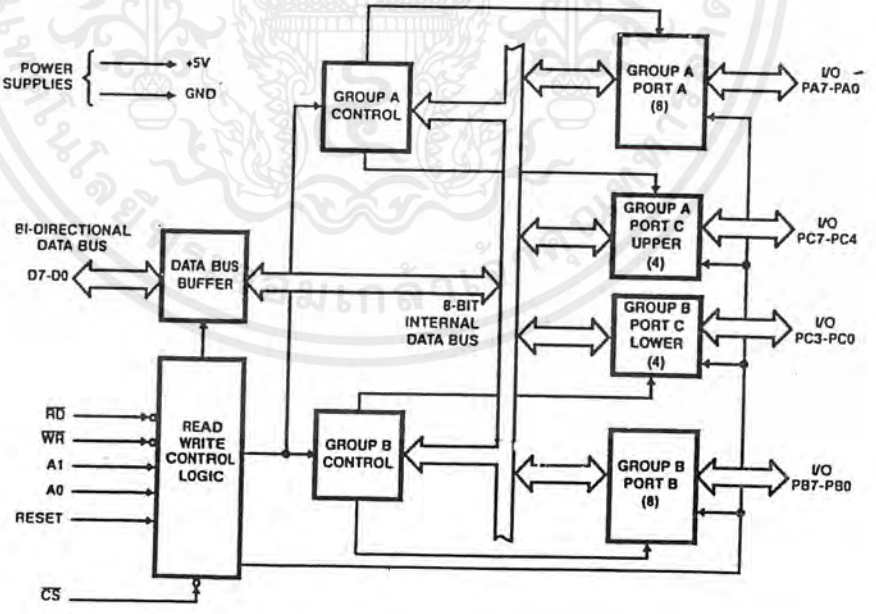
TABLE 2. Microprocessor Interface Table

PROCESSOR	READ	WRITE	INTERRUPT (COMMENT)
8080	MEMR	MEMW	INTR (Thru RST Circuit)
8085	RD	WR	INTR (Thru RST Circuit)
Z-80	RD	WR	INT (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA•φ2•R/W	VMA•φ•R/W	IRQA or IRQB (Thru PIA)

Pin Description

SYMBOL	PIN NUMBER	TYPE	DESCRIPTION
V_{CC}	26		V_{CC}: The +5V power supply pin. A 0.1µF capacitor between pins 26 and 7 is recommended for decoupling.
GND	7		GROUND
D0-D7	27-34	I/O	DATA BUS: The Data Bus lines are bidirectional three-state pins connected to the system data bus.
RESET	35	I	RESET: A high on this input clears the control register and all ports (A, B, C) are set to the input mode with the "Bus Hold" circuitry turned on.
$\overline{CS}$	6	I	CHIP SELECT: Chip select is an active low input used to enable the 82C55A onto the Data Bus for CPU communications.
$\overline{RD}$	5	I	READ: Read is an active low input control signal used by the CPU to read status information or data via the data bus.
$\overline{WR}$	36	I	WRITE: Write is an active low input control signal used by the CPU to load control words and data into the 82C55A.
A0-A1	8, 9	I	ADDRESS: These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the control word register. A0 and A1 are normally connected to the least significant bits of the Address Bus A0, A1.
PA0-PA7	1-4, 37-40	I/O	PORT A: 8-bit input and output port. Both bus hold high and bus hold low circuitry are present on this port.
PB0-PB7	18-25	I/O	PORT B: 8-bit input and output port. Bus hold high circuitry is present on this port.
PC0-PC7	10-17	I/O	PORT C: 8-bit input and output port. Bus hold circuitry is present on this port.

Functional Description



Functional Description

Data Bus Buffer

This three-state bi-directional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

(CS) Chip Select. A "low" on this input pin enables the communcation between the 82C55A and the CPU.

(RD) Read. A "low" on this input pin enables 82C55A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 82C55A.

(WR) Write. A "low" on this input pin enables the CPU to write data or control words into the 82C55A.

(A0 and A1) Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus (A0 and A1).

