การควบคุมดีซีมอเตอร์และสเตปมอเตอร์ด้วยไมโครโปรเซสเซอร์

DC MOTOR & STEP MOTOR CONTROL WITH MICROPROCESSOR

โดย

นายอัศวิน ปานขาว

นายอาคม หึมวัง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็คทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์ การควบคุมดีซีมอเตอร์และสเตปมอเตอร์ด้วยไมโครโปรเซสเซอร์
Dc motor & Step motor control with Microprocessor

ชื่อนักศึกษา นายอัศวิน ปานขาว รหัส 40013319
นายอาคม หึมวัง รหัส 40013320
อาจารย์ที่ปรึกษา อาจารย์มนชนก ศรีเสือขาม
ภาควิชา เทคนิคอุตสาหกรรม
คณะ วิศวกรรมศาสตร์
ปีการศึกษา 2542

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

....
( ) ( )

....
( ) ( )

....
( ) ( )

บทคัดย่อ

โครงงานฉบับนี้เสนอรายละเอียดเกี่ยวกับการศึกษาระบบการควบคุมดีซีมอเตอร์และสเตปปิ้งมอเตอร์ด้วยไมโครคอนโทรลเลอร์เพื่อการประยุกต์ใช้งาน โดยเราสามารถใช้ดีซีมอเตอร์และสเตปปิ้งมอเตอร์ประกอบกันขึ้นเพื่อประยุกต์ใช้งานในลักษณะการทำงานเป็นหุ่นยนต์สำหรับจับชิ้นงานและสำรวจสิ่งแวดล้อมได้ ซึ่งสามารถควบคุมได้ทั้งแบบอัตโนมัติและแบบบังคับด้วยมือผ่านคีย์สวิตช์ โดยการใช้ไมโครคอนโทรลเลอร์ MCS-51 เป็นตัวประมวลผล

Abstract

This project present the control dc motor and stepping motor by microcontroller for application in building a Robot. This Robot can catch up things and surround environment. It can be controlled automatic program and manual program through the key switch. MCS-51 Microcontroller is utilized for processing the program

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้รับความอนุเคราะห์จากอาจารย์ที่ปรึกษาโครงงาน อาจารย์มนชนก ศรีเสือขาม ที่ได้ให้คำแนะนำและข้อมูลที่เป็นประโยชน์กับโครงงานชิ้นนี้ พร้อมทั้งขอขอบคุณอาจารย์ทุกท่านที่ได้มอบความรู้รวมทั้งแง่คิดในการทำงานต่างๆ และสุดท้ายนี้ขอขอบคุณเพื่อนๆ และรุ่นพี่ภาค อสบ. ทุกคนที่คอยให้กำลังใจด้วยดีเสมอมาตลอดการศึกษาที่อยู่ในรั้วลาดกระบังแห่งนี้

จึงขอขอบคุณในความกรุณามา ณ ที่นี้

ผู้จัดทำโครงงาน

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

เนื่องจากในปัจจุบันนี้วงจรอิเล็กทรอนิกส์ได้เข้ามามีความสำคัญในชีวิตประจำวันของเรามากขึ้นโดยเฉพาะงานทางด้านอุตสาหกรรมก็เป็นงานอีกด้านหนึ่งที่ต้องนำเอาความรู้ทางด้านอิเล็กทรอนิกส์มาใช้ควบคุมในส่วนต่างๆ ของระบบ เช่น การควบคุมการทำงานของระบบลำเลียงชิ้นงานในโรงงานอุตสาหกรรม การทำงานอย่างอัตโนมัติของแขนกล ทั้งหมดนี้เป็นแนวคิดในการนำเอาอุปกรณ์ประเภทคอนโทรลเลอร์เข้ามาใช้เพื่อความสะดวก รวดเร็วและยังลดอุบัติเหตุที่อาจจะเกิดขึ้นระหว่างการทำงานได้ ที่กล่าวมานี้นับเป็นจุดเริ่มของโครงงานการควบคุมดีซีมอเตอร์และสเตปปิ้งมอเตอร์ด้วยไมโครโปรเซสเซอร์ เพื่อเป็นแนวทางในการศึกษาและพัฒนาในการนำมาใช้งานจริงต่อไป

1.2 วัตถุประสงค์ของโครงงาน

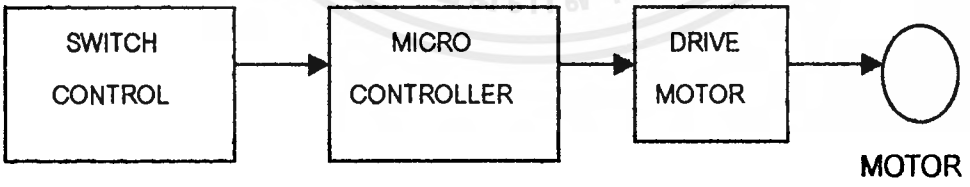
1.2.1 เพื่อศึกษาการทำงานของมอเตอร์ดีซีแบบสเตปปิ้งมอเตอร์

1.2.2 เพื่อศึกษาการทำงานของไมโครคอนโทรลเลอร์ที่ใช้ควบคุมเครื่องจักรในโรงงานอุตสาหกรรมได้

1.2.3 เพื่อพัฒนาการเขียนโปรแกรมควบคุมฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ

1.3 ขอบเขตของโครงงาน

ในส่วนของโครงงานในเทอมนี้เป็นระบบการควบคุมแขนกลที่ใช้ไมโครอนโทรลเลอร์เพื่อใช้ในการควบคุมตำแหน่ง ทิศทาง ความเร็ว ในระบบควบคุมที่เราต้องการ ซึ่งเราสามารถแบ่งการทำงานทั้งหมดได้ดังรูปที่ 1.1



รูปที่ 1.1 โครงสร้างของระบบควบคุมมอเตอร์

จากรูปที่ 1.1 เราจะเห็นได้ว่าโครงสร้างหลักระบบควบคุมการทำงานของแขนกลจะประกอบด้วย 3 ส่วนหลักๆ คือ

ส่วนที่ 1 คือภาคสวิทช์คอนโทรลโดยส่วนนี้จะถูกออกแบบให้เป็นส่วนควบคุมให้การเคลื่อนที่ของมอเตอร์ในทิศทางต่างๆ

ส่วนที่ 2 คือส่วนประมวลผลที่รับคำสั่งมาจากคีย์สวิทช์เพื่อไปสั่งการทำงานของภาคไดร์เวอร์อีกทีหนึ่ง

ส่วนที่ 3 คือภาคไดร์เวอร์ ส่วนนี้จะรับสัญญาณจากภาคประมวลผลเพื่อทำการควบคุมการทำงานของแขนกล

1.4 <u>ประโยชน์ที่คาดว่าจะได้รับ</u>

สามารถเข้าใจและนำไปประยุกต์ใช้งานของระบบควบคุมการทำงานต่างๆ ของดีซีมอเตอร์ด้วยไมโครคอนโทรลเลอร์ตะกูล MCS51 อันจะเป็นประโยชน์ในการเข้าใจระบบควบคุมหุ่นยนต์ที่ซับซ้อนต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 สเตปปิ้งมอเตอร์

2.1.1 การทำงานของสเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์มีความแตกต่างจากมอเตอร์ทั่วๆ ไป โดยเมื่อป้อนกำลังไฟฟ้าให้กับมัน มันจะหมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุด ซึ่งแตกต่างจากมอเตอร์ทั่วๆ ไป ซึ่งจะหมุนทันที และตลอดเวลาเมื่อป้อนแรงดันไฟฟ้า สเต็ปเปอร์มอเตอร์สามารถกำหนดตำแหน่งของการหมุนด้วยตัวเลขได้อย่างละเอียดโดยการใช้คอมพิวเตอร์เป็นตัวกำหนดและจัดเก็บตัวเลขเหล่านั้นไว้

สเตปปิ้งมอเตอร์โดยทั่วไปมีจำนวนของขั้วแม่เหล็กหรือจำนวนสเตปต่อรอบเป็นจำนวนมาก ปกติอยู่ที่ประมาณ 100 – 400 สเตปต่อรอบ การมีจำนวนสเตปมากๆ นี้ไม่ได้เพิ่มจำนวนขั้วแม่เหล็กไฟฟ้าที่สเตเตอร์ แต่ทำได้โดยการเพิ่มจำนวนซี่ของขั้วแม่เหล็กที่ โรเตอร์ จำนวนสเตปต่อรอบทั้งหมดจะได้จากการคูณจำนวนขั้วแม่เหล็กบนสเตเตอร์และจำนวนซี่ที่โรเตอร์ ดังเช่น ถ้ามีขั้วแม่เหล็ก 3 ขั้วบนสเตเตอร์ และ 8 ซี่ขั้วแม่เหล็กบนโรเตอร์ สเตปปิ้งมอเตอร์นี้จะทำงานที่ 24 สเตปต่อรอบหรือหมุนไปเป็นมุม 15 องศาต่อสเตป

การใช้วงจรดิจิตอลคอนโทรลการจ่ายกำลังไฟฟ้าเข้าสู่ขดลวดบนสเตเตอร์แบบซีเควนเชียล ทำให้สามารถควบคุมการเคลื่อนที่ทุกสเตปได้เช่นเดียวกับการควบคุมในวงจรดีซีเซอร์โว (DC servo) แต่การควบคุมด้วยดิจิตอลไม่จำเป็นต้องมีการป้อนกลับ การเคลื่อนที่ทุกสเตปได้จากการคำนวณจำนวนรอบหรือมุมในการหมุนที่ต้องการ แล้วจึงส่งข้อมูลที่ได้ไปควบคุมการหมุนของมอเตอร์, พิกัดในการทำงาน อาทิเช่น ความเร็ว, มุมในการเคลื่อนที่, ตำแหน่งของเพลา ทั้งหมดนี้จะถูกควบคุมด้วยข้อมูลที่ส่งมาควบคุม

2.1.2 ชนิดของสเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์แบ่งตามพื้นฐานได้เป็น 3 ชนิด คือ

1. วาริเอเบิลรีลักแตนซ์ (Variable Reluctance : VR) ชนิดวาริเอเบิลรีลักแตนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (multi-tooth) ทำจากเหล็กอ่อนเราจะทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมากคือใช้นิ้วหมุนเพลาของมอเตอร์ และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์จะไม่เกิดปรากฏการณ์ทางแม่เหล็ก (magnetism)มันจึงหมุนได้ตลอดโดยไม่ติดขัด

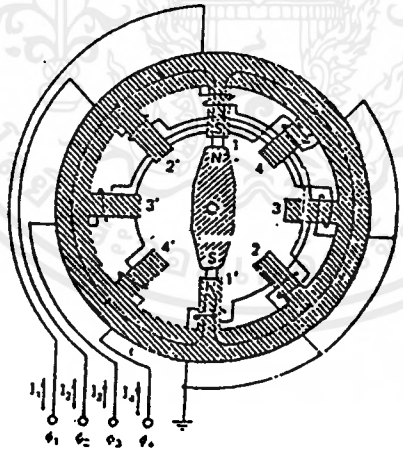
2. เพอร์มาเนนต์แมกเน็ต (Permanent Magnet : PM) ชนิดเพอร์มาเนนต์แมกเน็ตมีโครงสร้างของโรเตอร์แบบเรียบไม่มีซี่ขั้วแม่เหล็กและบนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยป้อนพัลซ์กระตุ้นที่ขดลวดบนสเตเตอร์ เช่น ถ้าเป็นสเตปปิ้งมอเตอร์แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้วซึ่งมีคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กถาวรบนโรเตอร์จะถูกดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์ เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวดโรเตอร์จะอยู่คงที่ ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้

ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดเป็นแรงยึดหน่วงขึ้น สเตปปิ้งมอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น

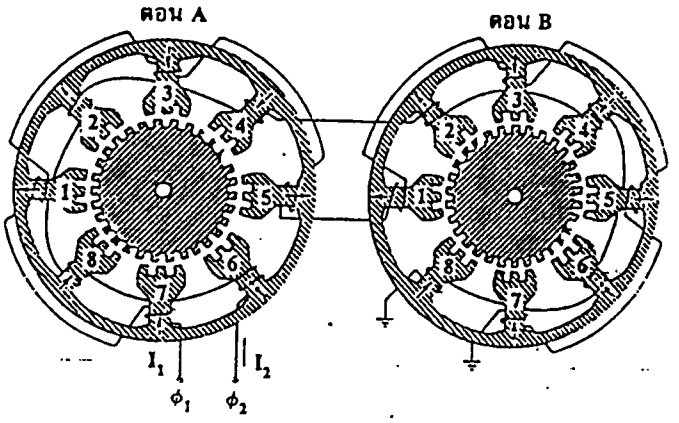
3. แบบไฮบริดจ์ (hybrid) ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานกันมากที่สุด โดยเฉพาะนำมาใช้งานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ ชนิดไฮบริดมีโครงสร้างภายในซึ่งได้จากการรวมเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากชนิดเพอร์มาเนนต์แมกเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ชนิดที่มีแรงยึดหน่วงสูง, มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง

รูปสเตปปิ้งแบบต่างๆ แสดงได้ดังในรูปข้างล่างนี้

(ก)



(ข)



(ค)

รูปที่ 2.1 แสดงโครงสร้างภายในพื้นฐานของสเตปปิ้งชนิดต่างๆ

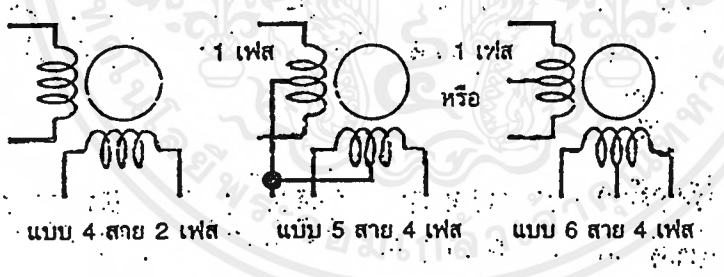
ก. แสดงรูปโครงสร้างของสเตปปิ้งแบบ VR

ข. แสดงรูปโครงสร้างของสเตปปิ้งแบบ PM

ค. แสดงรูปโครงสร้างของสเตปปิ้งแบบ hybrid

2.1.3 การจำแนกชนิดของสเต็ปเปอร์มอเตอร์ด้วยการพันคอยล์

การพันขดลวดหรือคอยล์บนสเตปปิ้งมอเตอร์มีอยู่ 2 วิธี คือ แบบไบโพลาร์ (bipolar) และแบบยูนิโพลาร์ (unipolar) ดังแสดงในรูปที่ 2.2



(ก) (ข)

รูปที่ 2.2 แสดงการพันขดลวดบนสเตเตอร์ของสเตปปิ้งมอเตอร์

(ก) สเตปปิ้งแบบไบโพลาร์

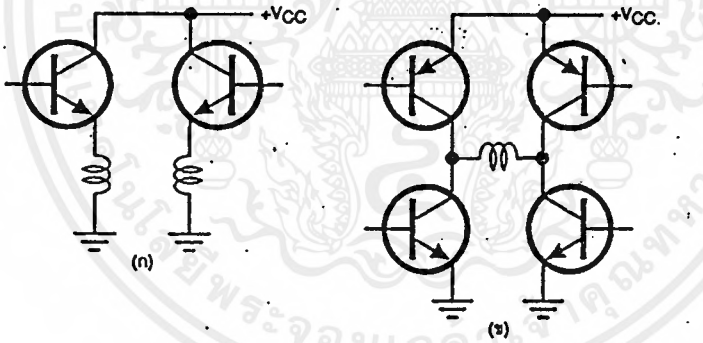
(ข) แบบยูนิโพลาร์ทั้งแบบ 5 สายและ 6 สาย ชนิด 4 เฟส

สเตปปิ้งมอเตอร์แบบไบโพลาร์มีการพันขดลวด 1 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้าและสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศทางการไหลของกระแสไฟฟ้าซึ่งการกำหนดทิศทางการไหล และการกลับทิศทางการไหลของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิทชิงกลับขั้วไฟฟ้า

สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขด บนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปมาทำได้โดยการสวิทชิงกระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดหนึ่งแทนเท่านั้น โดยปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วมเพื่อลดจำนวนของสายไฟที่ต่อจากมอเตอร์ วงจรจ่ายกำลังไฟฟ้าของมอเตอร์แบบ ยูนิโพลาร์ทำได้ง่ายกว่าชนิดไบโพลาร์ เพราะมันต้องการเพียงสวิทช์ธรรมดาในการเปิดและปิดกำลังไฟฟ้าให้กับขดลวดบนสเตเตอร์ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 2.3 แสดงวงจรจ่ายกำลังไฟฟ้าซึ่งใช้ทรานซิสเตอร์ทำหน้าที่เป็นตัวสวิทชิงให้กับสเตปปิ้งมอเตอร์ ที่มีการพันขดลวดทั้ง 2 แบบ จะเห็นได้ว่าในแบบของยูนิโพลาร์ เป็นวงจรที่ง่ายและไม่มีความซับซ้อนเลย

อย่างไรก็ตามการพันขดลวดแบบยูนิโพลาร์ก็มีจุดด้อยตรงที่การพันแบบนี้จะทำให้เกิดแรงบิดน้อยกว่าแบบไบโพลาร์เพราะจะมีเพียงครึ่งหนึ่งของขดลวดที่ถูกกระตุ้นให้ทำงานเท่านั้นในระยะเวลาหนึ่ง

การพิจารณาว่าสเตปปิ้งมอเตอร์ตัวใดมีการพันขดลวดแบบใดสังเกตได้ง่ายโดยถ้าเป็นแบบไบโพลาร์จะมีสายไฟต่อออกจากมอเตอร์เพียง 4 สาย และถ้าเป็นแบบยูนิโพลาร์จะมี 5 สาย หรือ 6 สายหรือทราบได้โดยการอ่านจากป้าย (name plate) ที่ติดอยู่กับมอเตอร์ได้



รูปที่ 2.3 แสดงวงจรจ่ายกำลังไฟฟ้าให้กับสเตปปิ้งทั้ง 2 แบบ

(ก) สำหรับชนิดยูนิโพลาร์ ซึ่งใช้ทรานซิสเตอร์สวิทช์เพียงตัวเดียวต่อ 1 คอยล์

(ข) สำหรับชนิดไบโพลาร์ ซึ่งต้องใช้ทรานซิสเตอร์สวิทช์ 4 ตัวต่อ 1 คอยล์

2.1.4 การกระตุ้นและควบคุมการหมุนของสเตปปิ้งมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องแบ่งได้เป็น 3 รูปแบบ คือ แบบเวฟ (wave) , แบบ 2 เฟส (two phase) และแบบครึ่งสเตป (half step) ทั้ง 3 แบบต่างก็มีข้อดีและข้อเสียแตกต่างกันออกไป

