การควบคุมและแสดงผลแขนกล

Control and Display the Movement of the Robot Arm

โดย

นางสาว สุทธิจิตต์ ไสวแสนยากร

นาย สุภชัย จวนสาง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การควบคุมและแสดงผลแขนกล

Control and Display the Movement of the Robot Arm

โดย
นางสาว สุทธิจิตต์ ไสวแสนยากร
นาย สุภชัย จวนสาง

อาจารย์ที่ปรึกษา
ผศ.พลผดุง ผดุงกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิคส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาอิเล็กทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมและแสดงผลแขนกล

Control and Display the Movement of the Robot Arm

ผู้จัดทำ 1. นางสาว สุทธิจิตต์ ไสวแสนยากร 39014584

2. นาย สุภชัย จวนสาง 40010895

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ผศ.พลผดุง ผดุงกุล)

การควบคุมและแสดงผลแขนกล

นางสาว สุทธิจิตต์ ไสวแสนยากร 39014584
นาย สุภชัย จวนสาง 40010895
ผศ. พลผดุง ผดุงกุล (อาจารย์ที่ปรึกษา)
ภาคการเรียนที่ 2 ปีการศึกษา 2543

บทคัดย่อ

โครงงานนี้นำเสนอการควบคุมการทำงานของแขนกลโดยการใช้ไมโครคอนโทรลเลอร์ 8051 ผู้ใช้จะทำการสั่งงานผ่านทางคอมพิวเตอร์ด้วยโปรแกรมที่เขียนขึ้นจากภาษาวิชวลเบสิก (Visual Basic language) นอกจากนี้ยังสามารถระบุตำแหน่งของแขนกลแต่ละส่วนโดยการใช้โพเทนชิโอมิเตอร์ซึ่งค่าแรงดันของมันจะขึ้นกับตำแหน่งของแขนกล แล้วนำค่านี้ไปแปลงเป็นสัญญาณดิจิตอลด้วยวงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลก่อนส่งให้ไมโครคอนโทรลเลอร์ และซอฟแวร์บนคอมพิวเตอร์เพื่อนำไปสร้างภาพ 3 ภาพซึ่งแทนตำแหน่งของแขนกลในขณะนั้น

Control and Display the Movement of the Robot Arm

Miss Suthijitt Swaisaenyakorn 39014584
Mr. Supachai Jeungsang 40010895
Mr. Phonpadung Padungkul (Advisor)
2nd Semester, Education Year 2000

ABSTRACT

This project presents how to control the movement of the robot arm by using a microcontroller 8051. User will use the program on the computer, that is generated from a Visual Basic language to control the movement of the robot arm. The position of the robot arm is monitored by using a potentiometer which its voltage is depended on the position of the robot arm. Then, the voltage will be sent to convert into a digital signal by an Analog to Digital Converter circuit. This signal will be sent to a microcontroller. Then to the software on a computer is used to generate 3 pictures. These 3 pictures will represent the recent position of the robot arm.

สารบัญ

สารบัญภาพ

หน้า

บทนำ

เนื่องจากในปัจจุบันเทคโนโลยีต่างๆได้ถูกพัฒนาไปอย่างรวดเร็ว การทำงานต่างๆโดยเฉพาะทางด้านอุตสาหกรรมมักจะนำเครื่องจักรกลมาใช้แทนการใช้แรงงานคน เครื่องจักรกลต่างๆจึงเข้ามามีบทบาทต่อโรงงานอุตสาหกรรมอย่างมาก การเลือกใช้เครื่องจักรกลแทนการใช้แรงงานคนนั้นมีข้อดีหลายอย่างคือ ความแม่นยำและความน่าเชื่อถือในการทำงาน และเพื่อใช้ในการทำงานที่เป็นอันตรายต่อมนุษย์

แขนกลถือเป็นเครื่องจักรอย่างหนึ่งที่นิยมใช้กันมากในปัจจุบัน โครงงานชิ้นนี้เป็นการสร้างแขนกลขึ้นเพื่อศึกษาการควบคุมการทำงานของแขนกลผ่านทางคอมพิวเตอร์ และการสร้างภาพ 3 มิติแสดงการทำงานของแขนกล โดยภาพที่สร้างขึ้นนี้ได้จากการใช้เซนเซอร์จับสัญญาณการเคลื่อนไหวของแขนกลแต่ละส่วนแล้วนำมาผ่านสมการทางคณิตศาสตร์เพื่อให้ได้เป็นภาพ 3 มิติออกมา

บทที่ 2 กล่าวถึงทฤษฎีของส่วนประกอบต่างๆที่นำมาใช้ในโครงงานอันได้แก่ สเต็ปปิ้งมอเตอร์และวงจรขับสเต็ปปิ้งมอเตอร์ มอเตอร์กระแสตรงและวงจรขับมอเตอร์กระแสตรง ส่วนของการควบคุมและการแสดงตำแหน่งของแขนกลซึ่งเป็นการทำงานร่วมกันระหว่างซอฟแวร์ที่เขียนขึ้นด้วยภาษาวิชวลเบสิก ไมโครคอนโทรลเลอร์ วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลและโพเทนชิโอมิเตอร์ ในบทนี้จะขอกล่าวถึงรายละเอียดโครงสร้างของไมโครคอนโทรลเลอร์เนื่องจากเป็นหัวใจสำคัญในการควบคุมการเคลื่อนที่ของแขนกลและการแสดงตำแหน่งเพื่อให้ผู้ใช้มีความเข้าใจในหลักการทำงานของดครงงานได้ดียิ่งขึ้น

บทที่ 3 กล่าวถึงการออกแบบและคำนวณค่าอุปกรณ์ต่างๆที่นำมาใช้ในโครงงาน พร้อมทั้งเหตุผลในการเลือกส่วนประกอบต่างๆที่นำมาใช้ในโครงงาน โดยการแสดงด้วยสมการทางคณิตศาสตร์สำหรับค่าอุปกรณ์อิเล็กทรอนิคส์ที่นำมาใช้ ซึ่งได้มาจากการอ้างอิงตามทฤษฎีในบทที่ 2 ที่นำมาจากหนังสืออ้างอิงตามที่ได้แสดงไว้ในบรรณานุกรม ในส่วนของการควบคุมและการแสดงตำแหน่งของแขนกลได้อธิบายแยกรายละเอียดการทำงานออกเป็น 3 ส่วนคือ ส่วนของโปรแกรมซอฟแวร์ในที่ได้กล่าวถึงขั้นตอนการทำงานของโปรแกรมโดยละเอียด รวมถึงการอ้างอิงแกนตำแหน่งในการเคลื่อนที่จริงเทียบกับภาพที่แสดงให้ผู้ใช้ทราบ และได้แสดงถึงการคำนวณที่นำมาใช้ในการสร้างภาพด้วย ส่วนที่ 2 คือไมโครคอนโทรลเลอร์ในที่นี้ก็ได้อธิบายการทำงานไว้ทุกขั้นตอนตามที่ได้เขียนโปรแกรม และส่วนสุดท้ายคือส่วนของวงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลซึ่งได้รับการควบคุมการทำงานจากไมโครคอนโทรลเลอร์

บทที่4 เป็นการแสดงผลการทดลองการทำงานของส่วนต่างๆของโครงงาน รวมถึงการทำงานเชื่อมต่อกับส่วนอื่นๆ ตั้งแต่มอเตอร์ วงจรขับมอเตอร์ ไมโครคอนโทรลเลอร์ วงจรแปลง

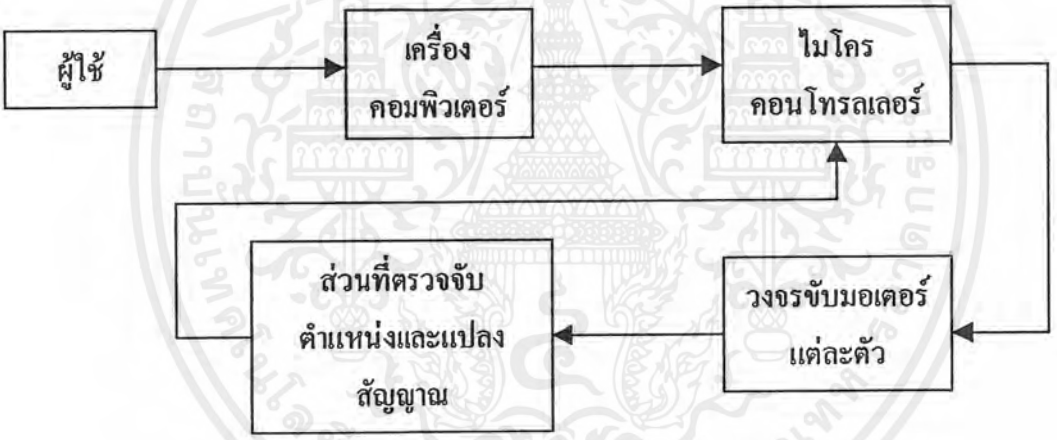
สัญญาณอะนาลอกเป็นสัญญาณดิจิตอล โพเทนชิโอมิเตอร์ และซอฟแวร์ที่เขียนด้วยภาษาวิชวลเบสิกโดยรวม การทดลองการทำงานในโครงงานนี้ทำโดยการเปรียบเทียบตำแหน่งที่แขนกลเคลื่อนที่ไปได้จริงกับภาพฉายบนระนาบ X-Y , Z-Y และภาพแสดงมุมที่มอเตอร์กระแสตรงตัวล่างหมุนรอบแกน Y โดยวัดเทียบแกน X ไปตามทิศทางตามเข็มนาฬิกา

บทที่ 5 คือการสรุปผลที่ได้รับจากการทำโครงงานในครั้งนี้ และปัญหาที่เกิดขึ้นในระหว่างการทำโครงงาน ตลอดจนแนวทางการแก้ไขหรือปรับปรุงเพื่อแก้ไขปัญหาที่เกิดขึ้นหรือพยายามลดให้ปัญหาเหลือน้อยที่สุด

ภาคผนวก เป็นส่วนที่ประกอบด้วยโปรแกรมที่ขึ้นด้วยภาษาวิชวลเบสิก รวมทั้งได้แสดงหน้าจอต่างๆ ที่ใช้ในการควบคุมการเคลื่อนที่ของแขนกล และโปรแกรมที่เขียนขึ้นมาด้วยภาษาแอสเซมบลี้เพื่อใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ รายละเอียดของไมโครคอนโทรลเลอร์เบอร์ MS 89C51 รายละเอียดของไอซีที่ใช้ในการแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลคือไอซีเบอร์ AD0808

บทที่ 2
ทฤษฎีพื้นฐานและหลักการทำงานของส่วนต่างๆของโครงงาน

โครงงานการแสดงผลและควบคุมแขนกลนี้เป็นโครงงานที่มีการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ โดยคอมพิวเตอร์จะทำหน้าที่ติดต่อกับผู้ใช้และส่งสัญญาณการควบคุมแขนกลตามที่ผู้ใช้ต้องการออกไปให้กับไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์จะมีหน้าที่ควบคุมวงจรขับมอเตอร์เพื่อให้แขนกลเคลื่อนที่ตามที่ผู้ใช้ต้องการ นอกจากนี้ยังมีหน้าที่นำสัญญาณค่าตำแหน่งที่ได้จากวงจรแปลงสัญญาณอะนาลอกไปเป็นสัญญาณดิจิตอล (Analog to Digital Converter) แล้วส่งค่าไปยังคอมพิวเตอร์เพื่อแสดงค่าตำแหน่งของแขนกลให้ผู้ใช้ทราบการทำงานทั้งหมดนี้สามารถแสดงได้ด้วยบล็อกไดอะแกรมดังรูปที่ 2.1



รูปที่ 2.1 บล็อกไดอะแกรมอธิบายการทำงานของแขนกล

ในบทนี้จะขอกล่าวถึงรายละเอียดของส่วนประกอบต่างๆที่ใช้ในโครงงานอันได้แก่ สเต็ปปิ้งมอเตอร์และมอเตอร์กระแสตรงพร้อมทั้งวงจรขับมอเตอร์แต่ละแบบ และส่วนที่ใช้ในการระบุตำแหน่งของแขนกล

2.1 สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นเครื่องจักรกลไฟฟ้าประเภทหนึ่งซึ่งมีคุณสมบัติพิเศษบางอย่างคือ มีการหมุนเป็นแบบสเต็ปตามจังหวะของสัญญาณไฟฟ้าที่ป้อนให้ ลักษณะเช่นนี้จึงทำให้สามารถควบคุมจำนวนสเต็ปของการหมุนได้ด้วยการควบคุมการป้อนสัญญาไฟฟ้า หรือสามารถควบคุมการหมุนแบบลูปเปิด (OPEN LOOP) ได้ และถ้านำสเต็ปปิ้งมอเตอร์ไปต่อใช้งานกับเครื่องมือ

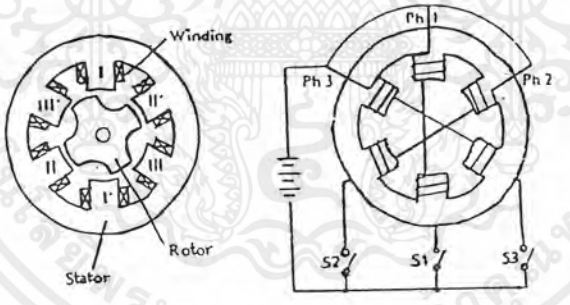
บางประเภท เช่น พล็อตเตอร์ เครื่องเจาะอัตโนมัติ ก็จะทำให้การควบคุมตำแหน่งเป็นไปตามที่ต้องการได้อย่างแม่นยำ

2.1.1 ชนิดของสเต็ปปิ้งมอเตอร์

สามารถแบ่งตามลักษณะโครงสร้างและการใช้งานได้ดังนี้

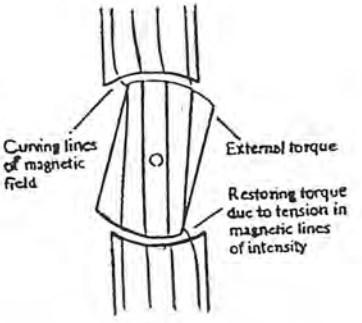
สเต็ปปิ้งมอเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ (VARIABLE RELUCTANCE STEPPING MOTOR

สเต็ปปิ้งมอเตอร์ชนิดนี้สามารถปรับค่ารีลักแตนซ์ได้ รูปที่ 2.2 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส โดยที่สเตเตอร์มีฟันทั้งหมด 6 ซี่ โดยซี่ที่ทำมุม 180 องศาจะเป็นเฟสเดียวกัน ขดลวดที่พันอยู่ที่ฟันของสเตเตอร์ในแต่ละเฟสจะต่ออนุกรมหรือขนานกันก็ได้ จากรูปที่ 2.2 เป็นการต่อแบบอนุกรม ส่วนโรเตอร์นั้นมีฟัน 4 ซี่ ทั้งโรเตอร์และสเตเตอร์ทำมาจากโลหะซิลิกอน ซึ่งมีสภาพซึมซับทางแม่เหล็กสูงและยอมให้สนามแม่เหล็กจำนวนมากไหลผ่านได้ ฟันของสเตเตอร์ในเฟสเดียวกันจะมีขั้วต่างกันหลังจากถูกกระตุ้น



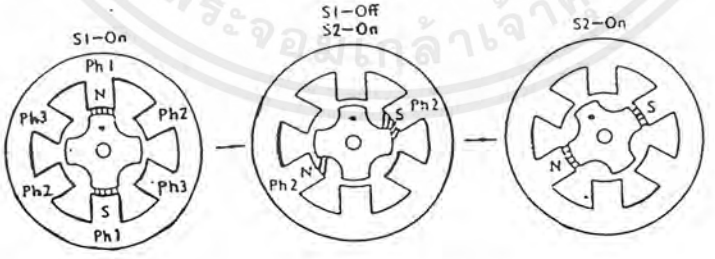
รูปที่ 2.2 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส

รูปที่ 2.3 แสดงตำแหน่งสมดุลย์เมื่อเฟสใดเฟสหนึ่งถูกกระตุ้น



รูปที่ 2.4 แสดงแรงภายนอกที่มีผลต่อเส้นแรงแม่เหล็ก

กระแสที่ไหลในแต่ละเฟสถูกควบคุมโดยสวิทซ์ปิด/เปิด ถ้าเฟส I ถูกกระตุ้นจะมีกระแสไหลและเกิดฟลักซ์แม่เหล็กดังแสดงในรูปที่ 2.3 แกนโรเตอร์จะอยู่ตำแหน่งเดียวกับซี่ I และ I' ทำให้ทั้งโรเตอร์และสเตเตอร์อยู่ในแนวเดียวกัน กรณีนี้จะทำให้ค่ารีลักแตนซ์มีค่าน้อยที่สุดซึ่งเป็นตำแหน่งที่สมดุลย์ ถ้าโรเตอร์ถูกกระทำจากแรงภายนอกจะทำให้เปลี่ยนตำแหน่งดังรูปที่ 2.4 แรงบิดกระทำกับโรเตอร์ในทิศตามเข็มนาฬิกาทำให้ตำแหน่งเปลี่ยนไป มีผลทำให้เส้นแรงแม่เหล็กเคลื่อนที่จากซี่ของโรเตอร์และสเตเตอร์ เมื่อโรเตอร์และสเตเตอร์ไม่ได้อยู่ในแนวเดียวกันแล้วค่ารีลักแตนซ์จะมีค่ามาก จากนั้นสเต็ปปิ้งมอเตอร์จะทำให้มีค่ารีลักแตนซ์น้อยที่สุด พอเฟส II ถูกกระตุ้นดังรูปที่ 2.5 โรเตอร์ถูกแรงภายนอกกระทำให้เคลื่อนไป 30 องศาในทิศทวนเข็มนาฬิกา จากนั้นก็จะย้ายจากมุมที่เกิดการกระตุ้นกลับไปยังตำแหน่งที่ค่ารีลักแตนซ์น้อยที่สุด การย้ายจากมุมที่เกิดการกระตุ้นแต่ละครั้งให้กลับไปยังตำแหน่งเดิมเรียกว่า สเต็ป



รูปที่ 2.5 แสดงขั้นตอนการเคลื่อนที่ของโรเตอร์เมื่อสเต็ปปิ้งมอเตอร์ถูกกระตุ้น

คุณสมบัติพื้นฐานของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์

1. ช่องว่างระหว่างฟันของโรเตอร์และสเตเตอร์ต้องเล็กที่สุดเท่าที่จะเป็นไปได้ เพื่อให้ทอร์คที่เกิดขึ้นมีค่ามากและมีความแม่นยำทางตำแหน่งมากขึ้น

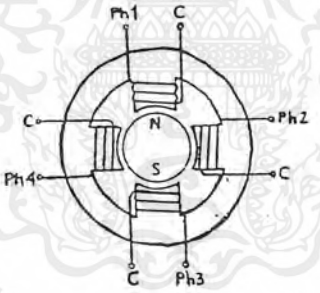
2. จะต้องมีมุมของการสเต็ปเล็กที่สุดเท่าที่จะเป็นไปได้ ซึ่งทำได้โดยการเพิ่มจำนวนขั้วของโรเตอร์และสเตเตอร์ ความสัมพันธ์ของมุมของการสเต็ป θ_s , มุมเฟส m , จำนวนซี่ฟันของโรเตอร์ Nr, จำนวนสเต็ป S แสดงดังสมการ

$$S = 360/\theta_s = mNr$$

3. การสร้างสเต็ปปิ้งมอเตอร์ให้มีโครงสร้างหลายสเต็ปเพื่อเพิ่มประสิทธิภาพ โครงสร้างของสเต็ปปิ้งมอเตอร์แบบนี้จะมี 1 เฟส โดยที่โรเตอร์และสเตเตอร์มีซี่ฟันเหมือนกัน ซึ่งจะเป็นการเพิ่มประสิทธิภาพในด้านทอร์คต่อหน่วยปริมาตรของโรเตอร์

สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

สเต็ปปิ้งมอเตอร์ชนิดนี้ใช้แม่เหล็กถาวรเป็นโรเตอร์และมีซี่ฟันของสเตเตอร์ล้อมรอบ ซี่ฟันของสเตเตอร์ถูกพันด้วยขดลวดสำหรับสร้างสนามแม่เหล็ก เมื่อต้องการให้สเต็ปปิ้งมอเตอร์มีขนาดมุมของสเต็ปเล็กลงจะต้องเพิ่มขั้วแม่เหล็กของโรเตอร์และจำนวนซี่ฟันของสเตเตอร์ แต่ก็มีขีดจำกัดในการเพิ่มจำนวนขั้วแม่เหล็กของโรเตอร์ เนื่องจากการสร้างแม่เหล็กถาวรสร้างโดยมีขั้วหลายขั้วทำได้ยาก

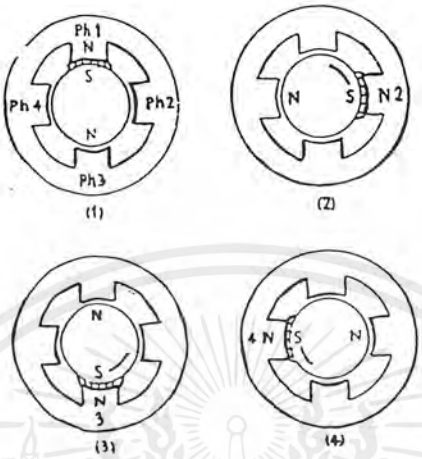


รูปที่ 2.6 แสดงโครงสร้างของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

ตัวอย่างการทำงานสมมติว่าสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรขนาด 4 เฟส มีโรเตอร์เป็นแม่เหล็กถาวรทรงกระบอกและสเตเตอร์มี 4 ซี่ฟันซึ่งรอบๆพันด้วยขดลวด มีรูปแบบของการทำงานคือ เมื่อสร้างสัญญาณกระตุ้นตามลำดับเฟส โรเตอร์จะหมุนไปตามทิศทางของการกระตุ้นดังแสดงในรูปที่ 2.7

ข้อเสียคือ มีขนาดมุมสเต็ปใหญ่ทำให้ความละเอียดของสเต็ปต่อรอบน้อย จากโครงสร้างของโรเตอร์ซึ่งเป็นแม่เหล็กถาวร การให้มีขั้วหลายขั้วทำได้ยาก ไม่สามารถสร้างสเต็ปขนาดเล็กได้ สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรส่วนใหญ่จะมีโครงสร้างขนาดเล็ก ทำให้ค่าทอร์ค

ที่ได้ต่อหน่วยปริมาตรต่ำ ถ้าต้องการปรับปรุงประสิทธิภาพในเรื่องของทอร์ค แม่เหล็กถาวรที่ใช้ต้องทำจากสารแม่เหล็กที่มีสภาพความเป็นแม่เหล็กสูง



รูปที่ 2.7 แสดงการทำงานของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรขนาด 4 เฟส

สเต็ปปิ้งมอเตอร์แบบไฮบริดจ์

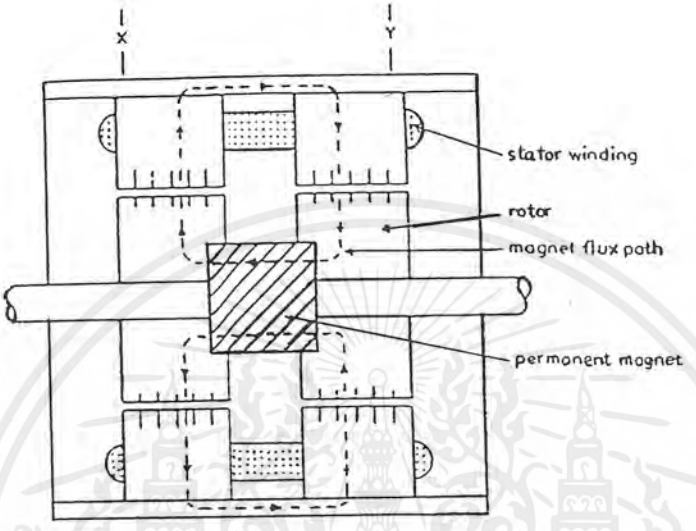
สเต็ปปิ้งมอเตอร์ชนิดนี้มีแกนโรเตอร์เป็นแม่เหล็กถาวร โดยมีการทำงานร่วมกันของมอเตอร์แบบแม่เหล็กถาวรและมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ได้ โดยมีโครงสร้างของสเตเตอร์คล้ายกับโครงสร้างของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ แต่ต่างกันที่การต่อขดลวดโดยที่แต่ละเฟสของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์จะมีขดลวด 2 ขดแต่ละขดมีขั้วต่างกัน แต่ไฮบริดจ์สเต็ปปิ้งมอเตอร์ขดลวดทั้ง 2 จะพันอยู่ที่ขั้วเดียวกันเรียกว่าไบโพลาร์ (BIPOLAR) ซึ่งในการกระตุ้นแต่ละครั้งจะให้ขั้วที่แตกต่างกัน

คุณสมบัติที่สำคัญของไฮบริดจ์สเต็ปปิ้งมอเตอร์

โครงสร้างของมอเตอร์จะมีแม่เหล็กถาวรอยู่ตรงกลางระหว่างเฟสทั้งสอง การเหนี่ยวนำสนามแม่เหล็กทำได้โดยใช้สนามแม่เหล็กที่สร้างจากสเตเตอร์ซึ่งเป็นสนามแม่เหล็กแบบเฮทเทอโรโพลาร์ (HETEROPOLAR FIELD) ดังนั้นทอร์คเกิดจากการทำงานร่วมกันของสนามแม่เหล็ก 2 ชนิด คือ สนามแม่เหล็กจากแม่เหล็กถาวรและสนามแม่เหล็กเหนี่ยวนำที่เกิดจากการกระตุ้นของขดลวดแต่ละขด โครงสร้างของซี่ฟันของสเตเตอร์จะใหญ่กว่าซี่ฟันของโรเตอร์เล็กน้อยเพื่อเพิ่มความถูกต้องแม่นยำทางตำแหน่งของการเคลื่อนที่

หลักการทำงานของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่แตกต่างจากสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์คือ แรงบิดที่เกิดจากสนามแม่เหล็กจะไม่ขึ้นอยู่กับกระแสที่ไหลผ่านขดลวด

เพียงอย่างเดียวแต่ขึ้นอยู่กับโครงสร้างของซี่ฟันด้วย ซึ่งถูกออกแบบเพื่อให้ได้โครงสร้างเล็กและใช้แม่เหล็กถาวรเป็นแกนกลางเพื่อลดผลของการออสซิลเลททางแมคคานิคส์



รูปที่ 2.8 แสดงโครงสร้างของไฮบริดจ์สเต็ปปิ้งมอเตอร์

ข้อดีของไฮบริดจ์สเต็ปปิ้งมอเตอร์คือ สเต็ปมีขนาดเล็ก มีความละเอียดของสเต็ปต่อรอบสูง มีค่าทอร์คสูงกว่าสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ แต่สเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์มีแรงเฉื่อยทางแมคคานิคส์น้อยกว่าไฮบริดจ์สเต็ปปิ้งมอเตอร์

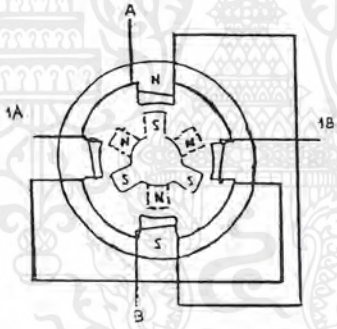
นอกจากสเต็ปปิ้งมอเตอร์ทั้ง 3 ชนิดที่กล่าวมาแล้วยังมีสเต็ปปิ้งมอเตอร์ชนิดอื่นๆ ที่ไม่ได้กล่าวถึงอีกเช่น ลิเนียร์สเต็ปปิ้งมอเตอร์ซึ่งเป็นมอเตอร์ที่ได้รับการออกแบบให้มีการเคลื่อนที่แบบเป็นเชิงเส้น อิเล็คโตรคไฮดรอลิคสเต็ปปิ้งมอเตอร์ซึ่งเป็นสเต็ปปิ้งมอเตอร์กำลังสูงใช้ในงานอุตสาหกรรม เป็นต้น

2.1.2 หลักการทำงานของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์สามารถแบ่งโครงสร้างทางกายภาพออกได้เป็น 2 ส่วนคือ สเตเตอร์ (STATOR) และโรเตอร์ (ROTOR) ตัวสเตเตอร์เป็นส่วนที่อยู่กับที่ประกอบด้วยขดลวดทองแดงซึ่งพันอยู่รอบแกนเหล็ก เพื่อสร้างสนามแม่เหล็กเมื่อมีการจ่ายกระแสผ่านขดลวด ส่วนโรเตอร์เป็นส่วนที่เคลื่อนที่มีลักษณะเป็นแท่งเหล็กทรงกลม และที่ผิวรอบนอกมีลักษณะเป็นซี่ฟัน ซึ่งทำจากแม่เหล็กถาวรดังรูปที่ 2.9 การที่มีจำนวนสเต็ปมากๆนี้ไม่ได้เพิ่มที่จำนวนขั้วแม่เหล็ก

ไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยเพิ่มจำนวนขั้วแม่เหล็กที่โรเตอร์ จำนวนสเต็ปต่อรอบทั้งหมดขึ้นอยู่กับผลคูณระหว่างจำนวนขั้วของสเตเตอร์และโรเตอร์

เมื่อยังไม่จ่ายกระแสให้กับขดลวดของมอเตอร์ซี่ฟันอันใดอันหนึ่งของโรเตอร์จะอยู่ในตำแหน่งที่ตรงกันกับซี่ฟันอันใดอันหนึ่งของสเตเตอร์ ทั้งนี้เป็นเพราะแม่เหล็กถาวรที่ตัวของโรเตอร์พยายามที่จะทำให้ค่าความต้านทานทางแม่เหล็กไฟฟ้า(RELUCTANCE) มีค่าน้อยที่สุด ซึ่งจุดที่ซี่ฟันของตัวโรเตอร์และสเตเตอร์ตรงกันนั้นมีค่าความต้านทานทางแม่เหล็กไฟฟ้าน้อยที่สุด ทำให้เกิดเส้นแรงแม่เหล็กไฟฟ้ามากที่สุด เส้นแรงแม่เหล็กไฟฟ้าจะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ขึ้นมา 2 คู่ทั้งที่ตัวสเตเตอร์และตัวโรเตอร์ดังรูป ค่าทอร์ค (TORQUE) ที่ทำให้ตัวโรเตอร์สามารถยึดอยู่ในตำแหน่งดังกล่าวนี้เรียกว่า ดีเท็นท์ทอร์ค (DETENT TORQUE) หมายความว่าการที่จะทำให้มอเตอร์เคลื่อนที่ในขณะที่ไม่ได้จ่ายกระแสให้กับขดลวดของมอเตอร์จะต้องออกแรงมากกว่าค่าของดีเท็นท์ทอร์คจึงจะทำให้โรเตอร์เคลื่อนที่ได้รูปที่ 2.9 นั้นมี 12 ตำแหน่งที่สามารถเกิดดีเท็นท์ทอร์คได้

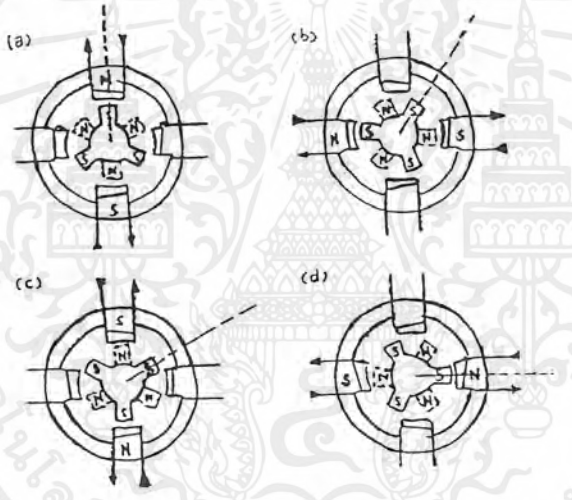


รูปที่ 2.9 แสดงโครงสร้างของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่มีจำนวนสเต็ปต่อรอบเท่ากับ 12

เมื่อจ่ายกระแสให้กับขดลวดที่อยู่ในสเตเตอร์คู่ใดคู่หนึ่งดังรูปที่ 2.10a จะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ที่ซี่ฟันของตัวสเตเตอร์ ซึ่งจะดึงดูดซี่ฟันของตัวโรเตอร์ที่มีขั้วแม่เหล็กที่มีศักย์ต่างกันที่อยู่ใกล้ที่สุดเข้าไว้ตำแหน่งนี้เรียกว่า สเตเบิ้ลโพสซิชั่น (STABLE POSITION) ของโรเตอร์จะมีจำนวนตำแหน่งเท่ากับจำนวนซี่ฟันของโรเตอร์ และแรงที่จะทำให้โรเตอร์เปลี่ยนตำแหน่งไปจากตำแหน่งสเตเบิ้ลโพสซิชั่นได้นี้เรียกว่า โฮลดิ้งทอร์ค (HOLDING TORQUE)

เมื่อสับเปลี่ยนการจ่ายกระแสให้แก่ขดลวดจากขดหนึ่งไปยังอีกขดหนึ่ง เนื่องจากขดลวดวางอยู่ในตำแหน่งที่ต่างกัน 90 องศา จะทำให้ตัวสเตเตอร์ดึงดูดซี่ฟันของตัวโรเตอร์อีก

ซี่หนึ่งที่ใกล้ที่สุดเข้าไว้ ซึ่งทำให้ตัวโรเตอร์เคลื่อนที่ไป 1 สเต็ปหรือ 30 องศาดังรูปที่ 2.10b จากนั้นก็เปลี่ยนไปจ่ายกระแสให้กับขดลวดชุดแรกโดยในครั้งนี้เปลี่ยนทิศทางการไหลของกระแสให้ตรงกันข้ามกับครั้งแรก จะทำให้ตัวโรเตอร์เคลื่อนที่ไปอีก 1 สเต็ป (หรือ 30 องศา) ดังรูปที่ 2.10c จากนั้นจ่ายกระแสให้กับขดลวดชุดที่สองโดยกลับทิศทางของกระแสที่ป้อนให้อีกเช่นกันทำให้โรเตอร์หมุนไป 90 องศาจากตำแหน่งเริ่มต้นดังรูปที่ 2.10d และหากป้อนกระแสให้กับมอเตอร์เหมือนที่เราป้อนในครั้งแรกแล้วซี่ฟันซี่ถัดไปของตัวโรเตอร์จะอยู่ในตำแหน่งที่เหมือนกับในรูปที่ 2.10a อีกครั้งหนึ่ง ถ้าหากต้องการเคลื่อนที่หนึ่งรอบต้องทำการกระตุ้นให้มอเตอร์เคลื่อนที่ไปจนครบ 12 สเต็ป และถ้าต้องการให้โรเตอร์หมุนไปอีกทิศทางหนึ่งต้องทำการสลับลำดับในการจ่ายกระแสจากรูปที่ 2.10a, 2.10d, 2.10c, 2.10b ตามลำดับในที่นี้เป็นการกระตุ้นแบบทีละเฟส