82C55A BASIC OPERATION

A1	A0	RD	WR	CS	INPUT OPERATION (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
1	1	0	1	0	Control Word → Data Bus
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control
					DISABLE FUNCTION
X	X	X	X	1	Data Bus → Three-State
X	X	1	1	0	Data Bus → Three-State

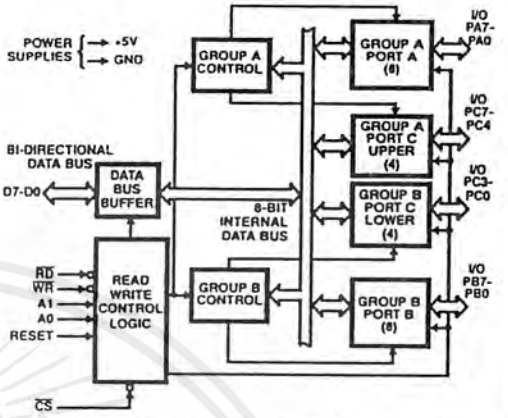


FIGURE 1. 82C55A BLOCK DIAGRAM. DATA BUS BUFFER, READ/WRITE, GROUP A & B CONTROL LOGIC FUNCTIONS

(RESET) Reset. A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode. "Bus hold" devices internal to the 82C55A will hold the I/O port inputs to a logic "1" state with a maximum hold current of 400µA

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7 - C4)

Control Group B - Port B and Port C lower (C3 - C0)

The control word register can be both written and read as shown in the "Basic Operation" table. Figure 4 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

Port A One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus-hold devices are present on Port A. See Figure 2A.

Port B One 8-bit data input/output latch/buffer and one 8-bit data input buffer. See Figure 2B.

Port C One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal output and status signal inputs in conjunction with ports A and B. See Figure 2B.

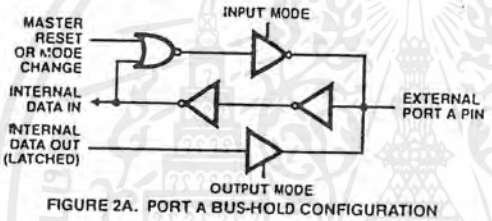


FIGURE 2A. PORT A BUS-HOLD CONFIGURATION

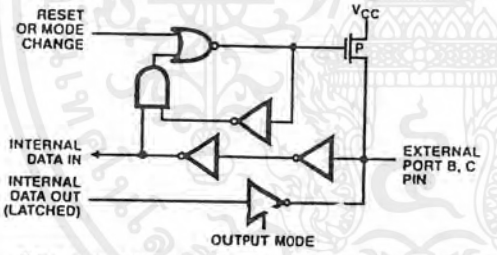


FIGURE 2B. PORT B AND C BUS-HOLD CONFIGURATION

FIGURE 2. BUS-HOLD CONFIGURATION

Operational Description

Mode Selection

There are three basic modes of operation than can be selected by the system software:

Mode 0 - Basic Input/Output
Mode 1 - Strobed Input/Output
Mode 2 - Bi-directional Bus

When the reset input goes "high", all ports will be set to the input mode with all 24 port lines held at a logic "one" level by internal bus hold devices. After the reset is removed, the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need to pullup or pull-down resistors in all-CMOS designs. The control word register will contain 9Bh. During the execution of the system program, any of the other modes may be selected using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine. Any port programmed as an output port is initialized to all zeros when the control word is written.