แบบเวฟเป็นการกระตุ้นรูปแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งและเรียงถัดกันไป ดังเช่นขดที่ 1,2,3,4,1 หรือ 1,4,3,2,1 ขึ้นอยู่กับทิศทางที่ต้องการให้หมุน ดังนั้นจึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 1

ตารางที่ 1 แสดงขั้นตอนการกระตุ้นเฟสแบบเวฟ

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

แบบ 2 เฟสเป็นการกระตุ้นอีกรูปแบบหนึ่งซึ่งคล้ายกับแบบเวฟแต่การกระตุ้นแบบนี้จะทำโดยการจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไปเช่นเดียวกับแบบเวฟคือขดลวดที่ถูกกระตุ้น 12,23,34,41,12 หรือ 14,43,32,21,14 ขึ้นอยู่กับทิศทางการหมุน การเพิ่มจำนวนของขดลวดที่ถูกกระตุ้นนี้ทำให้เพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ได้ด้วยแรงดึงแบบเต็มแรง จาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกันและต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือ การกระตุ้นแบบนี้ต้องใช้แหล่งจ่ายกำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 2

ตารางที่ 2 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบ 2 เฟส

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

แบบครึ่งสเต็ปเป็นรูปแบบที่เกิดจากการผสมผสานระหว่างการกระตุ้นแบบเวฟและแบบ 2 เฟสเพื่อเพิ่มจำนวนจำนวนของสเตปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ ขดลวดที่ถูกกระตุ้น 1,12,2,23,3,34,4,41,1 หรือในการหมุนอีกทิศทางหนึ่งจะได้เป็น 1,14,4,43,3,32,2,21,1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเตปมีระยะสั้นลงและแต่ละสเตปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของ

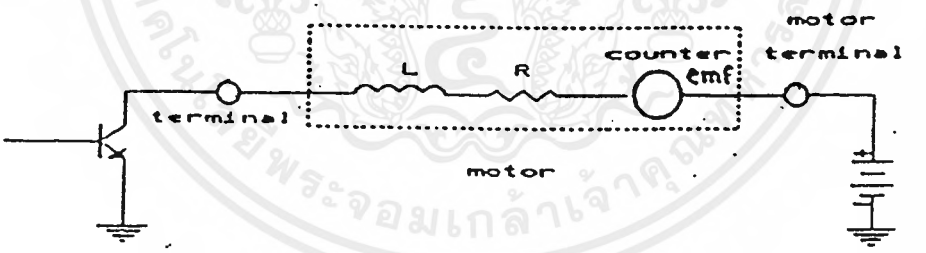
ตำแหน่งมีเพิ่มมากขึ้นแต่ต้องพึงระวังไว้อีกประการหนึ่งว่าเมื่อถูกกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเตปจึงจะได้เท่ากับ 1 สเตปเต็มเหมือนกับในการควบคุมแบบ 2 แบบแรก ขั้นตอนการทำงานต่างๆ ดังแสดงในตารางที่ 3

ตารางที่ 3 แสดงขั้นตอนการกระตุ้นขดลวดแต่ละเฟสแบบครึ่งสเตป

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

2.1.5 ปัญหาเกี่ยวกับวงจร DRIVE

ขดลวดของ สเตปปิ้งมอเตอร์ เป็น INDUCTIVE และมีค่าเปรียบเสมือนผลรวมของอินดัคแตนซ์อนุกรมกับความต้านทานดังรูปที่ 2.4

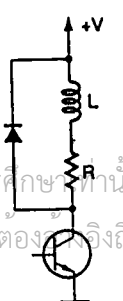


รูปที่ 2.4 วงจรสมบูรณ์ของขดลวดสเตปปิ้งมอเตอร์

2.1.6 **ซัพเพรสเซอร์**

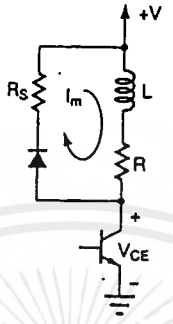
เมื่อทรานซิสเตอร์หยุดนำกระแส จะทำให้เกิดแรงดันค่าสูงจำนวนหนึ่ง เนื่องจากผลของการเปลี่ยนแปลงของกระแสในอินดัคแตนซ์ และแรงดันนี้อาจจะเป็นอันตรายแก่ทรานซิสเตอร์ได้ วิธีป้องกันสามารถทำได้ในหลายๆ วิธีเช่น

1) ไดโอดซัพเพรสเซอร์ ดังรูป



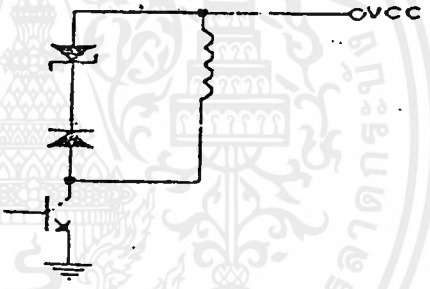
กระแสหมุนเวียน CIRCULATING CURRENT จะเริ่มไหลหลังจากทรานซิสเตอร์ หยุดกระแสและศักดาคอลเล็คเตอร์จะเท่ากับศักดาของแหล่งจ่าย ข้อเสียคือกระแสจะหมุนเวียนอยู่นานและจะทำให้เกิดแรงบิดห้ามล้อ (BREAKING TORQUE) พลังงานส่วนใหญ่สูญเสียในความต้านทานของขดลวด มีปัญหาเรื่องความเย็น

2) ไดโอดและรีจิสเตอร์ชัพเพรสเซอร์ ดังรูป



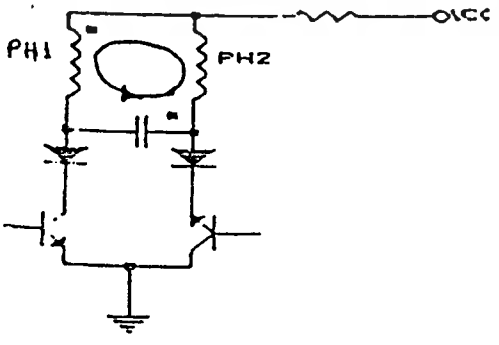
ถ้า Rs ยิ่งมากกระแสหมุนเวียนจะลดลงเร็วขึ้น แต่ศักดาของคอลเลคเตอร์จะมีค่าสูงขึ้น พลังงานส่วนใหญ่จึงสูญเสียใน Rs

3) ซีเนอร์ไดโอดชัพเพรสเซอร์ ดังรูป



เมื่อทรานซิสเตอร์อยู่ในสภาวะคัทออฟกระแสจะลดลงได้เร็วกว่า 2 แบบแรก และศักดาที่คอลเลคเตอร์จะเท่ากับ ศักดาของซีเนอร์บวกกับศักดาของแหล่งจ่าย ซึ่งเป็นอิสระต่อกระแสพลังงานส่วนใหญ่สูญเสียในซีเนอร์

4) คอนเดนเซอร์ชัพเพรสเซอร์ ดังรูป



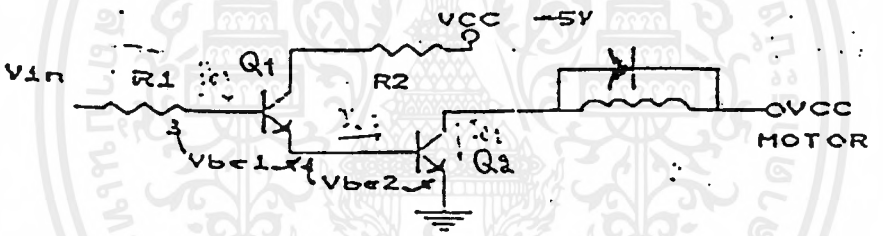
จะใส่ CONDENSER 01 กับ 03 และ 02 กับ 04 เมื่อทรานซิสเตอร์หยุดนำกระแส CONDENSER จะต่อกับทรานซิสเตอร์โดยผ่านทางไดโอดและจะดูดกลืน กระแสที่ค่อยๆ ลดลงจาก

ขดลวดของ มอเตอร์เพื่อป้องกันทรานซิสเตอร์เสียและยังช่วยลดความร้อนที่เกิดขึ้นในขดลวด สเตเตอร์เนื่องจากการ ออสซิลเลท (OSCILLATE) ของโรเตอร์ (ROTOR)

2.1.7 การแก้ไขการเพิ่มขึ้นของกระแส

เมื่อทรานซิสเตอร์นำกระแสเพื่อกระตุ้นเฟส แหล่งจ่ายไฟ (POWER SUPPLY) จะต้องชนะผลของอินดัคแตนซ์ของขดลวดก่อนที่กระแสจะเพิ่มขึ้นได้ถึงค่าที่กำหนด คือ อินดัคแตนซ์มีคุณสมบัติที่จะต้านการเพิ่มขึ้นของกระแสต่อไซเคิลซึ่งจะมีค่ามากขึ้น และเป็นผลให้แรงบิด (TORQUE) ลดลง และผลตอบสนองลดลง วิธีการลดเวลาการเพิ่มขึ้นของกระแส และปรับปรุงคุณลักษณะของแรงบิดที่มีความเร็วสูงให้ดีขึ้น โดยต่อความต้านทาน อนุกรมกับขดลวดของ มอเตอร์ซึ่งต่อกับแหล่งจ่ายไฟตรง จะต้องเลือกค่าความต้านที่เหมาะสม เพื่อให้กระแสที่ต้องการไหลผ่านขดลวดที่มีสภาวะคงที่ TIME CONSTANCE ของวงจรจะลดลงเนื่องจากผลรวมของความต้านทานในขดลวดเพิ่มขึ้น

การไบแอส (BIAS) วงจรภาคขยายกระแสไฟให้ได้กระแสแต่ละเฟสตามต้องการ สมมุติให้ B2 คือค่าขยายกระแสต่ำสุดของ Q2 และ IC2 คือกระแสขับสูงสุดต่อเฟสจะได้



$$R2 = \frac{(5 - Vbe2)}{Ib2} = \frac{(5 - Vbe2)*B}{Ic2}$$

สำหรับ R1 สมมุติให้ B คือ ค่าขยายกระแสต่ำสุดของ Q1 และ VIN มีค่าประมาณ 4 โวลท์ สำหรับลอจิก (LOGIC) "1" จะได้

$$R1 = \frac{(4 - Vbe1 - Vbe2)}{Ib1} = \frac{(4 - Vbe1 - Vbe2)*B1}{Ic1}$$

$$R1 = \frac{2.8\,B1}{Ib2}$$

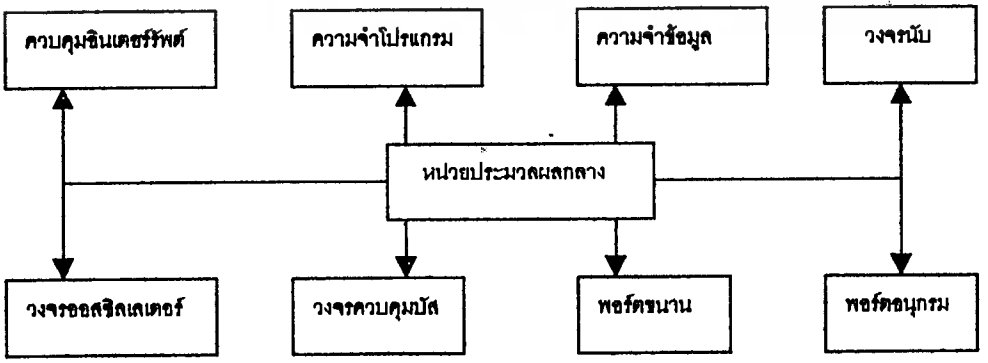
2.2 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรลเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้เอง ทำให้การใช้งานง่ายขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์ภายนอกมากเหมือนกับไมโครโปรเซสเซอร์ทั่วไป นอกจากนี้หากเราต้องการใช้งานไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์อื่นเพิ่มเติม เช่น ไอซี 8255 หรือหน่วยความจำภายนอกเราก็สามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโทรลเลอร์ได้อีกด้วย

2.2.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 8051 แสดงในรูปที่ 2.5 ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- หน่วยความจำภายในสำหรับเก็บข้อมูลขนาด 128 ไบต์ (Internal data memory 128 byte)
- หน่วยความจำภายในสำหรับเก็บโปรแกรมขนาด 4 กิโลไบต์ (Internal program memory 4 Kbytes)
- อุปกรณ์ควบคุมการอินเตอร์รัพต์ (Interrupt Control Unit)
- ตัวตั้งเวลาและตัวนับขนาด 16 บิต 2 ชุด(Timer/Counter0 and Time/Counter1)
- พอร์ตควบคุมการสื่อสารอนุกรมแบบ Full Duplex ซึ่งสามารถรับและส่งข้อมูลพร้อมกันได้
- พอร์ตขนานสำหรับติดต่อกับอุปกรณ์ภายนอกจำนวน 4 พอร์ตๆละ 8 บิต
- วงจรผลิตสัญญาณนาฬิกาภายใน



รูป 2.5 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

การทำงานของส่วนประกอบต่างๆ ที่อยู่ในไมโครคอนโทรลเลอร์ MCS-51 จะเป็นดังต่อไปนี้

2.2.2 หน่วยความจำสำหรับเก็บโปรแกรม (Internal Program Memory)

หน่วยความจำสำหรับเก็บโปรแกรมทำหน้าที่เก็บโปรแกรมที่ผู้ใช้เขียนขึ้นเพื่อควบคุมการทำงานของไมโครคอมโทรลเลอร์โดยหน่วยความจำจะเป็นแบบ ROM มีความจุ 4 Kbytes (ตำแหน่ง 0000H-0FFFH) ในการใช้งานเราสามารถกำหนดให้ไมโครคอนโทรลเลอร์ หรือโปรแกรมที่เก็บอยู่ในหน่วยความจำ (EPROM) ที่อยู่ภายนอกก็ได้ การเลือกการติดต่อทำได้โดยการป้อนสัญญาณควบคุมให้ที่ขา $\overline{EA}$ (External Access) ถ้าต้องการให้ไมโครอนโทรลเลอร์ติดต่อกับโปรแกรมที่อยู่ในหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์จะต่อขานี้กับลอจิก 1 หากต้องการให้ไมโครคอนโทรลเลอร์ติดต่อกับโปรแกรมที่อยู่ในหน่วยความจำภายนอกจะต่อขานี้กับลอจิก 0 การติดต่อกับหน่วยความจำโปรแกรมภายนอกจะติดต่อได้ทั้งหมด 64 Kbytes (ตำแหน่ง 0000H-FFFFH) ในกรณีที่กำหนดให้ไมโครคอนโทรลเลอร์ติดต่อกับโปรแกรมที่อยู่ในหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์จะติดต่อได้ 4 Kbytes (สำหรับเบอร์ 8051) หากตำแหน่งของโปรแกรมมีค่าเกินกว่าตำแหน่งของหน่วยความจำภายใน (โปรแกรมยาวเกินกว่า 4 Kbytes) ตัวไมโครคอนโทรลเลอร์จะทำการติดต่อกับโปรแกรมที่มีอยู่ในหน่วยความจำภายนอกอัตโนมัติ

2.2.3 หน่วยความจำข้อมูลภายใน (Internal Data Memory)

หน่วยความจำข้อมูลภายในทำหน้าที่เก็บข้อมูลทั่วไป และทำหน้าที่เป็นสแตก (Stack) บางส่วนหน่วยความจำข้อมูลภายในของเบอร์ 8051 มีอยู่ 128 ไบต์ โดยอยู่ในตำแหน่ง 00H-7FH

2.2.4 อุปกรณ์ควบคุมอินเตอร์รัพต์ (Interrupt Control Unit)

เป็นส่วนที่ทำหน้าที่ควบคุมการอินเตอร์รัพต์ของไมโครคอนโทรลเลอร์ ซึ่งมีการร้องขออินเตอร์รัพต์ได้จาก 6 แหล่งกำเนิด คือ สัญญาณจากภายนอก 2 สัญญาณ จากตัว Timer0, Timer1 และ Timer2 (เบอร์ 8051 มี Timer เพียง 2 ตัว ดังนั้นจะมีแหล่งกำเนิดสัญญาณอินเตอร์รัพต์ 5 แหล่ง) และจากอุปกรณ์รับส่งข้อมูลแบบอนุกรม 1 สัญญาณ สัญญาณอินเตอร์รัพต์ที่เกิดขึ้นเราสามารถควบคุมให้ไมโครคอนโทรลเลอร์ตอบรับหรือไม่ตอบรับก็ได้ นอกจากนี้เรายังสามารถจัดลำดับความสำคัญของอินเตอร์รัพต์ (Interrupt priority) จากสัญญาณต่างๆ ได้เป็น 2 ระดับแตกต่างกัน

2.2.5 ตัวตั้งเวลาและตัวนับ (Timer/Counter)

ในไมโครคอนโทรลเลอร์เบอร์ 8051 ประกอบด้วยรีจิสเตอร์ตัวตั้งเวลา/ตัวนับ ขนาด 16 บิตจำนวน 2 ชุด คือ Timer 0 และ Timer1 สำหรับในไมโครคอนโทรลเลอร์เบอร์ 8052 จะมี Timer2 เพิ่มขึ้นอีก 1 ตัว โดย Timer ทั้งหมดสามารถกำหนดให้ทำงานในลักษณะของตัวนับหรือตัวจับเวลาก็ได้

การทำงานในโหมดของตัวตั้งเวลา ค่าในรีจิสเตอร์จะเพิ่มขึ้นทุกๆ แมชชีนไซเกิลโดย 1 แมชชีนไซเกิลประกอบด้วยสัญญาณนาฬิกา 12 ลูก ดังนั้นอัตราการจับเวลาจะเป็น 1/12 เท่าของความถี่สัญญาณนาฬิกาของระบบ ค่าสูงสุดที่ตั้งได้คือ 2^{16}

การทำงานในโหมดการนับ ค่าของการนับจะเพิ่มขึ้นเมื่อมีสัญญาณเข้ามาที่ขา T0 หรือ T1 เปลี่ยนจาก 1 เป็น 0 ความเร็วในการนับสูงสุดคือ 1/24 เท่าของสัญญาณนาฬิกา โดยสัญญาณที่เข้ามาที่ขา T0 หรือ T1 จะมี Duty Cycle เท่าใดก็ได้

2.2.6 พอร์ตอินพุทเอาต์พุท

ในไมโครคอนโทรลเลอร์ MCS-51 ประกอบด้วยพอร์ตอินพุทเอาต์พุทแบบ 2 ทิศทาง (Bidirectional) จำนวน 4 พอร์ต แต่ละพอร์ตมีอุปกรณ์แลทช์ข้อมูลและเอาต์พุทไดรเวอร์ประกอบอยู่ทางด้านเอาต์พุทและทางด้านอินพุทจะมีบัฟเฟอร์ (พอร์ตทั้ง 4 เป็นรีจิสเตอร์พิเศษชื่อ P1, P2 และ P3) เราสามารถใช้งานแต่ละพอร์ตเป็นอินพุทหรือเอาต์พุทได้ตามต้องการ แต่ละบิตของพอร์ตสามารถเชื่อมต่อกับสัญญาณ TTL ได้โดยตรง

ในการติดต่อกับหน่วยความจำภายนอก พอร์ต P0 และ P2 จะใช้สำหรับกำหนดตำแหน่งของหน่วยความจำภายนอก โดยพอร์ต P0 จะทำงานในลักษณะของมัลติเพล็กซ์ คือเป็นทั้งพอร์ตตำแหน่งและพอร์ตข้อมูล โดย P0 จะเป็นตำแหน่งของหน่วยความจำด้านต่ำ (Low byte) และ P2 จะเป็นตำแหน่งของหน่วยความจำด้านสูง (High byte).