รูปที่ 2.10 แสดงขั้นตอนการทำงานของสเต็ปปิ้งมอเตอร์แบบ เต็มสเต็ปหนึ่งเฟส

การกระตุ้นและควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละเสต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องแบ่งออกเป็น 3 รูปแบบคือ แบบเวฟ, แบบ 2 เฟส และแบบครึ่งเสต็ป ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไปแบบเวฟเป็นแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขด ณ เวลาหนึ่งเรียงถัดไป ดังเช่น ขดที่ 1, 2, 3, 4, 1 หรือ 1, 4, 3, 2, 1 ขึ้นอยู่กับทิศทางที่ต้องการหมุน

ดังนั้นจึงมีขดลวดเพียงขดเดียวที่ถูกกระตุ้นเท่านั้นในแต่ละสเต็ป วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ ในเวลาแสดงดังตารางที่ 2.1

ตารางที่ 2.1 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบเวฟ

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

แบบที่ 2 เฟส เป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟ แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไปเช่นเดียวกับแบบเวฟแต่เป็นคู่คือ ขดลวดที่ถูกกระตุ้น 12, 23, 34, 41, 12 หรือ 14, 43, 32, 21, 14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้แรงบิดเพิ่มขึ้น โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกันและต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียก็คือการกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆแสดงดังตารางที่ 2.2

ตารางที่ 2.2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

แบบครึ่งสเต็ปเป็นรูปแบบที่เกิดขึ้นจากการผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปตามลำดับดังนี้ ขดลวดที่ถูกกระตุ้น 1 , 12, 2, 23, 3, 34, 4, 41, 1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1, 14, 4, 43, 3, 32, 2, 21, 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีกเพราะช่วงสเต็ปมีระยะสั้นลงและแต่ละสเต็ปเกิดจากแรงดึงของขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งจะมีมากขึ้น แต่ต้องระวังเมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุน 2 สเต็ปจึงจะได้เท่ากับ 1 สเต็ปเต็มเหมือนกับการควบคุมใน 2 แบบแรก

สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้เทียบเท่ากับแบบ 2 เฟสจึงจะเพียงพอ ขั้นตอนการทำงานต่างๆ แสดงได้ดังตารางที่ 2.3

ตารางที่ 2.3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเต็ป

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

2.1.3 วงจรขับสเต็ปปิ้งมอเตอร์ (DRIVE CIRCUITS)

สัญญาณที่ใช้สำหรับควบคุมการทำงานของสเต็ปปิ้งมอเตอร์มักจะเป็นสัญญาณที่สร้างจากวงจรดิจิตอล เช่น จากไมโครคอนโทรลเลอร์ซึ่งเป็นอุปกรณ์จำพวก TTL แรงดันที่ใช้มีค่าเท่ากับ 5 โวลต์และสามารถจ่ายกระแสได้ไม่มาก แต่เนื่องจากการทำงานของสเต็ปปิ้งมอเตอร์ต้องการแรงดันและกระแสที่สูงกว่านั้น ดังนั้นจึงจำเป็นที่จะต้องมีวงจรขับเพื่อทำหน้าที่จ่ายแรงดันและกระแสที่เพียงพอให้กับตัวสเต็ปปิ้งมอเตอร์ โดยทั่วๆไปวงจรขับมักจะสร้างจากไบโพลาร์ทรานซิสเตอร์ที่นำมาต่อใช้งานเป็นสวิตช์ ลักษณะของวงจรขับขึ้นอยู่กับชนิดของสเต็ปปิ้งมอเตอร์ที่ใช้ เช่น ถ้าใช้สเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ซึ่งจะมีอย่างน้อยสามเฟส วงจรขับที่ใช้จะทำงานในลักษณะของสวิตช์ ปิด/เปิดให้กระแสไหลผ่านไปยังขดลวดในทิศทางเดียวเราเรียกวงจรขับแบบนี้ว่า ยูนิโพลาร์ (การจ่ายกระแสเพียงทิศทางเดียว) แต่ถ้าใช้สเต็ปปิ้งมอเตอร์แบบไฮบริดจ์หรือแบบแม่เหล็กถาวรซึ่งมักจะมีเพียงสองเฟสจะต้องใช้วงจรขับที่สามารถจ่ายกระแสได้สองทิศทางเรียกวงจรขับประเภทนี้ว่า ไบโพลาร์ ซึ่งประกอบด้วย ทรานซิสเตอร์หลายตัวต่อเป็นวงจรแบบบริดจ์ ในที่นี้จะขอกล่าวถึงเฉพาะวงจรขับแบบยูนิโพลาร์ซึ่งเป็นแบบที่นำมาใช้ในโครงงาน

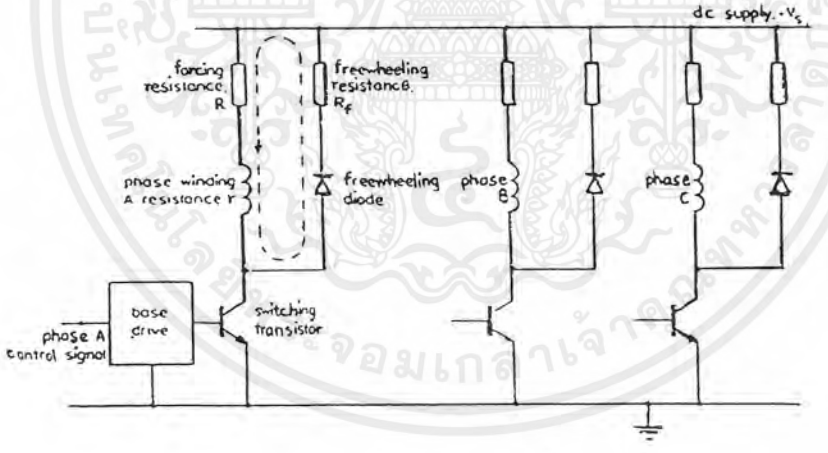
วงจรขับแบบยูนิโพลาร์ (UNIPOLAR DRIVE CIRCUIT)

เป็นวงจรขับที่เหมาะสำหรับสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์ ขดลวดแต่ละเฟสจะถูกกระตุ้นโดยวงจรขับแต่ละชุด ซึ่งวงจรขับแต่ละชุดก็จะได้รับสัญญาณควบคุมการทำงานจากไมโครคอนโทรลเลอร์ กระแสจะไหลผ่านขดลวดแต่ละเฟส เมื่อสวิตช์ซึ่งเลือกใช้เป็นทรานซิสเตอร์ที่อยู่ในสภาวะอิ่มตัวเนื่องจากกระแสที่ไปไบอัสทางด้านเบส ในสภาวะนี้แรงดันดีซี

จากแหล่งจ่ายไฟจะไหลผ่านตัวต้านทาน ผ่านขดลวดของสเต็ปปิ้งมอเตอร์และผ่านทรานซิสเตอร์ เนื่องจากแรงดัน ตกคร่อมทรานซิสเตอร์ในสภาวะอิ่มตัวมีค่าน้อย (ประมาณ 0.1 โวลต์) แรงดันจากแหล่งจ่ายไฟทั้งหมดจะทำให้เกิดกระแสไหลผ่านขดลวดโดยมีความสัมพันธ์กับผลรวมของความต้านของขดลวด (r) และความต้านทานของตัวต้านทาน (R) ดังนี้

$$Vs = I(r+R)$$

ตามปกติขดลวดของสเต็ปปิ้งมอเตอร์จะแสดงคุณสมบัติของตัวเหนี่ยวนำ (L) ซึ่งทำให้การตอบสนองต่อกระแสที่ไหลผ่านตัวมันช้า ส่งผลต่อการทำงานของมอเตอร์ที่ความเร็วสูง การใส่ตัวต้านทานอนุกรมเข้าไว้กับขดลวดจะช่วยแก้ปัญหานี้ได้ ความเหนี่ยวนำของขดลวดยังทำให้กระแสไม่หยุดไหลทันทีทันใดที่ทรานซิสเตอร์อยู่ในสถานะเปิด ทำให้เกิดแรงดันเหนี่ยวนำตกคร่อมคอลเล็คเตอร์และอิมิตเตอร์ ซึ่งเป็นสาเหตุที่ทำให้วงจรขับเสียหาย ปัญหานี้แก้ไขได้โดยการต่อฟรีวิลลิ่งไดโอด (FREEWHEELING DIODE) และ ฟรีวิลลิ่งรีซิสเตอร์ (FREEWHEELING RESISTANCE) เพื่อเป็นทางผ่านของกระแสแทน



รูปที่ 2.11 วงจรขับแบบยูนิโพลาร์

2.2 มอเตอร์กระแสตรง (DC Motor)

มอเตอร์กระแสตรงเป็นเครื่องจักรกลที่เปลี่ยนพลังงานไฟฟ้ากระแสตรงให้เป็นพลังงานกล โดยมีส่วนประกอบหลักอยู่ 3 ส่วนคือ คอมมิวเตเตอร์และแปรงถ่าน ขดอาร์เมเจอร์ (Amature Coil) ซึ่งพันอยู่ที่โรเตอร์(Rotor) ส่วนที่สร้างสนามแม่เหล็กซึ่งเป็นได้ทั้งขั้วแม่เหล็กถาวรหรืออาจเป็นขดลวด (Field Coil) ที่พันอยู่รอบแกนเหล็กก็ได้

2.2.1 หลักการทำงานของมอเตอร์กระแสตรง

เมื่อทำการจ่ายแรงดันไฟฟ้ากระแสตรงให้แก่มอเตอร์กระแสตรงจะมีกระแสไฟฟ้าไหลในขดลวดอาร์เมเจอร์ ทิศทางของกระแสไฟฟ้าจะตัดกับสนามแม่เหล็กที่เกิดจากขั้วแม่เหล็กถาวร หรือที่เกิดจากการป้อนไฟกระแสตรงให้กับขดลวดสร้างสนามแม่เหล็ก เพราะฉะนั้นจะทำให้เกิดแรงบิดขึ้นที่ขดลวดอาร์เมเจอร์ซึ่งพันอยู่ที่โรเตอร์ทำให้โรเตอร์หมุนไปในทิศทางตามกฎมือขวา คอมมิวเตเตอร์และแปรงถ่านมีหน้าที่จัดเรียงไฟฟ้ากระแสตรงให้เป็นกระแสสลับในขดลวดอาร์เมเจอร์ เพื่อให้ทิศทางของกระแสไฟที่หน้าขั้วแม่เหล็กเป็นทิศทางเดิมตลอด ซึ่งทำให้เกิดแรงบิดในทิศทางเดิมเสริมกันตลอด ขณะที่ขดลวดอาร์เมเจอร์หมุนตัดสนามแม่เหล็กจะเกิดแรงเคลื่อนไฟฟ้าเหนี่ยวนำขึ้นที่ขดอาร์เมเจอร์ เราสามารถแสดงสมการแรงเคลื่อนเหนี่ยวนำและสมการแรงบิดเนื่องจากสนามไฟฟ้า (Electromagnetic Torque) ของมอเตอร์กระแสตรงได้ดังนี้

$$E_a = K\phi\omega$$

$$T_e = K\phi I_a$$

โดยที่ E_a = แรงเคลื่อนไฟฟ้าเหนี่ยวนำซึ่งเกิดขึ้นที่ขดอาร์เมเจอร์ (Volt)

ϕ = เส้นแรงแม่เหล็ก (Waber)

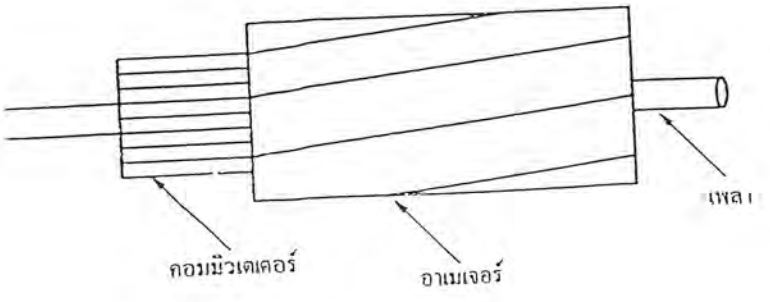
ω = ความเร็วเชิงมุมของโรเตอร์ (radian / second)

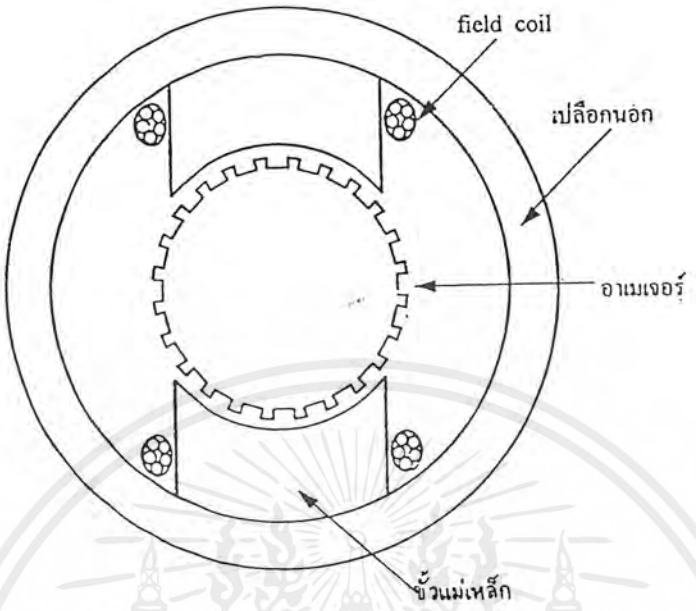
T_e = แรงบิดที่เกิดจากสนามแม่เหล็ก (Newton / meter)

I_a = กระแสที่ไหลในขดอาร์เมเจอร์ (Ampare)

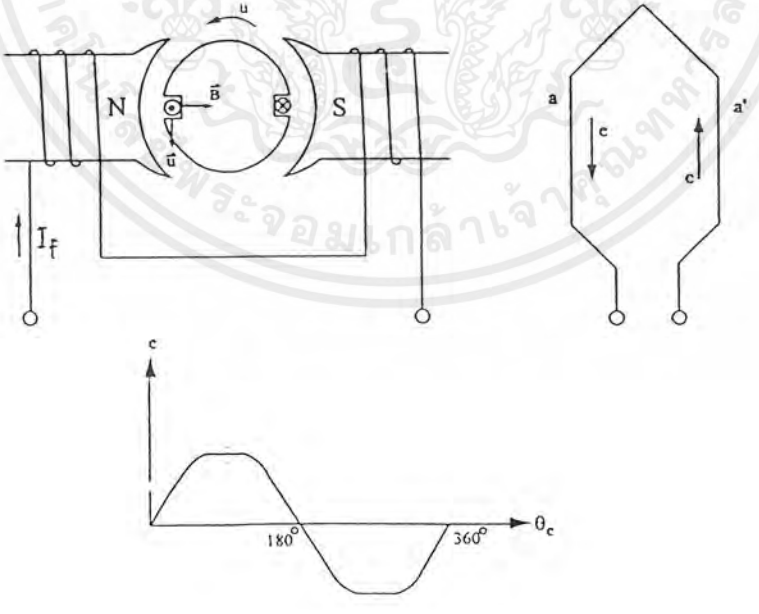
จะเห็นได้ว่าหาก ϕ มีค่าคงที่แรงเคลื่อนไฟฟ้าเหนี่ยวนำจะแปรค่าตามความเร็วเชิงมุมการหมุนของโรเตอร์ และแรงบิดที่เกิดจากสนามแม่เหล็กจะมีค่าแปรผันตามกระแสไฟฟ้าที่ไหลในขดอาร์เมเจอร์

มอเตอร์กระแสตรงที่เลือกใช้เป็นแบบที่อาร์เมเจอร์เป็นแม่เหล็กถาวร โครงสร้างมอเตอร์แบบนี้มีโมเม้นท์ (Moment) ของแรงเฉื่อยสูงที่สุด และมีค่าอินดัคแต้นซ์ของโรเตอร์สูงที่สุดด้วย ดังนั้นมอเตอร์แบบนี้จึงมีปริมาณการจุความร้อนได้สูง และสามารถทนโอเวอร์โหลดได้ในระยะเวลาที่ยาวนานโดยไม่ทำให้มอเตอร์เสียหาย

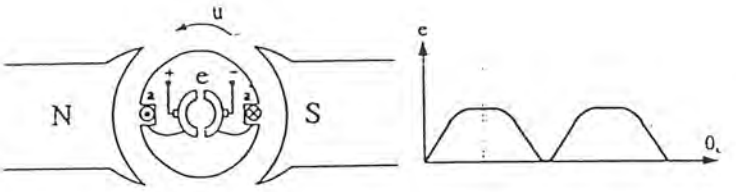




รูปที่ 2.12 แสดงส่วนประกอบต่างๆของมอเตอร์กระแสตรง



(a)



(b)

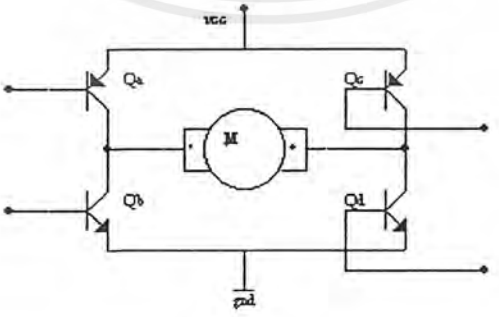
รูปที่ 2.13 (a) แสดงการเหนี่ยวนำศักคาไฟฟ้าในขดลวด

(b) แสดงการใช้วงแหวนครึ่งซี่กับแปรงถ่าน

2.2.2 วงจรขับมอเตอร์กระแสตรง

อัตราเร็วของมอเตอร์กระแสตรงควบคุมได้ด้วยการเปลี่ยนปริมาณพลังงานที่จ่ายให้กับมอเตอร์ และการเปลี่ยนโหลดของมอเตอร์ยังทำให้มีการเปลี่ยนอัตราเร็วได้อีกด้วย มีวิธีพื้นฐานอยู่ 2 วิธีที่จะควบคุมมอเตอร์ทางไฟฟ้าคือ ควบคุมแรงดันที่ให้กับมอเตอร์ด้วยวิธีเปิดและปิดแรงดันที่มีค่าคงที่อย่างรวดเร็ว และสำหรับระบบดิจิตอลนั้นจะง่ายกว่าแบบอะนาลอก วิธีพิเศษนี้เรียกว่า การมอดดูเลตความกว้างของพัลส์ (Pulse Width Modulation หรือ PWM) โดยความกว้างของช่วงคลื่นพลังงาน (Energy Pulse) จะถูกปรับในแต่ละรอบคลื่น

ตัวขยายการมอดดูเลตความกว้างของพัลส์แบบพื้นฐานของมอเตอร์ได้แก่วงจรไฟฟ้าแบบ H บริดจ์ (H Bridge circuit) จากวงจรถ้าทรานซิสเตอร์ A และ D นำกระแสมอเตอร์จะหมุนในทิศทางหนึ่ง แต่ถ้า A และ D หยุดนำกระแส ทรานซิสเตอร์ B และ C จะนำกระแสแทนทำให้มอเตอร์หมุนในทิศทางกลับกัน แต่หากทั้ง 4 ตัวทำงานพร้อมกันจะทำให้เกิดการลัดวงจรมีผลให้เกิดความเสียหายกับวงจรได้ ฉะนั้นในการใช้งานเราต้องมั่นใจว่าจะไม่เกิดความเสี่ยงนี้ขึ้น



รูปที่ 2.14 แสดงลักษณะของวงจร H บริดจ์

2.3 ส่วนที่ใช้ในการควบคุมและแสดงตำแหน่งของแขนกล

ส่วนนี้ประกอบด้วย 3 ส่วนหลักอันได้แก่ ส่วนซอฟแวร์ที่เขียนขึ้นด้วยภาษาวิชวลเบสิก ส่วนของไมโครคอนโทรลเลอร์ และส่วนของวงจรแปลงสัญญาณอะนาลอกไปเป็นสัญญาณดิจิตอลรวมทั้งโพเทนชิโอมิเตอร์ด้วย

2.3.1 ส่วนของซอฟแวร์

ซอฟแวร์เป็นโปรแกรมที่ใช้เขียนขึ้นด้วยภาษาวิชวลเบสิกโดยสามารถแบ่งหน้าที่ออกเป็น 2 ส่วนได้แก่

- การควบคุมตำแหน่งของแขนกลให้เป็นไปตามที่ผู้ใช้โปรแกรมต้องการ โดยการทำงานก็แบ่งเป็น 2 แบบคือ แบบที่ผู้ใช้ควบคุมเอง และแบบอัตโนมัติ ซึ่งข้อมูลที่นำไปใช้ควบคุมการเคลื่อนที่จะถูกเลือกโดยผู้ใช้จากข้อมูลในฐานข้อมูลที่มีอยู่แล้ว

- การแสดงผลตำแหน่งของแขนกลจะทำโดยเมื่อคอมพิวเตอร์ได้รับค่ามาจากไมโครคอนโทรลเลอร์ คอมพิวเตอร์จะนำค่าที่ได้มาสร้างเป็นภาพ 3 ภาพคือ ภาพที่โปรเจกชั่นไปบนระนาบ X-Y , Z-Y และภาพที่สร้างเป็นวงกลมเพื่อตำแหน่งของมุมที่แขนกลหมุนไปรอบแกน Y (โดยวัดแบบตามเข็มนาฬิกาเทียบกับแกน X)

2.3.2 ส่วนของไมโครคอนโทรลเลอร์

โครงงานนี้ไมโครคอนโทรลเลอร์จะมีหน้าที่ที่สำคัญด้วยกัน 2 อย่างคือ

1. การควบคุมวงจรขับมอเตอร์ ให้แต่ละวงจรควบคุมให้มอเตอร์แต่ละตัวทำการหมุนไปยังตำแหน่งที่ควรจะเป็น เพื่อให้แขนเคลื่อนที่ไปอยู่ ณ ตำแหน่งที่ต้องการ โดยแบ่งการควบคุมออกเป็น 2 แบบได้แก่ การควบคุมแบบผู้ใช้ควบคุมเอง (Manual Mode) และแบบอัตโนมัติ (Auto Mode)

2. การแสดงตำแหน่งของแขนกล ไมโครคอนโทรลเลอร์จะทำการนำค่าตำแหน่งของแขนกลทั้ง 3 ส่วนที่ได้รับมาจากวงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล แล้วไปส่งให้กับซอฟแวร์บนคอมพิวเตอร์เพื่อให้ทำการแสดงผลต่อไป

จะเห็นได้ว่าไมโครคอนโทรลเลอร์ถือเป็นส่วนที่สำคัญ ดังนั้นจึงขอแสดงรายละเอียดต่าง ๆ เกี่ยวกับไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ที่ได้เลือกใช้ในโครงงานนี้คือเบอร์ 89C51 ซึ่งจะมีรายละเอียดคร่าว ๆ ดังนี้ หัวข้อนี้จะกล่าวถึงเรื่องราวเกี่ยวกับไมโครโปรเซสเซอร์เบื้องต้น ซึ่งมีรายละเอียดเกี่ยวกับเลขฐานสอง ฐานสิบ และฐานสิบหก รวมทั้งภาษาแอสเซมบลี รหัสแอสกี (ASCII) รหัส BCD ฯลฯ

ไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ปัจจุบันนี้จะแทนรหัสคำสั่งในโปรแกรม ค่าคงที่ต่าง ๆ เช่น ตัวอักษร ตัวเลข ด้วยกลุ่มของเลขฐานสอง ทั้งนี้เพราะเลขฐานสอง

เหมาะสมที่จะนำมาใช้แทนความหมายของระดับสัญญาณไฟฟ้าได้เป็นอย่างดี เนื่องจากมีค่าเพียง 2 ค่าเท่านั้น คือ ศูนย์และหนึ่ง

MCS-51 เป็นชื่อตระกูลของไมโครโปรเซสเซอร์ที่มีขนาด 8 บิตนั่นคือ สามารถประมวลผลได้ครั้งละ 8 หลักในเลขฐานสอง ขนาดของรีจิสเตอร์และหน่วยความจำแต่ละตำแหน่งภายในชิปก็มีขนาด 8 บิตเช่นกัน เลขฐานสองขนาด 8 หลัก หรือที่เรียกกันว่า 8 บิต สามารถใช้แทนเลขจำนวนต่าง ๆ ได้มากถึง 256 ค่าแตกต่างกัน (2^8) ดังแสดงในตารางที่ 2.4

ในตารางที่ 2.4 แสดงให้เห็นการใช้เลขฐานสองขนาด 8 หลัก (8 บิต หรือ 1 ไบต์) แทนจำนวนต่าง ๆ ตั้งแต่ 0 ถึง 255 และยังแสดงให้เห็นการแทนเลขจำนวนเต็มลบด้วยเลขฐานสองโดยการใช้บิต 7 เป็นบิตบอกเครื่องหมาย (บิต 7 เป็น 0 หมายถึงเลขนั้นเป็นจำนวนเต็มบวก บิต 7 เป็น 1 หมายถึงเป็นเลขจำนวนเต็มลบ) ดังนั้นจะสามารถใช้เลขฐานสองแทนเลขจำนวนเต็มทั้งบวกและลบได้ตั้งแต่ –127 ถึง 128 หากต้องการใช้เลขฐานสองแทนจำนวนที่มีค่ามากกว่านี้ก็สามารถกระทำได้โดยการใช้เลขฐานสองที่มีจำนวนหลักหรือจำนวนบิตมากขึ้นนั่นเอง เช่น ใช้เลขฐานสองขนาด 16 บิตหรือ 2 ไบต์แทนเลขจำนวนเต็มบวกเพียงอย่างเดียวได้ตั้งแต่ 0 ถึง 65535

เลขฐานสิบขนาดสองหลักสามารถจะใช้เลขฐานสองขนาด 8 บิตแทนได้หลายวิธี เช่น ใช้ขนาดของเลขฐานสองเป็นตัวบอกขนาดของเลขฐานสิบ วิธีนี้มีข้อเสียคือ ต้องคำนวณค่าขนาดของเลขฐานสองเสียก่อนจึงจะทราบว่าเลขฐานสิบมีค่าเท่าใด ดังนั้นจึงมีวิธีในการแทนเลขฐานสองเป็นเลขฐานสิบอีกวิธีหนึ่ง นั่นคือใช้เลขฐานสองขนาด 4 บิตแทนเลขฐานสิบ 1 หลักดังนี้

0000 แทนเลข 0	0101 แทนเลข 5
0001 แทนเลข 1	0110 แทนเลข 6
0010 แทนเลข 2	0111 แทนเลข 7
0011 แทนเลข 3	1000 แทนเลข 8
0100 แทนเลข 4	1001 แทนเลข 9

วิธีการแทนเลขฐานสิบด้วยเลขฐานสองแบบนี้เรียกว่า การแทนเลขแบบ BCD การหาค่าเลขฐานสิบจากเลขฐานสองขนาด 8 หลักนี้ กระทำโดยการแบ่งเลขฐานสองออกเป็น 4 บิต แล้วเปลี่ยนค่ากลับไปเป็นเลขฐานสิบได้โดยตรง ซึ่งไม่จำเป็นต้องคำนวณใด ๆ ทั้งสิ้น ดังนั้นวิธีนี้มีข้อดีตรงที่หาค่าเลขฐานสิบได้ง่ายและสะดวกรวดเร็วมากดังตัวอย่าง 0110 0011 =63 BCD

ค่าต่ำสุดของเลขฐานสิบในรหัส BCD ขนาด 2 หลักคือ 00 ค่าสูงสุดของเลขฐานสิบในรหัส BCD ขนาด 2 หลักคือ 1001 1001 = 99 ส่วนเลขลบในเลขฐานสิบไม่สามารถใช้รหัส BCD แทนได้

รูปแบบของบิต	เลขไบนารี	เลขฐานแปด	เลขฐานสิบหก	เลขฐานสิบแบบไม่คิดเครื่องหมาย	เลขฐานสิบแบบคิดเครื่องหมาย
00000000	0B	0Q	00H	0	0
00000001	1B	1Q	01H	1	+1
.....	..	...	...	..	....
00000111	111B	7Q	07H	7	+7
00001000	1000B	10Q	08H	8	+8
00001001	1001B	11Q	09H	9	+9
00001010	1010B	12Q	0AH	10	+10
.....	..	...	...	..	....
00001111	1111B	17Q	0FH	15	+15
00010000	10000B	20Q	10H	16	+16
.....	..	...	...	..	....
01111111		177Q	7FH	127	+127
10000000		200Q	80H	128	-128
10000001		201Q	81H	129	-127
.....		...	...	...	....
11111110		376Q	0FEH	254	-2
11111111		377Q	0FFH	255	-1
	10000000B				
	10000001B				
	.....				
	11111110B				
	11111111B				

ตารางที่ 2.4 แสดงรหัสของเลขฐานสองที่ใช้แทนตัวเลข

อักขระต่าง ๆ (ตัวอักษร ตัวเลข เครื่องหมายพิเศษต่าง ๆ เช่น ? ! $ @) มีวิธีการแทนเป็นเลขฐานสองตามมาตรฐานของรหัสแอสกีซึ่งใช้กันทั่วไป อักขระเหล่านี้ต้องการเลขฐานสองเพียง 7 บิตเท่านั้น (แทนอักขระได้ทั้งหมด 128 ตัวซึ่งก็เพียงพอกับตัวเลข อักษรภาษาอังกฤษทั้งตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก รวมทั้งเครื่องหมายพิเศษต่างๆ) ดังนั้นเลขฐานสองขนาด 1 ไบต์ (8 บิต)จะสามารถแทนตัวอักขระได้ 1 ตัวพอดี ด้วยวิธีนี้การแทนข้อความสามารถกระทำได้โดยใช้เลขฐานสองขนาด 1 ไบต์มาประกอบกัน และเนื่องจากรหัสแอสกีก็ใช้เลขฐานสองเพียง 7 บิต จึงเหลือบิตที่ไม่ได้ใช้ 1 บิต คือบิตสูงสุดในแต่ละไบต์ (D7) โดยปกติบิตสูงสุดนี้จะมีค่าเท่ากับศูนย์เสมอ แต่ในบางกรณีมันอาจถูกใช้เป็นบิตตรวจสอบความถูกต้องของข้อมูลได้ โดยบิตนี้จะมีค่าเป็นศูนย์หรือหนึ่งขึ้นอยู่กับจำนวน "1" ใน 7 หลักที่เหลือของเลขฐานสอง 1 ไบต์

โปรแกรมที่ใช้ควบคุมการทำงานของไมโครโปรเซสเซอร์จะประกอบไปด้วยรหัสคำสั่งที่เป็นเลขฐานสองที่มีค่าแตกต่างกันขึ้นอยู่กับแต่ละคำสั่งมาเรียงต่อกัน ซีพียูในไมโครโปรเซสเซอร์จะปฏิบัติคำสั่งแต่ละคำสั่งในโปรแกรมไปเรื่อย ๆ ตราบเท่าที่ยังมีพลังงานจ่ายให้แก่ตัวไมโครโปรเซสเซอร์อยู่

ในไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์โปรแกรมจะถูกเก็บในหน่วยความจำประเภท ROM, PROM, EPROM เพื่อสะดวกในการใช้งาน โปรแกรมต่าง ๆ จะประกอบไปด้วยคำสั่งที่ควบคุมให้ซีพียูปฏิบัติงาน คำสั่งอาจมีความยาว 1 ไบต์หรือหลายไบต์ก็ได้ ในกรณีที่คำสั่งมีความยาวหลายไบต์ ไบต์แรกของคำสั่งจะเป็นไบต์ที่ระบุชนิดและลักษณะการทำงานของคำสั่งให้ซีพียูทราบ ส่วนไบต์อื่นที่เหลือจะเป็นข้อมูลที่ซีพียูนำมาประมวลผล คำสั่งของ MCS-51 จะมีขนาดสูงสุดเพียง 3 ไบต์

การเขียนโปรแกรมคือการเขียนคำสั่งแต่ละคำสั่งต่อกันไปเรื่อย ๆ คำสั่งแต่ละคำสั่งจะมีรหัสคำสั่งที่เป็นเลขฐานสองแตกต่างกัน การเขียนโปรแกรมที่เป็นรหัสคำสั่งหรือรหัสเลขฐานสองล้วน ๆ (Machine Language) จะมีความยากลำบากในการเขียนและพัฒนาโปรแกรมเป็นอันมาก เพราะมนุษย์คุ้นเคยกับตัวหนังสือมากกว่าตัวเลข ดังนั้นรหัสคำสั่งที่เป็นเลขฐานสองจึงถูกแทนด้วยรหัสที่เป็นคำอธิบายคำสั่งคร่าว ๆ เช่น MOV, LOAD, JUMP รหัสคำสั่งเหล่านี้เราเรียกว่ารหัสนีโมนิก (Mnemonic) ดังนั้นการเขียนโปรแกรมโดยใช้คำสั่งที่เป็นตัวหนังสือจะมีความสะดวกและทำความเข้าใจได้ง่ายมากขึ้น ภาษาที่ใช้รหัสนีโมนิกแทนเลขฐานสองมีชื่อเรียกว่า ภาษาแอสเซมบลี ดังนั้นการเขียนโปรแกรมที่พื้นฐานที่สุดคือ การเขียนโปรแกรมภาษาแอสเซมบลี ซึ่งจำเป็นจะต้องแปลรหัสนีโมนิกที่เป็นภาษาแอสเซมบลีแต่ละคำสั่งให้เป็นรหัสคำสั่งที่เป็นเลขฐานสองซึ่งตรงกับแต่ละคำสั่งที่ซีพียูรู้จัก การแปลรหัสนีโมนิกภาษาแอสเซมบลีไปเป็นรหัสภาษา