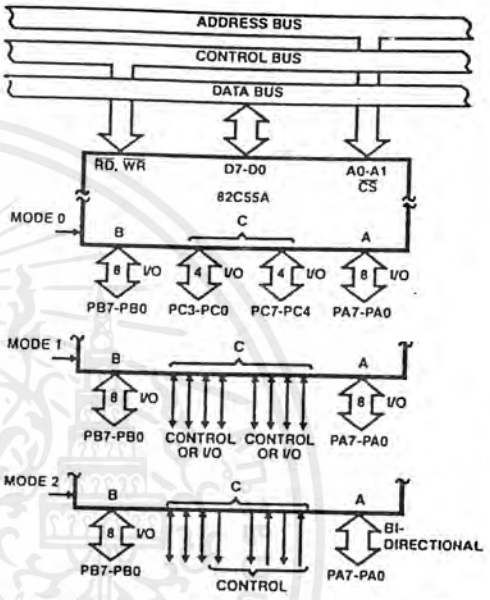


FIGURE 3. BASIC MODE DEFINITIONS AND BUS INTERFACE

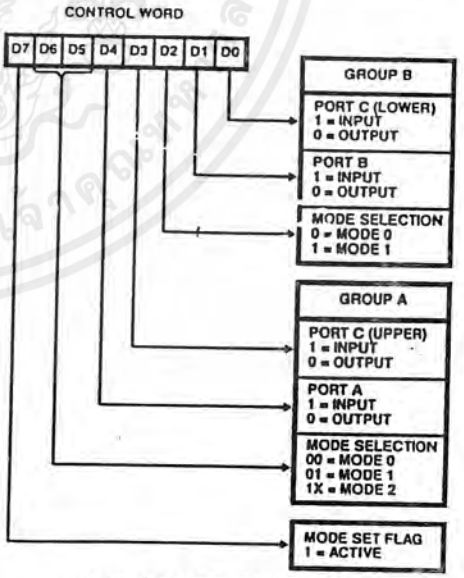
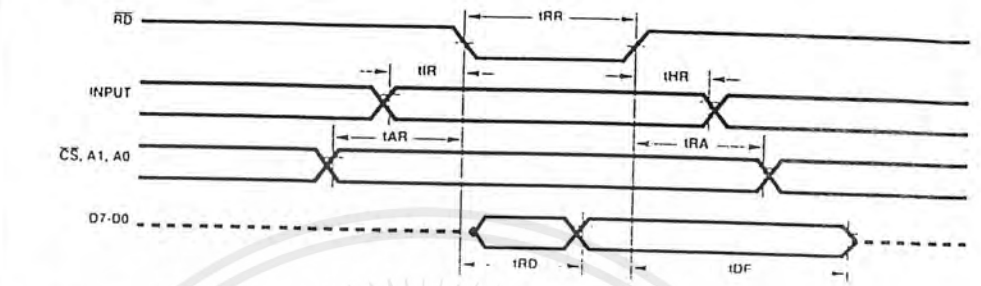
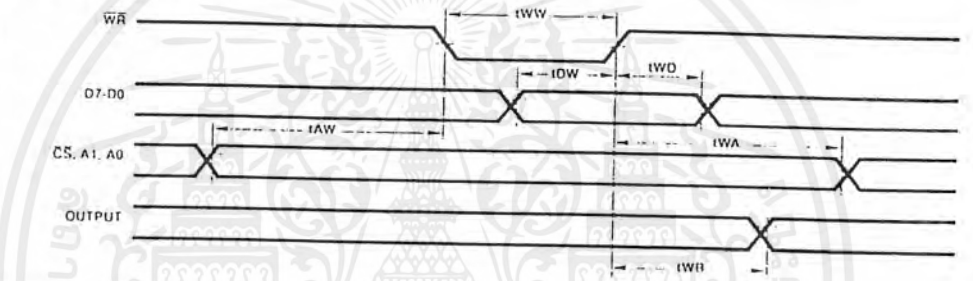


FIGURE 4. MODE DEFINITION FORMAT

Mode 0 (Basic Input)

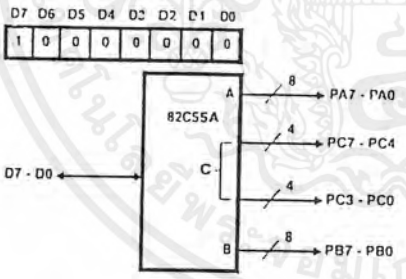


Mode 0 (Basic Output)

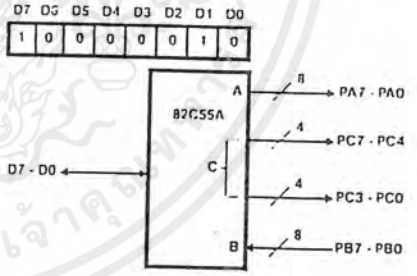


Mode 0 Configurations

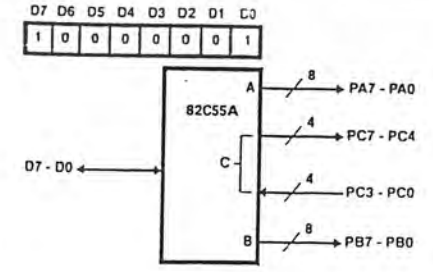
CONTROL WORD #0



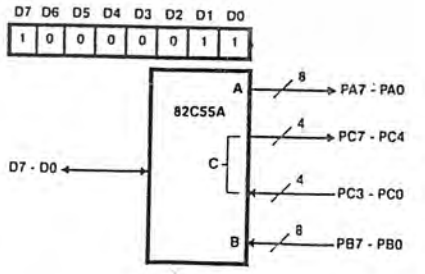
CONTROL WORD #2



CONTROL WORD #1



CONTROL WORD #3



The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance: Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

The mode definitions and possible mode combinations may seem confusing at first, but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs. PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature (Figure 5)

Any of the eight bits of Port C can be Set or Reset using a single Output instruction. This feature reduces software requirements in control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were output ports.

