การควบคุมสเต็ปปิงมอเตอร์ 3 แนวแกน

3 AXES CONTROL OF STEPPING MOTORS

โดย

นายชาย แซ่เล้า

นายทรงพล น้ำแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตร์ สาขาวิศวกรรมอิเล็กทรอนิกส
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2539

การควบคุมสเต็ปปิ้งมอเตอร์ 3 แนวแกน

นักศึกษา นายชาย แซ่เล้า (36014114)
นายทรงพล น้ำแก้ว (36014154)
อาจารย์ที่ปรึกษา อ. ขนิษฐา แซ่ตั้ง
ปีการศึกษา 2539

บทคัดย่อ

สำหรับโครงงานที่จัดทำขึ้นนี้ เราได้นำสเต็ปปิ้งมอเตอร์มาประยุกต์ใช้งานเป็นตัวควบคุมการเคลื่อนที่ใน 3 ทิศทาง หรือ 3 แนวแกน คือในแนวแกน X, แกน Y, และแกน Z พร้อมกันนี้ยังได้สร้างส่วนของกลไกเพื่อให้เห็นผลของการควบคุมได้ชัดเจนและเป็นแนวทางในการพัฒนาให้เป็นอุปกรณ์ที่สามารถใช้งานได้จริง เช่น ทำเป็นเครื่องเจาะ เป็นต้น ในส่วนของการควบคุมได้นำไมโครคอนโทรลเลอร์มาใช้งาน เนื่องจากการทำงานของสเต็ปปิ้งมอเตอร์อาศัยพัลส์ ทำให้การเชื่อมต่อกับอุปกรณ์ทางด้านดิจิตอล เช่น ไมโครคอนโทรลเลอร์ ทำได้ง่าย นอกจากนั้นไมโครคอนโทรลเลอร์ยังมีข้อดีในเรื่องความยืดหยุ่นของซอฟแวร์ ทำให้การควบคุมการทำงานมีประสิทธิภาพ

3 Axes control of stepping motors

Student Mr.Chai Saelou (36014114)

Mr.Songpol numkaew (36014154)

Adviser Lecturer Kanittha Seatang

Academic year 1996

Abstract

This project is to be the position control of three stepping motor in three dimentions or three axes , x axis , y axis and z axis . In addition we have made a mechanical parts to show how they work and lead the way for future development to be a practical useful product. The controller unit consist of Microcontroller MCS - 48 . So that is the easy way to control the stepping motor and is the advantage of flexible software.

คำนำ

ปริญญานิพนธ์ฉบับนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชา Project 1 ของเทอมที่ 1 และวิชา Project 2 ของเทอมที่ 2 ภาควิชาอิเล็กทรอนิคส์ ประจำปีการศึกษา 2539

เนื้อหาภายในเป็นการอธิบายรายละเอียดที่เกี่ยวกับโครงงานที่ได้สร้างขึ้นนั่นคือการควบคุมสเต็ปปิ้งมอเตอร์ 3 แนวแกน ภายในเล่มนี้แบ่งออกได้เป็น 4 บท ดังนี้

บทที่ 1 บทนำ เป็นการอธิบายโครงสร้างโดยรวมของโครงงาน โดยแสดงเป็นบล็อกไดอะแกรม และอธิบายการทำงานของแต่ละส่วน นอกจากนี้ยังบอกวัตถุประสงค์และประโยชน์ที่ได้รับจากโครงงาน

บทที่ 2 ทฤษฎีพื้นฐาน และ หลักการทำงานของสเต็ปปิ้งมอเตอร์ เป็นการอธิบายรายละเอียดของโครงสร้าง , ชนิด และ การใช้งานของสเต็ปปิ้งมอเตอร์

บทที่ 3 โครงสร้างของไมโครคอนโทรลเลอร์ MCS-48 และการใช้งาน ได้อธิบายโครงสร้างและสถาปัตย์กรรมของไมโครคอนโทรลเลอร์ที่ถูกเลือกใช้ในโครงงานนี้

บทที่ 4 รายละเอียดของโครงงานและการออกแบบ เป็นการอธิบายส่วนประกอบของโครงงานที่ทำขึ้น , วิธีการออกแบบ และ หลักการทำงาน

ผู้จัดทำหวังว่าโครงงานและปริญญานิพนธ์ ที่ได้จัดทำขึ้นนี้จะเป็นประโยชน์สำหรับผู้ที่สนใจศึกษาเรื่องการใช้งานของสเต็ปปิ้งมอเตอร์ และคงนำความรู้ที่ได้ไปพัฒนาให้เกิดประโยชน์ต่อไป

สารบัญ

บทที่ 1
บทนำ

มอเตอร์เป็นอุปกรณ์ที่ใช้ในการเปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล มอเตอร์มีหลายชนิด เช่น ดีซีมอเตอร์, เอซีมอเตอร์, สเต็ปปิ้งมอเตอร์ เป็นต้น มอเตอร์ถูกนำมาใช้ในการควบคุมตำแหน่งและความเร็ว เช่น การนำมอเตอร์มาเป็นตัวขับเคลื่อนให้แขนกลของหุ่นยนต์สามารถเคลื่อนไหวได้ตามการควบคุม การใช้มอเตอร์ที่เป็นแบบเอซีมอเตอร์หรือดีซีมอเตอร์ มีข้อเสียตรงที่การควบคุมตำแหน่งหรือความเร็วต่าง ๆ ทำได้ยาก เพราะต้องใช้การควบคุมแบบระบบปิดกล่าวคือต้องมีอุปกรณ์ที่ใช้ตรวจจับการหมุนของมอเตอร์ แล้วป้อนสัญญาณกลับมายังส่วนควบคุม ด้วยเหตุนี้เองจึงมีผู้นิยมหันมาใช้สเต็ปปิ้งมอเตอร์แทน เพราะมีคุณสมบัติที่ดีหลายประการคือ

1. สเต็ปปิ้งมอเตอร์เป็นอุปกรณ์ที่เปลี่ยนสัญญาณดิจิตอลไปเป็นการเคลื่อนที่ทางกล ดังนั้นการติดต่อกับอุปกรณ์ดิจิตอล เช่น ไมโครคอนโทรลเลอร์ ทำได้ง่ายและวงจรขยายกำลังจากสัญญาณดิจิตอล (Digital Power Amplifier) ที่อยู่ในส่วนวงจรขับ (Driver) ทำได้ง่ายและมีราคาถูก
2. การออกแบบวงจรควบคุมสเต็ปปิ้งมอเตอร์สามารถทำได้ง่ายกว่าวงจรควบคุมมอเตอร์แบบเซอร์โว และยังสามารถออกแบบวงจรให้เสต็ปปิ้งมอเตอร์ทำงานหรือหยุดได้แบบทันทีทันใด

1.1 รายละเอียดโครงงาน

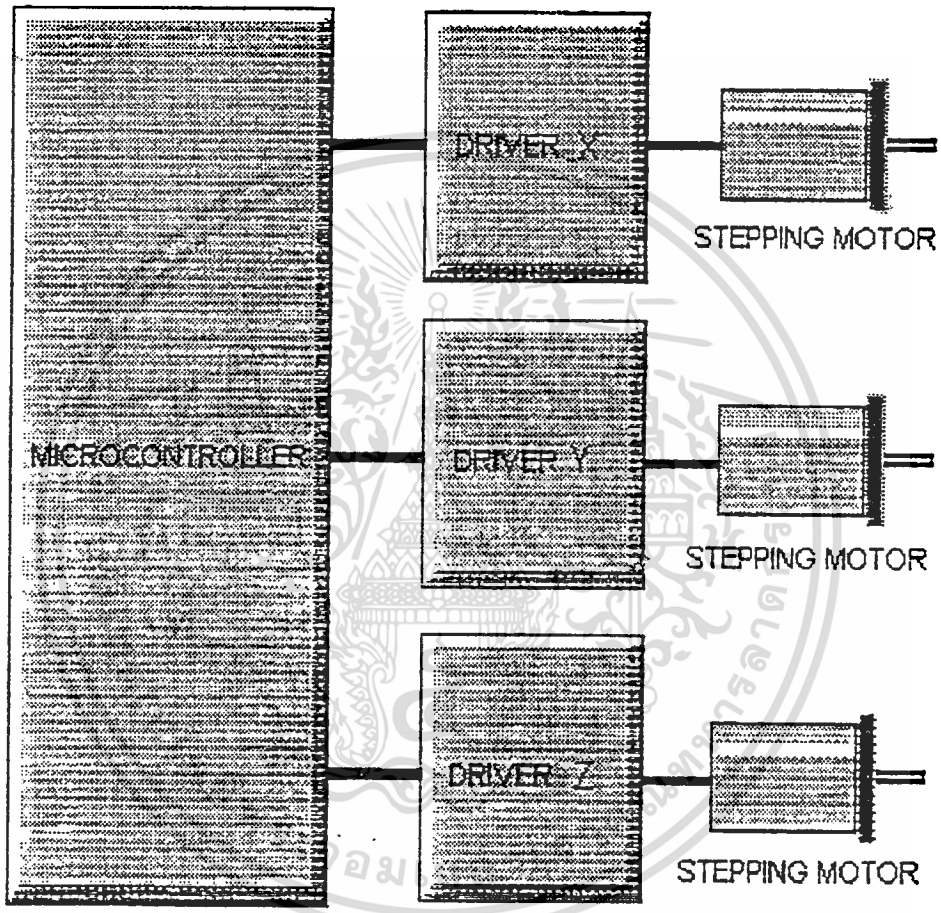
โครงงานที่ได้จัดทำขึ้นนี้เป็นการทดลองสร้างระบบควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ 3 ตัว โดยแต่ละตัวจะถูกนำไปประยุกต์ใช้งานเป็นตัวควบคุมการเคลื่อนที่ของกลไกใน 3 ทิศทาง หรือ 3 แนวแกน คือ แนวแกน X , แนวแกน Y, แนวแกน Z ยกตัวอย่างเช่น เครื่องมือที่ใช้งานได้จริง เช่น เครื่องเจาะ CNC ,เครื่องเจาะแผ่นปริ้นท์ เป็นต้น แต่โครงงานนนี้ไม่ได้มุ่งเน้นไปที่การสร้างอุปกรณ์ที่สามารถใช้งานได้จริง แต่เน้นถึงระบบการควบคุมสเต็ปปิ้งมอเตอร์ โดยการนำอุปกรณ์ทางด้านดิจิตอบที่เรียกว่า ไมโครคอนโทรลเลอร์ มาทำหน้าที่ในการควบคุม

โครงสร้างของโครงงาน แสดงดังรูปที่ 1.1 จากบล็อกไดอะแกรมแสดงโครงสร้างการควบคุมสเต็ปปิ้งมอเตอร์ จะเห็นว่ามีส่วนประกอบที่สำคัญอยู่ 3 ส่วนด้วยกันคือ ส่วนควบคุมซึ่งใช้ไมโครคอนโทรลเลอร์ ส่วนวงจรขับ และ ส่วนมอเตอร์

หัวใจสำคัญของระบบควบคุมสเต็ปปิ้งมอเตอร์อยู่ที่ทางไมโครคอนโทรลเลอร์ ซึ่งทำหน้าที่เป็นตัวควบคุมการเคลื่อนที่ทุกอย่างของสเต็ปปิ้งมอเตอร์ เช่น จำนวนรอบของการหมุน, ความเร็วรอบ, จังหวะการหมุน โดยการส่งสัญญาณพัลส์ที่มีลักษณเฉพาะ ซึ่งเราจะได้ทราบรายละเอียดในบทที่ 2

ส่วนถัดไปคือ วงจรขับ ทำหน้าที่ขยายสัญญาณลอจิกที่มาจากส่วนควบคุมให้มีขนาดใหญ่เพียงพอที่จะขับมอเตอร์ให้หมุนได้ นอกจากนั้นยังมีความสามารถในการควบคุมปริมาณกระแสที่จ่ายให้ขดลวดของมอเตอร์ ซึ่งมีประโยชน์ในกรณีต้องการให้มอเตอร์หมุนแบบครึ่งสเต็ป หรือ มินิสเต็ป สำหรับวงจรที่ใช้ขับสเต็ปปิ้งมอเตอร์แบบไบโพลาร์ (กระแสไหลผ่านขดลวดของมอเตอร์ในสองทิศทาง ส่วนยูนิโพลาร์กระแสจะไหลในทิศทางเดียว) ยังมีส่วนที่ควบคุมทิศทางการไหลของกระแสอีกด้วย

ส่วนที่สาม คือ ตัวมอเตอร์ ซึ่งจะเกิดการหมุนได้ก็ต่อเมื่อขดลวดที่สเตเตอร์มีกระแสไหลผ่านอย่างเหมาะสม



รูปที่ 1.1 บล็อกไดอะแกรมแสดงระบบควบคุมสเต็ปปิ้งมอเตอร์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาโครงสร้างและการทำงานของสเต็ปปิ้งมอเตอร์

2. เพื่อศึกษาวิธีการนำไมโครคอนโทรลเลอร์ มาใช้ควบคุมการหมุนของสเต็ปปิ้งมอเตอร์

3. เพื่อศึกษาวิธีนำไอซีที่มีจำหน่ายในท้องตลาดมาใช้งานเป็นวงจรขับโดยศึกษาคุณสมบัติต่าง ๆ จากข้อมูลที่บริษัทให้มาแล้วนำมาออกแบบเป็นวงจรที่สามารถใช้งานได้

4. สามารถสร้างระบบควบคุมสเต็ปปิ้งมอเตอร์ที่สมบูรณ์แบบขึ้นมา โดยสามารถนำไปประยุกต์และพัฒนาเพื่อสร้างเป็นเครื่องมือที่เป็นประโยชน์ได้

5. ศึกษาวิธีการเขียนโปรแกรม เพื่อให้สามารถใช้งานไมโครคอนโทรลเลอร์ได้อย่างมีประสิทธิภาพ

6. สามารถออกแบบและสร้างระบบกลไกที่ทำงานร่วมกับสเต็ปปิ้งมอเตอร์เพื่อแสดงให้เห็นการเคลื่อนที่ได้ชัดเจน และเป็นแนวทางในการพัฒนาให้เป็นเครื่องมือที่ใช้งานได้จริง

1.3 ประโยชน์ที่จะได้รับ

1. ได้รับความรู้ทั้งทางด้านฮาร์ดแวร์และซอฟแวร์ของไมโครคอนโทรลเลอร์ตระกูล MCS - 48

2. เข้าใจหลักการทำงานของสเต็ปปิ้งมอเตอร์ และสามารถนำไปประยุกต์ใช้งานได้ตามที่ต้องการ

3. สามารถควบคุมทำงานของสเต็ปปิ้งมอเตอร์จำนวนสามตัวหรือมากกว่านั้นและนำไปใช้กับระบบการเคลื่อนที่ที่ซับซ้อนได้