เครื่อง อาจกระทำโดยการเปิดตารางคำสั่งเทียบเคียงกัน หรืออาจเขียนโปรแกรมภาษาแอสเซมบลี เก็บไว้เป็นไฟล์ในเครื่องไมโครคอมพิวเตอร์ จากนั้นใช้โปรแกรมที่ทำหน้าที่แปลรหัสนีโมนิกที่เรียกว่า "โปรแกรมแอสเซมเบลอร์" ซึ่งใช้ในเครื่องไมโครคอมพิวเตอร์แปลงให้เป็นไฟล์ซึ่งเก็บรหัสคำสั่งโดยตรง โปรแกรมแปลภาษาแอสเซมบลีของ MCS-51 มีอยู่ด้วยกันหลายโปรแกรม เช่น ASM51 ของบริษัท Intel หรือ SXA51 ของบริษัท Binary Technology เป็นต้น

โครงสร้างของไมโครคอนโทรลเลอร์ตระกูล MCS-51

จะกล่าวถึงรายละเอียดคร่าว ๆ ของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีจุดประสงค์เพื่อให้ผู้อ่านทำความเข้าใจและมองเห็นภาพกว้าง ๆ ของไมโครคอนโทรลเลอร์ตระกูลนี้ เพื่อเป็นพื้นฐานในการศึกษารายละเอียดต่อไป

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีสมาชิกในตระกูลอยู่หลายเบอร์ด้วยกัน แต่ละเบอร์จะมีคุณสมบัติพิเศษบางอย่างแตกต่างกัน เช่น มีหน่วยความจำภายในสำหรับเก็บโปรแกรมและข้อมูลภายในชิปเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลในตัว สามารถรับสัญญาณอินเตอร์รัปต์ได้หลายชนิด ทำกระบวนการ DMA (Direct Memory Access) ได้ในตัว มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์เพิ่มขึ้น

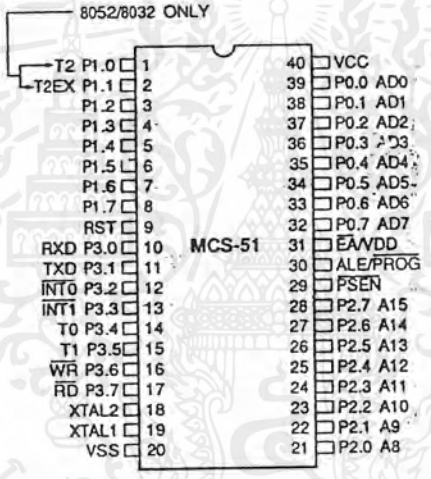
ไมโครคอนโทรลเลอร์เบอร์ที่นับได้ว่าเป็นเบอร์พื้นฐานสำหรับตระกูล MCS-51 นี้ ได้แก่ เบอร์ 8051, 8031 และ 8751 โดยเบอร์ 8051 เป็นสมาชิกตัวแรกในตระกูลที่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปเป็น ROM ขนาด 4 กิโลไบต์ และหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายใน MCS-51 (RAM) เองจำนวน 128 ไบต์ มีพอร์ต 8 บิต 4 พอร์ต มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิตรวม 2 ตัว รับสัญญาณอินเตอร์รัปต์จากภายนอกได้ 2 ชนิด สามารถรับและส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารข้อมูลแบบอนุกรม มีวงจรออสซิลเลเตอร์ (Oscillator) เพื่อสร้างสัญญาณนาฬิกาควบคุมการทำงานในตัวเอง ส่วนเบอร์ 8751 จะมีคุณสมบัติเหมือนเบอร์ 8051 ทุกอย่าง จะต่างกันก็เพียงชนิดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปของเบอร์ 8751 จะเป็น EPROM แทนที่จะเป็น ROM ส่วนเบอร์ 8031 จะเหมือนกับเบอร์ 8051 ต่างกันเพียง 8031 ไม่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปเท่านั้น

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์ใช้แรงดันไฟเพียง 5 โวลต์ในการทำงาน ส่วนกระแสไฟฟ้าที่ใช้จะแตกต่างกันไปตามชนิดของเทคโนโลยีที่ใช้ในการผลิต เบอร์ของไมโครคอนโทรลเลอร์ตระกูลนี้ที่มีตัวอักษร C อยู่ตรงกลางเบอร์ เช่น 80C31, 80C51 จะเป็นเบอร์ของชิปที่ผลิตโดยอาศัยเทคโนโลยี CHMOS ซึ่งใช้พลังงานในการทำงานน้อยกว่า และสามารถควบคุมการใช้พลังงานของตัวชิปได้จากโปรแกรมเพื่อการประหยัดพลังงานในระบบ

MCS-51 เป็นตระกูลของไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้นมาจากตระกูล MCS-48 ดังนั้นจึงมีความสามารถเหนือกว่าอยู่หลายอย่างซึ่งจะไม่ขอกล่าวรายละเอียดไว้ในที่นี้ ข้อดีของ MCS-51 เมื่อเทียบกับ MCS-48 เช่น ความเร็วในการประมวลผลของ MCS-51 สามารถใช้ความถี่ได้ถึง 12 เมกะเฮิรตซ์ หรือสำหรับบางเบอร์ในตระกูลสามารถใช้ได้ถึง 16 เมกะเฮิรตซ์ ทำให้ช่วงเวลาในการทำงานแต่ละคำสั่งน้อยมาก เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์ คำสั่งที่ใช้เวลาน้อยที่สุดจะใช้เวลาเพียง 1 ไมโครวินาที ส่วนคำสั่งที่ใช้เวลามากที่สุดจะใช้เพียง 4 ไมโครวินาทีเท่านั้น

ตำแหน่งขาของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาที่พื้นฐานที่เหมือนกันดังแสดงในรูปที่ 2.15



รูปที่ 2.15 แสดงการจัดตำแหน่งขาต่างๆ ของไมโครคอนโทรลเลอร์ตระกูลMCS-51

หน้าที่การใช้งานแต่ละขาของชิปไมโครคอนโทรลเลอร์ในตระกูล MCS-51 มีดังนี้

- ขา Vss (ขา 20) สำหรับต่อลงกราวด์
- ขา Vss (ขา 40) สำหรับต่อแหล่งจ่ายแรงดันกระแสตรงขนาด 5 โวลต์
- ขาพอร์ต 0 (ขา 32-39) ใช้เป็นขาสำหรับพอร์ต 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากจะใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ high impedance) นอกจากใช้งานเป็นอินพุตเอาต์พุตพอร์ตแล้วพอร์ต 0 ยังใช้ในการติดต่อกับหน่วยความจำสำหรับเก็บโปรแกรมและสำหรับข้อมูลภายนอกชิปด้วย โดยส่งค่า

แอดเดรสบัสไบต์ต่ำ(A0-A7) และมัลติเพลกซ์กับการรับส่งข้อมูล (D0-D7) จากหน่วยความจำภายนอกในระหว่างการเขียนหรืออ่านข้อมูลโดยมีวงจรพูลอัพ (Pull Up) อยู่ภายใน

- ขาพอร์ต 1 (ขา 1-8) มี 8 ขา (P1.0-P1.7) สามารถใช้งานเป็นอินพุตหรือเอาต์พุตพอร์ตทั่วไปได้ หากต้องการใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อให้มีสถานะ high impedance โดยมีวงจรพูลอัพภายใน

ขา P1.0, P1.1 ในเบอร์ 8052 ใช้งานในหน้าที่อย่างอื่นนอกเหนือจากใช้เป็นอินพุตเอาต์พุตพอร์ตทั่วไปด้วย

- ขาพอร์ต 2 (ขา21-28) มี 8 ขา (P2.0-P2.7) ขนาด 8 บิตแบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ โดยหากใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อบังคับให้ขาอยู่ในสถานะ ถูกปล่อยลอย นอกจากจะใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปแล้ว พอร์ต 2 ยังใช้ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลภายนอกด้วย โดยใช้สำหรับส่งค่าแอดเดรสบัสไบต์สูง(A8-A15) และมีวงจรพูลอัพภายใน

- ขาพอร์ต 3 (ขา10-17) มี 8 ขา (P3.0-P3.7) สามารถใช้งานเป็นอินพุตเอาต์พุตพอร์ตทั่วไปได้ หากต้องการใช้งานเป็นอินพุตพอร์ตต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้ เพื่อให้มีสถานะถูกปล่อยลอยโดยใช้วงจรพูลอัพภายใน นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆ อีกหลายอย่างดังนี้

ขา P3.0	ใช้รับข้อมูลจากภายนอกแบบอนุกรม
ขา P3.1	ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม
ขา P3.2	ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิดที่ 0
ขา P3.3	ใช้เป็นอินพุตเพื่อรับสัญญาณอินเตอร์รัปต์ชนิดที่ 1
ขา P3.4	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 0
ขา P3.5	สัญญาณอินพุตให้เคาน์เตอร์ของไทม์เมอร์ 1
ขา P3.6	ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป
ขา P3.7	ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป

การใช้งานพอร์ต 3 ในหน้าที่พิเศษดังกล่าวนี้จะต้องโหลดค่า 1 ไปยังแต่ละบิตที่ต้องการใช้ก่อนทุกครั้ง

- ขา RST (ขา9) ใช้สำหรับการรีเซตวงจรทุกอย่างภายในชิป เพื่อเริ่มต้นการทำงานใหม่ การรีเซตใช้เมื่อเริ่มจ่ายพลังงานหรือเมื่อโปรแกรมเกิดทำงานผิดพลาด เมื่อต้องการที่จะรีเซตชิป MCS-51 ขานี้ต้องมีสถานะ 1 เป็นเวลาอย่างน้อย 2 แมชชีนไซเกิลระหว่างที่ออสซิลเลเตอร์ยังทำงานอยู่ โดยต้องต่อตัวต้านทานค่า 8.2 กิโลโอห์มเพื่อทำหน้าที่พูลดาวน์ (รักษาค่าแรงดันไฟฟ้าให้มีสถานะเป็นกราวด์) และเพื่อให้ตัวชิปรีเซตเองเมื่อเริ่มจ่ายพลังงานให้ต่อตัวเก็บประจุขนาด 10 ไมโครฟารัดคร่อมระหว่างขา RST กับ Vcc

- ขา ALE/PROG (ขา 30) เป็นขาที่ใช้ส่งสัญญาณออกไปภายนอกเพื่อควบคุมการแลตช์ค่าแอดเดรสไบต์ต่ำ (Address Latch Enable) จากพอร์ต 0 ในระหว่างการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอก ปกติเมื่อไม่มีการติดต่อหน่วยความจำภายนอกขานี้จะส่งสัญญาณพัลส์ออกมาด้วยความถี่ 1/8 ของความถี่ออสซิลเลเตอร์ที่ใช้ตลอดเวลา ดังนั้นเราสามารถใช้ความถี่ที่ได้จากขานี้ไปใช้งานอย่างอื่นได้ แต่ความถี่ที่ขานี้จะลดลงครึ่งหนึ่งในระหว่างติดต่อกับหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิป นอกจากนี้ยังใช้สำหรับควบคุมการเขียนโปรแกรมลงไปใน EPROM สำหรับ MCS-51 เบอร์ที่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปเป็น EPROM

- ขา PSEN (ขา 29) ใช้ในการส่งสัญญาณสโตรบเพื่ออ่านคำสั่งจากโปรแกรมที่เก็บไว้ในหน่วยความจำภายนอกชิป (Program Strobe Enable) เมื่อชิปทำงานด้วยโปรแกรมจากภายนอกขานี้จะส่งสัญญาณสโตรบ 2 ครั้งในแต่ละแมชชีนไซเกิล แต่ในช่วงการเขียนหรืออ่านข้อมูลกับหน่วยความจำภายนอกหรือใช้โปรแกรมจากหน่วยความจำภายในชิปจะไม่มีสัญญาณนี้ออกมา

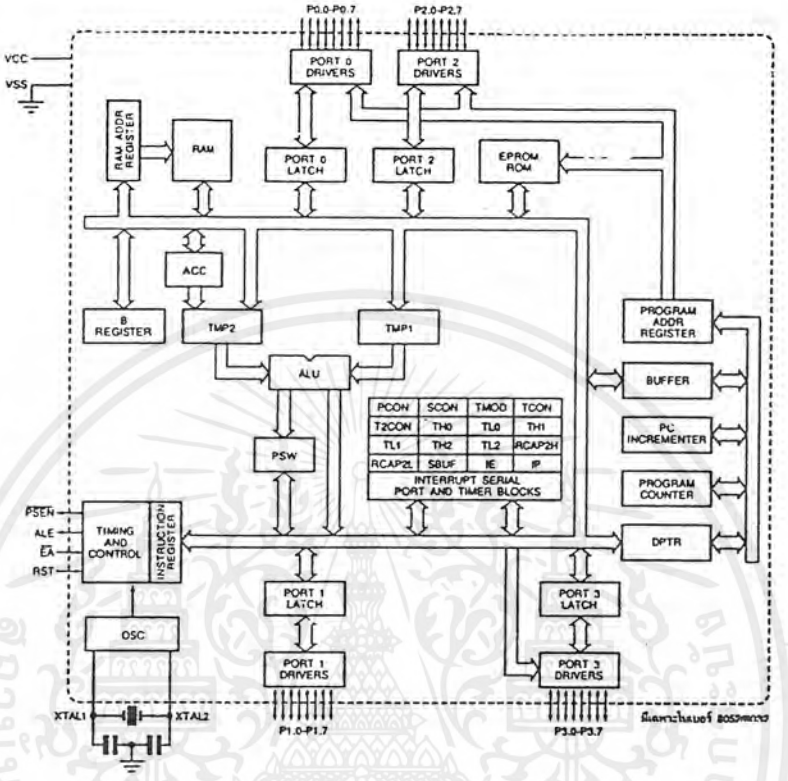
- ขา EA/Vpp (ขา 31) ใช้เลือกให้ชิปทำงานจากโปรแกรมที่อยู่ภายในหรือภายนอก หากขานี้มีสถานะเป็น 0 หมายถึงให้ใช้โปรแกรมจากหน่วยความจำที่เก็บโปรแกรมภายนอก แต่หากเป็น 1 หมายถึงให้ใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป MCS-51 ที่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปสามารถเลือกใช้ได้ทั้งสองแบบด้วยการต่อขา EA กับไฟเลี้ยงหรือกราวด์ตามลำดับ ส่วนตัวที่ไม่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในให้ต่อขานี้ลงกราวด์เสมอ

- ขา XTAL 1 (ขา 19) ใช้สำหรับต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิลเลเตอร์

- ขา XTAL 2 (ขา 18) ใช้สำหรับต่อคริสตอลภายนอก โดยเป็นเอาต์พุตออกมาจากวงจรออสซิลเลเตอร์

โครงสร้างภายในของ MCS-51

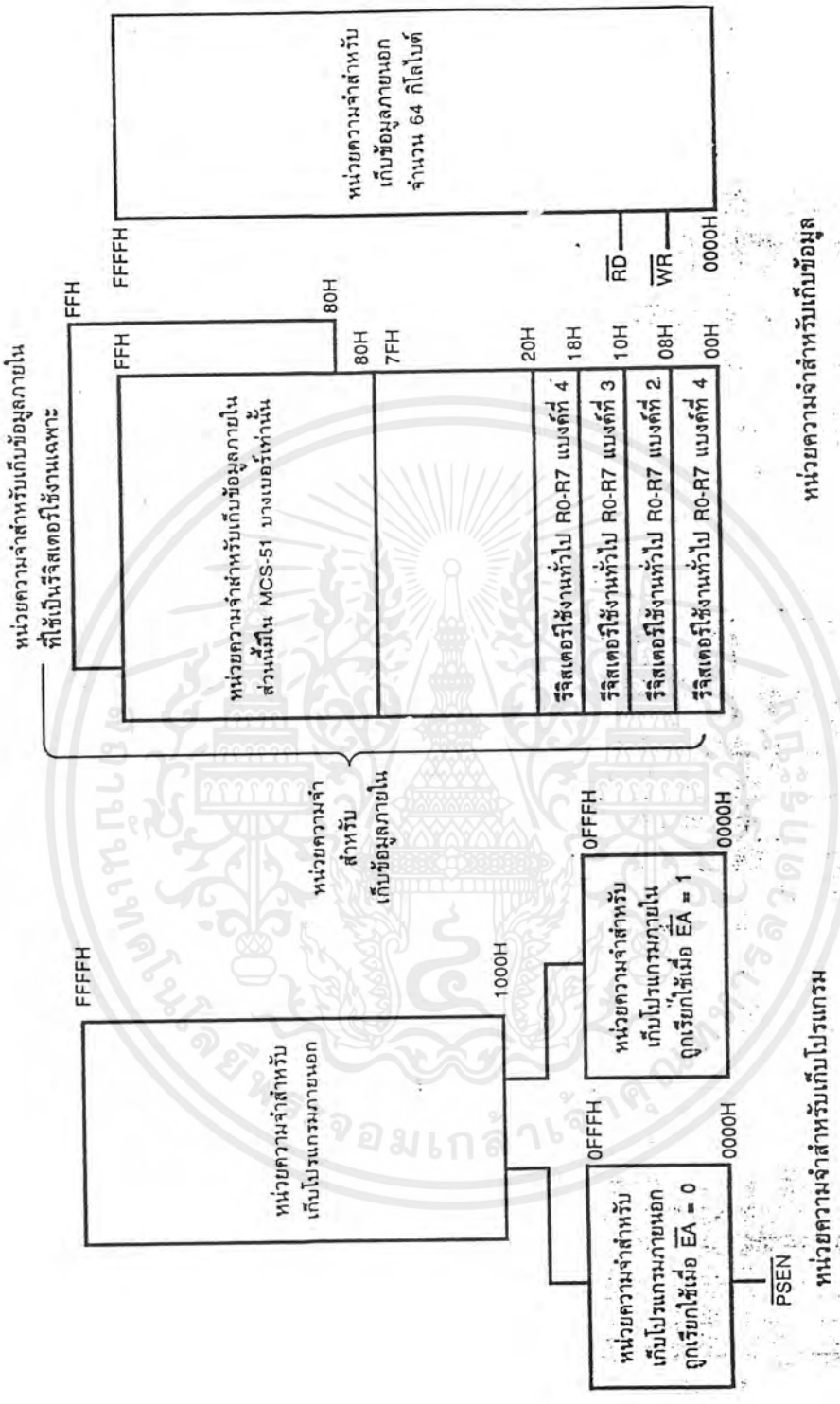
โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังรูปที่ 2.16



รูปที่ 2.16 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MSC-51

โครงสร้างหน่วยความจำภายใน MCS-51 ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ทุกเบอร์จะแบ่งหน่วยความจำออกเป็นสองส่วน คือ หน่วยความจำสำหรับเก็บโปรแกรม (Program Memory) และหน่วยความจำสำหรับเก็บข้อมูล (Data Memory)

หน่วยความจำสำหรับเก็บโปรแกรมจะใช้เก็บโปรแกรมควบคุมการทำงานของชิป MCS-51 บางเบอร์จะมีหน่วยความจำส่วนนี้อยู่ภายในชิป แต่บางเบอร์จะไม่มี ทำให้ต้องเก็บโปรแกรมไว้ในหน่วยความจำภายนอกทั้งหมด ส่วนหน่วยความจำส่วนที่สองคือหน่วยความจำสำหรับเก็บข้อมูลซึ่งใช้สำหรับเก็บข้อมูลระหว่างการทำงาน MCS-51 ทุกเบอร์จะมีหน่วยความจำส่วนนี้อยู่ภายในชิปจำนวนหนึ่ง แต่จะมีจำนวนมากหรือน้อยขึ้นกับเบอร์ของชิป โครงสร้างหน่วยความจำทั้งหมดของ MCS-51 มีดังในรูปที่ 2.17



รูปที่ 2.17 แสดงโครงสร้างของหน่วยความจำทั้งหมดของ MCS-51

- หน่วยความจำสำหรับเก็บโปรแกรม หน่วยความจำสำหรับเก็บโปรแกรมใน MCS-51 จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บโปรแกรมภายในชิป (Internal Program Memory) และหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป (External Program Memory) ขนาดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิปมีได้เต่ 0, 4, 8, 16 กิโลไบต์ ขึ้นอยู่กับเบอร์ของชิป

- หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บข้อมูลของ MCS-51 จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิป และหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป หน่วยความจำสำหรับเก็บข้อมูลภายในชิปของ MCS-51 ยังแบ่งออกเป็น 2 ส่วนย่อยดังนี้คือ ส่วนที่ใช้เก็บข้อมูลทั่วไป (Internal RAM) และส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (Special Function Register)

หน่วยความจำส่วนที่ใช้เก็บข้อมูลทั่วไปภายในชิปเป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ภายใน MCS-51 หน่วยความจำส่วนนี้มีไว้สำหรับเก็บข้อมูลในขณะทำงาน ส่วนหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะเป็นหน่วยความจำสำหรับเก็บข้อมูลภายใน MCS-51 ซึ่งถูกกำหนดให้เป็นรีจิสเตอร์ใช้งานเฉพาะเพื่อควบคุมการทำงานและบอกสถานะของซีพียู แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิปทั้งสองบริเวณมีดังในรูปที่ 2.18

รูปที่ 2.18 แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิป

MCS-51 ทุกเบอร์จะมีหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปอย่างน้อย 128 ไบต์ ไปจนถึง 256 ไบต์ ทั้งนี้ขึ้นกับเบอร์ของชิป หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิป บริเวณ 128 ไบต์แรกมีชื่อเรียกว่า Lower 128 และในบริเวณ 128 ไบต์หลังที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า Upper 128 ดังแสดงในรูปที่ 2.18

หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์หลัง (ตำแหน่ง 80H ขึ้นไป) จะมีตำแหน่งตรงกับหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ(ตำแหน่ง 80H ขึ้นไปเช่นกัน) โดยมีวิธีการเข้าถึงข้อมูลในหน่วยความจำทั้งสองส่วนไม่เหมือนกัน

รีจิสเตอร์ใช้งานเฉพาะ เนื่องจาก MCS-51 ถูกออกแบบไว้สำหรับใช้ควบคุมระบบโดยเฉพาะ จึงทำให้มีความสามารถเฉพาะตัวหลายอย่าง ซึ่งจำเป็นต้องอาศัยวงจรภายในชิปที่มีเพิ่มขึ้นจากไมโครโปรเซสเซอร์ทั่วไป การควบคุมการทำงานจะกระทำผ่านรีจิสเตอร์ที่ถูกกำหนดหน้าที่ไว้แล้ว ดังนั้นหากต้องการใช้งานให้มีประสิทธิภาพจำเป็นต้องทราบหน้าที่การทำงานของรีจิสเตอร์ใช้งานเฉพาะแต่ละตัว รีจิสเตอร์ใช้งานเฉพาะทั้งหมดจะอยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปบริเวณที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะดังได้กล่าวมาแล้ว รีจิสเตอร์ใช้งานเฉพาะทั้งหมดแสดงได้ดังรูปที่ 2.20

รูปที่ 2.19 แสดงแผนที่การจัดแบ่งช่วงแอดเดรสใช้งานของ RAM ภายใน

ตำแหน่งแอดเดรส	(MSB) บิตแอดเดรส (LSB)								รีจิสเตอร์หน้าที่พิเศษ
	WDT	T32	SERR	IZC	P3HZ	P2HZ	P1HZ	ALF	
0F8H	FF	FE	FD	FC	FB	FA	F9	F8	IOCON
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	F0	RS1	RS0	0V	F1	P	
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0CDH	ไม่สามารถเข้าถึงได้ระดับบิต								TH2
0CCH	ไม่สามารถเข้าถึงได้ระดับบิต								TL2
0CBH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2H
0CAH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2L
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	
0C8H	CF	CE	CD	CC	CB	CA	C9	C8	T2CON
	PCT		PT2	PS	PT1	PX1	PT0	PX0	
0B8H	BF	—	BD	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA		ET2	ES	ET1	EX1	ET0	EX0	
0A8H	AF	—	AD	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
99H	ไม่สามารถเข้าถึงได้ระดับบิต								SBUF
	SM0	SM1	SM2	REN	TB8	RB8	T1	R1	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
8DH	ไม่สามารถเข้าถึงได้ระดับบิต								TH1
8CH	ไม่สามารถเข้าถึงได้ระดับบิต								TH0
8BH	ไม่สามารถเข้าถึงได้ระดับบิต								TL1
8AH	ไม่สามารถเข้าถึงได้ระดับบิต								TL0
89H	ไม่สามารถเข้าถึงได้ระดับบิต								TMOD
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
87H	ไม่สามารถเข้าถึงได้ระดับบิต								PCON
83H	ไม่สามารถเข้าถึงได้ระดับบิต								DPH
82H	ไม่สามารถเข้าถึงได้ระดับบิต								DPL
81H	ไม่สามารถเข้าถึงได้ระดับบิต								SP
80H	87	86	85	84	83	82	81	80	P0

รูปที่ 2.20 แสดงการจัดหน่วยความจำและตำแหน่งของรีจิสเตอร์หน้าที่พิเศษต่างๆ

ในส่วนของหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลที่อยู่ภายนอกชิปจะเป็นหน่วยความจำส่วนที่อยู่ภายนอกชิปที่ผู้ใช้ต้องหามาติดตั้งเพิ่มเอง การติดต่อระหว่างชิปกับหน่วยความจำทั้งสองส่วนจะใช้ขา 32 ถึง 39 (พอร์ต 0) เป็นตัวส่งค่าแอดเดรสไบต์ต่ำ (A0-A7) และใช้รับส่งข้อมูลกับหน่วยความจำด้วย(ใช้เป็นดาต้าบัส) ส่วนค่าแอดเดรสไบต์สูง (A8-A15) จะใช้ขา 21 ถึง 28 (พอร์ต 2) ดังนั้นเมื่อพอร์ต 0 และ พอร์ต 2 ถูกใช้ในการติดต่อกับหน่วยความจำภายนอก (ทั้งหน่วยความจำสำหรับเก็บโปรแกรมและสำหรับเก็บข้อมูล) จะทำให้เหลือพอร์ตสำหรับใช้งานอื่นๆ น้อยลง

รีจิสเตอร์สำหรับใช้งานทั่วไป MCS-51 มีรีจิสเตอร์ใช้งานทั่วไปที่สามารถนำมาใช้งานได้คือรีจิสเตอร์ A , B(อยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ แต่นับเป็นรีจิสเตอร์ใช้งานทั่วไปเพราะไม่ถูกกำหนดหน้าที่ใช้งานโดยตรง) และรีจิสเตอร์ใช้งานทั่วไป R0-R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์แรกดัง แสดงในรูปที่ 2.19 รีจิสเตอร์ใช้งานทั่วไป R0-R7 ในชิปมีด้วยกันทั้งหมด 4 กลุ่ม แต่ละกลุ่มประกอบด้วยรีจิสเตอร์จำนวน 8 ตัว (R0-R7) ซึ่งมีชื่อเรียกเหมือนกัน ดังนั้นจำนวนของรีจิสเตอร์ใช้งานทั่วไปจึงมีทั้งหมด 32 ตัว ในการทำงานขณะใดๆ รีจิสเตอร์ทั้ง 4 กลุ่มนี้จะถูกเลือกใช้งานเพียงกลุ่มเดียวเท่านั้นซึ่งกระทำได้โดยการเซตหรือเคลียร์บิต RS0 , RS1 ในรีจิสตอร์ใช้งานเฉพาะ PSW ดังแสดงในรูปที่ 2.21 ค่าที่เปลี่ยนแปลงไปในรีจิสเตอร์ทั่วไปที่ถูกเลือกใช้งานในขณะนั้นจะไม่มีผลต่อรีจิสเตอร์ใช้งานทั่วไปที่มีชื่อเดียวกันแต่อยู่คนละกลุ่มเลย โครงสร้างเช่นนี้ทำให้มีความสะดวกในการเขียนโปรแกรมเป็นอันมาก

โครงสร้างพอร์ต MCS-51 ทุกเบอร์จะมีพอร์ตขนาด 8 บิตจำนวน 4 พอร์ต(P0,P1,P2,P3) โดยสามารถกำหนดให้ทำงานแบบพอร์ตขนานขนาด 8 บิต 4 พอร์ตหรือจะใช้เป็นพอร์ตขนาด 1บิตได้ถึง 32 พอร์ต ทั้งนี้ผู้ใช้ยังสามารถกำหนดให้แต่ละพอร์ตใช้งานเป็นอินพุตพอร์ตหรือเอาต์พุตพอร์ตอย่างใดอย่างหนึ่งได้อย่างอิสระ

กรณีที่ผู้ออกแบบต้องการใช้หน่วยความจำภายนอกไม่ว่าจะเป็นหน่วยความจำสำหรับเก็บข้อมูลหรือสำหรับโปรแกรม พอร์ต 0 จะถูกกำหนดการใช้งานเป็นดาต้าบัสและแอดเดรสบัสไบต์ต่ำ ส่วนพอร์ต 2 จะถูกกำหนดให้เป็นตัวส่งค่าแอดเดรสไบต์สูงและบางส่วนของพอร์ต 3 จะถูกใช้ส่งสัญญาณควบคุมหรือเป็นคอนโทรลบัส (สัญญาณที่ใช้ควบคุมการอ่านหรือเขียนข้อมูล) หากหน่วยความจำที่ใช้ภายนอกต้องการไม่ถึง 64 กิโลไบต์ พอร์ต 2 ที่ใช้เป็นแอดเดรสไบต์สูงจะไม่ถูกนำมาใช้ทั้งหมด แต่พอร์ต 0 จะถูกใช้หมดทั้ง 8 เส้นเพราะต้องใช้เป็นดาต้าบัส ส่วนพอร์ต 3 จะนำมาใช้หรือไม่นั้นขึ้นอยู่กับว่ามีหน่วยความจำภายนอกส่วนที่ใช้เก็บข้อมูลด้วยหรือไม่ (ต้องการสัญญาณควบคุมการอ่านหรือเขียนข้อมูลหรือไม่นั่นเอง) ดังนั้นในการ

ออกแบบระบบหากต้องการใช้หน่วยความจำภายนอกมากขึ้นเท่าใดก็จะทำให้เหลือพอร์ตที่จะนำมาใช้งานลดลง ในการออกแบบจริงจึงต้องพยายามลดขนาดหน่วยความจำภายนอกให้เหลือน้อยที่สุด

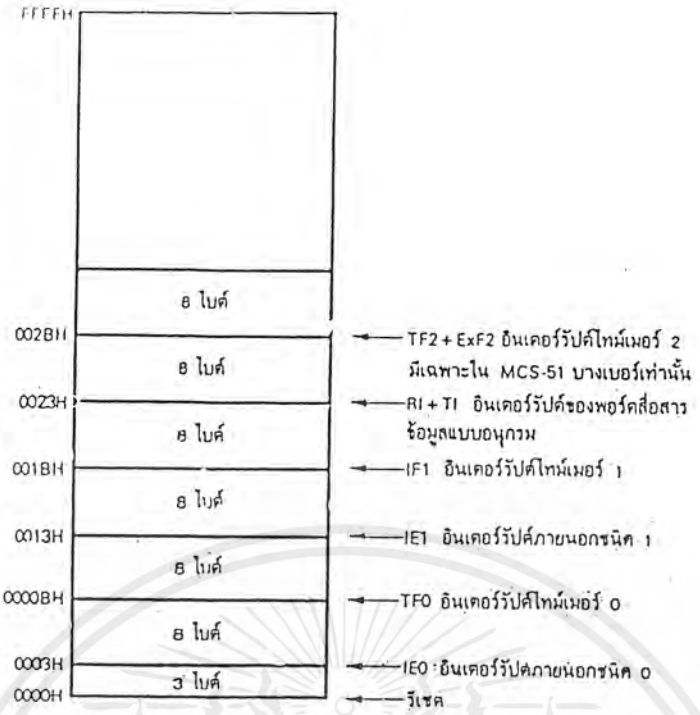
พอร์ต 3 ซึ่งมีขนาด 8 บิต นอกจากจะใช้ส่งสัญญาณสำหรับการอ่านหรือเขียนข้อมูลกับหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิปแล้ว ยังถูกนำมาใช้เป็นตัวรับสัญญาณอินเตอร์รัปต์ (INT0,INT1) สัญญาณอินพุตที่ต้องการนับสำหรับเคาน์เตอร์(T0,T1) รวมทั้งใช้ในการติดต่อสื่อสารข้อมูลแบบอนุกรมกับอุปกรณ์ภายนอก(รับและส่งข้อมูลผ่านขา RXD,TXD)

ในแต่ละพอร์ตที่ใช้เป็นอินพุตหรือเอาต์พุตสามารถกำหนดให้ทำงานได้อย่างอิสระโดยอาศัยการควบคุมการโปรแกรม ซึ่งสามารถควบคุมให้แต่ละพอร์ตถูกใช้เป็นอินพุตในช่วงเวลาหนึ่งและเป็นเอาต์พุตในอีกช่วงเวลาหนึ่งได้

ไทม์เมอร์/เคาน์เตอร์ ใน MCS-51 มีรีจิสเตอร์ใช้งานเฉพาะที่สามารถนับจำนวนสัญญาณนาฬิกาหรือแมชชีนไซเกลของวงจรออสซิลเลเตอร์ภายใน (ทำงานเป็นไทม์เมอร์) หรือนับจำนวนครั้งของการเปลี่ยนสถานะของสัญญาณภายนอก (นับจำนวนพัลส์ภายนอก) ที่ขา T0 ,T1 ของพอร์ต 3 (ทำงานเป็นเคาน์เตอร์) รีจิสเตอร์ที่ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์นี้มีขนาด 16 บิตจำนวน 2 ตัวคือ รีจิสเตอร์ไทม์เมอร์ 0 และรีจิสเตอร์ไทม์เมอร์ 1 ตามลำดับ(ในเบอร์ 8052 มีรีจิสเตอร์ไทม์เมอร์ 2 เพิ่มให้อีก 1 ตัว) เมื่อต้องการใช้ไทม์เมอร์ 0 หรือ 1 จะต้องโหลดค่าที่ต้องการนับไปไว้ที่รีจิสเตอร์ไทม์เมอร์ 0 หรือ 1 และเมื่อนับได้ครบตามจำนวนที่ตั้งไว้แล้วนั้นจะมีสัญญาณอินเตอร์รัปต์เพื่อบอกให้ซีพียูทราบ