2.2.7 การจัดหน่วยความจำของ MCS-51

การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 จะจัดหน่วยความจำออกเป็น 3 กลุ่มคือ

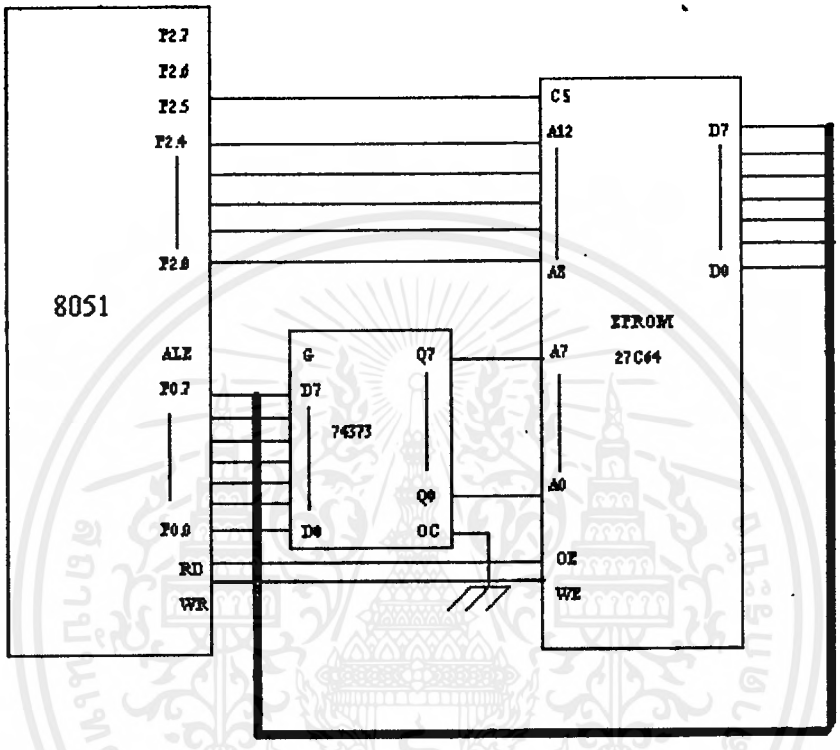
1. หน่วยความจำสำหรับเก็บโปรแกรม (Program Memory) หน่วยความจำสำหรับเก็บโปรแกรมจะเป็นที่เก็บชุดคำสั่งต่างๆ และข้อมูลที่โปรแกรมใช้งาน

2. หน่วยความจำสำหรับเก็บข้อมูล (Data Memory) หน่วยความจำข้อมูลทำหน้าที่เก็บข้อมูลต่างๆ ในขณะที่โปรแกรมทำงานและทำหน้าที่เป็นสแตกบางส่วน

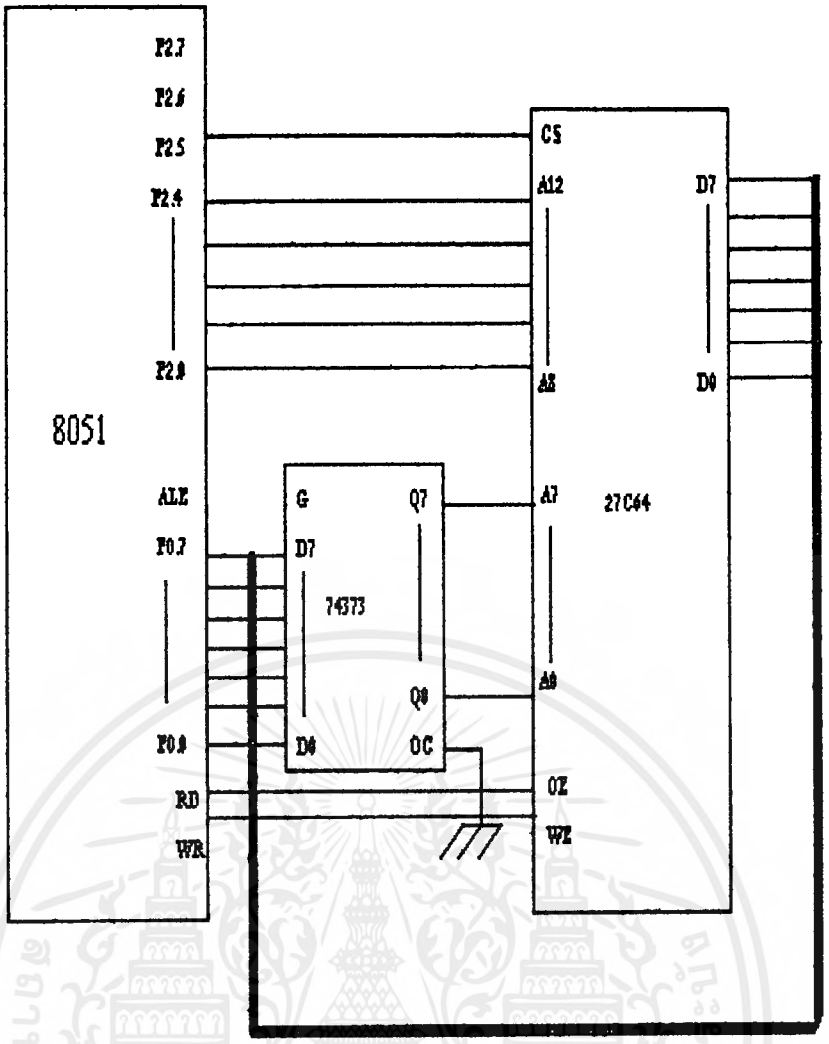
3. รีจิสเตอร์ที่ทำหน้าที่เฉพาะ (Special Function Register)

2.2.8 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (Extemal Program Memory)

วงจรการเชื่อมต่อวงจรไมโครคอนโทรลเลอร์เบอร์ 8051 กับหน่วยความจำโปรแกรมภายนอกจะเป็นดังรูปที่ 2.6 และรูปที่ 2.7 ซึ่งแสดงการเชื่อมต่อกับหน่วยความจำทั้งแบบชนิดอีพรอมและแบบชนิดแรมตามลำดับ



รูปที่ 2.6 การเชื่อมต่อไมโครคอนโทรลเลอร์กับ EPROM 6264

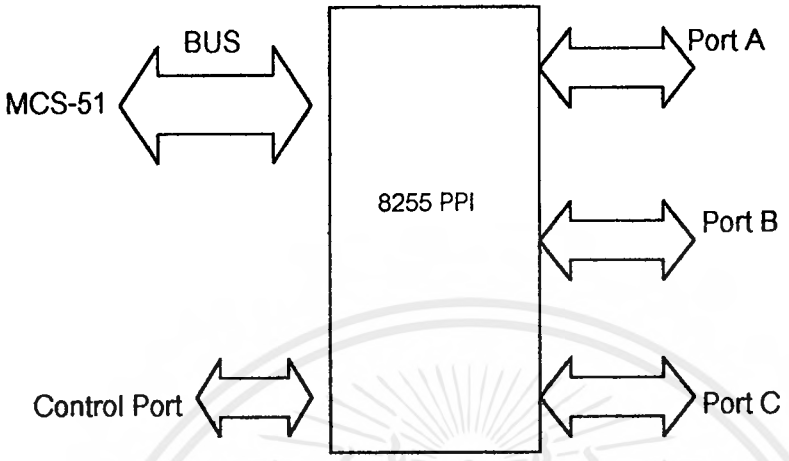


รูปที่ 2.7 การเชื่อมต่อไมโครคอนโทรลเลอร์กับ RAM 6264

2.3 I/O PORT 8255 PPI

8255 เป็นไอซีขนาด 40 ขา มีพอร์ตให้ใช้งานถึง 3 พอร์ต (เป็นขนาด 8 บิท) พอร์ต A, พอร์ต B, พอร์ต C. โดยพอร์ต C นี้สามารถแยกได้เป็น 2 ส่วน คือ พอร์ต C บนตั้งแต่ PC4-PC7 จำนวน 4 บิตและพอร์ต C ล่างตั้งแต่ PC0-PC3 โดยพอร์ตทุกพอร์ต (A,B,C) สามารถโปรแกรมได้ให้เป็นอินพุทหรือเอาท์พุท การนำเอาไมโครโปรเซสเซอร์ไปใช้งานนั้น จำเป็นต้องให้ไมโครโปรเซสเซอร์ติดต่อกับโลกภายนอกซึ่งก็คือให้มันสามารถส่งสัญญาณมาควบคุมอุปกรณ์ต่างๆ ได้ การทำงานของ 8255 จะใช้สัญญาณควบคุมจากตัวไมโครโปรเซสเซอร์ มาควบคุมโดยจะมีการส่งคำสั่ง (Control word) มาที่กลุ่มควบคุมชุด A,B แล้วกลุ่มควบคุมชุดนี้ก็จะสั่งต่อไปที่พอร์ตเพื่อให้เป็นไปตามข้อกำหนดของคำสั่งนั้นๆ เช่น ให้พอร์ต A เป็นอินพุทพอร์ต B เป็นเอาท์พุทพอร์ตเหล่านี้เป็นต้น ส่วนกรณีเมื่อมีการอ่านหรือเขียนพอร์ตจาก CPU นั้น กลุ่มควบคุมลอจิกการอ่าน-เขียนจะเป็นตัวที่ส่งสัญญาณไปบอกแก่กลุ่มควบคุมชุดในแต่ละชุดอีกที ทั้งนี้แล้วแต่ว่า CPU จะมีการอ่าน-เขียนพอร์ตของกลุ่มควบคุมชุดใด

ส่วนที่ทำให้ไมโครโปรเซสเซอร์ติดต่อกับโลกภายนอกได้ที่เรารู้จักกันคือ พอร์ต (PORT) ซึ่งประกอบด้วย พอร์ต A พอร์ต Bพอร์ต C รวมถึงพอร์ตคอนโทรลดังรูปที่ 2.8 การใช้งานก็ให้กำหนดแอดเดรสเพื่ออ้างถึงโดยให้ซีพียูมองเหมือนหน่วยความจำหนึ่งๆ เท่านั้น

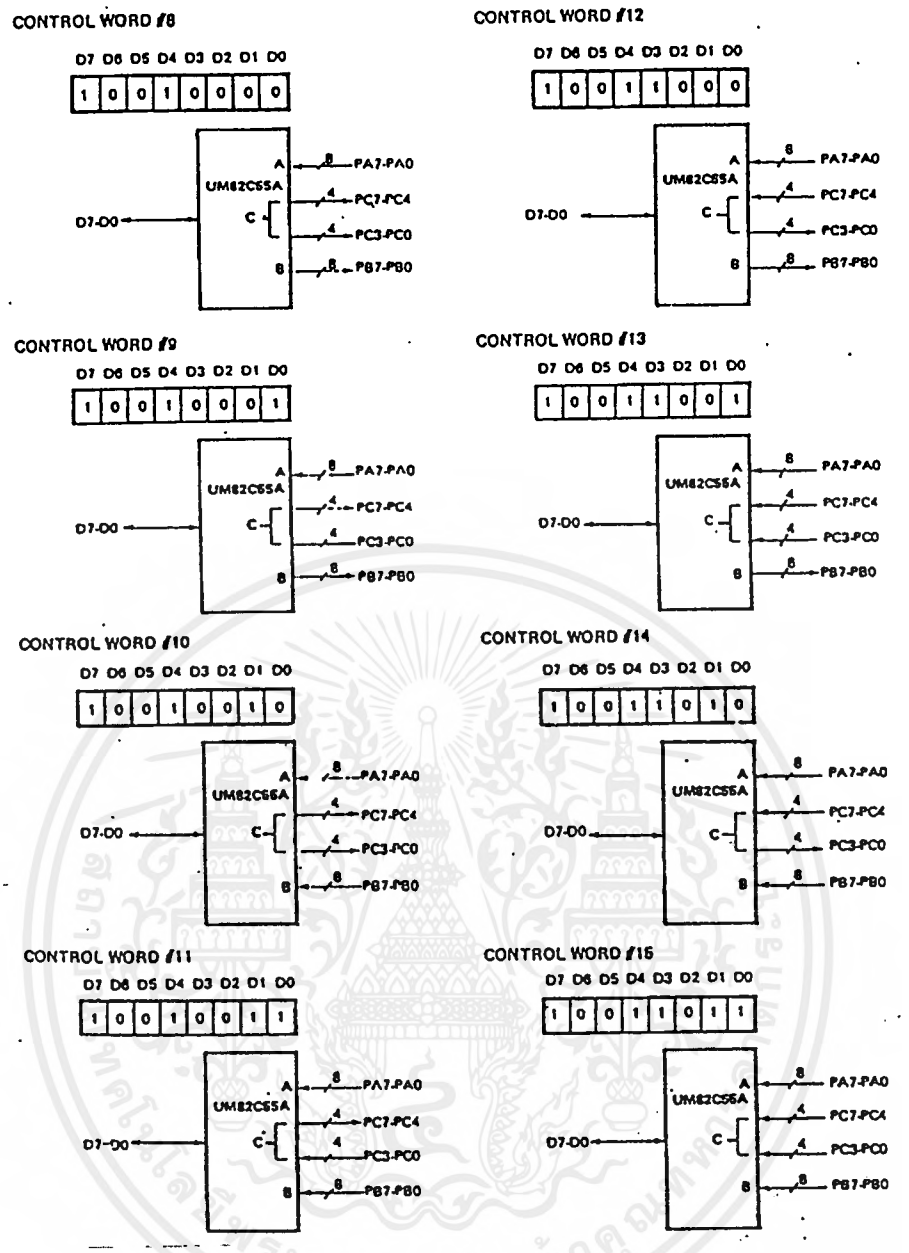


รูปที่ 2.8 แผนผังโครงสร้างของไอซี 8255

2.3.1 การโปรแกรม 8255

เราได้ทราบแล้วในรูปที่ 2.8 ว่าโครงสร้างภายในของ 8255 มีกลุ่มควบคุมชุดอยู่ 3 กลุ่ม ซึ่งทั้ง 3 กลุ่มนี้จะทำงานร่วมกันดังที่กล่าวมาและเราสามารถจะควบคุมการทำงานของพอร์ตจาก CPU โดยสั่งงานมาที่พอร์ตควบคุมดังกล่าว แต่ตัว CPU จะมองเห็น 8255 เป็น 4 พอร์ตด้วยกันโดยแต่ละพอร์ตเสมือนเป็นรีจิสเตอร์ที่ CPU สามารถจะทำการอ่าน / เขียนได้ซึ่งหากมีการอ่านเขียนไปยังพอร์ตดังกล่าวก็จะใช้ร่วมกับสัญญาณ RD โดยสัญญาณ WR หมายถึง เอาท์พุทข้อมูลและ RD แอคทีฟ หมายถึง อินพุทข้อมูล

การใช้งานเราจะต้องส่งรหัสควบคุม (Control Code) เข้าไปยังพอร์ตควบคุม (หรือเรียกอีกอย่างว่ารีจิสเตอร์ควบคุม) ซึ่งจะเป็นข้อมูลขนาด 1 ไบท์ส่งไปที่ แอดเดรส 13H (กรณีนี้เราถอดรหัสไว้ที่ 13H) โดยความหมายของแต่ละบิตที่เราส่งไปควบคุมการทำงานเราสามารถแสดงความหมายของข้อมูลที่ส่งไปได้รูปที่ 2.9

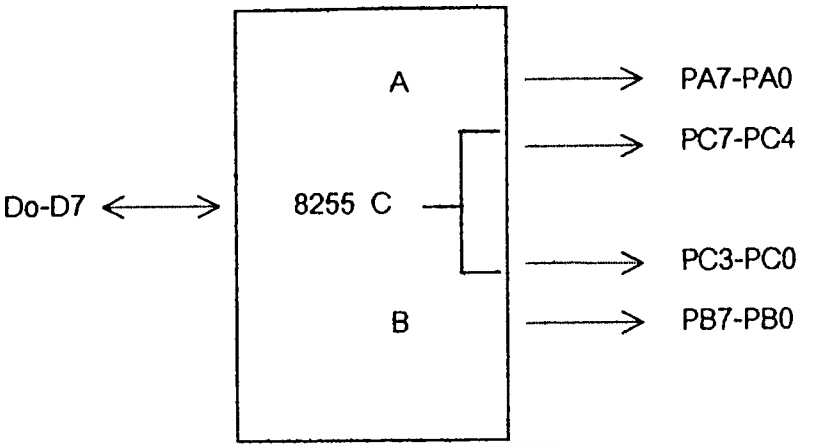


รูปที่ 2.9 คำสั่งควบคุม (control code) ของ 8255

ซึ่งหากเราดูที่รหัสคำสั่งแล้วจะเห็นว่ามีอยู่ 4 บิต ที่ถูกกำหนดตายตัวคือบิต 7, บิต 6, บิต 5 และบิต 2, ส่วนที่เหลืออีก 4 บิต ก็คือข้อกำหนดว่าจะให้พอร์ตใดเป็นพอร์ตอินพุท / เอาท์พุท นั้นเอง ซึ่งหากเราให้พอร์ตใดเป็นอินพุทเราก็ใส่ลอจิก "1" ที่บิตนั้นหรือหาต้องการให้พอร์ตใดเป็นเอาท์พุท ก็ใส่ "0" ที่บิตนั้น จะเห็นได้ว่าจะมีความเป็นไปได้ในการกำหนดลักษณะของพอร์ตในโหมดนี้อยู่ 16 อย่างด้วยกัน ยกตัวอย่างเช่น

