หุ่นยนต์แมลงหกขา

SIX LEGGED INSECT ROBOT

โดย

นายกังวาน แสนสวาสดิ์ รหัสประจำตัว 40013248

นายสรพล บรรจงราชเสนา ณ อยุธยา รหัสประจำตัว 40013311

อาจารย์ที่ปรึกษา

ดร. ปีติเขต สู้รักษา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ชื่อโครงงานภาษาไทย	หุ่นยนต์แมลงหกขา
ชื่อโครงงานภาษาอังกฤษ	SIX LEGGED INSECT ROBOT
ผู้จัดทำ	นายกังวาน แสนสวาสดิ์
	นายสรพล บรรจงราชเสนา ณ อยุธยา
อาจารย์ที่ปรึกษา	ดร. ปิติเขต สู้รักษา
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2542

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้รับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ดร. ปิติเขต สู้รักษา)

คณะกรรมการการสอบปริญญานิพนธ์

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หุ่นยนต์แมลงหกขา

จัดทำโดย	นาย กังวาน แสนสวาสดิ์	40013284
	นาย สรพล บรรจงราชเสนา ณ อยุธยา	40013311
อาจารย์ที่ปรึกษา	ดร. ปิติเขต สู้รักษา	
ปีการศึกษา	2542	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้เสนอ การสร้างและควบคุมหุ่นยนต์หกขา ชนิดขาในการสร้างเป็น แบบ เพนโทกราฟเลก (Pantograph Leg) โดยนำ 68HC11 ไมโครคอนโทรลเลอร์มาควบคุมการทำงานของเซอร์โวมอเตอร์ ซึ่งจะทำหน้าที่ควบคุมการก้าวเดินของตัวหุ่นยนต์ให้สามารถเคลื่อนที่และหลบหลีกสิ่งกีดขวางได้ ซึ่งหุ่นยนต์หกขาที่สร้างขึ้นนี้ สามารถทำงานได้สองโหมด คือโหมดการควบคุมโดยอัตโนมัติ และ โหมดการควบคุมจากเครื่องคอมพิวเตอร์

SIX LEGGED INSECT ROBOT

By	Mr. Kangwan Saensawasd	40013284
	Mr. Sorapon Bunjongrajasena Na Ayuttaya	40013311
Adviser	Dr. Pitikhate Sooraksa	
Academic Year	1999	

Abstract

This thesis presents building and controlling six legged insect robot. The leg of the robot are Pantograph type servomoters located at the other ends. The servos are controlled by 68HC11 micro controller for motion obstacle avoidance. The robot can be operated in two modes which are an automatic control mode and a manual control mode via computer printer part

กิตติกรรมประกาศ

จากความสำเร็จในการสร้าง หุ่นยนต์แมลงหกขา คณะผู้จัดทำขอขอบพระคุณ ท่าน อาจารย์ ดร. ปิติเขต สู้รักษา (อาจารย์ที่ปรึกษา) ที่ได้ให้คำชี้แนะให้การสนับสนุน ให้ความช่วยเหลือในทุกๆด้าน และขอขอบพระคุณ ภาคเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้ให้การสนับสนุน จนกระทั่งโครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

ลงชื่อ.....

(นาย กังวาน แสนสวาสดิ์)

ลงชื่อ.....

(นาย สรพล บรรจงราชเสนา ณ อยุธยา)

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันได้มีการสร้างและพัฒนา หุ่นยนต์ในรูปแบบต่าง ๆ โดยการควบคุมการทำงานของไมโครคอนโทรลเลอร์ ซึ่งหุ่นยนต์ที่นิยมสร้างโดยส่วนมาก เป็นหุ่นยนต์ประเภทเคลื่อนที่ด้วยล้อ (Mobile Robot) แต่การเคลื่อนที่ของหุ่นยนต์จะจำกัดอยู่บนพื้นที่เรียบเท่านั้น

จึงได้มีแนวความคิดที่จะสร้างและพัฒนาหุ่นยนต์ ซึ่งสามารถเคลื่อนที่โดยใช้ขา เป็นตัวขับเคลื่อน จึงได้เลือกรูปแบบขาที่ใช้งาน ในการพยุงและขับเคลื่อนตัวหุ่นยนต์ เป็นหุ่นยนต์ชนิดหกขา ซึ่งหุ่นยนต์ชนิดที่ขับเคลื่อนด้วยขา สามารถที่จะพัฒนาให้ไปทำงานแทนมนุษย์ ในจุดที่เป็นอันตรายได้ เช่น หุ่นยนต์ทำความสะอาดกระจกบนตึกสูง หุ่นยนต์ตรวจสอบและตรวจซ่อมโครงสร้างสะพานในที่สูง หุ่นยนต์ตรวจสอบรอยรั่วบนถังก๊าซระบบการส่งจ่ายก๊าซ หรือการปิโตรเคมี เป็นต้น

1.2 ชื่อโครงงาน

หุ่นยนต์แมลงหกขา (Six Legged Insect Robot)

1.3 วัตถุประสงค์

- เพื่อสร้างหุ่นยนต์หกขา ขนาดเล็ก หนึ่งตัว
- เพื่อควบคุมหุ่นยนต์หกขาให้สามารถเคลื่อนที่ได้โดยไมโครคอนโทรลเลอร์
- เพื่อเป็นหุ่นยนต์ต้นแบบ ในการพัฒนาหุ่นยนต์หกขาในรุ่นต่อไป

1.4 ขีดความสามารถของโครงงาน

- ควบคุมหุ่นยนต์จากเครื่องคอมพิวเตอร์ให้เคลื่อนที่ได้
- ควบคุมให้หุ่นยนต์เคลื่อนที่ เดินหน้า ถอยหลัง เดินเป็นวงกลม และหมุนกลับตัวในที่แคบได้

1.5 เนื้อหาโดยสังเขป

บทที่ 2 เป็นการออกแบบและโครงสร้างของหุ่นยนต์หกขา ซึ่งกล่าวถึงการออกแบบขาหุ่นยนต์ที่ใช้งาน ลักษณะของโครงสร้างตัวหุ่นยนต์หกขา และส่วนประกอบต่างๆ

บทที่ 3 เป็นระบบการควบคุม จะกล่าวถึง ไมโครคอนโทรลเลอร์ หลักการควบคุมหุ่นยนต์หกขา control algorithum และ Solfware

บทที่ 4 จะกล่าวถึงผลการทดสอบ การควบคุมการเคลื่อนที่ ของหุ่นยนต์หกขา

บทที่ 5 จะกล่าวถึงปัญหาและข้อเสนอแนะ

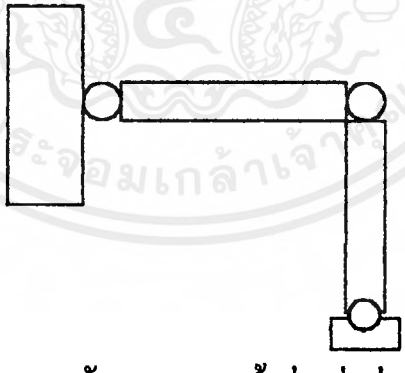
บทที่ 2
โครงสร้างหุ่นยนต์แมลงหกขา

2.1 การออกแบบขา

สำหรับการเลือกลักษณะขาของหุ่นยนต์ สิ่งที่สำคัญต่อการเลือกขาคือ จะต้องเลือกแบบที่มีลักษณะการเคลื่อนไหวมากที่สุด ให้เหมาะสมกับรูปร่างลักษณะโครงสร้างทางกายภาพของตัวหุ่นยนต์ ลักษณะการเดินของขาแต่ละแบบ จะขึ้นอยู่กับข้อจำกัดทางกายภาพของขาด้วย ซึ่งในปัจจุบันมีการออกแบบขาหุ่นยนต์อยู่หลายชนิด แต่ละชนิดจะมีคุณสมบัติเฉพาะแบบ ดังต่อไปนี้

แบบที่1 ขาสองข้อต่ออย่างง่าย (Simple Two Link Leg)

ขาชนิดนี้จะมีลักษณะเป็นสองท่อน แต่ละท่อนจะต่อผ่านข้อต่อดังรูปที่ 2.1 ซึ่งสามารถควบคุมลักษณะการเดินได้ โดยการ ควบคุมมุมของขาแต่ละท่อน ซึ่งจะเป็นตัวกำหนดตำแหน่งปลายขาของหุ่นยนต์ ส่วนของขาทั้งหมดจะต่อเข้ากับเดือยที่โคนขาเพื่อใช้ในการก้าวขาและยึดหดขา



รูปที่ 2.1 แสดงลักษณะขาสองข้อต่ออย่างง่าย

วิธีการทำงานของข้อต่อ

มีหลายวิธีที่จะทำให้ข้อต่อทำงานได้ โดยใช้ลักษณะการขับเคลื่อนของข้อต่อ สำหรับแบบนี้อาจติดตั้งมอเตอร์เข้าที่ข้อต่อโดยตรง หรือใช้โซ่ สายพาน สกรู และส่ง

กำลังจากมอเตอร์ที่ติดตั้งอยู่ในลำตัวบริเวณโคนขา เพื่อกำหนดมุมที่ข้อต่อในการก้าวเดินของขาหุ่น

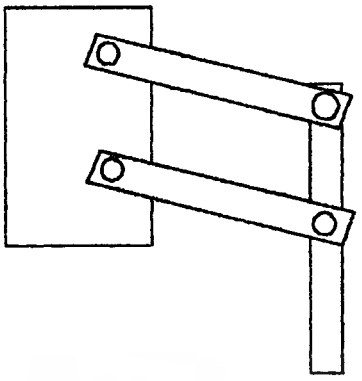
จุดด้อยประการสำคัญของขาแบบนี้คือ เราจำเป็นต้องให้ตัวขับเคลื่อน อยู่ใกล้กับข้อต่อมากที่สุด การติดตั้งตัวขับเคลื่อนเข้าที่ข้อต่อเข่าทำให้เกิดผลกระทบทางไดนามิคต่อขาหุ่น ซึ่งต้องมีการชดเชยโดยใช้ตัวควบคุม ซึ่งจะทำให้ต้องเพิ่มความซับซ้อนให้กับอัลกอริธึมในการเคลื่อนที่ของขา

รวมทั้งยังต้องการมอเตอร์ที่มีกำลังสูงที่ข้อต่อส่วนสะโพก เพื่อใช้ในการเคลื่อนขาที่มีมวลมาก ซึ่งเราก็สามารถแก้ปัญหาเหล่านี้ได้โดยการติดตั้งตัวขับเคลื่อนที่ฐานของขา แต่จะเป็นการเพิ่มความซับซ้อนทางแมคคานิกส์

แบบที่ 2 ขาเพนโทกราฟ (Pantograph Leg)

ขาแบบนี้จะประกอบด้วยคานสี่ท่อนขนานกันเป็นรูปสี่เหลี่ยมด้านขนาน ดังรูปที่ 2.2 เป็นแบบที่มีผู้ใช้กันอย่างแพร่หลาย เพราะการควบคุมและระบบทางแมคคานิกส์ของขาทำได้ง่าย ซึ่งลดการประมวลผลที่ซับซ้อนในการควบคุมลงได้

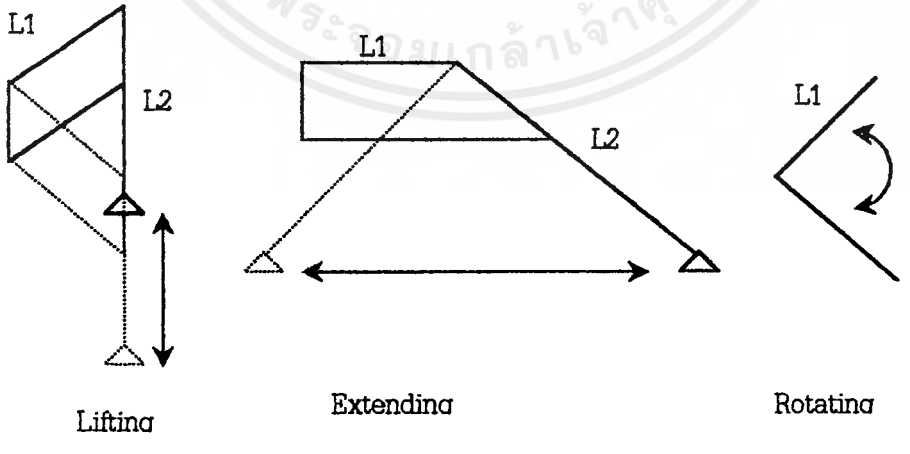
ขอบเขตการเคลื่อนไหวของขา (Workspace) แสดงให้เห็นว่าโครงสร้างแบบนี้ยังคงมีการคัปปลิ้ง (Coupling) ของข้อต่อเกิดขึ้นในการเคลื่อนปลายขา ทำให้ปลายขาเคลื่อนที่เป็นเส้นโค้ง เนื่องจากว่าขาแบบนี้เป็นแบบที่มีลักษณะทางเรขาคณิตอย่างง่ายๆ ทำให้ผู้ออกแบบมักเลือกขาแบบนี้มาใช้ก่อนแบบอื่น แต่ปัญหาทางแมคคานิกส์ที่พบระหว่างการสร้างขาต้นแบบ ทำให้เราต้องดัดแปลงขาแบบนี้อีกครั้งก่อนที่จะตัดสินใจใช้ขาแบบนี้ให้เป็นขาที่จะนำไปใช้งาน



รูปที่ 2.2 แสดงลักษณะขาแบบ เพนโทรกราฟ

วิธีการทำงานของข้อต่อ

ส่วนประกอบที่ใช้ในการควบคุมเพื่อให้ขาเกิดการเคลื่อนที่อาจใช้ กระบอกสูบ ในระบบของไฮโดรริกส์ นิวเมตริกส์ หรือ มอเตอร์ ซึ่งขึ้นอยู่กับความเหมาะสมและความต้องการที่จะนำไปใช้ ซึ่งไม่ว่าจะเป็นแบบใด ข้อควรพิจารณาที่สำคัญในระหว่างการออกแบบขา คือขนาดของขอบเขตการเคลื่อนไหวของขา (Workspace) ที่ต้องการ คือ การยกขา (Lifting), การยืดขา(Extending) และ การแกว่งขา (Rotating) ดังที่แสดงไว้ในรูปที่ 2.3



รูปที่ 2.3 แสดงการเคลื่อนที่ของขาแบบ เพนโทรกราฟ

ซึ่งมีผลโดยตรงต่อขนาดของขา การเปลี่ยนแปลงขนาดความยาวของแต่ละส่วนของขาจะทำให้ความสูงที่หุ่นยนต์สามารถยกขาได้ หรือระยะที่หุ่นยนต์สามารถยืดขาได้ เปลี่ยนไป ซึ่งส่งผลให้ขนาดของการก้าวแต่ละก้าวเปลี่ยนแปลงไปด้วย

ดังนั้นในการออกแบบสิ่งที่สำคัญที่ควรพิจารณาอีกประการหนึ่งก็คือ การพยายามให้ตัวขับเคลื่อน อยู่ติดกับฐานของขา การติดตั้งตัวขับเคลื่อนให้อยู่ที่จุดศูนย์กลางจะทำให้ผลกระทบทางไดนามิคลดลง ทั้งยังลดภาระทางไดนามิคของแต่ละขาอีกด้วย เพราะไม่ต้องเคลื่อนมวลของตัวขับเคลื่อนที่ติดตั้งอยู่ในตำแหน่งที่ห่างออกไปตามข้อต่อต่างๆ อีกทั้งวัสดุที่นำมาใช้ในการสร้างหุ่นยนต์นั้นมีอยู่หลายชนิด ซึ่งราคาก็เป็นปัจจัยอีกอย่างหนึ่ง แต่อย่างไรก็ดีน้ำหนักก็มีบทบาทที่สำคัญกว่า ที่จะต้องนำมาพิจารณาเพื่อประสิทธิภาพโดยรวมของหุ่นยนต์

2.2 การออกแบบโครงสร้างตัวหุ่นยนต์

วัสดุที่ใช้สร้าง ประกอบด้วย แผ่น PVC และ อลูมิเนียม ซึ่งในส่วนของ PVC แผ่นจะนำมาสร้างเป็นตัวหุ่น และขาหุ่น เนื่องจากมีคุณสมบัติ เหนียว ทนทาน น้ำหนักเบา และ ตกแต่งง่าย สำหรับอลูมิเนียมเรานำมาสร้างเป็นแท่นยึดเพื่อติดตั้งเซอร์โว ส่วนจุดหมุนและข้อต่อทุกจุด เราใส่ลูกปืนแบริ่งขนาด 0.25 นิ้วตามขนาดและสัดส่วนของตัวหุ่นยนต์ เพื่อลดแรงเสียดทานของจุดหมุน ทุกจุด

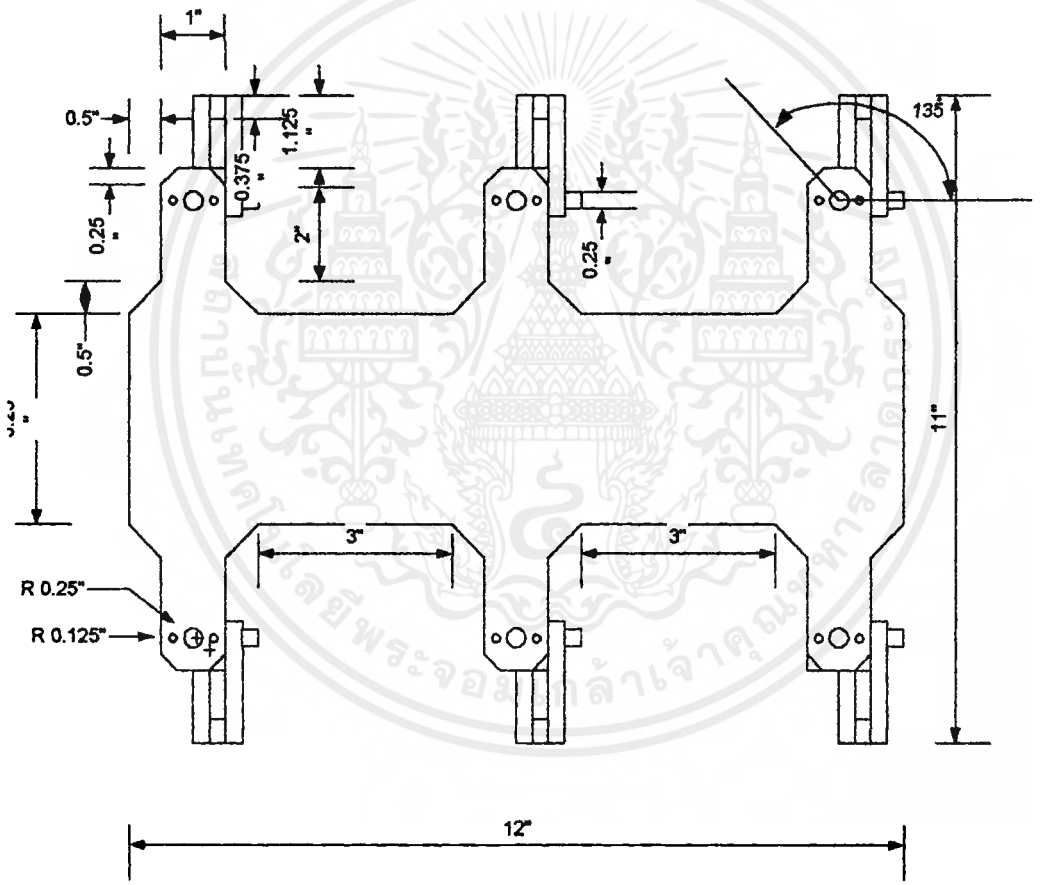
หุ่นยนต์ที่สร้างมีขนาด 5.2 × 11 ×12 นิ้ว

หุ่นยนต์มีน้ำหนัก 2 กิโลกรัม

ขาหุ่นยนต์มีรัศมีการหมุน 2 นิ้ว

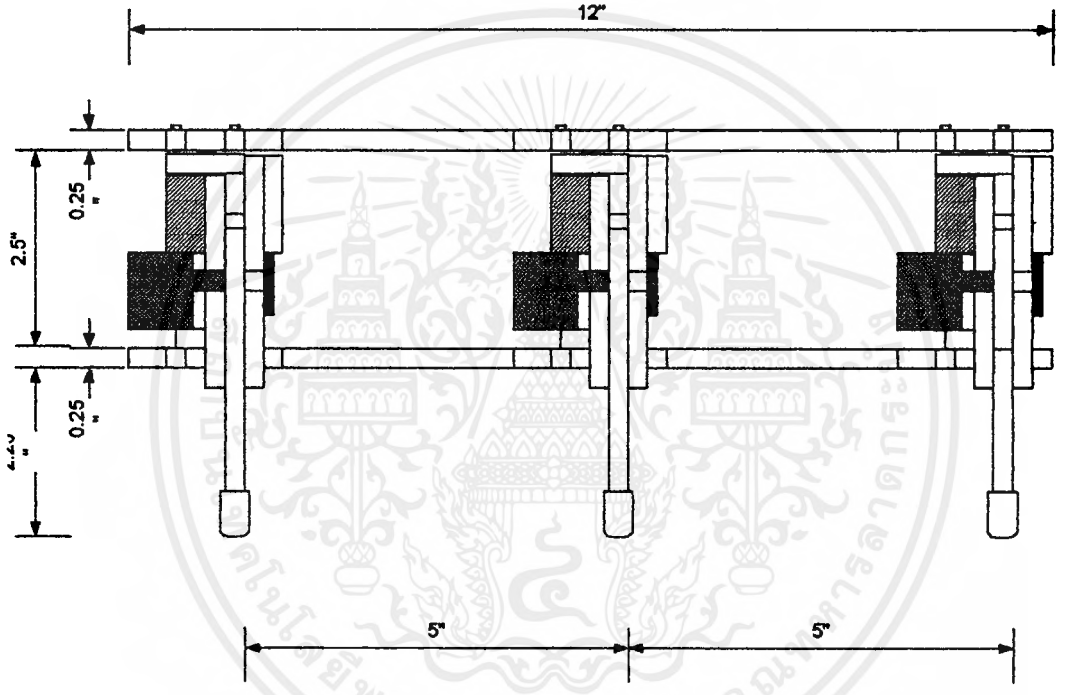
แต่ละขามีเคลื่อนไหวที่เป็นอิสระต่อกัน

แผ่นลำตัวของหุ่นยนต์มีลักษณะเป็นสี่เหลี่ยมผืนผ้า ทำหน้าที่ยึดขาทั้งหมดเข้าด้วยกันโดยจัดวางขาในด้านยาวของลำตัว ด้านบนของลำตัวมีพื้นที่สำหรับจัดวางอุปกรณ์ควบคุม ลำตัวของหุ่นมีความยาว 12 นิ้วและมีความกว้าง 6.75 นิ้ว (ไม่รวมความยาวของขา) ดังที่แสดงไว้ในรูปที่ 2.4



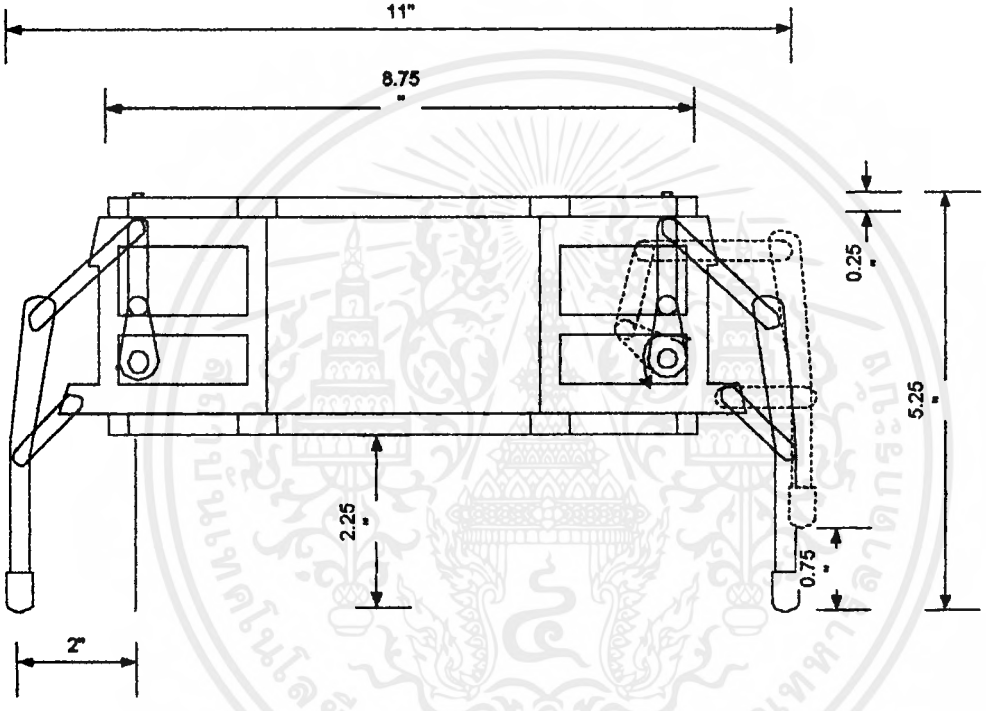
รูปที่ 2.4 แสดงภาพด้านบนของตัวหุ่นยนต์

การยึดขาเข้ากับลำตัวของหุ่นจะใช้แผ่นยึดลำตัวสองแผ่นประกบด้านบนและด้านล่างเพื่อยึดจุดหมุนส่วนขาให้มีความคงที่ หุ่นยนต์มีความสูงโดยรวม 5.25 นิ้ว ตำแหน่งแต่ละขาวางห่างกัน 5 นิ้ว ดังรูปต่อไปนี้



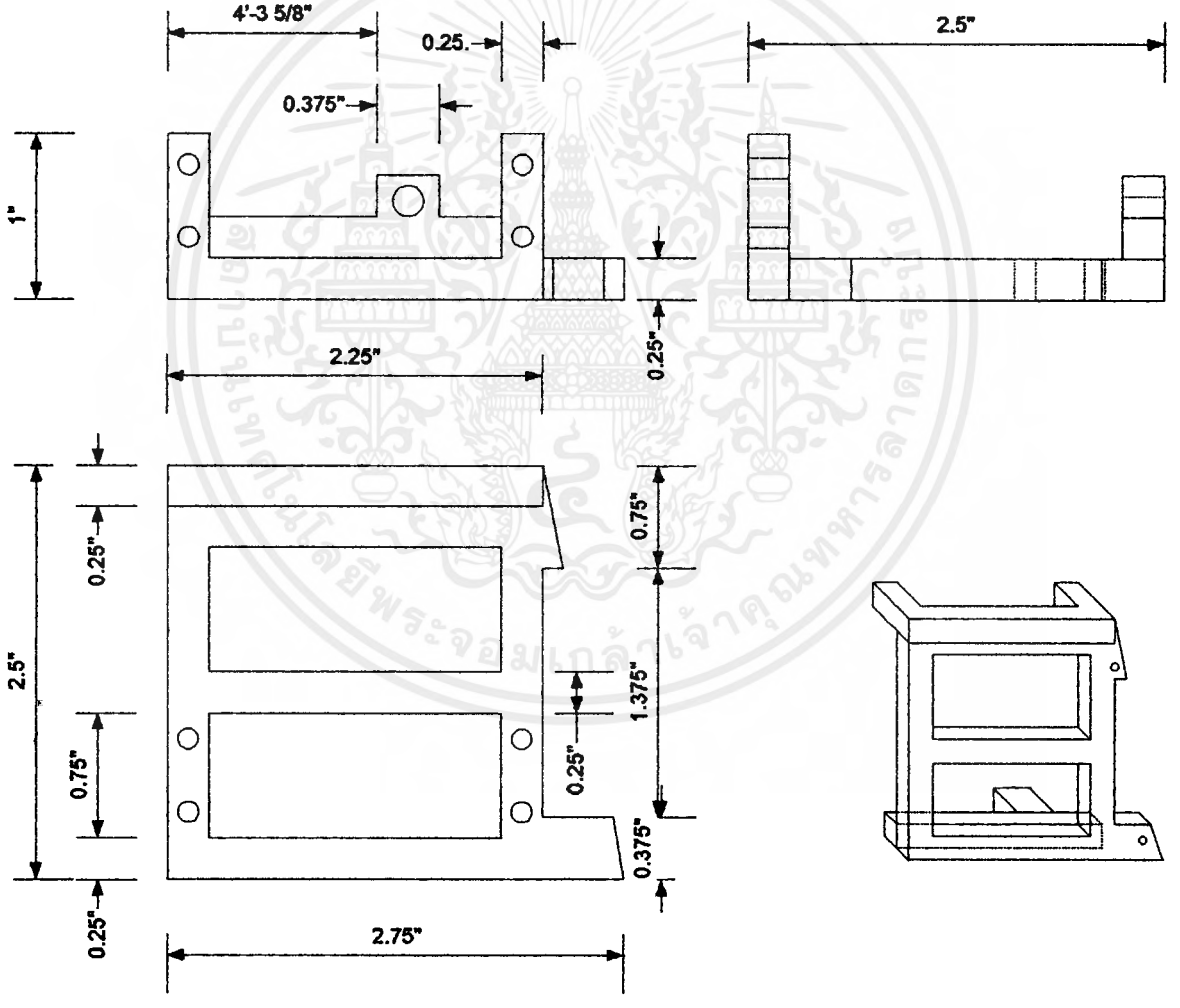
รูปที่ 2.5 แสดงภาพด้านข้างของตัวหุ่นยนต์

ช่องว่างกลางลำตัวระหว่างแผ่นยึดลำตัวทั้งสองสามารถวางแบตเตอรี่ เพื่อจ่ายกำลังงานไฟฟ้าให้กับส่วนควบคุมและส่วนขับเคลื่อนของหุ่นยนต์ได้ ลำตัวของหุ่นยนต์มีความสูงจากพื้น 2.25 นิ้ว และสามารถยกขาได้สูง 0.75 นิ้ว ดังที่แสดงไว้ตามรูปที่ 2.6