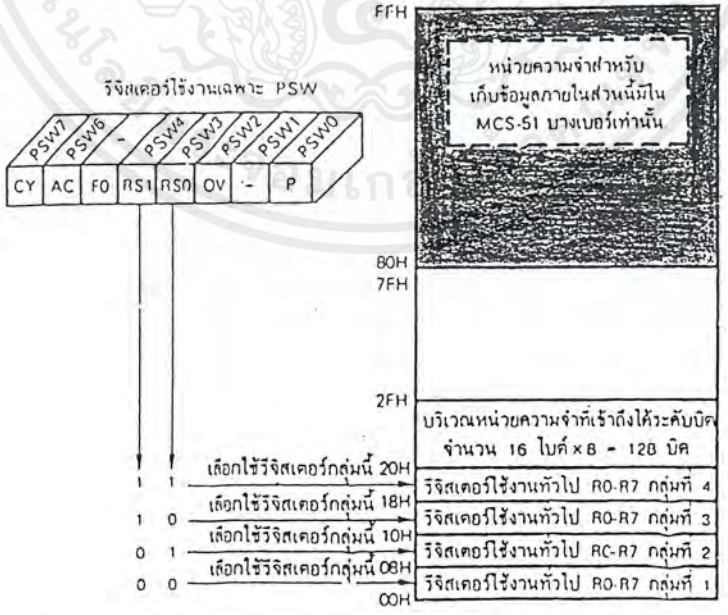
การควบคุมการทำงานของไทม์เมอร์หรือเคาน์เตอร์ สามารถควบคุมได้จากวงจรภายนอก(ควบคุมด้วยสัญญาที่ขา INT0,INT1) หรือควบคุมจากคำสั่งในโปรแกรมก็ได้ ดังนั้นรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์จะสามารถวัดช่วงห่างของเวลา วัดความกว้างของพัลส์ หรือนับจำนวนครั้งของเหตุการณ์ที่เกิดขึ้นภายนอกที่เปลี่ยนให้อยู่ในรูปของสัญญาณไฟฟ้าแล้ว รวมทั้งใช้กำเหนิดสัญญาณอินเตอร์รัปต์ที่มีคาบเวลาแน่นอนได้

พอร์ตสื่อสารข้อมูลแบบอนุกรม MCS-51 สามารถรับและส่งข้อมูลแบบอนุกรมได้โดยไม่ต้องพึ่งพาอุปกรณ์ภายนอกอื่นๆ ในด้านอัตราเร็วของการรับส่งข้อมูลก็สามารถกำหนดค่าได้ตามความต้องการของผู้ใช้ โดยสามารถเลือกอัตราความเร็วในการรับส่งข้อมูล (Baud Rate) มาตรฐานได้ตั้งแต่ 110,1.2K,2.4K,4.8K,9.6K, 19.2K,375K ตามมาตรฐานของ UART นอกจากนี้ยังสามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 รูปแบบ



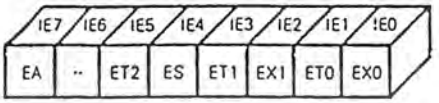
รูปที่ 2.21 แสดงการเลือกรีจิสเตอร์ใช้งานทั่วไป R0-R7 แต่ละกลุ่ม

โครงสร้างการอินเตอร์รัปต์ MCS-51 สามารถรับสัญญาณอินเตอร์รัปต์ได้ถึง 5 ชนิด โดยจะเป็นสัญญาณที่เกิดจากภายนอก 2 ชนิด และจากภายในชิปอีก 3 ชนิด เมื่อมีสัญญาณอินเตอร์รัปต์เกิดขึ้น MCS-51 จะละการทำงานที่กำลังทำอยู่และข้ามไปทำโปรแกรมบริการอินเตอร์รัปต์(Interrupt Service Routine) ที่อยู่ในหน่วยความจำตำแหน่งต่างๆ ขึ้นอยู่กับชนิดของสัญญาณอินเตอร์รัปต์ดังแสดงในรูปที่ 2.22



รูปที่ 2.22 แสดงตำแหน่งหน่วยความจำของโปรแกรมบริการอินเตอร์รัปต์แต่ละชนิดใน MCS-51

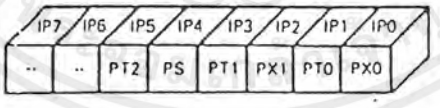
เราสามารถเลือกให้ซีพียูใน MCS-51 ถูกอินเตอร์รัปต์โดยสัญญาณอินเตอร์รัปต์ที่เกิดขึ้นได้ โดยการกำหนดค่าในรีจิสเตอร์ใช้งานเฉพาะ IE นอกจากนี้ยังสามารถควบคุมลำดับความสำคัญในการตอบสนองได้โดยการใช้รีจิสเตอร์ใช้งานเฉพาะ IP



บิต	ชื่อบิต	
IE7	EA	ใช้ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ทั้งหมด 0 : MCS-51 จะไม่ตอบสนองต่อสัญญาณอินเตอร์รัปต์ใด ๆ ทั้งสิ้น 1 : การตอบสนองต่อสัญญาณอินเตอร์รัปต์แต่ละชนิดจะถูกควบคุมโดยตรงจากบิตที่ทำหน้าที่ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ซึ่งอยู่ในรีจิสเตอร์นี้เช่นกัน
IE6	--	ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ๆ ในอนาคต)
IE5	ET2	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 2 เมื่อเกิด overflow (มีใช้เฉพาะ MCS-51 บางเบอร์ที่มีไทม์เมอร์ 2 เช่น 8052)
IE4	ES	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม
IE3	ET1	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 1 เมื่อเกิด overflow
IE2	EX1	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1
IE1	ET0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0 เมื่อเกิด overflow
IE0	EX0	ควบคุมการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 0

หมายเหตุ ถ้าบิตที่ควบคุมการตอบสนองสัญญาณต่ออินเตอร์รัปต์แต่ละบิต มีค่าเป็น 1 หมายถึง อนุญาตให้ MCS-51 ตอบสนองต่อสัญญาณอินเตอร์รัปต์ได้ หากมีค่าเป็น 0 หมายถึงไม่ให้ MCS-51 ตอบสนองต่อสัญญาณอินเตอร์รัปต์ที่เกิดขึ้น

รูปที่2.23 รีจิสเตอร์ใช้งานเฉพาะ IE



บิต	ชื่อบิต	
IP7	--	ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคต)
IP6	--	ไม่ถูกกำหนดการใช้งาน (สำรองไว้ใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคต)
IP5	PT2	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 2
IP4	PS	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของพอร์ตสื่อสารอนุกรม
IP3	PT1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 1
IP2	PX1	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 1
IP1	PT0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ของไทม์เมอร์ 0
IP0	PX0	กำหนดลำดับความสำคัญในการตอบสนองต่อสัญญาณอินเตอร์รัปต์ภายนอกชนิด 0

หมายเหตุ ค่าในบิตที่กำหนดลำดับความสำคัญของสัญญาณอินเตอร์รัปต์แต่ละชนิด หากเป็น 1 หมายถึง กำหนดให้มีลำดับความสำคัญสูง หากเป็น 0 หมายถึงกำหนดให้มีลำดับความสำคัญต่ำ

รูปที่ 2.24 รีจิสเตอร์ใช้งานเฉพาะ IP

2.3.3 วงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลและโพเทนชิโอมิเตอร์

โพเทนชิโอมิเตอร์ (potentiometer)

การตรวจวัดตำแหน่งของวัตถุเป็นสิ่งจำเป็นและยากที่จะหลีกเลี่ยงได้ ตัวอย่างเช่น การตรวจจับตำแหน่งของชิ้นงานบนสายพาน การเคลื่อนที่ของอุปกรณ์ในระบบไฮดรอลิกส์ ระบบนิวแมติกส์ หรือมอเตอร์ไฟฟ้า โพเทนชิโอมิเตอร์ เป็นทรานดิวเซอร์ในการตรวจวัดการขจัด (Displacement) อย่างง่ายที่สุด โครงสร้างโดยทั่วไปจะประกอบด้วย ตัวต้านทานซึ่งทำมาจาก คาร์บอน หรือ ขดลวด และหน้าสัมผัสที่สามารถเลื่อนไปมาบนตัวต้านทานได้ การวัดการขจัดทำได้โดยการต่อแกนที่มาจากหน้าสัมผัสเข้ากับวัตถุที่ต้องการตรวจวัด ตำแหน่งแกนเลื่อนของโพเทนชิโอมิเตอร์จะเป็นผลมาจากการเปลี่ยนตำแหน่งของหุ่นยนต์ โดยจะแปลงไปเป็นสัญญาณไฟฟ้าโดยใช้หลักการแบ่งแรงดัน ซึ่งแบ่งออกได้ 2 แบบคือ แบบเชิงมุม (angular) และแบบเชิงเส้น (linear) ดังแสดงในรูปที่ 2.25 ค่าของแรงดันเอาต์พุต (Vo) จะขึ้นอยู่กับตำแหน่งของแกนเลื่อน สามารถคำนวณหาได้ตามสมการซึ่งเป็นลักษณะของวงจรแบ่งแรงดัน (Voltage Divider)

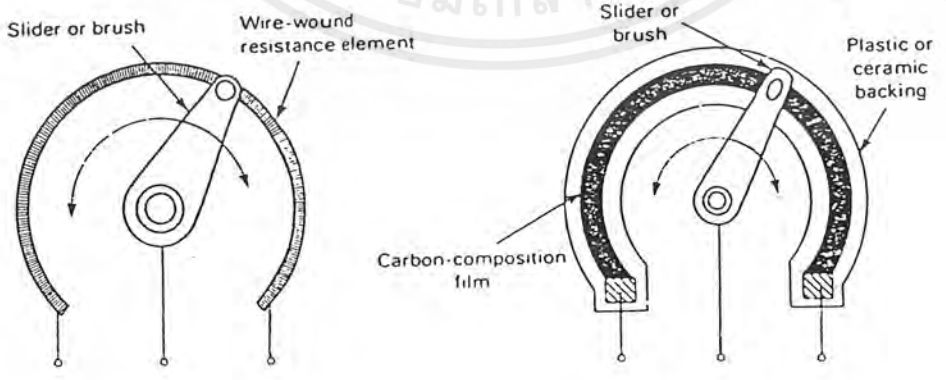
$$Vo = (R_2 * V_T) / (R_1 + R_2)$$

โดยที่ Vo = แรงดันที่จะส่งไปแปลงเพื่อบอกตำแหน่ง

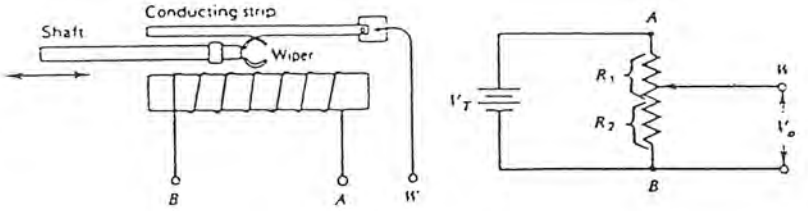
V_T = แรงดันที่จ่ายให้ทั้งหมด

R_1 และ R_2 = ค่าความต้านทานของโพเทนชิโอมิเตอร์

ชนิดของแถบความต้านทานที่ใช้ทำโพเทนชิโอมิเตอร์มีด้วยกันหลายชนิดแตกต่างกันไป เช่น ชนิดฟิล์มคาร์บอน (Carbon film) ซึ่งมีราคาถูก แต่ถ้าเป็นชนิดฟิล์มเซอร์เมต (Cermet film) ก็จะแพงขึ้นนิดหน่อย แต่จะลดสัญญาณรบกวนบางอย่างได้ ส่วนชนิดไวร์วาวด์ (Wire Wound) นั้นสามารถทนกำลังไฟฟ้าได้สูง



รูปที่ 2.25 แสดงตัวอย่างของโพเทนชิโอมิเตอร์



รูปที่ 2.26 แสดงโพเทนชิโอมิเตอร์ที่เคลื่อนที่ในลักษณะเส้นตรงพร้อมสัญญลักษณ์การต่อใช้งาน

ข้อดีของโพเทนชิโอมิเตอร์คือ มีขนาดเล็กและราคาถูก นิยมนำมาใช้งานมากกับหุ่นยนต์ขนาดเล็ก แต่ข้อเสียคือ หน้าสัมผัสของแถบความต้านทานเกิดการสึกหรอได้ง่าย ทำให้อายุการใช้งานสั้นและการติดต่อกับคอมพิวเตอร์ต้องมีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล

วงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล (ADC)

วงจรนี้ทำการแปลงค่าแรงดันซึ่งได้มาจากโพเทนชิโอมิเตอร์ซึ่งเป็นสัญญาณอะนาลอกเป็นดิจิตอล แล้วส่งให้กับไมโครคอนโทรลเลอร์ต่อไป ซึ่งมีด้วยกันหลายชนิด เช่น วงจรเปรียบเทียบขนาน วงจรเอทูดีที่ใช้การอินทิเกรต วงจรเปลี่ยนสัญญาณเอทูดีที่ใช้วงจรนับและวงจรดีทูเอ (Digital to Analog Converter) ประกอบกัน และวงจรเอทูดีแบบใช้การประมาณค่า เป็นต้น

ในที่นี้ขอกล่าวถึงเฉพาะวงจรเปลี่ยนสัญญาณอะนาลอกเป็นดิจิตอลแบบใช้การประมาณค่า (Successive Approximation A/D Converter) เนื่องจากไอซีเบอร์ 0808 ที่นำมาใช้ในวงจรอาศัยหลักการทำงานแบบนี้ในการแปลงสัญญาณ

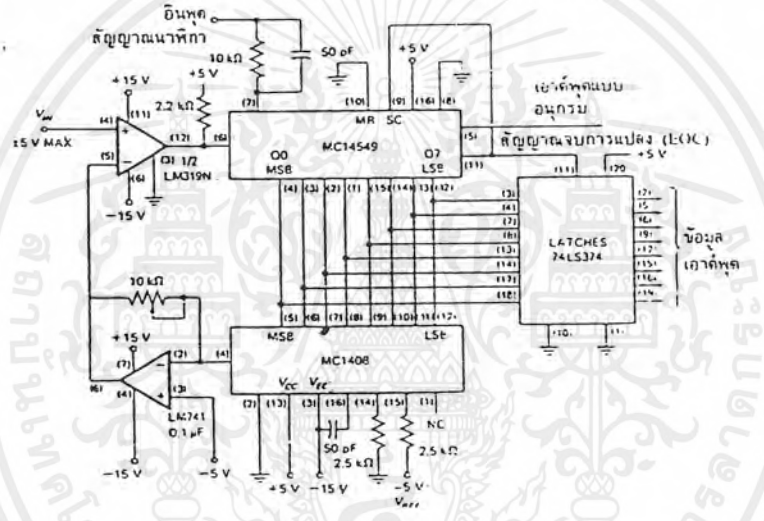
วงจรเปลี่ยนสัญญาณอะนาลอกเป็นดิจิตอลแบบใช้การประมาณค่า

เป็นวงจรที่มีความละเอียดสูงขนาด n บิตซึ่งสามารถกำหนดได้จากสัญญาณนาฬิกา n ลูก เช่น ถ้าต้องการความละเอียดขนาด 8 บิต ก็จะต้องการพัลส์นาฬิกาเพียง 8 ลูก หัวใจสำคัญของวงจรคือ Successive Approximation Register (SAR) ที่มีการทำงานดังนี้

เมื่อเริ่มทำงานพัลส์ลูกแรกจะทำการส่งบิตที่มีนัยสำคัญสูงสุดไปยังดีทูเอเบอร์ MC1408 โดย SAR จะรอสัญญาณจากวงจรเปรียบเทียบ LM319 ซึ่งทำการเปรียบเทียบค่าเอาต์พุตของวงจรดีทูเอกับแรงดันอินพุต Vin ถ้าเอาต์พุตของวงจรเปรียบเทียบมีค่าเป็นระดับสูง นั่นคือเอาต์พุตของเอทูดีต่ำกว่า Vin SAR จะทำการเก็บค่าบิตนี้ แต่ถ้าในทางกลับกัน SARจะทำการรีเซตบิตที่มีนัยสำคัญสูงสุดนี้ พัลส์ลูกต่อไปจะทำงานเช่นเดียวกันโดยค่าที่ได้เป็นของบิตที่มีนัยสำคัญรองลงมาต่อไปเรื่อยๆจนถึงค่าของบิตที่มีนัยสำคัญต่ำที่สุด

เมื่อทำการแปลงเสร็จทั้งหมดแล้ว SAR ก็จะส่งสัญญาณ EOC (End of Conversion) ออกไป สัญญาณนี้เป็นตัวบอกว่าสัญญาณเอาต์พุตที่ขนานกันทุกเส้นมีข้อมูลแล้ว ถ้าสัญญาณ EOC ถูกต่อไปยังอินพุตจุดเริ่มต้นของการเปลี่ยนสัญญาณการแปลงก็จะเกิดอย่างต่อเนื่อง วงจรจ่ายแรงดันอินพุต –5 ถึง +5 โวลต์ การต่อขาลบของออปแอมป์เข้ากับไฟ –5 โวลต์เพื่อทำการยกระดับ –5 ถึง +5 โวลต์ด้วย 0 ถึง 10 โวลต์ สัญญาณไฟสลับรูปซายน์จึงสามารถป้อนให้วงจรได้

ข้อดีของวงจรนี้คือ มีความเร็วสูงและมีความละเอียดสูง ซึ่งทำให้มีการใช้กันอย่างแพร่หลาย ไอซีที่เลือกใช้ได้รวมเอาวงจรทั้งหมดนี้ไว้ภายในแล้ว

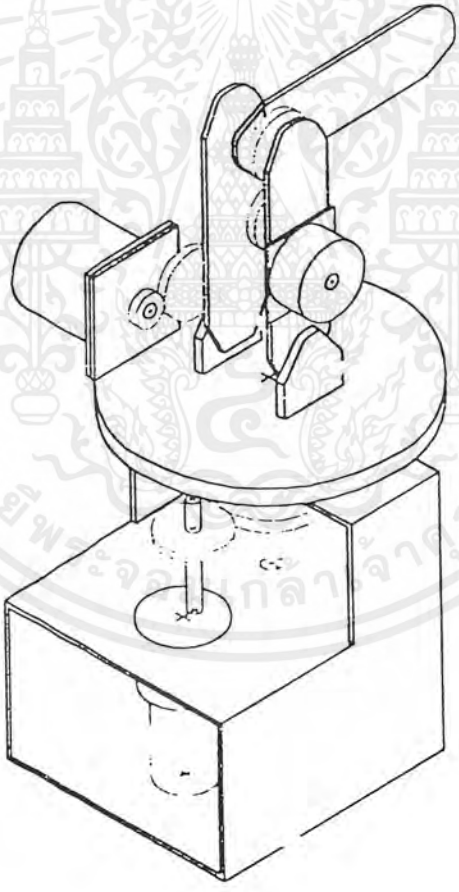


รูปที่ 2.27 แสดงวงจรเปลี่ยนสัญญาณเอทูดีแบบการประมาณค่า

บทที่ 3
การออกแบบส่วนต่างๆของแขนกล

โครงงานชิ้นนี้ได้ทำการสร้างแขนกลขึ้น โดยมีขอบเขตของโครงงานอยู่ที่เป็นแขนกลซึ่งมีชิ้นส่วนเคลื่อนที่ได้ 3 ชิ้นแบ่งเป็น 2 ลักษณะ คือ

1. หมุนรอบแกนแนวตั้ง จำนวน 1 ชิ้น(ฐาน)
2. หมุนรอบแกนแนวนอน จำนวน 2 ชิ้น



รูปที่ 3.1 แสดงภาพแขนกล

การควบคุมการเคลื่อนที่แบ่งเป็น 3 ส่วน คือ

- ส่วนของมอเตอร์ ในที่นี้ใช้มอเตอร์ทั้งหมด 3 ตัว ส่วนฐานใช้เป็นมอเตอร์กระแสตรง และส่วนบนใช้เป็นสเต็ปปิ้งมอเตอร์อีก 2 ตัว

- ส่วนวงจรขับมอเตอร์ ทำหน้าที่ขยายสัญญาณที่ออกมาจากไมโครคอนโทรลเลอร์รวมทั้งควบคุมกระแสและทิศทางการไหลของกระแสที่จะนำไปให้กับมอเตอร์แต่ละตัวเมื่อได้รับคำสั่งให้เคลื่อนที่ในแต่ละแบบ

- ส่วนของไมโครคอนโทรลเลอร์และส่วนของการระบุตำแหน่ง ซึ่งไมโครคอนโทรลเลอร์เป็นตัวหัวใจสำคัญในการควบคุมให้ส่วนต่างๆเคลื่อนที่ไปตามที่ได้รับคำสั่งมาจากผู้ใช้ผ่านทางคอมพิวเตอร์ รวมทั้งการระบุตำแหน่งของแขนกล ซึ่งคอมพิวเตอร์จะมีหน้าจอโปรแกรมเขียนด้วยภาษาวิชวลเบสิกเป็นตัวติดต่อสื่อสารกับผู้ใช้ แล้วส่งสัญญาณจากคอมพิวเตอร์ไปให้ไมโครคอนโทรลเลอร์

ส่วนประกอบต่างๆ มีรายละเอียดดังนี้

3.1 มอเตอร์ ในที่นี้ใช้เป็นมอเตอร์ 2 ชนิดตามความเหมาะสมของงานในแต่ละส่วนดังนี้

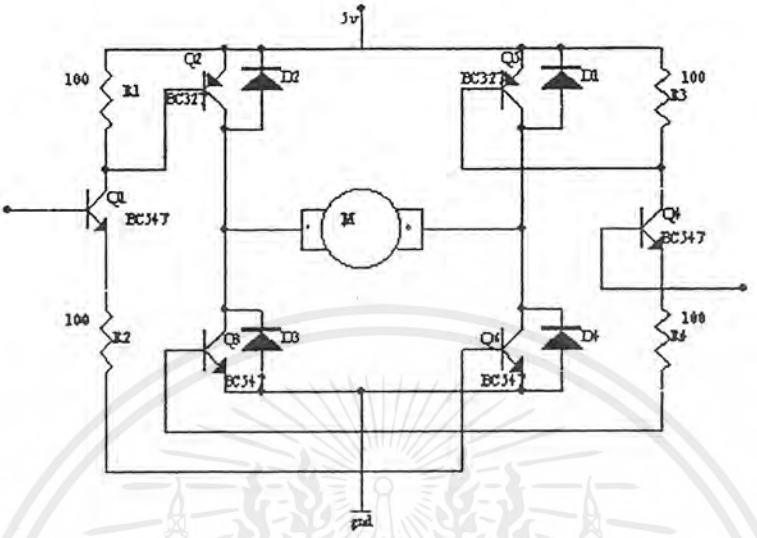
3.1.1 ส่วนฐาน ใช้มอเตอร์กระแสตรงที่มีชุดเกียร์ต่ออยู่ภายในสำเร็จรูปคือ มีการทดรอบให้ความเร็วน้อยลง แต่แรงบิดเพิ่มขึ้นทำให้หมุนช้าลง นอกจากนี้เมื่อหยุดจ่ายไฟแล้วมอเตอร์จะหยุดอยู่ ณ ตำแหน่งสุดท้ายโดยอัตโนมัติ เหตุผลที่เลือกใช้มอเตอร์กระแสตรงมาเป็นฐานเพราะต้องการให้รับภาระได้มาก ซึ่งหากจะเลือกใช้สเต็ปปิ้งมอเตอร์ที่สามารถรับภาระได้มากขนาดนี้จะมีขนาดใหญ่มากทำให้เปลืองพื้นที่ รวมทั้งยังต้องออกแบบวิธีที่จะทำให้หยุดอีกด้วย

3.1.2 แขน 2 ชิ้นบน ใช้เป็นสเต็ปปิ้งมอเตอร์เนื่องจากต้องการมอเตอร์ซึ่งมีขนาดเล็ก ความเร็วไม่มากนัก และที่สำคัญคือสามารถควบคุมตำแหน่งให้แม่นยำได้ ซึ่งหากเลือกใช้มอเตอร์กระแสตรงตัวเล็กจะไม่มีการทดรอบภายในทำให้มีความเร็วสูงจึงควบคุมตำแหน่งได้ยาก และหากต้องการลดความเร็วก็ต้องออกแบบการทดรอบเอง รวมทั้งต้องออกแบบวิธีที่จะทำให้หยุดอีกด้วย การใช้สเต็ปปิ้งมอเตอร์นี้สามารถทำการควบคุมเป็นแบบระบบเปิด(Open loop)ได้ ทำให้ลดความยุ่งยากลง ซึ่งต่างกับมอเตอร์กระแสตรงที่ต้องควบคุมการทำงานแบบระบบปิด(Close loop)

ในที่นี้เลือกใช้เป็นสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลักแตนซ์เนื่องจากเป็นแบบที่สามารถทำให้มีขนาดสเต็ปเล็กได้ โดยที่ตัวมอเตอร์ก็มีขนาดเล็กด้วยดีกว่าแบบแม่เหล็กถาวร และมีแรงเฉื่อยทางแมคคานิคส์น้อยกว่าแบบไฮบริดจ์

3.2 วงจรขับมอเตอร์ มี 2 แบบคือ วงจรขับมอเตอร์กระแสตรงและวงจรขับสเต็ปปิ้งมอเตอร์

3.2.1 วงจรขับมอเตอร์กระแสตรง มีลักษณะดังรูป



รูปที่ 3.2 แสดงวงจรขับมอเตอร์กระแสตรง

มอเตอร์กระแสตรงที่ใช้มีขนาด 30 โวลต์ 80 มิลลิแอมป์ แต่ในโครงงานนี้ทำการจ่ายแรงดันไฟฟ้าให้แก่มอเตอร์แค่ 5 โวลต์ เพื่อให้มอเตอร์หมุนได้ช้าลงและมีแรงบิดเพิ่มขึ้น โดยสามารถคำนวณค่าอุปกรณ์ต่างๆ ที่ใช้ในวงจรได้ดังนี้

$$V_{BE2} = 0.7\ V$$

เนื่องจากเลือกใช้ R ขนาด 100 โอมห์ $I_{R1} = I_{C1}$

$$= 0.7 / 100$$

$$\therefore\ I_{C1} = 7\ mA$$

$(\beta_1 = 100)$ $\therefore\ I_{B1} = I_{C1} / \beta_1 = 70\ \mu A$

จากวงจรพบว่า $I_{C1} = I_{E1}$

Imotor = 50 mA $\therefore$ Ic2 = 50 mA

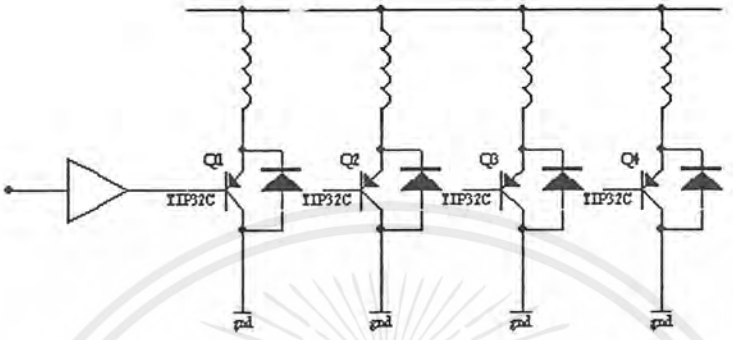
$(\beta_2 = 63)$ $\therefore\ I_{B2} = I_{C2} / \beta_2 = 0.8\ mA$

ในที่นี้เลือกใช้ - ฟรีวิลลิ่งไดโอดเป็น 1N4004 มีขนาด 10A 100V ทั้ง 4 ตัว

- ทรานซิสเตอร์ตัวที่ 1, 3 และ 4 เป็น BC547 มี $\beta = 100$
- ทรานซิสเตอร์ตัวที่ 2 เป็น BC547 มี $\beta = 63$
- ความต้านทานทุกตัวเลือกใช้ขนาด 100 โอมห์

โดยวงจรนี้เป็นวงจรแบบ H บริดจ์ ซึ่ง 2 ข้างของวงจรมีลักษณะสมมาตรกัน ฉะนั้นค่าต่างๆของอีกซีกหนึ่งของวงจรจะมีค่าเท่ากันกับของซีกนี้จึงขอแสดงแค่ข้างเดียว

3.2.2 วงจรขับสเต็ปปิ้งมอเตอร์ เป็นวงจรแบบยูนิโพล่าร์ มีลักษณะดังรูป



รูปที่ 3.3 แสดงวงจรขับสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ที่ใช้มี 2 ขนาดคือ 12 โวลต์ 1 แอมป์ (สำหรับแขนส่วนบนสุด) และ 24 โวลต์ 0.3 แอมป์ (สำหรับแขนส่วนล่าง) ซึ่งสามารถคำนวณค่าอุปกรณ์ต่างๆ ที่ใช้ในวงจรได้ดังนี้ เนื่องจากวงจรขับของทั้ง 4 เฟสมีลักษณะเหมือนกัน ในที่นี้จะขอแสดงการคำนวณค่าอุปกรณ์แค่เฟสเดียวเท่านั้น

<u>มอเตอร์ขนาด 12 โวลต์ 1 แอมป์</u>

Ic มีค่าเท่ากับ I ของมอเตอร์ $\therefore$ Ic = 1 A

(β = 25) $\therefore$ I_B = Ic / β = 400 mA

วงจรนี้เลือกใช้ - ทรานซิสเตอร์เบอร์ TIP 32 C ที่มีค่า β = 25

- ใช้บัฟเฟอร์เบอร์ 74LS244 ซึ่งมีค่า Vcc = 5 V , I_{OL} = 24 mA และ I_{OH} = 15 mA

<u>มอเตอร์ขนาด 24 โวลต์ 0.6 แอมป์</u>

Ic มีค่าเท่ากับ I ของมอเตอร์ $\therefore$ Ic = 0.6 A

(β = 25) $\therefore$ I_B = Ic / β = 12 mA

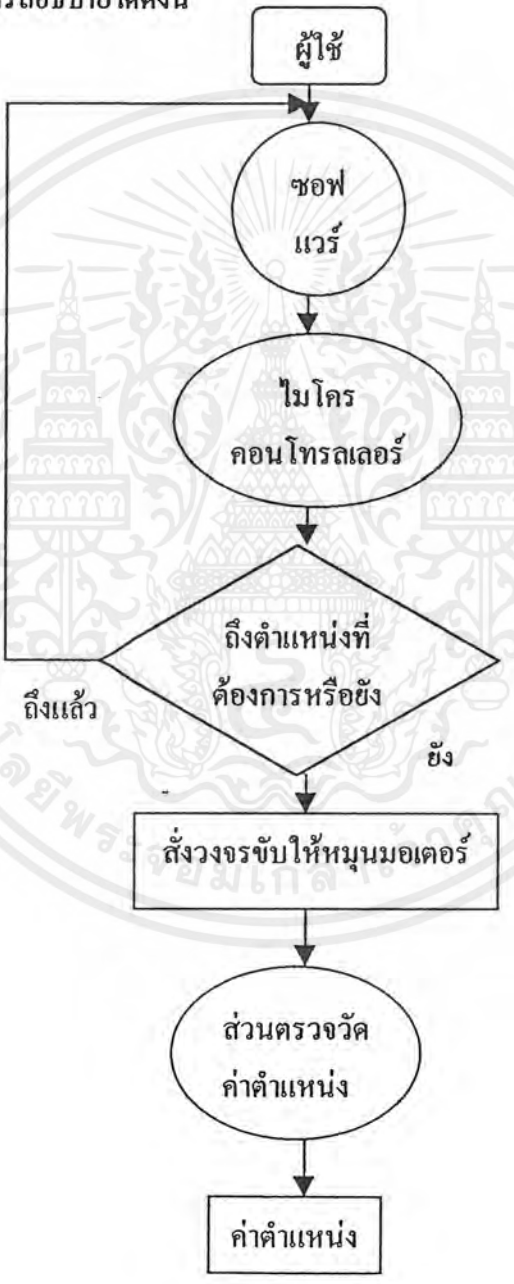
วงจรนี้เลือกใช้ - ทรานซิสเตอร์เบอร์ TIP 32 C ที่มีค่า β = 25

- ใช้บัฟเฟอร์เบอร์ 74LS244 ซึ่งมีค่า Vcc = 5 V , I_{OL} = 24 mA และ I_{OH} = 15 mA