ตัวอย่าง 1. ต้องการให้ทุกพอร์ตเป็นเอาท์พุทหมด เราจะได้รหัสคำสั่งเป็น

1	0	0	0	0	0	0	0



รูปที่ 2.10 แสดงตัวอย่างพอร์ตที่ถูกโปรแกรมในโหมด 0

สังเกตว่าขณะสัญญาณ CS แอคตีฟสัญญาณแอดเดรส A_7, A_6, A_5, A_4, A_3, A_2 จะต้องมีข้อมูล 00100 และเมื่อรวมกับ A_1, A_0, จะเป็น 000100XX พอร์ตที่เกิดขึ้นเมื่อ A_0, A_1 เป็น 00 คือ พอร์ต 10H และถ้า A_0, A_1, เป็น 11 พอร์ตจะเป็น 13H การกำหนดพอร์ตของ Z-80 จะได้ข้อมูลบนแอดเดรส 8 เส้น คือ A_0- A_7 เท่านั้น สัญญาณที่จะควบคุม 8255 อีกชุดหนึ่งคือ สัญญาณควบคุมการเขียนและการอ่าน หากสัญญาณ WR แอคตีฟเป็น "0" จะหมายถึง การเขียนพอร์ตหรือส่งข้อมูลให้พอร์ตเอาต์พุท แต่ถ้าสัญญาณ RD แอคตีฟเป็น "0" จะหมายถึง การอ่านพอร์ตหรือรับข้อมูลอินพุท

การโปรแกรม 8255ทำงานจะทำงานได้ 3 โหมด คือ โหมด 0 โหมด 1 โหมด 2

2.3.3 โหมด 0 หรืออินพุทเอาท์พุทแบบพื้นฐาน

การกำหนดโหมดการทำงาน จะต้องส่งข้อมูลคำสั่งเข้าโปรแกรมในโหมดควบคุมของ 8255 ซึ่งในที่นี้โหมดหมายเลข 13H แต่ละบิตของข้อมูลที่ส่งไปจะมีความหมายในตัวเอง ลักษณะความหมายแต่ละบิตลักษณะควบคุมแสดงดังรูป การโปรแกรม 8255 คือ การให้ค่ารหัสบิตต่างๆ เข้าไปในรหัสควบคุมแล้วส่งไปยังรีจิสเตอร์ของพอร์ตควบคุม ความหมายบิตต่างๆ มีดังรูปที่ 2.11

บิต D_7 เป็นบิตที่แสดงรหัสคำสั่งควบคุม ถ้าบิตนี้เป็น "1" หมายถึงรหัสควบคุมนี้จะมีผลต่อการเปลี่ยนแปลงการเซตโหมดต่างๆ ของ 8255

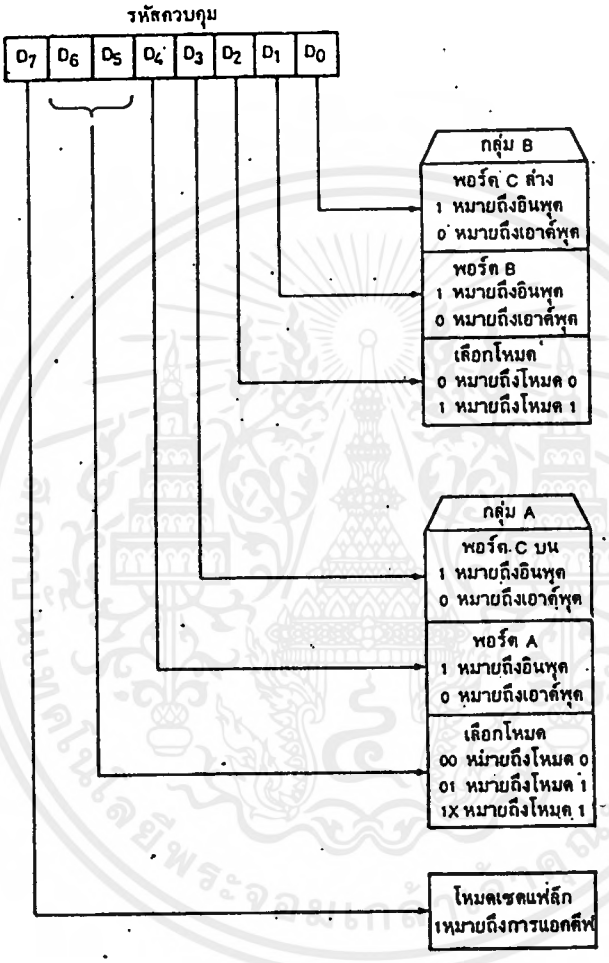
บิต D_6 และ D_5 เป็นการเลือกโหมดของพอร์ต A ซึ่งมี 3 โหมดคือ โหมด 0 โหมด 1 และโหมด 2

บิต D_4 ถ้ามีค่าเป็น "0" หมายถึงเอาต์พุท ถ้ามีค่าเป็น "1" จะหมายถึงการกำหนดให้พอร์ต A เป็นอินพุต

บิต D_3 เป็นบิตที่บอกถึงการเซตของพอร์ต C บน ถ้าเป็น "0" จะทำให้พอร์ต C บนเป็นเอาท์พุต

บิต D_1 เป็นการกำหนดอินพุทเอาท์พุทพอร์ต B ถ้าเป็น "0" หมายถึง เอาต์พุท ถ้าเป็น "1" หมายถึง อินพุท

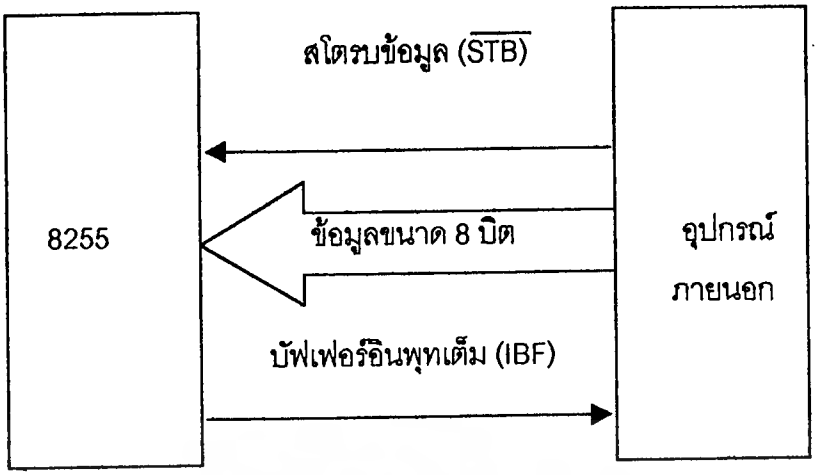
บิต D_0 เป็นการกำหนดอินพุทเอาท์พุทของพอร์ต C ล่าง ถ้าเป็น "0" หมายถึงเอาต์พุท ถ้าเป็น "1" หมายถึง อินพุท



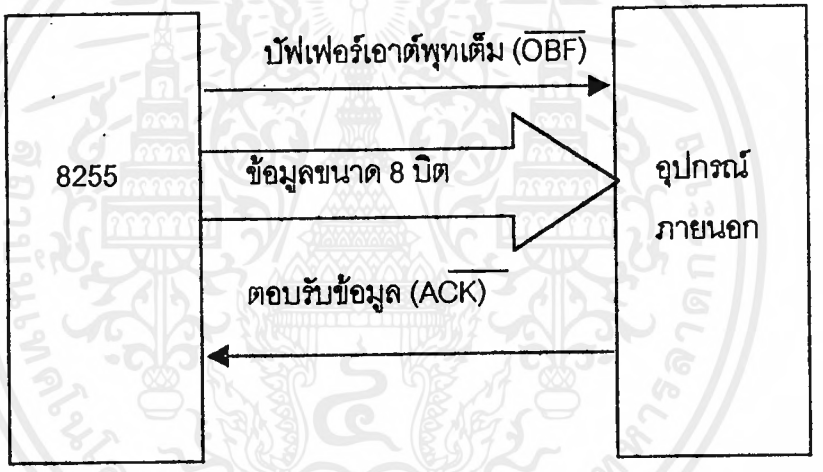
รูปที่ 2.11 ความหมายของบิทต่างๆ ในรหัสควบคุม

2.3.4 การทำงานของ 8255 ในโหมด 1

การทำงาน 8255 ในโหมด 1 เป็นโหมดที่ทำให้อินพุทเอาต์พุทมีการตรวจสอบสัญญาณ (handshaking) โดยใช้อินพุทเอาต์พุทของพอร์ต A และพอร์ต B เป็นหลัก และใช้พอร์ต C บนเป็นตัวตรวจสอบสัญญาณ (handshahake) ของพอร์ต A ส่วนพอร์ต C ล่างเป็นตัวตรวจสอบสัญญาณของพอร์ต B การจัดสัญญาณต่างๆ เหล่านี้แสดงดังรูปแสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ตอินพุทและพอร์ตเอาต์พุท



(ก) พอร์ตอินพุท



(ข) พอร์ตเอาต์พุท

รูปที่ 2.12 โครงสร้างตัวตรวจสอบสัญญาณของพอร์ตอินพุทและพอร์ตเอาต์พุท

แนวความคิดของการใช้พอร์ตอินพุทเอาต์พุทโดยมีตัวตรวจสอบสัญญาณก็เพื่อให้มีการซิงโครไนซ์ระหว่างอุปกรณ์ภายนอกที่ทำงานได้ช้ากับการทำงานของคอมพิวเตอร์ที่ทำงานได้เร็ว เช่น เครื่องพิมพ์ทำงานได้ช้า เมื่อคอมพิวเตอร์ส่งตัวอักษรตัวแรกมาพิมพ์ เครื่องพิมพ์รับตัวอักษรและกำลังจะพิมพ์ คอมพิวเตอร์ก็ส่งตัวที่ 2 ตัวที่ 3 ตามมา ทำให้การประมวลผลของอุปกรณ์เครื่องพิมพ์ทำงานไม่ทัน ซึ่งอาจทำให้ข้อมูลสูญหาย ดังนั้นเครื่องพิมพ์จึงส่งสัญญาณบอกคอมพิวเตอร์ว่า "อย่าเพิ่งส่งมาเพราะยังไม่พร้อมที่จะรับ"

2.3.5 การทำงานของ 8255 ในโหมด 2

8255 ยังมีโหมดการทำงานอีกโหมดหนึ่งคือโหมด 2 ซึ่งทำได้เฉพาะพอร์ต A ในโหมดนี้ 8255 จะใช้พอร์ต A ทำหน้าที่เป็นพอร์ตแบบ 2 ทิศทางคือสามารถเป็นได้ทั้งพอร์ตอินพุทและเอาต์

พุทโดยโครงสร้างของพอร์ต A ทั้งอินพุทเอาต์พุทมีการตรวจสอบสัญญาณทั้งคู่ ส่วนพอร์ต C จะทำหน้าที่ เป็นสัญญาณตรวจสอบ

สังเกตว่าเมื่อโปรแกรมพอร์ต A เป็นโหมด 2 แล้ว พอร์ต B จะต้องโปรแกรมเป็นโหมด 0 หรือ โหมด 1 ก็ได้ ซึ่งก็ทำงานแบบแยกอิสระอีก ในการใช้งานพอร์ตแบบ 2 ทิศทางนี้ใช้ได้กับงานบาง ประเภท เช่น ใช้ในการรับส่งข้อมูลของพอร์ตมาตราฐานของประเภทเช่น IEEE 488 หรือใช้เชื่อมโยง ระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ในการรับส่งข้อมูลสลับกันไปและกลับ

บทสรุป

การใช้ 8255 เป็นการใช้ที่เราพบเห็นในไมโครโปรเซสเซอร์ทั่วไป เพราะใช้งานได้ง่ายและมีความคล่องตัว ผู้ออกแบบที่ต้องใช้พอร์ตขนานมักเลือกใช้ 8255 เป็นตัวอินเตอร์เฟสกับอุปกรณ์ภายนอก

บทที่ 3
โครงสร้างของระบบโครงงาน

จากทฤษฎีที่กล่าวมาแล้วข้างต้นในบทที่ผ่านมา ซึ่งจะถูกนำมาอ้างอิงในส่วนของโครงงานที่ใช้จริงในบทนี้ด้วย โดยเราสามารถแบ่งออกเป็น 2 ส่วนหลักด้วยกันคือ

1 ส่วนของฮาร์ดแวร์

2 ส่วนของซอร์ฟแวร์

3.1 ฮาร์ดแวร์

ในที่นี้เราสามารถแบ่งออกเป็นส่วนใหญ่ ๆ ได้ 2 ส่วนด้วยกันคือ ส่วนของบอร์ดคอนโทรลกับภาคของวงจรขับมอเตอร์ และชุดแมคคานิคที่ประกอบกันเป็นหุ่นยนต์โดยจะได้แยกกล่าวอย่างละเอียดตามลำดับต่อไปนี้

3.1.1 บอร์ดคอนโทรล โดยในส่วนนี้เราจะใช้ไมโครโปรเซสเซอร์ในตระกูล MCS-51 ซึ่งสามารถเรียกอีกอย่างว่าไมโครคอนโทรลเลอร์ก็ได้เนื่องจากในตัวมันมีพอร์ตซึ่งสามารถใช้งานได้ทันทีโดยไม่ต้องอาศัยอุปกรณ์ประเภทอินเตอร์เฟสภายนอกอีก ซึ่งเหมาะกับการศึกษาเรียนรู้และใช้งานทางด้านการคอนโทรลระบบควบคุมมอเตอร์ได้เป็นอย่างดี โดยบอร์ดที่ใช้นี้ถูกออกแบบและผลิตโดยบริษัท อีทีทีจำกัด เราสามารถทำการดีโค้ดแอดเดรสของหน่วยความจำต่างๆ รวมถึงการเพิ่มพอร์ตขยายระบบเพื่อพร้อมกับการทำงานในโครงงานออกมาได้ดังแสดงในรูปที่ 3.1

คุณลักษณะของบอร์ดที่ใช้

- ใช้ ซีพียูเบอร์ 8031 เป็นตัวประมวลผล

- หน่วยความจำของบอร์ด จากทฤษฎีของซีพียู 8031 นี้เราสามารถแบ่งหน่วยความจำออกได้เป็น DATA MEMORY กับ CODE MEMORY แต่ในบอร์ดของเราได้กำหนดหน่วยความจำออกเป็นดังนี้

ก. ส่วนของ CODE MEMORY เรากำหนดเป็นส่วนของโปรแกรมมอนิเตอร์ (ซึ่งใช้ทำงานเป็นส่วนที่จัดการในการรับส่งข้อมูลจากพีซีเข้าสู่ระบบผ่านทางพอร์ตอนุกรม) ซึ่งถูกบรรจุอยู่ใน ไอซีอีพรอมเบอร์ 27C256 โดยอีพรอมเบอร์นี้สามารถอ้างถึงตำแหน่งที่อยู่ได้ทั้งหมดคือตั้งแต่ แอดเดรส 0000h-7FFFh รวมแล้ว 32 kbyte หรือ 32*8= 256 kbit แต่เนื่องจากโปรแกรมที่ใช้ใส่ในอีพรอมนี้ใช้เนื้อที่ไม่มากเราจึงกำหนดตำแหน่งในการอ้างถึงเพียง 8 kbyte เท่านั้นคือที่ แอดเดรส 0000h-1FFFh

ข. ส่วนของ DATA MEMORY ในส่วนนี้เราจะใช้ หน่วยความจำชนิดแรมในการเก็บข้อมูลส่วนนี้ไว้ โดยให้ใช้ ไอซี แรมเบอร์ 6464 โดยเราได้ดีโค้ดแอดเดรสทางฮาร์ดแวร์คือ 2000h- 3FFFh รวมทั้งหมด 8 kbyte และเนื่องจากเมื่อเราทำการรีเซ็ทซีพียูแล้ว ตัวซีพียูจะทำการอ่านค่าที่ตำแหน่ง 0000h ก่อน เราจึงอาศัยความรู้นี้ในการพัฒนาระบบให้ผู้ใช้กับพีซีสามารถทำการติดต่อกันได้โดย

ผ่านทางพอร์ตอนุกรมในการรับส่งข้อมูลหรือโปรแกรมของเราในหน่วยความจำชั่วคราวนี้โดยใช้โปรแกรมมอนิเตอร์ที่อยู่ในอีพรอมเป็นตัวควบคุมการโหลดข้อมูลจากพีซีมายังบอร์ดของเรา แต่ในการทำงานจริงเราจะให้ระบบเริ่มทำงานได้ตั้งแต่แอดเดรส 2200h เนื่องจากเราจะจองพื้นที่ในตำแหน่ง 2000h-21FFh ไว้สำหรับการทำงานในการเก็บค่าของสแตคในกรณีที่เราไม่ได้กำหนดที่อยู่ของสแตคเอาไว้แน่นอน และหลังจากที่เราพัฒนาโปรแกรมได้สมบูรณ์แล้วเราก็สามารถทำการ copy โปรแกรมเราลงในไอซีอีพรอมเพื่อใช้งานได้เลย

- พอร์ตที่ต่อใช้งาน ในซีพียู 8031 นี้มีพอร์ตต่อใช้งานได้ทั้งหมด 4 พอร์ต คือ พอร์ต 0-3 แต่เนื่องจากบอร์ดของเราได้ใช้พอร์ต 0 กับ พอร์ต 2 ไปในการอ้างกับหน่วยความจำภายนอกแล้วและพอร์ต 3 ถูกใช้ไปในการรับส่งข้อมูลกับพีซีแล้วจึงเหลือเพียงพอร์ต 1 เท่านั้น เราจึงได้เพิ่มอุปกรณ์อินเตอร์เฟสเข้าไปในบอร์ดเพื่อขยายจำนวน I/O ของระบบซึ่งก็คือ 8255PPI โดยในตัวของ 8255 จะมีพอร์ตสำหรับการเชื่อมต่อกับอุปกรณ์ภายนอกแบบขนานอยู่ 3 พอร์ตด้วยกัน แต่ละพอร์ตมีจำนวน 8 บิท โดยตำแหน่งของพอร์ตใน 8255 เราจะทำการต่อโดยการกำหนดให้ 8255 เป็นส่วนหนึ่งของหน่วยความจำข้อมูลภายนอก โดยเราจะใช้ขา A0-A1 ของซีพียูต่อโดยตรงกับขา A0-A1 ของ 8255 ซึ่งในบอร์ดนี้เราได้กำหนดแอดเดรสพอร์ตของ 8255 ดังนี้

Port A = 0E0E0h

Port B = 0E0E1h

Port C = 0E0E2h

Port cont = 0E0E3h

โดยการเรียกใช้งานพอร์ตต่างๆ เราจะกำหนดให้ซีพียูมองพอร์ตทั้งหมดเหมือนกับการอ้างหน่วยความจำภายนอกตัวหนึ่งดังแอดเดรสข้างบนนี้ก็สามารถใช้งานได้