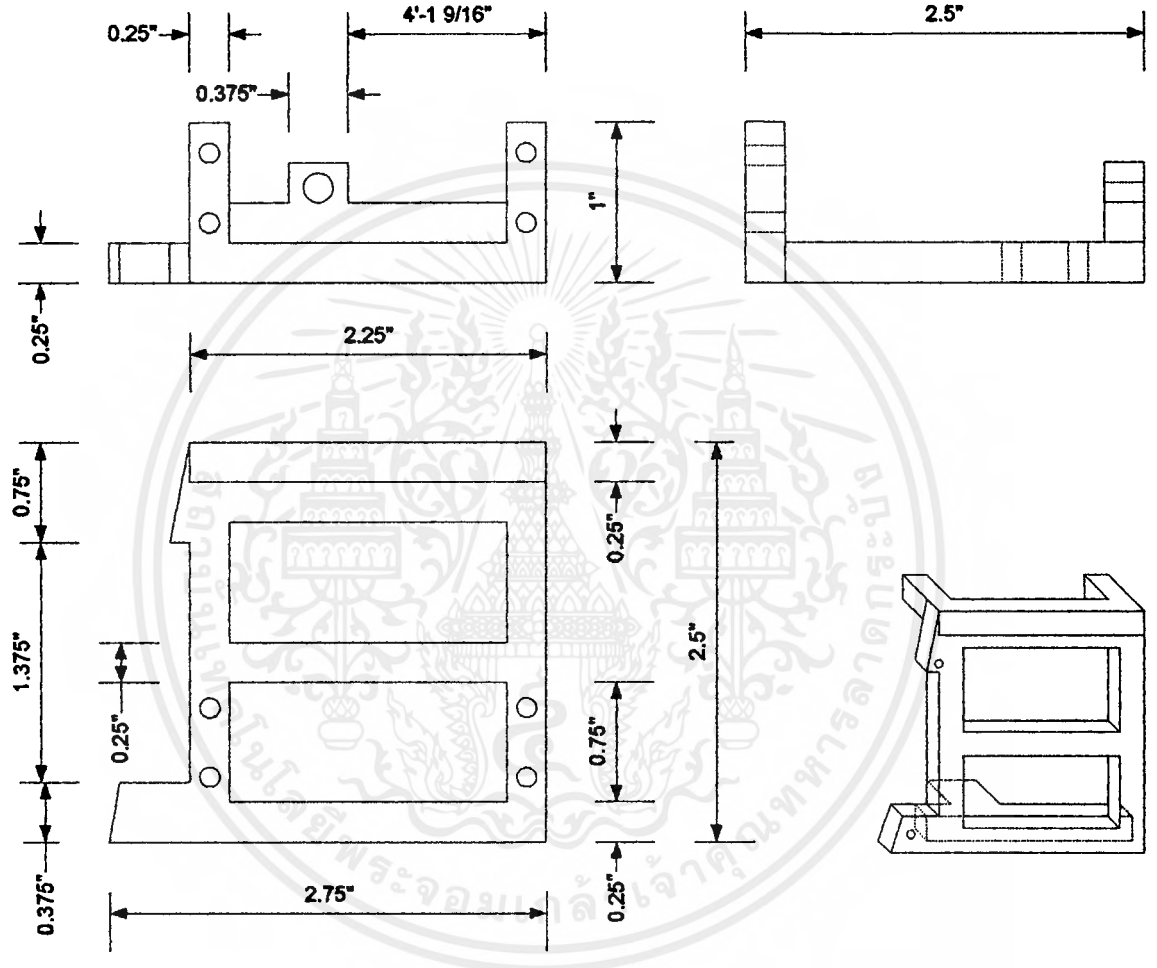


รูปที่ 2.6 แสดงภาพด้านหน้าของตัวหุ่นยนต์

แท่นยึดแกนหมุนทำหน้าที่ยึดเซอร์โวมอเตอร์สองตัวและชิ้นส่วนของขา ประกอบด้วยชิ้นส่วน 3 ชิ้น ส่วนบนจะทำหน้าที่ยึดเซอร์โวมอเตอร์ทำหน้าที่หมุนขา ส่วนหน้ายึดเซอร์โวมอเตอร์ทำหน้าที่ยกขา ส่วนล่างเป็นแกนยึดกับลำตัวด้านล่าง เนื่องจากส่วนนี้เป็นส่วนประกอบที่สำคัญและมีโครงสร้างที่ที่ซับซ้อนเมื่อเทียบกับส่วนอื่น จึงเลือกใช้อลูมิเนียมเป็นส่วนประกอบในการสร้าง ดังที่แสดงไว้ตามรูปที่ 2.7 และ รูปที่ 2.8

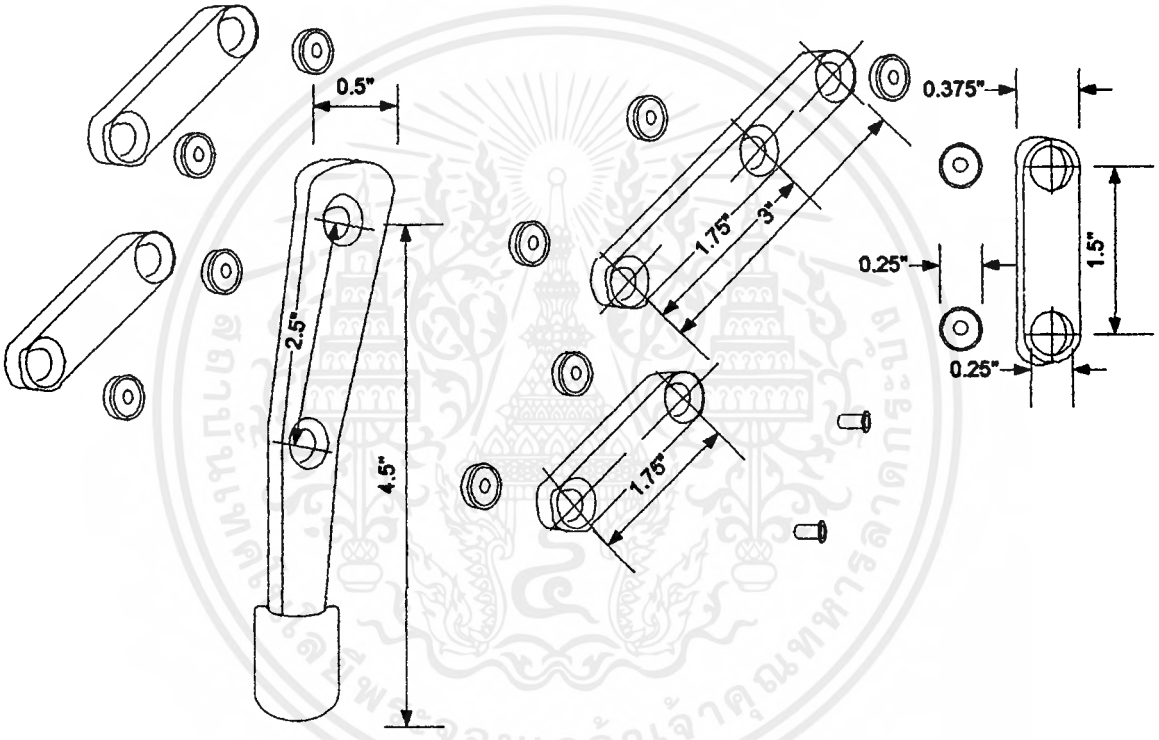


รูปที่ 2.7 แสดงภาพแท่นยึดแกนหมุนขของขาด้านซ้าย



รูปที่ 2.8 แสดงภาพแท่นยึดแกนหมุนของขาด้านขวา

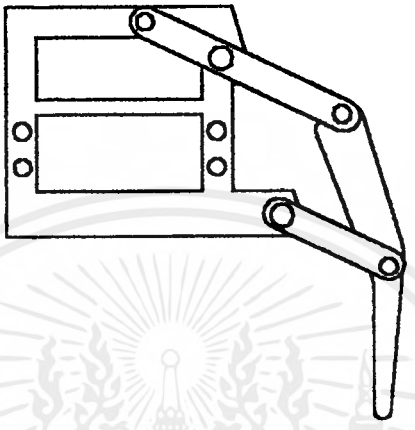
ส่วนประกอบของขาเป็นส่วนที่มีความละเอียดอ่อนที่สุด ประกอบด้วยจุดหมุน 4 จุดแต่ละจุดจะใส่ลูกปืนแบริ่งเพื่อลดแรงเสียดทานขณะหมุน ตามที่แสดงไว้ในรูปที่ 2.9โดยจะฝังลงไปในชิ้นส่วนของขา 2 จุด และแท่นยึดแกนหมุนอีก 2 จุด และชิ้นส่วนของการยึดขามีมุดสลักสอดยึดเข้ากับส่วนของหลับลูกปืน



รูปที่ 2.9 แสดงภาพส่วนประกอบของขาหุ่นยนต์

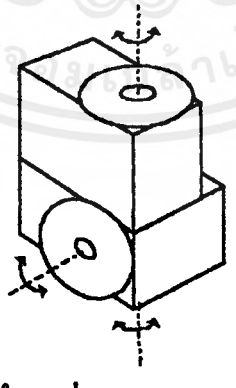
2.3 การสร้างหุ่นยนต์ 6 ขา

ในการสร้างเลือกใช้ขาแบบ Pantograph Leg ตามที่แสดงไว้ในรูปที่ 2.10 เพราะโครงสร้างของขาง่ายต่อการสร้าง ไม่ซับซ้อนยุ่งยากในระบบแม็กคานิกส์ และใช้เซอร์โวมอเตอร์เป็นตัวขับเคลื่อน



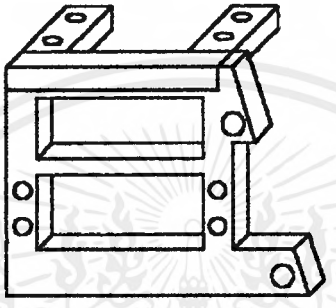
รูปที่ 2.10 แสดงรูปแบบขาของหุ่น

โดยใช้เซอร์โวมอเตอร์ตัวหนึ่งทำหน้าที่ยกขาให้ลอยขึ้นและเซอร์โวมอเตอร์อีกตัวหนึ่งทำการหมุนขาเพื่อเคลื่อนย้ายตำแหน่งเท้าของหุ่นยนต์ ตามรูปที่ 2.11

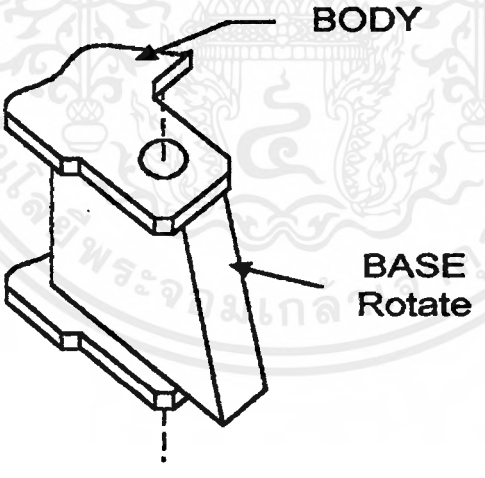


รูปที่ 2.11 แสดงตำแหน่งการวางของเซอร์โวใน 1 ขา

สำหรับเซอร์โวที่ทำหน้าที่หมุนส่วนขานั้นจะหมุนอยู่บนแกนที่ยึดติดกับตัวหุ่นตาม รูปที่ 2.12 และ รูปที่ 2.13 ซึ่งเซอร์โวมอเตอร์ทั้งสองจะถูกยึดอยู่กับส่วนที่ใช้ยึดขา และส่วนดังกล่าวจะมีด้วยกัน 6 ชุด



รูปที่ 2.12 แสดงแท่นยึดแกนหมุนติดตั้งเซอร์โวมอเตอร์



รูปที่ 2.13 แสดงการติดตั้งแทนหมุนเข้ากับลำตัวของหุ่นยนต์

3.1.3 การทำงานของเซอร์โวมอเตอร์

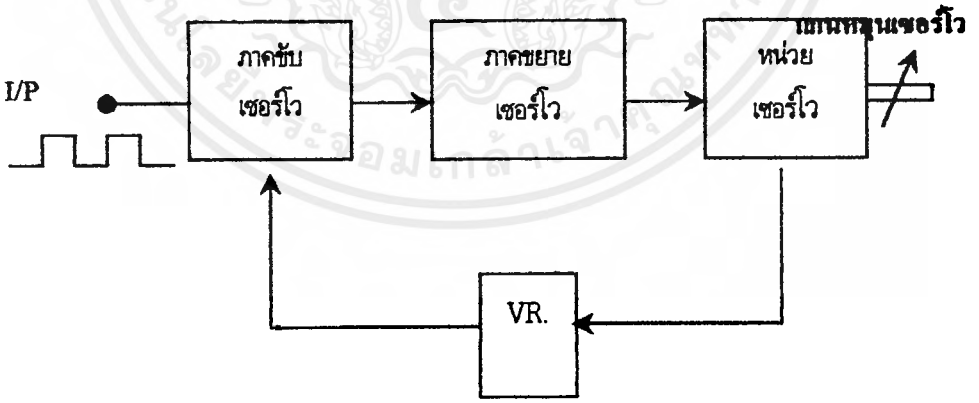
ในเซอร์โวมอเตอร์หนึ่งตัวจะประกอบไปด้วย 3 ภาคการทำงานแต่ละภาคมีหน้าที่และการทำงานดังนี้

ภาคขับเซอร์โว ประกอบด้วย วงจรสร้างสัญญาณพัลซ์ และวงจรเปรียบเทียบสัญญาณพัลซ์ที่สร้างขึ้น กับสัญญาณพัลซ์ I/P ที่รับเข้ามา

ภาคขยายเซอร์โว ประกอบด้วย วงจร RC Network ที่ช่วยหน่วงสัญญาณให้เซอร์โวสามารถทำงานได้ตลอดช่วงคาบเวลา จนกระทั่งมีสัญญาณถูกต่อไปมา รวมถึงวงจรกลับขั้วแรงดันไฟฟ้าควบคุมทิศทางการหมุนของมอเตอร์

หน่วยเซอร์โว ประกอบด้วย มอเตอร์ความเร็วสูง เฟืองทดรอบ แกนหมุน อุปกรณ์ต่างๆ และ VR (ตัวต้านทานปรับค่าได้) ทำหน้าที่ป้อนกลับตำแหน่ง (Position Feedback)

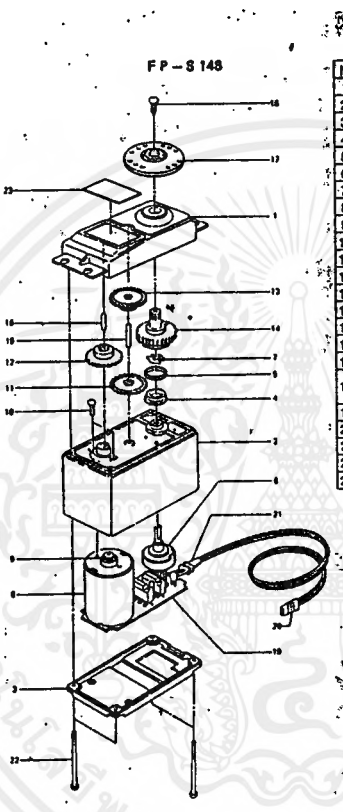
ซึ่งในขณะที่มอเตอร์หมุน VR จะถูกปรับค่า Feedback กลับมาปรับและเปรียบเทียบค่าความกว้างของพัลซ์ที่ภาคขับเซอร์โว เมื่อขนาดความกว้างของพัลซ์ มีค่าเฉลี่ยของค่าแรงดันเท่ากันมอเตอร์จะหยุดหมุนทันที ซึ่งรูปที่ 3.2 ได้แสดงภาคการทำงานของเซอร์โวมอเตอร์ตามที่ได้กล่าวไว้ในข้างต้น และได้แสดงไว้ดังรูปต่อไปนี้



รูปที่ 3.2 แสดงภาคการทำงานของเซอร์โวมอเตอร์

3.1.2 โครงสร้างและส่วนประกอบของเซอร์โว รุ่น FT 148

ภายในชุดเซอร์โวประกอบด้วย ชุดมอเตอร์ความเร็วสูง และชุดเฟืองทดรอบ จะทำให้มอเตอร์มีแรงบิดมากขึ้น ซึ่งส่วนประกอบต่างได้แสดงไว้ตามรูปที่ 3.3



No.	Part name	Part No.
1.	Upper case	1144
2.	Middle case	1144
3.	Bottom case	1144
4.	Metal bearing	S04137
5.	Metal bearing	S04137
6.	Potentiometer	139668
7.	Potentiometer drive plate	S02753
8.	Motor	S91239
9.	Motor pinion	S02461
10.	Screw	J50002
11.	1st gear	1157
12.	2nd gear	1157
13.	3rd gear	1157
14.	Final gear	1157
15.	Intermediate shaft	S02495
16.	2nd shaft	S02494
17.	Servo horn D	1168
18.	Binding head tapping screw 2.6×8	1122
19.	Printed wiring board	AS1157
20.	3PB-WRB300G	AT2453
21.	w/gum bush	S90045
22.	Pan head truss screw	J50360
23.	Nameplate	S60099

รูปที่ 3.3 แสดงส่วนประกอบภายในเซอร์โวมอเตอร์

3.2ไมโครคอนโทรเลอร์ 68HC11

3.2.1 ลักษณะโดยทั่วไปของไมโครคอนโทรลเลอร์ 68HC11

ไมโครคอนโทรลเลอร์ 68HC11 เป็นชิปไมโครคอนโทรลเลอร์เบอร์หนึ่งของโมโตโรล่า ที่ได้ถูกออกแบบมาโดยรวบอุปกรณ์ที่จำเป็นในการใช้งานของระบบไมโครคอนโทรลเลอร์ไว้ภายในตัวมันเองไม่ว่าจะเป็นพอร์ตอินพุต เอาต์พุต วงจรตั้งเวลา ตัวนับ ระบบการอินเตอร์รัปต์ หน่วยความจำรอม แรม และอีอีพรอม วงจรแปลงสัญญาณแอนาล็อกเป็นดิจิตอล ระบบป้องกันความผิดพลาดที่จะเกิดขึ้นจากโปรแกรม และส่วนติดต่อสื่อสารข้อมูลผ่านทางพอร์ตอนุกรม สามารถสรุปคุณสมบัติโดยทั่วไปของ 68HC11 ได้ 2 ลักษณะคือ ทางฮาร์ดแวร์ และซอฟต์แวร์ดังนี้

คุณสมบัติทางฮาร์ดแวร์

- ซีพียูขนาด 8 บิท
- มีหน่วยความจำรอมภายในสำหรับเก็บโปรแกรมขนาด 4.8 หรือ 12 กิโลไบต์
- มีหน่วยความจำอีอีพรอมภายในขนาด 512 ไบต์ หรือ 2 กิโลไบต์
- มีหน่วยความจำแรมภายในขนาด 192 , 256 , หรือ 512 ไบต์
- มีตัวตั้งเวลาและตัวนับขนาด 8 บิต
- มีวงจรแปลงสัญญาณแอนาล็อกเป็นดิจิตอล 8 บิต 8 ช่อง
- มีวงจรพัลส์แอกคิวมูลเลเตอร์ขนาด 8 บิต
- มีส่วนติดต่อสื่อสารข้อมูลผ่านทางพอร์ตอนุกรม (Serial Communication Interface)
- มีวงจรเชื่อต่ออุปกรณ์อนุกรม (Serial Peripheral Interface)
- มีวงจรรีลไทม์อินเตอร์รัปต์
- มีระบบวอชด็อก
- สามารถขยายตัวตั้งเวลาเป็น 16 บิตได้โดยฟังก์ชันการทำงานเพิ่มเติมคือ วงจรปรีสเกลเลอร์ที่สามารถโปรแกรมได้ 4 สเกล อินพุตสำหรับตรวจจับสัญญาณ 3 อินพุตและเอาท์พุตสำหรับวงจรเปรียบเทียบ 5 เอาท์พุต
- มีระบบอินเตอร์รัปต์ 2 ระดับจาก 21 แหล่ง
- สามารถติดต่อหน่วยความจำภายนอกได้ 64 กิโลไบต์
- สามารถต่อเชื่อมการทำงานเป็นแบบมัลติโปรเซสเซอร์ได้

คุณสมบัติทางซอฟต์แวร์

- มีชุดคำสั่งเพิ่มเติมมากกว่าไมโครโปรเซสเซอร์เบอร์ 6800 และ 6801 จึงสามารถใช้ชุดคำสั่งเดียวกับ 6800 หรือ 6801 ได้
- สามารถทำการหารเลข 16 บิตโดยได้ผลลัพธ์เป็นตัวเลข 16 บิตและเศษขนาด 16 บิต
- สามารถประมวลผลข้อมูลละเอียดถึงระดับบิต
- มีโหมดการทำงาน WAIT และโหมด STOP เพื่อประหยัดพลังงาน

3.2.2 สถาปัตยกรรมของ 68HC11

ภายในชิป ประกอบด้วยส่วนหลัก ๆ 6 ส่วนคือ

1. ซีพียู
2. หน่วยความจำ
3. แอกคิวมูเลเตอร์ – วอตช์ด็อก – ตัวตั้งเวลาหลัก
4. พอร์ตอินพุต เอาต์พุต
5. วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล
6. สโตรบ / แฮนด์เชก

ซีพียู ในส่วนนี้มีส่วนประกอบย่อยอีก 3 ส่วนคือ ส่วนควบคุมโหมดการทำงานของชิป (MODE) ส่วนกำเนิดสัญญาณนาฬิกา (CLK) และส่วนลอจิกอินเตอร์รัปต์ (INT) โดยทั้งสามส่วนนี้จะได้รับการควบคุมการทำงานจากแก่นของซีพียู

หน่วยความจำ ภายในชิป จะมีหน่วยความจำครบทั้ง 3 แบบคือ รอม แรม และอีอีพรอม ซึ่งการต่อและเรียกใช้จะขึ้นอยู่กับซีพียูเป็นหลัก

แอกคิวมูเลเตอร์ – วอตช์ด็อก – ตัวตั้งเวลาหลัก ในส่วนนี้จะมีการติดต่อกับพอร์ต A โดยใช้พอร์ต A เป็นทางผ่านของข้อมูล พัลส์แอกคิวมูลเลเตอร์จะใช้พอร์ต PA7 ในขณะที่ส่วนตัวตั้งเวลาหลักจะใช้ PA3 - PA6 ในชิปยังมีวงจรวอตช์ด็อก เพื่อช่วยให้ชิปสามารถทำงานได้อย่างต่อเนื่อง แม้ว่าจะมีการรีเซตระบบอยู่บ่อย ๆ ก็ตาม

พอร์ตอินพุต เอาต์พุต ใน 68HC11 มีพอร์ตอินพุตเอาต์พุตอยู่ด้วยกัน 5 พอร์ตดังนี้

พอร์ต A เป็นทั้งพอร์ตอินพุทและเอาท์พุต โดยมีการทำงานแยกกันคือ PA7 เป็นพอร์ตที่สามารถส่งข้อมูลได้ 2 ทิศทาง ในขณะที่ PA3-PA6 เป็นพอร์ตเอาท์พุตของตัวตั้งเวลาหลัก และ PA0-PA2 เป็นพอร์ทอินพุต

พอร์ต B เป็นพอร์ตเอาท์พุต ในขณะที่พอร์ต C เป็นพอร์ตอินพุตเอาท์พุตสามารถส่งข้อมูลได้ 2 ทิศทาง โดยที่พอร์ต C นี้จะทำหน้าที่เป็นบัสแอดเดรสไบต์ต่ำ และบัสข้อมูลด้วย โดยได้รับการควบคุมการทำงานแบบมัลติเพล็กซ์

พอร์ต D เป็นพอร์ตที่ใช้ในการถ่ายทอดสัญญาณ จากส่วนเชื่อมต่ออุปกรณ์อนุกรมและส่วนสื่อสารข้อมูลอนุกรม จึงมีลักษณะเป็นพอร์ตอินพุตเอาท์พุตที่สามารถส่งผ่านข้อมูลได้สองทิศทาง

พอร์ต E เป็นพอร์ตอินพุตสำหรับวงจรแปลงสัญญาณแอนาล็อกเป็นดิจิตอล (Analog Digital Converter)

วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล (ADC) เป็นวงจรหนึ่งที่ช่วยเสริมให้ชิปไมโครคอนโทรเลอร์มีประสิทธิภาพมากขึ้น การเรียกใช้งานวงจรส่วนนี้จะได้รับการควบคุมจากซีพียู

สโตรบ / แฮนด์เชก ในส่วนนี้จะทำงานร่วมกับส่วนขยายบัสด้วย ทั้งนี้เพื่อให้ซีพียูสามารถทำงานได้กับแอดเดรสถึง 16 บิต ที่ส่วนนี้จะมีสัญญาณที่สำคัญ ๆ อยู่ 2 สัญญาณคือ *STRB/R/$\overline{W}$* และ *STRA / AS* โดยทั้งสองสัญญาณคือ สัญญาณสโตรบเพื่อให้ชิปสามารถทำการอ่านเขียนข้อมูลได้ และส่วนนี้ยังมีวงจรแฮนด์เชก เพื่อตรวจสอบความพร้อมในการรับส่งข้อมูลของชิปกับอุปกรณ์ภายนอก

3.2.3 โหมดการทำงานของ 68HC11

68HC11 มีโหมดการทำงาน 4 โหมดดังนี้

1. โหมดซิงเกิลชิป (Single – chip Operating Mode)
2. โหมดมัลติเพล็กซ์ขยาย (Expanded Multiplexed Operating Mode)
3. โหมดบูตสแตร็ปพิเศษ (Special Bootstrap Operating Mode)
4. โหมดทดสอบพิเศษ (Special test Operating Mode)

โหมดซิงเกิลชิป

ในโหมดนี้ 68HC11 จะทำงานโดยลำพังตัวเดียว ภายใน 68HC11 มีหน่วยความจำและพอร์ตถึง 5 พอร์ต ดังนั้นเมื่อทำงานโหมดซิงเกิลชิปตัวซีพียูจะทำการเรียกข้อมูลที่อยู่ในหน่วยความจำภายนอกออกมาใช้งานและใช้พอร์ตที่มีอยู่รับหรือส่งข้อมูลออกไป

ดังนั้นในโหมดนี้ 68HC11 จะไม่ต้องการหน่วยความจำภายนอกหรือชิปพอร์ตอินพุตเอาต์พุตภายนอก เมื่อใช้งานในโหมดนี้จะช่วยลดค่าใช้จ่ายลงได้มาก แต่อย่างไรก็ตามในโหมดนี้ 68HC11 จะไม่สามารถเพิ่มหน่วยความจำหรือพอร์ตเพื่อขยายระบบได้

โหมดมัลติเพล็กซ์ขยาย

การทำงานของ 68HC11 ในโหมดนี้ จะแตกต่างกับโหมดซิงเกิลชิปโดยโปรแกรมควบคุมการทำงานของซีพียูจะอยู่ในหน่วยความจำภายนอกโดยที่พอร์ต B และ C ของ 68HC11 ทำหน้าที่เป็นบัสแอดเดรสและข้อมูล พอร์ต B ทำหน้าที่เป็นบัสแอดเดรสไบต์สูง ส่วนพอร์ต C ทำหน้าที่เป็นบัสแอดเดรสไบต์ต่ำและบัสข้อมูล

การที่พอร์ต C สามารถเป็นทั้งบัสแอดเดรสและบัสข้อมูลได้นั้น ต้องใช้กระบวนการที่เรียกว่า การมัลติเพล็กซ์ (Multiplexed) การแยกสัญญาณแอดเดรสและข้อมูลมาใช้งาน จะต้องต่อวงจรเพื่อมัลติเพล็กซ์ภายนอก

สัญญาณควบคุมที่นำมาใช้ดีมัลติเพล็กซ์สัญญาณแอดเดรสและข้อมูล ได้แก่ สัญญาณ *STROBE B / R / $\overline{W}$ (STRB / R / $\overline{W}$)* และสัญญาณ *STROBE A / ADRESS STOBE (STRA / AS)*

ในโหมดนี้ 68HC11 จะสามารถติดต่อกับหน่วยความจำภายนอกได้สูงสุดถึง 64 กิโลไบต์ แต่มีข้อเสียคือต้องใช้อุปกรณ์ต่อเพิ่มมากทำให้ค่าใช้จ่ายของระบบเพิ่มสูงขึ้น

โหมดบูตสแตร็ปพิเศษ

เป็นโหมดการทำงานที่คล้ายกับแบบซิงเกิลชิปคือ ตัวไมโครคอนโทรลเลอร์สามารถทำงานได้โดยลำพัง แต่จะมีข้อพิเศษกว่าแบบซิงเกิลชิปอยู่หลายประการ สามารถเขียนโปรแกรมแล้วโหลดลงไปในชิปได้เลย โปรแกรมที่ใช้ในการบูตจะถูกบรรจุไว้ในบูตสแตร็ปรอมจำนวน 192 ไบต์ รอมนี้จะสามารถจะสามารถเรียกข้อมูลออกมาใช้งานได้ก็ต่อเมื่อ 68HC11 ทำงานในโหมดบูตสแตร็ปนี้เท่านั้น โดยโปรแกรมบูตนี้จะเก็บไว้ที่แอดเดรส $BF40 - $BFFF

เมื่อเรียกใช้งานโปรแกรมบูต ส่วน SCI (Serial Communication Interface) จะส่งข้อมูลโปรแกรมไปยังแรมภายในชิปที่แอดเดรส $0000 - $00FF หลังจากที่ส่งข้อมูลไปที่แอดเดรส 00FF เรียบร้อยแล้ว ซีพียูจะเริ่มมาทำงานที่แอดเดรส $0000 ต่อไป

ในโหมดการทำงานแบบนี้ 68HC11 จะส่งผ่านข้อมูลผ่านทางพอร์ต SCI หลังจากทำการรีเซตข้อมูลแล้ว ส่วน SCI จะทำงานที่สัญญาณนาฬิกา E/16 (หรือมีอัตราบอดเท่ากับ 7,812 สำหรับสัญญาณนาฬิกาที่มีความถี่ 2 เมกะเฮิรตซ์)

ในโหมดนี้ยังมีลักษณะการทำงานที่ค่อนข้างพิเศษอีกประการหนึ่งคือ การป้องกันการคัดลอก โดยจะมีบิทป้องกัน (Security bit) เข้ามาเกี่ยวข้อง ถ้าหากบิตป้องกันนี้ถูกเซตเป็น "1" ที่เอาต์พุตของส่วน SCI จะส่งค่า $FF ออกไป ทำให้ข้อมูลในอีอีพรอมถูกลบ

โหมดทดสอบพิเศษ

เป็นโหมดการทำงานที่โรงงานผู้ผลิตกำหนด การทำงานในโหมดนี้จะคล้ายคลึงกับโหมดมัลติเพล็กซ์ขยาย การรีเซตและอินเตอร์รัปต์เวกเตอร์จะได้รับการเฟตช์จากหน่วนความจำภายนอกที่แอดเดรส $BFC0 - $BFFF รีจิสเตอร์ TEST1 จะได้รับการอีนาเบิลเพื่อเป็นการบ่งบอกให้ทราบว่าซีพียูพร้อมรับการตรวจสอบแล้ว

3.2.4 รีจิสเตอร์ของซีพียู

ซีพียูของไมโครคอนโทรลเลอร์ 68HC11 จะมีรีจิสเตอร์ใช้งานอยู่ 7 ตัวได้แก่

1. แอกคิวมูเลเตอร์ A และ B
2. แอกคิวมูเลเตอร์ D
3. รีจิสเตอร์อินเด็กซ์ IX
4. รีจิสเตอร์อินเด็กซ์ IY
5. รีจิสเตอร์ตัวชี้สแต็ก (Stack Pointer : SP)
6. รีจิสเตอร์โปรแกรมเคาน์เตอร์ (Program Counter : PC)
7. รีจิสเตอร์รหัสเงื่อนไข (Condition Code Register : CCR)