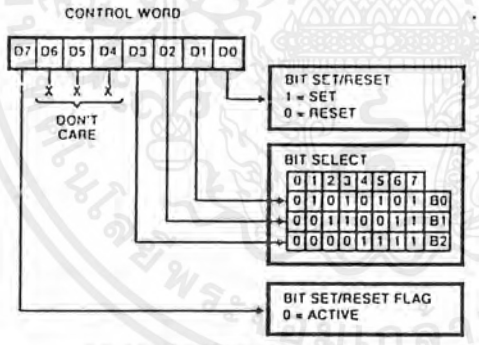


FIGURE 5. BIT SET/RESET FORMAT

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the programmer to enable or disable a CPU interrupt by a specific I/O device without affecting any other device in the interrupt structure.

INTE Flip-Flop Definition

(BIT-SET)-INTE is SET - Interrupt Enable

(BIT-RESET)-INTE is Reset - Interrupt Disable

NOTE: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No handshaking is required, data is simply written to or read from a specific port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports
- Any Port can be input or output
- Outputs are latched
- Input are not latched
- 16 different Input/Output configurations possible

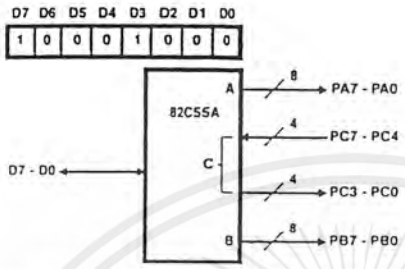
MODE 0 PORT DEFINITION

A		B		GROUP A			GROUP B	
D4	D3	D1	D0	PORT A	PORT C (Upper)	#	PORT B	PORT C (Lower)
0	0	0	0	Output	Output	0	Output	Output
0	0	0	1	Output	Output	1	Output	Input
0	0	1	0	Output	Output	2	Input	Output
0	0	1	1	Output	Output	3	Input	Input
0	1	0	0	Output	Input	4	Output	Output
0	1	0	1	Output	Input	5	Output	Input
0	1	1	0	Output	Input	6	Input	Output
0	1	1	1	Output	Input	7	Input	Input
1	0	0	0	Input	Output	8	Output	Output
1	0	0	1	Input	Output	9	Output	Input
1	0	1	0	Input	Output	10	Input	Output
1	0	1	1	Input	Output	11	Input	Input
1	1	0	0	Input	Input	12	Output	Output
1	1	0	1	Input	Input	13	Output	Input
1	1	1	0	Input	Input	14	Input	Output
1	1	1	1	Input	Input	15	Input	Input

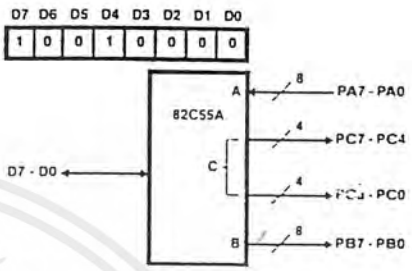
82C55A

Mode 0 Configurations (Continued)