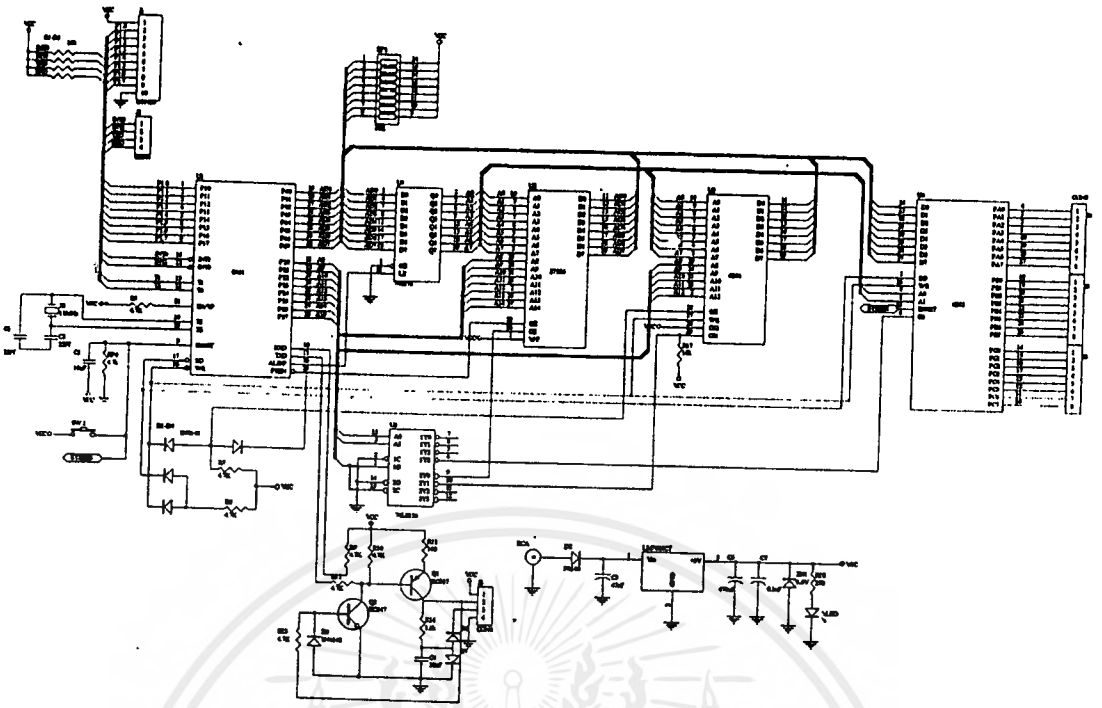
นอกจากในส่วนของบอร์ดคอนโทรลแล้วในส่วนของอินพุทที่รับคำสั่งการหมุนจากผู้ใช้ซึ่งเราใช้พอร์ต P1 เป็นพอร์ตอินพุทดังวงจรในรูปที่ 3.2 ซึ่งสวิตช์ทุกตัวจะต่อด้านหนึ่งลงกราวด์ และอีกด้านหนึ่งต่อกับพอร์ตโดยมีความด้านทาน R1-R8 ต่อพูลอัพอยู่ เมื่อสวิตช์เปิดจะได้ค่าลอจิก 1 เข้าที่ขาของพอร์ตและเมื่อสวิตช์ปิดจะได้ค่าลอจิก 0 โดยมีหน้าที่ของสวิตช์แต่ละตัวดังนี้

SW. 1 ทำหน้าที่เลือกการทำงานแบบ Auto หรือ Manual

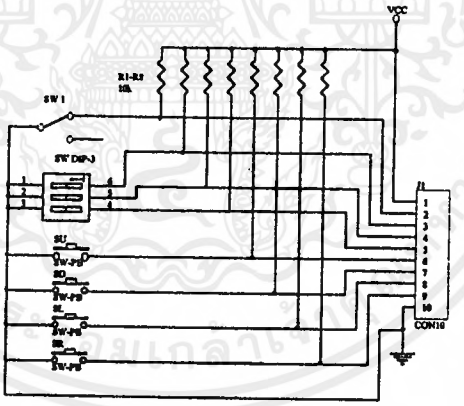
SW. dip 3 ทำหน้าที่เลือกชุดการขับมอเตอร์ชุด 1- 3

SW. U, SW. D, SW. L, SW. R ทำหน้าที่ปรับตำแหน่งการเคลื่อนที่ของมอเตอร์ตามสวิตช์การกดของผู้ใช้

การอ่านข้อมูลจากพอร์ตเราใช้คำสั่งให้การทำงานเป็น loop เพื่อเช็คการกดสวิตช์จากพอร์ต P1 โดยหากมีการกดสวิตช์ตัวใดตัวหนึ่งก็ให้ไปทำโปรแกรมย่อยตามที่กำหนดไว้แล้วกลับมาเช็คคีย์ที่เรากดไว้ในการทำคำสั่งครบหนึ่งรอบ



รูปที่ 3.1 แสดงวงจรคอนโทรลเลอร์ MCS-51 ที่ใช้



รูปที่ 3.2 สวิตช์ควบคุมการทำงานของชุดขับมอเตอร์

3.1.2 *ภาคการขับมอเตอร์* ก่อนที่เราจะทำการออกแบบภาคขับกำลังของมอเตอร์เราจะต้องรู้ถึงโครงสร้างของขดลวดภายในสเตปปิ้งมอเตอร์เสียก่อน ซึ่งสเตปปิ้งมอเตอร์ในโครงงานที่ใช้เป็นแบบยูนิโพล่าร์ 4 เฟส ชนิด 5 ,6 เส้น โครงสร้างภายในของสเตปปิ้งมอเตอร์จะประกอบด้วยขดลวด 2 ขด ซึ่งถ้าเป็นแบบ 5 เส้นก็นำมาต่อใช้งานได้ดังรูปเลย แต่ถ้าเป็นแบบ 6 เส้นเมื่อนำมาใช้ก็จะต้องทำการรวมขาที่แทปกลางของแต่ละขดรวมกันก่อนใช้งาน ส่วนตำแหน่งของขั้วแม่เหล็กคงที่ที่อยู่ในตัวมอเตอร์ นั้นก็จะมีการจัดเรียงแบบเฟสสลับกันไป

การควบคุมการหมุนของสเตปปิ้งมอเตอร์

สเตปปิ้งมอเตอร์จะหมุนได้นั้น ต้องมีการป้อนพัลซ์กระตุ้นเข้าที่ขดลวดแต่ละเฟสของ สเตปปิ้งมอเตอร์ให้เรียงกันไปในทิศทางเดียวกันถ้าต้องการให้หมุนกลับก็ป้อนพัลซ์ในทิศทางตรงข้ามกัน การป้อนพัลซ์กระตุ้นเพื่อขับสเตปปิ้งมอเตอร์มีด้วยกัน 3 วิธีดังที่ได้กล่าวมาแล้ว

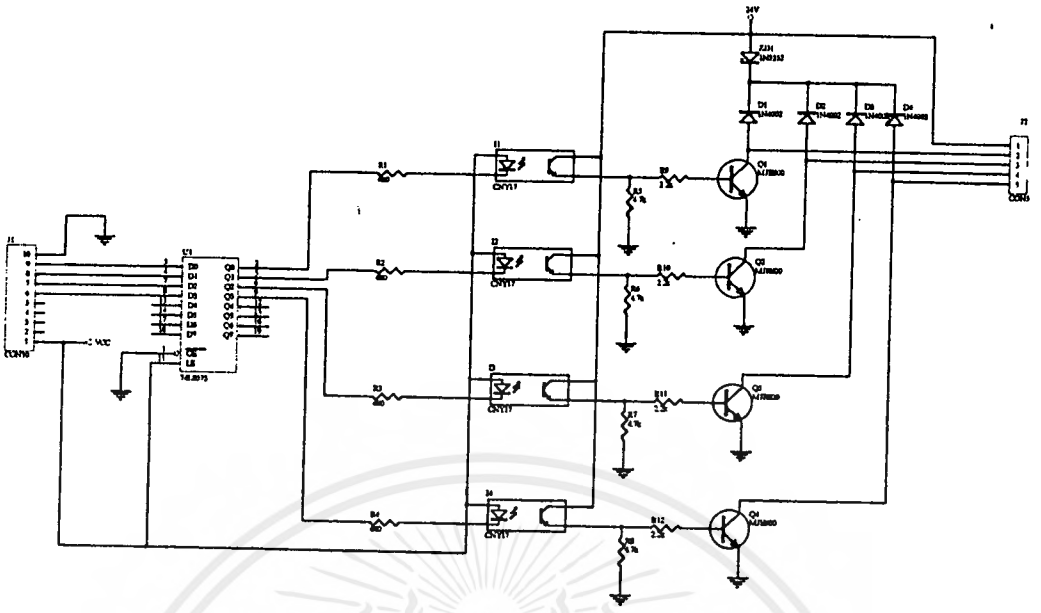
ถ้าพิจารณาจากไทม์มิ่งไดอะแกรมของการป้อนพัลซ์กระตุ้นการขับเฟส จะเห็นได้ว่าพัลซ์ที่ส่งออกไปกระตุ้นนั้นจะมีการเลื่อนไปทีละขั้น ซึ่งมีลักษณะเหมือนกับการเลื่อนข้อมูลในระบบไมโครโปรเซสเซอร์ ดังนั้นถ้าหากเราส่งข้อมูลดิจิตอลผ่านออกไปทางพอร์ตแล้วนำข้อมูลเหล่านั้นเข้าวงจรขับกระแสก่อน เอาต์พุทที่ได้จากวงจรขับกระแสนี้ก็สามารถนำไปควบคุมการหมุนของมอเตอร์ได้ เมื่อเปลี่ยนแปลงข้อมูลดิจิตอลก็จะทำให้ทิศทางการหมุนของมอเตอร์เปลี่ยนไป เช่น ถ้าต้องการขับมอเตอร์ให้หมุนด้วยการป้อนพัลซ์กระตุ้นแบบเดี่ยวเราอาจใช้ข้อมูลดิจิตอลไปป้อนวงจรไดร์เวอร์เพื่อขับสเตปปิ้งได้ดังนี้

ข้อมูลเริ่มต้น 1110
ข้อมูลเฟส1 0111
ข้อมูลเฟส2 1011
ข้อมูลเฟส3 1101
ข้อมูลเฟส4 1110

แล้ววนเช่นนี้ไปเรื่อยๆ มอเตอร์ก็จะหมุนไปในทิศทางเดียวกันอย่างต่อเนื่อง ส่วนวงจรที่ใช้จริงจะแสดงได้ดังรูปที่ 3.3 ซึ่งสามารถอธิบายการทำงานได้ดังที่จะกล่าวต่อไปนี้

การทำงานของภาคไดร์เวอร์สเตปปิ้งมอเตอร์

การทำงานของวงจรจะอาศัยการไบอัสค่าที่จะป้อนให้กับคู่ทรานซิสเตอร์ที่ต่อกันในลักษณะ Darlington pair ซึ่งจะทำหน้าที่ขยายกระแสที่จะป้อนให้กับขดลวดของมอเตอร์แต่ในที่นี้เราได้ใช้ TR npn Darl เบอร์ MJE 800 แทน ซึ่งเป็นทรานซิสเตอร์กำลังโดยข้อกำหนดทางไฟฟ้าสามารถทนกระแสได้สูงสุด (Ic max) ถึง 4 Amp เป็นตัวขับสเตปปิ้งมอเตอร์ ส่วนของ Opto-isolator เบอร์ 4N37 เป็นตัวทำหน้าที่แยกสัญญาณทางไฟฟ้าระหว่างบอร์ดกับภาคไดร์เวอร์ โดยมันจะรับลอจิกจากไอซีแลทช์ 74373 มาทำการสวิตช์แรงดันไฟ 24 V.เพื่อป้อนให้กับ MJE800 ทำการขับสเตปมอเตอร์ต่อไป



รูปที่ 3.3 วงจรภาคขับสเตปปิ้งมอเตอร์

3.1.3 การออกแบบ แมคคานิค(MACANICAL DESIGN)

1. การออกแบบส่วนหัว (HEAD) ส่วนหัวของหุ่นยนต์ใช้สำหรับติดตั้งกล้องโดยสามารถควบคุมให้หมุนกล้อง ซ้าย ขวา ได้โดยใช้มอเตอร์ทำงานในลักษณะแพน (pan head) หาเป้าหมายที่กำหนดได้

2. การออกแบบส่วนลำตัว (BODY) ส่วนตัวของหุ่นยนต์ใช้สำหรับเลื่อนแท่นแขนกลให้เลื่อนขึ้น- ลง โดยใช้แกนเกลียวเป็นตัวทำให้แท่นแขนกลเคลื่อนที่ได้ ปลายของแกนเกลียวด้านล่างเชื่อมกับแกนของมอเตอร์และใช้เหล็กยาว 2ชิ้นเป็นตัวยึดส่วนเคลื่อนที่ให้ตรงส่วนบนและล่าง ใช้แผ่นอลูมิเนียมหนา 1 ซม.เป็นตัวยึดเหล็กทั้ง 2 ชิ้น ด้านล่างประกบด้วยเหล็กฉาก 2 อันเพื่อประกอบลงบนส่วนเคลื่อนที่

3. การออกแบบส่วนแขน (ARM) ส่วนแขนใช้สำหรับยึดส่วนจับสิ่งของออกแบบง่าย ๆ โดยใช้ท่ออลูมิเนียมสี่เหลี่ยม 2 ท่อน ต่อกัน ส่วนที่ต่อกันมีเพลาเพื่อทำให้ท่อทั้งสองงอได้ปลายด้านหนึ่ง ติดตั้งมอเตอร์เชื่อมด้วยโซ่เพื่อควบคุมให้แขนกลงอขึ้นลงได้

4. การออกแบบส่วนจับสิ่งของ (GRIPPER) ส่วนจับสิ่งของติดตั้งตรงปลายของแขน โดยใช้แผ่นพลาสติกตัดเป็นรูปคล้ายตัวแอลหันชนกันติดตั้งบนแกน 2 อัน และใช้สปริงดันทั้ง 2 ส่วนออกจากกันตรงกลางใช้เชือกร้อยไปยังมอเตอร์เพื่อควบคุมให้จับหรือวางสิ่งของได้

5. ส่วนเคลื่อนที่(MOTION)ใช้สำหรับติดตั้งตัวหุ่นยนต์โดยส่วนนี้จะใช้ ดีซีมอเตอร์ เป็นตัวขับเคลื่อนล้อและอีกตัวใช้สำหรับบังคับให้หุ่นยนต์เลี้ยว ซ้าย-ขวา

3.2 **ซอร์ฟแวร์**

เมื่อเราออกแบบทางฮาร์ดแวร์เรียบร้อยแล้วหรือทราบความต้องการในงานของเราแล้วก็ถึงขั้นตอนของการเขียนโปรแกรมให้ระบบคอนโทรลเลอร์ทำงานตามวัตถุประสงค์ที่เราต้องการซึ่งขั้นตอนก็คือนำเอาโปรแกรมภาษาแอสเซมบลีไปทำการแปลงเป็นภาษาเครื่องเอาไปใส่ในอีพรอมแล้วเอามาต่อในวงจรที่ต้องการจะนำไปทำงานแต่ก็เป็นวิธีที่ไม่สะดวกนัก เราจึงอาศัยโปรแกรมมอนิเตอร์ที่เขียนอยู่ในอีพรอมโดยที่หน้าที่ของตัวโปรแกรมจะทำหน้าที่รับข้อมูลจากพีซีเข้าไปเก็บไว้ในหน่วยความจำแรมทางพอร์ตอนุกรม RS-232 เพื่อง่ายแก่การแก้ไขในส่วนของโปรแกรม ซึ่งในที่นี้เราสามารถเขียนโปรแกรมรับส่งข้อมูลทางพอร์ตอนุกรมได้หลายแบบแต่ในที่นี้เราต้องคำนึงถึงข้อกำหนดทางฮาร์ดแวร์โดยเราได้กำหนดไว้ดังนี้

กำหนดให้รับส่งข้อมูลทางพอร์ตอนุกรมด้วยอัตราการรับส่ง 9600 บิต/วินาที โดยใช้ Timer 1 สร้างอัตราส่งข้อมูลที่ทำงานในโหมด 1 เมื่อใช้ความถี่จากคริสตอลที่ 11.059 MHz เพื่อรับข้อมูลเข้ามาไว้ในหน่วยความจำแรมที่ตำแหน่ง 2200h เป็นต้นไปและจะหยุดรับข้อมูลเมื่อได้รับค่า FF ติดต่อกัน 2 ไบท์ (เพราะเมื่อใช้โปรแกรมคอมไพลเลอร์ เช่น cross-32 แล้วรูปแบบของ Hex file ที่ได้รับจะจบด้วย FF)

ตัวอย่างโปรแกรมมอนิเตอร์อย่างย่อๆ

```
                ORG 0000H
BUFFER          EQU 2200H     ; ส่วนที่นำข้อมูลไปเก็บไว้
                SJMP  MAIN    ; กระโดดข้ามส่วนอินเตอร์รัพต์
                ORG 0023H     ; ส่วนของการอินเตอร์รัพต์
                PUSH  PSW
                JNB   RI,END_RECEIVE
                MOV   A,SBUF
                CJNE  A,#0FFH,SAVE_BYTE  ;กระโดดเมื่อพบ FF
                INC   R7
                SJMP  SAVE1
SAVE_BYTE:      MOV   R7,#00
SAVE1:          MOVX  @DPTR,A
                INC   DPTR
                CLR   RI
END_RECEIVE:    POP   PSW
```

```
                RETI                ; จบส่วนการทำงานอินเตอร์รัพต์
MAIN:           MOV  TMOD,#20H      ; เซ็ทTimer
                MOV  TH1,#0FDH      ; อัตรา Baud 9600
                MOV  SCON,#40H
                ANL  PCON,#01111111B
                SETB TR1
                SETB ES
                SETB EA
                ........
                .......
                LCALL GETSERIAL
L1:             SJMP  L1
                ......
; ส่วนของการรับข้อมูลทางพอร์ตอนุกรมและหยุดรับเมื่อได้ข้อมูล FF
GETSERIAL:      PUSH  07
                MOV  R7,#00
                SETB SCON.4
                MOV  DPTR,#BUFFER
WAIT:           NOP
                CJNE R7,#2,WAIT
                CLR SCON.4
                POP  R7
                RET
                END
```