แอกคิวมูเลเตอร์ A และ B

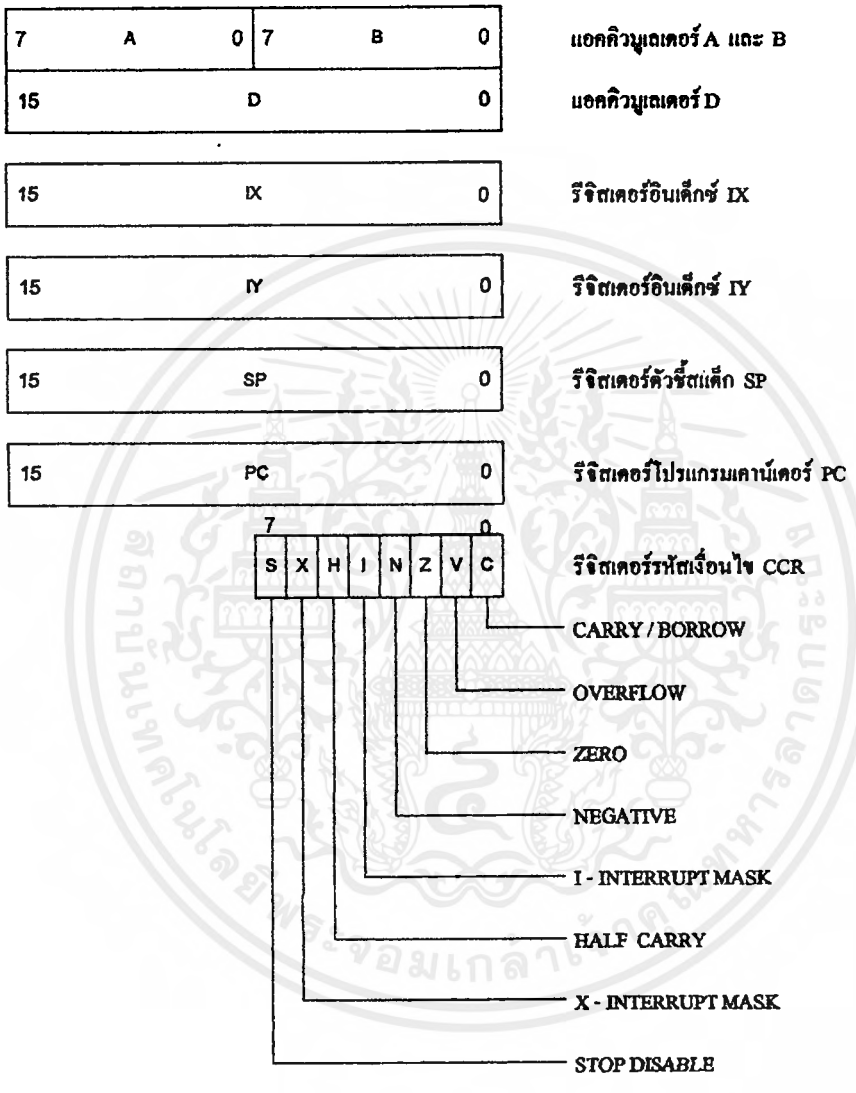
เป็นรีจิสเตอร์ขนาด 8 บิต ที่ใช้ในการเก็บค่าโอเปอแรนด์และผลของการคำนวณทางคณิตศาสตร์หรือผลจากการจัดการข้อมูลโดยตัวซีพียู ในการประมวณผลทางคณิตศาสตร์หรือลอจิกจะต้องนำข้อมูลเหล่านั้นมาเก็บในรีจิสเตอร์ทั้งสองตัวนี้ จึงจะสามารถประมวลผลได้

แอกคิวมูเลเตอร์ D

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้ในการประมวลผลและเก็บค่าจากการคำนวณทางคณิตศาสตร์และลอจิก แอกคิวมูเลเตอร์ D เกิดจากการรวมกันของ แอกคิวมูเลเตอร์ A และ B จึงทำให้มีขนาด 16 บิต

รีจิสเตอร์อินเด็กซ์ IX

เป็นรีจิสเตอร์ขนาด 16 บิตใช้ในการชี้ตำแหน่งแอดเดรส เพื่อเข้าไปจัดการประมวลผลกับข้อมูลในแอดเดรสนั้น ๆ นอกจากนั้น IX สามารถใช้เป็นตัวนับหรือรีจิสเตอร์เก็บข้อมูลชั่วคราวได้ด้วย



รูปที่ 3.4 การจัดจำนวนบิตและความหมายของรีจิสเตอร์ในซีพียู

รีจิสเตอร์อินเด็กซ์ IY

เป็นรีจิสเตอร์ขนาด 16 บิต มีหน้าที่เหมือนกับ IX แต่จะแตกต่างกันตรงที่ในทุกคำสั่งที่ต้องเกี่ยวข้องกับ IY จะต้องมีข้อมูลไบต์พิเศษของรหัสแมชชีน และมีไซเกิลพิเศษของ

ช่วงเวลาในการเอ็กซีคิวต์คำสั่งด้วย จึงทำให้คำสั่งที่มี IY ไปเกี่ยวข้องต้องมีขนาดเพิ่มขึ้นอย่างน้อย 2 ไบต์ขึ้นไปหรือที่เรียกว่า พรีไบต์ (Prebyte)

รีจิสเตอร์ตัวชี้สแต็ก : SP

เป็นรีจิสเตอร์ขนาด 16 บิต ใช้เก็บแอดเดรสบนสแต็ก (Stack) โดยในสแต็กใน 68HC11 นี้จะมีลักษณะการเก็บข้อมูลเข้าและนำข้อมูลออกมาเป็นแบบ LIFO (Last – In – First – Out) หรือข้อมูลที่เข้าไปเก็บในสแต็กหลังสุด เมื่อจะเรียกออกมาจะถูกเรียกออกมาก่อน สแต็กใช้เก็บข้อมูลของรีจิสเตอร์ เมื่อจะเรียกออกมาจะถูกเรียกออกมาก่อน สแต็กใช้เก็บข้อมูลของรีจิสเตอร์ เมื่อต้องมีการนำรีจิสเตอร์ตัวนั้นไปทำงานในโปรแกรมย่อยอื่น หรือต้องไปใช้ในการตอบสนองการอินเตอร์รัปต์เพื่อเป็นการป้องกันข้อมูลเดิมสูญหายจึงต้องเก็บข้อมูลนั้นไว้ในหน่วยความจำสำรองแห่งหนึ่ง ซึ่งก็คือสแต็กนั้นเอง

ทุกครั้งที่มีการเก็บข้อมูลลงแต็กค่าของ SP จะลดลง ในทางตรงกันข้ามถ้าเรียกข้อมูลออกจากสแต็กค่าของ SP จะเพิ่มขึ้น

รีจิสเตอร์โปรแกรมเคาน์เตอร์ : PC

เป็นรีจิสเตอร์ 16 บิต ใช้เก็บค่าของแอดเดรสของคำสั่งถัดไปที่ซีพียูจะไปทำการเอ็กซีคิวต์

รีจิสเตอร์รหัสเงื่อนไข : CCR (Condition Code Register)

เป็นรีจิสเตอร์ขนาด 8 บิต ในแต่ละบิตจะแสดงความหมายของผลจากการกระทำคำสั่งที่เพิ่งจะเอ็กซีคิวต์ไปของซีพียู ดังแสดงในรูป แต่ละบิตของ CCR เป็นอิสระต่อกันจึงสามารถตรวจสอบสถานะได้โดยใช้โปรแกรม และยังสามารถนำผลการตรวจสอบนั้นไปดำเนินการต่อได้ด้วย

รายละเอียดของแต่ละบิตใน CCR มีดังนี้

บิต Carry/Borrow (C) : บิตนี้จะเซตเมื่อซีพียูทำการประมวลผลทางคณิตศาสตร์แล้วเกิดการทดค่า (Carry) หรือยืมค่า (Borrow) บิตนี้สามารถที่จะใช้งานร่วมกับคำสั่งการหมุน (Rotate) และเลื่อนบิต (Shift) ข้อมูลได้ หรือที่เรียกว่า บิตทด

บิต Over Flow (V) : บิตนี้จะเซตเป็น "1" เมื่อซีพียูกระทำคำสั่งคณิตศาสตร์แล้วเกิดค่าเกินออกมา นอกเหนือจากเงื่อนไขดังกล่าวบิตนี้จะเป็น "0"

บิต Zero (Z) : บิตนี้จะเซตเมื่อผลของการกระทำทางคณิตศาสตร์หรือลอจิก การประมวลผลข้อมูลแล้วทำให้เกิดเป็นค่าศูนย์ขึ้นมา

บิต Negative (N) : หากผลของการกระทำทางคณิตศาสตร์หรือลอจิก การประมวลผลข้อมูลแล้ว ทำให้เกิดค่าเป็นลบ บิตนี้จะเซต ผลลัพธ์ที่เป็นลบสามารถสังเกตได้จากบิตที่มีนัยสำคัญสูงสุด (MSB) มีค่าเป็น "1"

บิต I – Interrupt Mask (I) : สามารถเซตบิตนี้ให้เป็น "1" ได้ 2 วิธีคือโดยวิธีการทางฮาร์ดแวร์ และโดยการใช้คำสั่งที่ใช้ในการดิสเอเบิลการอินเตอร์รัปต์แบบมาสเดเบิลทั้งภายในและภายนอก

บิต Half Carry (H) : จะเซตเป็น "1" เมื่อซีพียูได้รับคำสั่งทางคณิตศาสตร์แล้วเกิดการทดข้ามจากบิตที่ 3 มายังบิตที่ 4

บิต X-Interrupt Mark (X) : บิตนี้จะถูกเซตด้วยวิธีการทางฮาร์ดแวร์เท่านั้น โดยการป้อนสัญญาณเข้าที่ขา RESET และ XIRQ และจะรีเซตบิตนี้ด้วยการใช้คำสั่ง TAP หรือ RTI เท่านั้น

บิต Stop Disable (S) : บิตนี้จะเซตเมื่อต้องการดิสเอเบิลคำสั่ง STOP และถ้ารีเซตบิตนี้ ก็จะเป็นการอีนาเบิลคำสั่ง STOP บิตนี้ถูกควบคุมโดยโปรแกรม และเมื่อบิตนี้ถูกเซตจะทำให้คำสั่ง STOP มีผลเช่นเดียวกับคำสั่ง NOP

3.2.5 การอ้างแอดเดรสของ 68HC11

68HC11 มีกระบวนการอ้างแอดเดรส (Addressing) ทั้งสิ้นถึง 6 โหมดด้วยกัน ประกอบด้วย การอ้างแอดเดรสแบบทันทีทันใด (Immediate) , แบบโดยตรง (Direct) แบบขยาย (Extended) , แบบอินเด็กซ์ (Index) , แบบอินเฮียเรนต์ (Inherent) และแบบสัมพัทธ์ (Relative) ในบางคำสั่งเมื่อถูกใช้ในการอ้างแอดเดรสบางแบบ จำเป็นต้องเพิ่มข้อมูลเข้าไปอีก 1 ไบต์ก่อนออปโค้ด เพื่อปรับให้ 68HC11 สามารถทำงานในลักษณะมัลติเพจออปโค้ดแมป (Multi-page opcode map) ได้เหมาะสมขึ้น ข้อมูลไบต์นั้นจะเรียกว่าพรีไบต์ (Prebyte)

การอ้างแอดเดรสแบบทันทีทันใด (Immediate addressing)

ในการอ้างแอดเดรสแบบนี้เป็นการติดต่อเพื่อจัดการข้อมูล ซึ่งเป็นค่าใดๆ โดยตรงในทันทีทันใด จำนวนไบต์ของคำสั่งในการอ้างแอดเดรสแบบนี้จะมีขนาดตั้งแต่ 2-4 ไบต์ ขึ้นอยู่กับขนาดของรีจิสเตอร์ที่ต้องเกี่ยวข้องด้วย เช่น ถ้าเป็นแอกคิวมูเลเตอร์ A ก็จะมีจำนวนไบต์ของคำสั่ง 2 ไบต์ (1 ไบต์แรกเป็นออปโค้ด อีก 1 ไบต์หลังเป็นขนาดของโอเปอแรนด์ ซึ่งเทากับขนาดของรีจิสเตอร์แอกคิวมูเลเตอร์ A) รูปแบบของคำสั่งที่มีการอ้างแอดเดรสแบบนี้ หลังจาก

คำสั่งแล้วต้องตามด้วยเครื่องหมาย # เสมอเพื่อเป็นการบ่งบอกให้ซีพียูทราบว่า คำสั่งที่จะกระทำต่อไปนี้ใช้การอ้างแอดเดรสแบบทันทีทันใด

การอ้างแอดเดรสแบบโดยตรง (Direct addressing)

เป็นการติดต่อเพื่อประมวลผลข้อมูลขนาด 8 บิตที่อยู่ในหน่วยความจำแรมภายในชิป ซึ่งมีอยู่ 256 ไบต์ โดยมีแอดเดรสตั้งแต่ $0000 - $00FF ดังนั้นค่าโอเปอแรนด์ที่ตามหลังออปโค้ดคำสั่งก็คือ ค่าแอดเดรสของแรมนั่นเอง

ในการอ้างแอดเดรสแบบนี้ บางทีเรียกว่า การอ้างแอดเดรสเพจศูนย์ (Zero page addressing)

การอ้างแอดเดรสแบบขยาย (Extended addressing)

ในการอ้างแอดเดรสแบบนี้ ข้อมูลในไบต์ที่ 2 และ 3 ที่ตามหลังออปโค้ด จะเก็บค่าแอดเดรสจริงของหน่วยความจำที่ต้องการนำข้อมูลในหน่วยความจำออกมาประมวลผลหรือจัดเก็บข้อมูลลงไปใหม่ เมื่อใช้การอ้างแอดเดรสแบบนี้คำสั่งจะมีขนาด 3-4 ไบต์ โดยเป็นออปโค้ดไบต์ที่ 1 หรือ 2 (กรณีถ้ามีขนาด 4 ไบต์) ส่วน 2 ไบต์หลังจะเป็นค่าแอดเดรสที่อ้างถึงในหน่วยความจำ

การอ้างแอดเดรสแบบอินเด็กซ์ (Indexed addressing)

การอ้างแอดเดรสแบบนี้จะมีการนำรีจิสเตอร์ชี้ข้อมูล (Index register) ทั้ง 2 ตัวคือ IX และ IY มาใช้ในการคำนวณค่าแอดเดรสที่ต้องการเรียกใช้ข้อมูล ดังนั้นค่าแอดเดรสที่ต้องการใช้งานจะเปลี่ยนแปลง หรือขึ้นอยู่กับองค์ประกอบหลัก 2 ประการคือ

1. ค่าแอดเดรสที่ถูกกำหนดอยู่ในปัจจุบัน
2. ค่าออฟเซต 8 บิต ที่บรรจุอยู่ในคำสั่ง (หรือค่าโอเปอแรนด์)

ในการอ้างแอดเดรสแบบนี้ สามารถที่จะใช้หน่วยความจำที่ตำแหน่งใดก็ได้ในจำนวน 6 กิโลไบต์เป็นจุดอ้างอิง

ส่วนในด้านขนาดของคำสั่งในการอ้างแอดเดรสแบบนี้ ถ้าใช้รีจิสเตอร์ IX จะมีขนาด 2 ไบต์ โดยไบต์แรกเป็นออปโค้ด ส่วนไบต์ที่สองเป็นค่าออฟเซต หากใช้รีจิสเตอร์ IY จะมีขนาด 3 ไบต์ ไบต์แรกก็คือ พรีไบต์ ตามด้วยออปโค้ดและค่าออฟเซตขนาด 8 บิต

การอ้างแอดเดรสแบบอินเฮียรนต์ (Inherent addressing)

การอ้างแอดเดรสแบบนี้จะไม่ยุ่งเกี่ยวกับข้อมูลในหน่วยความจำแต่อย่างใด แต่จะเข้าไปจัดการในรีจิสเตอร์แทน ดังนั้นขนาดของคำสั่งในการอ้างแอดเดรสแบบนี้จะมีเพียง 1-2 ไบต์ โดยเป็นออปโค้ดทั้งสิ้นไม่มีโอเปอแรนด์

การอ้างแอดเดรสแบบสัมพันธ์ (Relative addressing)

การอ้างแอดเดรสแบบนี้จะใช้ในชุดคำสั่งกระโดด (Jump and branch instructions) ถ้าหากเงื่อนไขในการกระโดดเป็นจริง ค่าออฟเซตขนาด 8 บิต ที่อยู่ตามหลังออปโค้ด ก็จะถูกบวกเข้าไปในรีจิสเตอร์โปรแกรมเคาน์เตอร์ (PC) เพื่อกำหนดแอดเดรสต่อไปที่จะข้ามไปทำงานของซีพียู ปกติแล้วขนาดของคำสั่งในการอ้างแอดเดรสแบบนี้จะเท่ากับ 2 ไบต์โดยไบต์แรกเป็นออปโค้ด ส่วนไบต์ที่สองเป็นค่าออฟเซตเพื่อบอกจำนวนที่จะเข้าไปเพิ่มใน PC

พรีไบต์ (Prebyte)

เป็นข้อมูลขนาด 8 บิต (1 ไบต์) ที่ใส่เข้าไปก่อนหน้าออปโค้ด เพื่อประโยชน์ในการบอกให้ซีพียูทราบถึงลักษณะของการจัดเพจของออปโค้ด โดยถ้าหาก 68HC11 ทำงานในเพจ 1 ข้อมูลพรีไบต์ก็ไม่ต้องมี แต่ถ้าทำงานในเพจ 2 จะต้องเพิ่มค่าพรีไบต์เท่ากับ $18 เข้าไปก่อนออปโค้ดเสมอ แล้วตามด้วยออปโค้ด ในกรณีเพจ 3 จะใช้ค่า $1A และใช้ ค่า $CD สำหรับเพจ 4 ค่าพรีไบต์จะเข้าไปเกี่ยวข้อง เมื่อมีคำสั่งใดๆ ก็ตามกำหนดให้รีจิสเตอร์ IY ทำงาน

3.2.6 ชุดคำสั่งของ 68HC11

ซีพียูภายในไมโครคอนโทรลเลอร์ 68HC11 มีลักษณะการทำงานคล้ายๆ กับซีพียู MC6801 แต่ได้มีการเพิ่มเติมความสามารถของ 68HC11 นี้ด้วยการเพิ่มรีจิสเตอร์และคำสั่งให้มากขึ้น เช่น เพิ่มรีจิสเตอร์อินเด็กซ์ขนาด 16 บิต อีก 1 ตัวคือ IY เพิ่มคำสั่งเกี่ยวกับการหารเลข 16 บิตอีก 2 แบบ เพิ่มคำสั่งควบคุมและเพิ่มคำสั่งจัดการข้อมูลระดับบิต (Bit manipulation instruction)

ในที่นี้จะแบ่งชุดคำสั่งของ 68HC11 ออกเป็น 7 กลุ่มย่อยๆ ดังนี้

1. กลุ่มคำสั่งจัดการเกี่ยวกับข้อมูล (Data handling instructions)
2. กลุ่มคำสั่งทางคณิตศาสตร์ (Arithmatics instructions)
3. กลุ่มคำสั่งทางลอจิก (Logic instructions)
4. กลุ่มคำสั่งการกระโดด (Jump and branch instructions)
5. กลุ่มคำสั่งเปรียบเทียบและตรวจสอบข้อมูล (Data test instructions)
6. กลุ่มคำสั่งจัดการกับรีจิสเตอร์รหัสเงื่อนไข (CCR instructions)
7. กลุ่มคำสั่งควบคุม (Control instructions)

รายละเอียดการใช้งานของชุดคำสั่งสามารถศึกษาเพิ่มเติมได้จากหนังสือการใช้งานไมโครคอนโทรเลอร์ 68HC11

3.2.7 พอร์ตขนานอินพุตเอาต์พุต

ในตัวไมโครคอนโทรลเลอร์ 68HC11 มีขาอินพุตเอาต์พุตทั้งสิ้น 40 ขา กำหนดให้เป็นพอร์ตขนาด 8 บิตได้ 4 พอร์ต และพอร์ตขนาด 6 บิตอีก 1 พอร์ต ส่วนอีก 2 ขาที่เหลือเป็นขาสัญญาณสโตรบ แต่ละขาของแต่ละพอร์ตสามารถทำงานได้หลายหน้าที่ขึ้นอยู่กับการกำหนดโหมดการทำงานของชิปและข้อมูลในรีจิสเตอร์ควบคุม ในบทนี้จะอธิบายเฉพาะหน้าที่ของการเป็นพอร์ตขนานอินพุตเอาต์พุตสำหรับส่งผ่านข้อมูลธรรมดาเท่านั้น

แนะนำพอร์ตขนานอินพุตเอาต์พุต

จากคุณสมบัติดังกล่าวข้างต้นที่ว่า 68HC11 มีพอร์ต 5 พอร์ต ในการแนะนำพอร์ตในหัวข้อนี้จะเริ่มต้นจากพอร์ต A

พอร์ต A : หน้าที่ของพอร์ตนี้มีหลายแบบ ได้แก่ เป็นขาอินพุตเอาต์พุตปกติ เป็นขาสัญญาณในระบบตัวตั้งเวลาหลักและพัลส์แอกคิวมูเลเตอร์ เป็นต้น ที่พอร์ต A นี้มีอยู่ 3 ขา สัญญาณที่ถูกกำหนดให้เป็นขาอินพุตได้เท่านั้น ส่วนที่ถูกกำหนดเป็นขาเอาต์พุตมี 4 ขา ที่เหลืออีก 1 ขาเป็นขาสัญญาณที่สามารถเป็นได้ทั้งอินพุตและเอาต์พุต ขาที่ว่านั้นคือ PA7 ทิศทางการส่งผ่านข้อมูลของ PA7 จะถูกควบคุมโดยข้อมูลในบิตที่ 7 ของรีจิสเตอร์ควบคุมพัลส์แอกคิวมูเลเตอร์ (PACTL) โดยบิตที่ 7 นั้นมีชื่อว่า DDRA7 ข้อมูลที่ต้องการอ่านและเขียนของพอร์ต A จะถูกเก็บที่รีจิสเตอร์ PORTA

พอร์ต B : ถูกกำหนดให้เป็นพอรต์เอาต์พุต เมื่อทำงานในโหมดซิงเกิลชิปและบูตสแตร์ปพิเศษ ข้อมูลที่ต้องการส่งออกมาที่พอร์ตนี้ ต้องเขียนเข้าไปเก็บไว้ที่รีจิสเตอร์ PORTB และจะส่งออกไปก็ต่อเมื่อมีสัญญาณ STRB เกิดขึ้น นั่นคือในการส่งข้อมูลพอร์ต B จะมีการแฮนด์เชกระหว่างอุปกรณ์ที่ต่ออยู่กับพอร์ตและตัวไมโครคอนโทรลเลอร์ แต่ถ้าหาก 68HC11 ทำงานในโหมดมัลติเพล็กซ์ขยายหรือโหมดทดสอบพิเศษ พอร์ต B จะทำหน้าที่เป็นบัสแอดเดรสไบต์สูง (A8-A15) เพื่อติดต่อกับหน่วยความจำภายนอกชิป

พอร์ต C : เมื่อทำงานในโหมดซิงเกิลชิปและบูตสแตร์ปพิเศษ พอร์ต C จะเป็นพอร์ตอินพุตเอาต์พุตสามารถส่งผ่านข้อมูลได้ 2 ทิศทาง โดยแต่ละขาของพอร์ต (หรือจะเรียกว่าแต่ละบิตก็ได้) จะเป็นอิสระต่อกันในด้านการกำหนดทิศทาง ในการกำหนดทิศทางของขาพอร์ต C แต่ละขานั้น สามารถกำหนดได้ที่รีจิสเตอร์ควบคุมทิศทางข้อมูลพอร์ต C (Data Direction for Port C : DDRC) ถ้าบิตใดในรีจิสเตอร์ DDRC ถูกกำหนดให้เป็น "1" ขาของพอร์ต C ที่ตำแหน่งเดียวกันนั้นจะเป็นขาพอร์ตเอาต์พุต และถ้าต้องการให้เป็นอินพุตก็เขียนข้อมูล "0" ไปยังบิตที่ต้องการในรีจิสเตอร์ DDRC

พอร์ต D : เป็นพอร์ตอินพุตเอาต์พุตขนาด 6 บิต PD0 และ PD1 จะถูกกำหนดให้ทำหน้าที่เป็นขารับส่งข้อมูลอนุกรม เมื่อให้ 68HC11 ทำการสื่อสารข้อมูลอนุกรม (SCI) ส่วนอีก 4 ขาที่เหลือจะนำมาใช้ในการเชื่อมต่อกับอุปกรณ์อนุกรม (SPI) การกำหนดทิศทางการส่งผ่านข้อมูลของพอร์ต ทำได้โดยเขียนข้อมูลไปยังรีจิสเตอร์ควบคุมทิศทางข้อมูลพอร์ต D (DDRD) ส่วนข้อมูลที่ต้องการจะส่งเข้ามาหรือส่งออกไปทางพอร์ต D ให้เขียนมาเก็บไว้ที่รีจิสเตอร์ PORTD

พอร์ต E : ถูกกำหนดให้พอร์ตอินพุตขนาด 8 บิต และใช้เป็นพอร์ตอินพุตของวงจรแปลงสัญญาณแอนาล็อกเป็นดิจิตอล (ถ้าเป็นรุ่น 48 ขา พอร์ต E นี้จะมีเพียง 4 ขา)

รีจิสเตอร์ที่เกี่ยวข้องกับพอร์ตขนานอินพุตเอาต์พุต

68HC11 มีรีจิสเตอร์อยู่หลายตัวเกี่ยวข้องกับพอร์ตขนานอินพุตขนานอินพุตเอาต์พุต ได้แก่

รีจิสเตอร์เก็บข้อมูลของพอร์ต

มีด้วยกัน 5 ตัว เท่ากับจำนวนพอร์ตที่มีคือ

รีจิสเตอร์ PORTA มีแอดเดรสอยู่ที่ $1000 ใช้งานได้ 8 บิต

รีจิสเตอร์ PORTB มีแอดเดรสอยู่ที่ $1004 ใช้งานได้ 8 บิต

รีจิสเตอร์ PORTC มีแอดเดรสอยู่ที่ $1003 ใช้งานได้ 8 บิต

รีจิสเตอร์ PORTD มีแอดเดรสอยู่ที่ $1008 ใช้งานได้ 6 บิต อีก 2 บิตต้องกำหนดเป็น "0"

รีจิสเตอร์ PORTE มีแอดเดรสอยู่ที่ $100A ใช้งานได้ 8 บิต (ถ้าเป็นรุ่น 48 ขา จะใช้ได้ 4 บิต)

รีจิสเตอร์ PORTCL มีแอดเดรสอยู่ที่ $1005 ใช้งานได้ 8 บิต ทำหน้าที่แลตช์ข้อมูลที่ต้องการส่งออก หรือรับเข้ามาของพอร์ต C จะทำงานร่วมกับสัญญาณสโตรบเพื่อการแฮนด์เชกข้อมูล

รีจิสเตอร์ควบคุมทิศทางของพอร์ต

มี 2 ตัวใช้สำหรับเก็บข้อมูลเพื่อบอกลักษณะของพอร์ตที่บิตใดๆ ว่า เป็นขาของพอร์ตอินพุตหรือเอาต์พุต ได้แก่ รีจิสเตอร์ DDRC และ DDRD

รีจิสเตอร์ DDRC มีแอดเดรสอยู่ที่ $1007 ใช้ควบคุมทิศทางข้อมูลของพอร์ต C สมมติว่า ถ้าบิต 0 เป็น "1" บิตที่เหลือเป็น "0" หมด จะทำให้ขาพอร์ต C บิตที่ 0 (PC0) เป็นขาเอาต์พุตเพียงขาเดียว ที่เหลือ (PC1-PC7) จะเป็นขาอินพุตทั้งหมด

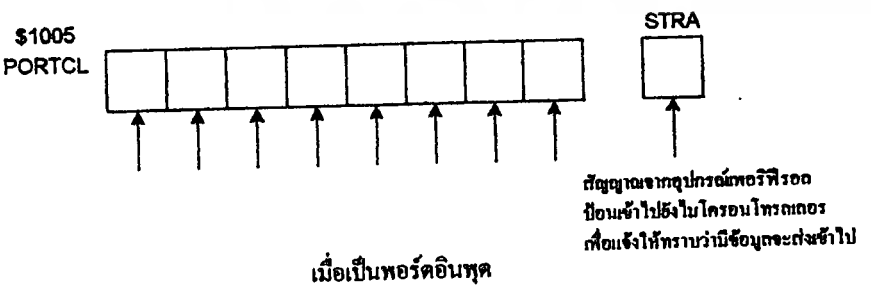
รีจิสเตอร์ DDRD มีแอดเดรสอยู่ที่ $1009 ใช้ควบคุมทิศทางข้อมูลของพอร์ต D ลักษณะการทำงานเป็นเช่นเดียวกับ DDRC แต่จะใช้งานเพียง 6 บิตอีก 2 บิตคือ บิต 6 และ 7 จะต้องกำหนดให้เป็น "0" ทั้งคู่

การติดต่อระหว่างพอร์ตขนานกับอุปกรณ์ภายนอก

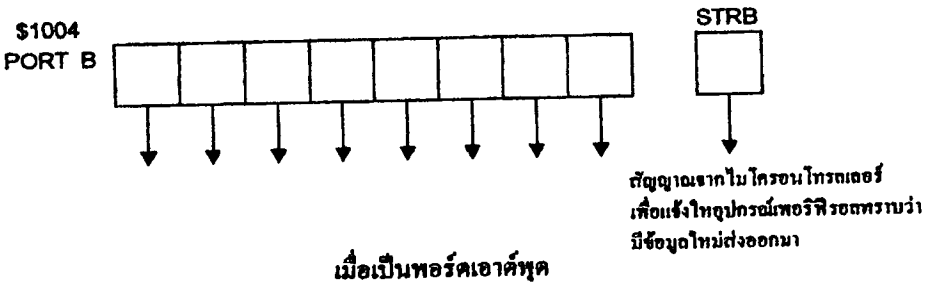
การติดต่อส่งผ่านข้อมูลระหว่างพอร์ตขนานของ 68HC11 กับอุปกรณ์ภายนอกมีลักษณะการติดต่อ 2 แบบคือ แบบสโตรป (Strobe) และแบบฟูลแฮนด์เชก (Full handshake) ซึ่งมีรายละเอียดดังนี้