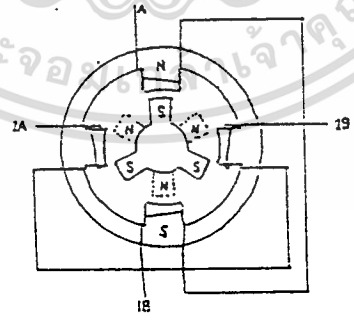
4. สามารถออกแบบกลไกที่ประกอบเข้ากับสเต็ปปิ้งมอเตอร์แล้วทำให้เกิดการควบคุมแบบ 3 แนวแกนได้

บทที่ 2
ทฤษฎีพื้นฐานและหลักการทำงานของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์เป็นเครื่องจักรกลไฟฟ้าประเภทหนึ่งเช่นเดียวกับ ดีซีมอเตอร์ หรือ เอซีมอเตอร์ แต่สเต็ปปิ้งมอเตอร์มีคุณสมบัติพิเศษบางอย่างที่แตกต่างจากมอเตอร์ชนิดอื่น ๆ กล่าวคือ มีการหมุนเป็นแบบสเต็ปตามจังหวะของสัญญาณไฟฟ้าที่ป้อน ลักษณะเช่นนี้เองจึงทำให้เราสามารถควบคุมจำนวนสเต็ปของการหมุนได้ด้วยการควบคุมการป้อนสัญญาณไฟฟ้า หรือสามารถควบคุมการหมุนแบบลูปเปิด (OPEN LOOP) ได้นั่นเอง และถ้านำสเต็ปปิ้งมอเตอร์ ไปต่อใช้งานกับเครื่องมือบางประเภทเช่นพล็อตเตอร์ , เครื่องเจาะอัตโนมัติ ก็จะทำให้การควบคุมตำแหน่งเป็นไปตามที่ต้องการได้อย่างแม่นยำ

2.1 หลักการทำงานของสเต็ปปิ้งมอเตอร์

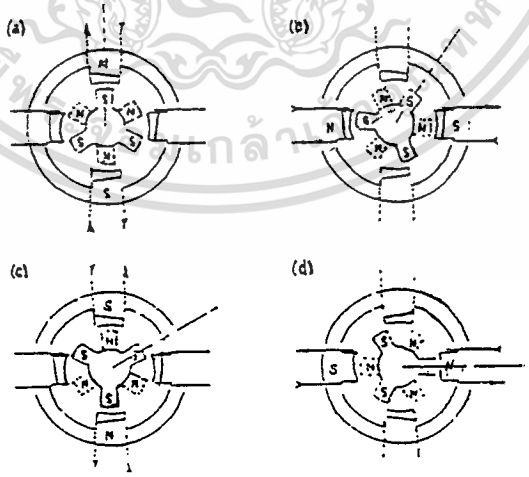
สเต็ปปิ้งมอเตอร์สามารถแบ่งโครงสร้างทางกายภาพออกได้เป็น 2 ส่วน คือ สเตเตอร์ (STATOR) และ โรเตอร์ (ROTOR) ตัวสเตเตอร์เป็นส่วนที่อยู่กับที่ ประกอบด้วยขดลวดทองแดงซึ่งพันอยู่รอบแกนเหล็ก เพื่อสร้างสนามแม่เหล็กเมื่อมีการจ่ายกระแสผ่านขดลวด ส่วนโรเตอร์เป็นส่วนที่เคลื่อนที่ มีลักษณะเป็นแท่งเหล็กทรงกลม และที่ผิวรอบนอกมีลักษณะเป็นซีกฟันซึ่งทำจากแม่เหล็กถาวร ดังรูปที่ 2.1



รูปที่ 2.1 แสดงโครงสร้างของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่มีจำนวนสเต็ปต่อรอบเท่ากับ 12

เมื่อยังไม่มีการจ่ายกระแสให้กับขดลวดของมอเตอร์ซีกฟันอันใดอันหนึ่งของโรเตอร์จะอยู่ในตำแหน่งที่ตรงกันกับซีกฟันอันใดอันหนึ่งของสเตเตอร์ ทั้งนี้เป็นเพราะแม่เหล็กถาวรที่ตัวของโรเตอร์พยายามที่จะทำให้ค่าความต้านทานทางแม่เหล็กไฟฟ้า (RELUCTANCE) มีค่าน้อยที่สุด ซึ่ง ณ. ที่จุดที่ซีกฟันของตัวโรเตอร์และสเตเตอร์ตรงกันนั้นมีค่าความต้านทานทางแม่เหล็กไฟฟ้าน้อยที่สุด ทำให้เกิดเส้นแรงแม่เหล็กไฟฟ้ามากที่สุด และจากรูปที่ 2.1 เส้นแรงแม่เหล็กไฟฟ้าจะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ขึ้นมา 2 คู่ ทั้งที่ตัวสเตเตอร์และตัวโรเตอร์ดังรูป ค่าทอร์ค (TORQUE) ที่ทำให้ตัวโรเตอร์สามารถยึดอยู่ในตำแหน่งดังกล่าวนี้เรียกว่าดีเท็นท์ ทอร์ค (DETENT TORQUE) (หมายความว่า การที่จะทำให้มอเตอร์เคลื่อนที่ในขณะที่ไม่ได้จ่ายกระแสให้กับขดลวดของมอเตอร์จะต้องออกแรงมากกว่าค่าของดีเท็นท์ทอร์ค จึงจะทำให้โรเตอร์เคลื่อนที่ได้) รูปที่ 2.1 นั้นมี 12 ตำแหน่งที่สามารถเกิดดีเท็นท์ทอร์คได้

เมื่อจ่ายกระแสให้กับขดลวดที่อยู่ในสเตเตอร์คู่ใดคู่หนึ่ง ดังรูปที่ 2.2a จะทำให้เกิดขั้วแม่เหล็กเหนือและใต้ที่ซีกฟันของตัวสเตเตอร์ ซึ่งจะดึงดูดซีกฟันของตัวโรเตอร์ที่มีขั้วแม่เหล็กที่มีศักย์ต่างกันที่อยู่ใกล้ที่สุดเข้าไว้ ตำแหน่งนี้เรียกว่า สเตเบิ้ลโพสซิชั่น (STABLE POSITION) ของโรเตอร์ จะมีจำนวนตำแหน่งเท่ากับจำนวนซีกฟันของโรเตอร์ และแรงที่จะทำให้โรเตอร์เปลี่ยนตำแหน่งไปจากตำแหน่งสเตเบิ้ล โพสซิชั่นได้นี้เรียกว่า โฮลดิ้ง ทอร์ค (HOLDING TORQUE)



รูปที่ 2.2 แสดงขั้นตอนการทำงานของเสต็ปปิ้งมอเตอร์แบบ เต็มสเต็ปหนึ่งเฟส

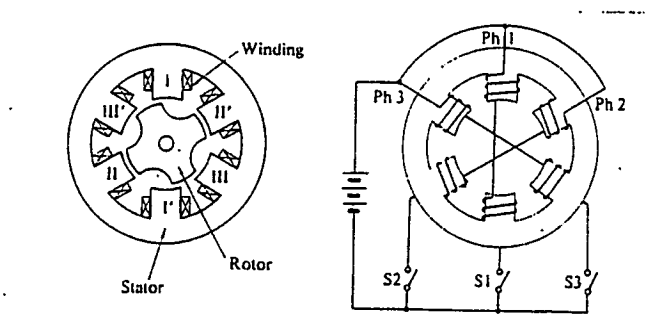
เมื่อสับเปลี่ยนการจ่ายกระแสให้แก่ขดลวด จากขดหนึ่งไปยังอีกขดหนึ่ง เนื่องจากขดลวดวางอยู่ในตำแหน่งที่ต่างกัน 90 องศา ก็จะทำให้ตัวสเตเตอร์ดึงดูดซึกฟันของตัวโรเตอร์อีกซึกหนึ่งที่ใกล้ที่สุดเข้าไว้ ซึ่งจะทำให้ตัวโรเตอร์เคลื่อนที่ไป 1 สเต็ปหรือ 30 องศา ดังรูปที่ 2.2b จากนั้นก็เปลี่ยนไปจ่ายกระแสให้กับขดลวดชุดแรกโดยในคราวนี้เปลี่ยนทิศทางการไหลของกระแสให้ตรงกันข้ามกับครั้งแรก ซึ่งจะทำให้ตัวโรเตอร์เคลื่อนที่ไปอีก 1 สเต็ป (เคลื่อนที่ไป 30 องศา) ดังรูปที่ 2.2c หลังจากนั้นก็ไปจ่ายกระแสให้กับขดลวดชุดที่สองโดยกลับทิศทางของกระแสที่ป้อนให้อีกเช่นกันทำให้โรเตอร์หมุนไป 90 องศา ดังรูปที่ 2.2d และถ้าหากเราป้อนกระแสให้กับมอเตอร์เหมือนที่เราป้อนในครั้งแรกแล้ว ซึกฟันซึกถัดไปของตัวโรเตอร์จะอยู่ในตำแหน่งที่เหมือนกับใน รูปที่ 2.2a อีกครั้งหนึ่ง ถ้าหากเราต้องการเคลื่อนที่หนึ่งรอบ เราต้องทำการกระตุ้นให้มอเตอร์เคลื่อนที่ไปจนครบ 12 สเต็ป และถ้าต้องการให้โรเตอร์หมุนไปอีกทิศทางหนึ่ง ก็จะทำการสลับลำดับในการจ่ายกระแส จากรูปที่ 2.2a, 2.2d , 2.2c, 2.2b ตามลำดับ

2.2 ชนิดของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ สามารถแบ่งออกได้หลายชนิดตามลักษณะโครงสร้างและการใช้งานดังต่อไปนี้

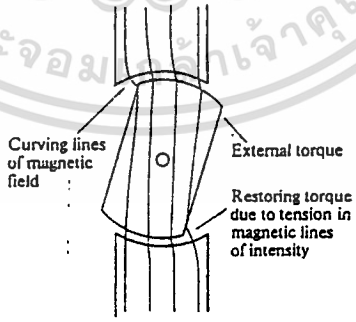
2.2.1 สเต็ปปิ้งมอเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ (VARIABLE RELUCTANCE STEPPING MOTOR)

สเต็ปปิ้งมอเตอร์ชนิดนี้สามารถปรับค่ารีลัคแตนซ์ได้ ซึ่งรูปที่ 2.3 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส โดยที่สเตเตอร์มีฟันทั้งหมด 6 ซี่ ซี่ที่อยู่ตรงข้ามกันหรือทำมุม 180 องศา ซึ่งกันและกันจะเป็นเฟสเดียวกัน ขดลวดที่พันอยู่ที่ฟันของสเตเตอร์ในแต่ละเฟสจะต่ออนุกรมหรือขนานก็ได้ จากรูปที่ 2.3 เป็นการต่อแบบอนุกรม ส่วนโรเตอร์นั้นมีฟัน 4 ซี่ ทั้งโรเตอร์และสเตเตอร์ทำมาจากโลหะซิลิกอน ซึ่งมีสภาพซึมซับทางแม่เหล็กสูงและยอมให้สนามแม่เหล็กจำนวนมากไหลผ่านได้ ฟันของสเตเตอร์ในเฟสเดียวกันจะมีขั้วต่างกันโดยซี่ I, II, III เป็นขั้วเหนือและซี่ I', II', III' เป็นขั้วใต้หลังจากถูกกระตุ้น

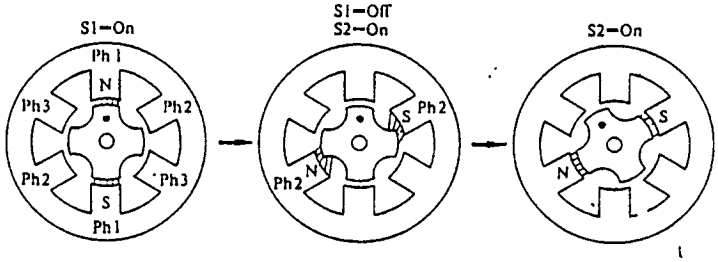


รูปที่ 2.3 แสดงภาพตัดขวางของสเต็ปปิ้งมอเตอร์แบบ 3 เฟส

รูปที่ 2.4 แสดงตำแหน่งสมดุลย์ เมื่อเฟสใดเฟสหนึ่งของสเต็ปปิ้งมอเตอร์ถูกกระตุ้น



รูปที่ 2.5 แสดงแรงภายนอกที่มีผลต่อเส้นแรงแม่เหล็ก



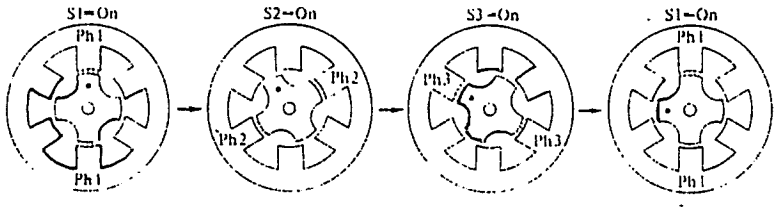
รูปที่ 2.6 แสดงขั้นตอนการเคลื่อนที่ของโรเตอร์เมื่อสเต็ปปิ้งมอเตอร์ถูกกระตุ้น

กระแสที่ไหลในแต่ละเฟสถูกควบคุมโดยสวิทช์ ปิด/เปิด ถ้าเฟส I ถูกกระตุ้นจะมีกระแสไหลและเกิดฟลักซ์แม่เหล็กดังแสดงในรูปที่ 2.4 แกนโรเตอร์จะอยู่ตำแหน่งเดียวกับขั้ว I และ I' ทำให้ทั้งโรเตอร์และสเตเตอร์อยู่ในแนวเดียวกันกรณีนี้จะทำให้ค่ารีลัคแตนซ์มีค่าน้อยที่สุดซึ่งเป็นตำแหน่งที่สมดุลย์ถ้าโรเตอร์ถูกกระทำจากแรงภายนอกจะทำให้เปลี่ยนตำแหน่งดังรูปที่ 2.5 แรงบิดกระทำกับโรเตอร์ในทิศตามเข็มนาฬิกาทำให้ตำแหน่งเปลี่ยนไป มีผลทำให้เส้นแรงแม่เหล็กเคลื่อนที่จากขั้วของโรเตอร์และสเตเตอร์ เมื่อโรเตอร์และสเตเตอร์ไม่ได้อยู่ในแนวเดียวกันแล้วค่ารีลัคแตนซ์จะมีค่ามาก จากนั้นสเต็ปปิ้งมอเตอร์จะทำให้มีค่ารีลัคแตนซ์น้อยที่สุด พอเฟส II ถูกกระตุ้น ดังรูปที่ 2.6 โรเตอร์ถูกแรงภายนอกกระทำให้เคลื่อนไป 30 องศา ในทิศทวนเข็มนาฬิกา จากนั้นก็จะย้ายจากมุมที่เกิดการกระตุ้นกลับไปยังตำแหน่งที่ค่ารีลัคแตนซ์น้อยที่สุด การย้ายจากมุมที่เกิดการกระตุ้นแต่ละครั้งให้กลับไปยังตำแหน่งเดิมเรียกว่า สเต็ป