3.3 ส่วนของการควบคุมและแสดงตำแหน่งของแขนกล

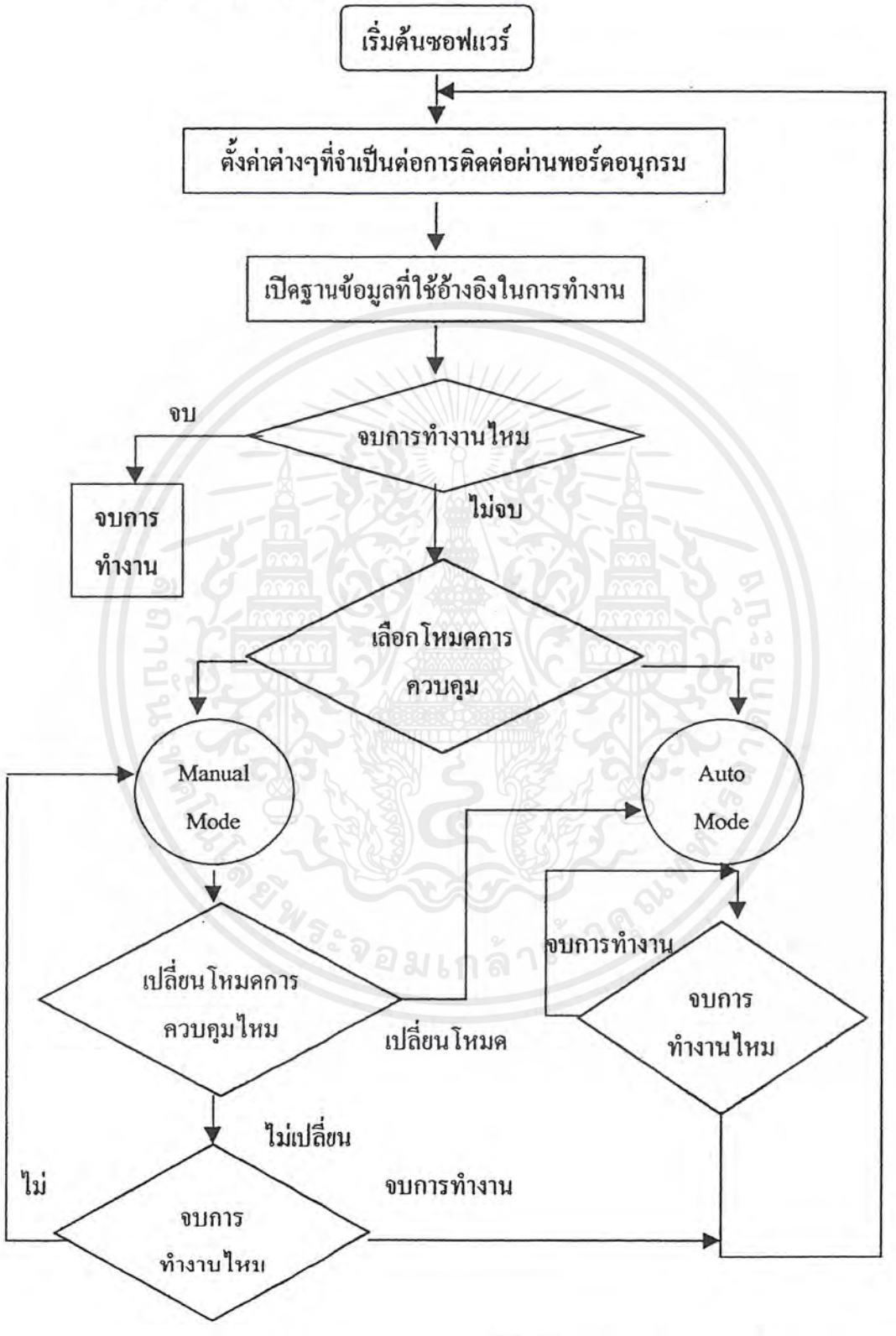
ส่วนนี้ประกอบด้วย 3 ส่วนหลักๆ คือ ส่วนของไมโครคอนโทรลเลอร์ซึ่งทำหน้าที่ควบคุมให้แขนกลเคลื่อนที่ไปตามที่ต้องการทั้งในแบบที่ผู้ใช้ควบคุมเอง และแบบอัตโนมัติ โดยต้องอาศัยการทำงานร่วมกับวงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลที่จะรับสัญญาณมา

จากโพเทนชิโอมิเตอร์ที่ติดอยู่กับมอเตอร์แต่ละตัวมาแปลงเป็นสัญญาณดิจิตอลส่งให้ไมโครคอนโทรลเลอร์ทราบตำแหน่งของมอเตอร์แต่ละตัวซึ่งก็คือตำแหน่งของแขนกลแต่ละชิ้น และส่วนสุดท้ายคือการทำงานร่วมกับซอฟแวร์ซึ่งเป็นโปรแกรมที่เขียนขึ้นโดยภาษาวิชวลเบสิก ซอฟแวร์นี้จะทำหน้าที่ในการติดต่อกับผู้ใช้โดยตรง ผู้ใช้จะใช้โปรแกรมนี้ในการควบคุมให้แขนกลเคลื่อนที่ และผู้ใช้ก็สามารถทราบตำแหน่งของแขนกลได้จากภาพ 2 มิติที่แสดงไว้บนหน้าจอควบคุม การทำงานอย่างละเอียดสามารถอธิบายได้ดังนี้



รูปที่ 3.4 แสดงแผนภาพการทำงานของโครงงานโดยรวม

3.3.1 ส่วนของซอฟแวร์



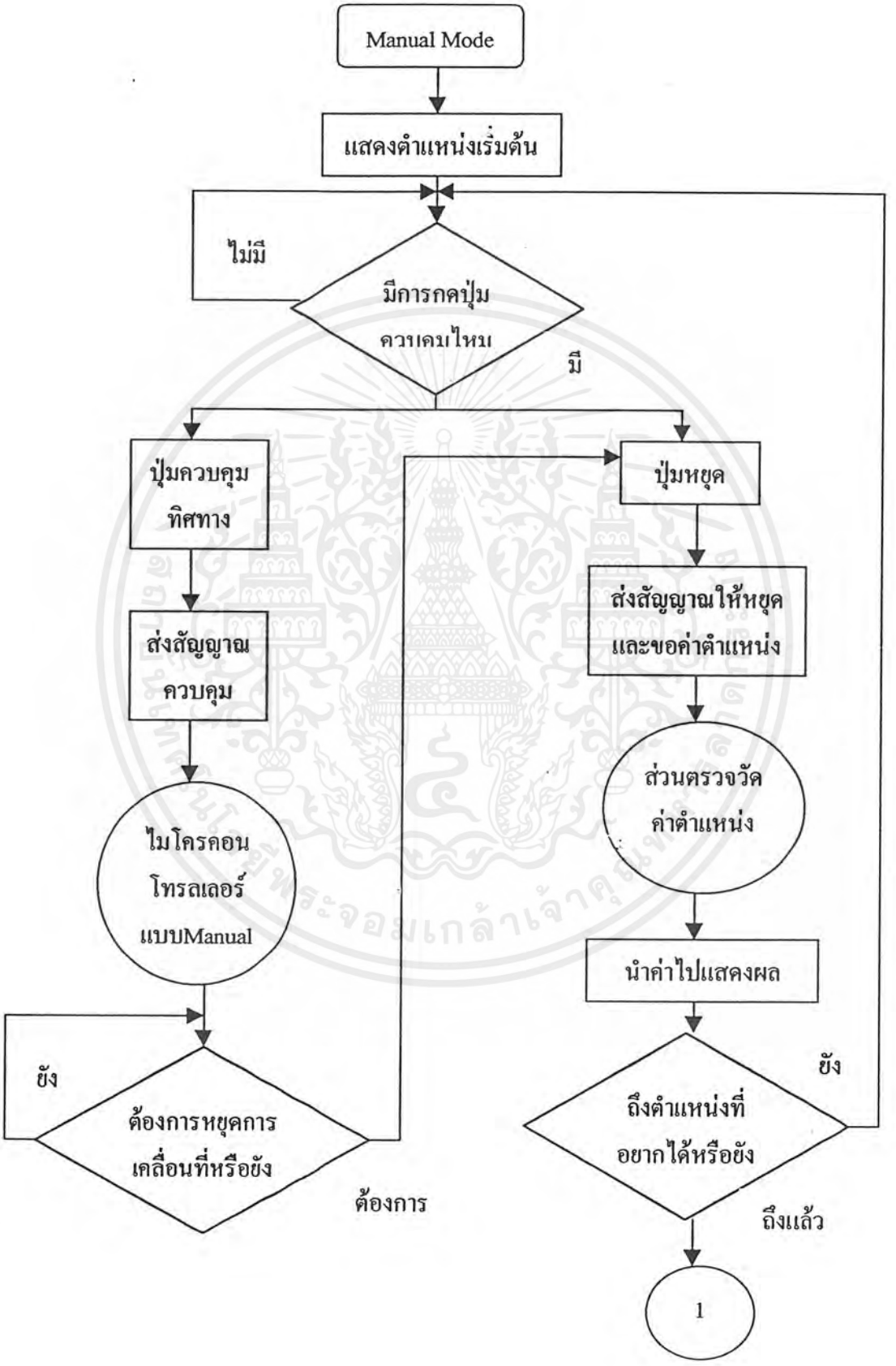
รูปที่ 3.5 แสดงแผนภาพการทำงานโดยรวมของซอฟแวร์ทั้งหมด

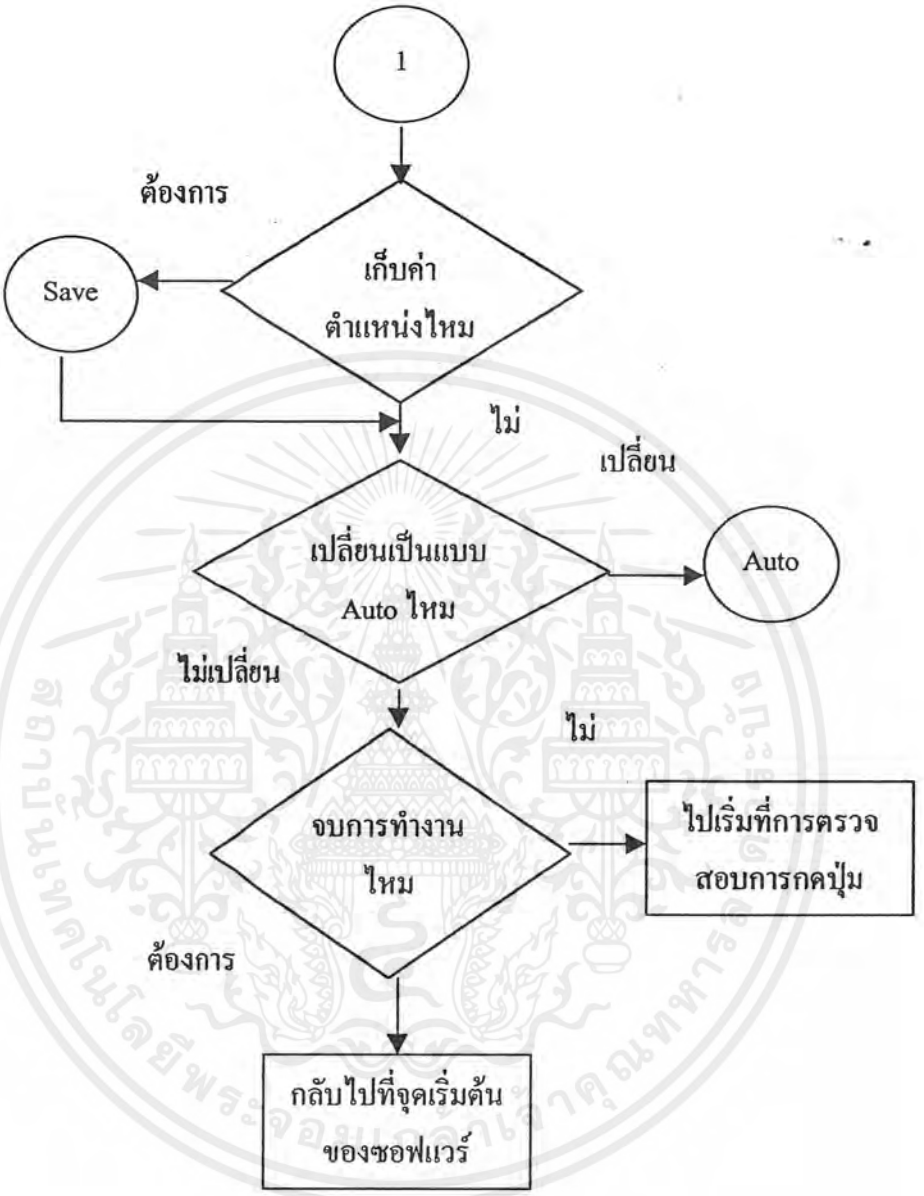
โปรแกรมที่เขียนขึ้นนี้เน้นประโยชน์หลักเพื่อให้ผู้ใช้สามารถทำการควบคุมแขนกลให้เคลื่อนที่ไปตามที่ตนเองต้องการได้โดยง่ายและสามารถควบคุมตำแหน่งของแขนกลได้แม้ผู้ใช้จะอยู่ในบริเวณที่มองไม่เห็นแขนกล โดยอาศัยดูภาพตำแหน่งปัจจุบันของแขนจากหน้าจอควบคุมของซอฟแวร์ ซึ่งมีทั้งเป็นรูปภาพ 3 รูปคือ ภาพฉายบนระนาบ X-Y, ภาพฉายบนระนาบ Z-Y และภาพแสดงมุมที่หมุนไปของมอเตอร์ตัวที่ฐาน นอกจากนี้ยังมีการแสดงผลเป็นตัวเลข 3 ค่า คือ ค่าระยะห่างจากฐาน , ความสูงจากฐาน และค่ามุมที่ฐานหมุนไปอีกด้วย การทำงานของส่วนนี้แบ่งออกเป็น 2 อย่างคือ การควบคุมและการแสดงตำแหน่งของแขนกล ทั้งตอนเริ่มใช้โปรแกรมโดยการแสดงตำแหน่งเริ่มต้นของแขนกล และระหว่างรวมทั้งหลังจากที่แขนกลเคลื่อนที่ไปแล้ว

จากรูปที่ 3.5 โปรแกรมจะเริ่มต้นการทำงานด้วยการตั้งค่าต่าง ๆ ที่จำเป็นต่อการติดต่อส่งรับข้อมูลระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ในที่นี้ใช้อัตราเร็วในการส่งข้อมูลเท่ากับ9600 BPS (Bits Per Second) , ไม่มีพาริตี้บิต (Parity Bit) ส่งข้อมูลเป็นแบบอนุกรมชุดละ 8 บิตกับบิตเริ่มต้นอีก 1 บิต และทำการเปิดฐานข้อมูลที่ใช้อ้างอิงในโปรแกรมนี้ ผู้ใช้ทำการเลือกโหมดการควบคุมการเคลื่อนที่ของแขนกลที่ต้องการซึ่งมีด้วยกัน 2 แบบคือ การควบคุมแบบผู้ใช้ควบคุมเองและการควบคุมแบบอัตโนมัติ หากผู้ใช้เลือกการควบคุมแบบแรกจะส่งสัญญาณ 08H ออกไป พอทำงานเสร็จก็สามารถเลือกเปลี่ยนเป็นโหมดที่ 2 ได้โดยไม่ต้องกลับไปเริ่มต้นที่หน้าจอหลักใหม่ แต่หากจะไม่เปลี่ยนรูปแบบการคำนวณก็สามารถใช้แบบเดิมต่อหรือจบการทำงานก็ได้ ส่วนถ้าผู้ใช้เลือกการควบคุมแบบอัตโนมัติจะส่งสัญญาณ 03H ออกไปจะสามารถควบคุมได้แค่ 1 ครั้งและหากต้องการใช้งานแบบอัตโนมัติอีกหรือต้องการเปลี่ยนเป็นการควบคุมในแบบที่ผู้ใช้ควบคุมเองก็ต้องเลือกจบการทำงานเพื่อกลับไปยังหน้าจอหลักอีกครั้งแล้วเลือกโหมดการควบคุมใหม่อีกครั้ง วนไปเช่นนี้จนกว่าพอกลับมาหน้าจอหลักแล้วจะเลือกกดปุ่ม “Exit” เพื่อปิดโปรแกรมทั้งหมด รายละเอียดการควบคุมการเคลื่อนที่ของแขนกลสามารถอธิบายได้ดังนี้

การควบคุมแบบที่ผู้ใช้ควบคุมเอง (Manual Mode)

เมื่อผู้ใช้ทำการเลือกรูปแบบการควบคุมเป็นแบบนี้แล้ว เริ่มแรกโปรแกรมจะทำการแสดงตำแหน่งปัจจุบันของแขนกลให้เห็น จากนั้นจะทำการตรวจสอบว่ามีการกดปุ่มควบคุมหรือยัง ถ้ามีแล้วก็จะดูว่าเป็นปุ่มอะไร ซึ่งปุ่มในการควบคุมแบ่งออกเป็น 2 ประเภทคือ ปุ่มควบคุมทิศทางการเคลื่อนที่ซึ่งมีด้วยกันทั้งหมด 6 ปุ่มได้แก่ ปุ่มควบคุมให้แขนกลหมุนรอบแกน Y ไปทางซ้าย (ทวนเข็มนาฬิกา) หรือไปทางขวามือ (ตามเข็มนาฬิกา) โดยทั้ง 2 ปุ่มนี้จะทำหน้าที่ควบคุมการหมุนของมอเตอร์กระแสตรงนั่นเอง , ปุ่มควบคุมแขน 2 ชิ้นบนจะเหมือนกัน 2 ชุด โดยชุดหนึ่งให้ควบคุมสเตปปิ้งมอเตอร์ตัวล่างและอีกชุดใช้ควบคุมสเตปปิ้งมอเตอร์ตัวบนให้ยกขึ้นหรือลง ซึ่งจะเป็นการควบคุมระยะห่างจากฐานและความสูงจากฐานนั่นเอง



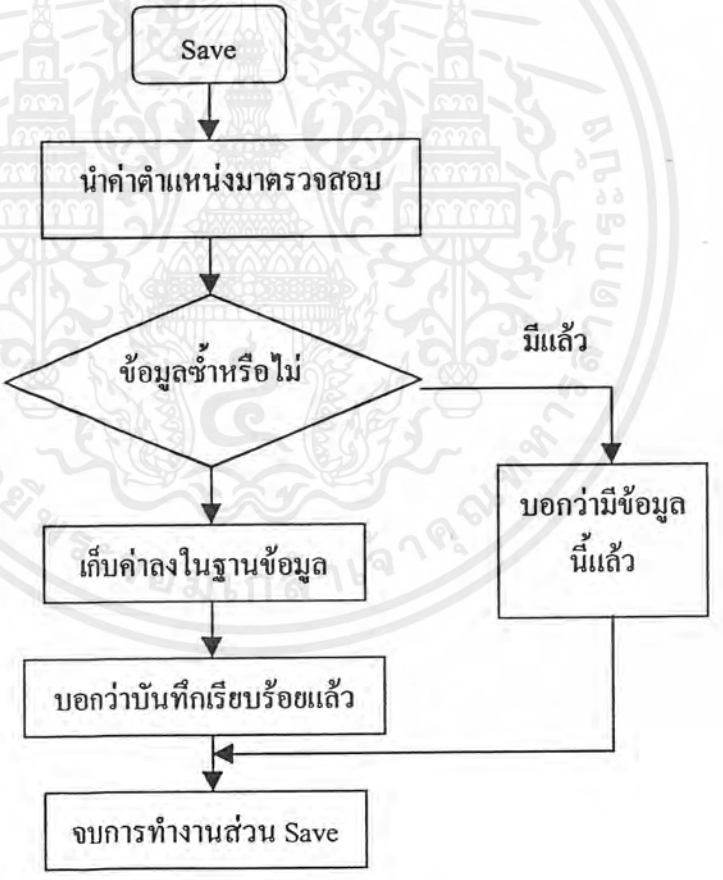


รูปที่ 3.6 โฟลชาร์ทแสดงการทำงานแบบผู้ใช้ควบคุมเอง

ทั้ง 6 ปุ่มจะมีสัญญาณเฉพาะแตกต่างกันไป ดังนั้นเมื่อเรากดปุ่มใดก็จะมีสัญญาณเฉพาะของปุ่มนั้นส่งออกไปให้ไมโครคอนโทรลเลอร์ทราบว่าผู้ใช้ต้องการควบคุมให้แขนกลเคลื่อนที่ไปทางใด หากต้องการหยุดการเคลื่อนที่ในทิศทางนั้นแล้วก็ให้ผู้ใช้กดปุ่มหยุด ซึ่งจะมีขั้นตอนการทำงานคือ โปรแกรมจะส่งค่า FFH เพื่อบอกให้ไมโครคอนโทรลเลอร์ทราบว่าต้องการ

หยุดการเคลื่อนที่ในทิศทางนี้ นอกจากนี้สัญญาณ FFH ยังจะเป็นตัวบอกให้ไมโครคอนโทรลเลอร์ทราบว่าให้ส่งค่าตำแหน่งกลับมายังคอมพิวเตอร์ เพื่อให้โปรแกรมนำค่าตำแหน่งของมอเตอร์แต่ละตัวไปคำนวณแล้วแสดงออกมาเป็นรูปภาพและตัวเลขให้ผู้ใช้ได้ทราบตำแหน่งของแขนกล ซึ่งหากตำแหน่งนี้ยังไม่ใช่ตำแหน่งที่ผู้ใช้ต้องการก็สามารถกดปุ่มควบคุมได้อีกวนไปเช่นนี้เรื่อยๆ จนกระทั่งแขนกลเคลื่อนที่ไปถึงยังตำแหน่งที่ต้องการก็สามารถเก็บค่าตำแหน่งของแขนกลไว้ในฐานข้อมูลได้เพื่อนำไปใช้ในการควบคุมแบบอัตโนมัติได้ต่อไป

การทำงานหลังจากกดปุ่ม "Save" แล้ว โปรแกรมจะทำการตรวจสอบว่ามีข้อมูลนี้อยู่ในฐานข้อมูลแล้วหรือยัง หากยังไม่มีก็จะทำการเก็บข้อมูลลงในฐานข้อมูลแล้วทำการแสดงข้อความว่า "บันทึกข้อมูลเรียบร้อยแล้ว" แต่หากมีข้อมูลชุดนี้อยู่แล้วก็จะมีข้อความว่า "มีข้อมูลนี้อยู่แล้ว"พอเสร็จก็จะจบการทำงานในส่วนของการบันทึกข้อมูลดังแสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 แสดงแผนภาพการเก็บข้อมูล

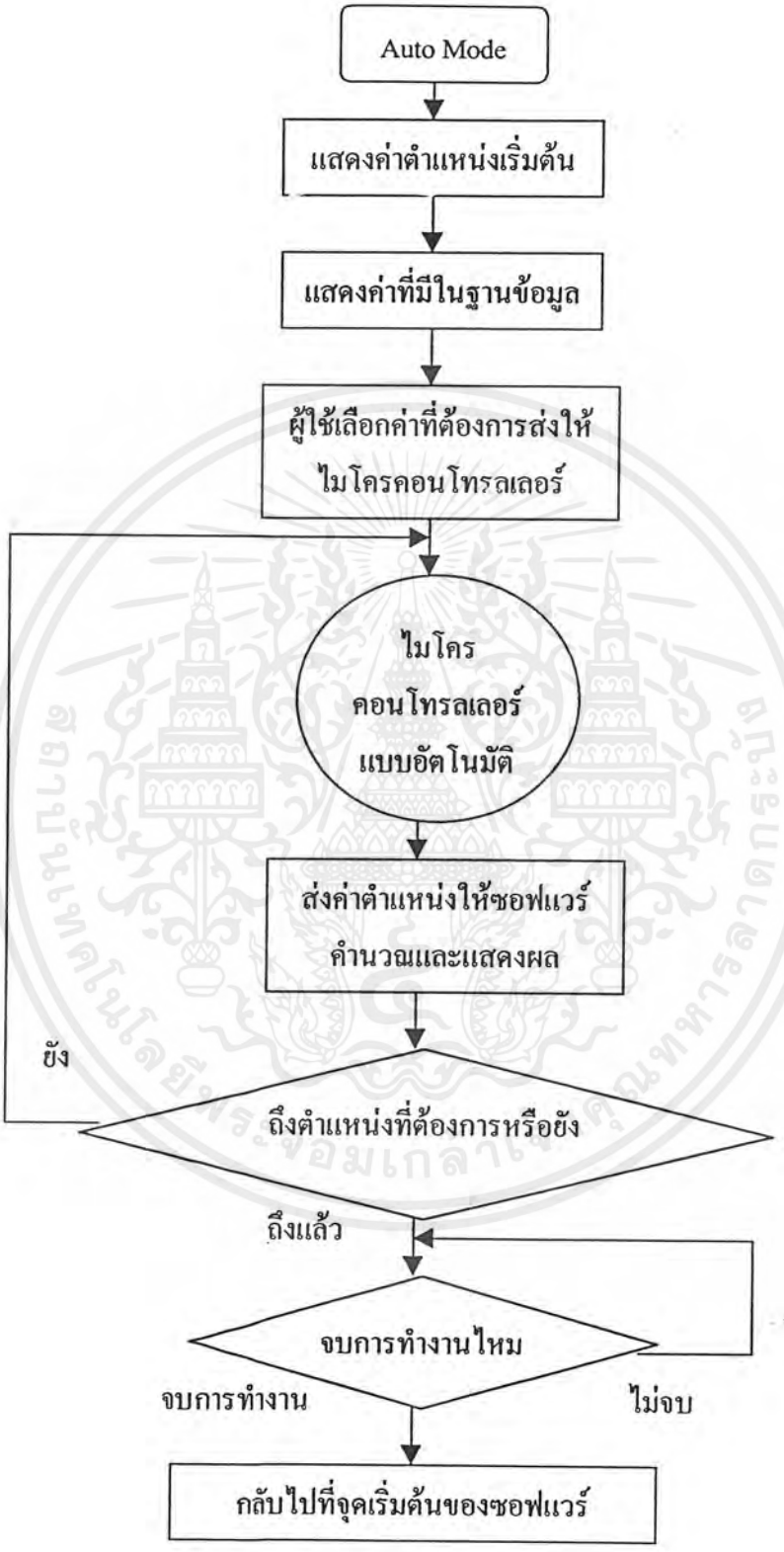
พอเก็บข้อมูลเรียบร้อยแล้วหรือหากผู้ใช้ไม่ต้องการบันทึกข้อมูล ผู้ใช้ก็สามารถเลือกเปลี่ยนไปใช้การควบคุมแบบอัตโนมัติได้ทันที แต่ถ้าหากไม่ต้องก็สามารถทำการควบคุมแบบนี้ได้ต่อไป หรือสามารถจบการทำงานเลยก็ได้ และเมื่อกดปุ่ม "Close" แล้วก็จะกลับไปยังหน้าจอหลักอีกครั้งหนึ่งพร้อมส่งค่า 00H ออกมา ขั้นตอนการทำงานทั้งหมดนี้สามารถแสดงได้ดังรูปที่ 3.6

การควบคุมแบบอัตโนมัติ (Auto Mode)

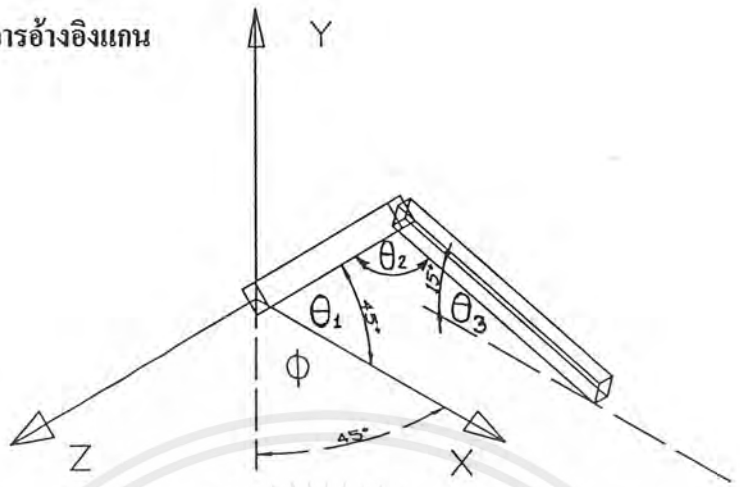
การทำงานในแบบนี้ก็จะเริ่มต้นจากการแสดงตำแหน่งปัจจุบันของแขนกลทั้งแบบรูปภาพและแบบเป็นตัวเลข รวมทั้งแสดงตารางซึ่งบรรจุข้อมูลตำแหน่งที่แขนกลได้เคยเคลื่อนที่ไปแล้วและมีการบันทึกค่าไว้ในฐานข้อมูลออกมาให้ผู้ใช้ทำการเลือกข้อมูลที่ต้องการให้แขนกลเคลื่อนที่ไปในการควบคุมครั้งนี้ เมื่อผู้ใช้เลือกชุดข้อมูลที่ต้องการได้แล้วก็เพียงแต่ใส่ค่ารัศมีของข้อมูลชุดนี้ลงในช่องที่กำหนดแล้วกดปุ่ม "O.K." โปรแกรมก็จะทำการค้นหาข้อมูลชุดที่เลือกนั้นแล้วส่งค่าตำแหน่งของมอเตอร์ทั้ง 3 ตัว ออกไปให้กับไมโครคอนโทรลเลอร์เก็บไว้เพื่อใช้ในการควบคุมให้แขนกลเคลื่อนที่ไปถึงยังตำแหน่งที่มีระยะห่างจากฐาน(รัศมี) ,ความสูงจากฐานและมุมที่ฐานหมุนไปรอบแกน Y

ไมโครคอนโทรลเลอร์จะทำการนำค่าที่ได้จากส่วนของการตรวจวัดตำแหน่งมาส่งให้กับซอฟแวร์เพื่อนำค่าต่างๆ ไปคำนวณและแสดงผลต่อไป และตัวมันเองก็จะนำค่าตำแหน่งที่ได้รับมานี้มาเปรียบเทียบกับค่าตำแหน่งที่ได้รับมาตอนแรกจากโปรแกรมว่ามีค่าเท่ากันหรือยัง ถ้ายังไมโครคอนโทรลเลอร์ก็จะทำการควบคุมวงจรขับมอเตอร์อีก แล้วก็นำค่าตำแหน่งมาส่งให้โปรแกรมและมาตรวจสอบอีก ทำเช่นนี้เรื่อยๆ จนกว่าค่าตำแหน่งที่วัดได้จะเท่ากับที่ผู้ใช้ต้องการ เมื่อเคลื่อนที่ถึงตำแหน่งที่ต้องการแล้วก็จะหยุดการทำงาน หากผู้ใช้ยังไม่กดปุ่ม "Close" การทำงานก็จะค้างไว้ที่จุดนี้ ถ้าหากมีอะไรมากระทำให้แขนกลเปลี่ยนตำแหน่งไปไมโครคอนโทรลเลอร์จะทำการควบคุมแบบป้อนกลับให้แขนกลกลับมาอยู่ ณ ตำแหน่งเดิม แต่ถ้าหากกดปุ่มแล้วโปรแกรมก็จะปิดการทำงานของหน้าจอนี้แล้วกลับไปยังหน้าจอหลัก ถ้าผู้ใช้ต้องการเปลี่ยนรูปแบบการควบคุมก็ทำได้ที่หน้าจอหลักนี้หรือหากต้องการทำงานควบคุมแบบเดิมอีกก็สามารถเลือกได้เช่นกัน การทำงานในแบบนี้สามารถแสดงได้ดังรูปที่ 3.8

นอกจากการควบคุมแล้วจะขออธิบายการอ้างอิงแกนและการคำนวณตำแหน่งของแขนกลซึ่งจะนำไปสู่การแสดงผลให้ผู้ใช้ทราบ การอธิบายในหัวข้อนี้ทำเพื่อให้ผู้ใช้สามารถเข้าใจตำแหน่งที่แสดงให้เห็นได้โดยง่าย



รูปที่ 3.8 แสดงแผนภาพการทำงานของการควบคุมแบบอัตโนมัติ



รูปที่ 3.9 แสดงให้เห็นถึงการอ้างอิงค่าตำแหน่งต่างๆและการคิดค่ามุม

ϕ = มุมที่มอเตอร์ตัวล่างหมุนรอบแกน y โดยวัดเทียบกับแกน x (แบบตามเข็มนาฬิกา) มีค่าระหว่าง 0 – 290 องศา

θ_1 = มุมที่แขนท่อนล่างทำกับแกน x มีค่าระหว่าง 0 – 180 องศา

θ_2 = มุมที่แขนท่อนบนทำกับแขนท่อนล่างมีค่าระหว่าง 0 – 270 องศา

การคำนวณตำแหน่ง

ค่าที่รับมาจากไมโครคอนโทรลเลอร์มีด้วยกันทั้งหมด 3 ค่าคือ ค่าตำแหน่งของมอเตอร์กระแสตรงซึ่งจะถูกนำไปเก็บไว้ในตัวแปรชื่อ value1 ,ค่าตำแหน่งของสเตปปิ้งมอเตอร์ตัวล่างซึ่งจะนำไปเก็บไว้ในตัวแปรชื่อ value2 และค่าตำแหน่งของสเตปปิ้งมอเตอร์ตัวบนซึ่งจะนำไปเก็บไว้ในตัวแปรชื่อ value3 โดยที่ค่าทั้ง 3 จะมีค่าอยู่ระหว่าง 0 – 255 เพราะในโครงการนี้เลือกใช้วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต(จึงมีค่าได้ทั้งหมด 256 ค่า)

จากการติดตั้งโพเทนชิโอมิเตอร์เข้ากับแกนของมอเตอร์พบว่าค่าที่ได้จากวงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลมีดังนี้

- มอเตอร์กระแสตรงมีค่าระหว่าง 0 –255
- สเตปปิ้งมอเตอร์ตัวล่างมีค่าระหว่าง 74 – 253
- สเตปปิ้งมอเตอร์ตัวบนมีค่าระหว่าง 0 – 247

โพเทนชิโอมิเตอร์สามารถหมุนได้ประมาณ 290 องศา ส่วนวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลสามารถแสดงค่าได้ 256 ค่า เพราะฉะนั้นอัตราส่วนในการแปลงค่าระหว่าง 2 อันนี้คือ

มุม 290 องศาจะเท่ากับค่าจากส่วนตรวจวัดตำแหน่ง 256 ค่า

มุม 1 องศาจะเท่ากับค่าจากส่วนตรวจวัดตำแหน่ง 256/290 = 0.883

หรือ ค่าจากส่วนวัดตำแหน่ง 1 ค่าแทนมุมที่หมุนไปเท่ากับ 290/256 = 1.133 องศา

ขั้นตอนการคำนวณทำได้ดังนี้

1. นำค่าที่ได้รับมาจากไมโครคอนโทรลเลอร์มาหักลบกับค่าต่ำสุดที่เป็นไปได้ของมเอเตอร์แต่ละตัวซึ่งจะมีค่าแตกต่างกันไปคือ 0 สำหรับมอเตอร์กระแสตรง ,74 สำหรับสเตปปิ้งมอเตอร์ตัวล่าง และ 0 สำหรับสเตปปิ้งมอเตอร์ตัวบน

2. นำค่าที่ได้ลบแล้วไปเก็บไว้ที่ตัวแปร Z0, X0 และY0 ตามลำดับ จากนั้นนำค่าที่ได้ไปคูณด้วย 1.133 องศาเพื่อแปลงเป็นค่าองศาแล้วนำไปเก็บไว้ยังตัวแปรเดิม

3. นำค่าทั้ง 3 ค่ามาคูณด้วย π แล้วหารด้วย 180 องศาเพื่อแปลงค่ามุมที่ได้ให้อยู่ในรูปของเรเดียน(radian) เนื่องจากการใช้การคำนวณด้วยฟังก์ชั่นตรีโกณในโปรแกรมวิชวลเบสิกจะใช้ได้เมื่อค่ามุมอยู่ในหน่วยเรเดียนเท่านั้น