การติดต่อแบบสโตรบ

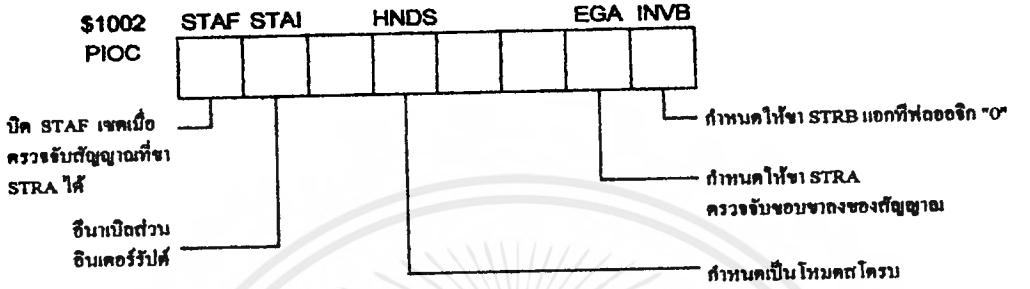
การเลือกลักษณะการติดต่อระหว่างพอร์ตขนานกับอุปกรณ์ภายนอกนี้ สามารถทำได้โดยการกำหนดที่รีจิสเตอร์ควบคุมพอร์ตขนาน หรือ PIOC ถ้าต้องการเลือกแบบสโตรบ บิต HNDS ใน PIOC จะต้องเป็น "0" พอร์ต C จะกลายเป็นพอร์ตอินพุตสโตรบทำงานร่วมกับขา STRA ซึ่งสามารถเลือกขอบขาของสัญญาณที่ขานี้ที่ต้องการตรวจจับได้ที่บิต EGA ใน PIOC เรียกขา STRA ว่าเป็นสโตรบอินพุต ในขณะที่พอร์ต B จะเป็นพอร์ตเอาต์พุตสโตรบ ทำงานร่วมกับขา STRB ซึ่งถูกกำหนดให้เป็นสโตรบเอาต์พุต โดยสามารถเลือกระดับลอจิกของ STRB ที่ต้องการตรวจจับได้ที่บิต INVB ในรีจิสเตอร์ PIOC รูปที่ 3.5 ซึ่งแสดงการติดต่อแบบสโตรบ และการกำหนดบิตใน PIOC เพื่อให้การติดต่อระหว่างพอร์ตขนานกับอุปกรณ์ภายนอกเป็นแบบสโตรบ



เมื่อเป็นพอร์ตอินพุต



เมื่อเป็นพอร์ตเอาต์พุต



การกำหนดบิตในรีจิสเตอร์ PIOC

รูปที่ 3.5 การติดต่อระหว่างพอร์ตขนานกับอุปกรณ์ภายนอกโดยใช้สัญญาณสโตรบ

สโตรบอินพุต พอร์ต C

ในลักษณะนี้จะสามารถอ่านข้อมูลของพอร์ต C ได้ 2 แห่งคือ ที่รีจิสเตอร์ PORTC และที่รีจิสเตอร์ PORTCL ทิศทางของข้อมูลยังคงถูกควบคุมด้วยรีจิสเตอร์ DDRC นอกจากนี้เมื่อเลือกการติดต่อแบบสโตรบแล้ว ขาของพอร์ตทั้งหมดยังคงสามารถใช้งานเป็นขาอินพุตเอาต์พุตสำหรับส่งผ่านข้อมูลได้เช่นเดิม

ขา STRA ถูกใช้เป็นสโตรบอินพุต ซึ่งต้องมีการตรวจจับขอบขาของสัญญาณที่ขานี้ เพื่อให้ 68HC11 สามารถรับข้อมูลได้ สามารถเลือกขอบขาของสัญญาณได้โดยการกำหนดที่บิต EGA ในรีจิสเตอร์ PIOC ว่าจะตรวจจับที่ขอบขาลงหรือขาขึ้นของสัญญาณ

เมื่อใดก็ตามที่สามารถตรวจจับขอบขาของสัญญาณที่ขา STRA นี้ได้ ข้อมูลที่พอร์ต C จะถูกแลตช์เข้าไปยังรีจิสเตอร์ PORTCL และบิต STAF ใน PIOC จะเซต ถ้าหากบิต STAI ใน PIOC เซตอยู่ด้วย ก็จะเกิดการร้องขอการอินเตอร์รัปต์ขึ้นภายในชิป

บิต STAF จะเคลียร์โดยอัตโนมัติ เมื่อมีการอ่านข้อมูลจากรีจิสเตอร์ PIOC และ PORTCL กระบวนการแลตช์ข้อมูลจะเกิดขึ้นอีกครั้งเมื่อสัญญาณสโตรบที่ STRA ถูกตรวจจับได้ ถ้าหากเกิดสัญญาณสโตรบขึ้นก่อนที่ซีพียูจะอ่านข้อมูลจาก PORTCL ข้อมูลใน PORTCL จะหายไป ดังนั้นจึงมีบิต STAF เพื่อแยกสภาวะการแลตช์และอ่านข้อมูล นั่นคือ ถ้าบิต STAF เป็น "1" แสดงว่ากำลังเกิดการแลตช์ข้อมูล ถ้าเป็น "0" คือสภาวะกำลังอ่านข้อมูล

สโตรบเอาต์พุตพอร์ต B

ย้อนกลับไปพิจารณารูปการติดต่อระหว่างพอร์ตขนานกับอุปกรณ์ภายนอกโดยใช้สัญญาณสโตรบ สายอินพุตของอุปกรณ์เพอริฟีรอลจะเชื่อมต่อกับพอร์ต B และสายสโตรบของมันจะเชื่อมต่อกับ STRB จะแอกทีฟเป็นระดับลอจิก "0" หรือ "1" ขึ้นอยู่กับการกำหนดที่บิต INVB ในรีจิสเตอร์ PIOC เมื่อไมโครคอนโทรลเลอร์ต้องการเขียนข้อมูลไปยังพอร์ต B ขา STRB จะแอกทีฟเป็นพัลส์ที่มีความกว้างเท่ากับ 2 ไซเกิลของสัญญาณนาฬิกา (E)

ในกรณีที่มีสัญญาณเอาต์พุตจากอุปกรณ์เพอริฟีรอลส่งเข้ามาที่ขา STRA นั้นจะหมายความว่า ขณะนี้อุปกรณ์นั้น ๆ ต้องการข้อมูลจากไมโครคอนโทรลเลอร์ จากนั้นไมโครคอนโทรลเลอร์ก็จะจัดการส่งข้อมูลไปยังพอร์ต B สัญญาณ STRB จะเป็นตัวบอกอุปกรณ์นั้นว่า กำลังจะส่งข้อมูลไปให้แล้ว ขณะเดียวกันก็ต้องอ่านข้อมูลจาก PORTCL เคลียร์บิต STAF ให้ทันก่อนที่สัญญาณ STRB จะหมดลง โปรแกรมด้านล่างเป็นโปรแกรมตัวอย่างของการส่งข้อมูลออกไปยังพอร์ต B

การติดต่อแบบฟูลแฮนด์เชก

ในการติดต่อระหว่างอุปกรณ์ภายนอกกับไมโครคอนโทรลเลอร์ นอกจากจะมีสัญญาณสโตรบดังที่ได้กล่าวไปแล้ว ในบางครั้งก็มีความจำเป็นต้องการสัญญาณควบคุมมากกว่านั้น สัญญาณอินพุตสโตรบเป็นสัญญาณที่อุปกรณ์เพอริฟีรอลแจ้งแก่ไมโครคอนโทรลอลอร์ว่าพร้อมจะส่งข้อมูลแล้ว ส่วนสัญญาณเอาต์พุตสโตรบก็เช่นกัน เป็นเพียบสัญญาณจากไมโครคอนโทรลเลอร์แจ้งแก่อุปกรณ์เพอริฟีรอลว่า พร้อมที่จะส่งข้อมูลออกไป แต่ก็ไม่มีสัญญาณใดที่จะแจ้งกลับแก่ไมโครคอนโทรลเลอร์ว่า อุปกรณ์นั้นพร้อมรับข้อมูลแล้ว

เมื่อมีการถ่ายเทส่งผ่านข้อมูล จึงต้องมีการกำหนดกฎเกณฑ์มาตรฐานบางอย่าง เพื่อให้การส่งผ่านข้อมูลถูกต้องไม่เกิดความผิดพลาด กฎเกณฑ์ดังกล่าวเรียกว่า โปรโตคอล (Protocol) โปรโตคอลเป็นกระบวนการมาตรฐานที่ใช้ในการสื่อสารข้อมูลระหว่างการับและการส่ง

ในตัว 68HC11 ก็มีโปรโตคอลเช่นกัน เรียกว่า โปรโตคอลของการแฮนด์เชก (Handshake Protocol) โดยกำหนดไว้ว่า ตัวส่งข้อมูลจะส่งข้อมูลก็ต่อเมื่อได้รับสัญญาณตอบรับจากตัวรับก่อน (Acknowledge) และในการถ่ายเทข้อมูลนั้นสามารถส่งได้ทีละไบต์หรือจะส่งเป็นบล็อกก็ได้ 68HC11 มีระบบแฮนด์เชกอัตโนมัติสำหรับพอร์ต C โดยการกำหนดบิต HNDS ในรีจิสเตอร์ PIOC ให้เป็น "1" และกำหนดสถานะของบิตอื่นๆ ที่หหลือให้เกี่ยวข้องกับการแฮนด์เชก

ระบบแฮนด์เชกอัตโนมัตินี้หมายความว่า ในส่วนฮาร์ดแวร์สามารถที่จะตอบสนองสัญญาณได้อย่างอัตโนมัติ โดยที่ไม่ต้องใช้คำสั่งโปรแกรมหรือซอฟต์แวร์ใดๆ ช่วย ยกตัวอย่าง

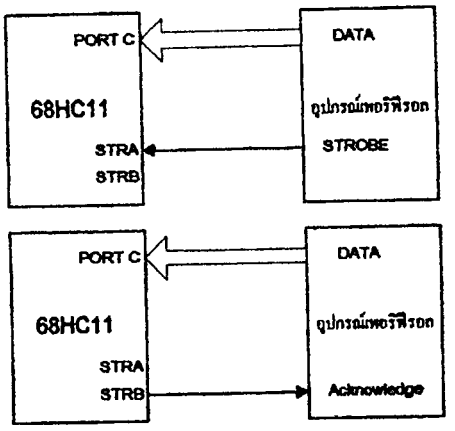
เช่น 68HC11 สามารถแลตช์ข้อมูลที่เข้ามาในพอร์ต C ไปยังรีจิสเตอร์ PORTCL ได้ทันที เมื่อตรวจสอบขอบขาสัญญาณแฮกทีฟที่ขา STRA ได้ โดยไม่มีการใช้คำสั่งตรวจสอบสภาวะที่ขา STRA และไม่มีการใช้คำสั่งอ่านค่าจาก PORTCL อย่างไรก็ตาม ระบบแฮนด์เชกอัตโนมัตินี้จะมีเฉพาะพอร์ต C เท่านั้น ส่วนพอร์ตอื่นๆ ต้องมีการใช้ซอฟต์แวร์เข้าช่วยด้วย

โปรโตคอลการแฮนด์เชกอินพุต

ในรูปด้านล่างเป็นไดอะแกรมการติดต่อระหว่าง 68HC11 กับอุปกรณ์เพอริฟีรอล โดยใช้โปรโตคอลการแฮนนด์เชกอินพุต จากรูปขา STRA จะถูกกำหนดให้เป็นสายอินพุตสโตรบ ส่วนขา SSTRB จะเป็นเอาต์พุตสโตรบ และพอร์ต C เป็นพอร์ตอินพุตสโตรบ ในขณะที่บิต OIN ในรีจิสเตอร์ PIOC ต้องถูกเคลียร์ให้เป็น "0" เพื่อกำหนดเป็นการแฮนด์เชกอินพุต ในการแฮนด์เชกไม่ว่าจะเป็นแฮนด์เชกอินพุตหรือแฮนด์เชกเอาต์พุต จะมีลักษณะการทำงาน 2 แบบ คือ แบบพัลส์ (Pulse) และแบบอินเตอร์ล็อค (Interlock) โดยสามารถเลือกได้โดยกำหนดที่บิต PLS ในรีจิสเตอร์ PIOC

การแฮนด์เชกอินพุตแบบพัลส์ อุปกรณ์เพอริฟีรอลจะกำเนิดพัลส์ส่งมาที่ STRA เพื่อแสดงว่า ต้องการส่งข้อมูล 68HC11 จะตอบสนองพัลส์นั้น แล้วจะมีโปรแกรมเพื่อตอบสนองข้อมูลที่ได้รับเข้ามาเมื่อ 68HC11 อ่านข้อมูลเข้ามา มันจะส่งพัลส์ออกไปที่ STRB ซึ่งมีความกว้างของพัลส์นี้เท่ากับ 2 ไซเกิลของสัญญาณนาฬิกา E ลักษณะที่กล่าวมาข้างต้นนี้เองที่เรียกว่าการแฮนด์เชกอินพุตแบบพัลส์

ในการกำหนดการแฮนด์เชกอินพุตเป็นแบบพัลส์ต้องเซตบิต PLS ใน PIOC ให้เป็น "1" ก่อน จากนั้น 68HC11 จะเริ่มตรวจจับขอบขาสัญญาณที่ต้องการที่ขา STRA และเมื่อตรวจจับได้แล้ว จะทำการอ่านข้อมูลจากรีจิสเตอร์ PORTCL ทันที ในรูปที่3.6 เป็นไทมิ่งไดอะแกรมของการแฮนด์เชกแบบนี้ โดยกำหนดให้ตรวจจับขอบขาลงของสัญญาณ STRA ในขณะนี้สัญญาณ STRB จะแฮกทีฟที่ลอจิก "0" เมื่อได้รับสัญญาณจากอุปกรณ์เพอริฟีรอลแล้ว 68HC11 จะส่งสัญญาณตอบกลับไป ซึ่งก็คือสัญญาณ STRB พร้อมกันนั้น 68HC11 จะเคลียร์บิต STRF พร้อมกับอ่านข้อมูลจากรีจิสเตอร์ PORTCL



รูปที่ 3.6 ไดอะแกรมการติดต่อระหว่างพอร์ตขนานของ 68HC11 กับอุปกรณ์เพอริฟิรอล

การแฮนด์เชกอินพุตแบบอินเตอร์ล็อค ในการเลือกการแฮนด์เชกแบบนี้ต้องทำการเคลียร์บิต PLS ในรีจิสเตอร์ PIOC ในกรณีนี้สัญญาณ STRB จะคล้ายกับสัญญาณแจ้งความพร้อม หรือ READY มันจะแอกทีฟเมื่อไมโครคอนโทรลเลอร์พร้อมจะอ่านข้อมูลเท่านั้น ในรูปด้านบน จะแสดงไทมิ่งไดอะแกรมของสัญญาณ STRB เมื่อมีการแฮนด์เชกแบบอินเตอร์ล็อค

เมื่ออุปกรณ์เพอริฟิรอลตรวจจับได้ว่า ยังไม่มีสัญญาณ READY มา นั่นคือ 68HC11 ยังไม่พร้อมรับข้อมูล มันก็จะยังไม่ส่งข้อมูลใหม่ออกมา จนกระทั่งเมื่อ 68HC11 อ่านข้อมูลจากรีจิสเตอร์ PORTCL เรียบร้อยแล้วก็จะทำให้ STRB แอกทีฟ จากในรูปได้กำหนดการแอกทีฟของ STRB อยู่ที่ลอจิก "0" ซึ่งทำได้โดยการกำหนดบิต INVB ในรีจิสเตอร์ PIOC ให้เป็น "0" การที่ STRB แอกทีฟแสดงว่า ขณะนี้ 68HC11 พร้อมรับข้อมูลใหม่แล้ว แต่อย่างไรก็ตาม อุปกรณ์เพอริฟิรอลจะยังคงไม่ส่งข้อมูลจนกว่าจะแน่ใจว่า STRB แอกทีฟแล้วจริงๆ

โปรโตคอลของการแฮนด์เชกเอาต์พุต

รูปด้านล่างเป็นไดอะแกรมของการแฮนด์เชกเอาต์พุต เมื่อกำหนดให้ 68HC11 ทำการแฮนด์เชกเอาต์พุต ขา STRA จะเป็นอินพุต Ready หรือสัญญาณ Busy ในขณะที่สัญญาณที่ STRB จะเป็นสัญญาณสโตรบเอาต์พุต และต้องทำการเซตบิต OIN ในรีจิสเตอร์ PIOC เพื่อกำหนดให้พอร์ต C มีการแฮนด์เชกเอาต์พุต เช่นเดียวกับการแฮนด์เชกอินพุต จะแบ่งลักษณะการทำงานได้ 2 แบบคือ แบบพัลส์และอินเตอร์ล็อค

การแฮนด์เชกเอาต์พุตแบบพัลส์ เมื่อ 68HC11 ส่งข้อมูลไปยังรีจิสเตอร์ PORTCL เพื่อเตรียมส่งออกก็จะส่งสัญญาณสโตรบ ซึ่งในที่นี้คือสัญญาณที่ขา STRB ออกไปด้วยอย่างอัตโนมัติ โดยสัญญาณดังกล่าวจะมีความกว้าง 2 ไซเกิลของสัญญาณนาฬิกา E เพื่อบอกให้อุปกรณ์เพอริฟิรอลเตรียมพร้อมที่จะรับข้อมูล เมื่ออุปกรณ์เพอริฟิรอลได้รับข้อมูลแล้วก็จะส่ง

สัญญาณ Ready กลับมาให้ 68HC11 ตรวจจับได้ จะทำให้บิต STAF เซต 68HC11 จึงจะเตรียมการสำหรับการส่งข้อมูลไบต์ถัดไป ไปยังรีจิสเตอร์ PORTCL

ในการกำหนดการทำงานแบบพัลส์นี้ จะต้องเซตให้บิต PLS ในรีจิสเตอร์ PIOC เป็น "1" โดยในรูปด้านล่างเป็นไทมิ่งไดอะแกรมของการแฮนด์เชกเอาต์พุตแบบพัลส์ โดยกำหนดให้ตรวจจับขอบขาลงของสัญญาณที่ขา STRA และสัญญาณที่ขา STRB จะแอกทีฟที่ลอจิก "0" เมื่อ 68HC11 ได้รับสัญญาณตอบกลับจากอุปกรณ์เพอริฟีรอล จะทำการอ่านค่าของรีจิสเตอร์ PIOC ซึ่งทำให้บิต STRF ถูกเคลียร์อัตโนมัติ จากนั้นจะเตรียมการเขียนข้อมูลไปยังรีจิสเตอร์ PORTCL

การแฮนด์เชกเอาต์พุตแบบอินเตอร์ล็อค ในการแฮนด์เชกแบบนี้ จะมีการเขียนข้อมูลไปยังรีจิสเตอร์ PORTCL ก่อนสัญญาณ STRB จะแอกทีฟ เมื่อ 68HC11 ได้รับสัญญาณตอบกลับจากอุปกรณ์เพอริฟีรอล มันจะเขียนข้อมูลใหม่ไปยังรีจิสเตอร์ PORTCL จากนั้นจะทำการแอกทีฟ STRB ข้อมูลก็จะถูกส่งออกไป ในการกำหนดการแฮนด์เชกแบบนี้ต้องเคลียร์บิต PLS และเซตบิต OIN ในรีจิสเตอร์ PIOC ก่อน

การเปลี่ยนแปลงของลอจิกสามสถานะของการแฮนด์เชกเอาต์พุต

เหตุการณ์นี้จะเกิดขึ้นเมื่อพอร์ต C มีการกำหนดลักษณะของพอร์ตไม่เหมือนกันในแต่ละขากล่าวคือ บางขาเป็นอินพุต บางขาเป็นเอาต์พุต เมื่อสัญญาณที่ขา STRA ยังไม่แอกทีฟ ขาของพอร์ต C ทั้งหมดจะมีทิศทางการส่งผ่านข้อมูลตามที่กำหนดในรีจิสเตอร์ DDRC ถ้าหากสัญญาณที่ขา STRA แอกทีฟ ขาของพอร์ต C ทั้งหมดจะแสดงตัวเป็นขาเอาต์พุตทันที โดยระดับการแอกทีฟของ STRA จะถูกกำหนดโดยอัตโนมัติ หลังจากที่กำหนดสถานะที่บิต EGA ในรีจิสเตอร์ PIOC ถ้าบิต EGA เป็น "0" ขอบขาของสัญญาณที่จะเลือกตรวจจับคือ ขอบขาลง ส่วนระดับลอจิกที่จะแอกทีฟก็จะเป็นระดับลอจิก "1" ในทางตรงข้ามถ้าบิต EGA เป็น "1" ขอบขาของสัญญาณที่ถูกเลือกจะเป็นขอบขาขึ้นและระดับลอจิกที่จะแอกทีฟก็จะเป็นระดับลอจิก "0" นั่นคือ จะต้องเกิดระดับลอจิกแอกทีฟก่อนการแอกทีฟขอบขาของสัญญาณ หรืออาจกล่าวโดยสรุปได้ว่า การแอกทีฟของสัญญาณที่ขา STRA จะเกิดขึ้นเมื่อระดับลอจิกถูกต้องและขอบขาของสัญญาณถูกต้อง ซึ่งจะต้องตรวจจับที่ระดับลอจิกก่อนขอบขาของสัญญาณ

รีจิสเตอร์ควบคุมพอร์ตขนาน (Parrallel I/O control register)

เป็นรีจิสเตอร์ควบคุมการทำงานของพอร์ตขนานอินพุตเอาต์พุต มีขนาด 8 บิต โดย 7 บิตแรกคือ บิต 0-6 สามารถอ่านและเขียนได้ แต่ในบิตที่ 7 จะสามารถอ่านได้เพียงอย่างเดียว

รีจิสเตอร์นี้จะถูกใช้งานเมื่อ 68HC11 ทำงานในโหมดซิงเกิลชิปเท่านั้น มีแอดเดรสอยู่ที่ $1002 ความหมายและหน้าที่ของบิตต่างๆ ในรีจิสเตอร์ PIOC มีดังนี้

7	6	5	4	3	2	1	0 PIOC
STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB

STAF (Strobe A Interrupt Status Flag) : บิตนี้จะเซตเมื่อสัญญาณสโตรบ A เกิดขึ้น ส่วนการเคลียร์บิตนี้จะขึ้นอยู่กับสถานะของบิต HNDS และ ION เมื่อพอร์ตขนานนี้ทำงานในโหมดสโตรบและโหมดอินพุตแฮนด์เชก บิตนี้จะเคลียร์เมื่อมีการอ่านรีจิสเตอร์ PIOC และจะเซตใหม่หลังจากที่มีการอ่านข้อมูลรีจิสเตอร์ PORTCL แล้ว ถ้าหากทำงานในโหมดเอาต์พุตแฮนด์เชก บิต STAF จะเคลียร์เมื่อมีการอ่านรีจิสเตอร์ PICO และจะเซตเมื่อเขียนข้อมูลลงในรีจิสเตอร์ PORTCL เรียบร้อยแล้ว

STAI (Strobe A Interrupt Enable Mask) : เมื่อบิต I ใน CCR เคลียร์ บิต STAI นี้จะเซต และถ้าหากบิต STAF เซตด้วย จะเป็นการร้องขอการอินเตอร์รัปต์

CWOM (Port C Wire OR Mode) : บิต C พอร์ตนี้จะมีผลต่อขาพอร์ต C ทั้ง 8 ขา โดยถ้าเป็น "0" พอร์ต C เอาต์พุตจะมีลักษณะเป็นเอาต์พุตแบบซีมอสธรรมดา ถ้าเป็น "1" พอร์ต C จะมีลักษณะเป็นเอาต์พุตแบบเดรนเปิด (open drain)

HNDS (Handshake Modes) : ถ้าบิตนี้เคลียร์ สัญญาณสโตรบ A จะแสดงตนเป็นสัญญาณสโตรบอินพุต เพื่อแลตช์ข้อมูลที่จะเข้าสู่ PORTCL และสัญญาณสโตรบ B จะเป็นสัญญาณสโตรบเอาต์พุต หลังจากเขียนข้อมูลลงในพอร์ต B แต่ถ้าหากบิตนี้เซตจะทำให้พอร์ตขนานนี้เกิดการแฮนด์เชกขึ้น หรือกล่าวง่ายๆ คือ บิตนี้ใช้เลือกโหมดการทำงานส่งผ่านข้อมูลของพอร์ตขนานนั่นเอง ถ้าเป็น "0" จะเป็น โหมดสโตรบ แต่ถ้าเป็น "1" จะเป็นโหมดแฮนด์เชก

OIN (Output or Input Handshaking) : เป็นบิตที่ใช้กำหนดลักษณะการแฮนด์เชก ถ้าบิตนี้เป็น "0" จะเป็นอินพุตแฮนด์เชก ถ้าเป็น "1" จะเป็นเอาต์พุตแฮนด์เชก บิตนี้จะไม่มีผลอะไร หากบิต HNDS เป็น "0"

PLS (Pulse/Interlocked Handshake Operation) : ถ้าบิตนี้เป็น "0" จะเป็นการเลือกการแฮนด์เชกแบบอินเตอร์ล็อคสัญญาณ STRB จะแอกทีฟ 1 ครั้ง และจะแอกทีฟไปจนกระทั่งขอบขาของสัญญาณ STRA ที่เลือกไว้จะถูกตรวจจับได้ แต่ถ้า

เป็น "1" เป็นการเลือกการแฮนด์เชกแบบพัลส์โดย STRB จะเป็นพัลส์กว้าง 2 ไซเกิล ของสัญญาณนาฬิกา E

EGA (Active Edge for Strobe A) : ถ้าเป็น "0" ขอบขาลงของสัญญาณ STRA จะถูกเลือก เมื่อเลือกการแฮนด์เชกเอาต์พุตสายสัญญาณของพอร์ต C จะมีทิศทางตามที่กำหนดไว้ในรีจิสเตอร์ DDRC ในขณะที่ STRA เป็น "0" แต่ถ้า STRA เป็น "1" พอร์ต C จะกลายเป็นพอร์ตเอาต์พุตทันที ถ้าหากบิตนี้เป็น "1" จะเป็นการเลือกขอบขาขึ้นของสัญญาณ STRA มาควบคุมให้พอร์ต C ทำงานตามทิศทางที่กำหนดไว้ในรีจิสเตอร์ DDRC แต่ถ้าหากสัญญาณ STRA เป็น "0" พอร์ต C จะแสดงตนเป็นพอร์ตเอาต์พุต ในกรณีที่เป็นการแฮนด์เชกเอาต์พุต

INVB (Invert Strobe B) : ถ้าบิตนี้เป็น "0" จะเป็นการกำหนดระดับแอกทีฟของ STRB เป็นลอจิก "0" ถ้าหากเป็น "1" ระดับแอกทีฟของ STRB จะเป็นลอจิก "1"

3.3 บอร์ดพัฒนาไมโครคอนโทรลเลอร์ CP-68HC11

3.3.1 คุณสมบัติทางเทคนิค

ไมโครคอนโทรลเลอร์ : MC68HC11AFN

หน่วยความจำ : แรม 6264 (8 กิโลไบต์) หรือ 62256 (32 กิโลไบต์) ใช้งานที่แอดเดรส $2000 - $9FFF อีพรอม 2764 (8 กิโลไบต์) หรือ 27256 (32 กิโลไบต์) แต่ใช้งานได้สูงสุด 24 กิโลไบต์ ช่วงแอดเดรสคือ $A000 - $FFFF แรมภายในขนาด 256 ไบต์

พอร์ตอินพุตเอาต์พุต : ขนาด 24 บิต โดยใช้ชิป 8255

พอร์ตพรินเตอร์ : 1 พอร์ต

พอร์ตเชื่อมต่อคีย์บอร์ด : แบบเมตริกซ์ 3 x 4 หรือ 4x4 จำนวน 1 พอร์ต

พอร์ต LCD : 1 พอร์ต ใช้ได้ทั้งแบบตัวอักษรและกราฟิก

พอร์ตตั้งเวลา : อินพุต 3 ช่อง เอาต์พุต 4 และอินพุตเอาต์พุต 1 ช่อง

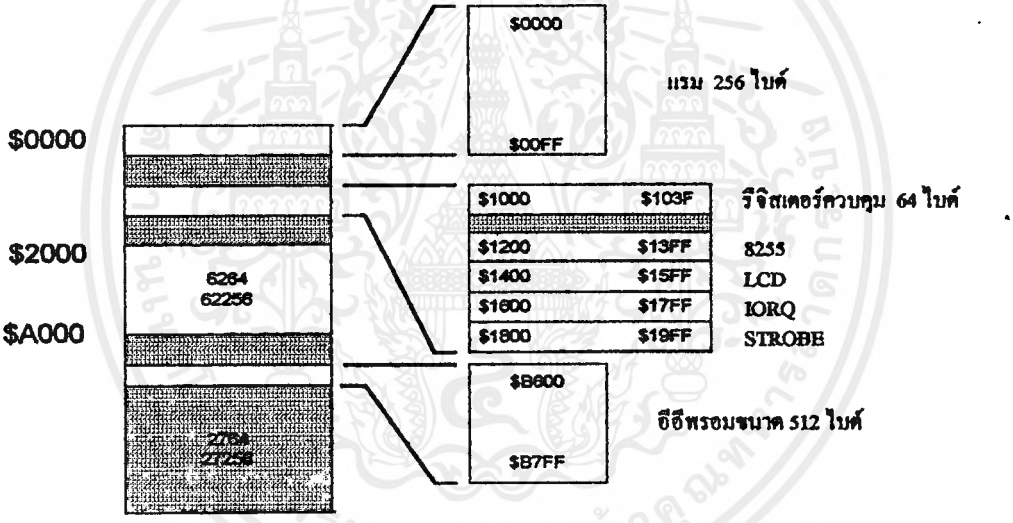
พอร์ตอนุกรม : 2 ช่อง แบบ SCI และ SPI

สัญญาณนาฬิกา : 8 เมกะเฮิรตซ์

ขนาด : 9.3 เซ็นติเมตร x 14 เซ็นติเมตร

3.3.2 การจัดหน่วยความจำของบอร์ด

บอร์ด CP – 68HC11 มีการจัดหน่วยความจำแสดงในรูปที่3.7 ส่วนแรกขนาด 256 ไบต์แอดเดรส $0000 - $00FF จัดสรรไว้สำหรับหน่วยความจำในชิป ที่แอดเดรส $1000 - $103F เป็นพื้นที่ของรีจิสเตอร์ควบคุมขนาด 64 ไบต์ ส่วนแอดเดรสตั้งแต่ $1200 - $13FF เป็นแอดเดรสของ พอร์ต 8255 สำหรับแอดเดรสของ LCD จัดสรรไว้ที่ $1400 - $15FF แอดเดรส $1600 - $17FF เป็นของสัญญาณ IORQ แอดเดรส $1800 - $19FF เป็นของสัญญาณสโตรป ส่วนแอดเดรสระหว่าง $B600 - $B7FF เป็นของหน่วยความจำอีอีพรอมภายในชิป 68CH11 สุดท้ายตั้งแต่แอดเดรส $A000 - $FFFF เป็นพื้นที่สำหรับหน่วยความจำอีพรอมภายนอก



รูปที่ 3.7 แสดงการจัดหน่วยความจำบนบอร์ด CP – 68HC11

การเลือกหน่วยความจำแรม

สามารถเลือกใช้แรมได้ 2 เบอร์ คือ 6264 ขนาด 8 กิโลไบต์ และ 62256 ขนาด 32 กิโลไบต์ โดยเลือกจัมเปอร์ J1 สำหรับกำหนดตำแหน่งของหน่วยความจำสามารถเลือกได้ที่จัมเปอร์ J3 ซึ่งการใส่จัมเปอร์ 1 ตัวสามารถอ้างตำแหน่งในหน่วยความจำได้ 8 กิโลไบต์ ถ้าใส่ครบทั้ง 4 ตัว ก็สามารถอ้างตำแหน่งได้ 32 กิโลไบต์ โดยที่ตำแหน่งของหน่วยความจำจะอยู่ในช่วง $2000 - $9FFF