คุณสมบัติพื้นฐานของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์

1. ช่องว่าง (AIR GAP) ต้องมีขนาดเล็กที่สุด

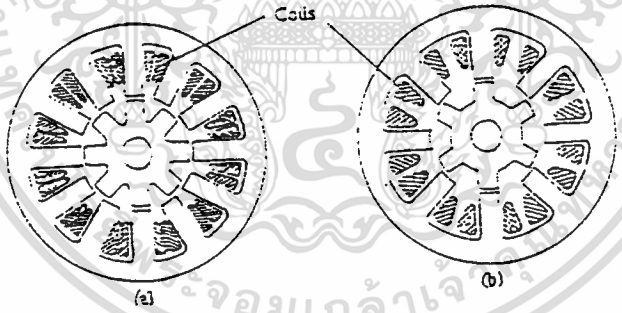
ช่องว่างระหว่างฟันของโรเตอร์และสเตเตอร์ต้องเล็กที่สุดเท่าที่จะเป็นไปได้เพื่อให้ทอร์คที่เกิดขึ้นมีค่ามากและมีความแม่นยำทางตำแหน่งมากขึ้น



รูปที่ 2.7 แสดงขั้นตอนการเคลื่อนของสเต็ปปิ้งมอเตอร์

2. มุมของสเต็ปแคบ

คุณสมบัติอีกประการของสเต็ปคือจะต้องมีมุมของการสเต็ปเล็กที่สุดเท่าที่จะเป็นไปได้ มุมที่แสดงในรูปที่ 2.4 ยังไม่ถือว่าเป็นมุมที่เล็ก แต่รูป 2.8a แสดงมอเตอร์ 3 เฟส ซึ่งมีจำนวนฟันของโรเตอร์และสเตเตอร์เป็น 2 เท่าของรูปที่ 2.4 ส่วนรูปที่ 2.8b แสดงมอเตอร์ 4 เฟส มุมของการสเต็ปของทั้ง 2 โครงสร้างนี้เท่ากับ 15 องศา นอกจากนี้ยังมีบางชนิดที่มีมุมของการสเต็ป 7.5 องศา โดยฟันของโรเตอร์และสเตเตอร์มี 12 และ 16 ชี่ตามลำดับ



รูปที่ 2.8a แสดงมอเตอร์ 3 เฟส

รูปที่ 2.8b แสดงมอเตอร์ 4 เฟส

ความสัมพันธ์ของมุมของการสเต็ป θ_S, มุมเฟส m, จำนวนชี่ฟันของโรเตอร์ Nr, จำนวนสเต็ป S แสดงดังสมการ

$$S = 360/\theta_S = mNr$$

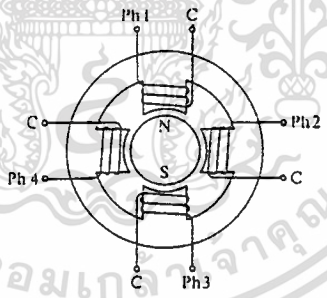
เพื่อที่จะลดขนาดของมุมสเต็ปลงต้องเพิ่มจำนวนขี่ฟันของโรเตอร์โดยที่โครงสร้างของแต่ละขั้วของเฟสใด ๆ สเตเตอร์จะมีหลายขี่ฟัน แต่ก็ไม่ใช่องค์ประกอบโดยตรงที่จะกำหนดมุมของสเต็ปปิ้งมอเตอร์

3. การสร้างสเต็ปปิ้งมอเตอร์ให้มีโครงสร้างหลายสแต็คเพื่อเพิ่มประสิทธิภาพ

โครงสร้างของสเต็ปปิ้งมอเตอร์แบบนี้จะมี 1 เฟส โดยที่โรเตอร์และสเตเตอร์มีขี่ฟันเหมือนกันซึ่งจะเป็นการเพิ่มประสิทธิภาพในด้านทอร์คต่อหน่วยปริมาตรของโรเตอร์

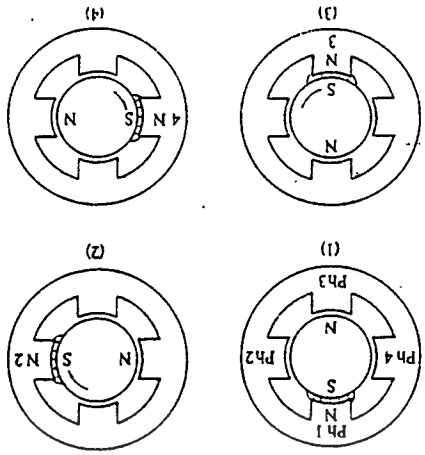
2.2.2 สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

สเต็ปปิ้งมอเตอร์ชนิดนี้ใช้แม่เหล็กถาวรเป็นโรเตอร์ และมีขี่ฟันของสเตเตอร์ล้อมรอบ ขี่ฟันของสเตเตอร์ถูกพันด้วยขดลวดสำหรับสร้างสนามแม่เหล็ก เมื่อต้องการให้สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรมีขนาดมุมของสเต็ปเล็กลงจะต้องเพิ่มขั้วแม่เหล็กของโรเตอร์และจำนวนขี่ฟันของสเตเตอร์ แต่ก็มีขีดจำกัดในการเพิ่มจำนวนขั้วแม่เหล็กของโรเตอร์เนื่องจากการ สร้างแม่เหล็กถาวรสร้างโดยมีขั้วแม่เหล็กหลายขั้วทำได้ยาก



รูปที่ 2.9 แสดงโครงสร้างของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร

ตัวอย่างการทำงานของสเต็ปปิ้งมอเตอร์แบบแม่แหล็กถาวร สมมติว่าสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร ขนาด 4 เฟส มีโรเตอร์เป็นแม่เหล็กถาวรทรงกระบอกและสเตเตอร์ มี 4 ขี่ฟันซึ่งรอบ ๆ พันด้วยขดลวด มีรูปแบบพื้นฐานของการทำงานคือ เมื่อสร้างสัญญาณกระตุ้นตามบลำดับเฟส โรเตอร์จะหมุนไปตามทิศทางของการกระตุ้น ดังแสดงในรูปที่ 2.10



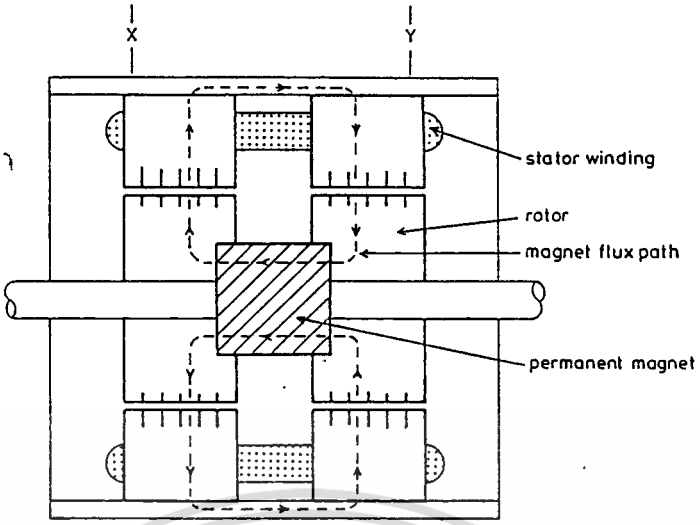
รูปที่ 2.10 แสดงการทำงานของเสต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรขนาด 4 เฟส

ข้อเสียของสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรคือ มีขนาดมุมสเต็ปใหญ่ทำให้ความละเอียดของสเต็ปต่อรอบน้อยเนื่องจากโครงสร้างของโรเตอร์เป็นแม่เหล็กถาวร การสร้างแม่เหล็กถาวรให้มีขั้วหลายขั้วทำได้ยากทำให้ไม่สามารถสร้างสเต็ปขนาดเล็กได้ สเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวรส่วนใหญ่จะมีโครงสร้างขนาดเล็ก ทำให้ค่าทอร์คที่ได้ต่อหน่วยต่อปริมาตรต่ำ ถ้าต้องการปรับปรุงประสิทธิภาพในเรื่องของทอร์ค แม่เหล็กถาวรที่ใช้ต้องทำจากสารแม่เหล็กที่มีสภาพความเป็นแม่เหล็กสูง

2.2.3 สเต็ปปิ้งมอเตอร์แบบไฮบริด์

สเต็ปปิ้งมอเตอร์ชนิดนี้มีแกนโรเตอร์เป็นแม่เหล็กถาวร โดยมีการทำงานร่วมกันของมอเตอร์แบบแม่เหล็กถาวรและมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ได้ ไฮบริดจ์สเต็ปปิ้งมอเตอร์นี้มีโครงสร้างของสเตเตอร์คล้ายกับโครงสร้างของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ แต่ต่างกันที่การต่อขดลวดโดยที่แต่ละเฟสของสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซจะมีขดลวด 2 ขดแต่ละขดมีขั้วต่างกัน แต่ไฮบริดจ์สเต็ปปิ้งมอเตอร์ขดลวดทั้งจะพันอยู่ที่ขั้วเดียวกันเรียกว่าไบโพลาร์ (BIPOLAR) ซึ่งในการกระตุ้นแต่ละครั้งจะให้ขั้วที่แตกต่างกัน

คุณสมบัติที่สำคัญของไฮบริดจ์สเต็ปปิ้งมอเตอร์



รูปที่ 2.11 แสดงโครงสร้างของไฮบริดจ์สเต็ปปิ้งมอเตอร์

โครงสร้างของมอเตอร์จะมีแม่เหล็กถาวรอยู่ตรงกลางระหว่างเฟสทั้งสอง การเหนี่ยวนำสนามแม่เหล็กทำได้โดยใช้สนามแม่เหล็กซึ่งสร้างจากสเตเตอร์ซึ่งเป็นสนามแม่เหล็กแบบเฮทเทอโรโพลาร์ (HETEROPOLAR FIELD) ดังนั้นทอร์คเกิดจากการทำงานร่วมกันของสนามแม่เหล็ก 2 ชนิดคือ สนามแม่เหล็กจากแม่เหล็กถาวรและสนามแม่เหล็กเหนี่ยวนำที่เกิดจากการกระตุ้นของขดลวดแต่ละขด โครงสร้างของซี่ฟันของสเตเตอร์จะใหญ่กว่าซี่ฟันของโรเตอร์เล็กน้อยเพื่อเพิ่มความถูกต้องแม่นยำทางตำแหน่งของการเคลื่อนที่

หลักการทำงานของไฮบริดจ์สเต็ปปิ้งมอเตอร์ที่แตกต่างจากสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์คือแรงบิดที่เกิดจากสนามแม่เหล็กจะไม่ขึ้นอยู่กับกระแสที่ไหลผ่านขดลวดเพียงอย่างเดียวแต่ขึ้นอยู่กับโครงสร้างของซี่ฟันด้วย ซึ่งซี่ฟันถูกออกแบบเพื่อให้ได้โครงสร้างขนาดเล็ก และใช้แม่เหล็กถาวรเป็นแกนกลางเพื่อลดผลของการออสซิลเลททางแมคคานิคส์

ข้อดีของไฮบริดจ์สเต็ปปิ้งมอเตอร์คือ มีขนาดสเต็ปขนาดเล็ก มีความละเอียดของสเต็ปต่อรอบสูง มีค่าทอร์คสูงกว่าสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ แต่สเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ มีแรงเฉื่อยทางแมคคานิคส์น้อยกว่าไฮบริดจ์สเต็ปปิ้งมอเตอร์

นอกจากจากสเต็ปปิ้งมอเตอร์ทั้ง 3 ชนิดที่กล่าวมาแล้วยังมีสเต็ปปิ้งมอเตอร์ชนิดอื่น ๆ ที่ไม่ได้กล่าวถึงอีกเช่น ลิเนียร์สเต็ปปิ้งมอเตอร์ ซึ่งเป็นมอเตอร์ที่ได้รับการออกแบบให้

มีการเคลื่อนที่แบบเป็นเชิงเส้น อิเล็คโตรดไฮดรอลิคสเต็ปปิ้งมอเตอร์ซึ่งเป็นสเต็ปปิ้งมอเตอร์กำลังสูงที่ใช้ในงานอุตสาหกรรม เป็นต้น

2.3 วงจรขับ (DRIVE CIRCUITS)

สัญญาณควบคุมที่ใช้สำหรับควบคุมการทำงานของสเต็ปปิ้งมอเตอร์มักจะเป็นสัญญาณที่สร้างจากวงจรดิจิตอล เช่น จากไมโครคอนโทรลเลอร์ ซึ่งเป็นอุปกรณ์จำพวก TTL แรงดันที่ใช้มีค่าเท่ากับ 5 โวลต์ และสามารถจ่ายกระแสได้ไม่มาก แต่เนื่องจากการทำงานของสเต็ปปิ้งมอเตอร์ ต้องการแรงดันและกระแสที่สูงกว่านั้น ดังนั้นจึงจำเป็นที่จะต้องมีวงจรขับ เพื่อทำหน้าที่จ่ายแรงดันและกระแสที่เพียงพอให้กับตัวสเต็ปปิ้งมอเตอร์ โดยทั่ว ๆ ไป วงจรขับมักจะสร้างจากไบโพลาร์ทรานซิสเตอร์ที่นำมาต่อใช้งานเป็นสวิทช์ ลักษณะของวงจรขับขึ้นอยู่กับชนิดของสเต็ปปิ้งมอเตอร์ที่ใช้ เช่น ถ้าใช้สเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ ซึ่งจะมีอย่างน้อยสามเฟส วงจรขับที่ใช้จะทำงานในลักษณะของสวิทช์ปิด/เปิด ให้กระแสไหลผ่านไปยังขดลวดในทิศทางเดียวเราเรียกวงจรขับแบบนี้ว่า ยูนิโพลาร์ การจ่ายกระแสเพียงทิศทางเดียว แต่ถ้าใช้สเต็ปปิ้งมอเตอร์แบบไฮบริดจ์หรือแบบแม่เหล็กถาวร ซึ่งมักจะมีเพียงสองเฟส จะต้องใช้วงจรขับที่สามารถจ่ายกระแสได้สองทิศทาง เรียกวงจรขับประเภทนี้ว่าไบโพลาร์ ซึ่งประกอบด้วย ทรานซิสเตอร์หลายตัวต่อเป็นวงจรแบบบริดจ์