4. จากรูปที่3.9จะพบว่า

$$\theta_2 = a + b$$

$$\theta_1 + a = 90$$

$$\theta_3 + b = 90$$

$$\theta_1 + \theta_2 + \theta_3 = 180$$

$$\theta_3 = 180-(\theta_1 + \theta_2)$$

$(L = L_1 = L_2)$

$$R = L\,Cos(\theta_1) + L\,Cos[180-(\theta_1 + \theta_2)]$$

$$= L\,Cos\,(\theta_1) - L\,Cos\,(\theta_1 + \theta_2)$$

$$H_1 = L\,Sin(\theta_1) + L\,Sin\,[180 - (\theta_1 + \theta_2)]$$

$$= L\,Sin(\theta_1) + L\,Sin\,(\theta_1 + \theta_2)$$

<u>หมายเหตุ</u> R = ระยะห่างจากฐาน

H_1 = ความสูงจากฐานวงกลม หากต้องการความสูงจากพื้นก็ทำได้โดยการนำไปบวกกับความสูงจากพื้นถึงฐานวงกลม

ส่วนค่ามุมที่มอเตอร์กระแสตรงหมุนไปได้จะมีค่าเท่ากับค่าที่เก็บไว้ในตัวแปร Z0 เลย

5. การสร้างรูปมีด้วยกันทั้งหมด 3 รูปโดยแต่ละรูปมีขั้นตอนดังนี้

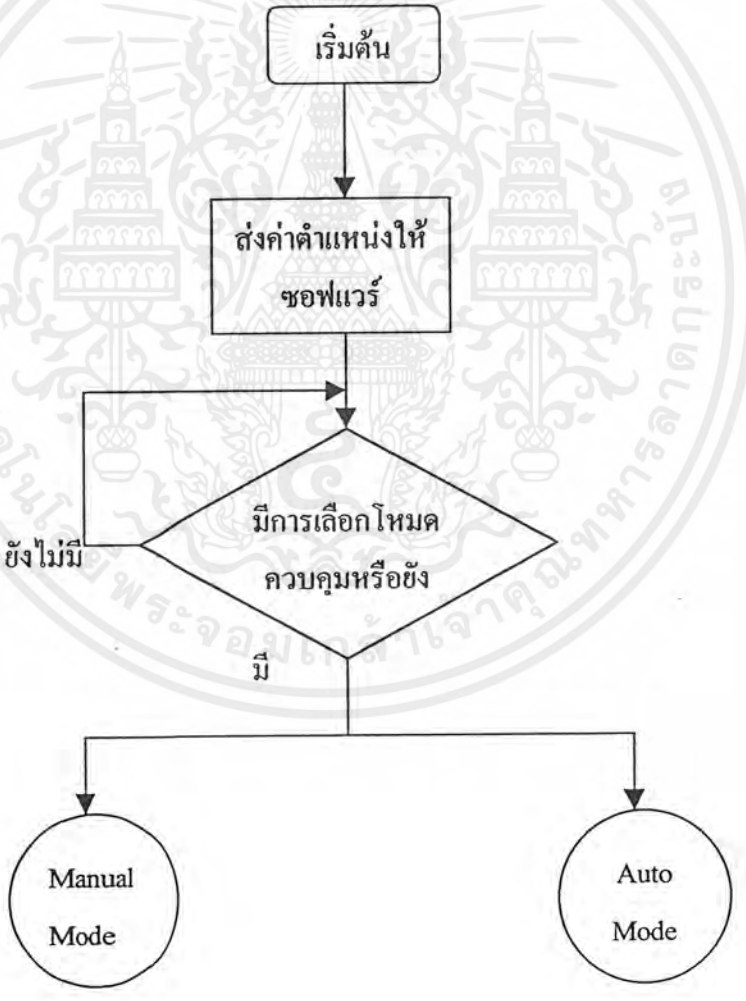
- ภาพฉายไปบนระนาบ X-Y คือการนำความยาวทางแกน X มาคูณด้วย $Cos(\phi)$ ส่วนความยาวทางแกน Y ไม่ต้องทำอะไรให้คงค่าเดิมไว้

- ภาพฉายไปบนระนาบ z-y คือการนำความยาวทางแกน x มาคูณด้วย Sin(ϕ) ส่วนความยาวทางแกน y ก็คงค่าเดิมไว้เช่นกัน

- ภาพแสดงมุมที่มอเตอร์กระแสตรงหมุนไปแทนได้ด้วยวงกลมซึ่งจะมีเส้นภายในแสดงว่าแขนหมุนไปแล้วกี่องศาแบบตามเข็มนาฬิกาเทียบกับแกน x

3.3.2 ส่วนของไมโครคอนโทรลเลอร์

ในการใช้งานไมโครคอนโทรลเลอร์เพื่อควบคุมแขนกลจะมีการแบ่งพอร์ตออกเป็นส่วนๆ ได้ดังนี้



รูปที่ 3.10 แสดงการทำงานของไมโครคอนโทรลเลอร์

ส่วนขับมอเตอร์

จะใช้เป็นเอาต์พุตพอร์ตสำหรับขับมอเตอร์ 3 ตัวโดยแบ่งเอาต์พุตพอร์ตของ 8255 ซึ่งต่อเข้ากับไมโครคอนโทรลเลอร์แบบ Memory Map โดยมีตำแหน่งดังนี้

พอร์ต A = 8000H สำหรับสเตปปิ้งมอเตอร์ตัวบน

พอร์ต B = 8001H สำหรับสเตปปิ้งมอเตอร์ตัวล่าง

พอร์ต C = 8002H สำหรับมอเตอร์กระแสตรงและพอร์ตนี้ยังใช้ควบคุมการทำงานของวงจรแปลงสัญญาอะนาลอกเป็นดิจิตอลอีกด้วย

ส่วนรับค่าตำแหน่ง

จะใช้พอร์ต 1 เป็นอินพุตพอร์ตในการรับค่าตำแหน่งของแขนกลแต่ละส่วนจากวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลเข้ามาเพื่อทำการแสดงผลและตรวจเช็คว่าตรงกับค่าที่ได้รับมาจากคอมพิวเตอร์หรือไม่

ส่วนรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์

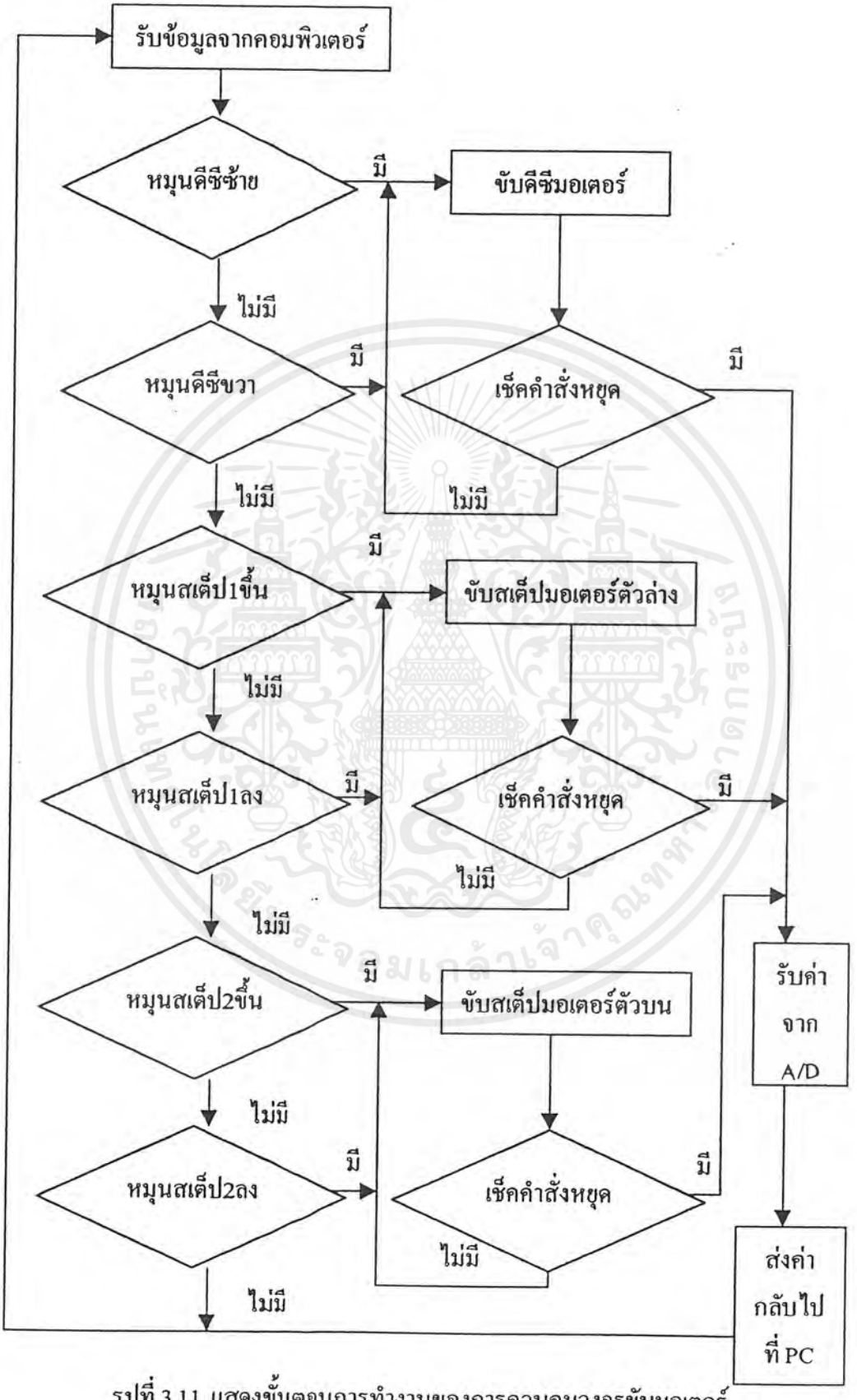
จะใช้พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ในการรับส่งข้อมูล โดยข้อมูลที่ได้รับมาจากคอมพิวเตอร์จะมีทั้งคำสั่งในการเคลื่อนที่และค่าตำแหน่งที่ต้องการให้เคลื่อนที่ไปถึง ส่วนค่าที่ไมโครคอนโทรลเลอร์ส่งไปจะมีแต่ค่าตำแหน่งเท่านั้น

ส่วนต่อไปจะเป็นการอธิบายถึงขั้นตอนการทำงานต่างๆ ของไมโครคอนโทรลเลอร์แยกเป็นส่วนๆ ได้ดังนี้

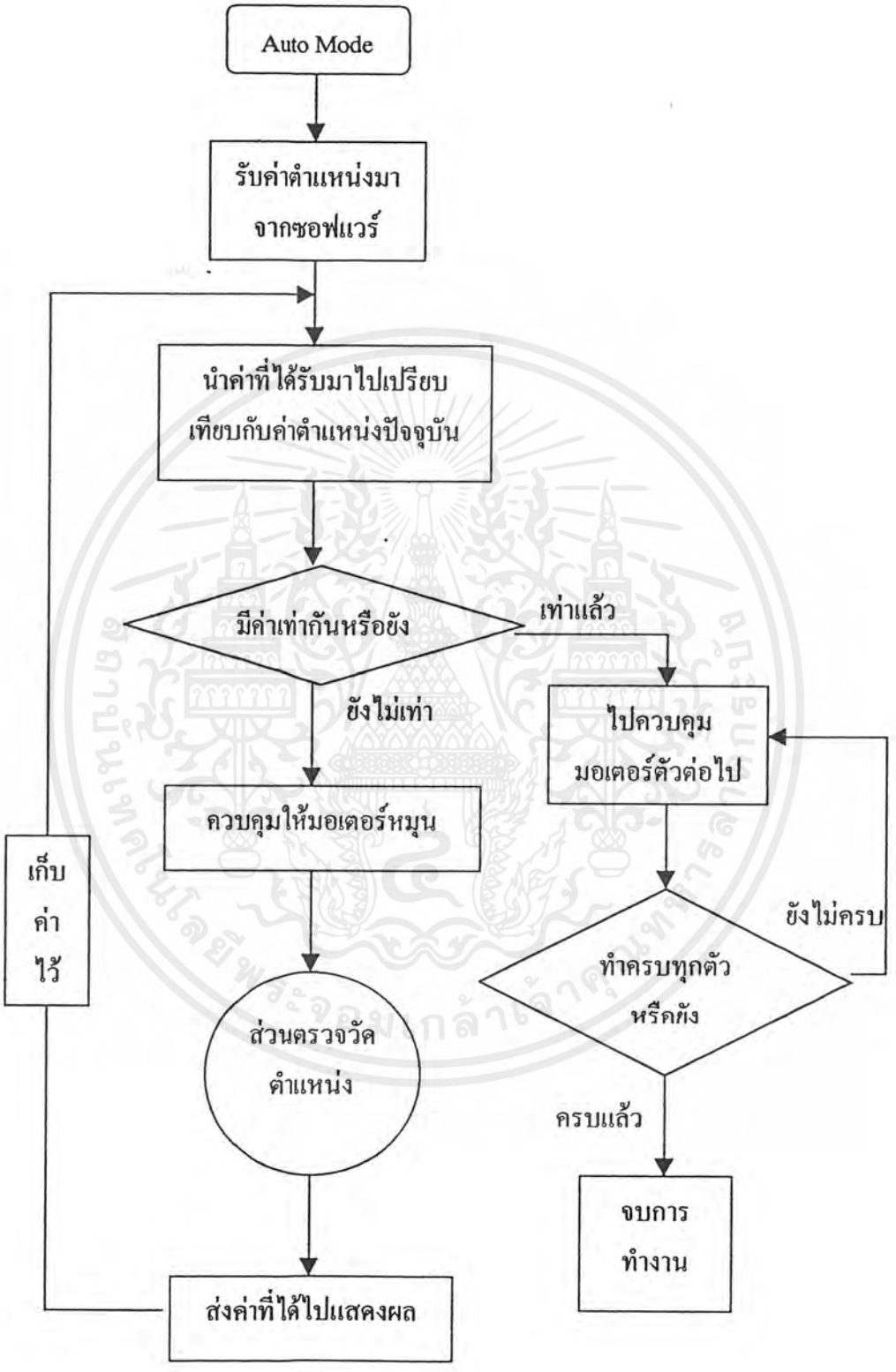
ส่วนตรวจสอบรูปแบบการควบคุม

เป็นการเริ่มต้นการทำงานของระบบโดยไมโครคอนโทรลเลอร์จะรับคำสั่งจากคอมพิวเตอร์มาตรวจดูว่าผู้ใช้ต้องการเลือกการควบคุมแบบใดโดยรับข้อมูลเข้ามาทางพอร์ตอนุกรมแล้วนำมาตรวจสอบกับค่าที่กำหนดไว้ในโปรแกรมโดยแบ่งการควบคุมได้เป็น 2 แบบคือ การควบคุมแบบผู้ใช้ควบคุมเองจะได้รับค่า F7H มาจากคอมพิวเตอร์ แต่หากเป็นการควบคุมแบบอัตโนมัติจะได้รับค่า FDH มาจากคอมพิวเตอร์

การควบคุมแบบผู้ใช้ควบคุมเอง การทำงานแบบนี้ผู้ใช้จะควบคุมให้แขนกลเคลื่อนที่ไปยังตำแหน่งที่ต้องการด้วยตัวเองโดยจะมีขั้นตอนดังนี้ ไมโครคอนโทรล เลอร์จะรับคำสั่งเข้ามาจากคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมมาตรวจสอบดูว่าคำสั่งที่ได้รับเป็นของมอเตอร์ตัวใดและให้เคลื่อนที่ไปในทิศทางใดโดยใช้รีจิสเตอร์ภายใน R0 – R7 กลุ่มที่ 1 ในการควบคุมการหมุนของมอเตอร์ทั้ง 3 ตัว ในระหว่างที่ทำคำสั่งควบคุมอยู่นั้นจะมีการวนกลับไปตรวจสอบอยู่ตลอดเวลาว่ามีคำสั่งหยุดส่งมาหรือไม่ เพื่อที่จะสามารถหยุดการเคลื่อนที่ได้ตามที่ต้องการ จากนั้นก็จะไปทำงานในส่วนของการแสดงผลและเก็บค่าตำแหน่งซึ่งจะอธิบายไว้ในส่วนต่อไป



รูปที่ 3.11 แสดงขั้นตอนการทำงานของการควบคุมวงจรขับมอเตอร์



รูปที่ 3.12 แสดงแผนภาพการทำงานแบบการควบคุมอัตโนมัติ

การควบคุมแบบอัตโนมัติ

การทำงานในรูปแบบนี้ผู้ใช้เพียงแต่เลือกค่าตำแหน่งที่ตนต้องการจากฐานข้อมูลระบุลงในโปรแกรมที่เขียนขึ้นแล้วโปรแกรมจะทำการส่งค่าของมอเตอร์แต่ละตัวเพื่อให้แขนกลเคลื่อนที่ไปถึงตำแหน่งที่ต้องการออกมาเอง และเมื่อเคลื่อนที่ถึงตำแหน่งที่ต้องการแล้วก็จะหยุดเองอัตโนมัติ การควบคุมแบบนี้สามารถทำได้ครั้งละ 1 ตำแหน่งเท่านั้นหากต้องการควบคุมแบบนี้อีกต้องกลับไปเลือกใหม่ที่หน้าจอหลักของโปรแกรมบนคอมพิวเตอร์ จะมีขั้นตอนการทำงานดังนี้

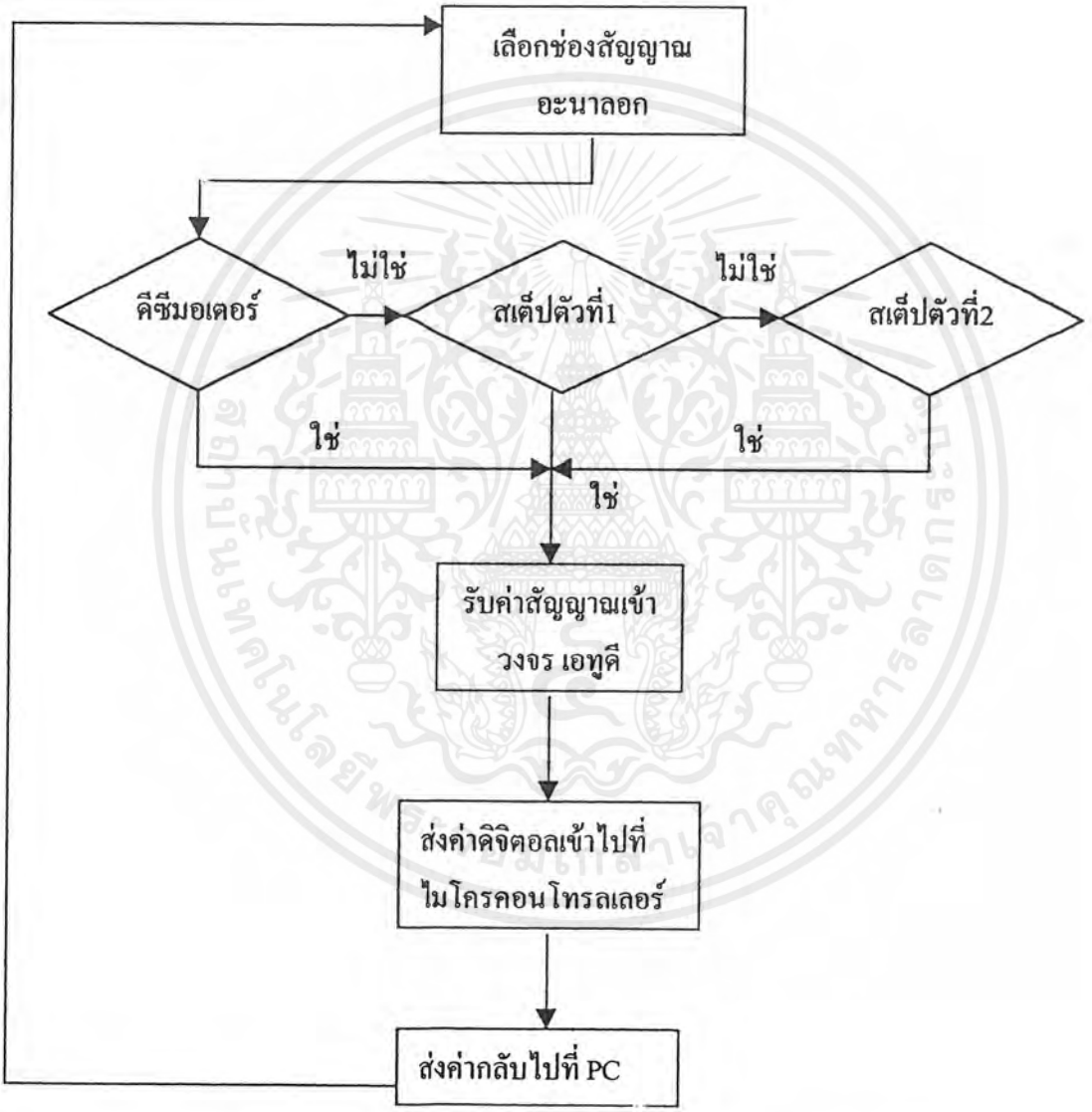
เมื่อผู้ใช้เลือกการทำงานแบบนี้แล้วไมโครคอนโทรลเลอร์จะรนับค่าต่างๆ ที่ใช้ในการควบคุมตำแหน่งของมอเตอร์ทั้ง 3 มาเก็บไว้ในรีจิสเตอร์ภายใน R0 – R7 กลุ่มที่ 2 จากนั้นจะทำการนำค่าเหล่านี้ไปเปรียบเทียบกับค่าตำแหน่งปัจจุบันที่ได้รับมาจากวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล เมื่อเปรียบเทียบค่าของมอเตอร์แต่ละตัวกับค่าที่ได้มาจากคอมพิวเตอร์เรียบร้อยแล้วก็จะทำการควบคุมให้มอเตอร์เคลื่อนที่ไปตามทิศทางที่จะทำให้แขนกลเคลื่อนที่ไปถึงตำแหน่งที่ได้รับมาจากคอมพิวเตอร์ ในระหว่างที่เคลื่อนที่ไปนั้นจะมีการนำค่าตำแหน่งที่เปลี่ยนไปมาเปรียบเทียบกับค่าตำแหน่งที่ต้องการอยู่ตลอด จนกระทั่งค่าทั้ง 2 ตรงกัน แล้วค่อยไปทำการควบคุมการเคลื่อนที่ของมอเตอร์ตัวอื่นต่อไป ทำเช่นนี้จนกระทั่งมอเตอร์ทั้ง 3 ตัวเคลื่อนที่ไปถึงยังตำแหน่งที่ต้องการจากนั้นก็จะส่งค่าตำแหน่งไปให้กับส่วนของการแสดงผลและเก็บตำแหน่งต่อไป

ส่วนของการแสดงผลและเก็บค่าตำแหน่ง

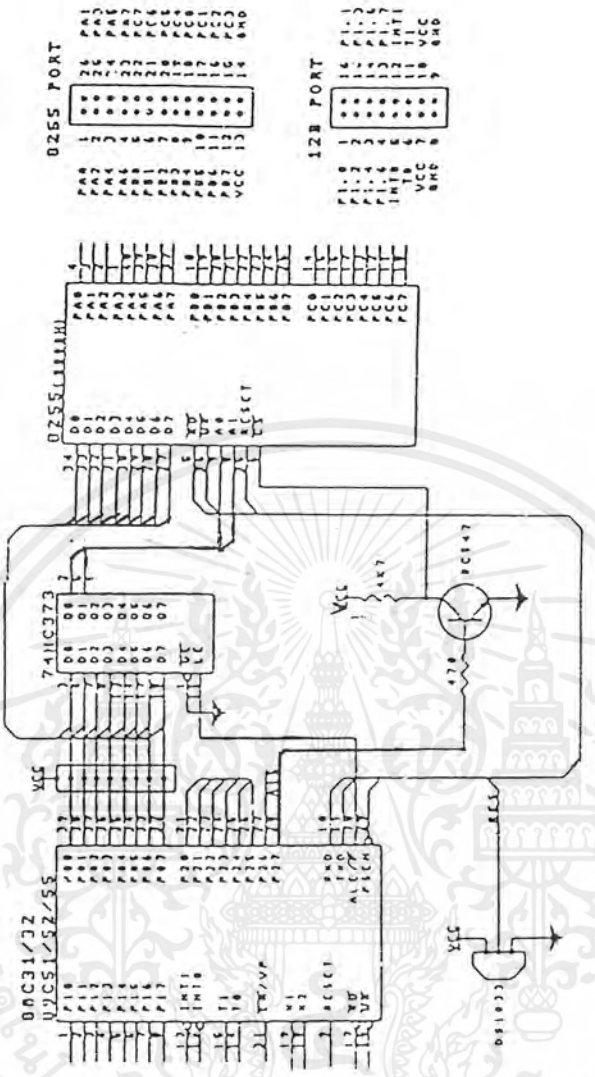
เป็นส่วนของการแสดงค่าตำแหน่งปัจจุบันของมอเตอร์โดยใช้โพเทนชิโอมิเตอร์ในการตรวจวัดตำแหน่ง แล้วนำค่าที่ได้ไปผ่านวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต จากนั้นนำค่าตำแหน่งที่ได้ไปเก็บไว้เพื่อเปรียบเทียบ(ใช้ในกรณีที่เป็นการควบคุมแบบอัตโนมัติเท่านั้น) และเพื่อนำไปแสดงผลโดยส่งค่ากลับไปยังคอมพิวเตอร์ทางพอร์ตอนุกรมขั้นตอนการทำงานมีดังนี้

เมื่อเข้าสู่ส่วนนี้แล้วจะใช้พอร์ต C ของ 8255 มาควบคุมการทำงานของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลในที่นี้ใช้ไอซีเบอร์ AD0808 เพื่อเลือกช่องสัญญาณอะนาลอกที่จะใช้โดยกำหนดได้จากค่าแอดเดรสที่เราจะส่งออกมาจากทางพอร์ต C จากนั้นวงจรก็จะทำการแปลงค่าของช่องสัญญาณที่เลือกแล้วให้ไปเป็นเลขฐาน 2 ขนาด 8 บิตแล้วส่งให้กับไมโครคอนโทรลเลอร์ทางพอร์ต 1 เพื่อให้ไมโครคอนโทรลเลอร์นำภาพไปเก็บไว้ในรีจิสเตอร์ภายใน R0 – R7 กลุ่มที่ 2 แล้ววนกลับไปส่งค่าแอดเดรสใหม่เพื่อเลือกช่องสัญญาณช่องต่อไปให้วงจรนำค่าอะนาลอกไปแปลงเป็นค่าดิจิตอลเสร็จแล้วก็ส่งมาให้ไมโครคอนโทรลเลอร์อีกทำเช่นนี้จนครบทั้ง 3 ช่องสัญญาณ จากนั้นไมโครคอนโทรลเลอร์ก็จะส่งค่าตำแหน่งที่เก็บไว้ในรีจิสเตอร์ไปให้กับ

คอมพิวเตอร์ ส่วนค่าตำแหน่งที่อยู่ในรีจิสเตอร์ชุดนี้ก็ยังเก็บไว้เพื่อใช้ในการเปรียบเทียบในการควบคุมแบบอัตโนมัติ เมื่อส่งค่าตำแหน่งไปให้ครบทั้ง 3 ค่าแล้วโปรแกรมก็จะกลับไปเริ่มต้นที่การควบคุมใหม่



รูปที่ 3.13 แสดงขั้นตอนการทำงานในการระบุตำแหน่งของแขนกล



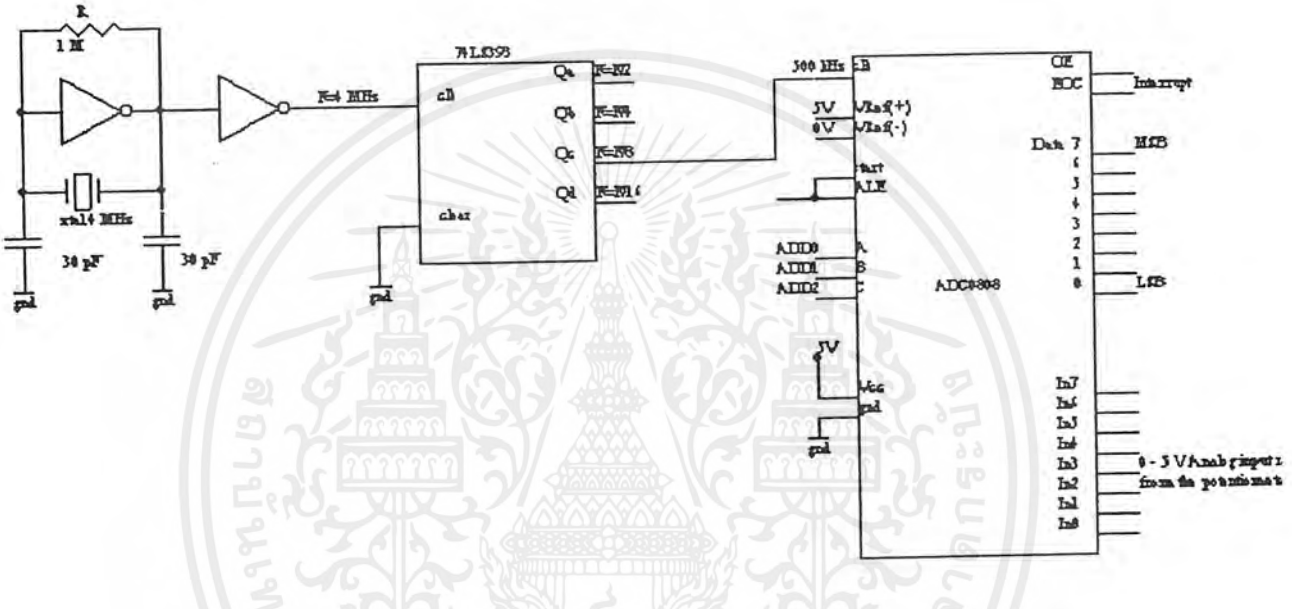
รูปที่ 3.14 การเชื่อมต่อส่วนต่างๆในวงจรไมโครคอนโทรลเลอร์

3.3.3 วงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล

โครงงานนี้เลือกใช้ไอซี ADC 0808 ซึ่งใช้หลักการทำงานแบบการประมาณค่า (Successive Approximation) แล้วนำสัญญาณที่ได้ไปแปลงเป็นเลขฐาน 2 ส่งให้กับไมโครคอนโทรลเลอร์ต่อไป การทำงานของวงจรนี้จะรับอินพุตมาจากโพเทนชิโอมิเตอร์ซึ่งแกนของมันจะต่อกับแกนการเคลื่อนที่ของแขนกลแต่ละส่วน ในที่นี้เลือกใช้ค่าความต้านทานปรับค่าได้ขนาด 50 กิโลโอมห์ จำนวน 3 ตัวติดไว้ที่แกนการหมุนของแขนแต่ละส่วน

สัญญาณควบคุมที่นำมาใช้ควบคุมวงจรนี้จะได้รับมาจากพอร์ต C ของ 8255 โดยที่ค่านี้ 8255 ก็รับมาจากไมโครคอนโทรลเลอร์อีกทีหนึ่ง สัญญาณที่ส่งมาควบคุมการทำงานมีดังนี้ สัญญาณที่ใช้ระบุค่าแอดเดรส ,สัญญาณ ALE ,สัญญาณ START และสัญญาณ OEB โดยการ

ทำงานของวงจรจะทำงานที่ความถี่ 500 กิโลเฮิร์ตซึ่งได้มาจากคริสตอล 4 เมกกะเฮิร์ตแล้วนำมาหารความถี่ลงด้วยการใช้อินเวิสเตอร์และวงจรนับหาร 8 เพื่อให้ได้ความถี่ 500 กิโลเฮิร์ตตามที่ต้องการ และค่าของข้อมูลที่ได้เป็นเอาต์พุตจากวงจรนี้จะถูกส่งกลับไปให้ไมโครคอนโทรลเลอร์ทางพอร์ต เป็นการทำงานร่วมกันระหว่างซอฟแวร์กับไมโครคอนโทรลเลอร์ ซึ่งสามารถแสดงขั้นตอนการทำงานได้ดังรูปที่ 3.15



รูปที่ 3.15 แสดงการต่อวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลที่ใช้ในโครงงาน

บทที่ 4

ผลการทดลอง

การทดลองประสิทธิภาพของโครงงานนี้แบ่งออกเป็น 2 ส่วนคือ การทำงานของไมโครคอนโทรลเลอร์ทั้งในการควบคุมวงจรขับ, การรับส่งข้อมูลกับคอมพิวเตอร์ และการระบุตำแหน่งของแขนกลแต่ละชิ้นส่วน และส่วนที่สองคือการทำงานของโปรแกรมในการสั่งงานไปให้กับไมโครคอนโทรลเลอร์และการรับข้อมูลกลับมาเพื่อแสดงค่าออกมาที่หน้าจอ