3.2.1 ส่วนของโปรแกรมควบคุมสเตปปิ้งมอเตอร์

การควบคุมให้สเตปปิ้งมอเตอร์ทำงานก็คือการกำหนดการขับเฟสของขดลวดภายในมอเตอร์ให้ได้รับพัลซ์การกระตุ้นในลักษณะต่อเนื่องกันไปเพื่อไปขับภาคไดร์เวอร์มอเตอร์อีกทีหนึ่ง โดยที่นี้ได้เขียนโปรแกรมอย่างง่ายเพื่อเป็นแนวคิดในการที่จะนำไปควบคุมมอเตอร์ในลักษณะของการต่อหลายชุดต่อไป

ตัวอย่างการโปรแกรมการขับมอเตอร์ 1 ชุด

```
                ORG 2200H
```

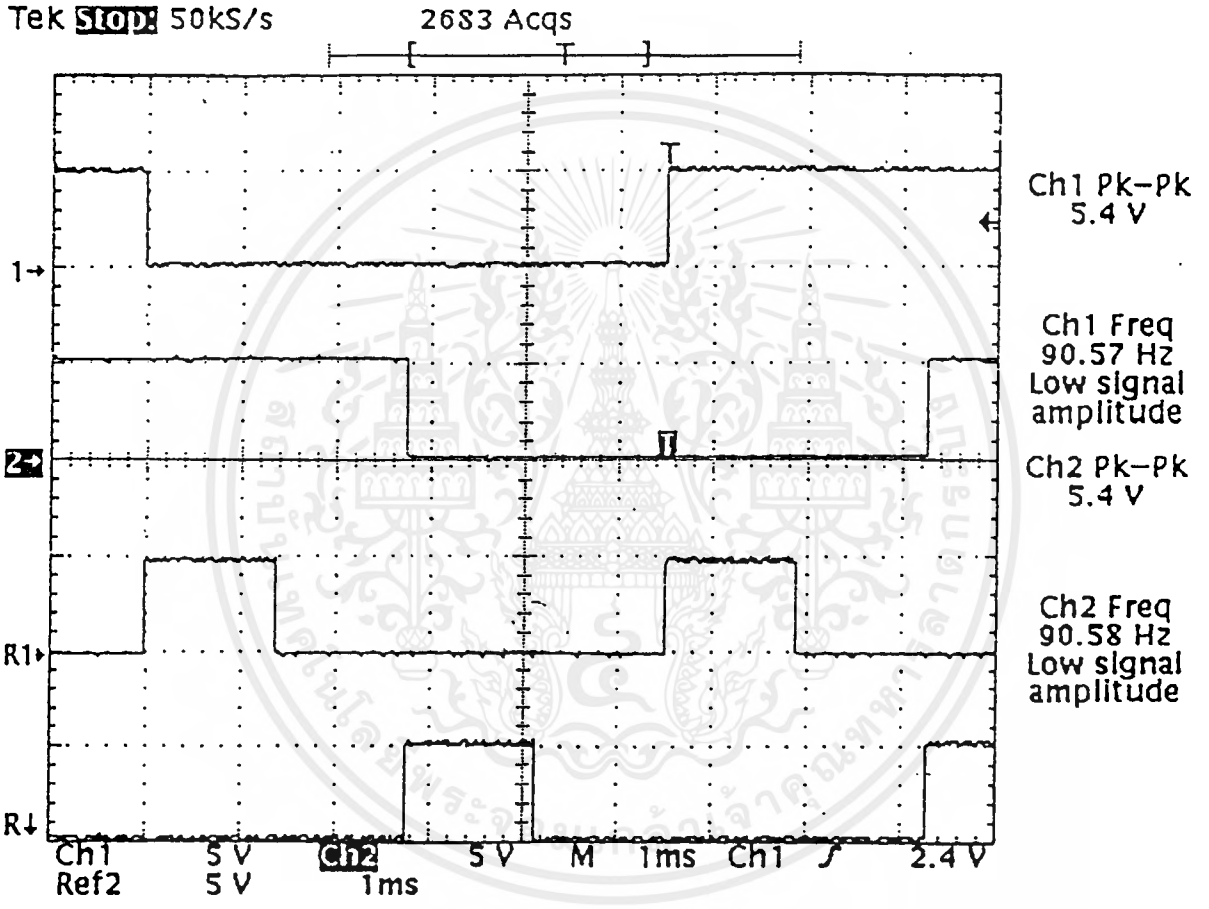
```
PORTA  EQU 0E0E0H        ; แอดเดรสพอร์ต 8255
PORTB  EQU 0E0E1H
PORTC  EQU 0E0E2H
CONT   EQU 0E0E3H
MAIN:  MOV A,#80H        ; กำหนด 8255 เป็นพอร์ต o/p
       MOV DPTR,#CONT
       MOVX @DPTR,A
       MOV A,#0EEH       ; initial สเตปของมอเตอร์
       MOV DPTR,#PORTB
       MOVX @DPTR,A
GO:    MOV A,P1          ; เริ่มต้นคอนโทรล
       JNB ACC.0,MANUAL  ; เลือกการทำงาน ถ้า Acc.0 =0
manual
       LJMP GO
       MANUAL:  JNB ACC.1,CHU1;check คีย์บน
       LJMP GO
CHU1:  JB ACC.4,CHD1     ; check คีย์ต่อไป
       LCALL UP1
       LJMP GO
UP1:   MOV R0,#0
LUP1:  MOV A,R0
       MOV DPTR,#TABLE1  ; สเต่ปการขับ
       MOVX  @DPTR,A
       MOV DPTR,#PORTB   ;เอาท์พอร์ต 8255
       MOVX @DPTR,A
       LCALL DELAY
       INC R0
       CJNE R0,#4,LUP1   ;1 loop = 4step
       MOV A,#0EEH
       MOVX @DPTR,A
       RET
```

```
CHD1:  JB ACC.5,CHL1          ; ตรวจสอบคีย์ต่อไป
       ....................
       DELAY: MOV R7,#0FFH    ; รูทีนปรับความเร็ว
       DEL1:  MOV R6,#0FH
       DEL2:  MOV R5,#01H
       DEL3:  NOP
         DJNZ R5,DEL3
         DJNZ R6,DEL2
         DJNZ R7,DEL1
         RET
TABLE1:  DB 11101110B    ; ตารางการขับ
         DB 01111110B
         DB 10111110B
         DB 11011110B
           ..............
           ..............
         END
```

บทที่ 4
การทดลองและผลการทดลอง

4.1 ผลการทดลอง

จากการเขียนโปรแกรมทดสอบภาคควบคุมสเตปปิ้งมอเตอร์วัดสัญญาณที่เอาท์พุตและแรงดันได้ดังนี้



รูปที่ 4.1 สัญญาณที่ออกจากชุดควบคุมไปยังภาคขับมอเตอร์

ตารางที่ 4 ผลการวัดแรงดันจากการทดลองที่ภาคขับ (driver)

Step	1	2	3	4	5
A	24v.	0.7v.	24v.	24v.	24v
B	0.7v.	24v.	24v.	24v.	24v.
C	24v.	24v.	24v.	0.7v.	24v.
D	24v.	24v.	0.7v.	24v.	24v.

Step	1	2	3	4	5
Sw.1	3.6v	0	3.6v.	3.6v.	3.6v.
Sw.2	0	3.6v.	3.6v.	3.6v.	3.6v.
Sw.3	3.6v.	3.6v.	3.6v.	0	3.6v.
Sw.4	3.6v.	3.6v.	0	3.6v.	3.6v.

บทที่ 5
สรุปและวิเคราะห์ผลการดำเนินงาน

จากการดำเนินการทำโครงงานดังกล่าวในขั้นแรกได้ทำการศึกษาถึงรายละเอียดต่างๆ ของดีซีมอเตอร์และสเตปปิ้งมอเตอร์โดยการศึกษาถึงทฤษฎีจากข้อมูลต่างๆ เมื่อทำการทดลองผลที่ได้ออกมามิใคร่จะตรงกับทางทฤษฎีเท่าใดนัก เช่น การทดลองการควบคุมดีซีมอเตอร์ด้วย ไมโครคอนโทรลเลอร์ โดยต่อเป็นวงจร PWM เพื่อควบคุมความเร็วของมอเตอร์ ค่าลอจิกที่ป้อนเข้าไปแปลงเป็นความถี่ไม่เที่ยงตรงทั้งนี้อาจเกิดจากส่วนของวงจรมีปัญหา อีกอย่างหนึ่งของการทดลองการขับสเตปปิ้งมอเตอร์เมื่อกำหนดความถี่ของสเตปซึ่งเรากำหนดโดยใช้ ลูปดีเลย์ผลที่ได้จะไม่ตรงกับทฤษฎีเท่าใดนักเนื่องคำนวณจำนวนสเตปของมอเตอร์ที่ใช้แต่ละจุดผิดพลาด

ในการทดลองต่อวงจรโดยรวมส่วนขับทั้งหมดแล้วทำการรันโปรแกรมจะเกิดปัญหาขึ้นคือมอเตอร์แต่ละตัวจะทำงานไม่เป็นอิสระเกิดการดึงหรือหน่วงขึ้นของชุดที่ไม่ได้ถูกสั่งให้ทำงานเป็นผลให้ชุดแมคคานิคของเราเกิดการผิดพลาดได้ โดยเราสามารถสรุปปัญหาที่เกิดขึ้นได้ดังนี้

1. ส่วนของบอร์ดคอนโทรล MCS-51 ต้องใช้เวลาในการศึกษาใหม่ทั้งหมดจึงทำให้เสียเวลาในส่วนนี้มาก

2. ในการออกแบบด้านแมคคานิคนั้น มีความจำกัดทางด้านความรู้และการจัดหาเครื่องมือ,อุปกรณ์ได้ลำบาก

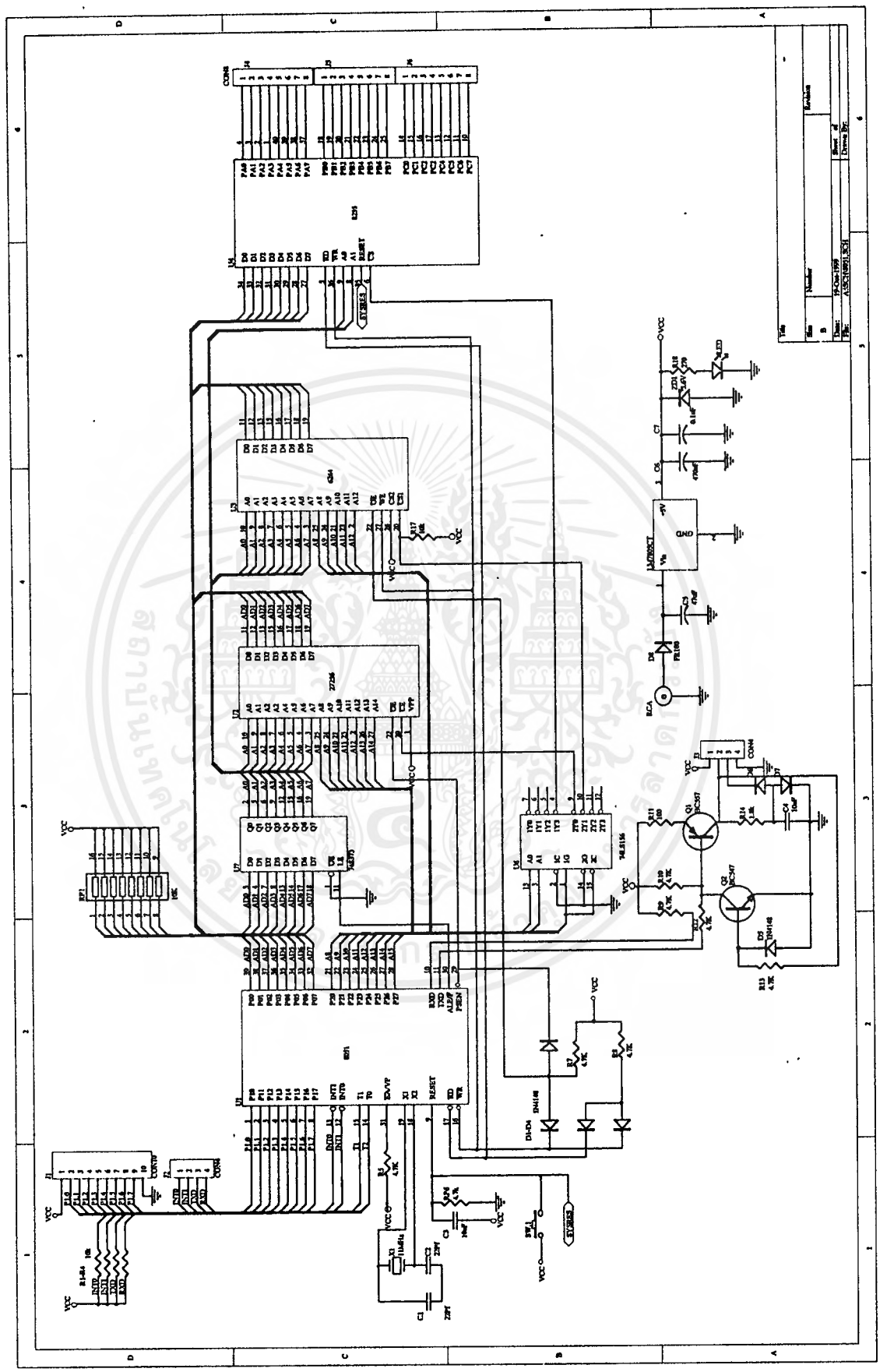
3. ในส่วนของวงจรภาคขับมอเตอร์นั้นต้องการแหล่งจ่ายไฟที่ให้กำลังในการขับสูงเนื่องจากการขับมอเตอร์ให้ทำงานรวมกับส่วนของแมคคานิคที่มีภาระการทำงานหนักจึงต้องคำนึงในส่วนภาคจ่ายไฟด้วย

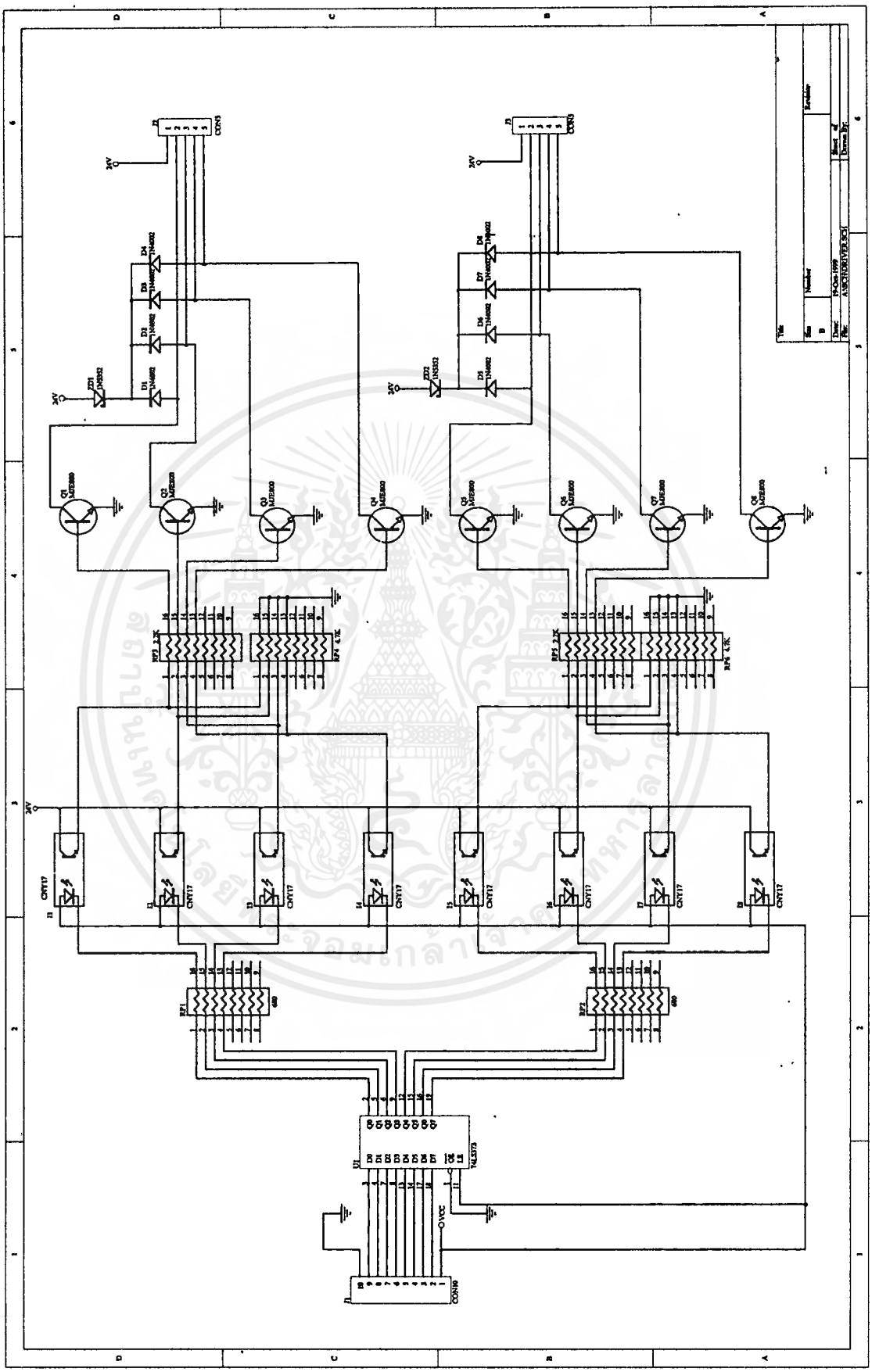
เมื่อนำทุกส่วนประกอบเข้ากันแล้วทำการทดลองควบคุมในส่วนต่างๆ แล้วผลที่ได้ไม่ค่อยมีความลงตัว ไม่สามารถควบคุมบางจุดที่มีภาระของแมคคานิคสูงให้ทำงานในส่วนที่ต้องการได้จึงจำเป็นต้องมีการพัฒนาต่อไป