การเลือกหน่วยความจำอีพรอม

สามารถเลือกใช้อีพรอมได้ 2 เบอร์ คือ 2764 กิโลไบต์ และ 27256 ขนาด 32 กิโลไบต์ โดยเลือกที่จัมเปอร์ J2 สำหรับกำหนดตำแหน่งของหน่วยความจำสามารถเลือกได้ที่จัมเปอร์ J4 ซึ่งการใส่จัมเปอร์ 1 ตัว สามารถอ้างตำแหน่งในหน่วยความจำได้ 8 กิโลไบต์ ถ้าใส่ครบทั้ง 4 ตัว ก็สามารถอ้างตำแหน่งได้ 24 กิโลไบต์ โดยที่ตำแหน่งของหน่วยความจำจะอยู่ในช่วง $A000 - $FFFF

แอดเดรสของพอร์ตอินพุตเอาต์พุต 8255

เป็นคอนเน็กเตอร์ขนาด 34 ขา ซึ่งต่อออกมาจากพอร์ตของ 8255 สำหรับแอดเดรสที่ใช้ในการติดต่อกับพอร์ตของ 8255 มีรายละเอียดดังนี้

พอร์ต A อยู่ที่แอดเดรส $1200

พอร์ต B อยู่ที่แอดเดรส $1201

พอร์ต C อยู่ที่แอดเดรส $1202

พอร์ตควบคุม อยู่ที่แอดเดรส $1203

การเชื่อมต่อกับคีย์บอร์ด

เป็นคอนเน็กเตอร์ 10 ขา ซึ่งต่อมาจากพอร์ต C ของ 8255 สามารถต่อเข้ากับคีย์บอร์ดแบบ 4x4 หรือ 3x4 ได้โดยการเลือกที่จัมแปอร์ J5 ในกรณีที่เลือกคีย์บอร์ดเป็นแบบ 3x4 ขา PC0 ของ 8255 จะถูกนำมาใช้เป็นสัญญาณ BUSY เพื่ออ่านสถานะของเครื่องพิมพ์ ส่วนในกรณีที่เลือกคีย์บอร์ดแบบ 4x4 จะไม่สามารถใช้เชื่อมต่อกับพรินเตอร์ได้ เนื่องจากขา PC0 ของ 8255 ถูกต่อเข้ากับคอนเน็กเตอร์ 10 ขาของคีย์บอร์ดแทน

พอร์ตเชื่อมต่อวงจร A/D

มีคอนเน็กเตอร์ขนาด 14 ขา ซึ่งต่อออกมาจากขา PE0 – PE7 ของ 68HC11 เป็นอินพุตของวงจรแปลงสัญญาณแอนาล็อกเป็นดิจิตอล (A/D) ขนาด 8 บิต 8 ช่อง โดยใช้แรงดันอ้างอิงขนาด 5 โวลต์ จากภายในบอร์ด หรือใช้จากภายนอกก็ได้ โดยให้มีค่าของแรงดันอยู่ในช่วง 2.5 – 5 โวลต์ สามารถเลือกได้โดยถอดจัมเปอร์ J9

พอร์ตของตัวตั้งเวลาและตัวนับ

เป็นคอนเน็กเตอร์ขนาด 10 ขา ซึ่งต่อออกมาจากขา PA0 – PA7 ของ 68HC11 สามารถนำไปประยุกต์ได้หลายอย่าง เช่น นับความถี่ นับจำนวนพัลส์คาบเวลา กำเนิดสัญญาณรูปสี่เหลี่ยมหรือจะใช้เป็นพอร์ตอินพุต เอาต์พุตก็ได้

การพัฒนาบอร์ด CP - 68HC11

การพัฒนาโปรแกรมบนบอร์ด CP – 68HC11 นี้จะใช้ซอฟต์แวร์ ET – DEBUGGER 68HC11 โดยติดตั้งอีพรอมที่บันทึกโปรแกรมลงบนบอร์ดก่อนแล้วต่อบอร์ดเข้ากับคอมพิวเตอร์ผ่านทางพอร์ต RS – 232 จากนั้นจ่ายแรงดันให้กับบอร์ด แล้วนำแผ่นดิสก์โปรแกรม ET-DEBUDDER 68HC11 ใส่เข้าไปในคอมพิวเตอร์แล้วรันโปรแกรม

ผู้ใช้สามารถเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ แล้วรันบนบอร์ดได้เลยขนาดของโปรแกรมที่เขียนขึ้นเพื่อพัฒนานั้นจะมีความยาวได้ไม่เกิน 8 กิโลไบต์ ถ้าใช้แรม บนบอร์ดเป็น 6264 หรือถ้าใช้แรมเบอร์ 62256 จะเพิ่มความยาวในการเขียนโปรแกรมได้ถึง 32 กิโลไบต์

บทที่ 4

ระบบการควบคุมและผลการทดลอง

4.1 หลักการควบคุม

การควบคุมเซอร์โวมอเตอร์ทำได้โดยการป้อนความถี่พัลซ์บวกดังที่ได้กล่าวไว้ในบทที่3 ดังนั้นเราจึงใช้ไมโครคอนโทรนเลอร์สร้างความถี่พัลซ์ ขึ้นมาควบคุมเซอร์โวมอเตอร์ทั้ง 12 ตัว เพื่อให้หมุนไปในตำแหน่งที่ต้องการ และพัลซ์ที่สร้างขึ้นมาจะมีความถี่ไม่เท่ากัน ถ้าหากตำแหน่งการหมุนของของเซอร์โวมอเตอร์ต่างกันตลอดเวลาของการทำงาน (Power ON) จะต้องสร้างพัลซ์ให้กับเซอร์โวมอเตอร์ตลอดเวลา เพื่อให้เซอร์โวแต่ละตัวรักษาตำแหน่งของตนเองไว้ และด้านแรงกดทับจากน้ำหนักของตัวหุ่นยนต์

จากโครงสร้างของตัวหุ่นยนต์ที่สร้างขึ้นเซอร์โวทางด้านซ้ายของตัวหุ่น และเซอร์โวทางด้านขวาของตัวหุ่นจะวางอยู่ในตำแหน่งตรงกันข้ามทั้งสองด้าน นั่นหมายถึงความถี่พัลซ์ที่ป้อนให้กับเซอร์โวทั้งด้านซ้ายและขวาจะต้องต่างกัน

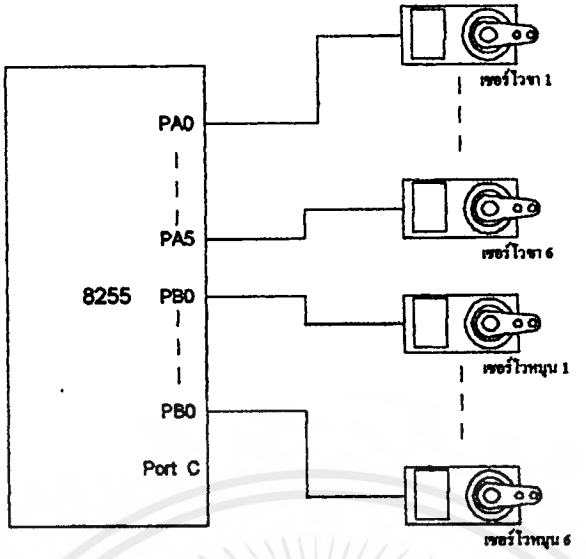
4.1.1 การเชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับเซอร์โวมอเตอร์

ในการควบคุมเลือกใช้ ไมโครคอนโทรลเลอร์ 68HC11 ผ่านทาง ไอซีพอร์ต 8255 โดย ใช้สัญญาณควบคุมเพียง 1 บิตต่อเซอร์โวมอเตอร์ 1 ตัว ดังแสดงในรูปที่ 4.1 ซึ่งเป็นการแสดงไดอะแกรมการควบคุมเซอร์โวมอเตอร์ และมีการเชื่อมต่อพอร์ตเข้ากับเซอร์โวมอเตอร์ดังนี้

พอร์ต A บิตที่ 0 ถึงบิตที่ 5 เชื่อมต่อกับ เซอร์โวมอเตอร์ที่ทำหน้าที่ยกขา ทั้ง 6 ขา

พอร์ต B บิตที่ 0 ถึงบิตที่ 5 ของ เชื่อมต่อกับ เซอร์โวมอเตอร์ที่ทำหน้าที่เปลี่ยนตำแหน่งของขา

พอร์ต C สำรองไว้ใช้งานในส่วนเพิ่มเติม



รูปที่ 4.1 แสดงไดอะแกรมการควบคุมเซอร์โวมอเตอร์

4.1.2 การพัฒนาโปรแกรม ไมโครคอนโทรลเลอร์ 68HC11

โครงงานนี้เลือกใช้ บอร์ดไมโครคอนโทรลเลอร์ CP-68HC11 ผลิตโดยบริษัท อีทีที จำกัด เหตุเพราะ เป็นบอร์ดที่มีประสิทธิภาพในการทำงานสูง สามารถเชื่อมต่อเพื่อควบคุมอุปกรณ์ต่าง ๆ ได้เป็นอย่างดี

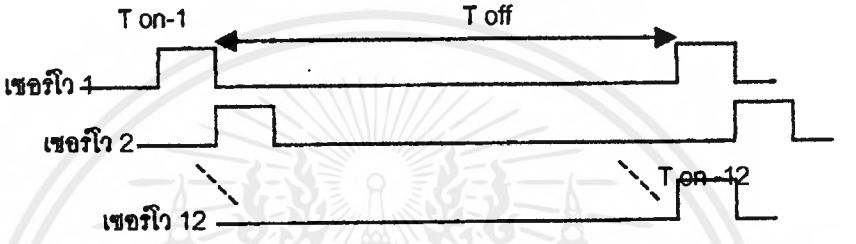
การพัฒนาโปรแกรมบนบอร์ด CP – 68HC11 นี้จะใช้ซอฟต์แวร์ ET – DEBUGGER 68HC11 โดยติดตั้งอีพรอมที่บันทึกโปรแกรมลงบนบอร์ดก่อนแล้วต่อบอร์ดเข้ากับคอมพิวเตอร์ผ่านทางพอร์ต RS – 232 จากนั้นจ่ายแรงดันให้กับบอร์ด จากนั้นรันโปรแกรม PROCOM+ เพื่อทำการติดต่อกับโปรแกรมมอนิเตอร์ของบอร์ดCP– 68HC11 จากจุดนี้สามารถทำการ อัพโหลด HEX ไฟล์ , รันโปรแกรม , ดูค่าในรีจิสเตอร์และค่าในหน่วยความจำได้

ในการสร้าง HEX ไฟล์เพื่อทำการอัพโหลดเข้าไปในบอร์ด อันดับแรกจะต้องทำการ เขียนโปรแกรม เป็นภาษาแอสแซมบลี้ ของ CPU 68HC11 ให้ถูกต้องตามรูปแบบ จากนั้นใช้โปรแกรม AS11NEW ทำการ แปลง Code จาก TEXT ไฟล์ เป็น HEX ไฟล์ มีนามสกุล .S19 และไฟล์ที่มีนามสกุลเป็น .S19 นี้สามารถใช้ อัพโหลดไปยังบอร์ดCP-68HC11 ได้

4.2 แนวทางการแก้ปัญหาในการควบคุม

4.2.1 การจัดสัญญาณพัลซ์ให้กับเซอร์โวมอเตอร์

การที่จะให้ไมโครคอนโทรนเลอร์สร้างความถี่ที่ต่างกันในเวลาที่พร้อมกันนั้นทำได้ยาก เราหลีกเลี่ยงโดยการสร้างสัญญาณพัลซ์บวกตามเวลาที T - on ให้กับเซอร์โวมอเตอร์แต่ละตัว โดยอาศัย ช่วงจังหวะ T – off สร้างสัญญาณ T – on ให้กับเซอร์โวมอเตอร์ตัวต่อไป จนครบทั้ง 12 ตัว ดังรูปที่ 4.2



รูปที่ 4.2 ลักษณะของสัญญาณควบคุมเซอร์โว

เมื่อครบรอบการทำงานจะมีการวนลูป ประมวลผลเพื่อควบคุมขาหุ่นยนต์ในแต่ละตำแหน่ง ซึ่ง ตำแหน่งของขาในแต่ละลำดับ จะวนมาสร้างสัญญาณสัญญาณพัลซ์ให้กับเซอร์โวในแต่ละตัวอีก เพื่อย้ายตำแหน่งของขาหุ่นยนต์ให้เคลื่อนที่ในตำแหน่งที่ต้องการ ซึ่งการ วนลูป อย่างนี้เรื่อยๆไป จะทำให้ดูเหมือนว่า เซอร์โวได้รับพัลซ์พร้อมกัน

4.2.2 การสร้างสัญญาณพัลซ์โดยไมโครคอนโทรลเลอร์

หลักการสร้างสัญญาณพัลซ์โดยไมโครคอนโทรลเลอร์ คือ ใช้สัญญาณข้อมูล 1 บิต สำหรับ 1 ช่องสัญญาณพัลซ์โดยมีขั้นตอนดังนี้

1.ทำการส่งข้อมูล '1' ไปยังบิตที่ต้องการ ให้เกิดเป็นพัลซ์บวก หรือ ช่วง T ON

2.ทำการหน่วงเวลาตามระยะเวลาของ T ON

3.ส่งข้อมูล '0' เมื่อต้องการให้อยู่ในช่วง T OFF

4. ทำการหน่วงเวลาตามระยะเวลาของ T OFF

5.ทำการวน ลูปกลับไปขั้นตอนที่ 1 อีกครั้ง

หลักการดังกล่าว เป็นหลักการที่ง่ายสำหรับการสร้างพัลซ์ ช่องสัญญาณเดียว แต่ถ้าหากต้องการสร้างพัลซ์หลายๆช่องสัญญาณ ที่ต้องการความถี่ต่างกันตลอดเวลา ซึ่งหลักการดังกล่าวจะนำไปสู่ความซับซ้อนในการเขียนโปรแกรมเป็นอย่างมาก

ข้อได้เปรียบอย่างหนึ่งของไมโครคอนโทรนเลอร์ 68HC11 คือ รีจิสเตอร์ตัวนับเวลา (Time counter register : TCNT) ซึ่งเป็นรีจิสเตอร์ขนาด 16 บิต หลักการ รีเซตตัวนับเวลานี้จะเริ่มนับเวลาตามความถี่ของสัญญาณนาฬิกาทำให้กับ CPU ตัวนับนี้จะไม่ขึ้นอยู่กับการประมวลผลตามคำสั่งของ CPU และ CPU สามารถที่จะตั้งการนับได้ตลอดเวลา

จากข้อได้เปรียบอันนี้จะถูกนำมาใช้ให้เป็นประโยชน์ในการหน่วงเวลาของ T ON และ T OFF ของสัญญาณพัลซ์ ดังขั้นตอนการหน่วงเวลาดังต่อไปนี้

1.อ่านค่าของตัวนับจากรีจิสเตอร์ TCNT

2.บวกกับค่าแกนเวลา ที่ต้องการหน่วงเวลา

3.เปรียบเทียบหากว่า

4.หากพัลซ์ของการบากเพิ่มเท่ากับ TCNT ให้เสร็จสิ้นการหน่วงเวลา

ค่าแทนเวลา (N) หาได้จากสมการ

$$N = t \times F$$

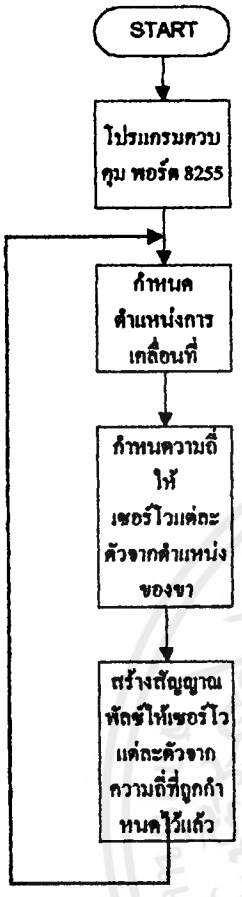
เมื่อ N คือ ค่าแทนเวลา ที่ใช้บวกเพิ่มค่ากับ รีจิสเตอร์ TCNT

t คือ เวลาที่ต้องการหน่วง (วินาที)

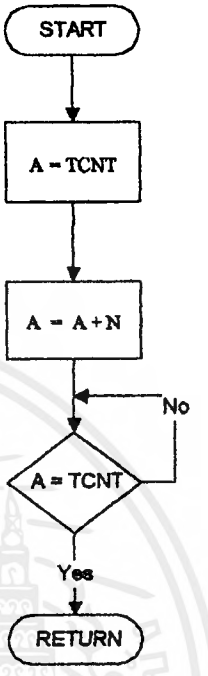
F คือ ความถี่ของสัญญาณนาฬิกาของ CPU (Hz)

ซึ่งการสร้างสัญญาณพัลซ์โดยไมโครคอนโทรลเลอร์มีลำดับการควบคุมการทำงานและลำดับการหน่วงเวลาโดยแสดงเป็น Flowchart ในรูปที่ 4.3 ดังต่อไปนี้

Flowchart



Flowchart ลำดับการควบคุม



Flowchart ลำดับการหน่วงเวลา

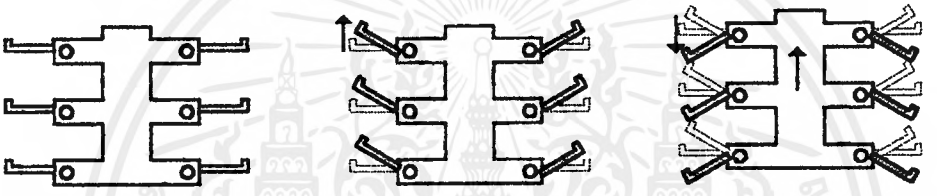
รูปที่ 4.3 Flowchart ของโปรแกรมควบคุม

4.3 การทดลองการเคลื่อนที่ของหุ่นยนต์

ในหัวข้อนี้ได้นำเสนอหลักการทดสอบแนวคิดและหลักการเคลื่อนที่ไปในทิศทางต่าง ๆ ของหุ่นยนต์ โดยทำการเขียนโปรแกรมเป็นภาษาแอสแซมบลี้

4.3.1 การทดลองการเคลื่อนที่ไปด้านหน้า และเคลื่อนที่ไปด้านหลัง

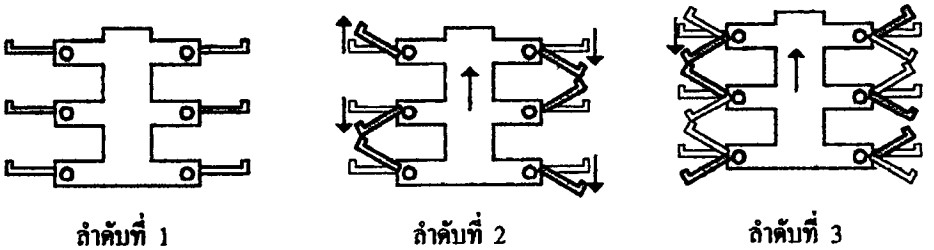
แนวความคิดแรกในการเคลื่อนที่ไปด้านหน้าคือ เปลี่ยนตำแน่งของขาให้อยู่หน้าจุดหมุน ทีละขา ซ้าย ขวา สลับกันไป และหมุนเปลี่ยนตำแหน่งของขาให้มาอยู่ ณ อีกตำแหน่งหนึ่ง เพื่อส่งให้ลำตัวของหุ่นเคลื่อนที่ไปข้างหน้า ดังรูปที่ 4.4 ซึ่ง วิธีนี้จะมีข้อเสีย คือ จะเกิดแรงกดบีบจากขาเข้าหาลำตัวของหุ่นอันเนื่องมาจาก การเคลื่อนที่ของขาไม่เป็นเส้นตรงแต่เป็นลักษณะรัศมีของวงกลม



รูปที่ 4.4 ลำดับหลักการ การเคลื่อนที่ไปข้างหน้า

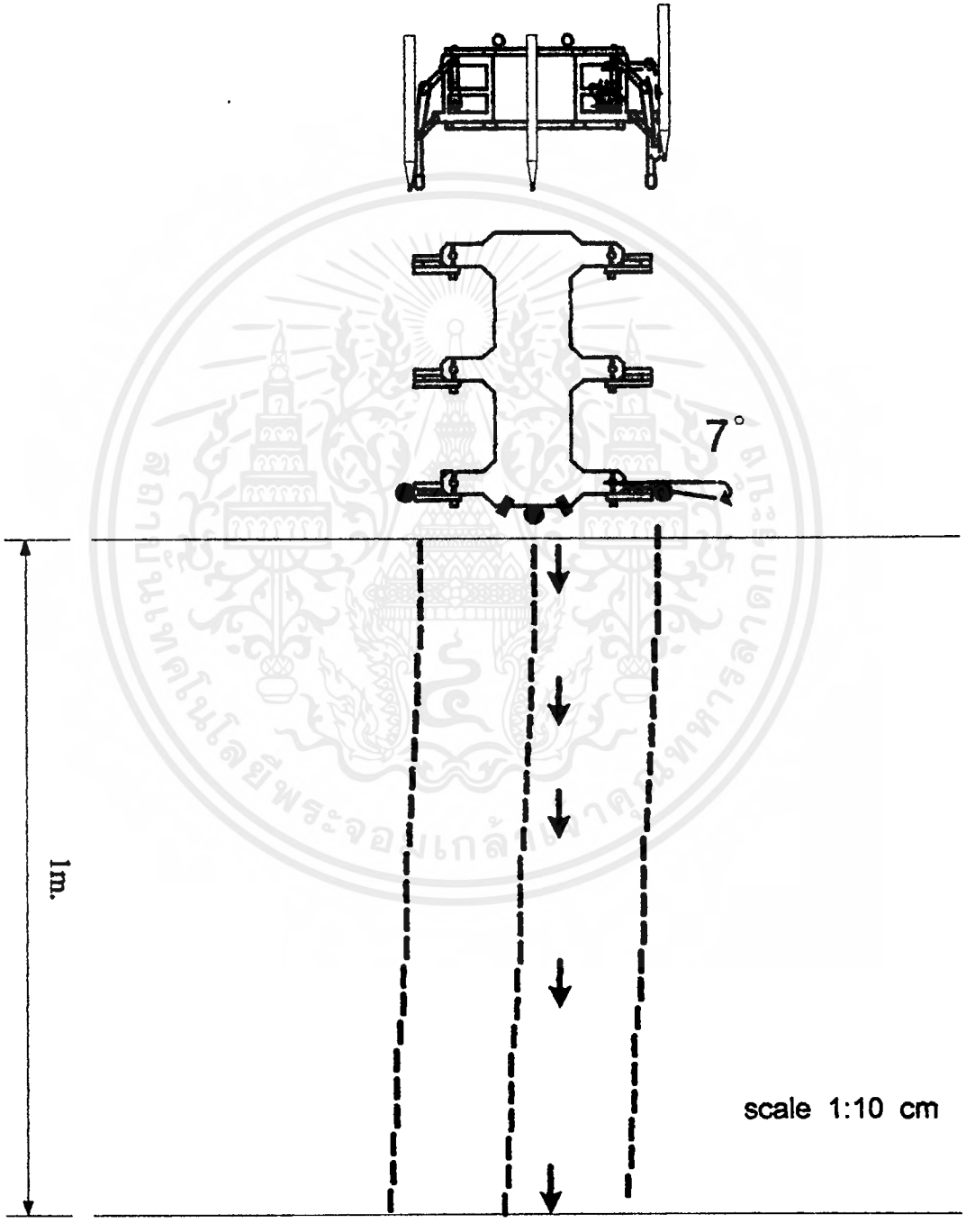
การเคลื่อนที่มาด้านหลัง จะตรงข้ามกับการเคลื่อนที่ไปด้านหน้า และมีข้อเสียเหมือนกัน คือเกิดแรงกดบีบเข้าหาลำตัว ขณะการเปลี่ยนตำแหน่งการเดิน

จากข้อเสียในแบบแรก จึงพยายามลดแรงกดบีบจากขาให้เหลือน้อยลงโดยการ ใช้หลักการเดินทีละ 3 ขา แบบสัมพันธ์ โดยจะมีการเคลื่อนย้ายทีละ 3 ขา พร้อมกันและสัมพันธ์กัน ดังแสดงในรูปที่ 4.5



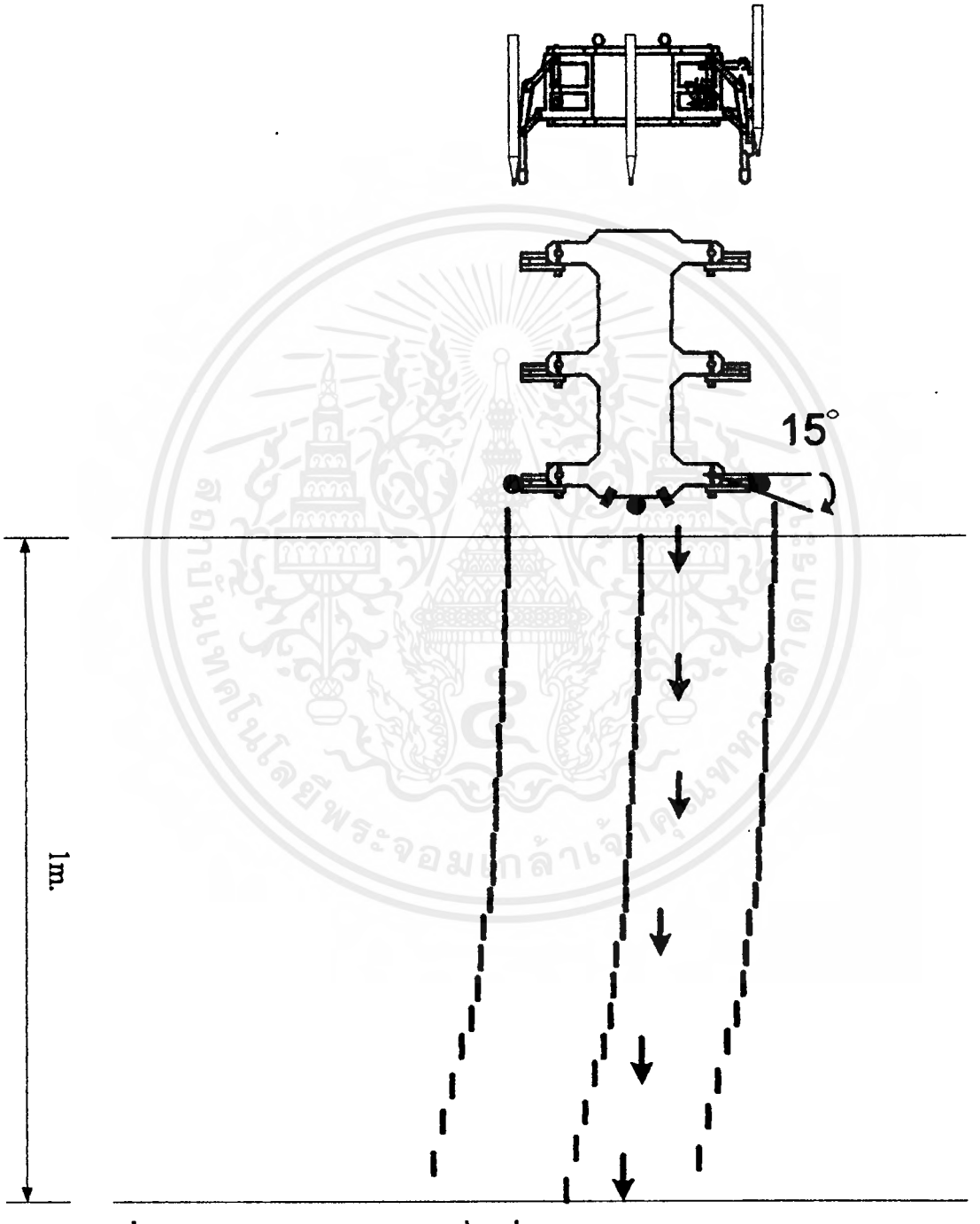
รูปที่ 4.5 ลำดับการเดินไปด้านหน้าทีละ 3 ขา

ดังนั้นจากผลการทดลอง การเคลื่อนที่ไปด้านหน้า โดยใช้ค่าทดสอบมุมก้าวขา 7 องศา ดังแสดงในรูปที่ 4.6 ได้แสดงลักษณะเส้นทางการเคลื่อนที่ของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.03125 เมตร/วินาที โดยใช้ระยะทดสอบ 1 เมตร



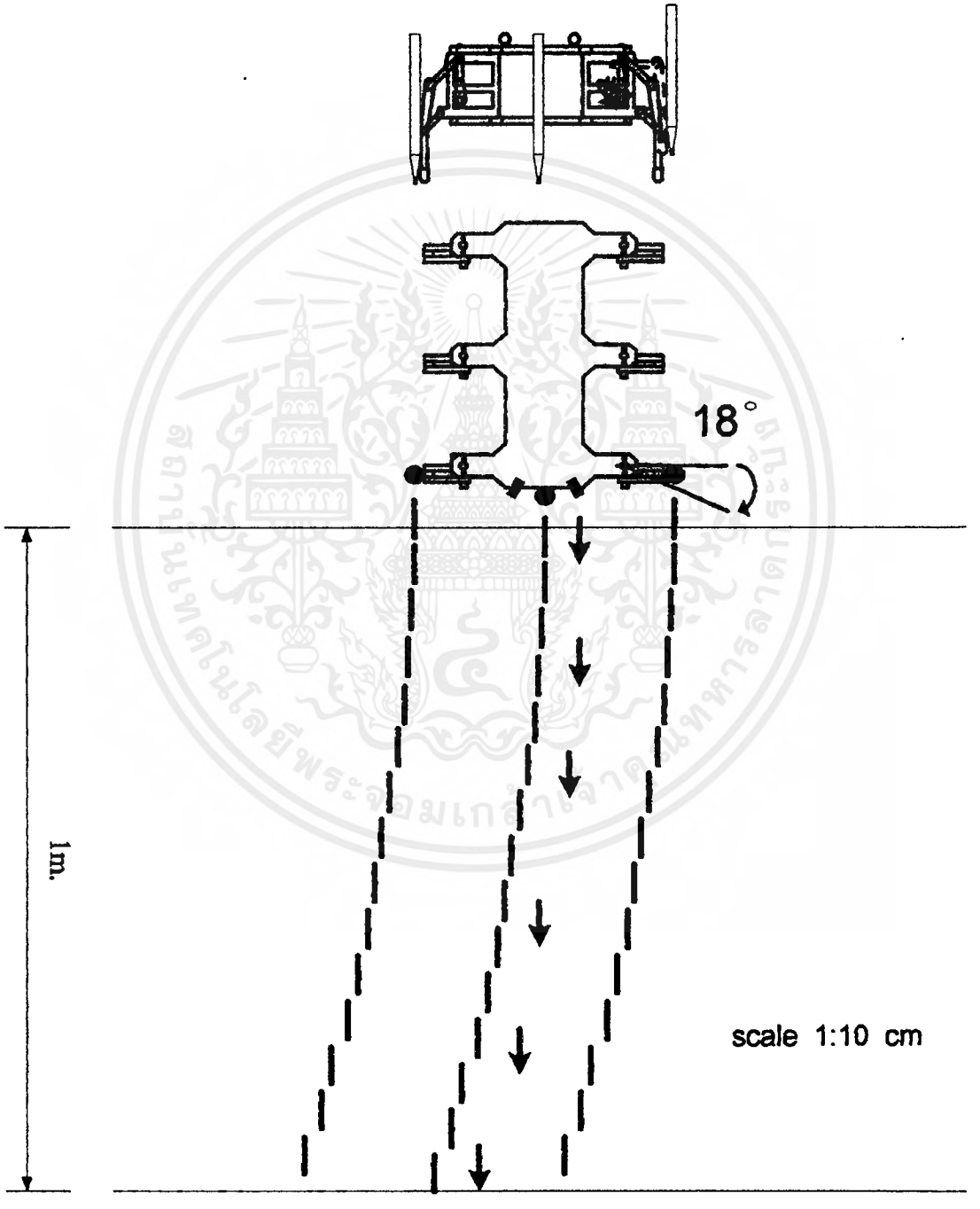
รูปที่ 4.6 แสดงผลการทดสอบการเคลื่อนที่ไปด้านหน้าด้วยมุมก้าวขา 7 องศา