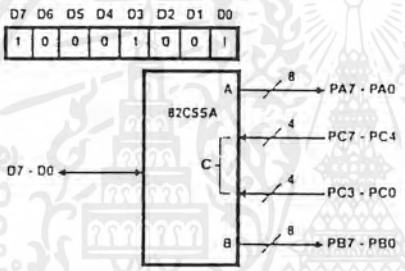
CONTROL WORD #4



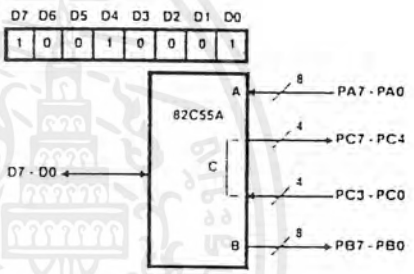
CONTROL WORD #8



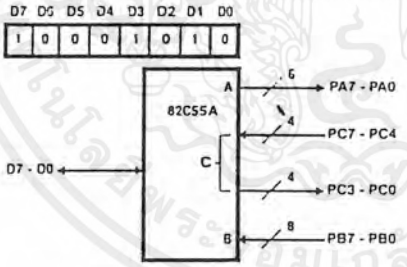
CONTROL WORD #5



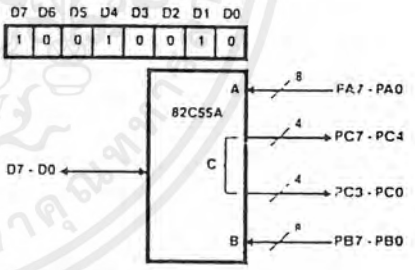
CONTROL WORD #9



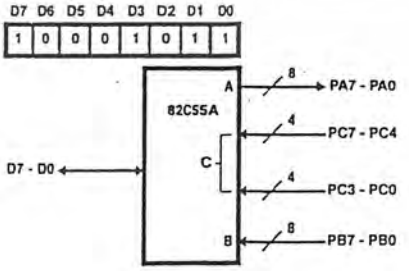
CONTROL WORD #6



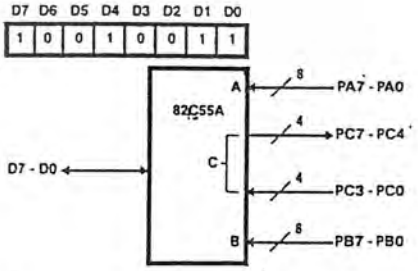
CONTROL WORD #10



CONTROL WORD #7

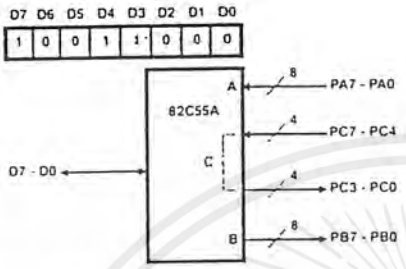


CONTROL WORD #11

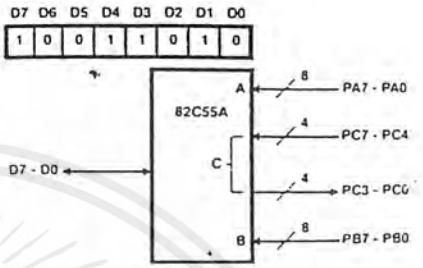


Mode 0 Configurations (Continued)

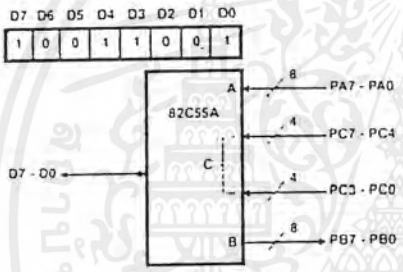
CONTROL WORD #12



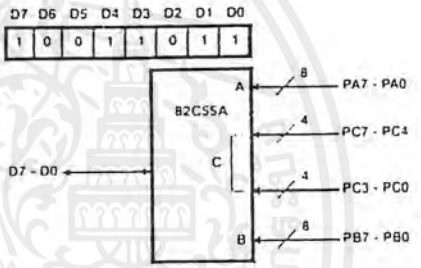
CONTROL WORD #14



CONTROL WORD #13



CONTROL WORD #15



Operating Modes

Mode 1 - (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "hand shaking" signals. In mode 1, port A and port B use the lines on port C to generate or accept these "hand shaking" signals.

Mode 1 Basic Function Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit port and one 4-bit control/data port
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit port.

Input Control Signal Definition

(Figures 6 and 7)

STB (Strobe Input)

A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch: in essence, and acknowledgment. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

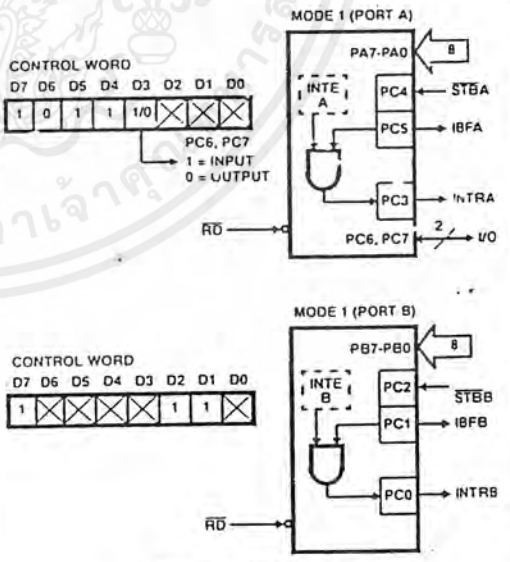


FIGURE 6. MODE 1 INPUT