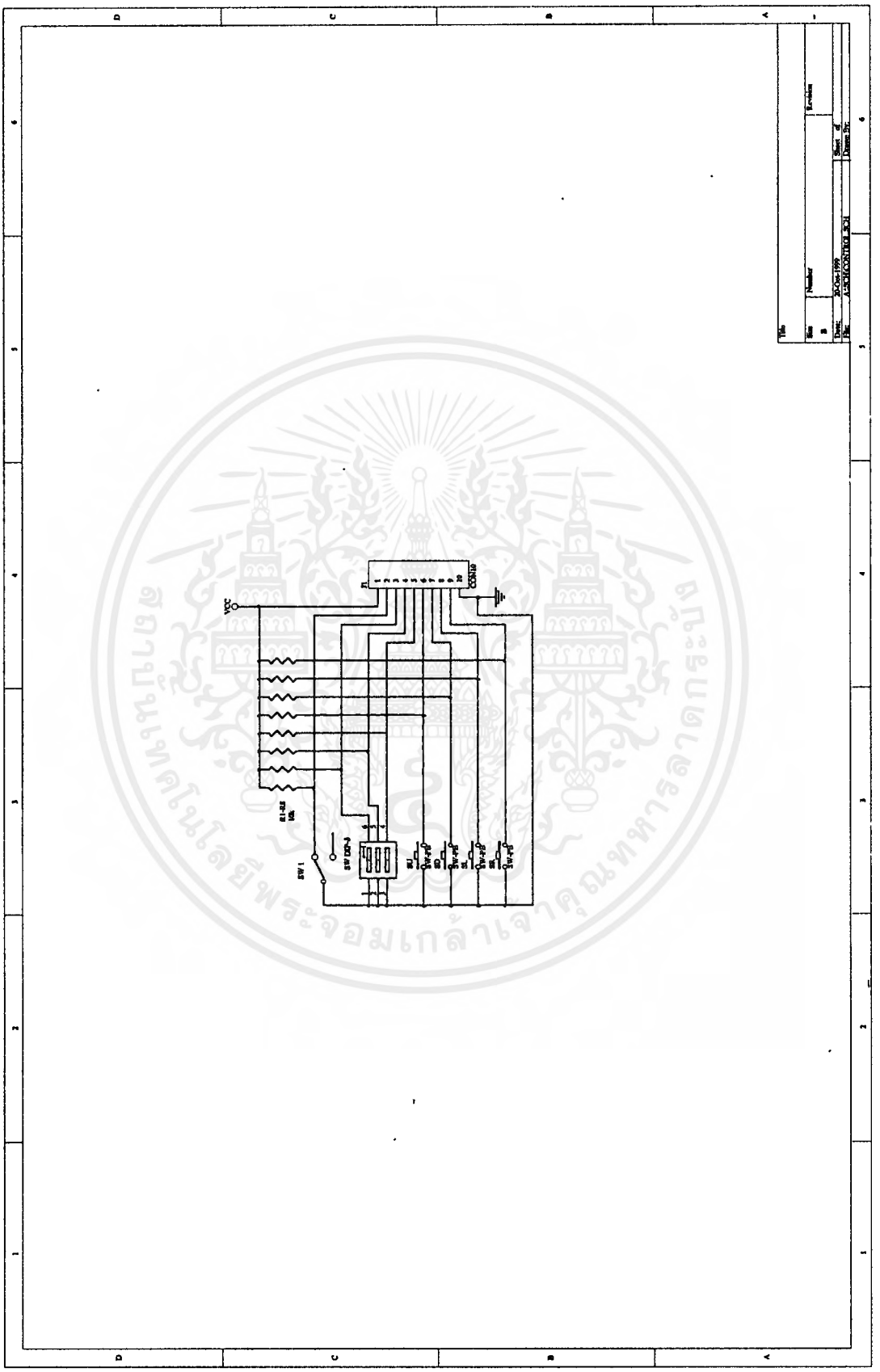
<u>ข้อเสนอแนะ</u> ควรมีการพัฒนาในส่วนของแมคคานิคให้มีคล่องตัวและสามารถที่จะคอนโทรลได้ง่ายกว่าที่เป็นอยู่ และในส่วนของการใช้งานจริงควรมีส่วนของการควบคุมในระยะไกลได้โดยใช้คลื่นวิทยุในการควบคุมก็ได้

ภาคผนวก ก

วงจรที่ใช้งาน







ภาคผนวก ข

โปรแกรมที่ใช้งาน

```
            ORG 0000H
PORTA   EQU 0E0E0H          ;decode 8255 port;
PORTB   EQU 0E0E1H
PORTC   EQU 0E0E2H
CONT    EQU 0E0E3H
 MAIN:  MOV A,#80H          ;set port 8255= O/P;
        MOV DPTR,#CONT
        MOVX @DPTR,A
        MOV A,#0EEH         ;set all port 8255 intitail;
       MOV DPTR,#PORTA
        MOVX @DPTR,A
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        MOV DPTR,#PORTC
        MOVX @DPTR,A
  GO1:  MOV A,P1            ;read p1;
       JNB ACC.0,MANUAL     ;if p1.0=0 run manual;
       LJMP AUTO
MANUAL: JNB ACC.1,CHU1      ;check p1.1=0 run drive motor1;
        JNB ACC.2,FOR_1GO   ;check p1.2=0 run drive motor2;
        JNB ACC.3,FOR_2GO   ;check p1.3=0 run drive motor3;
       LJMP GO1
FOR_1GO: LCALL CHU2
        RET
FOR_2GO: LCALL CHU3
        RET

;------ DRIVE MOTOR 1 -----;  ;for motor drive1;
  CHU1: JB ACC.4,CHD1       ;check sw.u if close run;
      LCALL UP1
      LJMP GO1              ;go to check p1 again;
   UP1: MOV R0,#0           ;loop drive;
  LUP1: MOV A,R0
      MOV DPTR,#TABLE1      ;look up step drive;
      MOVC A,@A+DPTR
```

```
        MOV DPTR,#PORTA
        MOVX @DPTR,A
        LCALL DELAY          ;delay loop for speed;
        INC R0
        CJNE R0,#4,LUP1      ;1 loop for 4step;
        MOV A,#0EEH          ;intitial step;
        MOVX @DPTR,A
        RET
 CHD1:  JB ACC.5,CHL1        ;check sw.d if close run;
        LCALL DOWN1
        LJMP GO1             ;go to check p1 again;
 DOWN1: MOV R0,#0            ;loop drive;
LDOWN1: MOV A,R0
        MOV DPTR,#TABLE2     ;look up step drive;
        MOVC A,@A+DPTR
        MOV DPTR,#PORTA
        MOVX @DPTR,A
        LCALL DELAY          ;delay loop for speed;
        INC R0
        CJNE R0,#4,LDOWN1
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
 CHL1:  JB ACC.6,CHR1
        LCALL LEFT1
        LJMP GO1
 LEFT1: MOV R0,#0
LLEFT1: MOV A,R0
        MOV DPTR,#TABLE3
        MOVC A,@A+DPTR
        MOV DPTR,#PORTA
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LLEFT1
```

```
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
  CHR1: JB ACC.7,GO1
        LCALL RIGHT1
        LJMP GO1
 RIGHT1: MOV R0,#0
LRIGHT1: MOV A,R0
        MOV DPTR,#TABLE4
        MOVC A,@A+DPTR
        MOV DPTR,#PORTA
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LRIGHT1
        MOV A,#0EEH
        MOVX @DPTR,A
        RET

;------ DRIVE MOTOR 2 ------;    ;for motor drive2;
  CHU2: JB ACC.4,CHD2
        LCALL UP2
        LJMP GO1
   UP2: MOV R0,#0
  LUP2: MOV A,R0
        MOV DPTR,#TABLE1
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LUP2
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
```

```
 CHD2:  JB ACC.5,CHR2
        LCALL DOWN2
        LJMP GO1
DOWN2:  MOV R0,#0
LDOWN2:  MOV A,R0
        MOV DPTR,#TABLE2
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LDOWN2
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
 CHL2:  JB ACC.6,CHR2
        LCALL LEFT2
        LJMP GO1
LEFT2:  MOV R0,#0
LLEFT2:  MOV A,R0
        MOV DPTR,#TABLE3
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LLEFT2
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
 CHR2:  JB ACC.7,FOR_3GO
        LCALL RIGHT2
        LJMP GO1
FOR_3GO:  LCALL GO1
        RET
```

```
 RIGHT2:  MOV R0,#0
LRIGHT2:  MOV A,R0
          MOV DPTR,#TABLE4
          MOVC A,@A+DPTR
          MOV DPTR,#PORTB
          MOVX @DPTR,A
          LCALL DELAY
          INC R0
          CJNE R0,#4,LRIGHT2
          MOV A,#0EEH
          MOVX @DPTR,A
          RET

;------- DRIVE MOTOR 3 ------;   ;for motor drive3;
   CHU3:  JB ACC.4,CHD3
          LCALL UP3
          LJMP GO1
    UP3:  MOV R0,#0
   LUP3:  MOV A,R0
          MOV DPTR,#TABLE1
          MOVC A,@A+DPTR
          MOV DPTR,#PORTC
          MOVX @DPTR,A
          LCALL DELAY
          INC R0
          CJNE R0,#4,LUP3
          MOV A,#0EEH
          MOVX @DPTR,A
          RET
   CHD3:  JB ACC.5,CHL3
          LCALL DOWN3
          LJMP GO1
  DOWN3:  MOV R0,#0
 LDOWN3:  MOV A,R0
          MOV DPTR,#TABLE2
```

```
        MOVC A,@A+DPTR
        MOV DPTR,#PORTC
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LDOWN3
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
 CHL3:  JB ACC.6,CHR3
        LCALL LEFT3
        LJMP GO1
LEFT3:  MOV R0,#0
LLEFT3: MOV A,R0
        MOV DPTR,#TABLE3
        MOVC A,@A+DPTR
        MOV DPTR,#PORTC
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LLEFT3
        MOV A,#0EEH
        MOVX @DPTR,A
        RET
 CHR3:  JB ACC.7,FOR_4GO
        LCALL RIGHT3
        LJMP GO1
FOR_4GO: LCALL GO1
        RET
RIGHT3: MOV R0,#0
LRIGHT3: MOV A,R0
        MOV DPTR,#TABLE4
        MOVC A,@A+DPTR
        MOV DPTR,#PORTC
        MOVX @DPTR,A
```

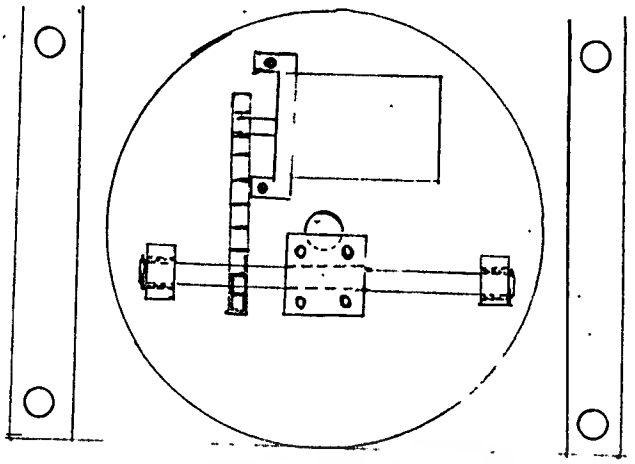
```
        LCALL DELAY
        INC R0
        CJNE R0,#4,LRIGHT3
        MOV A,#0EEH
        MOVX @DPTR,A
        RET

;****** AUTO DRIVE ******;      ;for auto drive;
  AUTO:  MOV A,#0
  GO2:  MOV R1,#0H              ;set start table;
        MOV R2,#0H
        MOV R3,#0H
        MOV R4,#0H
START1:  MOV R0,#0H
 LOOP1:  MOV A,R0
        MOV DPTR,#TABLE1        ;look up step drive;
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB         ;select motor drive?;
        MOVX @DPTR,A
        LCALL DELAY             ;loop delay for speed;
        INC R0
        CJNE R0,#4,LOOP1        ;1 loop for 4 step;
        INC R1
        CJNE R1,#032H,START1    ;amount cycle?;
        MOV A,#0EEH
        MOVX @DPTR,A
        LCALL DELAY             ;loop delay = stop;
START2: MOV R0,#0H
 LOOP2: MOV A,R0
        MOV DPTR,#TABLE2
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
```

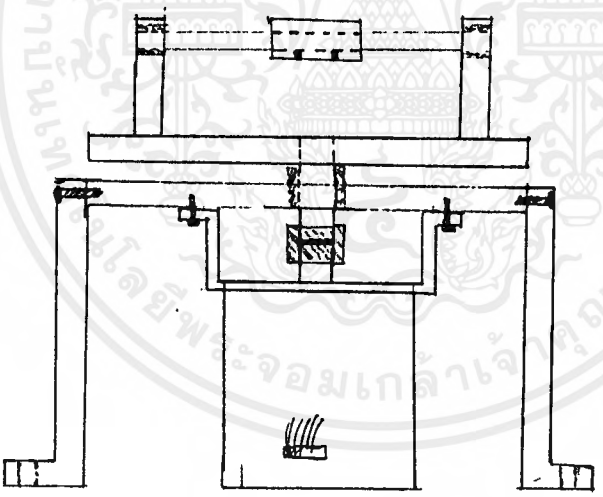
```
        CJNE R0,#4,LOOP2
        INC R2
        CJNE R2,#032H,START2
        MOV A,#0EEH
        MOVX @DPTR,A
        LCALL DELAY
START3:  MOV R0,#0H
 LOOP3:  MOV A,R0
        MOV DPTR,#TABLE3
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LOOP3
        INC R3
        CJNE R3,#032H,START3
        MOV A,#0EEH
        MOVX @DPTR,A
        LCALL DELAY
START4:  MOV R0,#0H
 LOOP4:  MOV A,R0
        MOV DPTR,#TABLE4
        MOVC A,@A+DPTR
        MOV DPTR,#PORTB
        MOVX @DPTR,A
        LCALL DELAY
        INC R0
        CJNE R0,#4,LOOP4
        INC R4
        CJNE R4,#032H,START4
        MOV A,#0EEH
        MOVX @DPTR,A
        LCALL DELAY
        SJMP GO2
```

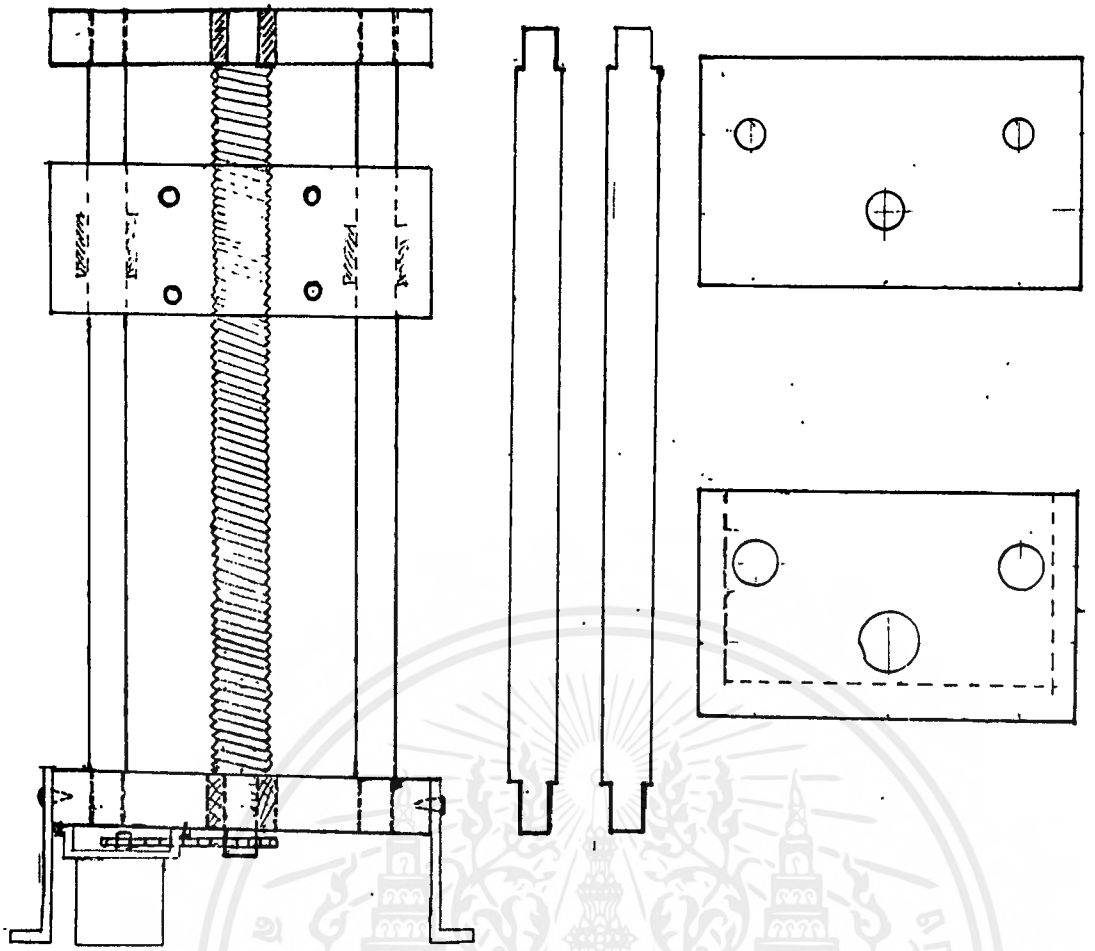
```
DELAY: MOV R7,#0FFH          ;loop set delay;
DEL1:  MOV R6,#0FH
DEL2:  MOV R5,#01H
DEL3:  NOP
       DJNZ R5,DEL3
       DJNZ R6,DEL2
       DJNZ R7,DEL1
       RET
TABLE1:  DB 11101110B        ;table for step drive;
       DB 01111110B
       DB 10111110B
       DB 11011110B
TABLE2:  DB 11101110B
       DB 11011110B
       DB 10111110B
       DB 01111110B
TABLE3:  DB 11101110B
       DB 11100111B
       DB 11101011B
       DB 11101101B
TABLE4:  DB 11101110B
       DB 11101101B
       DB 11101011B
       DB 11100111B
       END
```