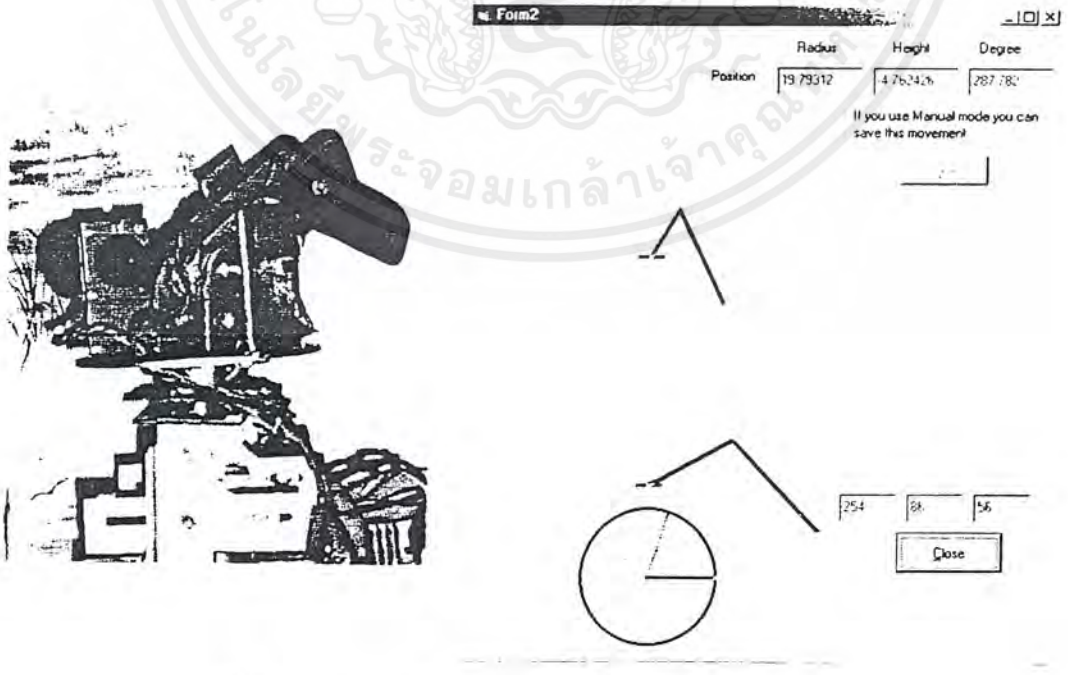
4.1 **ส่วนของไมโครคอนโทรลเลอร์** แบ่งการทำงานออกเป็น 3 อย่างคือ

4.1.1 **การควบคุมวงจรขับมอเตอร์ให้ทำงาน**

ตรวจสอบได้โดยการเขียนโปรแกรมทดสอบให้ไมโครคอนโทรลเลอร์จ่ายสัญญาณออกมาควบคุมวงจรขับแต่ละอัน แล้วดูว่าวงจรขับทำงานถูกต้องตามที่เราเขียนโปรแกรมควบคุมไว้หรือไม่ ตลอดจนแขนกลสามารถหยุดค้าง ณ ตำแหน่งที่เราต้องการได้หรือไม่

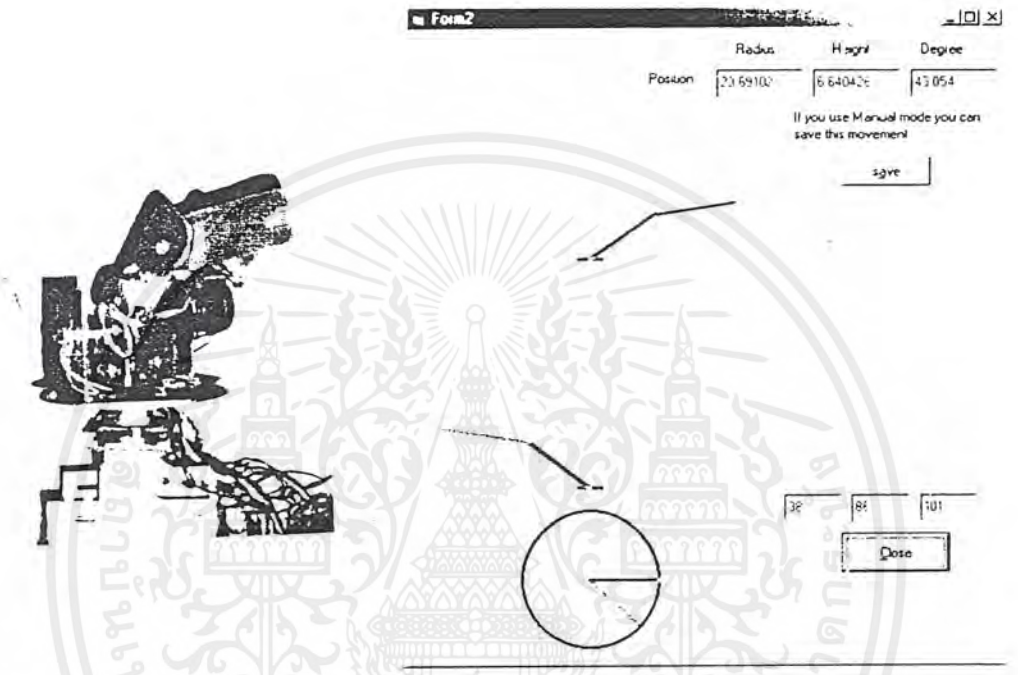
การควบคุมแบ่งออกเป็น 2 แบบด้วยกันคือ แบบที่ผู้ใช้ควบคุมเอง และการควบคุมแบบอัตโนมัติ

- แบบที่ผู้ใช้ควบคุมเอง คือการควบคุมที่ผู้ใช้สามารถสั่งให้แขนกลเคลื่อนที่ไปยังตำแหน่งใดก็ได้ตามต้องการ ดังจะแสดงให้เห็นได้ตามตัวอย่างข้างล่าง



รูปที่ 4.1 แสดงภาพตัวอย่างการควบคุมแบบผู้ใช้ควบคุมเอง

- แบบอัตโนมัติ คือ การควบคุมที่ผู้ใช้สามารถเลือกตำแหน่งที่จะให้แขนกลเคลื่อนที่ไปโดยเลือกได้จากข้อมูลที่มีอยู่ในฐานข้อมูลตำแหน่งที่ได้บันทึกไว้ล่วงหน้าแล้ว จากนั้นแขนกลก็จะเคลื่อนที่ไปถึงตำแหน่งที่เลือก



รูปที่ 4.2 แสดงภาพตัวอย่างการควบคุมแบบอัตโนมัติ

จากการทดสอบพบว่าวงจรขับสามารถทำงานตามที่ไมโครคอนโทรลเลอร์สั่งงานได้อย่างถูกต้อง อีกทั้งแขนกลยังสามารถหยุดค้าง ณ ตำแหน่งที่ต้องการได้

4.1.2 **การรับส่งข้อมูลกับคอมพิวเตอร์**

เมื่อทำการส่งค่าจากคอมพิวเตอร์หลังจากที่มีการกดปุ่มเพื่อควบคุมการเคลื่อนที่ของแขนกล พบว่าแขนกลแต่ละส่วนสามารถเคลื่อนที่ได้อย่างถูกต้องตามที่เราต้องการ นั่นก็คือไมโครคอนโทรลเลอร์สามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างถูกต้อง และคอมพิวเตอร์ก็สามารถรับข้อมูลที่ส่งมาจากไมโครคอนโทรลเลอร์ได้

4.1.3 **การระบุตำแหน่ง** โดยเป็นการทำงานร่วมกับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลและโพเทนชิโอมิเตอร์

การทดสอบส่วนนี้แบ่งออกเป็น 2 ส่วนด้วยกันคือ การทดสอบการทำงานของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลในการแปลงค่าแรงดันที่ส่งมาจากโพเทนชิโอมิเตอร์

แต่ละตัวว่าสามารถทำการแปลงได้อย่างถูกต้องหรือไม่ โดยในการทดสอบส่วนนี้จะยังไม่เชื่อมต่อกับคอมพิวเตอร์ เพราะฉะนั้นสัญญาณต่างๆที่จำเป็นจะต้องจ่ายออกมาจากไมโครคอนโทรลเลอร์เราจะใช้การสวิตช์แทนไปก่อน ซึ่งหลังการทดลองพบว่าวงจรสามารถทำงานได้ถูกต้อง

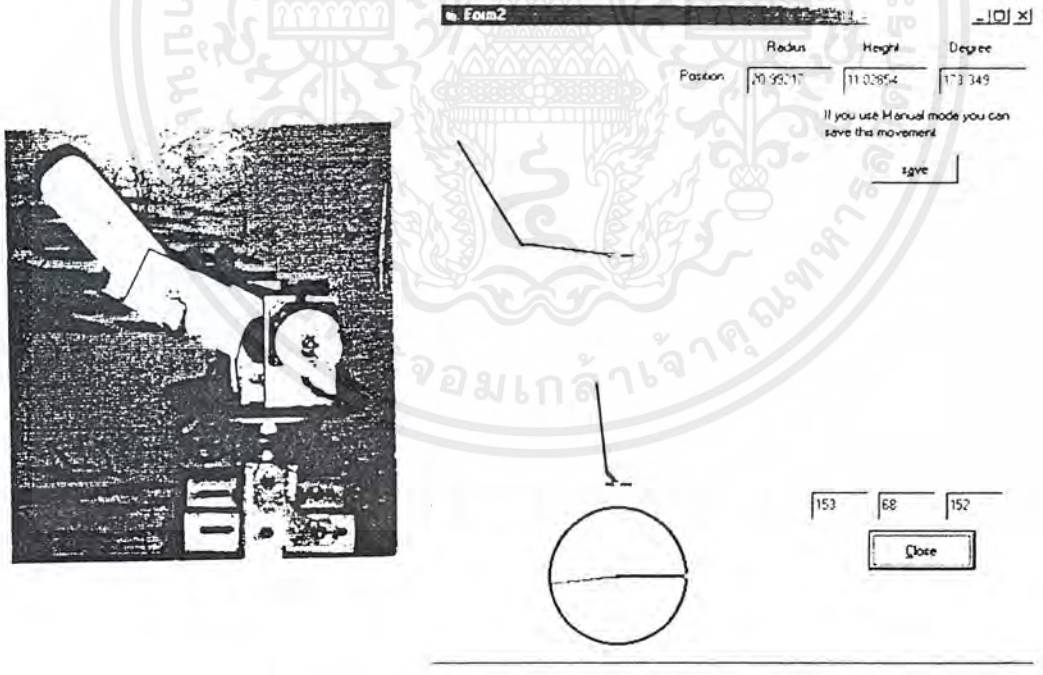
ส่วนที่ 2 คือ การส่งค่าเลขฐาน 2 ที่ได้จากการแปลงเรียบร้อยแล้วไปให้กับไมโครคอนโทรลเลอร์ และการส่งสัญญาณควบคุมการทำงานของวงจรระบุตำแหน่งออกมาจากไมโครคอนโทรลเลอร์ จากการทดสอบพบว่า ไมโครคอนโทรลเลอร์สามารถส่งข้อมูลออกไปควบคุมให้วงจรทำงานได้อย่างถูกต้อง และไมโครคอนโทรลเลอร์สามารถรับข้อมูลมาได้อย่างถูกต้อง

4.2 ส่วนของซอฟแวร์ แบ่งออกเป็น 2 ส่วนคือ

4.2.1 ในการส่งค่าออกไปให้กับไมโครคอนโทรลเลอร์ทำงานตามที่ต้องการ

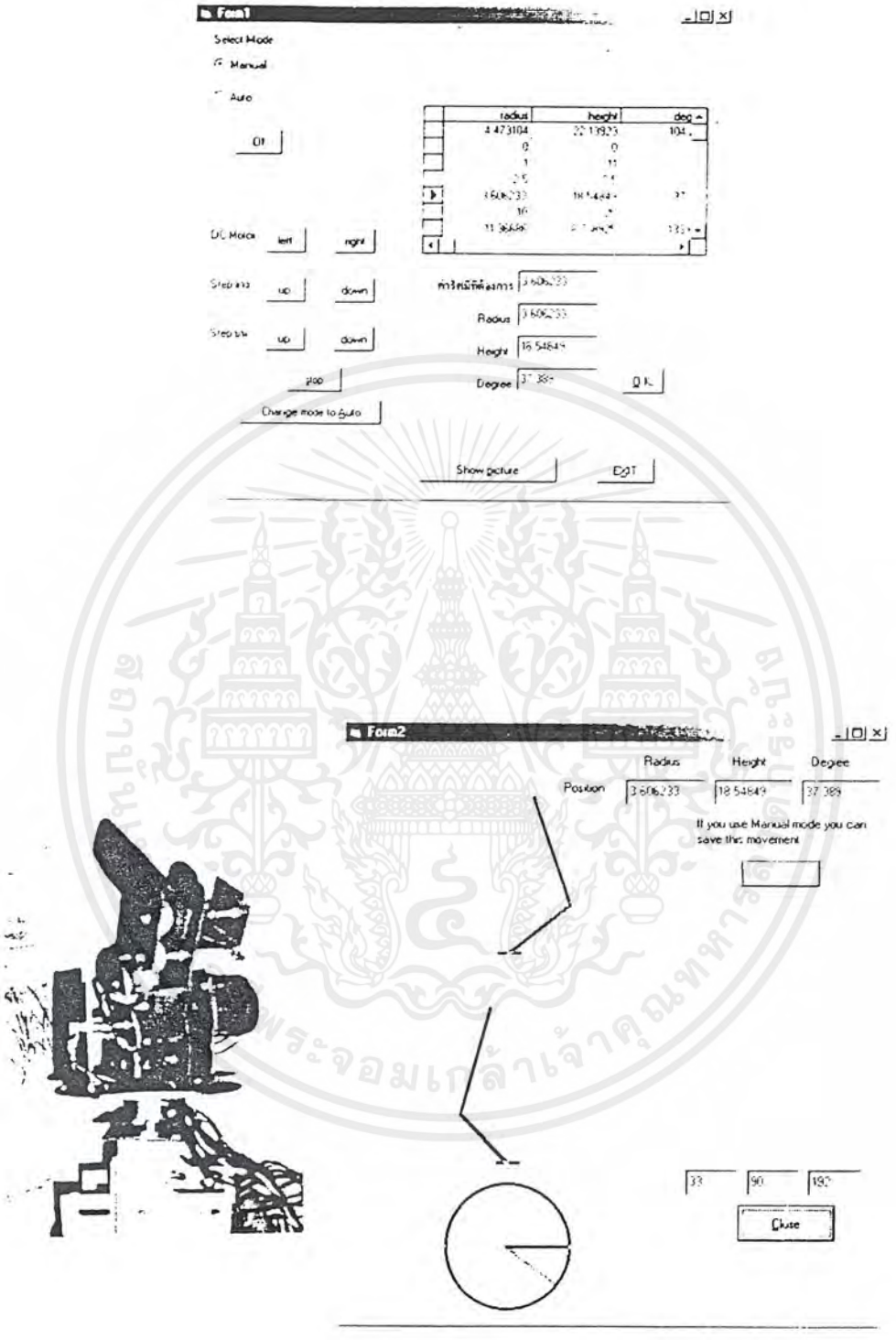
การทำงานของส่วนนี้แบ่งออกเป็น 2 แบบด้วยกันคือ การควบคุมแบบที่ผู้ใช้ควบคุมเอง และแบบอัตโนมัติ

- การควบคุมแบบที่ผู้ใช้ควบคุมเอง จะมีปุ่มกดควบคุมแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ กลุ่มของปุ่มที่ใช้ควบคุมทิศทางที่จะให้มอเตอร์แต่ละตัวหมุนไป และปุ่มหยุด



รูปที่ 4.3 แสดงภาพตัวอย่างการควบคุมแบบผู้ใช้ควบคุมเอง

- การควบคุมแบบอัตโนมัติ เป็นการควบคุมที่ผู้ใช้จะเลือกตำแหน่งที่ต้องการจะให้แขนกลเคลื่อนที่ไปจากค่าตำแหน่งที่เก็บไว้ในฐานข้อมูล



รูปที่ 4.4 แสดงภาพตัวอย่างการควบคุมแบบอัตโนมัติทั้งหน้าจอควบคุมและแสดงผล

จากการทดลองควบคุมให้แขนกลเคลื่อนที่ทั้ง 2 แบบพบว่าแขนกลสามารถเคลื่อนที่ไปถึงตำแหน่งที่ต้องการได้ดีทั้งสองแบบ ในแบบที่ผู้ใช้ควบคุมเองแขนกลก็สามารถหมุน ได้ถูกทิศทางตามที่ผู้ใช้ต้องการและหยุดได้เมื่อต้องการ ส่วนในแบบอัตโนมัติแขนกลก็สามารถเคลื่อนที่ไปหยุด ณ ตำแหน่งที่ต้องการได้พอดี

4.2.2 การนำค่าที่รับมาจากไมโครคอนโทรเลอร์มาแสดงผล

คอมพิวเตอร์สามารถรับค่ามาแสดงได้อย่างถูกต้อง ซึ่งจะเห็นได้จาก หากไม่มีการเปลี่ยนค่าตำแหน่งของแขนกลค่าที่แสดงที่หน้าจอก็จะคงเดิม และในทางกลับกันหากส่วนใดๆ ก็ตามของแขนกลมีการเคลื่อนไหวค่าตำแหน่งของส่วนนั้นก็จะเปลี่ยนเพียงส่วนเดียว จากการทดสอบซอฟแวร์ก็สามารถทำงานในส่วนนี้ได้อย่างถูกต้อง

ในส่วนของภาพที่แสดงก็มีความใกล้เคียงกับตำแหน่งจริงของแขนกลทั้งภาพฉายไปบนระนาบ X-Y, ระนาบ Z-Y และภาพวงกลมแสดงมุมที่มอเตอร์กระแสตรงหมุนไปแบบตามเข็มนาฬิกา(วัดเทียบแกน X)

บทที่ 5
สรุปผลและวิจารณ์การทำโครงงาน

จากการทำโครงงานพบว่าการทำงานของส่วนประกอบต่างๆของโครงงานสามารถทำงานได้ตามที่ต้องการ ทั้งในส่วนของตัวขับมอเตอร์ทั้งสองตัว ไมโครคอนโทรลเลอร์ ส่วนการระบุตำแหน่ง ตลอดจนโปรแกรมที่เขียนด้วยภาษาวิชวลเบสิก โดยส่วนสำคัญที่ต้องคำนึงถึงคือ การรับส่งข้อมูลระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ การควบคุมวงจรขับให้ทำงานโดยไมโครคอนโทรลเลอร์ซึ่งได้รับคำสั่งมาจากผู้ใช้ผ่านทางคอมพิวเตอร์ และการแสดงค่าตำแหน่งซึ่งอาศัยการทำงานของโพเทนชิโอมิเตอร์ร่วมกับวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล แล้วส่งต่อไปให้ไมโครคอนโทรลเลอร์

การทำงานที่ผ่านมาทำให้ได้รับประโยชน์หลายประการด้วยกัน อีกทั้งยังประสบปัญหาหลายๆประการ โดยผลที่ได้รับและปัญหาที่เกิดขึ้นสามารถสรุปเป็นข้อๆได้ดังนี้

ปัญหาที่เกิดขึ้น

1. ปัญหาการเขียนโปรแกรมซอฟแวร์ ในส่วนของการเขียนโปรแกรมเพื่อสร้างภาพต่างๆ เนื่องจากขาดประสบการณ์ในการคำนวณสมการของภาพฉาย3 มิติไปบนระนาบต่างๆ

2. การระบุตำแหน่งของชิ้นส่วนต่างๆ การอ้างอิงแกนที่นำมาใช้

3. ปัญหาที่เกิดจากการขาดความรู้และความชำนาญในการทำงาน โดยเฉพาะการทำงานทางด้านแมคคานิคส์

4. เครื่องคอมพิวเตอร์ที่เลือกใช้ในตอนแรกมีคุณลักษณะต่ำกว่าที่ซอฟแวร์ต้องการ

5. การขาดความระมัดระวังในการทำงานทำให้เกิดความเสียหาย

ผลที่ได้รับ

1. ได้รับความรู้เกี่ยวกับการเขียนโปรแกรมด้วยภาษาวิชวลเบสิก และการนำไปใช้ในการเขียนโปรแกรมเพื่อสร้างภาพฉายต่างๆในการแสดงผล

2. ทำให้มีความรู้ความเข้าใจในการทำงานของไมโครคอนโทรลเลอร์ทั้งการรับและส่งข้อมูลกับส่วนอื่นของโครงงาน

3. ได้รับประสบการณ์ในการทำงานทางด้านแมคคานิคส์ รวมทั้งการใช้โปรแกรมเพื่อเขียนแบบภาพแขนกลในการออกแบบโดยการใช้โปรแกรมออโต้แคด

บรรณานุกรม

1. ไกรวุฒิ โรจน์ประเสริฐสุด , เข้าใจสร้างและเล่น ไมโครคอนโทรลเลอร์ 2 , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2539 , หน้าที่ 25 - 31

2. ไชยันต์ สุวรรณชีวะศิริ , เซมิคอนดักเตอร์อิเลคทรอนิคส์ การควบคุมตำแหน่ง ทิศทาง และความเร็ว , บริษัท ซีเอ็ดยูเคชั่น , 2535 , หน้าที่ 96 – 101

3. ไชยันต์ สุวรรณชีวะศิริ , เซมิคอนดักเตอร์อิเลคทรอนิคส์ สเต็ปปิ้งมอเตอร์และการควบคุม , บริษัท ซีเอ็ดยูเคชั่น , 2541 , หน้าที่ 102 – 110

4. ประเมษฐ์ ประนยานันทน์และปิยพงศ์ เผ่าวณิช , คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MSC – 51 , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2536 , หน้าที่ 18 - 45

5. ประมวิตร วิสุทธิศิริ , เซมิคอนดักเตอร์อิเลคทรอนิคส์ วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล , บริษัท ซีเอ็ดยูเคชั่น , 2533 , หน้าที่ 302 - 309

6. พรชัย ประทุมสุวรรณ , เครื่องมือวัดอุตสาหกรรมเซนเซอร์และทรานดิวเซอร์ , เรือนแก้วการพิมพ์ , 2541 , หน้าที่ 79 – 81

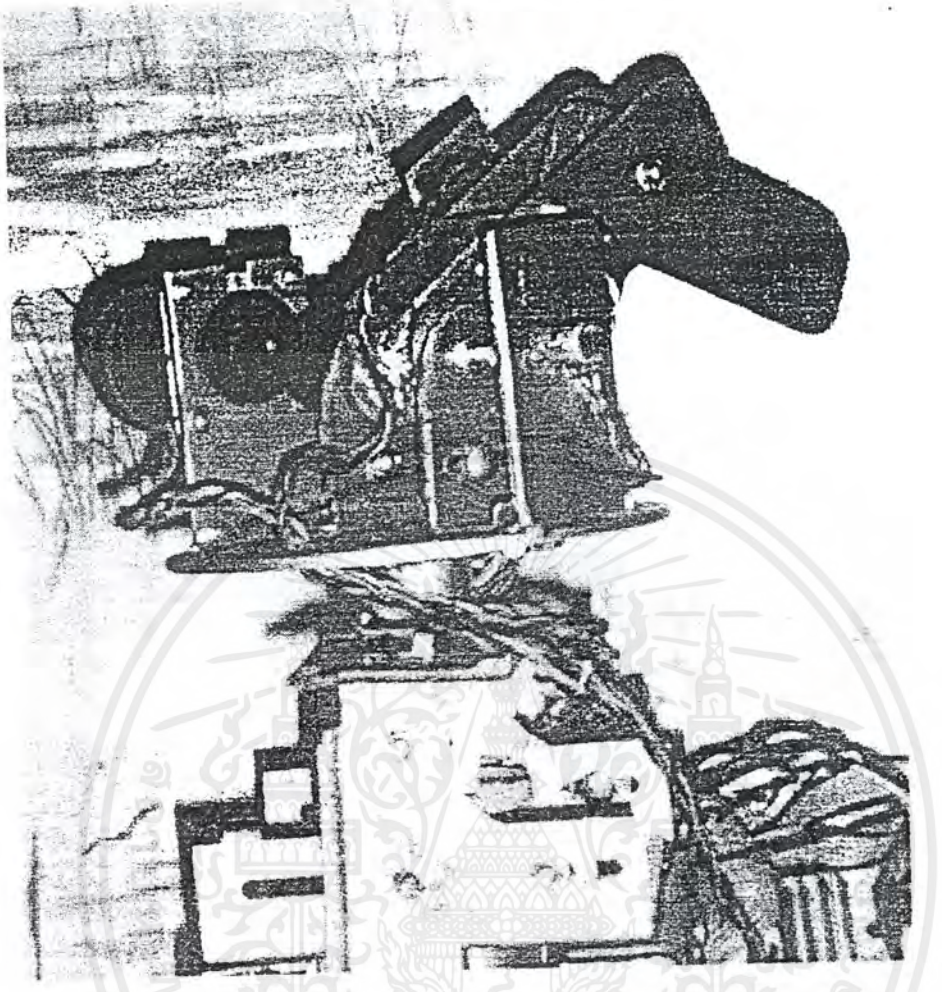
7. พิชิต ลำยอง , เครื่องจักรกลไฟฟ้า 1 , ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีเจ้าคุณทหารลาดกระบัง , 2537 , หน้าที่ 107 – 111

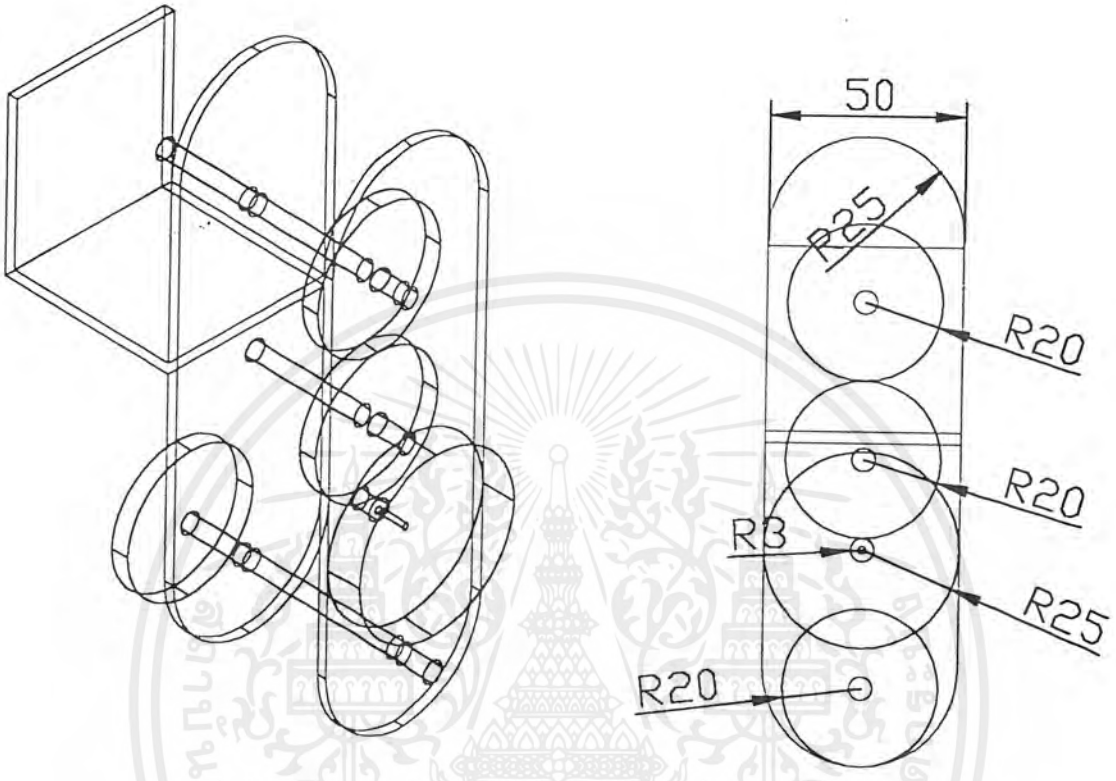
8. วินัย ชนปรมัตถ์ , เซมิคอนดักเตอร์อิเลคทรอนิคส์ การเคลื่อนไหวของหุ่นยนต์ , บริษัท ซีเอ็ดยูเคชั่น , 2541 , หน้าที่ 228 – 233

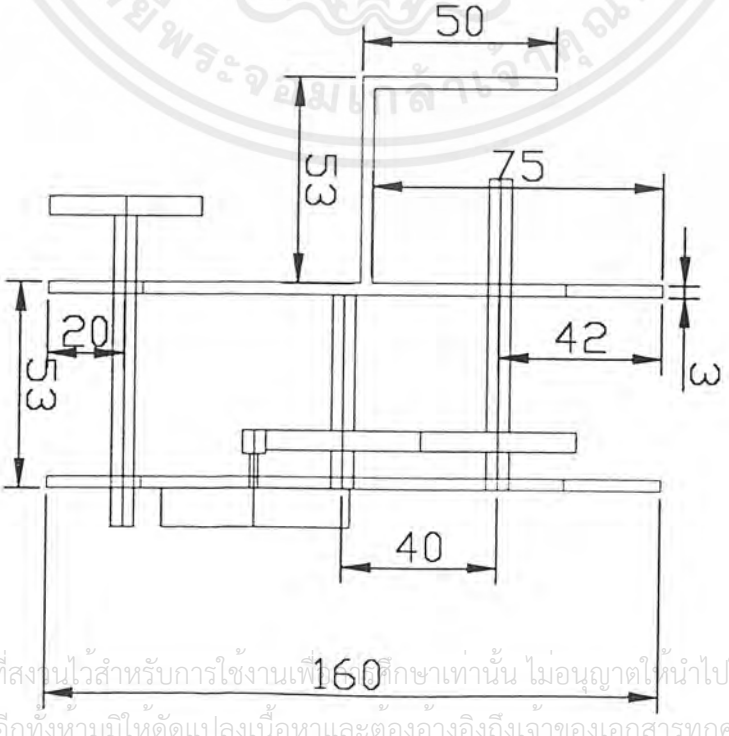
9. Jan Axelson , Serial Port Complete , Ladeview Research Madison USA , 1998 , p. 45 - 90

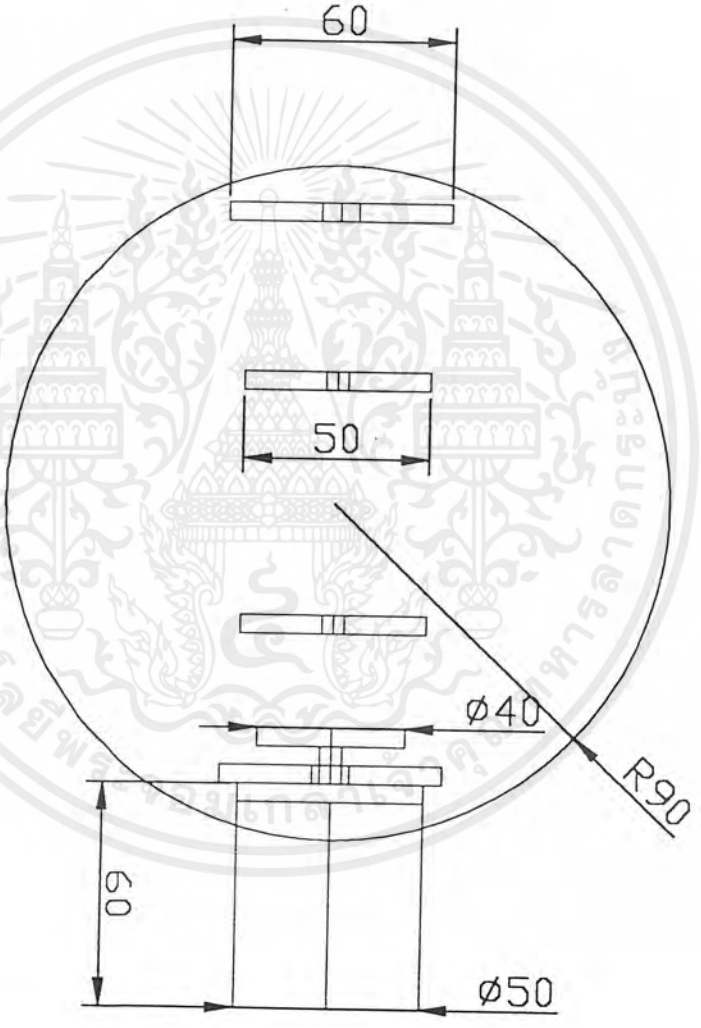
ภาคผนวก

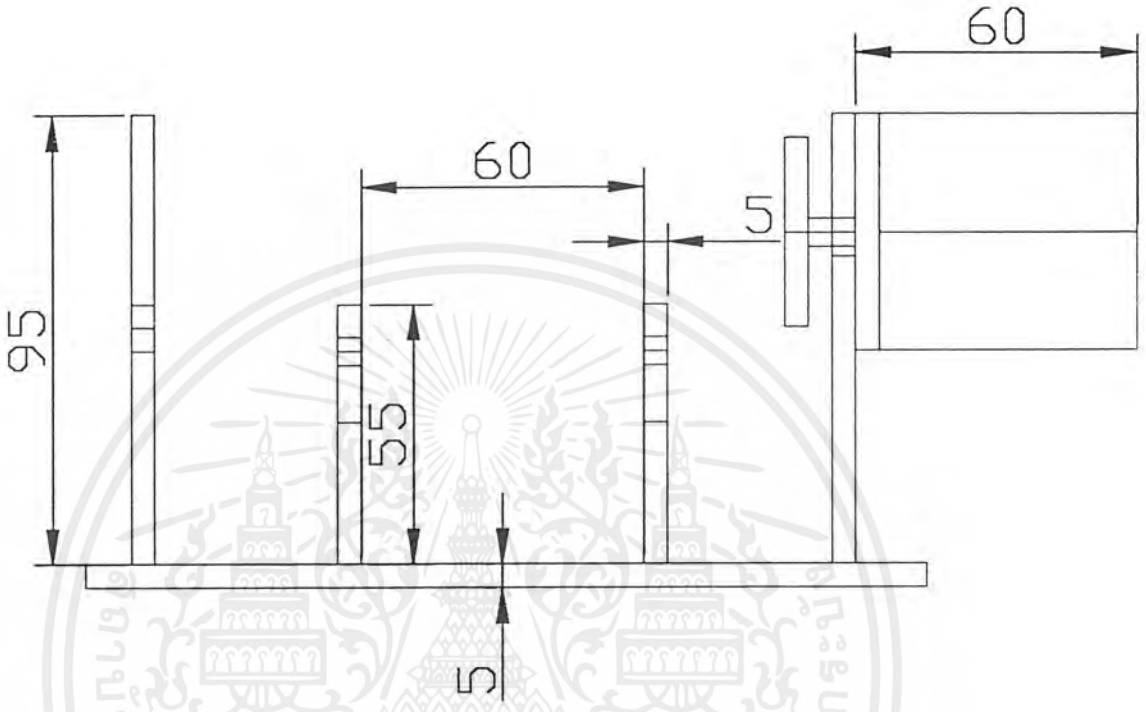
auto221.jpg (450x489x256 jpeg)