ผลการทดลอง การเคลื่อนที่ไปด้านหน้า โดยใช้ค่าทดสอบมุมก้าวขา 15 องศา ดังแสดงในรูปที่ 4.7 ได้แสดงลักษณะเส้นทางการเคลื่อนที่ของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.04348 เมตร/วินาที โดยใช้ระยะทดสอบ 1 เมตร



รูปที่ 4.7 แสดงผลการทดสอบการเคลื่อนที่ไปด้านหน้าด้วยมุมก้าวขา 15 องศา

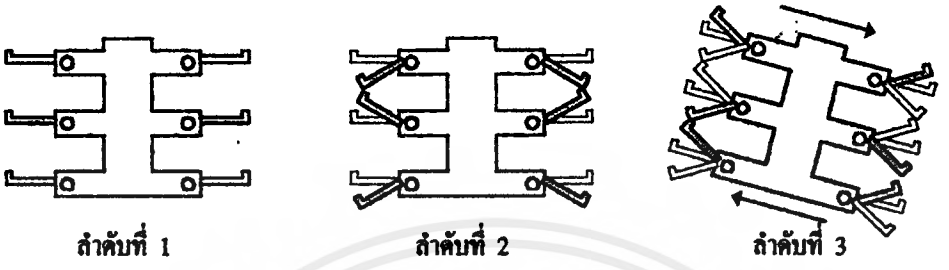
ผลการทดลอง การเคลื่อนที่ไปด้านหน้า โดยใช้ค่าทดสอบมุมแกว่งขา 18 องศา ดังแสดงในรูปที่ 4.8 ได้แสดงลักษณะเส้นทางการเคลื่อนที่ของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.05263 เมตร/วินาที โดยใช้ระยะทดสอบ 1 เมตร



รูปที่ 4.8 แสดงผลการทดสอบการเคลื่อนที่ไปด้านหน้าด้วยมุมแกว่งขา 18 องศา

4.3.2 การทดลองการเคลื่อนที่แบบหมุนตัว

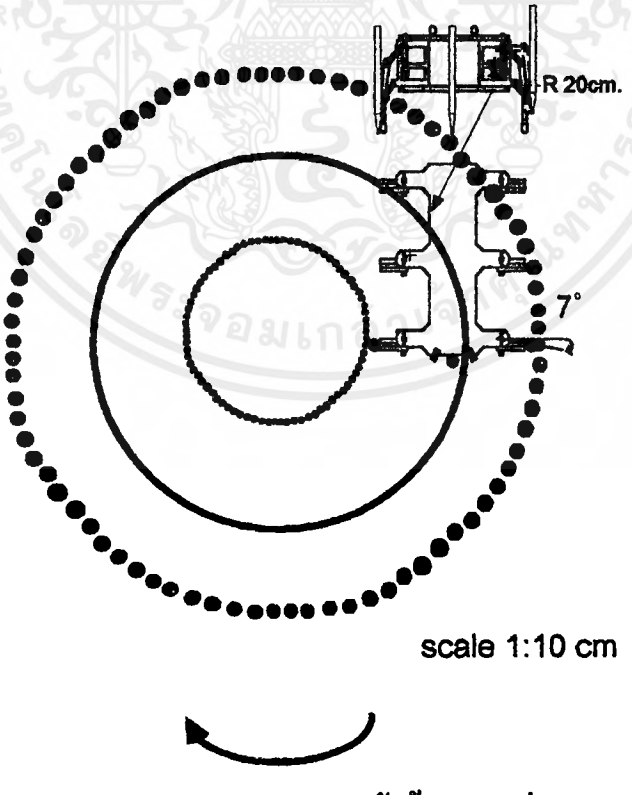
จากแนวความคิดการเคลื่อนที่ไปข้างหน้าแบบที่สอง สามารถนำมาประยุกต์เป็นแนวคิดในการหมุนตัวโดยการสลับทิศทาง ตำแหน่งของขาขณะทำการหมุนบิดกับลำตัวดังแสดงในรูปที่4.9



รูปที่ 4.9 ลำดับการหมุนตัว

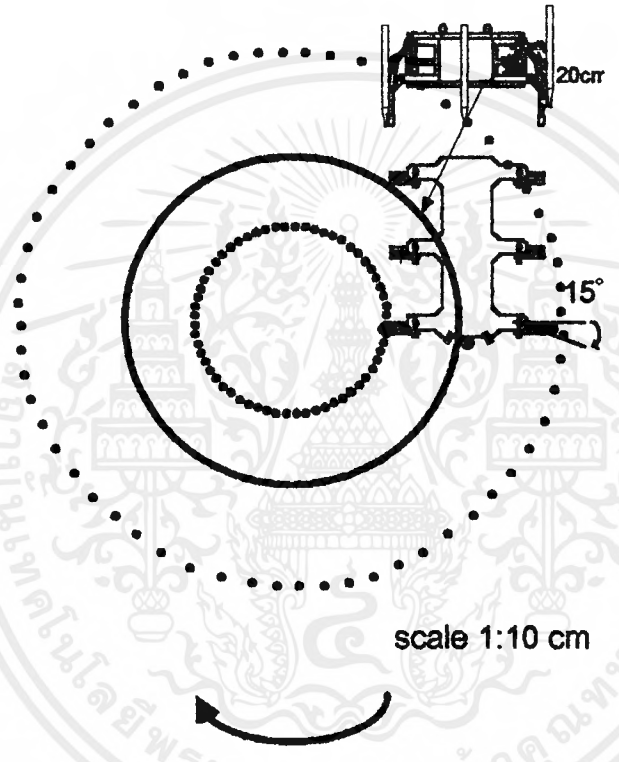
ข้อเสียของลำดับขั้นตอนการหมุนแบบนี้คือเกิดแรงเหวี่ยงฝืนขึ้นกับขาใดขาหนึ่งของข้างที่มีขาสัมพันธ์ ทำให้ขาดังกล่าวถูกลากเปลี่ยนตำแหน่งไปกับพื้น

สำหรับผลการทดลอง การหมุนตัว โดยใช้ค่าทดสอบมุมแกว่งขา 7 องศา ดังแสดงในรูปที่ 4.10 ได้แสดงลักษณะเส้นทางการหมุนตัวของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.0099 รอบ/วินาที



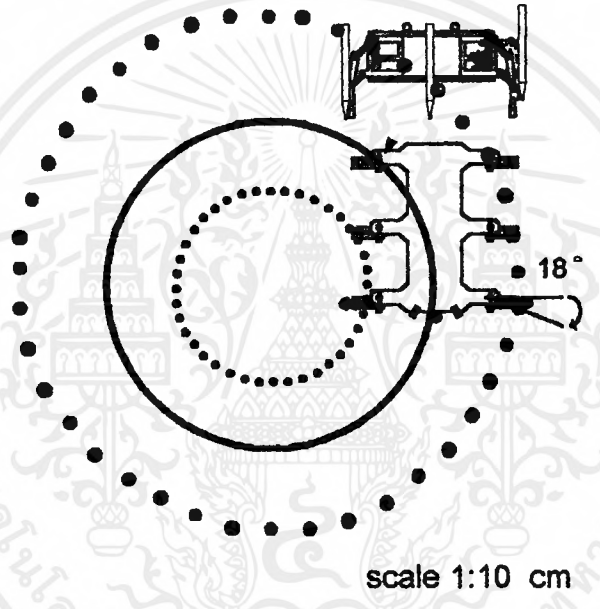
รูปที่ 4.10 แสดงผลการทดสอบการหมุนตัวด้วยมุมแกว่งขา 7 องศา

ผลการทดลอง การหมุนตัว โดยใช้ค่าทดสอบมุมแกว่งขา 15 องศา ดังแสดงใน รูปที่ 4.11 ได้แสดงลักษณะเส้นทางการหมุนตัวของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.0155 รอบ/วินาที



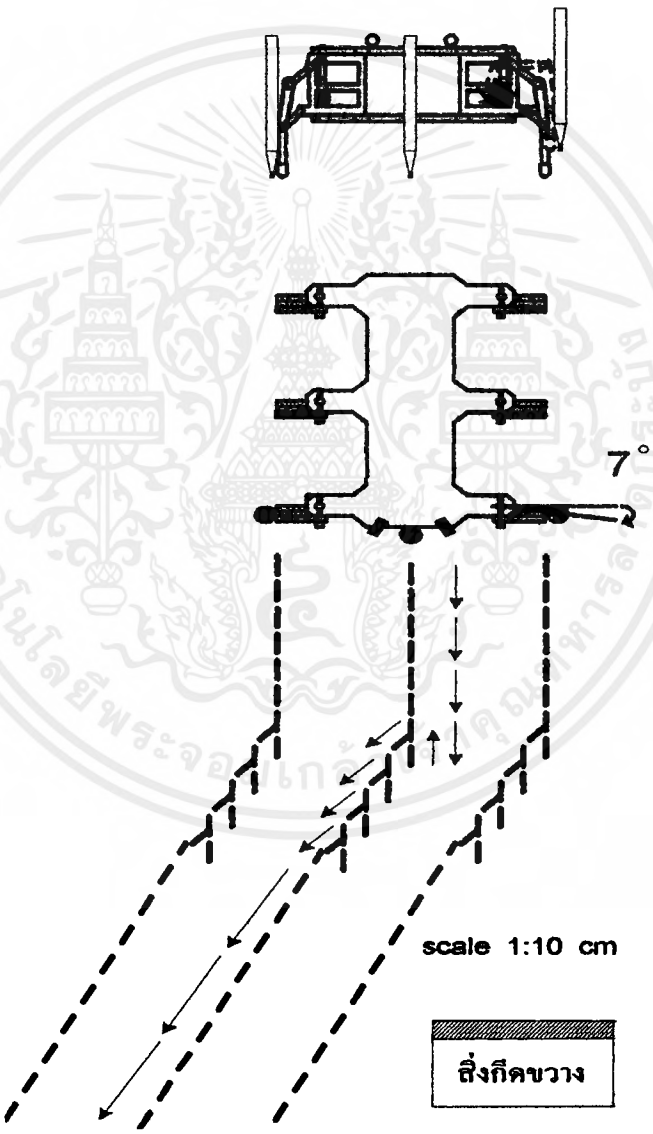
รูปที่ 4.11 แสดงผลการทดสอบการหมุนตัวด้วยมุมแกว่งขา 15 องศา

ผลการทดลอง การหมุนตัว โดยใช้ค่าทดสอบมุมก้าวขา 18 องศา ดังแสดง ในรูปที่ 4.12 ได้แสดงลักษณะเส้นทางการหมุนตัวของหุ่นยนต์ และมีความเร็วในการเคลื่อนที่ 0.0196 รอบ/วินาที



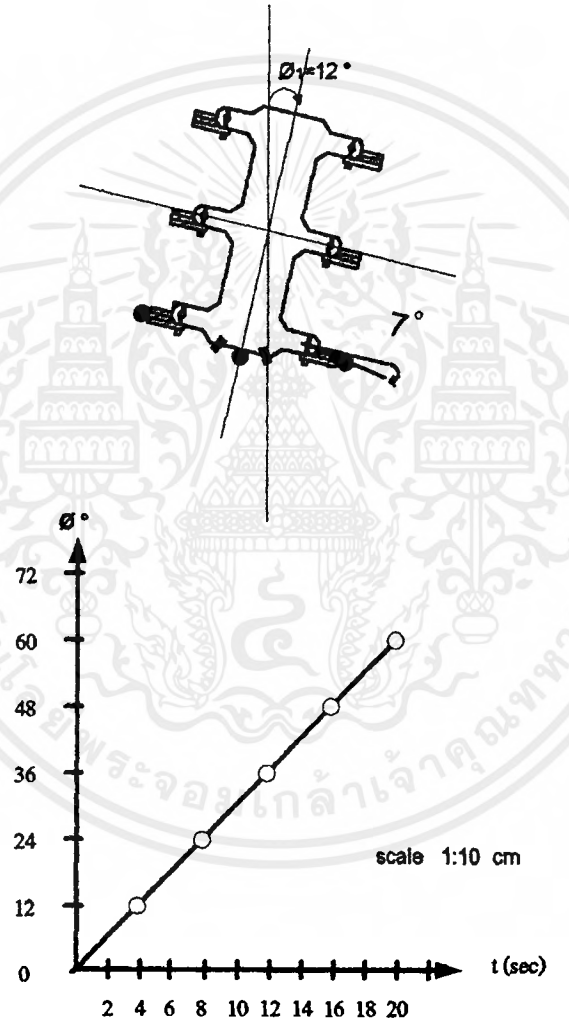
รูปที่ 4.12 แสดงผลการทดสอบการหมุนตัวด้วยมุมก้าวขา 18 องศา

ผลการทดลอง การเคลื่อนที่หลบสิ่งกีดขวาง โดยใช้ค่าทดสอบ มุมก้าวขา 7 องศา ดังแสดง ในรูปที่ 4.13 ได้แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวางของหุ่นยนต์ โดยมีสิ่งกีดขวางขนาด กว้าง 15 เซ็นติเมตร ยาว 25 เซ็นติเมตร สูง 30 เซ็นติเมตร โดยมีระยะตรวจจับสิ่งกีดขวาง 40 เซนติเมตร เมื่อพบสิ่งกีดขวาง หุ่นยนต์จะถอย สองก้าว และบิดตัวหลบสิ่งกีดขวาง และได้ผลการทดลองดังรูปที่ 4.13



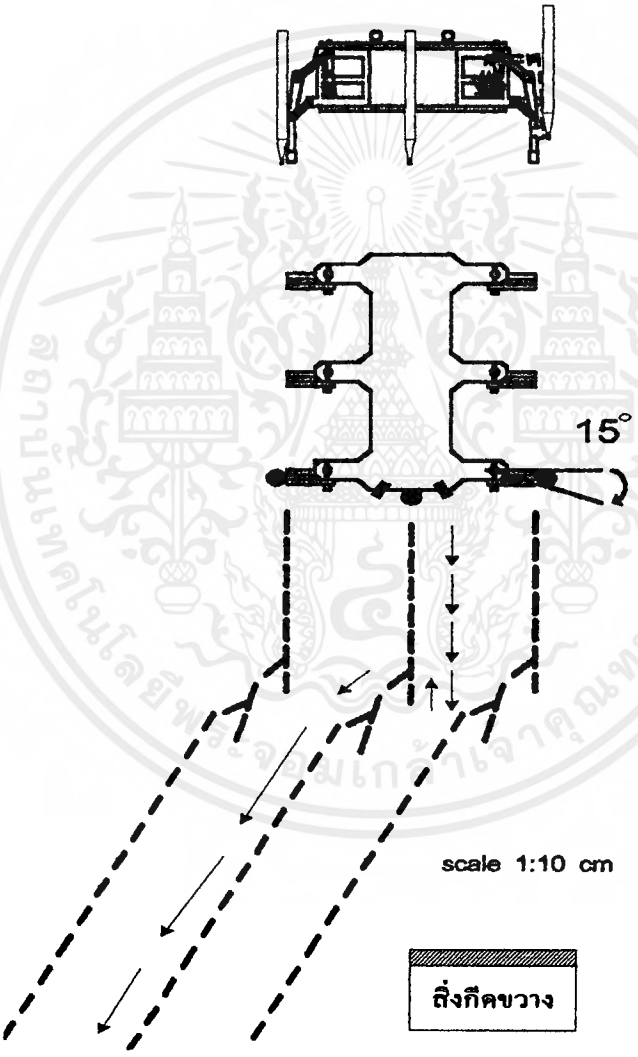
รูปที่ 4.13 ได้แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวาง ด้วยมุมก้าวขา 7 องศา

จากรูปที่ 4.13 ทำให้ทราบว่าในการหลบสิ่งกีดขวางของหุ่นยนต์ต้องทำการบิดตัวเพื่อหลบสิ่งกีดขวางดังนั้น จึงกำหนดให้ θ คือมุมบิดตัว ซึ่งในรูปที่ 4.14 ได้แสดง มุมการบิดตัวในแต่ละครั้ง เทียบกับเวลาที่ใช้ในการเคลื่อนที่หลบสิ่งกีดขวาง ในแต่ละครั้ง โดยกำหนดให้มุมของการก้าวขาที่ใช้ในการทดสอบ คือ 7 องศา



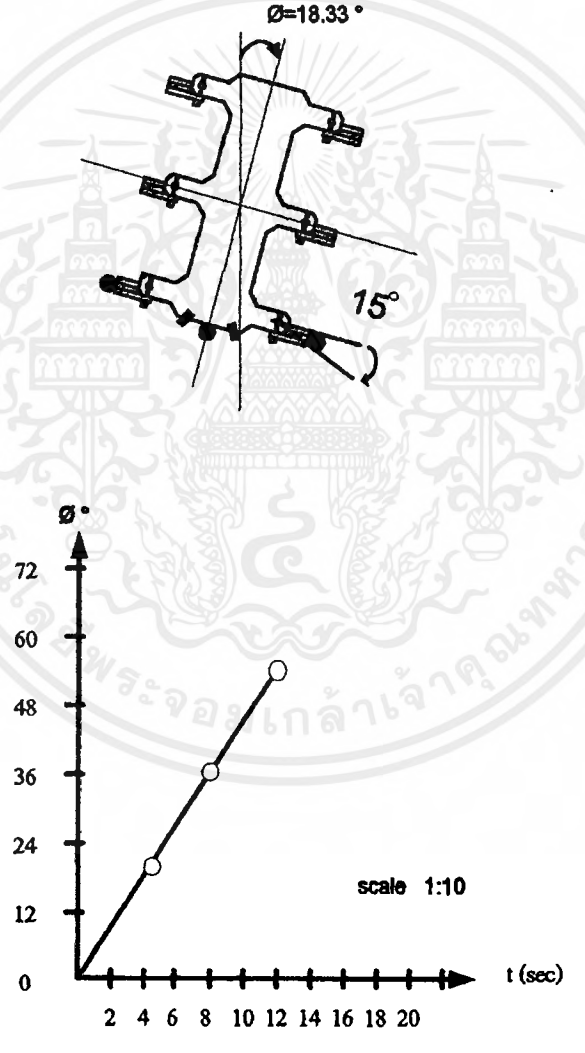
รูปที่ 4.14 กราฟแสดงการบิดตัวในแต่ละครั้ง ของหุ่นยนต์โดยใช้มุมของการก้าวขา7 องศา

ผลการทดลอง การเคลื่อนที่หลบสิ่งกีดขวาง โดยใช้ค่าทดสอบ มุมก้าวขา 15 องศา ดังแสดง ในรูปที่ 4.15 ได้แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวางของหุ่นยนต์ โดยมีสิ่งกีดขวางขนาด กว้าง 15 เซ็นติเมตร ยาว 25 เซ็นติเมตร สูง 30 เซ็นติเมตร โดยมีระยะตรวจจับสิ่งกีดขวาง 40 เซนติเมตร เมื่อพบสิ่งกีดขวาง หุ่นยนต์จะถอย สองก้าว และบิดตัวหลบสิ่งกีดขวาง และได้ผลการทดลองดังรูปต่อไปนี้



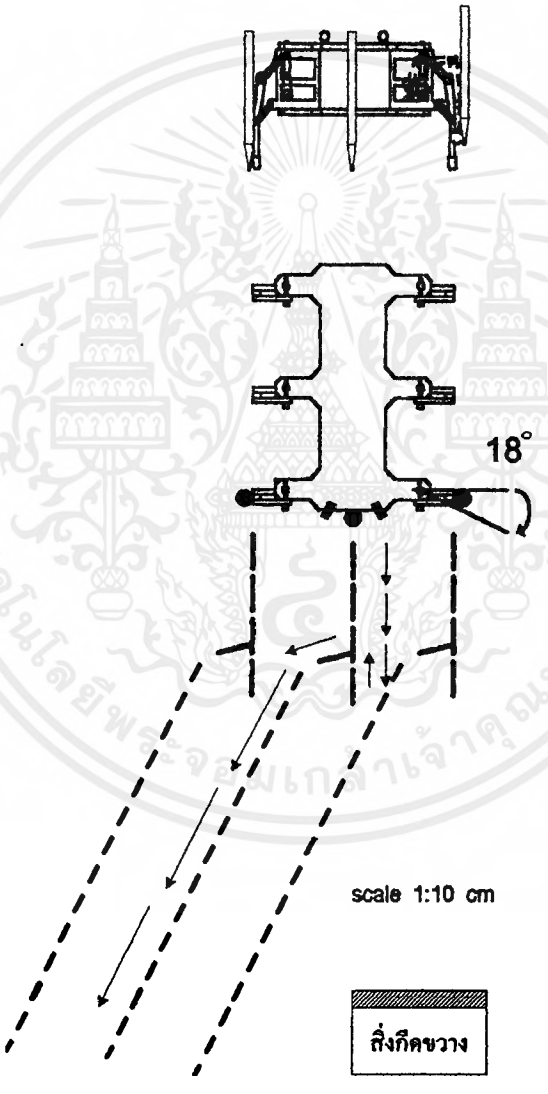
รูปที่ 4.15 ได้แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวาง ด้วยมุมก้าวขา 15 องศา

จากรูปที่ 4.15 ทำให้ทราบว่าในการหลบสิ่งกีดขวางของหุ่นยนต์ต้องทำการบิดตัวเพื่อหลบสิ่งกีดขวางดังนั้น จึงกำหนดให้ θ คือมุมบิดตัว ซึ่งในรูปที่ 4.16 ได้แสดง มุมการบิดตัวในแต่ละครั้ง เทียบกับเวลาที่ใช้ในการเคลื่อนที่หลบสิ่งกีดขวาง ในแต่ละครั้ง โดยกำหนดให้มุมของการแกว่งขาที่ใช้ในการทดสอบ คือ 15 องศา



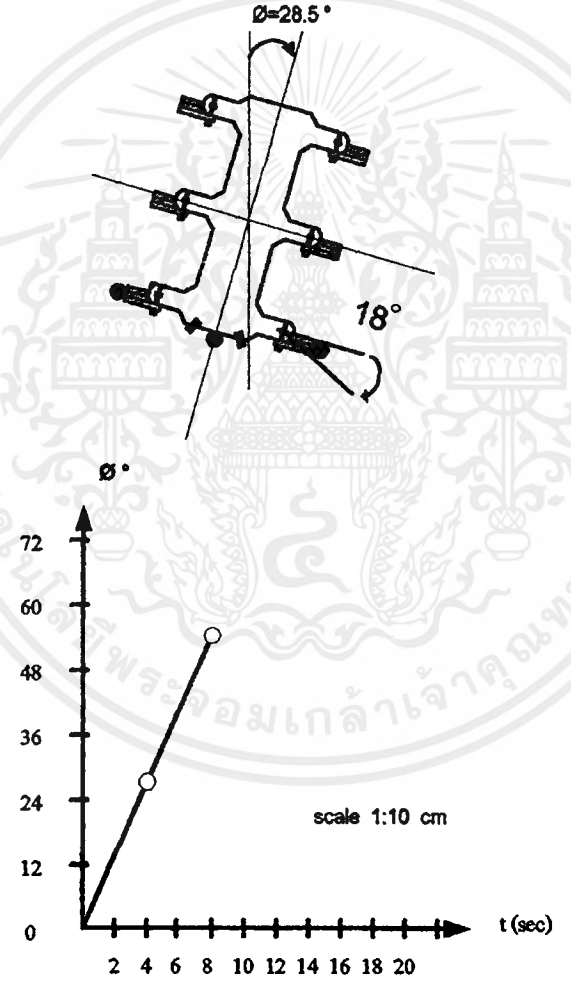
รูปที่ 4.16 กราฟแสดงการบิดตัวหุ่นยนต์ในแต่ละครั้ง โดยใช้มุมของการก้าวขา 15 องศา

ผลการทดลอง การเคลื่อนที่หลบสิ่งกีดขวาง โดยใช้ค่าทดสอบ มุมก้าวขา 18 องศา ดังแสดง ในรูปที่ 4.17 ได้แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวางของหุ่นยนต์ โดยมีสิ่งกีดขวางขนาด กว้าง 15 เซ็นติเมตร ยาว 25 เซ็นติเมตร สูง 30 เซ็นติเมตร โดยมีระยะตรวจจับสิ่งกีดขวาง 40 เซนติเมตร เมื่อพบสิ่งกีดขวาง หุ่นยนต์จะถอย สองก้าว และบิดตัวหลบสิ่งกีดขวาง และได้ผลการทดลองดังรูปต่อไปนี้



รูปที่ 4.17 แสดงลักษณะการเคลื่อนที่หลบสิ่งกีดขวาง ด้วยมุมก้าวขา 18 องศา

จากรูปที่ 4.17 ทำให้ทราบว่าในการหลบสิ่งกีดขวางของหุ่นยนต์ต้องทำการบิดตัว เพื่อหลบสิ่งกีดขวางดังนั้น จึงกำหนดให้ θ คือมุมบิดตัว ซึ่งในรูปที่ 4.18 ได้แสดง มุมการบิดตัวในแต่ละครั้ง เทียบกับเวลาที่ใช้ในการเคลื่อนที่หลบสิ่งกีดขวาง ในแต่ละครั้ง โดยกำหนดให้มุมของการก้าวขาที่ใช้ในการทดสอบ คือ 18 องศา



รูปที่ 4.18 กราฟแสดงการบิดตัวหุ่นยนต์ในแต่ละครั้ง โดยใช้มุมของการก้าวขา 18 องศา

บทที่ 5
สรุปโครงงาน ปัญหาและข้อเสนอแนะ

5.1 สรุปโครงงาน

หุ่นยนต์แมลงหกขาที่สร้างขึ้นมีลักษณะขาแบบเพนโท กรวฟ ในแต่ละขาประกอบด้วยเซอร์โวมอเตอร์สองตัว ตัวแรกทำหน้าที่ยกขึ้นขาในแนวระดับ สำหรับเซอร์โวมอเตอร์อีกตัวทำหน้าที่หมุนเพื่อควบคุมในการก้าวเดิน โดยที่เซอร์โวมอเตอร์ทั้งหมดถูกสั่งการด้วยไมโครคอนโทรลเลอร์ 68HC11 ตัวหุ่นยนต์ที่สร้างมีขนาด 5.2 × 11 ×12 นิ้ว มีน้ำหนัก 2 kg ขาหุ่นยนต์มีรัศมีการหมุน 2 นิ้ว แต่ละขามีการเคลื่อนไหวที่เป็นอิสระต่อกัน ใช้พลังงานจากแบตเตอรี่ขนาด 7.2 โวลท์ 1.7 แอมป์ต่อชั่วโมง ระยะเวลาใช้งาน 2~3 ชั่งโมง มีระบบตรวจจับสิ่งกีดขวางเป็นชนิดอินฟาเรด โดยมีระยะตรวจจับ 16 นิ้ว

จากการทดสอบการทำงานของหุ่นยนต์แมลงหกขา สามารถเคลื่อนที่ เดินหน้า ถอยหลัง เดินเป็นวงกลม และเคลื่อนที่หลบสิ่งกีดขวางได้ ในพื้นที่ราบ ซึ่งค่าที่ใช้ทดสอบในการก้าวขา 18 องศา ทำให้หุ่นยนต์แมลงหกขาสามารถเคลื่อนที่ได้เร็วที่สุด โดยมีความเร็ว 0.05263 เมตร/วินาที

ซึ่งในจุดที่ต้องปรับปรุงแก้ไข คือ ระบบแมคนิกส์ ในส่วนโครงสร้าง ส่วนประกอบของขาหุ่นยนต์ รวมถึงระบบการตรวจจับที่ขาหุ่นยนต์ การควบคุมความเร็ว และการควบคุมให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งที่ต้องการ

5.2 ปัญหาในการทำโครงงาน

5.2.1 ปัญหาในการสร้างตัวหุ่นยนต์

- ผู้จัดทำขาดความรู้ทางด้านเครื่องกล จึงไม่สามารถแก้ปัญหาที่เกิดขึ้นได้ทั้งหมด
- ขบวนการเลือกและจัดหาวัสดุที่นำมาสร้างหุ่นใช้เวลานานเกินควร
- ปัญหาความไม่มั่นคงของค่าเงินบาท ทำให้อุปกรณ์และวัสดุที่นำเข้าจากต่างประเทศมีราคาสูงกว่า งบประมาณที่ตั้งไว้

5.2.2 ปัญหาในการควบคุม

- ในการควบคุมการหมุนของเซอร์โวนั้นจะเคลื่อนที่ด้วยความเร็วคงที่ไม่สามารถเพิ่มหรือลดความเร็วในขณะการเคลื่อนที่ให้มีเสถียรภาพได้
- CPU ไม่สามารถทราบตำแหน่งการเคลื่อนที่ที่แท้จริงได้เพราะไม่มีการตรวจสอบตำแหน่งการเคลื่อนที่
- ขณะทำการเคลื่อนไหวมีการใช้กำลังงานจากแบตเตอรี่ค่อนข้างสูง

5.3 ข้อเสนอแนะ

ข้อเสนอแนะนี้จัดทำขึ้นเพื่อเป็นแนวทางในการพัฒนา หุ่นยนต์เดินหกขาให้มีความสามารถในการทำงานได้ดีขึ้น โดยจะนำเสนอเป็นข้อ ๆ ดังนี้

- ติดตั้งระบบตรวจสอบตำแหน่งการเคลื่อนที่ของขาทุกขา เพื่อ CPU จะได้ทราบตำแหน่งการเคลื่อนที่ที่แน่นอน
- ติดตั้งระบบประจุพลังงานสำหรับแบตเตอรี่
- พัฒนาโปรแกรมคอมพิวเตอร์ที่สามารถสั่งการให้หุ่น เคลื่อนที่ไปยังเป้าหมายโดยการชี้เป้าหมายบนหน้าจอคอมพิวเตอร์