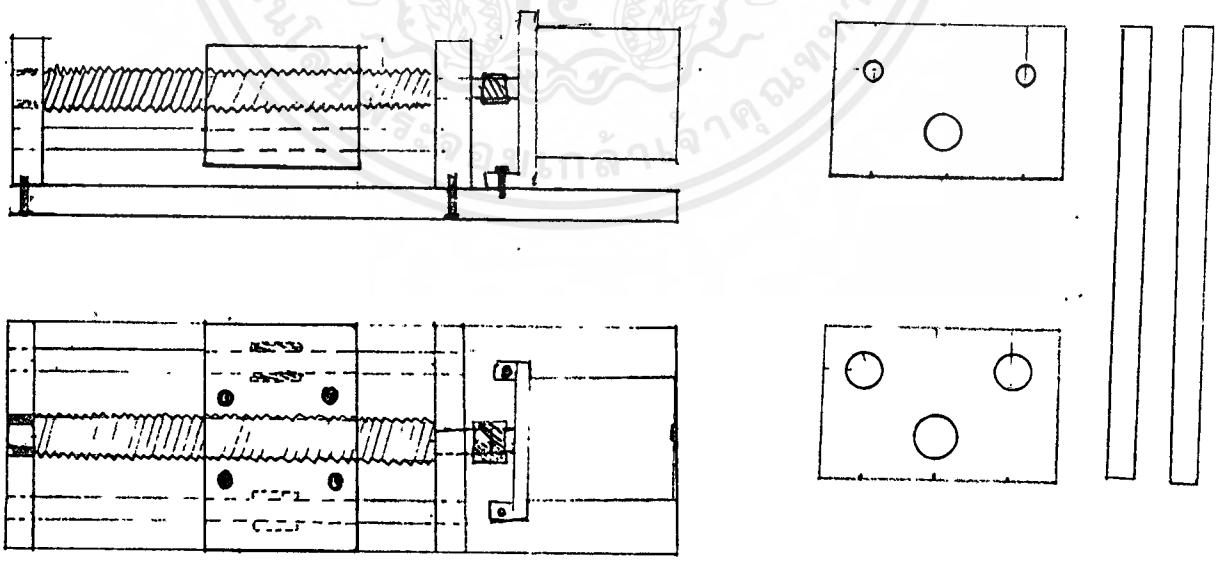
ภาคผนวก ค

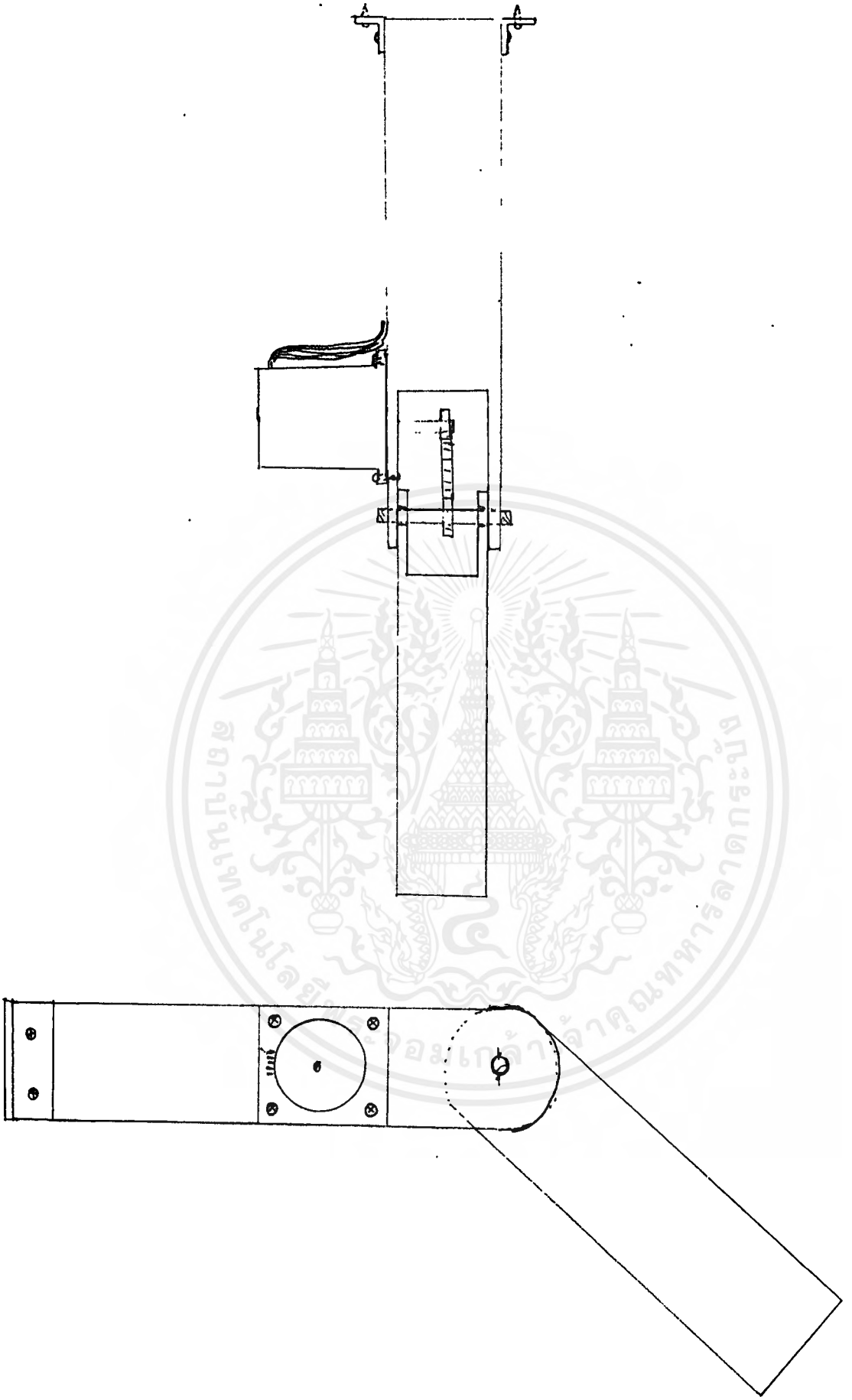
ชุดแมคคานิค

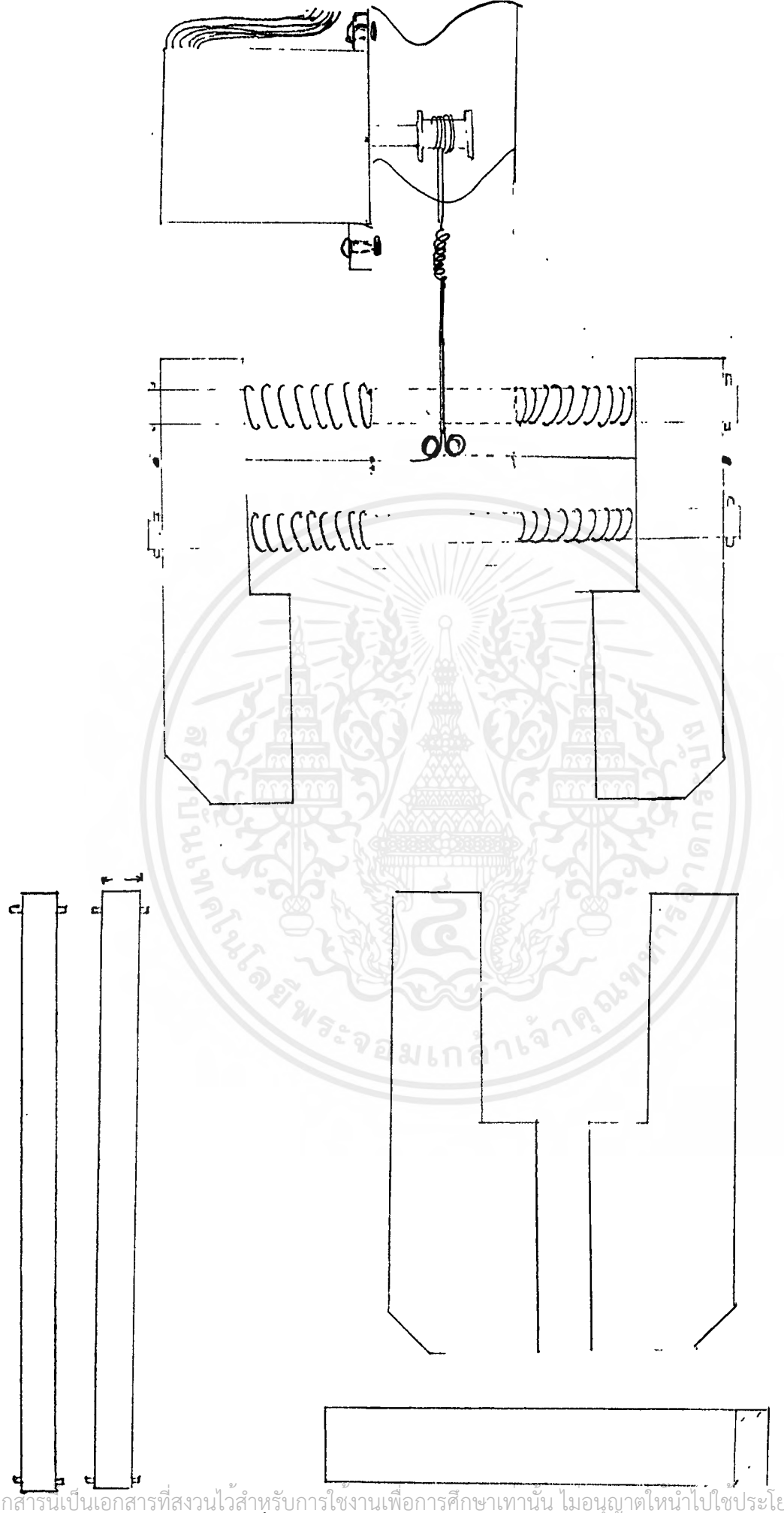


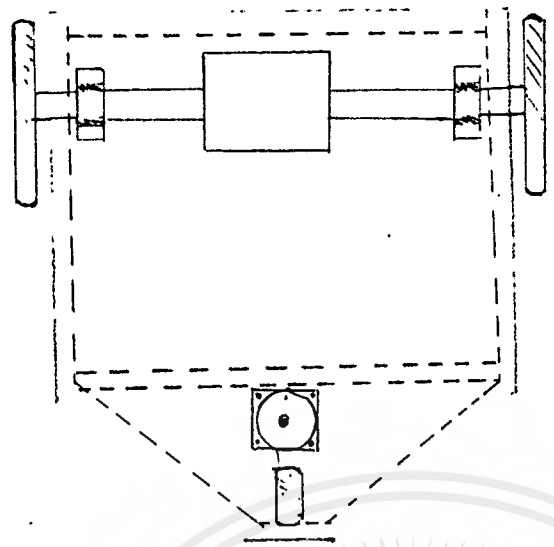


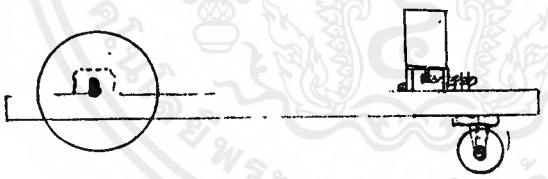












ภาคผนวก ง

DATA SHEET

intel

PRELIMINARY

MCS®-51
8-BIT CONTROL-ORIENTED MICROCOMPUTERS
8031/8051
8031AH/8051AH
8032AH/8052AH
8751H/8751H-8

- High Performance HMOS Process
- Internal Timers/Event Counters
- 2-Level Interrupt Priority Structure
- 32 I/O Lines (Four 8-Bit Ports)
- 64K Program Memory Space
- Security Feature Protects EPROM Parts Against Software Piracy
- Boolean Processor
- Bit-Addressable RAM
- Programmable Full Duplex Serial Channel
- 111 Instructions (64 Single-Cycle)
- 64K Data Memory Space

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

The 8051 is the original member of the MCS-51 family. The 8051AH is identical to the 8051, but it is fabricated with HMOS II technology.

The 8751H is an EPROM version of the 8051AH; that is, the on-chip Program Memory can be electrically programmed, and can be erased by exposure to ultraviolet light. It is fully compatible with its predecessor, the 8751-8, but incorporates two new features: a Program Memory Security bit that can be used to protect the EPROM against unauthorized read-out, and a programmable baud rate modification bit (SMOD). The 8751H-8 is identical to the 8751H but only operates up to 8 MHz.

The 8052AH is an enhanced version of the 8051AH. It is backwards compatible with the 8051AH and is fabricated with HMOS II technology. The 8052AH enhancements are listed in the table below. Also refer to this table for the ROM, ROMless, and EPROM versions of each product.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	8
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

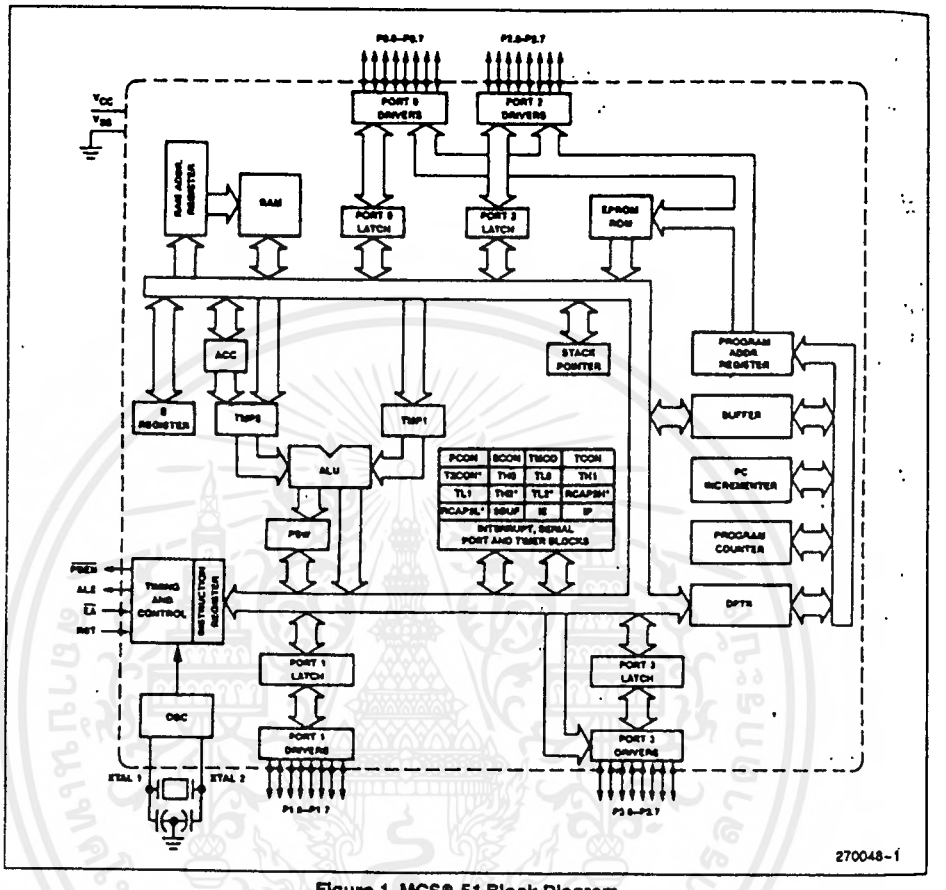


Figure 1. MCS®-51 Block Diagram

PACKAGES

Part	Prefix	Package Type
8051AH/ 8031AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8052AH/ 8032AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8751H/ 8751H-8	D	40-Pin CERDIP
	R	44-Pin LCC

PIN DESCRIPTIONS

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

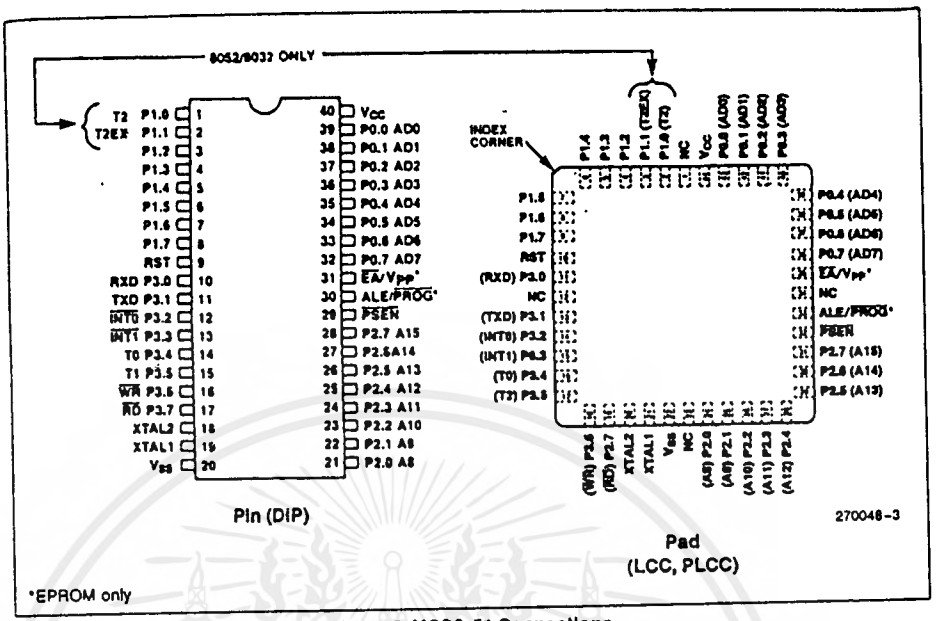


Figure 2. MCS®-51 Connections

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during programming of the EPROM parts.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}$/V_{PP}: External Access enable $\overline{EA}$ must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations starting at 0000H up to FFFFH. $\overline{EA}$ must be strapped to V_{CC} for internal program execution.

Note, however, that if the Security Bit in the EPROM devices is programmed, the device will not fetch code from any location in external Program Memory.

This pin also receives the 21V programming supply voltage (VPP) during programming of the EPROM parts.

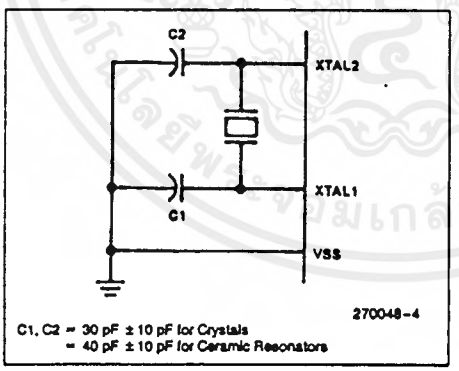


Figure 3. Oscillator Connections

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

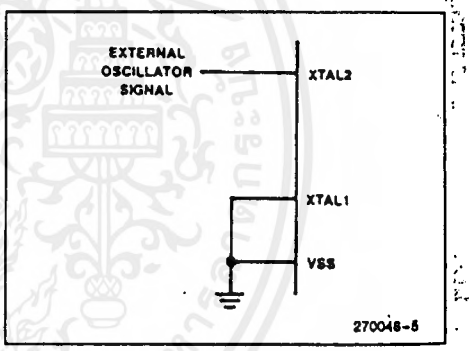


Figure 4. External Drive Configuration

DESIGN CONSIDERATIONS

If an 8751BH or 8752BH may replace an 8751H in a future design, the user should carefully compare both data sheets for DC or AC Characteristic differences. Note that the V_{IH} and I_{IH} specifications for the $\overline{EA}$ pin differ significantly between the devices.

Exposure to light when the EPROM device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

บรรณานุกรม

โยธิน เปรมปราณีรัชต์. ระบบเซอร์โวและอิเล็คทรอนิกส์คอนโทรลมอเตอร์. พิมพ์ครั้งที่1 :สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

อุดม จีนประดับ. ไมโครคอนโทรลเลอร์ MCS 51. พิมพ์ครั้งที่1 : สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

INTEL.Microprocesseor Data book MCS-51 Microcontrollers. Intel Coporation,1983

คู่มือบอร์ดคอนโทรลเลอร์ MSC-51.บริษัทอีทีจำกัด.