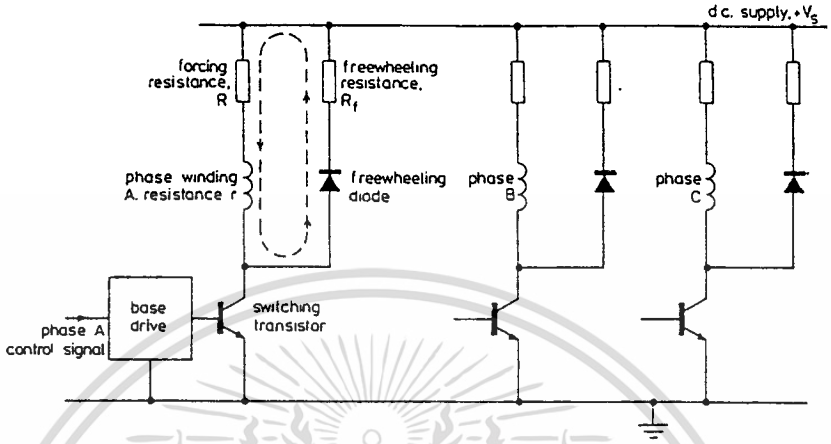
2.3.1 วงจรขับแบบยูนิโพลาร์ (UNIPOLAR DRIVE CIRCUIT)

วงจรขับพื้นฐานที่เหมาะสำหรับสเต็ปปิ้งมอเตอร์แบบวาริเอเบิลรีลัคแตนซ์ดังที่แสดงไว้ในรูปที่ 2.12 ขดลวดแต่ละเฟสจะถูกกระตุ้นโดยวงจรขับแต่ละชุด ซึ่งวงจรขับแต่ละชุดก็จะได้รับสัญญาณควบคุมจากไมโครคอนโทรลเลอร์

กระแสจะไหลผ่านขดลวดแต่ละเฟสเมื่อสวิทช์ซึ่งทรานซิสเตอร์อยู่ในสภาวะอิ่มตัว เนื่องจากกระแสที่ไบอัสทางด้านเบส ในสภาวะเช่นนี้แรงดันดีซีจากแหล่งจ่ายไฟจะไหลผ่านตัวต้านทาน ผ่านขดลวดของสเต็ปปิ้งมอเตอร์และไหลผ่านทรานซิสเตอร์ เนื่องจากแรงดันตกคร่อมทรานซิสเตอร์ในสภาวะอิ่มตัวมีค่าน้อย (ประมาณ 0.1 โวลต์) แรงดันจากแหล่ง

จ่ายไฟทั้งหมดจะทำให้เกิดกระแสไหลผ่านขดลวดโดยมีความสัมพันธ์ กับผลรวมของความต้านของขดขวด (r) และความต้านทานตัวต้านทาน (R) ดังนี้

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



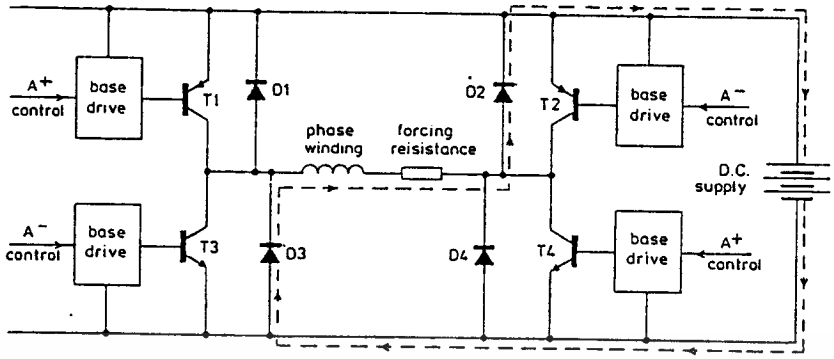
รูปที่ 2.12 วงจรขับแบบยูนิโพลาร์

ตามปกติขดลวดของสเต็ปปิ้งมอเตอร์จะแสดงคุณสมบัติของตัวเหนี่ยวนำ (L) ซึ่งทำให้การตอบสนองต่อกระแสที่ไหลผ่านตัวมันช้า ส่งผลต่อการทำงานของมอเตอร์ที่ความเร็วสูง การใส่ตัวต้านทานอนุกรมเข้าไว้กับขดลวดจะช่วยแก้ปัญหานี้ได้

ความเหนี่ยวนำของขดลวดยังทำให้กระแสไม่หยุดไหลทันทีทันใดที่ทรานซิสเตอร์อยู่ในสถานะปิด ทำให้เกิดแรงดันเหี่ยวนำตกคร่อมคอลเล็คเตอร์และอิมิตเตอร์ ซึ่งเป็นสาเหตุที่ทำให้วงจรขับเสียหาย ปัญหานี้แก้ไขได้โดยการต่อ ฟรีวิลลิ่งไดโอด (FREEWHEELING DIODE) และ ฟรีวิลลิ่งรีซิสเตอร์ (FREEWHEELING RESISTANCE) เพื่อเป็นทางผ่านของกระแสแทน

2.3.2 วงจรขับแบบไบโพลาร์ (BIPOLAR DRIVE CIRCUIT)

วงจรขับชนิดนี้จะต่อทรานซิสเตอร์เป็นแบบบริดจ์ วงจรบริดจ์หนึ่งชุดจะขับมอเตอร์ได้หนึ่งเฟส เหมาะที่ใช้กับสเต็ปปิ้งมอเตอร์แบบแม่เหล็กถาวร หรือแบบไฮบริดจ์ ดังแสดงในรูปที่ 2.13



รูปที่ 2.12 วงจรขับแบบไบโพลาร์

ทรานซิสเตอร์จะผลัดกันทำงานทีละคู่ตามทิศทางของกระแสที่ต้องการ สำหรับการกระตุ้นขดลวดในทิศบวก ทรานซิสเตอร์ T1 และ T4 จะทำงาน ทำให้เกิดทางเดินของกระแสจากแหล่งจ่ายไฟ ไหลผ่านทรานซิสเตอร์ T1 จากนั้นไหลผ่านขดลวดของมอเตอร์ผ่านตัวต้านทานแล้วจึงไหลเข้าทรานซิสเตอร์ T4 กลับเข้าสู่แหล่งจ่ายไฟ ในทางกลับกันในกรณีการจ่ายกระแสในทิศลบ (ทิศตรงกันข้าม) ทรานซิสเตอร์ T2 และ T3 จะทำงาน เพื่อให้กระแสไหลผ่านขดลวดในทิศตรงกันข้าม

จากรูปจะเห็นว่ามีไดโอดสี่ตัว ต่อขนานกับทรานซิสเตอร์แต่ละตัว จุดประสงค์เพื่อให้เกิดเส้นทางไหลของกระแสฟรีวิลลิ่ง (FREEWHEELING CURRENT) บริเวณไดโอด D2 และ D3 จะเป็นทางผ่านของกระแสหลังจากที่ทรานซิสเตอร์ T1 และ T4 หยุดทำงาน ส่วน D1 และ D4 จะมีกระแสไหลผ่านขณะที่ทรานซิสเตอร์ T2 และ T3 หยุดทำงาน กระแสฟรีวิลลิ่งในวงจรขับแบบไบโพลาร์จะสิ้นสุดเร็วกว่าแบบยูนิโพลาร์ เพราะฉะนั้นจึงไม่จำเป็นต้องต่อฟรีวิลลิ่งรีซิสแตนซ์ เหมือนยูนิโพลาร์

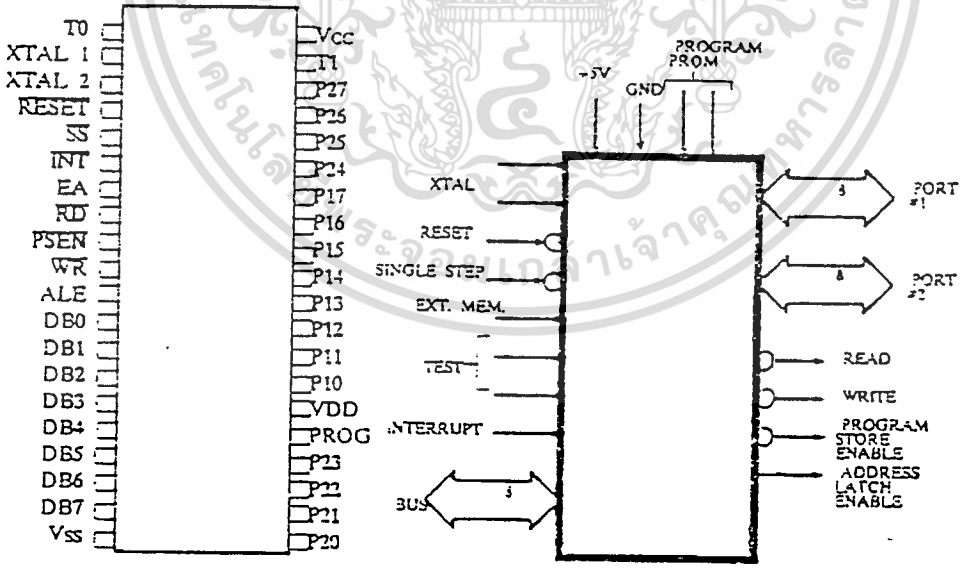
บทที่ 3
โครงสร้างของไมโครคอนโทรลเลอร์ MCS-48 และการใช้งาน

ลักษณะโครงสร้างของ MCS-48 แบ่งเป็น 2 ลักษณะ เช่นเดียวกับไมโครโพรเซสเซอร์ ทั่วไปคือ

1. โครงสร้างภายนอก
2. โครงสร้างภายใน

3.1 โครงสร้างภายนอก

ดูจากรายละเอียดของขาต่าง ๆ ใน MCS-48 ขาทั้งหมดของ MCS-48 ยกเว้นขาไฟเลี้ยงและขาสัญญาณนาฬิกาแล้ว ทุกขาทำหน้าที่เป็นสัญญาณอินพุทและเอาท์พุท โดยมีรายละเอียดของขาต่าง ๆ ดังนี้



รูปที่ 3.1 ลักษณะการจัดขาภายนอกของ MCS-48

ขา 1 T0 สามารถทดสอบสัญญาณเข้า เพื่อใช้ในการถ่ายเทข้อมูลต่าง ๆ ตามข้อกำหนดที่ตั้งไว้ โดยการใช้คำสั่ง JT0 และ JNT0 และสามารถที่จะใช้ T0 เป็นสัญญาณนาฬิกาส่งออก ได้ด้วยคำสั่ง ENT0 CLK TO ยังถูกใช้เป็นขาควบคุมโหมดการโปรแกรมมิ่งและซิ้งค์ได้ด้วย

ขา 2 XTAL 1 สำหรับต่อขาข้างหนึ่งของคริสตอลภายนอกเพื่อใช้ออสซิเลเตอร์ภายใน หรือแหล่งกำเนิดสัญญาณจากภายนอกต่อเข้าขานี้

ขา 3 XTAL 2 สำหรับต่อกับขาอีกข้างหนึ่งของคริสตอล

ขา RESET / เป็นอินพุทสำหรับควบคุมให้โพรเซสเซอร์เริ่มทำงานและยังเป็นสัญญาณสำหรับแลทช์ตำแหน่งของหน่วยความจำโปรแกรมภายในเพื่อประโยชน์ในการอ่านรหัสข้อมูลโปรแกรม

ขา 5 SS/ ขาซิงเกิ้ลสเต็ป จะใช้ร่วมกับ ALE เพื่อทดสอบการทำงานครั้งละหนึ่งคำสั่ง

ขา 6 INT/ เป็นขาอินพุทรับสัญญาณอินเตอร์รัพต์ภายนอก ถ้าต้องการจะให้อินเตอร์รัพต์ได้ จะต้องสั่งด้วยซอฟต์แวร์ด้วยการติดตั้งคำสั่ง EN I สัญญาณอินเตอร์รัพต์ที่เข้ามาจะต้องอยู่ในสถานะต่ำอย่างน้อย 3 วัฏจักรแมชชีน เพื่อให้มั่นใจว่าตัวโพรเซสเซอร์จะรับอินเตอร์รัพต์ได้ ปกติหลังจากรีเซตหรือเปิดเครื่องทุกครั้งจะยังไม่สามารถใช้งานด้วยอินเตอร์รัพต์ได้

ขา 7 EA เมื่อเป็นสถานะสูง จะเป็นการติดต่อรับรหัสข้อมูลจากภายนอกของชิงเกิ้ลชิปตัวนี้ และควบคุมให้ตัวนับโปรแกรม (PROGRAM COUNTER) เฟทช์ รหัสจากโปรแกรมภายนอกทั้งหมด เพื่อใช้ประโยชน์ในการแก้ไข และทำอีมูเลตหรือทดสอบกับเครื่องต้นแบบ

ขา 8 RD/ จะเป็นสัญญาณควบคุมการรับหรืออ่านข้อมูลจากภายนอก ผ่านเข้าขาข้อมูลระหว่างการอ่าน และสัญญาณนี้จะสโตรบพัลส์ต่ำทุกครั้งที่มีการอ่านข้อมูลไม่ว่าจะอ่านรหัสจากภายนอกหรือภายใน

ขา 9 PSEN/ เป็นสัญญาณ PROGRAM STORE ENABLE สัญญาณควบคุมการอ่านรหัสคำสั่งทุกคำจากหน่วยความจำภายนอก และสัญญาณนี้จะเกิดแอ็คทีฟต่ำให้มีการเฟทช์ข้อมูลโปรแกรมจากภายนอกได้เท่านั้น

ขา 10 WR/ เป็นสัญญาณควบคุมการส่ง หรือเขียนข้อมูลออกภายนอกผ่านบัสข้อมูลและสัญญาณนี้จะสโตรบพัลส์ต่ำออกทุกครั้งที่มีการเขียนข้อมูล

ขา 11 ALE เป็นสัญญาณ ADDRESS LATCH ENABLE สัญญาณนี้จะเกิดทุกวัฏจักรของการเฟทช์ในแต่ละครั้ง และใช้ประโยชน์ในการส่งสัญญาณนาฬิกาเอาท์พุทด้วยการใช้สัญญาณขอบขาลงของสัญญาณนี้ เป็นการสโตรบแลทช์แอดเดรสเพื่อถอดรหัสข้อมูลจากหน่วยความจำข้อมูลและโปรแกรม

ขา 12-19 D0-D7 เป็น BIDIRECTIONAL PORT ทำหน้าที่ต่าง ๆ ดังนี้

1. เป็นที่ส่งผ่านข้อมูลเข้าและออก ในขณะที่ซิงค์ด้วยสัญญาณ RD หรือ WR สโตรบ และสามารถทำหน้าที่เป็นพอร์ตสำหรับแลทช์ข้อมูล

2. ใช้เป็นตัวส่งข้อมูลตัวนับโปรแกรม (PC) บิตอันดับต่ำระหว่างการเฟทช์โปรแกรมจากภายนอก และรับรหัสคำสั่งภายใต้การส่งสัญญาณควบคุม PSEN

ขา 20 V_{SS} เป็นสายดินของวงจร

ขา 21-24 P24-P27 สี่บิตแรกของ QUASI-BIDIRECTIONAL พอร์ต 2 ใช้เป็นตัวส่งค่า

ขา 35-38 P25-P27 อันดับสูงของแอดเดรสตัวนับโปรแกรม และใช้ในการติดต่อกับหน่วยความจำภายนอก และใช้เป็น 4 บิตไอโอบัสสำรหับการขยายพอร์ตไอโอของ 8243 และอีกสี่บิตอันดับสูงใช้เป็นพอร์ตไอโอธรรมดา