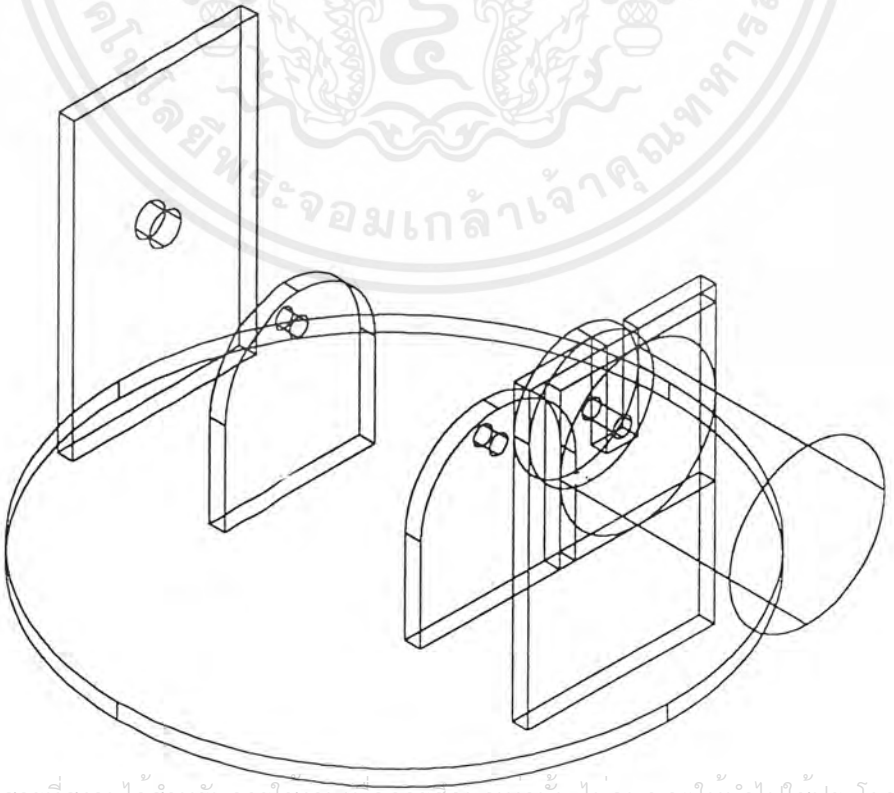












โปรแกรมแอสเซมบลี้

```
                ORG     0000H
                jmp     config
                org     0003h
                jmp     input
                org     100h
config:         MOV     TMOD,#20H       ;BAUD RATE = 9.6K
                MOV     SCON,#50H       ;SERIAL MODE1
                MOV     TH1,#0FDH
                SETB    TR1
                Setb    ea
                Setb    ex0
                MOV     PSW,#00H        ;used register bank0
                mov     sp,#60h
                MOV     R0,#02H
                MOV     R1,#10H
                MOV     R2,#02H
                MOV     R3,#10H
                Mov     r6,#02h
                Mov     r7,#02h
                MOV     A,#80H
                MOV     DPTR,#8003H
                MOVX    @DPTR,A
    ;***************CHECK MODE**************
check:          jbc     ri,select
                lcall   stable
                jmp     check
select:         mov     a,sbuf
;               mov     r4,a
                jmp     mode
```

```
mode:       cjne    a,#08,mode2
            jmp     mode1
;           jb      p2.1,mode1
;           jb      p2.3,mode2
;           lcall   stable
;           jmp     display
;           jmp     check
mode1:      jmp     manual
mode2:      cjne    a,#03,return
            jmp     auto
return:     lcall   stable
            jmp     display
            jmp     check
    ;***************MANUAL*****************
MANUAL:     MOV     PSW,#00H    ;used register bank0
            MOV     R0,#02H
            MOV     R1,#10H
            MOV     R2,#02H
            MOV     R3,#10H
            Mov     r6,#02h
            mov     r7,#02h
;           jmp     display
            JMP     BEGIN
BEGIN:      JBC     RI,RECIVE   ;recive data
            Lcall   stable
            JMP     BEGIN
    ;***************RECIVE DATA************
RECIVE:     MOV     A,SBUF
            MOV     P2,A
```

```
                JMP     SUBMAIN
    ;***************CHECKPUSH*******************
SUBMAIN:        JNB     P2.0,TURN_R1   ;1ST_RIGHT_PUSH
                JNB     P2.1,TURN_L1   ;1ST_LEFT_PUSH
                JNB     P2.2,TURN_L2   ;2ND_RIGHT_PUSH
                JNB     P2.3,TURN_R2   ;2ND_LEFT_PUSH
                JNB     P2.4,TURN_R3   ;3RD_RIGTH_PUSH
                JNB     P2.5,TURN_L3   ;3RD_LEFT_PUSH
                JNB     P2.7,change    ;change mode to auto
;               jmp     display
                JMP     BEGIN
change:         jmp     auto
    ;***************1ST_MOTOR*******************
TURN_R1:        CJNE    R0,#20H,TURN_R11
                MOV     R0,#02H
TURN_R11:       MOV     DPTR,#8000H
                MOV     A,R0        ;RIGHT_STEP
                MOV     R6,A
                MOVX    @DPTR,A
                LCALL   DELAY
                RL      A
                MOV     R0,A
                LCALL   STOP1
                JMP     turn_r1
TURN_L1:        CJNE    R1,#01H,TURN_L11
                MOV     R1,#10H
TURN_L11:       MOV     DPTR,#8000H
                MOV     A,R1        ;LEFT_STEP
                MOV     R6,A
                MOVX    @DPTR,A
```

```
                LCALL DELAY
                RR    A
                MOV   R1,A
                LCALL STOP2
                JMP   turn_l1
    ;***************2ND_MOTOR*******************
TURN_R2:    CJNE  R2,#20H,TURN_R22
                MOV   R2,#02H
TURN_R22:   MOV   DPTR,#8001H
                MOV   A,R2        ;RIGHT_STEP
                MOV   R7,A
                MOVX  @DPTR,A
                LCALL DELAY
                RL    A
                MOV   R2,A
                LCALL STOP3
                JMP   turn_r2
TURN_L2:    CJNE  R3,#01H,TURN_L22
                MOV   R3,#10H
TURN_L22:   MOV   DPTR,#8001H
                MOV   A,R3        ;LEFT_STEP
                MOV   R7,A
                MOVX  @DPTR,A
                LCALL DELAY
                RR    A
                MOV   R3,A
                LCALL STOP4
                JMP   turn_l2
    ;***************3RD_MOTOR*****************
TURN_R3:    MOV   DPTR,#8002H
```

```
                MOV     A,#01H      ;TURN RIGHT
                MOVX    @DPTR,A
                LCALL   STOP
                JNB     P2.4,TURN_R3
                CLR     A
                MOVX    @DPTR,A
                JMP     BACK
TURN_L3:        MOV     DPTR,#8002H
                MOV     A,#02H      ;TURN LEFT
                MOVX    @DPTR,A
                LCALL   STOP
                JNB     P2.5,TURN_L3
                CLR     A
                MOVX    @DPTR,A
                JMP     BACK
    ;**************CHECK_STOP*************
STOP:           JBC     RI,RECIVE1
                RET
    ;**************RECIVE DATA*************
RECIVE1:        MOV     A,SBUF
                MOV     P2,A
                RET
    ;**************CHECK_STOP*************
STOP1:          JBC     RI,RECIVE2
                RET
    ;**************RECIVE DATA*************
RECIVE2:        MOV     A,SBUF
                MOV     P2,A
                jb      p2.0,back
                RET
```

```
    ;***************CHECK_STOP**************
STOP2:      JBC     RI,RECIVE3
            RET
    ;***************RECIVE DATA*************
RECIVE3:    MOV     A,SBUF
            MOV     P2,A
            jb      p2.1,back
            RET
    ;***************CHECK_STOP**************
STOP3:      JBC     RI,RECIVE4
            RET
    ;***************RECIVE DATA*************
RECIVE4:    MOV     A,SBUF
            MOV     P2,A
            jb      p2.2,back
            RET
    ;***************CHECK_STOP**************
STOP4:      JBC     RI,RECIVE5
            RET
    ;***************RECIVE DATA*************
RECIVE5:    MOV     A,SBUF
            MOV     P2,A
            jb      p2.3,back
            RET
BACK:       JMP     display
    ;***************MANUAL DISPLAY**************
DISPLAY:    push    psw
            MOV     PSW,#08H      ;use register bank1
            JMP     display1
    ;***************DISPLAY*****************
```

```
display1:   mov   r6,#20h      ;set address
            mov   dptr,#8002h
display2:   mov   a,r6
            MOVX  @DPTR,A
            add   a,#08h       ;start
            MOVX  @DPTR,A
            mov   a,r6
            MOVX  @DPTR,A      ;jump  interrupt ex0
            mov   r3,a
            jmp   display3
display3:   mov   r6,#40h
            mov   dptr,#8002h
            mov   a,r6
            MOVX  @DPTR,A
            add   a,#08h
            MOVX  @DPTR,A
            mov   a,r6
            MOVX  @DPTR,A      ;jump  interrupt ex0
            mov   r4,a
            jmp   display4
display4:   mov   r6,#80h
            mov   dptr,#8002h
            mov   a,r6
            MOVX  @DPTR,A
            add   a,#08h
            MOVX  @DPTR,A
            mov   a,r6
            MOVX  @DPTR,A      ;jump  interrupt ex0
            mov   r5,a
            lcall   move
```

```
                jmp     clear1
clear1:         pop     psw
                jmp     begin
    ;***************AUTO****************
auto:           push    psw
                mov     psw,#08h        ;used register bank1
                jmp     data1
data1:          jbc     ri,input1       ;input 1st data
                jmp     data1
input1:         mov     a,sbuf
                mov     r0,a            ;keep data in r0
                jmp     data2
data2:          jbc     ri,input2       ;input 2nd data
                jmp     data2
input2:         mov     a,sbuf
                mov     r1,a            ;keep data in r1
                jmp     data3
data3:          jbc     ri,input3       ;input 3rd data
                jmp     data3
input3:         mov     a,sbuf
                mov     r2,a            ;keep data in r2
                jmp     compare1
    ;***************COMPARE*************
compare1:       clr     c
                mov     a,r0
                subb    a,r3
                jnc     low1            ;carry is 0
                lcall   left1           ;carry is 1
                lcall   show1
                jmp     compare1
```

```
low1:       cjne    a,#00h,high1    ;not macth
            jmp     compare2
high1:      lcall   rigth1
            lcall   show1
            jmp     compare1
compare2:   clr     c
            mov     a,r1
            subb    a,r4
            jnc     low2            ;carry is 0
            lcall   left2           ;carry is 1
            lcall   show2
            jmp     compare2
low2:       cjne    a,#00h,high2    ;not macth
            jmp     compare3
high2:      lcall   rigth2
            lcall   show2
            jmp     compare2
compare3:   clr     c
            mov     a,r2
            subb    a,r5
            jnc     low3            ;carry is 0
            lcall   left3           ;carry is 1
            lcall   show3
            jmp     compare3
low3:       cjne    a,#00h,high3    ;not macth
            jmp     clear
high3:      lcall   rigth3
            lcall   show3
            jmp     compare3
```

```
    ;***************AUTO DRIVE**************
    ;***************1ST MOTOR***************
left1:      push    psw             ;DC MOTOR
            mov     psw,#00h
            MOV     DPTR,#8002H
            MOV     A,#02H          ;TURN LEFT
            MOVX    @DPTR,A
            LCALL   delay
            CLR     A
            MOVX    @DPTR,A
            pop     psw
            ret
rigth1:     push    psw
            mov     psw,#00h
            MOV     DPTR,#8002H
            MOV     A,#01H          ;TURN RIGHT
            MOVX    @DPTR,A
            LCALL   delay
            CLR     A
            MOVX    @DPTR,A
            pop     psw
            ret
    ;***************2ND MOTOR**************
left2:      push    psw
            mov     psw,#00h
            CJNE    R3,#01H,left22
            MOV     R3,#10H
left22:     MOV     DPTR,#8001H
            MOV     A,R3            ;LEFT_STEP
            MOV     R7,A
```

```
                MOVX  @DPTR,A
                LCALL DELAY
                RR    A
                MOV   R3,A
                pop   psw
                ret
rigth2:         push  psw
                mov   psw,#00h
                CJNE  R2,#20H,rigth22
                MOV   R2,#02H
rigth22:        MOV   DPTR,#8001H
                MOV   A,R2          ;RIGHT_STEP
                MOV   R7,A
                MOVX  @DPTR,A
                LCALL DELAY
                RL    A
                MOV   R2,A
                pop   psw
                ret
    ;***************3RD MOTOR***************
left3:          push  psw
                mov   psw,#00h
                CJNE  R1,#01H,left33
                MOV   R1,#10H
left33:         MOV   DPTR,#8000H
                MOV   A,R1          ;LEFT_STEP
                MOV   R6,A
                MOVX  @DPTR,A
                LCALL DELAY
                RR    A
```

```
                MOV   R1,A
                pop   psw
                ret
rigth3:         push  psw
                mov   psw,#00h
                CJNE  R0,#20H,rigth33
                MOV   R0,#02H
rigth33:        MOV   DPTR,#8000H
                MOV   A,R0          ;RIGHT_STEP
                MOV   R6,A
                MOVX  @DPTR,A
                LCALL DELAY
                RL    A
                MOV   R0,A
                pop   psw
                ret
    ;***************AUTO SHOW***************
show1:          mov   r6,#20h       ;set address
                mov   dptr,#8002h
                mov   a,r6
                MOVX  @DPTR,A
                add   a,#08h        ;start
                MOVX  @DPTR,A
                mov   a,r6
                MOVX  @DPTR,A       ;jump  interrupt ex0
                mov   r3,a
                lcall move
                ret
show2:          mov   r6,#40h
                mov   dptr,#8002h
```

```
                mov     a,r6
                MOVX  @DPTR,A
                add     a,#08h
                MOVX  @DPTR,A
                mov     a,r6
                MOVX  @DPTR,A       ;jump  interrupt ex0
                mov     r4,a
                lcall   move
                ret
show3:          mov     r6,#80h
                mov     dptr,#8002h
                mov     a,r6
                MOVX  @DPTR,A
                add     a,#08h
                MOVX  @DPTR,A
                mov     a,r6
                MOVX  @DPTR,A       ;jump  interrupt ex0
                mov     r5,a
                lcall   move
                ret
    ;***************interrupt_ex0***********
input:          mov     a,#10h              ;set oe
                mov     dptr,#8002h
                movx    @dptr,a
                lcall   delay5
                mov     a,p1                ;recive 8bit data
                reti
    ;***************MOVE DATA***************
move:           mov     a,r3
                lcall   trans
```

```
                call    delay
                 mov    a,r4
                lcall   trans
                lcall   delay
                mov     a,r5
                lcall   trans
                lcall   delay
                 ret
    ;***************TRANSFER DATA***********
trans:          clr     ti
                clr     es
                mov     sbuf,a
trans1:         jnb     ti,trans1
                ret
    ;***************FINISH**************
clear:          pop     psw
                 jmp    finish
finish:         jmp     check
    ;***************STABLE**************
STABLE:         mov     psw,#00h
                MOV     DPTR,#8000H
                CLR     A
                MOVX    @DPTR,A
                LCALL   DELAY_B
                MOV     A,R6
                MOVX    @DPTR,A
                LCALL   DELAY_A
                MOV     DPTR,#8001H
                CLR      A
                MOVX    @DPTR,A
```

```
                LCALL DELAY_B
                MOV   A,R7
                MOVX  @DPTR,A
                LCALL DELAY_A
                RET
    ;***************DELAY***************
DELAY:          push  psw
                mov   psw,#00h
                MOV   R4,#0FFH
                MOV   R5,#070H
DELAY1:         DJNZ  R4,DELAY1
                DJNZ  R5,DELAY1
                pop   psw
                RET
DELAY_A:        push  psw
                mov   psw,#00h
                MOV   R4,#0FFH
                MOV   R5,#06H
DELAY2:         DJNZ  R4,DELAY2
                DJNZ  R5,DELAY2
                pop   psw
                RET
DELAY_B:        push  psw
                mov   psw,#00h
                MOV   R4,#0FFH
                MOV   R5,#08H
DELAY3:         DJNZ  R4,DELAY3
                DJNZ  R5,DELAY3
                pop   psw
                RET
```

```
DELAY5:   push   psw
          mov    psw,#00h
          MOV    R4,#40H
DELAY6:   DJNZ   R4,DELAY6
          pop    psw
          RET
```

โปรแกรมวิชวลเบสิก

Form1 หน้าจอควบคุมการทำงานหลัก

```
Option Explicit

Private Sub Command13_Click()
On Error GoTo handleerror:
Unload Me
Unload Form2
Unload Form3
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub

Private Sub Form_Activate()
On Error GoTo handleerror:
Form2.Cls
Form3.Cls
Frame1.Enabled = True
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub

Private Sub Form_Load()
On Error GoTo handleerror:
With MSComm1
'Set bitrate = 9600
.Settings = "9600,n,8,1"
```

```
'Set number of byte to read=1.
.InputLen = 1
'Set port1 to open.
.PortOpen = True
'Type of data is text.
.InputMode = comInputModeText
'To detect when data has arrived at a port1.
.RThreshold = 1
'To set the number of characters in the transmit buffer that will trigger an OnComm event.
.SThreshold = 1
'Ask for a start position.
MSComm1.Output = Chr$(&HFF)
End With
' เลือกใช้ Workspace(0)
Set TablenoteWs = DBEngine.Workspaces(0)
' เปิดฐานข้อมูล tablenote.mdb ทำการแสดงตารางทั้งหมด
Set TablenoteDb = TablenoteWs.OpenDatabase("D:\data\#4\johnny\project\vb6\tablenote.mdb",
False, False)
c = 0
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub

Private Sub MSComm1_OnComm()
On Error GoTo handleerror:
'Returns the most recent events or error.
'When amount of bytes >= 1 have arrives.
If MSComm1.CommEvent = 2 And MSComm1.InBufferCount > 0 Then
'To define that which motor that this data comes from.
```

```
    buffer = MSComm1.Input
    c = c - 1
    Call hello(c)
    If c = 3 Then
      c = 0
    End If
  End If
  MSComm1.InBufferCount = 0
  Exit Sub
  handleerror:
        MsgBox Error(Err.Number)
  End Sub

  Public Sub hello(ByVal c As Integer)
  On Error GoTo handleerror:
  'Show position for dc motor
  If c = 1 Then
    value1 = Asc(buffer)
  'Show position for step1 motor
    ElseIf c = 2 Then
        value2 = Asc(buffer)
   'Show position for step1 motor
        ElseIf c = 3 Then
            value3 = Asc(buffer)
  End If
  'ตัวแปร send เอาไว้เก็บค่าที่ส่งออกหรือรับมา
  send1 = value1
  send2 = value2
  send3 = value3
  'ค่ามุมของ DC
```

```
z0 = send1 * 1.133
'ค่ามุมของ S1
x0 = (send2 * 1.133)
x0 = x0 - 39.655
x0 = x0 * pi / 180
'ค่ามุมของ S2
y0 = (send3 * 1.133)
y0 = y0 * pi / 180
'ค่าของแขนท่อนที่1
x1 = 15 * Cos(x0)
y1 = 15 * Sin(x0)
'ค่ามุมที่ DC หมุนไป
position1 = z0
'ค่ารัศมี
position2 = x1 - 15 * Cos(x0 + y0)
'ค่าความสูง
position3 = y1 + 15 * Sin(x0 + y0)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command1_Click()
'On Error GoTo handleerror:
If Option1.Value = True Then
   MSComm1.Output = Chr$(&H8)
   Form2.Show
   Form2.Text1.Text = position2
   Form2.Text2.Text = position3
   Form2.Text3.Text = position1
```

```
    send1 = value1
    send2 = value2
    send3 = value3
    Form2.Text4.Text = send1
    Form2.Text5.Text = send2
    Form2.Text6.Text = send3
    ElseIf Option2.Value = True Then
        MSComm1.Output = Chr$(&H3)
        Form3.Show
        Form3.Text5.Text = position2
        Form3.Text6.Text = position3
        Form3.Text7.Text = position1
        send1 = value1
        send2 = value2
        send3 = value3
End If
Frame1.Enabled = False
Option1.Enabled = True
Option2.Enabled = True
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub
```

Form2 หน้าจอการควบคุมแบบ Manual Mode

```
Option Explicit

Private Sub Command21_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้DCหมุนไปทางซ้าย
Form1.MSComm1.Output = Chr$(&HDF)
'Form2.Cls
'Call draw(send1, send2, send3, position2, position3)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command22_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้DCหมุนไปทางขวา
Form1.MSComm1.Output = Chr$(&HEF)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command23_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้S1หมุนขึ้น
Form1.MSComm1.Output = Chr$(&HF7)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
```

```
End Sub
Private Sub Command24_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้S1หมุนลง
Form1.MSComm1.Output = Chr$(&HFB)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command25_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้S2หมุนขึ้น
Form1.MSComm1.Output = Chr$(&HFE)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command26_Click()
On Error GoTo handleerror:
'ส่งค่าออกไปควบคุมให้S2หมุนลง
Form1.MSComm1.Output = Chr$(&HFD)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub
```

```
Private Sub Command27_Click()
On Error GoTo handleerror:
'ส่งออกไปให้หยุดหมุนและส่งค่าตำแหน่งกลับมาแสดง
Form1.MSComm1.Output = Chr$(&HFF)
'ถ้ามีค่าเข้ามาให้สร้างรูปใหม่
If Form1.MSComm1.CommEvent = 2 Then
  Call draw(send1, send2, send3, position1, position2, position3)
End If
Call draw(send1, send2, send3, position1, position2, position3)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command28_Click()
'เปลี่ยนโหมดการทำงานไปเป็นแบบกำหนดค่าให้แขนกลโดยตรง
'ลบรูปที่แสดงไว้เมื่อกี้
On Error GoTo handleerror:
Form2.Cls
Form2.Hide
Form3.Show
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
'ส่งค่าออกไปบอกภายนอกว่าทำการเปลี่ยนโหมดการควบคุมไปเป็นauto
Form1.MSComm1.Output = Chr$(&H7F)
```

```
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command29_Click()
'ผู้ใช้ต้องการบันทึกข้อความทำการเรียกการตรวจสอบและบันทึกหากค่าไม่ซ้ำกับ
ข้อมูลที่มีอยู่แล้วในตาราง
On Error GoTo handleerror:
'ใช้เมดทอด Addnew เพื่อเพิ่มข้อมูลเรคคอร์ดใหม่
Call checktable
Command29.Enabled = False
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command210_Click()
On Error GoTo handleerror:
'ปิดหน้าจอที่แสดงรูปนี้
'ลบค่าที่แสดงไว้ออกให้หมด
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Form2.Cls

Form1.MSComm1.Output = Chr$(&H0)
```

```
'ทำการลบรูปที่เคยวาดทั้งหมดออก
Form2.Hide
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Form_Activate()
On Error GoTo handleerror:
Form1.MSComm1.Output = Chr$(&HFF)
Call draw(send1, send2, send3, position1, position2, position3)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Form_Load()
On Error GoTo handleerror:
'แสดงค่า radius , height , degree ให้เห็นเป็นตัวเลขกำกับไว้
' เลือกใช้ Workspace(0)
Set TablenoteWs = DBEngine.Workspaces(0)
' เปิดฐานข้อมูล tablenote.mdb ทำการแสดงตารางทั้งหมด
Set TablenoteDb = TablenoteWs.OpenDatabase("D:\data\#4\johnny\project\vb6\tablenote.mdb",
False, False)
Set Table1 = TablenoteDb.OpenRecordset("Select * from table1 order by radius,height,degree",
dbOpenDynaset)
Call draw(send1, send2, send3, position1, position2, position3)
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
```

```
End Sub

Private Sub draw(ByVal send1 As Integer, send2 As Integer, send3 As Integer, position1 As
Single, position2 As Single, position3 As Single)
On Error GoTo handleerror:
'ถ้าเลือกการควบคุมแบบ manual
Form2.Cls
DrawMode = vbNotXorPen
 'ค่ามุมของ DC
z0 = send1 * 1.133
z0 = z0 * pi / 180
'ค่ามุมของ S1
x0 = send2 * 1.133
x0 = x0 - 39.655
x0 = x0 * pi / 180
'ค่ามุมของ S2
y0 = send3 * 1.133
 y0 = y0 * pi / 180
'ค่าของแขนท่อนที่1
'Create picture for x-y plane.
x2 = 2 * Cos(x0)
y2 = 2 * Sin(x0)
r1 = 4
h1 = 4
r2 = x2 * Cos(z0)
r2 = r2 + 4
h2 = 4 - (y2)
Line (r1, h1)-(r2, h2), RGB(255, 0, 255)
Line (3.75, 4)-(4.25, 4)
r3 = position2 / 7.5
```

```
r3 = r3 - x2
r3 = r3 * Cos(z0)
r3 = r3 + r2
h3 = (position3) / 7.5
h3 = 4 - h3
Line (r2, h2)-(r3, h3), RGB(255, 255, 0)
'Create picture for z-y plane.
r4 = 4
h4 = 8
x3 = 2 * Cos(x0)
y3 = 2 * Sin(x0)
r5 = x3 * Cos(z0)
r5 = 4 - r5
h5 = 8 - (y3)
r6 = position2 / 7.5
r6 = r6 - x3
r6 = r6 * Sin(z0)
r6 = r5 - r6
h6 = (position3) / 7.5
h6 = 8 - h6
Line (r4, h4)-(r5, h5), RGB(0, 44, 100)
Line (3.75, 8)-(4.25, 8)
Line (r5, h5)-(r6, h6), RGB(40, 128, 255)
Circle (4, 12), 1.5, RGB(18, 59, 150)
Line (4, 12)-(5.5, 12), RGB(18, 59, 150)
'ค่าของแขนท่อนที่1
x1 = 15 * Cos(x0)
y1 = 15 * Sin(x0)
```

```
'ค่ามุมที่ DC หมุนไป
position1 = z0
'ค่ารัศมี
position2 = x1 - 15 * Cos(x0 + y0)
'ค่าความสูง
position3 = y1 + 15 * Sin(x0 + y0)
Text1.Text = position2
Text2.Text = position3
Text3.Text = position1
Text4.Text = send1
Text5.Text = send2
Text6.Text = send3
If position2 < 0 And z0 > pi Then
  z0 = z0 - pi
End If
r7 = 1.5 * Cos(z0)
h7 = 1.5 * Sin(z0)
Line (4, 12)-(4 - r7, 12 + h7), RGB(240, 169, 10)
r1 = 0: r2 = 0: r3 = 0: h1 = 0: h2 = 0: h3 = 0
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub save()
On Error GoTo handleerror:
'บันทึกค่าต่างๆลงในตาราง
Table1.AddNew
Table1("radius") = Text1.Text
Table1("height") = Text2.Text
```

```
Table1("degree") = Text3.Text
Table1("DC position") = Text4.Text
Table1("S1 position") = Text5.Text
Table1("S2 position") = Text6.Text
Table1.Update
Table1.Requery
MsgBox "บันทึกข้อมูลเรียบร้อยแล้ว"
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub checktable()
On Error GoTo handleerror:
'ตรวจว่าในตารางนี้มีข้อมูลที่มีค่าเท่ากับเท่านี้อยู่หรือยัง?
position2 = Text1.Text
position3 = Text2.Text
position1 = Text3.Text
'เริ่มจากการตรวจค่าradius
Table1.FindFirst ("radius=" & position2 & "")
'ถ้าค่าไม่ซ้ำให้ทำการเรียกฟังก์ชั่นที่ทำหน้าที่บันทึกข้อมูล
If Table1.NoMatch Then
 Call save
 Else
   'ตรวจค่าheight
   Table1.FindFirst ("height=" & position3 & "")
   'ถ้าค่าไม่ซ้ำให้ทำการเรียกฟังก์ชั่นที่ทำหน้าที่บันทึกข้อมูล
   If Table1.NoMatch Then
    Call save
    Else
```

```
        'ตรวจค่า degree
        Table1.FindFirst ("degree=" & position1 & "")
        'ถ้าค่าไม่ซ้ำให้ทำการเรียกฟังก์ชั่นที่ทำหน้าที่บันทึกข้อมูล
        If Table1.NoMatch Then
          Call save
          'ถ้าค่าซ้ำให้แสดงข้อความว่ามีข้อมูลบันทึกอยู่แล้ว
          Else
            MsgBox "มีข้อมูลนี้อยู่แล้ว"
        End If
    End If
End If
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub
```

Form3 หน้าจอควบคุมแบบ Auto Mode

```
Option Explicit


Private Sub Form_Activate()
On Error GoTo handleerror:
Set Table1 = TablenoteDb.OpenRecordset("Select * from table1 order by radius,height,degree",
dbOpenDynaset)
Set Data2.Recordset = Table1
Call draw(send1, send2, send3, position1, position2, position3)
If Form1.MSComm1.CommEvent = 2 Then
  Call draw(send1, send2, send3, position1, position2, position3)
End If
Text1.SetFocus
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub


Private Sub Form_Load()
On Error GoTo handleerror:
Set Table1 = TablenoteDb.OpenRecordset("Select * from table1 order by radius,height,degree",
dbOpenDynaset)
Set Data2.Recordset = Table1
Text5.Text = position2
Text6.Text = position3
Text7.Text = position1
Call draw(send1, send2, send3, position1, position2, position3)
Exit Sub
handleerror:
```

```
        MsgBox Error(Err.Number)
End Sub

Private Sub Command31_Click()
Dim a As String
Dim a1 As Single
'ทำการดึงข้อมูลที่ผู้ใช้ต้องการออกมาจากฐานข้อมูล แล้วส่งออกไปควบคุมมอเตอร์
On Error GoTo handleerror:
a = Text1.Text
'ถ้ามีการกรอกค่าที่ต้องการเรียบร้อยแล้วให้ค้นหาได้
If a <> "" Then
   a1 = Val(a)
   Table1.FindFirst ("radius=" & a1 & "")
   If Table1.NoMatch Then
      MsgBox "ไม่มีข้อมูลนี้อยู่ในตาราง"
      Frame31.Enabled = True
      Text1.Text = ""
      Text1.SetFocus
      Exit Sub
   End If
   ' แสดงค่าต่างๆลงตามช่องที่กำหนดไว้
   Call displayfields
End If
'นำค่าที่ได้จากตารางส่งออกทางเอาต์พุท
   Form1.MSComm1.Output = Chr$(send1)
   Form1.MSComm1.Output = Chr$(send2)
   Form1.MSComm1.Output = Chr$(send3)
'ทำเพื่อให้รับการควบคุมหลังบังคับแบบ auto ได้แค่ครั้งเดียว
'ให้ไปรอรับคำสั่งใหม่ตั้งแต่การเลือกโหมดการควบคุม
Frame31.Enabled = False
```

```
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub displayfields()
On Error GoTo handleerror
'แสดงค่าข้อมูลชุดที่เราเลือก
Text2.Text = Table1.Fields("radius")
Text3.Text = Table1.Fields("height")
Text4.Text = Table1.Fields("degree")
'นำค่ามาเก็บไว้ส่งให้controllertเป็นเลข 0-255
send1 = Table1.Fields("DC position")
send2 = Table1.Fields("S1 position")
send3 = Table1.Fields("S2 position")
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub

Private Sub Text1_Change()
On Error GoTo handleerror:
Command31.Enabled = True
Exit Sub
handleerror:
    MsgBox Error(Err.Number)
End Sub

Private Sub draw(ByVal send1 As Integer, send2 As Integer, send3 As Integer, position1 As
Single, position2 As Single, position3 As Single)
```

```
On Error GoTo handleerror:
'ถ้าเลือกการควบคุมแบบ auto
Form3.Cls
DrawMode = vbNotXorPen
  'ค่ามุมของ DC
z0 = send1 * 1.133
z0 = z0 * pi / 180
'ค่ามุมของ S1
x0 = send2 * 1.133
x0 = x0 - 39.655
x0 = x0 * pi / 180
'ค่ามุมของ S2
y0 = send3 * 1.133
y0 = y0
y0 = y0 * pi / 180
'ค่าของแขนท่อนที่1
'Create picture for x-y plane.
x2 = 2 * Cos(x0)
y2 = 2 * Sin(x0)
r1 = 4
h1 = 4
r2 = x2 * Cos(z0)
r2 = r2 + 4
h2 = 4 - (y2)
Line (r1, h1)-(r2, h2), RGB(255, 0, 255)
Line (3.75, 4)-(4.25, 4)
r3 = position2 / 7.5
r3 = r3 - x2
r3 = r3 * Cos(z0)
r3 = r3 + r2
```

```
h3 = (position3) / 7.5
h3 = 4 - h3
Line (r2, h2)-(r3, h3), RGB(255, 255, 0)
'Create picture for z-y plane.
r4 = 4
h4 = 8
x3 = 2 * Cos(x0)
y3 = 2 * Sin(x0)
r5 = x3 * Cos(z0)
r5 = 4 - r5
h5 = 8 - (y3)
r6 = position2 / 7.5
r6 = r6 - x3
r6 = r6 * Sin(z0)
r6 = r5 - r6
h6 = (position3) / 7.5
h6 = 8 - h6
Line (r4, h4)-(r5, h5), RGB(0, 44, 100)
Line (3.75, 8)-(4.25, 8)
Line (r5, h5)-(r6, h6), RGB(40, 128, 255)
Circle (4, 12), 1.5, RGB(18, 59, 150)
Line (4, 12)-(5.5, 12), RGB(18, 59, 150)
'ค่าของแขนท่อนที่1
x1 = 15 * Cos(x0)
y1 = 15 * Sin(x0)
'ค่ามุมที่ DC หมุนไป
position1 = z0
'ค่ารัศมี
position2 = x1 - 15 * Cos(x0 + y0)
'ค่าความสูง
```

```
position3 = y1 + 15 * Sin(x0 + y0)
Text5.Text = position2
Text6.Text = position3
Text7.Text = position1
If position2 < 0 And z0 > pi Then
  z0 = z0 - pi
End If
r7 = 1.5 * Cos(z0)
h7 = 1.5 * Sin(z0)
Line (4, 12)-(4 + r7, 12 + h7), RGB(240, 169, 10)
r1 = 0: r2 = 0: r3 = 0: h1 = 0: h2 = 0: h3 = 0
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub

Private Sub Command32_Click()
On Error GoTo handleerror:
Frame31.Enabled = True
Form3.Hide
Form1.Show
Exit Sub
handleerror:
        MsgBox Error(Err.Number)
End Sub
```