ภาคผนวก ก.
ส่วนโปรแกรมควบคุม

โปรแกรมควบคุมการทำงานเขียนด้วยภาษาแอสแซมบลี้ของไมโครคอนโทรเลอร์ 68HC11 มีฟังก์ชั่นการทำงาน 16 ฟังก์ชั่นสามารถเลือกฟังก์ชั่นการทำงานได้ โดยทำการเซ็ทดิปสวิทช์ 4 บิท ที่ลำตัวด้านหลัง เมื่อสวิทช์ ON แสดงด้วยบิท 1 และสวิทช์ OFF แสดงด้วยบิท 0 หน้าที่การทำงานในแต่ละฟังก์ชั่น แสดงในตาราง

ตารางแสดงหน้าที่การทำงานของหุ่นในแต่ละฟังก์ชั่น

Function	DIP Switch 4 bit	หน้าที่การทำงาน
0	0000	ยืนอยู่กับที่
1	0001	เคลื่อนที่ไปด้านหน้า
2	0010	เคลื่อนที่ไปด้านหลัง
3	0011	เคลื่อนที่เป็นวงกลม วนด้านขวา
4	0100	เคลื่อนที่เป็นวงกลม วนด้านซ้าย
5	0101	เคลื่อนที่ไปด้านหน้าโดยหลบสิ่งกีดขวาง
6	0110	เคลื่อนที่ตามที่กำหนดไว้ในโปรแกรม
7	0111	ย่อตัวอยู่กับที่
8	1000	สำรองไว้
9	1001	ตรวจสอบอุปกรณ์ Senser ด้านขวา
10	1010	ตรวจสอบอุปกรณ์ Senser ด้านซาย
11	1011	ตรวจสอบอุปกรณ์ Senser ด้านซ้ายและขวา
12	1100	สำรองไว้
13	1101	สำรองไว้
14	1110	สำรองไว้
15	1111	สำรองไว้

```
*        Program DEMO Control SIX LEGGED INSECT ROBOT.        *

REGBAS   EQU   $1000
P8255_A  EQU   $1200          ; Port A 8255.
P8255_B  EQU   $1201          ; Port B 8255.
P8255_C  EQU   $1202          ; Port C 8255.
CTL_P82  EQU   $1203          ; Port Control 8255.
TCNT     EQU   $100E          ; Timer Counter Register.
TOC2     EQU   $1018          ; Input Capture2 Register.
TFLG1    EQU   $1023          ; Timer Interrupt Flag Register.
LL_UP    EQU   $0AA0
LL_DOWN  EQU   $03A0
LR_UP    EQU   $07A0
LR_DOWN  EQU   $0EA0
HIP_CEN  EQU   $0AF0
HIP_FOR  EQU   $0CF0
HIP_BAK  EQU   $08F0

* ————Set address Variable————
P_OUT    EQU   $2400
EYE      EQU   $2402
W_COUNT  EQU   $2403
PER      EQU   $2404
COUNT    EQU   $2405
C_ASS    EQU   $2406
BIN      EQU   $2407
MODE     EQU   $2408
P_ON_1   EQU   $2409
P_ON_2   EQU   $240B
TEMP     EQU   $240D
P1_TMP   EQU   $240F
P2_TMP   EQU   $2411

         ORG $A000

SET_SP   LDS   #$2300         ; Set stackpointer.

         LDA   #$00
SYS_DLY  DECA                 ; Wait 8255 Port rady.
         BNE   SYS_DLY

INIT_P   LDAA  #$89           ; Set port A -> Output mode 0.
         STAA  CTL_P82              port B -> Output mode 0.
*                                   port C -> Input mode 0.

MAIN     LDX   #P8255_A
WAIT     BRSET $02,X $0C WAIT ; Wait press buttom.

DEMO     LDAA  #$01           ; Set mode .
         STAA  MODE
         LDAA  P8255_C        ; Load function from DIP_SW.
```

```
        ANDA    #$F0            ; Extract function value.

        CMPA    #$F0            ; function 0 yes or no.
        BNE     NEXT_1          ; Don't function jump to next_1.
        JSR     STAND           ; function 0 call STAND routine.
        BRA     DEMO            ; branch to demo label.

NEXT_1  CMPA    #$E0            ; function 1 yes or no.
        BNE     NEXT_2          ; same it up.
        JSR     FORW
        BRA     DEMO

NEXT_2  CMPA    #$D0
        BNE     NEXT_3
        JSR     REW
        BRA     DEMO

NEXT_3  CMPA    #$C0
        BNE     NEXT_4
        JSR     TURN_R
        BRA     DEMO

NEXT_4  CMPA    #$B0
        BNE     NEXT_5
        JSR     TURN_L
        BRA     DEMO

NEXT_5  CMPA    #$A0
        BNE     NEXT_6
        JSR     PROG_1
        BRA     DEMO

NEXT_6  CMPA    #$90
        BNE     NEXT_7
        JSR     DANCE
        BRA     DEMO

NEXT_7  CMPA    #$80
        BNE     NEXT_8
        JSR     DROP
        BRA     DEMO

NEXT_8  CMPA    #$60
        BNE     NEXT_9
        LDAA    #$40
        JSR     TEST_IR
        BRA     DEMO

NEXT_9  CMPA    #$50
        BNE     NEXT_10
        LDAA    #$80
        JSR     TEST_IR
        BRA     DEMO

NEXT_10 CMPA    #$40
        BNE     NEXT_11
        LDAA    #$C0
```

```
            JSR     TEST_IR

NEXT_11     BRA     DEMO                ; Not all function branch to DEMO.



*-------------------------------------------*
*           Advance Moving Routine.         *
*-------------------------------------------*
*     Routine function 0 Standing mode
STAND       LDAA    #20                 ; set peraid for stand mode.
            STAA    PER
            LDD     #STEP_0             ; set Table motion.
            STD     TEMP
            JSR     WALK                ; call WALK rotion.
            RTS
* ---------------------------------------

DROP        LDAA    #20                 ; set peraid for drop mode.
            STAA    PER
            LDD     #STEP_11            ; set Table motion.
            STD     TEMP
            JSR     WALK                ; call WALK rotion.
            RTS

*--------Routine Motion detect obstacle----------
PROG_1      PSHX
            LDX     #$2500              ; Set pointer Index.
            LDAA    #$00                ; Clear value detect.
            STAA    EYE

            LDAA    #$80                ; Eye left.
            STAA    P8255_A
            JSR     DELAY_T             ; wait detecting.
            LDAA    P8255_C             ; load detecting.
            ANDA    #$02                ; extract Value.
            BEQ     DETEC_1             ; Don't have obstacle jump to next
.LABEL
            BSET    EYE,%10000000       ; have obstacle set value.

DETEC_1     LDAA    #$40                ; Eye right.
            STAA    P8255_A
            JSR     DELAY_T             ; same it up.
            LDAA    P8255_C
            ANDA    #$01
            BEQ     DETEC_2
            BSET    EYE,%01000000

DETEC_2     LDAA    EYE                 ; Discover front obstacle.
            CMPA    #$C0                ; Yes or No.
            BNE     DETEC_3             ; no jump to DETECT_3.
            JSR     REW                 ; Yes, to ran away from obstacle.
            JSR     REW
            JSR     REW
            JSR     TURN_L
```

```
                JSR     TURN_L
                JSR     REW
                JSR     TURN_L
                JSR     TURN_L
                JSR     TURN_L
                BRA     END_PROG_1

DETEC_3 CMPA    #$80                    ; Discover left obstacle.
                BNE     DETEC_4         ; Yes or No, no jump to DETECT_4.
                JSR     REW             ; Yes, to ran away from obstacle.
                JSR     TURN_R
                JSR     TURN_R
                BRA     END_PROG_1

DETEC_4 CMPA    #$40                    ; Discover left obstacle.
                BNE     DETEC_5         ; Yes or No, no jump to DETECT_5.
                JSR     REW             ; Yes, to ran away from obstacle.
                JSR     TURN_L
                JSR     TURN_L
                BRA     END_PROG_1

DETEC_5 JSR     FORW                    ; Don't have obstacle go ahaed.

END_PROG_1
                PULX
                RTS
* ---------------------------------------
*       Routine Delay wait detecting.
* ---------------------------------------
DELAY_T LDAA    #$58
DL_1T   LDAB    #$00
DL_2T   DECB
                BNE     DL_2T
                DECA
                BNE     DL_1T
                RTS
* ---------------------------------------
*       Routine for test IR Detector.
* ---------------------------------------
TEST_IR STAA    P8255_A
                BSR     DELAY_T
                RTS


*---------------------------------------
*       Routine pattern motion          *
*---------------------------------------
DANCE   JSR     FORW
                JSR     FORW
                JSR     FORW
                JSR     WAVE
                JSR     REW
                JSR     REW
                JSR     REW
                JSR     WAVE
                JSR     FORW
                JSR     TURN_L
```

```
        JSR     TURN_L
        JSR     TURN_L
        JSR     WAVE
        JSR     FORW
        JSR     FORW
        JSR     WAVE
        JSR     TURN_R
        JSR     TURN_R
        JSR     TURN_R
        JSR     WAVE
        RTS
```

```
*-------------------------------------------*
*              Basic Moving Routine.        *
*-------------------------------------------*

FORW    LDAA    #13             ; Routine load table motion .
        STAA    PER             ; for go ahead.
        LDD     #STEP_1
        STD     TEMP
        JSR     WALK

        LDD     #STEP_2
        STD     TEMP
        JSR     WALK

        LDD     #STEP_3
        STD     TEMP
        JSR     WALK

        LDD     #STEP_4
        STD     TEMP
        JSR     WALK
        RTS

REW     LDAA    #13             ; Routine load table motion.
        STAA    PER             ; for to go back.
        LDD     #STEP_3
        STD     TEMP
        JSR     WALK

        LDD     #STEP_2
        STD     TEMP
        JSR     WALK

        LDD     #STEP_1
        STD     TEMP
        JSR     WALK

        LDD     #STEP_4
```

```
        STD     TEMP
        JSR     WALK
        RTS


TURN_L  LDAA    #15             ; Routine load table motion.
        STAA    PER             ; for turn left.

        LDD     #STEP_1
        STD     TEMP
        JSR     WALK

        LDD     #STEP_5
        STD     TEMP
        JSR     WALK

        LDD     #STEP_3
        STD     TEMP
        JSR     WALK

        LDD     #STEP_6
        STD     TEMP
        JSR     WALK

        RTS

TURN_R  LDAA    #15             ; Routine load table motion.
        STAA    PER             ; for turn rigth.
        LDD     #STEP_3
        STD     TEMP
        BSR     WALK

        LDD     #STEP_7
        STD     TEMP
        BSR     WALK

        LDD     #STEP_1
        STD     TEMP
        BSR     WALK

        LDD     #STEP_8
        STD     TEMP
        BSR     WALK

        RTS

WAVE    LDAA    #15             ; Routine load table motion.
        STAA    PER PER         ; for Hip wave motion.

        LDD     #STEP_1
        STD     TEMP
        BSR     WALK

        LDD     #STEP_0
        STD     TEMP
        BSR     WALK
```

```
        LDD     #STEP_9
        STD     TEMP
        BSR     WALK

        LDD     #STEP_10
        STD     TEMP
        BSR     WALK

        LDD     #STEP_9
        STD     TEMP
        BSR     WALK

        LDD     #STEP_10
        STD     TEMP
        BSR     WALK

        LDD     #STEP_0
        STD     TEMP
        BSR     WALK

        LDD     #STEP_11
        STD     TEMP
        BSR     WALK

        LDD     #STEP_0
        STD     TEMP
        BSR     WALK
        RTS
*-------------------------------------------------------------------
* WALK Routine read  motion Table for translate pulse generator routine.
*-------------------------------------------------------------------
WALK    PSHX
        LDAA    PER             ; period for 1 loop.
        STAA    COUNT

LP1     LDAA    #$06            ; Set parameter.
        STAA    C_ASS
        LDAA    #$01            ; Set bit for servo.
        STAA    BIN
        LDX     TEMP            ; Load pointer .
ASSINE  LDD     0,X
        STD     P_ON_1          ; load time T_ON for servo  lift.
        LDD     2,X
        STD     P_ON_2          ; load time T_ON for servo rotate.
        BSR     MOVE            ; Branch to  pulse generator Routine.
        ASL     BIN             ; Shift bit for next servo .
        LDAB    #$04            ; Slide X pointer fill.
        ABX
        DEC     C_ASS           ; Count loop cover any legs.
        BNE     ASSINE

        BSR     M_DELY          ; call delay for  T_OFF.
        DEC     COUNT           ; Count period.
        BNE     LP1
        PULX
```

```
            RTS

M_DELY      LDAA    COUNT           ; Routine Delay for T_OFF.
            ADDA    #7
M_D_1       LDAB    #$00
M_D_2       DECB
            BNE     M_D_2
            DECA
            BNE     M_D_1
END_D       RTS

*-------------------------------------------------------------
*  MOVE Routine Generate pulse from P_ON_1 , P_ON_2.
*-------------------------------------------------------------
MOVE        PSHX
            LDX     #REGBAS         ; set  pointer register system.

            LDD     TCNT            ; set parameter.
            STD     TOC2
MOVE1       LDAA    BIN             ; set pulse ON.
            STAA    P8255_A
            LDAA    #$40            ; set Flag Timer.
            STAA    TFLG1
            LDD     TCNT            ; load Time at present.
            ADDD    P_ON_1          ; adding T_ON from motion table.
            STD     TOC2            ; keeping to TOC2 for compare.
MOVE2       BRCLR   TFLG1,X $40 MOVE2   ; wait present time = TOC2.
            LDAA    #$00            ; set pulse OFF.
            STAA    P8255_A
            LDAA    #$40
            STAA    TFLG1

            LDD     TCNT
            STD     TOC2
MOVE3       LDAA    BIN             ; same it up.
            STAA    P8255_B
            LDAA    #$40
            STAA    TFLG1
            LDD     TCNT
            ADDD    P_ON_2
            STD     TOC2
MOVE4       BRCLR   TFLG1,X $40 MOVE4
            LDAA    #$00
            STAA    P8255_B
            LDAA    #$40
            STAA    TFLG1
            PULX
            RTS

*-------------------------------------------------------------
*           Step walk position SERVO (motion table).
*-------------------------------------------------------------
STEP_0      DW      LL_DOWN
            DW      HIP_CEN
            DW      LR_DOWN
            DW      HIP_CEN
            DW      LL_DOWN
```

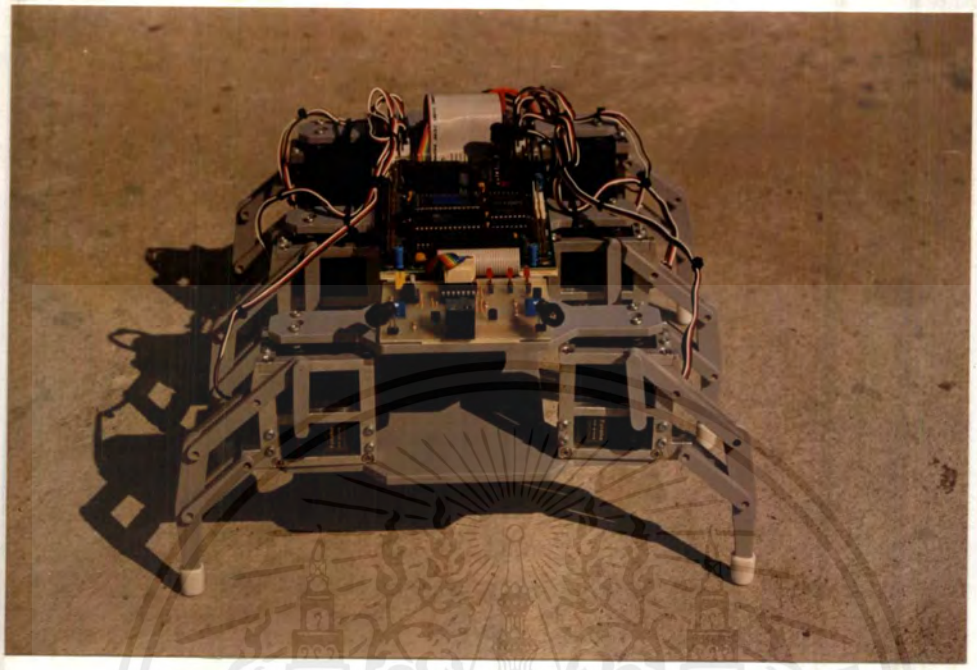
```
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN

STEP_1  DW      LL_UP
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN

STEP_2  DW      LL_DOWN
        DW      HIP_FOR
        DW      LR_DOWN
        DW      HIP_FOR
        DW      LL_DOWN
        DW      HIP_BAK
        DW      LR_DOWN
        DW      HIP_BAK
        DW      LR_DOWN
        DW      HIP_FOR
        DW      LL_DOWN
        DW      HIP_FOR

STEP_3  DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN

STEP_4  DW      LL_DOWN
        DW      HIP_BAK
        DW      LR_DOWN
        DW      HIP_BAK
        DW      LL_DOWN
        DW      HIP_FOR
        DW      LR_DOWN
        DW      HIP_FOR
        DW      LR_DOWN
        DW      HIP_BAK
```

```
        DW      LL_DOWN
        DW      HIP_BAK

STEP_5  DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_FOR
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_FOR

STEP_6  DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_BAK
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_BAK

STEP_7  DW      LL_DOWN
        DW      HIP_BAK
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_BAK
        DW      LL_DOWN
        DW      HIP_CEN

STEP_8  DW      LL_DOWN
        DW      HIP_FOR
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_FOR
        DW      LL_DOWN
        DW      HIP_CEN

STEP_9  DW      LL_UP
        DW      HIP_CEN
```

```
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN

STEP_10 DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_DOWN
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LR_DOWN
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN

STEP_11 DW      LL_UP
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LR_UP
        DW      HIP_CEN
        DW      LL_UP
        DW      HIP_CEN

        ORG $FFFE
        DW      $A000
```

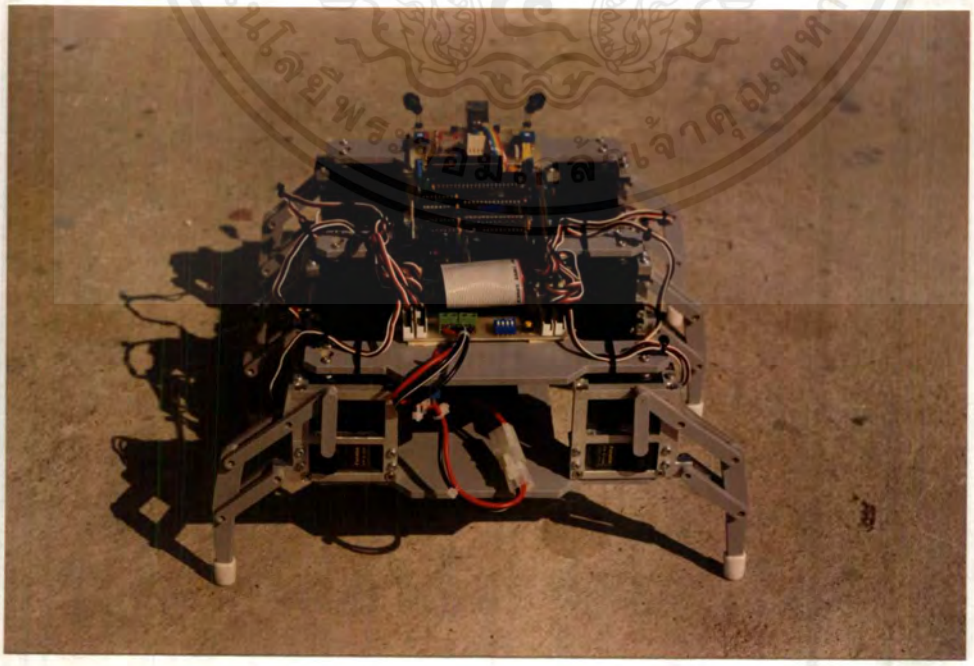
ภาคผนวก ข.
รูปถ่ายตัวหุ่นยนต์

ลักษณะของหุ่นยนต์แมลงหกขาที่สร้างเสร็จแล้วมีรูปร่างลักษณะด้านหน้าของตัวหุ่นยนต์ ซึ่งแสดงไว้ดังรูปที่ ข.1 ดังต่อไปนี้



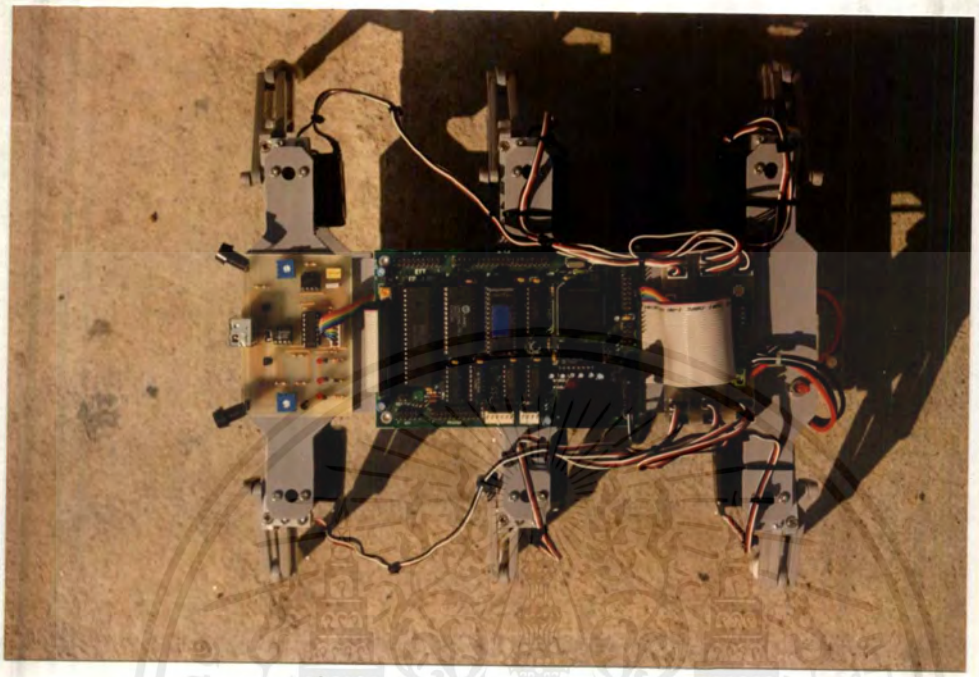
รูปที่ ข.1 แสดงด้านหน้าของตัวหุ่นยนต์

ลักษณะของหุ่นยนต์แมลงหกขาที่สร้างเสร็จแล้วมีรูปร่างลักษณะด้านหลังของตัวหุ่นยนต์ ซึ่งแสดงไว้ ดังรูปที่ ข.2 ดังต่อไปนี้



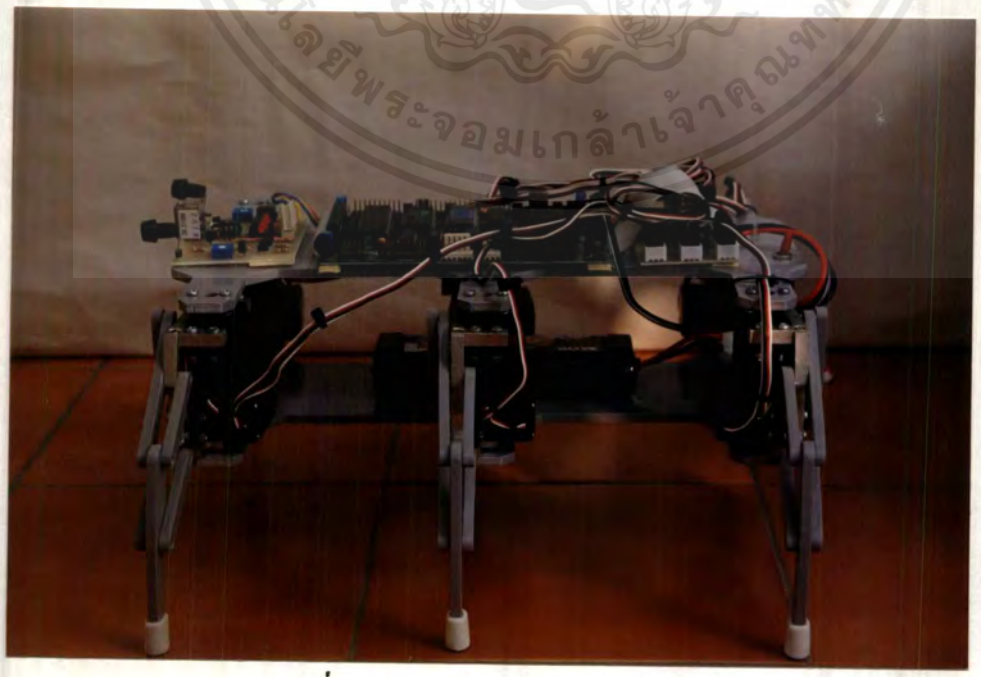
รูปที่ ข.2 แสดงด้านหลังตัวหุ่นยนต์

ลักษณะของหุ่นยนต์แมลงหกขาที่สร้างเสร็จแล้วมีรูปร่างลักษณะด้านบนของตัวหุ่นยนต์ ซึ่งแสดงไว้ดังรูปที่ ข.3 ดังต่อไปนี้



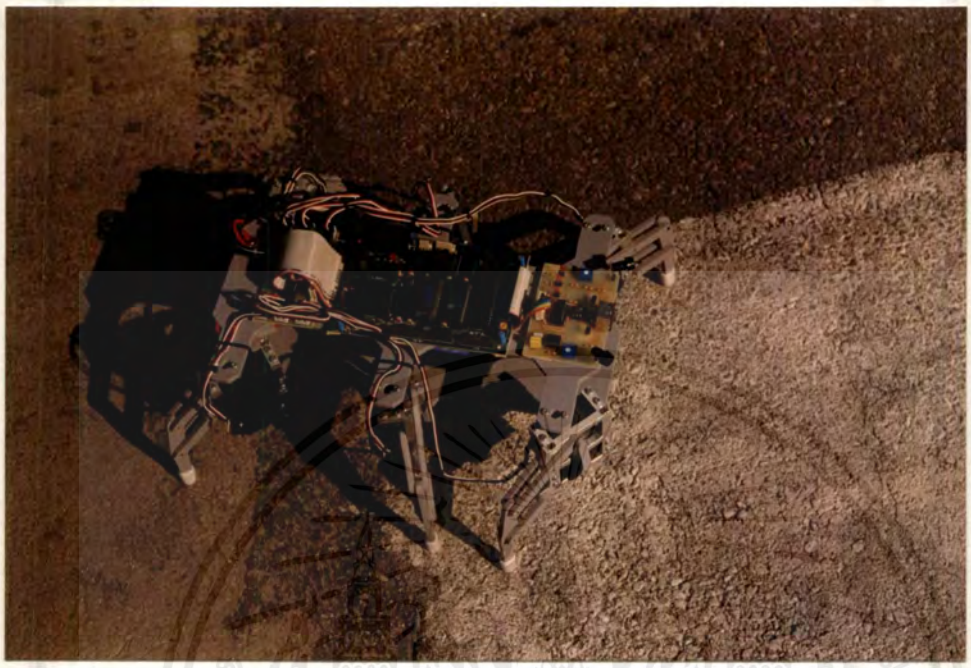
รูปที่ ข.3 แสดงด้านบนของตัวหุ่นยนต์

ลักษณะของหุ่นยนต์แมลงหกขาที่สร้างเสร็จแล้วมีรูปร่างลักษณะด้านข้างของตัวหุ่นยนต์ ซึ่งแสดงไว้ดังรูปที่ ข.4 ดังต่อไปนี้



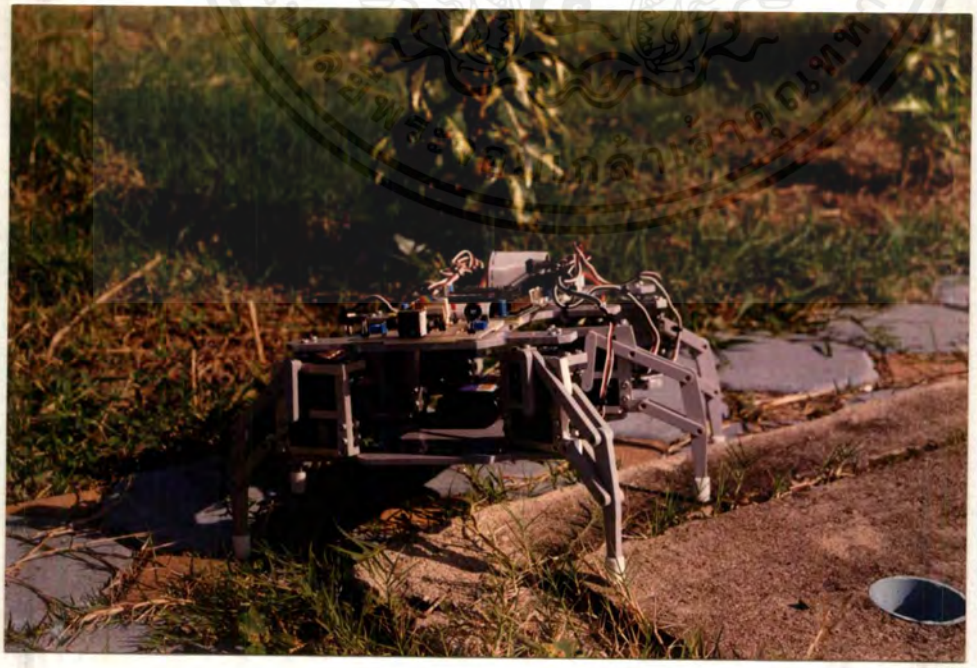
รูปที่ ข.4 แสดงด้านข้างของตัวหุ่นยนต์

ลักษณะการเดินของหุ่นยนต์แมลงหกขา สามารถเดินบนพื้นซีเมนต์ซึ่งเป็นพื้นที่เรียบได้ ซึ่งแสดงไว้ดังรูปที่ ข.5 ดังต่อไปนี้



รูปที่ ข.5 แสดงการเดินของหุ่นยนต์บนพื้นที่เรียบ

ลักษณะการเดินของหุ่นยนต์แมลงหกขา สามารถเดินบนพื้นที่ขรุขระได้ ซึ่งแสดงไว้ดังรูปที่ ข.6 ดังต่อไปนี้



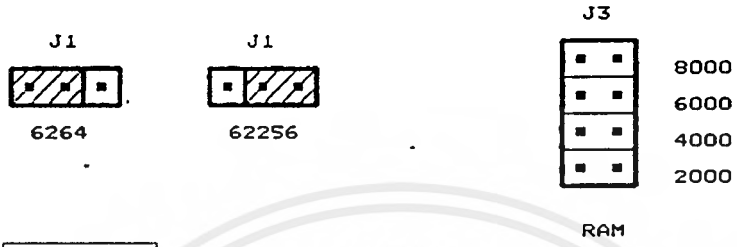
รูปที่ ข.6 แสดงการเดินของหุ่นยนต์บนพื้นที่ไม่เรียบ

ภาคผนวก ค.