ขา 25 PROG ใช้เป็นสัญญาณเอาท์พุทสโตรบสำหรับควบคุมการใช้ร่วมกับการขยายพอร์ตไอโอของ 8243 และใช้เป็นที่ป้อนโปรแกรมพัลส์ขนาด +18 V เข้าขานี้สำหรับการเขียนโปรแกรมเข้าตัวไมโครคอนโทรลเลอร์เบอร์ 8748 / 8749

ขา 26 V_{DD} ปกติตัว MCS-48 นี้จะทำงานได้ด้วยการจ่ายแรงดัน 5 โวลต์จ่ายเข้าที่ขานี้และถ้าแหล่งจ่ายไฟตกจะทำให้ MCS-48 เข้าโหมดการใช้แหล่งจ่ายไฟสำรองเข้าที่ขานี้ และการโปรแกรมรหัสเข้าตัวไมโครคอนโทรเลอร์เบอร์ 8748 / 8749 จะใช้การจ่ายแรงดันให้ 21 โวลต์ที่ขานี้เช่นกัน

ขา 27-34 P10-P17 เป็นพอร์ตขนาด 8 บิตแบบ QUASI-BIDIRECTIONAL พอร์ต 2 โดยมี 50 KOHMS พูลอัพภายในและขาเอาท์พุทจะเป็นแบบสามสถานะ

ขา 39 T1 เป็นสัญญาณที่ถูกทดสอบเข้าในตัวไมโครคอนโทรเลอร์ ด้วยคำสั่ง JT1,JNT1 สามารถใช้ขา T1 ให้เป็นตัวนับเหตุการณ์ที่เกิดขึ้นจากภายนอก ส่งเข้ามาด้วยการใช้คำสั่ง STRT CNT (START COUNTER)

ขา 40 V_{CC} จ่ายแรงดันไฟ 5 โวลต์เข้าที่ขานี้ ในขณะที่ทำงานปกติ และขณะที่อัดโปรแกรมรหัสเข้าตัว 7848 / 7849

3.2 โครงสร้างสถาปัตยกรรมภายในของ MCS-48

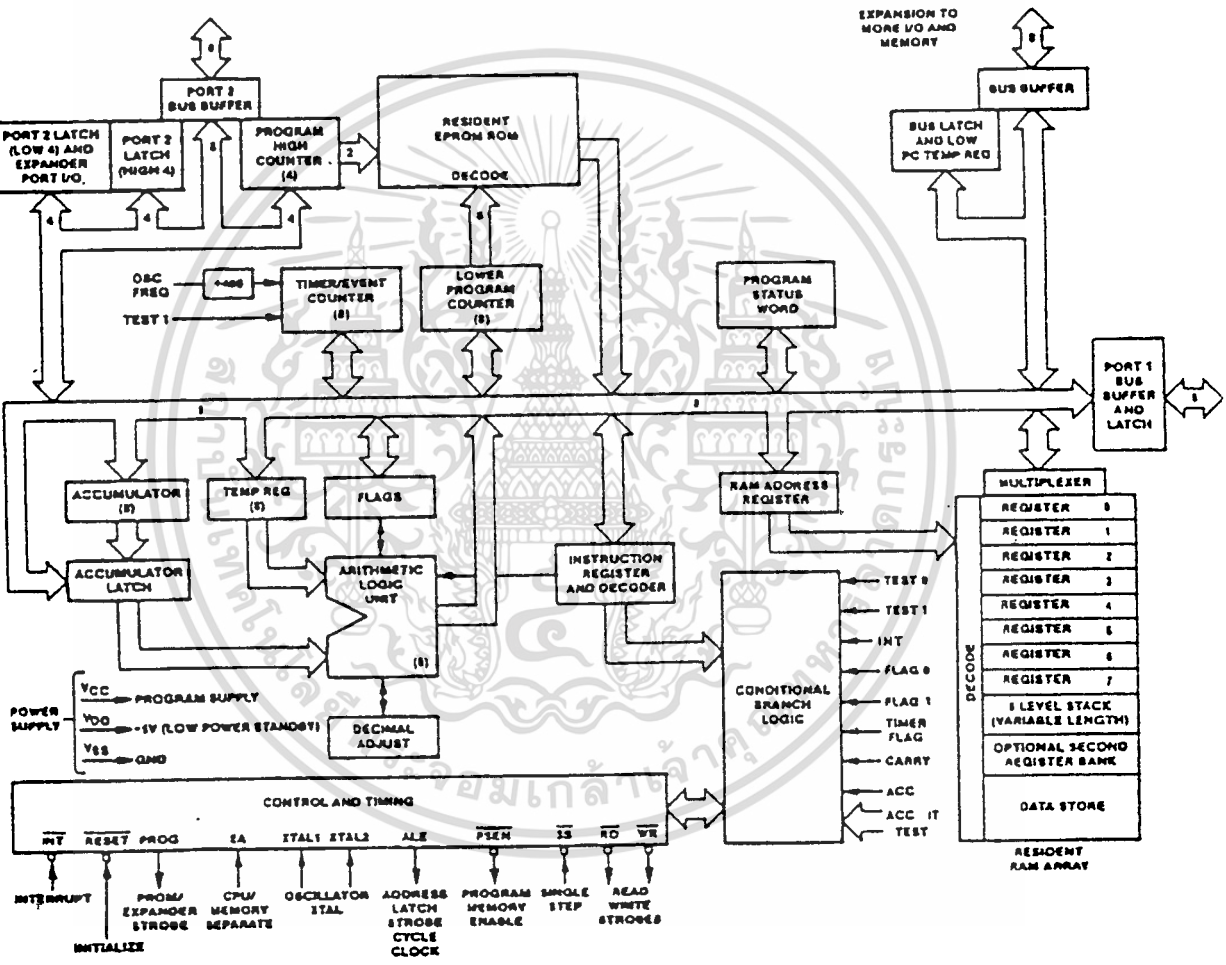
โครงสร้างภายในของ MCS-48 เหมือนกับโครงสร้างของไมโครคอมพิวเตอร์ทั่วไป จะประกอบด้วยบัสที่ทำหน้าที่เชื่อมต่อกับอุปกรณ์อื่น ๆ โดยมีส่วนประกอบภายในต่าง ๆ ดังรูปที่ 1.2 และรายละเอียดของแต่ละส่วนจะเป็นดังนี้

3.2.1 หน่วยคณิตศาสตร์ (ALU)

ALU จะรับหรือทำงานร่วมกับข้อมูลจากแหล่งข้อมูล 1 หรือ 2 แหล่งเพื่อไปทำการประมวลผลทางคณิตศาสตร์ภายใต้การควบคุมของตัวถอดรหัสคำสั่ง ALU มีความสามารถที่จะทำงานตามหน้าที่ต่าง ๆ ดังต่อไปนี้

1. บวกพร้อมตัวทดหรือปราศจากตัวทด (ADD WITH OR WITHOUT CARRY)
2. ทำงานทางตรรก (AND,OR,EXCLUSIVE OR)
3. การเพิ่มหรือลดค่าหนึ่งค่า (INCREMENT/DECREMENT)
4. การแปลงกลับค่าบิต (BIT COMPLEMENT)
5. วนบิตทางซ้ายหรือขวา (LEFT OR RIGHT ROTATE)
6. การสลับค่านิบเบิ้ล (SWAP NIBBLE)
7. การปรับค่าเป็น BCD DECIMAL (BCD DECIMAL ADJUST)

ถ้าการทำงานของ ALU ให้ผลลัพธ์มากกว่า 8 บิต จะเกิดตัวทดจากบิตหลักสูงสุด (MOST SIGNIFICANT BIT MSB) เข้าตัวแฟลกทด (CARRY FLAG) ภายในรีจิสเตอร์ PSW (PROGRAM STATUS WORD) จะถูกเซ็ต



รูปที่ 3.2 โครงสร้างสถาปัตยกรรมภายในของ MCS-48

3.2.2 แอกคิวมูเลเตอร์

จะเป็นตัวรีจิสเตอร์ที่สำคัญที่สุดของ MCS-48 ในตัวโพรเซสเซอร์ จะทำหน้าที่เป็นตัวรับข้อมูลจากแหล่งข้อมูลที่ส่งเข้ามา และเป็นตัวส่งผลลัพธ์ให้กับรีจิสเตอร์ตัวรับ (DESTINATION REGISTER) ในการทำงานของ ALU การติดต่อกับอุปกรณ์ภายนอก และติดต่อกับหน่วยความจำ จะต้องผ่านแอกคิวมูเลเตอร์เสมอ

3.2.3 แฟลกตัวทด (CARRY FLAG)

เป็นตัวบอกสถานะการทำงานของ MCS-48 ว่าการทำงานเกินจำนวนบิตที่มีอยู่หรือไม่ถ้าเกินแฟลกตัวทดนี้จะถูกเซ็ต แฟลกตัวนี้จะเป็นบิตหนึ่งของรีจิสเตอร์ PSW

3.2.4 ตัวถอดรหัส (INSTRUCTION REGISTER AND DECODER)

รหัสการทำงาน (OPERATION CODE = OP CODE) ของแต่ละคำสั่ง จะถูกเฟทช์เข้ามาเก็บอยู่ในรีจิสเตอร์ถอดรหัสนี้ จากนั้นจะถูกถอดรหัสเป็นสัญญาณควบคุมต่างๆ ตามรหัสของคำสั่งนั้น ๆ ต่อไป

3.3 หน่วยความจำ (MEMORY)

หน่วยความจำของ MCS-48 นั้นแบ่งการทำงานออกได้เป็น 2 ชนิด คือ

หน่วยความจำโปรแกรม (PROGRAM MEMORY) หน่วยความจำนี้จะใช้สำหรับเก็บชุดคำสั่งการทำงานของ MCS-48 ที่เขียนขึ้น ขนาดหน่วยความจำสูงสุดของส่วนนี้ คือ 4 กิโลไบต์ โดยส่วนหนึ่งจะอยู่ภายในชิปของไอซีหรือเป็นการต่อเพิ่มจากภายนอกก็ได้ ในกรณีที่หน่วยความจำโปรแกรมภายในชิป จะมีขนาดความจุและชนิดต่าง ๆ กันแล้วแต่เบอร์ในตระกูล MCS-48 ดังตารางที่ 3.1 ตำแหน่งที่สำคัญของหน่วยความจำส่วนนี้ คือ

เบอร์	หน่วยความจำภายใน (โปรแกรมรอม)	หน่วยความจำภายใน (ข้อมูลแรม)	แรมสำรอง
8050AH	4K X 8	256 X 8	มี
8049AH	2K X 8	128 X 8	มี
8048AH	1K X 8	64 X 8	มี
8040AHL	ไม่มี	256 X 8	มี
8039AHL	ไม่มี	128 X 8	มี
8035AHL	ไม่มี	64 X 8	มี
8749H	2K X 8 (EPROM)	128 X 8	ไม่มี

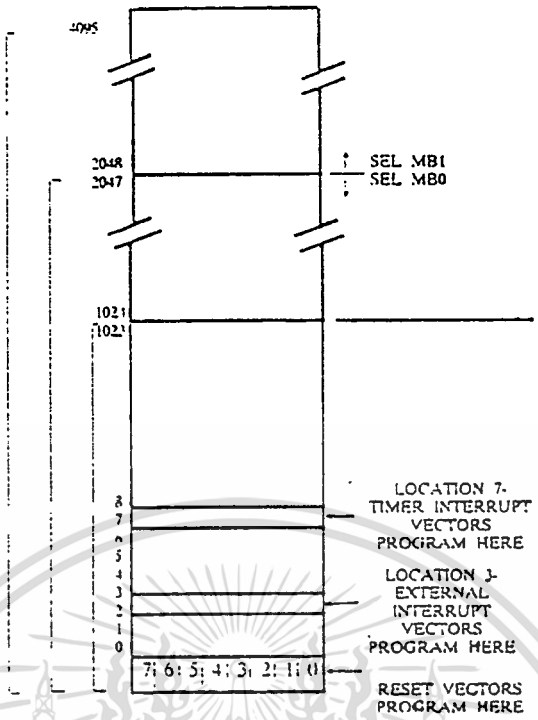
ตารางที่ 3.1 รายละเอียดต่าง ๆ ของตระกูล MCS-48

(แอดเดรส 000) เมื่อถูกรีเซต MCS-48 จะทำการเฟทช์คำสั่งตัวแรกที่ตำแหน่งนี้

(แอดเดรส003) MCS-48 จะเฟทช์คำสั่งแรกที่ตำแหน่งนี้ เมื่อมีสัญญาณอินเตอร์รัพต์เข้ามา ถ้าอินเตอร์รัพต์ถูกติดตั้งให้อีนาเบิ้ลจะเป็นเหตุให้โปรแกรมกระโดดไปทำงานโปรแกรมบริการอินเตอร์รัพต์ที่ตำแหน่งนี้

(แอดเดรส 007) MCS-48 จะเฟทช์คำสั่งแรกที่ตำแหน่งนี้ เมื่อมีการอินเตอร์รัพต์ทำงานแบบจับเวลา/ตัวนับ มีผลจากการที่ตัวจับเวลา/ตัวนับ เวลาเกิด OVERFLOW ตำแหน่งแรกของทั้ง 3 ตำแหน่งดังกล่าว โดยปกติแล้วจะเขียนด้วยคำสั่ง JUMP ไปยังโปรแกรมบริการต่าง ๆ ที่เขียนไว้ ดังนั้นคำสั่งกระโดดตัวแรกจะถูกใช้งานหลังจากที่โปรแกรมเริ่มทำงาน (Initialize) ถูกเก็บเข้าที่แอดเดรส 000 ในการทำงานเริ่มต้นคำสั่งของการกระโดดไปใช้บริการอินเตอร์รัพต์จากภายนอกเริ่มแรกจะถูกเก็บเข้าที่แอดเดรส 003 และคำสั่งของการกระโดดไปใช้บริการโปรแกรมการใช้อินเตอร์รัพต์ตัวจับเวลา/ตัวนับ เริ่มแรกจะถูกเฟทช์จากแอดเดรส 007

หน่วยความจำโปรแกรมสามารถที่จะใช้เป็นตัวเก็บข้อมูลคงที่ และโปรแกรมคำสั่งด้วยการใช้คำสั่ง MOVE และ MOVP3 ช่วยให้การอ่านตารางข้อมูลคงที่ทำได้ง่ายขึ้น



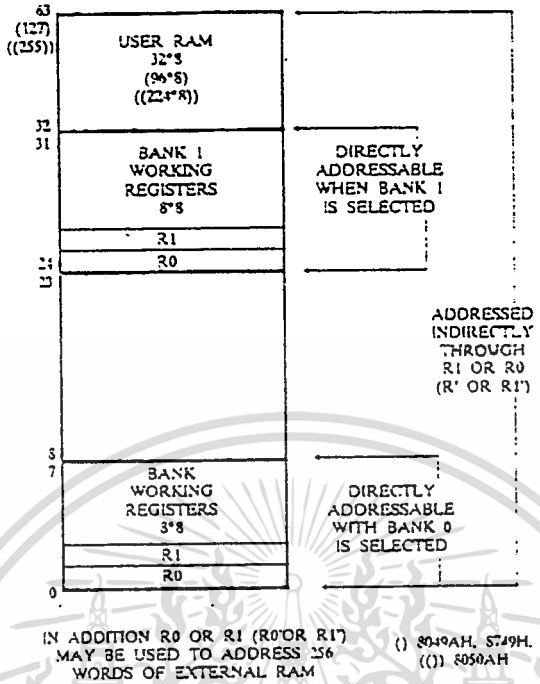
รูปที่ 3.3 PROGRAM MEMORY MAP

3.3.1 หน่วยความจำข้อมูล

หน่วยความจำส่วนนี้จะเป็นชนิดแรมที่อยู่ภายในชิปของ MCS-48 โดยมีขนาด 64,128 และ 256 ไบต์ ขึ้นอยู่กับเบอร์ของชิปตระกูล MCS-48 ดังตารางที่ 1.1 ภายในส่วนของหน่วยความจำส่วนนี้จะทำหน้าที่เป็นรีจิสเตอร์ใช้งาน 16 ตัว โดยแบ่งเป็น 2 ชุด ๆ ละ 8 ตัว (หรือแบงค์) ซึ่งชุดแรกอยู่ที่เลขที่อยู่ 0-7 เป็นแบงค์ 0 และชุดที่ 2 อยู่ที่เลขที่ 24-31 เป็นแบงค์ 1 ในขณะที่ ทำงานอยู่นั้น รีจิสเตอร์จะถูกใช้งานเพียงชุดเดียว โดยสามารถจะเลือกชุดของรีจิสเตอร์จากคำสั่ง SEL RB (Register Bank Switch) เป็นลักษณะที่สามารถจะเลือกเลขที่อยู่โดยตรง (Direct Address) ได้ จากเลขที่อยู่ 0-7 ในแบงค์ 0 และแบงค์ 1 อาจใช้งานเป็นรีจิสเตอร์ขยายจากแบงค์แรก หรือใช้เป็นรีจิสเตอร์สำรองสำหรับการใช้งานระหว่างการใช้งานโปรแกรมย่อย และในโปรแกรมหลักยังคงใช้ แบงค์แรกในการเก็บข้อมูลโดยทันทีด้วยแบงค์สวิตช์ ขณะเดียวกันถ้าแบงค์ 1 ไม่ถูกนำมาใช้ เลขที่อยู่ 24-31 ยังคงใช้เป็นแอด-เดรสด้วยการเก็บข้อมูลแบบแรมได้

นอกจากทำหน้าที่เป็นรีจิสเตอร์แล้วที่หน่วยความจำข้อมูลเลขที่อยู่ 8-23 ก็จะเป็นเนื้อที่สแตกด้วยการเก็บตัวนับโปรแกรมเป็นคู่ โดยใช้งานครั้งละ 2 ตำแหน่งหน่วยความจำ ตำแหน่งเหล่านี้จะถูกแอดเดรสด้วยตัวชี้แรมของ R0 กับ R1 การใช้พื้นที่นี้เหมาะสำหรับการใช้โปรแกรมย่อยหลายโปรแกรมเชื่อมโยงกัน แต่ต้องมากกว่า 8 โปรแกรมย่อยเพราเก็บได้เพียง 8 คู่ ทำนองเดียวกัน พื้นที่ของสแตกนี้ ถ้าไม่ถูกใช้เป็นรีจิสเตอร์สแตก ก็สามารถที่จะใช้เป็นที่เก็บข้อมูลแรมทั่วไปได้ แต่ถ้าใช้เป็นสแตกสำหรับโปรแกรมย่อยแล้ว ก็ไม่สามารถที่จะใช้เป็นที่เก็บข้อมูลแบบแรม ในเวลาเดียวกันได้

ในกรณีที่ผู้ใช้ต้องการใช้เนื้อที่แรมมากกว่าที่มีอยู่ในตัว MCS-48 ก็สามารถเพิ่มแรมภายนอกได้ โดยใช้คำสั่ง MOVX @R,A เมื่อใช้ R เป็นรีจิสเตจอร์ 0 หรือ รีจิสเตอร์ 1 ทำหน้าที่เป็นตัวชี้ในการติดต่อกับแรมที่เพิ่มขึ้นภายใน 256 ไบต์ หน่วยความจำส่วนนี้โดยปกติหรับเก็บข้อมูลชั่วคราว เมื่อมีการอ่านข้อมูลเข้าไปใน MCS-48 จะมองข้อมูลส่วนนี้ผิดกับข้อมูลที่อยู่ในส่วนของหน่วยความจำโปรแกรมที่กล่าวมาแล้ว ขณะเดียวกันการใช้รีจิสเตอร์ตัวชี้ R0 และ R1 ของแรม สามารถที่จะเพิ่มรีจิสเตอร์ตัวชี้ได้อย่างง่าย ๆ ด้วยการใช้คำสั่ง BANK SWITCH เมื่อต้องการใช้ตำแหน่งข้อมูลกับแรมถึง 4 ตำแหน่งในการเข้าถึงข้อมูลแต่ละครั้ง



รูปที่ 3.4 DATA MEMORY MAP

3.4 ส่วนติดต่ออุปกรณ์ภายนอก อินพุท/เอาต์พุท

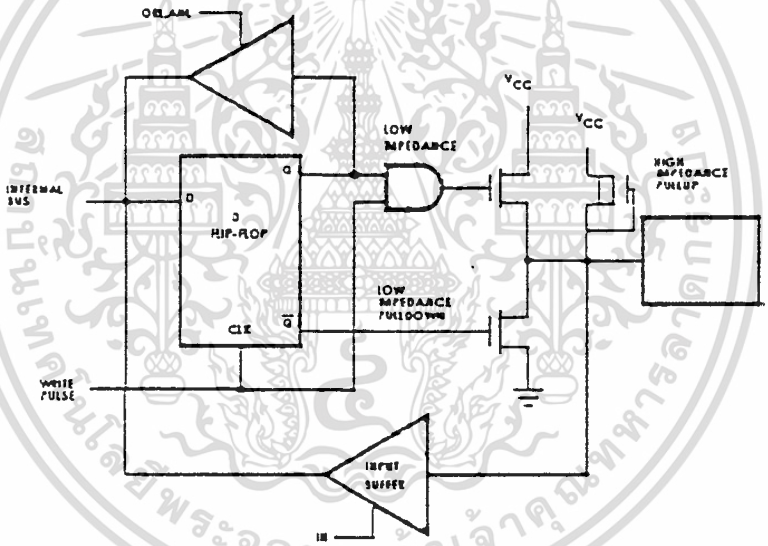
ในส่วนการติดต่ออุปกรณ์ภายนอก MCS-48 สามารถแบ่งออกได้ 2 ลักษณะคือ ทำหน้าที่เป็นพอร์ตกับบัส โดยจะแยกกล่าวดังรายละเอียดต่อไปนี้

3.4.1 พอร์ต (Port)

ภายในตัว MCS-48 จะมีพอร์ตคล้ายกัน 2 พอร์ต คือพอร์ต 1 และพอร์ต 2 ที่มีขนาด 8 บิต พอร์ตทั้งสองเป็นพอร์ตชนิด QUASI-BIDIRECTIONAL เนื่องจากโครงสร้างวงจรของพอร์ตแต่ละเส้นทำงานเป็นอินพุตและเอาต์พุท ด้วยลักษณะข้อมูลเอาต์พุทจะเป็นสแตติกแลทช์ (STATIC LATCH) และข้อมูลจะยังคงอยู่จนกว่าจะมีการเขียนข้อมูลใหม่ จากรูปที่ 3.5 แสดงวงจรภายในของขาแต่ละขาของพอร์ตทั้ง 2 จะเห็นได้ว่า ตัวพูลอัพ

High Impedance ที่ต่อกับ Vcc จะทำหน้าที่ยกระดับแรงดันและกระแสให้เพียงพอกับระดับลอจิกของทีทีแอลที่จะมาต่อพ่วง จากวงจรจะเห็นว่าในกรณีของการอินพุตนั้น สามารถจะรับการเปลี่ยนระดับลอจิกจากระดับสูงสู่ระดับต่ำเท่านั้น ดังนั้นในการใช้ขาใดขาหนึ่งของพอร์ตเป็นอินพุตสัญญาณนั้นต้องทำการเซตให้ขานั้นอยู่ในระดับลอจิก 1 ก่อนเสมอ ในการรีเซตเริ่มแรกที่สำคัญในการอ่านหรือเขียนจากพอร์ต จะใช้คำสั่ง ORL และ ANL เมื่อทำงานตัวไมโครจะอ่านข้อมูลจากพอร์ตด้วยคำสั่งอ่านข้อมูลและตามด้วยการเขียนข้อมูลกลับไปที่พอร์ต

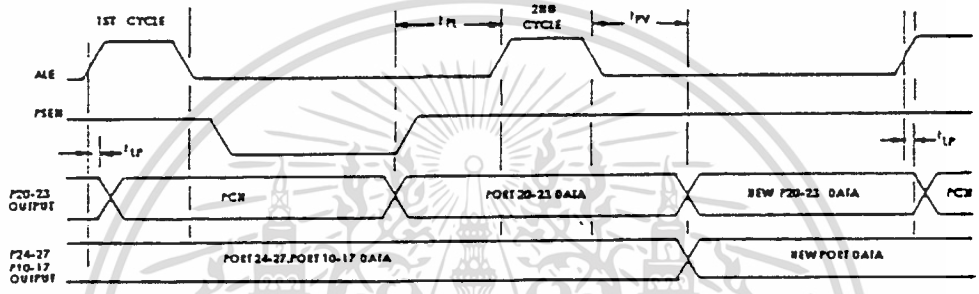
การเขียนข้อมูลมีความจำเป็นต้องมีเพื่อให้อีนาเบิ้ลพูลอัพอิมพิแดนซ์ต่ำเกิดขึ้นชั่วขณะหนึ่งแม้ข้อมูลจะเปลี่ยนแปลงจาก 1 ก็ตาม การกำหนดทำเช่นนี้ก็เพื่อที่จะติดตั้ง (Config) ใช้พอร์ตเป็นทั้งอินพุทและเอาท์พุทในพอร์ตเดียวกันได้



รูปที่ 3.5 โครงสร้างพอร์ต QUASI-BIDIRECTIONAL

ในกรณีที่ใช้หน่วยความภายนอก 4บิตแรกของพอร์ต 2 จะทำหน้าที่ในการกำหนดเลขที่อยู่ 4 บิตอันดับสูงของหน่วยความจำโปรแกรมเพื่อทำการอ่านคำสั่งการทำงาน ในกรณีนี้ถ้าต้องการใช้ 4 บิตแรกของพอร์ต 2 ทำหน้าที่สำหรับติดต่ออุปกรณ์ภายนอกอีก จะเกิดปัญหาที่ว่า MCS-48 ใช้พอร์ต 2 เป็นตัวชี้แอดเดรส 4 บิตสุดท้ายอยู่แล้ว สามารถแก้ปัญหานี้ได้โดยการใช้การแลทช์เข้าช่วย เพราะสัญญาณที่ออกมาที่ 4 บิตแรกของพอร์ต 2

จะเป็นลักษณะมัลติเพลกซ์ระหว่างข้อมูลที่ติดต่อกับอุปกรณ์ภายนอกกับเลขที่อยู่แอดเดรสที่ส่งออกมา โดยรูปแบบของการติดต่ออุปกรณ์ภายนอกจะออกมาในขณะช่วงขึ้นของสัญญาณ ALE แต่รูปแบบของแอดเดรสจะปรากฏเมื่อช่วงลงของสัญญาณ ALE ดังนั้นข้อมูลสำหรับติดต่ออุปกรณ์ภายนอกกับแอดเดรสจึงแยกออกมาได้ ดังรูปที่ 3.6 เป็นแผนภูมิจังหวะเวลาของการใช้พอร์ต 1 และ 2 เป็นพอร์ตเอาท์พุต ร่วมกับการส่งบิตแอดเดรสอันดับสูงของตัวนับโปรแกรม (PCH)

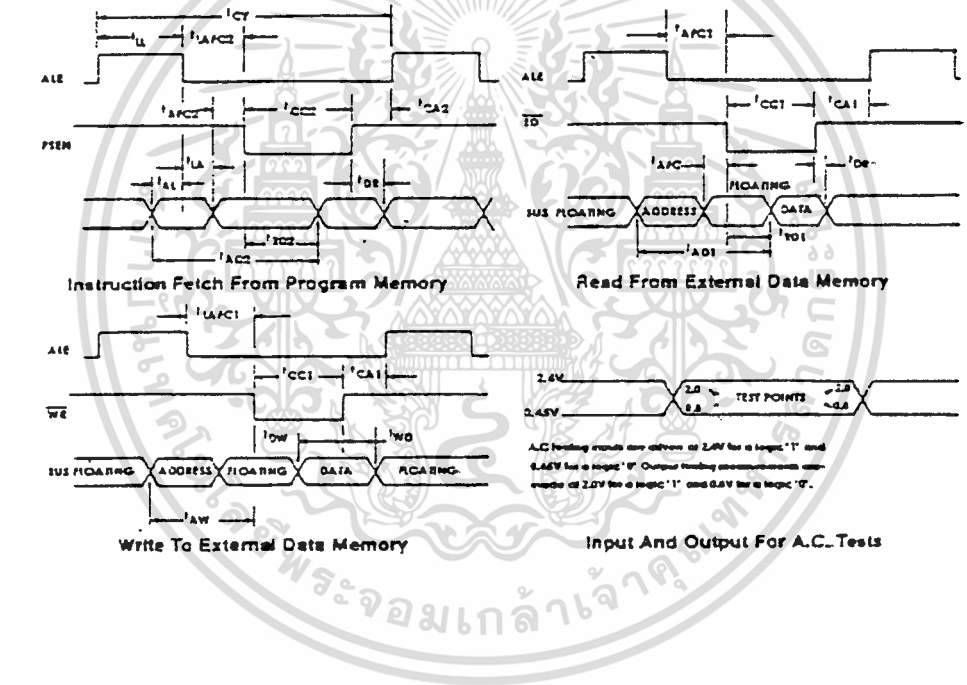


รูปที่ 3.6 แผนภูมิจังหวะเวลาของการใช้พอร์ต 1 และ 2 เป็นพอร์ตเอาต์พุท

3.4.2 บัส (BUS)