รายละเอียดของบอร์ด CP- 68HC11

RAM

สามารถเลือกใช้ RAM ได้ 2 เบอร์ คือ 6264 (8K) และ 62256 (32K) โดยเลือกที่ JUMPER J1 สำหรับการกำหนดตำแหน่งของหน่วยความจำสามารถเลือกได้ที่ JUMPER J3ซึ่งการใส่ JUMPER 1 ตัวสามารถอ้างตำแหน่งในหน่วยความจำได้ 8K ถ้าใส่ครบทั้ง 4 ตัวก็สามารถอ้างตำแหน่งได้ 32K โดยที่ตำแหน่งของหน่วยความจำจะอยู่ในช่วง $2000-$9FFF



EPROM

สามารถเลือกใช้ EPROM ได้ 2 เบอร์ คือ 2764 (8K) และ 27256 (32K) โดยเลือกที่ JUMPER J2 สำหรับการกำหนดตำแหน่งของหน่วยความจำสามารถเลือกได้ที่ JUMPER J4 ซึ่งการใส่ JUMPER 1 ตัวสามารถอ้างตำแหน่งในหน่วยความจำได้ 8K ถ้าใส่ครบทั้ง 4 ตัวก็สามารถอ้างตำแหน่งได้ 24K โดยที่ตำแหน่งของหน่วยความจำจะอยู่ในช่วง $A000-$FFFF

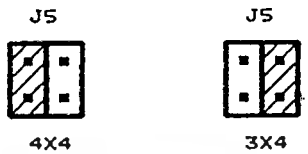
I/O PORT 8255

เป็น CONNECTOR ขนาด 34 PIN ซึ่งต่อออกมาจากพอร์ตของ 8255 โดยมีลักษณะขาเหมือนกับ 72IO CONNECTOR สามารถต่อกับบอร์ดสนับสนุนของทาง ETT ได้ สำหรับ ADDRESS ที่ใช้ในการติดต่อกับพอร์ตของ 8255 มีรายละเอียดดังนี้

PORT A	$1200
PORT B	$1201
PORT C	$1202
CONTROL PORT	$1203

KEYBOARD

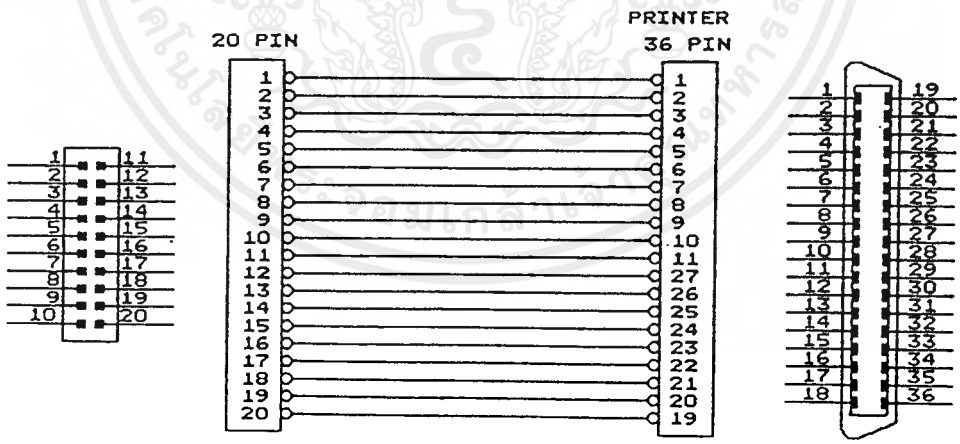
เป็น CONNECTOR ขนาด 10 PIN ซึ่งต่อมาจาก PORT C ของ8255 สามารถต่อเข้ากับ KEYBOARD แบบ 4x4 หรือ 3x4 ได้โดยการเลือกที่ JUMPER J5 ในกรณีที่เลือก KEYBOARD เป็นแบบ 3x4ขา PC0 ของ 8255 จะถูกนำมาใช้เป็นสัญญาณ BUSY เพื่ออ่านสถานะของเครื่องพิมพ์ ส่วนในกรณีที่เลือก KEYBOARD เป็นแบบ 4x4 จะไม่สามารถใช้ PRINTER ได้เนื่องจากขา PC0 ของ 8255 ถูกต่อเข้ากับ CONNECTOR 10 PIN ของ KEYBOARD แทน



PRINTER PORT

เป็น CONNECTOR ขนาด 20 PIN ใช้ต่อกับเครื่องพิมพ์ธรรมดาทั่วๆไป สำหรับสัญญาณ STOBE นั้นจะใช้ OUTPUT ของ 138 ซึ่งได้จากการ DECODE PORT โดยการส่ง data อะไรก็ได้ออกไปที่ PORT $1800ก็จะเป็นการ STOBE ข้อมูล สำหรับ ADDRESS ที่ใช้ในการติดต่อกับเครื่องพิมพ์มีรายละเอียดดังนี้

DATA	PB0-PB7	$1201
STOBE ขา 11 ของ	138	$1800
BUSY	PC0	$1202



LCD

เป็น CONNECTOR ขนาด 20 PIN สามารถใช้กับ LCD MODULE ได้ 2แบบ คือแบบ CHARACTOR และ แบบ GRAPHIC โดยเลือกที่ JUMPER J6 และสามารถปรับความสว่างของจอ LCD ได้โดยการปรับที่ VR 10K

สำหรับ ADDRESS ที่ใช้ในการติดต่อกับ LCD มีรายละเอียดดังนี้

WRITE DATA INSTRUCTION	$1400
READ BUSY FLAG ANND ADDRESS	$1400
WRITE DATA	$1401
READ DATA	$1401

สำหรับ LCD แบบ GRAPHIC

PAGE 1 อยู่ที่ ADDRESS $1400-$1401

PAGE 2 อยู่ที่ ADDRESS $1402-$1403

A TO D

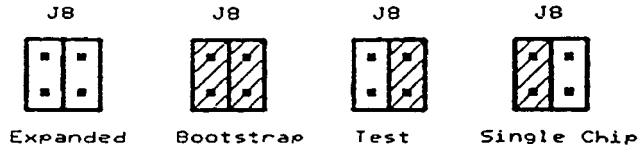
เป็น CONNECTOR ขนาด 14 PIN ซึ่งต่อออกมาจากขา PE0-PE7 ของ 68HC11 เป็น A/D ขนาด 8 บิต 8 CHANNEL โดยใช้แรงดันอ้างอิงภายในบอร์ด 5V สามารถใช้แรงดันอ้างอิงจากภายนอกได้ โดยให้แรงดัน อยู่ในช่วง 2.5 - 5V โดยการถอด JUMPER J9 ออกซึ่งเป็น CONNECTOR ขนาด 4 PIN

TIMER/COUNTER

เป็น CONNECTOR ขนาด 10 PIN ซึ่งต่อออกมาจากขา PA0-PA7 ของ 68HC11 สามารถนำไปประยุกต์ ได้หลายอย่าง เช่น นับความถี่,นับจำนวนพัลส์,คาบเวลา,กำเนิดสัญญาณ SQUARE WAVE หรือจะใช้เป็นพอร์ด I/O ก็ได้

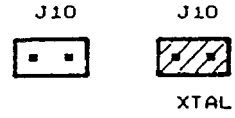
MODE (J8)

เป็น CONNECTOR 4 PIN ใช้ในการเลือก MODE การทำงานของ CPU ซึ่งสามารถกำหนดโหมดการทำงานได้ 4 โหมด คือ Single Chip , Expanded , Bootstrap และ Test mode แต่การทำงานโดยทั่วๆไปจะอยู่ที่โหมด Expanded Multiplexed ซึ่งเป็นการติดต่อกับหน่วยความจำภายนอก



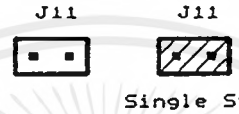
XTAL (J10)

เป็น CONNECTOR 2 PIN ใช้ในกรณีที่ต้องการนำสัญญาณนาฬิกาความถี่ 8 MHz ออกไปยัง CONNECTOR 40 PIN



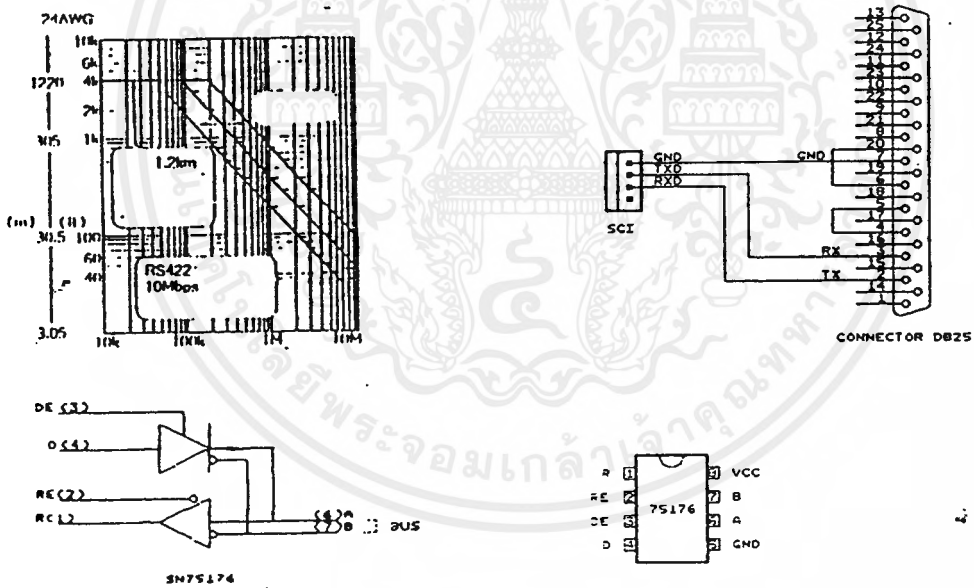
JOC5 (J11)

เป็น CONNECTOR 2 PIN ซึ่งต่ออยู่ระหว่างขา OC5 กับขา XIRQ ของ 68HC11 ซึ่งจะถูกใช้ในกรณีที่ต้องการทำ SINGLE STEP โดยการใช้ร่วมกับ ET-DEBUGGER 68HC11 เป็นการทำ SINGLE STEP โดยใช้ HARDWARE ซึ่งจะทำให้เกิด INTERRUPT ขึ้นที่ขา XIRQ ดังนั้นในกรณีที่ผู้ใช้ต้องการนำขา OC5 ไปใช้งานอื่นควรนำ JUMPER นี้ออกด้วย



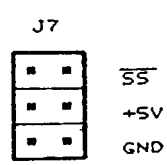
SCI

เป็น CONNECTOR ขนาด 4 PIN ใช้ทางด้าน SERIAL COMMUNICATION INTERFACE โดยต่อเข้ากับ MAX232 สามารถใช้เป็น RS422 ได้โดยการถอด MAX232 ออก แล้วใส่ 75176 แทนและสามารถหาค่า RZ ได้จากกราฟ



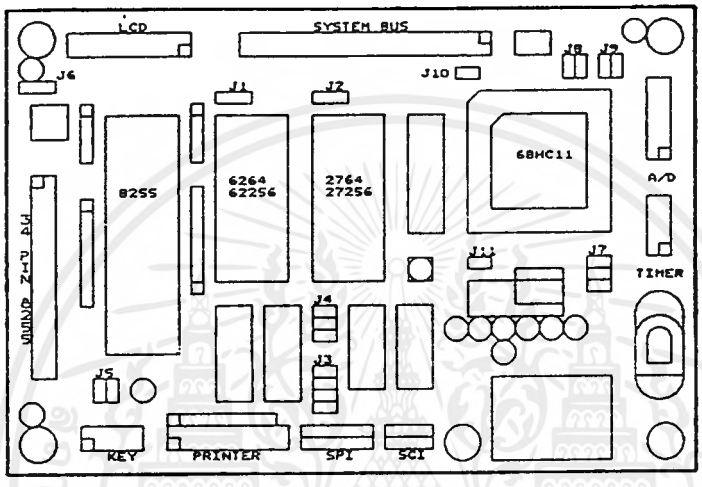
SPI

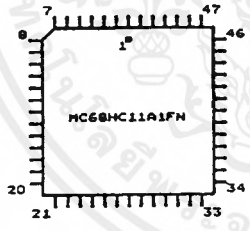
เป็น CONNECTOR ขนาด 6 PIN ใช้ทางด้าน SERIAL PERIPHERAL INTERFACE ซึ่งเป็น HIGH - SPEED SYNCHRONOUS SERIAL I/O โดยมี JUMPER J7 เป็นตัวเลือกว่าจะให้ต่อขา SS ของ 68HC11 ต่อเข้ากับ +5V,GND หรือต่อไปยัง CONNECTOR

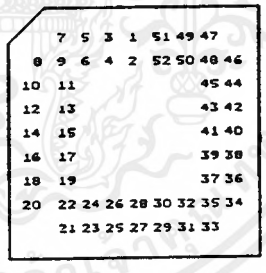


คำแนะนำในการประกอบวงจร

ควรใส่อุปกรณ์ที่มีขนาดเล็กก่อน เช่น ตัวต้านทาน ไดโอด ตัวเก็บประจุ และใส่ให้ถูกขั้ว ด้วย สำหรับการ ใส่ SOCKET 52 PIN PLCC นั้นที่ SOCKET และที่ไอซีจะมีรอยบากอยู่ ให้รอยบากนี้อยู่ทางซ้ายด้านบนดังแสดงในรูปการจัดวางอุปกรณ์ โดยใส่ไอซีให้รอยบากอยู่ทางด้านเดียวกัน ที่ตัวไอซีจะมีจุดวงกลมเล็กๆอยู่ จะแสดง ตำแหน่งขา 1 ของไอซีดังแสดงในรูป สำหรับการใส่ CONNECTOR ของ SERIAL PORT ทั้ง 2 CHANNEL นั้นให้หันด้านที่มีพลาสติกไว้ด้านใน ด้านที่เป็นขั้ว CONNECTOR จะหันออกด้านนอกแผ่น PRINNT





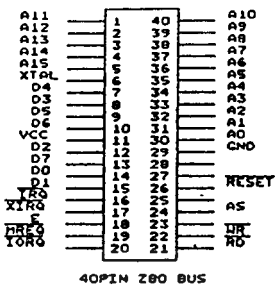


SOCKET 52 PIN (TOP VIEW)

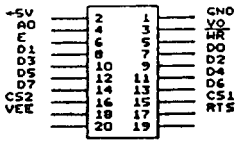
SPECIFICATION

CPU MC68HC11A1FN	52 PIN PLCC
MEMORY	6264/62256 (MAX 32K ADDRESS $2000-$9FFF)
	2764/27256 (MAX 24K ADDRESS $A000-$FFFF)
INTERNAL FEPROM 512 BYTE	
INTERNAL RAM 256 BYTE	
PORT	8255 I/O PORT 24 BIT
PRINTER PORT	1 PORT
KEYBOARD	1 PORT 3x4 OR 4x4
LCD	1 LCD MODULE DOT/GRAPHIC
TIMER	3 INPUT/4 OUTPUT AND 1 INPUT/OUTPUT
A/D 8 CHANNEL	
SERIAL PORT	2 CHANNEL SCI (RS232/RS422) & SPI
CLOCK RATE	8 MHz
POWER SUPPLY	INPUT 9V DC
CONNECTOR	
1 40-PIN EXPANSION HEADER-STRIP	(SYSTEM BUS)
1 34-PIN EXPANSION HEADER-STRIP	(PERIPHERAL)
1 20-PIN EXPANSION HEADER-STRIP	(PRINTER)
1 20-PIN EXPANSION HEADER-STRIP	(LCD)
1 14-PIN EXPANSION HEADER-STRIP	(A/D)
1 10-PIN EXPANSION HEADER-STRIP	(TIMER)
1 10-PIN EXPANSION HEADER-STRIP	(KEYBOARD)
1 6-PIN EXPANSION HEADER	(SPI)
1 4-PIN EXPANSION HEADER	(SCI)
1 8-PIN JUMPER	(RAM)
1 6-PIN JUMPER	(EPROM)
1 6-PIN JUMPER	(SS)
1 4-PIN JUMPER	(MODE)
1 4-PIN JUMPER	(VREF)
1 4-PIN JUMPER	(3x3/4x4)
1 3-PIN JUMPER	(2764/27256)
1 3-PIN JUMPER	(6264/62256)
1 3-PIN JUMPER	(CLCD/GLCD)
1 2-PIN JUMPER	(STEP)
1 2-PIN JUMPER	(XTAL)
LED	1 POWER RAD LED
	1 PC7 PORT 8255 ORANGE LED
PCB SIZE	9.3 x 14 cm.

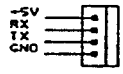
รายละเอียดของขา CONNECTOR ต่างๆ



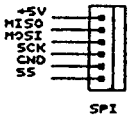
40PIN Z80 BUS



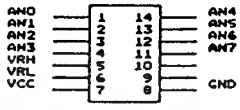
LCD



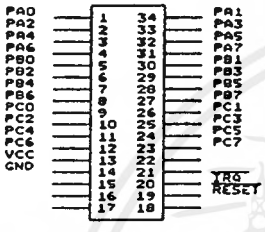
SCI



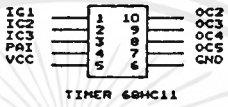
SPI



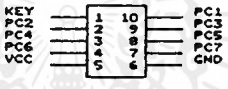
A/D 68HC11



34PIN 8255

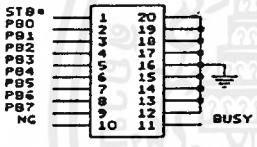


TIMER 68HC11



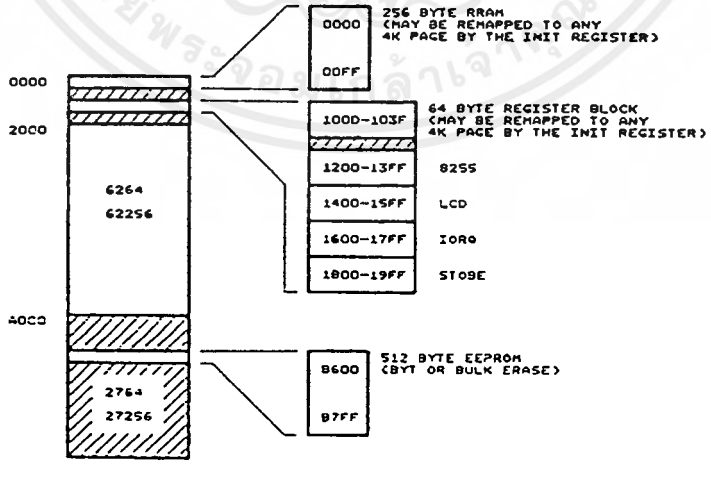
KEYBOARD 4X4 OR 3X4

8255



PRINTER PORT

8255



MEMORY MAP

รายการอุปกรณ์ของ CP-68HC11

อุปกรณ์สารกึ่งตัวนำ

1. CPU 68HC11A1	1 ตัว	2. RAM 6264	1 ตัว
3. 8255	1 ตัว	4. 74HCT373	1 ตัว
5. 74HCT138	1 ตัว	6. 74HCT156	1 ตัว
7. 74HCT04	1 ตัว	8. 74HC08	1 ตัว
9. MAX232	1 ตัว	10. XTAL 8MHz	1 ตัว
11. 1N4148	4 ตัว	12. 1N4001	1 ตัว
13. FR100	1 ตัว	14. ZENER 5.6V	1 ตัว
15. LED สีแดง	1 ตัว	16. LED สีเขียว	1 ตัว
17. 7805	1 ตัว	18. เฟอร์ไลค์	1 ตัว

ตัวเก็บประจุ

1. 470 uF 16V อิเล็กทรอไลต์	1 ตัว	2. 47 uF 16V อิเล็กทรอไลต์	1 ตัว
3. 33 uF 16V อิเล็กทรอไลต์	3 ตัว	4. 22 uF 16V อิเล็กทรอไลต์	1 ตัว
5. 10 uF 16V อิเล็กทรอไลต์	5 ตัว	6. 1uF 16V แทนทาลั่ม	1 ตัว
7. 0.1 uFแทนทาลั่ม	11 ตัว	8. 20 pF เซรามิก	2 ตัว

ตัวต้านทาน

1. 10 M 1/4 WATT	1 ตัว	2. 10 K 1/8 WATT	5 ตัว
3. 5.6 K 1/8 WATT	1 ตัว	4. 4.7 K 1/8 WATT	3 ตัว
5. 1 K	1/8 WATT 2 ตัว	6. 470 1/8 WATT	2 ตัว
7. R-PACK 10K 9 PIN	3 ตัว	8. R-PACK 10K 5 PIN	2 ตัว
9. VR 10K	1 ตัว		

SOCKET

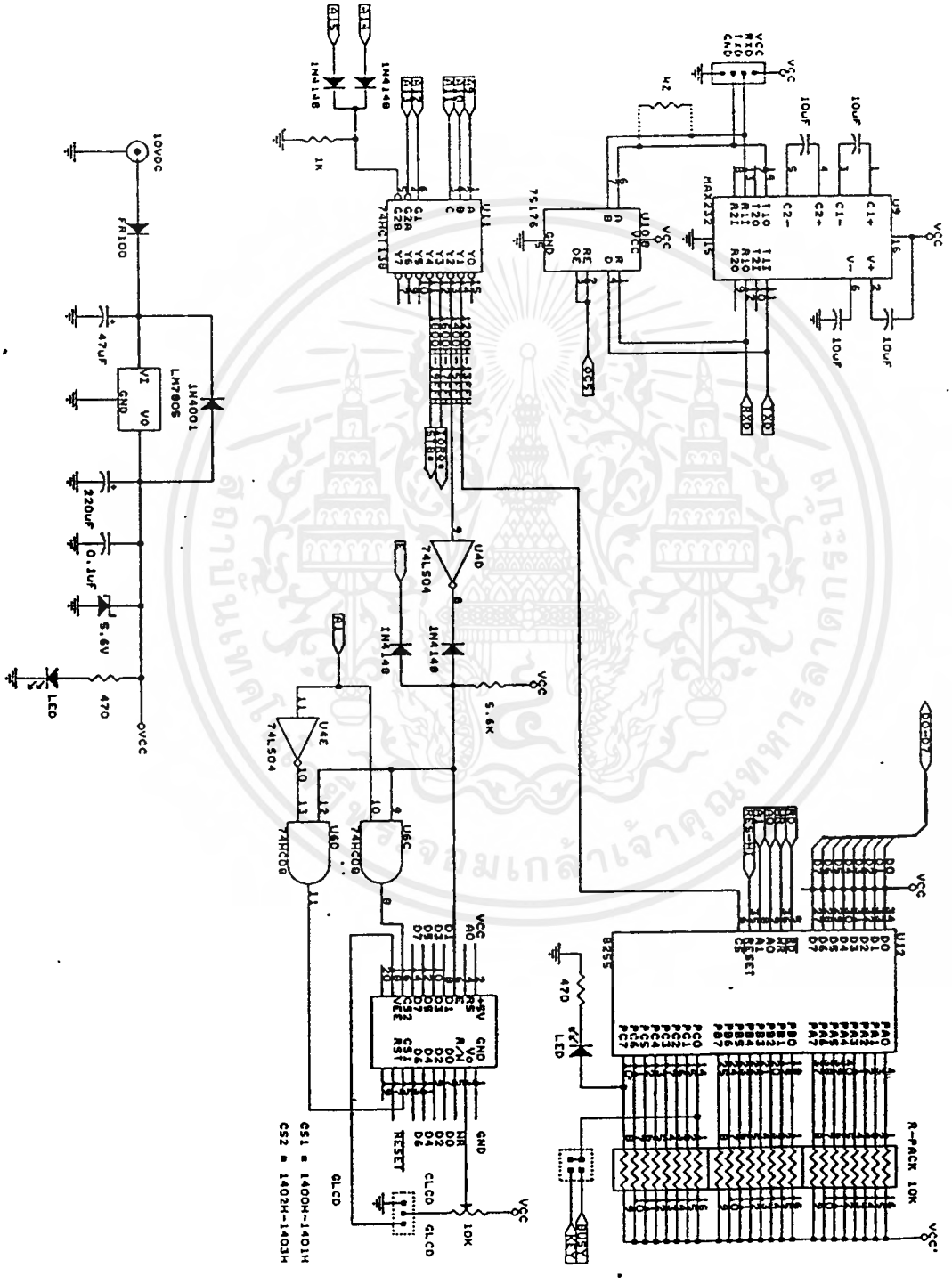
1. 52 PIN PLCC	1 ตัว	2. 40 PIN	1 ตัว
3. 28 PIN	2 ตัว	4. 20 PIN	1 ตัว
5. 16 PIN	2 ตัว	6. 14 PIN	1 ตัว

CONNECTOR

1. 40-PIN HEADER-STRIP	1 ตัว	2. 34-PIN HEADER-STRIP	1 ตัว
3. 20-PIN HEADER-STRIP	2 ตัว	4. 14-PIN HEADER-STRIP	1 ตัว
5. 10-PIN HEADER-STRIP	2 ตัว	6. 8-PIN JUMPER	1 ตัว
7. 6-PIN JUMPER	2 ตัว	8. 4-PIN JUMPER	3 ตัว
9. 3-PIN JUMPER	3 ตัว	10. 2-PIN JUMPER	2 ตัว
11. HEADER 6 PIN ตัวเล็ก	1 ตัว	12. HEADER 4 PIN ตัวเล็ก	1 ตัว

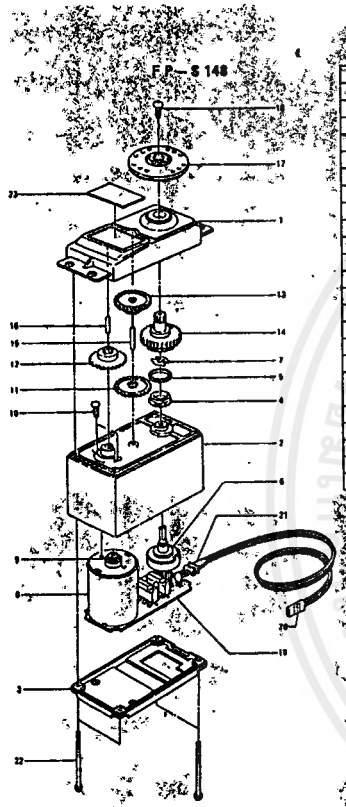
อื่นๆ

1. HEATSINK	1 อัน	2. SWITCH RESET	1 ตัว
3. JACK ADAPTOR	1 ตัว	4. MINI JUMPER	15 ตัว
5. แผ่น PCB	1 แผ่น		



ภาคผนวก ง.

รายละเอียดของ RC SERVO รุ่น FP S148



No.	Part name	Part No.
1.	Upper case	1144
2.	Middle case	1144
3.	Bottom case	1144
4.	Metal bearing	S04137
5.	Metal bearing	S04137
6.	Potentiometer	139668
7.	Potentiometer drive plate	S02753
8.	Motor	S91239
9.	Motor pinion	S02461
10.	Screw	J50002
11.	1st gear	1157
12.	2nd gear	1157
13.	3rd gear	1157
14.	Final gear	1157
15.	Intermediate shaft	S02495
16.	2nd shaft	S02494
17.	Servo horn D	1168
18.	Binding head tapping screw 2.6X8	1122
19.	Printed wiring board	AS1157
20.	3PB-WRB300G	AT2453
21.	w/gum bush	S90046
22.	Pan head truss screw	J50360
23.	Nameplate	S60099

FUTABA CORPORATION
Makuhari Techno Garden Bldg., B6F 1-3 Nakase, Mihama-ku, Chiba 261-01, Japan
Overseas Marketing & Sales Radio Control Systems
Phone:(043)296-5119 Facsimile:(043)296-5124

FUTABA CORPORATION OF AMERICA
4 Studebaker, Irvine California 92718, U.S.A.
Phone:714-455-9888 Telex:23-0691227 Facsimile:714-455-9899

The FP-S148 is a rugged low-profile servo for Futaba Corporation digital proportional radio control sets. Trouble caused by breaking of lead wires from shock and vibration have been eliminated by using new technology which eliminates wires inside the servo. Movement equal to that of high quality servos has been realized by using a new small, high-performance motor. Use them for channel expansion, or as replacement parts.

[FEATURES]

- The FP-S148 is a low 1.4 inches(36mm) high and has a thin design that can be easily mounted in all models.
- Vibration and shock resistance have been improved further by using a direct wiring system which directly connects the servo amp, motor and potentiometer.
- The height of the servo has been reduced and high torque, high speed, and smooth movement equal to that of the coreless servo have been realized by using a new small, high-performance motor. (Output torque 42 oz/in (3kg, cm), operating speed 0.22 sec/60°)
- New indirect drive/completely sealed potentiometer substantially improves vibration and shock resistance, and neutral accuracy.
- Unique Futaba power-saving custom IC provides high starting torque narrow dead band, and excellent trackability.
- Fiberglass PBT (polybutylene terephthalate) servo case is mechanically strong and is invulnerable to glow fuel.
- Strong polyacetyl resin precision servo gear featuring smooth operation, accurate neutral, and minimum backlash.
- Fiberglass epoxy PC board with THRU-THE-HOLE plating improves the servo amp vibration and shock resistance.
- Thick plated connector pins eliminate the problem of faulty contact, improve reliability against shock and vibration, and prevent reverse insertion.
- Special pad grommets simplify mounting of the servo, and are extremely vibration-resistant.
- Seven kind of special adjustable (splined) horns are available.
- High 42 oz-in (3kg, cm) output torque is perfect for almost all models

[RATINGS]

Control system ..... + pulse control 1520μs neutral
Operation angle ..... Rotary system, one side 45° or greater (including trim)
Power supply ..... 4.8V or 6.0V (shared with receiver)
Power consumption ..... 6.0V 8mA (at idle)
Output torque ..... 42 oz/in (3kg, cm)
Operating speed ..... 0.22 sec/60°
Dimensions ..... 1.59×0.77×1.4 inch (40.4×19.8×36m/m)
Weight ..... 1.5 oz (44.4 g)

[USAGE PRECAUTIONS]

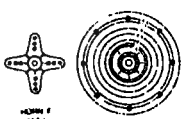
- Use the FP-S148 with a J, PCM, FGK-FM, CONQUEST-FM, MAGNUM, NEW ATTACK series other 2 ch.
- Futaba servos can not be used with other makes of RC sets.
- Select the servo mounting position and install the push rod, hinges, etc, so each control operates smoothly.
- Install the servos securely. Tighten the mounting screws until the rubber grommet is crushed slightly. If the screws are too tight, the cushioning effect will be adversely affected.
- Spare horns are provided. Use them according to the application.

(Specifications are subject to change without prior notice.)

SERVO S148

4 513886003004

540047
S148-003



MADE IN TAIWAN

บรรณานุกรม

ชัยวัฒน์ ลิ้มพรจิตรวิไล, *"เรียนรู้และใช้งานไมโครคอนโทรลเลอร์ 68HC11"*, บริษัทที่เอ็ดยูเคชั่น , 2538

"เทคนิคการควบคุมมอเตอร์." , 1998 , www.panmanee.com

"Robobug Assembly" , Florida , 1999 , www.mekatronic.com

"Robot Kits" , 1998 , www.lynxmotion.com

J. Billingsley , *"Robot and automated manufacture"* , Peter Peregrinus Ltd , 1990

L. Jones , *"Mobile Robots Inspiration to Implementation"* , A K Peters Wellesley Massachusetts , 1993