ในส่วนนี้พอร์ต 0 จะทำหน้าที่ 2 อย่างขึ้นอยู่กับกรณีที่ใช้คือในกรณีที่ใช้หน่วยความจำโปรแกรมภายในบัสจะทำหน้าที่เช่นเดียวกับพอร์ตทั่วไปที่เป็นพอร์ตชนิด BIDIRECTIONAL PORT ซึ่งจะต่างกับพอร์ต 1 และ พอร์ต 2 ที่กล่าวมาแล้ว โดยจะทำหน้าที่เป็นพอร์ตเอาต์พุตที่แลทช์ข้อมูลไว้ด้วยคำสั่ง OUTL และยังเป็นอินพุตพอร์ตที่ไปแลทช์ข้อมูลไว้ที่พอร์ตด้วยคำสั่ง INS และทั้งสองคำสั่งนี้ จะสร้างพัลส์สโตรบที่ขา RD หรือ WR และที่จังหวะขอบขาลงของ RD หรือ WR จะเป็นช่วงจังหวะข้อมูลที่ปรากฏที่พอร์ตนี้ แต่พอร์ตนี้จะทำหน้าที่ผสมเป็นทั้งอินพุต์และเอาต์พุตพร้อมกันไม่ได้ แต่ในกรณีที่ใช้หน่วยความจำโปรแกรมภายนอกแล้ว พอร์ตนี้จะทำหน้าที่เป็นบัสในการติดต่อกับหน่วยความจำโปรแกรมภายนอก ในขณะที่ทำการอ่านคำสั่ง จะให้แอดเดรสและอ่านคำสั่งเข้าที่บัสนี้ด้วยวิธีมัลติเพลกซ์เช่นเดียวกัน

พอร์ต 2 ที่กล่าวมาแล้ว ในการมัลติเพลกซ์นั้น MCS-48 จะส่งสัญญาณแอดเดรสออกมาที่พอร์ตนี้ก่อน ก็จะถูกแลทช์ไว้ที่อุปกรณ์การแลทช์แอดเดรสภายนอก แล้วจึงทำการอ่านคำสั่งที่ได้จากหน่วยความจำโปรแกรมภายนอก ในขณะที่ไม่มีการอ่านและเขียนบัสจะอยู่ในสถานะอิมพีแดนซ์สูง ดังรูปที่ 3.7 เป็นการแสดงช่วงจังหวะของการใช้พอร์ตบัสทำงานทั้งแบบการเฟทช์โปรแกรม การอ่านและเขียนข้อมูลจากภายนอก ซึ่งจะต้องส่งค่าแอดเดรสออกมาก่อน แล้วจึงตามด้วยรหัสคำสั่งหรือข้อมูล โดยใช้ขา ALE,PSEN,WR และ RD เป็นตัวควบคุมบอกจังหวะของข้อมูลในช่วงจังหวะใด ๆ ที่เกิดขึ้น



รูปที่ 3.7 แผนภูมิจังหวะเวลาของการเฟทช์โปรแกรม การอ่านและการเขียนข้อมูลจากภายนอก

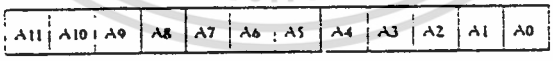
3.5 สัญญาณการตรวจสอบและสัญญาณอินเตอร์รัพต์ (TEST and INT INPUT)

ในตระกูล MCS-48 ได้จัดขาไว้ 3 ขาสำหรับเป็นอินพุต และไว้ใช้ทดสอบเงื่อนไข เพื่อทำงานร่วมกับคำสั่ง JUMP ซึ่งได้แก่ T0,T1,INT หลังจากทำการทดสอบเงื่อนไขของ

สัญญาณทั้งสามแล้ว การทำงานของ MCS-48 สามารถข้ามไปทำงานตามคำสั่งในโปรแกรมย่อย ที่เขียนไว้เพื่อทำงานตามเงื่อนไขที่ถูกต้อง โดยไม่จำเป็นต้องผ่านแอคคิวมูเลเตอร์ นอกจากนี้แล้วสัญญาณทั้งสามนี้ ยังสามารถทำหน้าที่อย่างอื่นได้อีก ซึ่งจะอธิบายต่อไป

3.6 ตัวนับโปรแกรม (PROGRAM COUNTER and STACK)

ตัวนับโปรแกรมเป็นตัวนับอิสระขนาด 12 บิต ดังรูปที่ 3.8 ในการทำงานปกติ จะเพิ่มครั้งละหนึ่งเพื่อใช้สำหรับการชี้แอดเดรสของหน่วยความจำโปรแกรมเพื่อเฟทช์รหัสคำสั่งตามลำดับด้วยขนาด 10,11,12 บิตของตัวนับโปรแกรมนี้จะใช้เป็นตัวชี้เลขที่อยู่ขนาด 1024,2048 และ4096 ไบต์ของหน่วยความจำโปรแกรมบนชิปเบอร์ 8048AH, 8049AH และ 8050AH ตามลำดับ สำหรับบิต 11 ของตัวนับโปรแกรมจะไม่เปลี่ยนไปตามการเพิ่มของตัวนับโปรแกรม บิตนี้สามารถจะเปลี่ยนได้ด้วยการใช้คำสั่ง SELECT PROGRAM BANK เท่านั้น เป็นการเลือกชุดของหน่วยความจำโปรแกรมส่วน 2 กิโลไบต์แรกหรือ 2 กิโลไบต์หลัง ในกรณีที่ใช้หน่วยความจำโปรแกรมภายนอก บิต 0-7 ของตัวนับโปรแกรมนี้จะส่งออกมาปรากฏที่บัสเมื่อสัญญาณ ALE เริ่มตกลง ส่วนบิต 8-11 ของตัวนับโปรแกรมจะถูกส่งออกทาง 4 บิต อันดับต่ำของพอร์ต 2 เพื่อใช้สำหรับเป็นตัวชี้แอดเดรสอันดับสูงของหน่วยความจำโปรแกรม

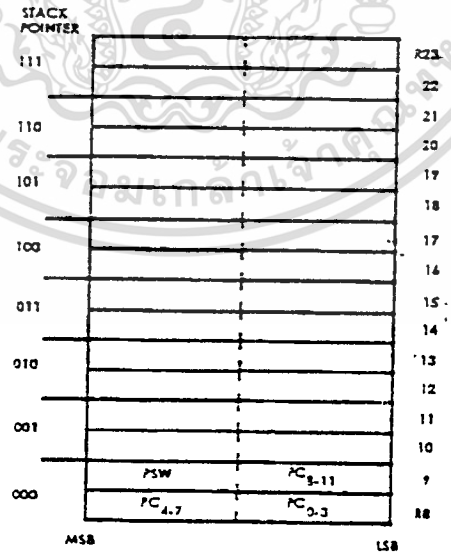


Conventional Program Counter

Counts 0001H to 7FFFH
overflows 7FFFH to 0001H

รูปที่ 3.8 ตัวนับโปรแกรม

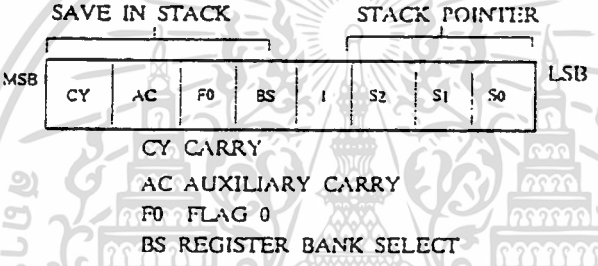
ในการอินเตอร์รัพหรือใช้คำสั่ง CALL เข้าสู่โปรแกรมย่อย จะเป็นเหตุให้ค่าตัวนับโปรแกรมถูกย้ายไปเก็บไว้ที่บริเวณสแตก 8 คู่รีจิสเตอร์เป็น PROGRAM COUNTER STACK ดังรูปที่ 1.9 ตัวชี้สแตกจะมีค่า 3 บิต ที่อยู่เป็นส่วนหนึ่งของ PSW จะเป็นตัวชี้ 8 คู่รีจิสเตอร์ในบริเวณสแตก โดยที่ข้อมูลของแรมภายในตำแหน่งที่ 8-23 จะใช้เป็นรีจิสเตอร์สแตก 8 คู่สำหรับเก็บค่าตัวชี้โปรแกรม 12 บิต และค่านิบเบิ้ลสูงของรีจิสเตอร์ที่ทำหน้าที่เป็นคำแสดงสถานะของโปรแกรม ดังรูปที่ 3.10 ตัวชี้สแตกเมื่อถูกติดตั้งเริ่มแรก จะชี้ที่ 000 ซึ่งจะคลุมตำแหน่งแรมที่ R8 และ R9 เป็นคู่รีจิสเตอร์สแตก ดังนั้นเมื่อเกิดมีการอินเตอร์รัพต์ หรือ มีการเรียกโปรแกรมย่อย จะมีการเก็บข้อมูลของตัวชี้โปรแกรมไว้ที่ตำแหน่ง R8 และ R9 และตัวชี้สแตกจะเพิ่มขึ้นอีกหนึ่ง คือ 001 ชี้ที่คู่ตำแหน่ง R10 และ R11 ดังนั้นบริเวณสแตก สามารถใช้โครงข่ายการเรียกโปรแกรมย่อยซ้อนภายในโปรแกรมย่อยซึ่งจะเรียกได้ 8 ครั้ง โดยไม่เกิด Overflow ของสแตก ถ้าเกิด Overflow ขึ้นตัวชี้สแตกจะเกิดการวนกลับมาเริ่มตำแหน่งตัวชี้สแตกต่ำใหม่อีกครั้งและจะทับค่าเดิม เนื่องจากการเกิด Overflow ของสแตกจากตำแหน่ง 111 มาหา 000 และจะเกิด Underflow จาก 000 มาหา 111 จากการสิ้นสุดโปรแกรมย่อยด้วยการใช้คำสั่ง RET หรือ RETR จะเป็นเหตุให้ตัวชี้สแตกลดค่าลงและจะเป็นการย้ายข้อมูลในคู่สแตกรีจิสเตอร์มาเข้าสู่ตัวชี้โปรแกรม



รูปที่ 3.9 บริเวณสแตกที่เก็บตัวนับโปรแกรม

3.7 รีจิสเตอร์คำแสดงสถานะโปรแกรม (PROGRAM STATUWORD: PSW)

คำแสดงสถานะโปรแกรมมีขนาด 8 บิต สามารถอ่านและเขียนผ่านแอคคิวมูเลเตอร์ได้เพื่อดูหรือเซตสถานะเริ่มต้นก่อนที่จะทำตามชุดคำสั่งที่ได้เขียนไว้ คำแสดงสถานะของโปรแกรมเป็นรีจิสเตอร์ที่เก็บตัวชี้สแตกและสภาวะผลลัพธ์ต่าง ๆ ของโปรแกรมที่ทำงานอยู่ ดังรูปที่ 3.10 การที่สามารถเขียนหรืออ่านได้ของคำแสดงสถานะโปรแกรม ทำให้สามารถเก็บสถานะต่าง ๆ หลังการเกิดปัญหาแหล่งจ่ายไฟและเรียกกลับคืนมาได้หลังจากการจ่ายไฟปกติ หรือหลังจากการจ่ายไฟปกติ หรือหลังการใช้บริการการอินเตอร์รัพต์



รูปที่ 3.10 PSW

คำแสดงสถานะของโปรแกรมจะมีส่วนประกอบดังนี้

บิต 0-2 : เป็นตัวชี้สแตก (s0,s1,s2)

บิต 3 : เป็นบิตที่ไม่ได้ใช้งาน โดยปกติเมื่อมีการอ่านเข้ามาจะมีค่าเป็น 1 เสมอ

บิต 4 : เป็นบิตใช้ในการเลือกชุดรีจิสเตอร์ (REGISTER BANK SWITCH) โดย

0 = รีจิสเตอร์ชุดที่ 1 (แอดเดรส 0-7) หรือ แบงค์ 0

1 = รีจิสเตอร์ชุดที่ 2 (แอดเดรส 24-31) หรือ แบงค์ 1

บิต 5 : เป็นบิตแฟลกศูนย์ ผู้ใช้สามารถควบคุมโดยเซตหรือรีเซตได้ตามสถานะของคำสั่ง เพื่อใช้ประโยชน์เป็นเงื่อนไขในการใช้คำสั่งกระโดด JF0

บิต 6 : AUXILIARY CARRY (AC) เป็นตัวทดบิตจากบิต 3 ไป บิต 4 เมื่อใช้คำสั่ง ADD ใช้สำหรับเป็นเงื่อนไขในการปรับค่าเป็นเลขฐานสิบ DAA (DECIMAL ADJUST)

บิต 7 : CARRY BIT บิตตัวทด เป็นบิตที่แสดงสถานะเป็นตัวทด เมื่อผลลัพธ์ที่แอคคิวมูเลเตอร์มีค่ามากกว่า 8 บิต

คำแสดงสถานะโปรแกรม บิต 4-7 หรือนิบเบิ้ลสูง จะถูกเก็บในสแตก เมื่อมีการเรียกโปรแกรมย่อยทำงาน และสามารถนำค่าของคำแสดงสถานะของโปรแกรมกลับมาได้ เมื่อใช้คำสั่ง RETR ส่วนคำสั่ง RET จะเป็นการสลับจากโปรแกรมย่อยที่ไม่เก็บคำแสดงสถานะของโปรแกรมเดิมไว้

3.8 การใช้ลอจิกเป็นเงื่อนไขในการกระโดด

เงื่อนไขต่าง ๆ ที่เกิดขึ้นทั้งภายในและภายนอกของ MCS-48 สามารถที่จะทำการทดสอบ เพื่อกระโดดไปยังโปรแกรมต่าง ๆ ที่ได้เขียนไว้ตามเงื่อนไขที่ถูกต้องโดยใช้คำสั่งแบบมีเงื่อนไข (CONDITION JUMP INSTRUCTION) เงื่อนไขต่าง ๆ ที่ใช้สำหรับการกระโดดแบบมีเงื่อนไข แสดงดังตารางที่ 3.2 ใช้ซอฟต์แวร์ด้วยการตรวจสอบ ค่าศูนย์ทั้งแปดบิตในแอคคิวมูเลเตอร์ หรือไม่เป็นศูนย์ ส่วนการทดสอบเฉพาะบิตสามารถทดสอบได้ด้วยการใช้คำสั่งทางลอจิก และทดสอบผลที่ได้ในแอคคิวมูเลเตอร์ในค่าที่ไม่เป็นศูนย์ การทดสอบบิตตัวทดสอบได้ทั้งแบบเซตและรีเซตค่านี้ การทดสอบบิต F0 F1 และแฟลก OVERFLOW ของตัวจับเวลาทดสอบได้เฉพาะการเซตค่า การทดสอบสัญญาณเข้า T0 T1 ทดสอบได้ทั้งแบบเซตและรีเซต ส่วนการ INT ทดสอบได้เฉพาการีเซตเท่านั้น

ตารางที่ 3.2 การโลอจิกเป็นเงื่อนไขในการกระโดด

Device Testable	Jump Condition	
Accumulator	All Zero	Not Zero
Accumulator Bit	--	1
Carry Flag	0	1
User Flag (F0, F1)	--	1
Timer Overflow Flag	--	1
Test Inputs (T0, T1)	0	1
Interrupt Input (INT)	0	--

3.9 การอินเตอร์รัพต์ (INTERRUPT)

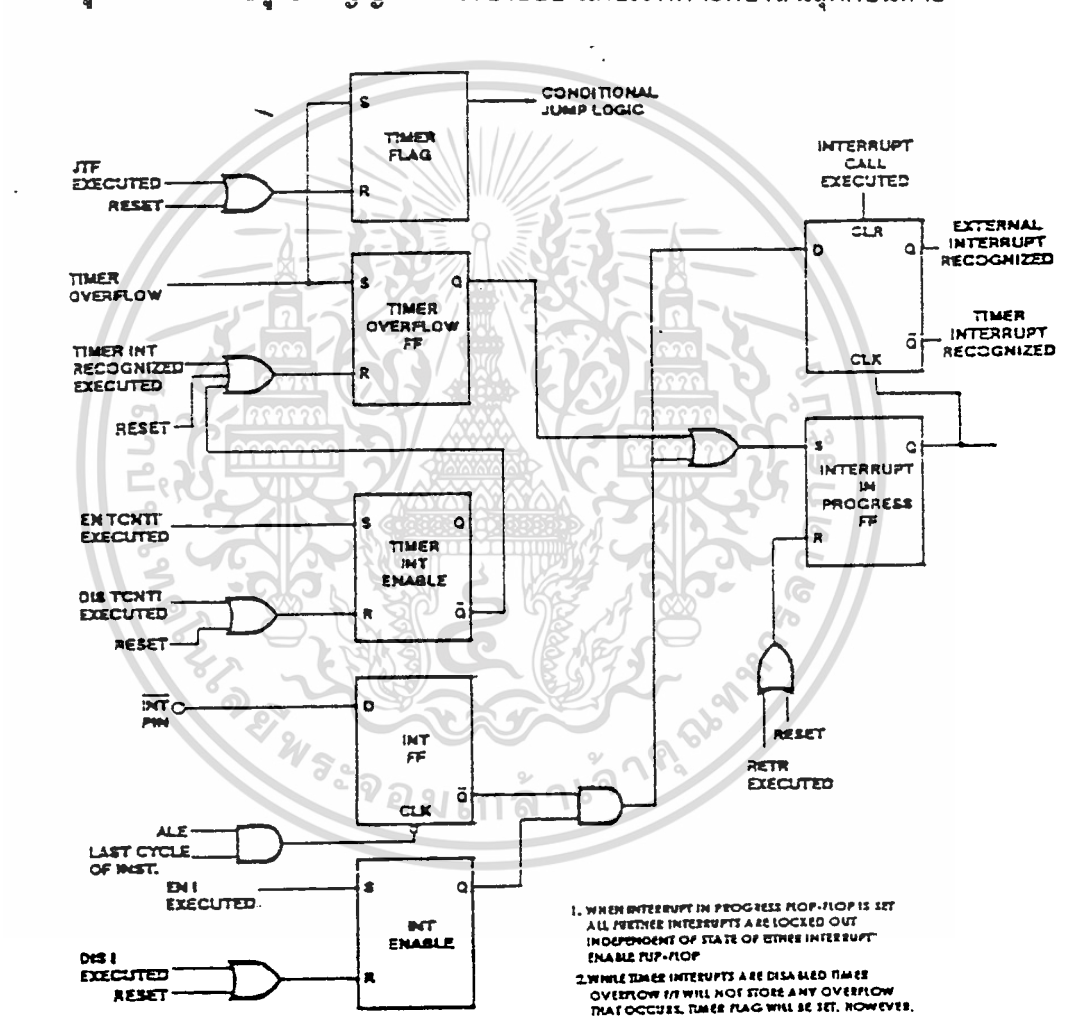
MCS-48 จะรับการอินเตอร์รัพต์ต่อเมื่อได้เซตอีนาเบิ้ลแล้ว ลำดับการอินเตอร์รัพต์จะเกิดขึ้น เมื่อมีสัญญาณสถานะลอจิกเป็น 0 เข้าที่ขา INT โดยที่การอินเตอร์รัพต์จะกระตุ้นด้วยสัญญาณระดับต่ำ และสามารถที่จะรับการอินเตอร์รัพต์เข้ามาได้หลายแหล่งด้วยกัน ดังแสดงการอินเตอร์รัพต์ทางลอจิก ในรูปที่ 3.11 ขาอินเตอร์รัพต์จะถูกแซมปลิ้ง (SAMPLE) ทุกวัฏจักรคำสั่งในช่วงวัฏจักรการทำงาน ALE และเมื่อตรวจพบ (ซึ่งสัญญาณอินเตอร์รัพต์จะต้องมีสถานะต่ำอย่างน้อย 3 วัฏจักรแมชชีน เพราะสัญญาณจะถูกตรวจที่วัฏจักรที่ 2 ของการทำงานของคำสั่ง) ก็จะเรียกโปรแกรมย่อยการอินเตอร์รัพต์ โดยก่อนที่จะเข้าสู่โปรแกรมย่อยบริการอินเตอร์รัพต์ ตัวนับโปรแกรมและ บิตที่ 4-7 ของคำแสดงสถานะของโปรแกรมจะถูกเก็บไว้ที่สแตก และการควบคุมการทำงานต่าง ๆ เมื่อมีการอินเตอร์รัพต์นั้น จะถูกส่งผ่านไปทำงานที่ตำแหน่ง 003 ของหน่วยความจำโปรแกรม ซึ่งโดยปกติจะเขียนด้วยคำสั่งกระโดดแบบไม่มีเงื่อนไข (Unconditional Jump) ไปยังโปรแกรมบริการที่ใช้ตอบสนองการอินเตอร์รัพต์ การกลับคืนหลังจากบริการโปรแกรมย่อยของการอินเตอร์รัพต์ จะกลับโดยใช้คำสั่ง RETR ซึ่งจะยังคงรักษาคำแสดงสถานะของโปรแกรมไว้ การอินเตอร์รัพต์ของ MCS-48 เป็นการอินเตอร์รัพต์ระดับเดียว ดังนั้นระหว่างการบริการโปรแกรมอินเตอร์รัพต์อยู่ การอินเตอร์รัพต์จะไม่สามารถได้รับบริการ ดังนั้นจะต้องมีการอีนาเบิ้ลอินเตอร์รัพต์ทุกครั้งด้วยคำสั่ง RETR เสมอ ซึ่งจะอีนาเบิ้ลได้ หลังวัฏจักรที่ 2 ของการทำงานคำสั่ง RETR

การอินเตอร์รัพต์ที่เกิดจากตัวจับเวลา/ตัวนับ ที่เกิดจากการ OVERFLOW ภายในรีจิสเตอร์ของตัวนับเวลา/ตัวนับ ก็จะทำงานเช่นเดียวกับการอินเตอร์รัพต์ภายนอก จะต่างกันก็ตรงที่ว่า การอินเตอร์รัพต์ที่เกิดจากตัวจับเวลา/ตัวนับนั้น จะไปทำที่ตำแหน่ง 007 ซึ่งเป็นตำแหน่งที่โปรแกรมย่อยการอินเตอร์รัพต์ของตัวจับเวลา/ตัวนับอยู่ การอินเตอร์รัพต์จากภายนอกจะมีลำดับสูงกว่าการอินเตอร์รัพต์จากตัวจับเวลา/ตัวนับ หมายความว่า ถ้ามีการอินเตอร์รัพต์จากภายนอกพร้อมกับการอินเตอร์รัพต์จากตัวตั้งเวลาแล้ว MCS-48 จะตอบสนองต่อการอินเตอร์รัพต์จากภายนอกก่อนเสมอเมื่อต้องการดิสเอเบิ้ลการอิน

เตอร์รัพต์ สามารถทำได้ 2 วิธีคือ การรีเซต MCS-48 กับการใช้คำสั่งดิสเอเบิ้ล DIS I และ DIS TCNTI ทั้งการอินเตอร์รัพต์ภายนอกกับตัวจับเวลา/ตัวนับ

3.9.1 จังหวะเวลาอินเตอร์รัพต์ (Interrupt Timing)

การอินเตอร์รัพต์จะมีอีนาเบิ้ลหรือดิสเอเบิ้ลได้ด้วยโปรแกรมควบคุมด้วยการใช้คำสั่ง EN I และ DIS I การอินเตอร์รัพต์จะดิสเอเบิ้ลได้ด้วยการรีเซต และจะยังคงดิสเอเบิ้ลจนกว่าจะถูกอีนาเบิ้ลด้วยผู้ใช้ สัญญาณการร้องขออินเตอร์รัพต์จะต้องสิ้นสุดก่อนที่จะ



รูปที่ 3.11 วงจรตรรกการอินเตอร์รัพต์

ทำงานกับคำสั่ง RETR เพื่อที่จะให้ตัวโพรเซสเซอร์สามารถที่จะบริการการอินเตอร์รัพต์ตัวต่อไปได้ทันที มิฉะนั้นจะเข้าอินเตอร์รัพต์เดิม ฉะนั้นอุปกรณ์ต่อพ่วงการอินเตอร์รัพต์จะมีการป้องกันลักษณะเช่นนี้ด้วยการรีเซตการร้องขอการอินเตอร์รัพต์ เมื่อไรที่ตัวไมโครคอนโทรลเลอร์ได้อ่านหรือเขียนข้อมูลจากรีจิสเตอร์บัฟเฟอร์ของอุปกรณ์ต่อพ่วงแล้ว ถ้าอุปกรณ์การอินเตอร์รัพต์ไม่ต้องการเข้าถึงไมโคร ก็อาจจะมีการใช้ MCS-48 เป็นตัวส่งสัญญาณตอบรับการรับรู้อินเตอร์รัพต์จากการร้องขอของอุปกรณ์ต่อพ่วงนั้นขา INT ยังอาจใช้ทดสอบด้วยการใช้คำสั่งกระโดดอย่างมีเงื่อนไข JNI คำสั่งนี้จะใช้เป็นตัวตรวจจับการอินเตอร์รัพต์ที่เกิดขึ้นก่อนการอินเตอร์รัพต์ MCS-48 จะเกิดขึ้น ถ้าการอินเตอร์รัพต์อยู่ในสภาวะดิสเอเบิ้ล ก็อาจใช้เป็นตัวตรวจสอบสัญญาณอินพุทธรรมดาทั่วไป เช่นเดียวกับขา T0 และ T1

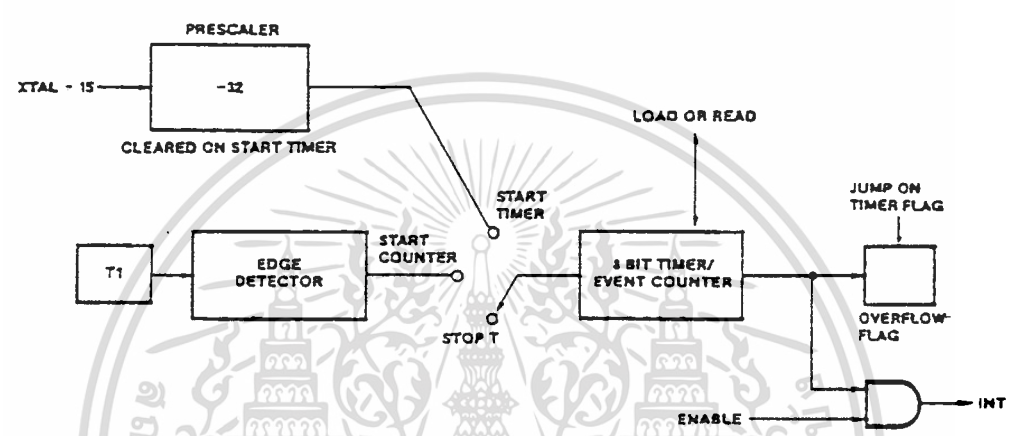
3.10 ตัวจับเวลา/ตัวนับ (TIMER/COUNTER)

MCS-48 จะประกอบด้วยตัวจับเวลา/ตัวนับ การทำงานของทั้งสองจะคล้ายกันเพียงแต่ว่าตัวนับจะมีการส่งอินพุตเข้าตัวนับเท่านั้น

3.10.1 ตัวนับ

เป็นตัวนับแบบเลขฐานสองขนาด 8 บิต ที่สามารถกำหนดหรืออ่านค่าได้ด้วยคำสั่ง MOV ที่ส่งผ่านเข้าหรือออกทางแอคคิวมูเลเตอร์และการรีเซ็ตจะไม่มีผลต่อค่าที่อยู่ในตัวนับ ตัวนับจะหยุดก็ต่อเมื่อมีการรีเซ็ตหรือใช้คำสั่ง STOP TCNT และจะหยุดจนกระทั่งใช้คำสั่ง START T ในการทำงานเป็นตัวจับเวลาหรือ START CNT ในการทำงานเป็นตัวนับอีกครั้ง ทันทีที่ตัวนับเริ่มนับจะเพิ่มค่าตัวเองทีละ 1 จนกระทั่งถึงค่าสูงสุด 00FH แล้วเริ่มนับที่ 00 ใหม่ การนับจะเป็นเช่นนี้เรื่อยไปจนกว่าจะมีคำสั่ง STOP TCNT หรือการรีเซ็ต การเพิ่มค่าจากค่าสูงสุดไปยังค่าศูนย์ (OVERFLOW) จะมีผลไปเซ็ต OVERFLOW FLAG FLIP-FLOP และสร้างสัญญาณการร้องขออินเตอร์รัพต์ขึ้น สถานะของ OVERFLOW FLAG สามารถตรวจสอบได้โดยคำสั่งกระโดดแบบมีเงื่อนไข (JTF) แฟลก OVERFLOW จะถูกรีเซ็ตก็ต่อเมื่อมีการทำงานของคำสั่ง JTF หรือการรีเซ็ต MCS-48 สัญญาณอินเตอร์รัพต์จะ

ถูกแลทช์และ OR กับการอินเตอร์รัพต์จากภายนอก การอินเตอร์รัพต์ของตัวนับสามารถอีนาเบิ้ลหรือดิสเอเบิ้ลด้วยคำสั่ง EN TCNTI และ DIS TCNTI หากมีการอีนาเบิ้ล เมื่อตัวนับเกิด OVERFLOW จะไปทำงานโปรแกรมย่อยบริการการทำงานของตัวจับเวลา/ตัวนับที่ตำแหน่ง 007 ทันที ตัวนับสามารถทำหน้าที่เป็นตัวนับเหตุการณ์ (EVENT COUNTER) ได้โดยที่ขา TI จะทำหน้าที่เป็นอินพุต โดยสัญญาณจากขา TI จะถูกแซมปลิ้งมาก ๆ ช่วง



รูปที่ 3.12 แสดงส่วนของตัวจับเวลา/ตัวนับของ MCS-48

เริ่มต้นสถานะ 3 หรือใน MCS-48 รุ่นใหม่ ๆ ก็จะเริ่มที่สถานะ 4 เมื่อมีการเปลี่ยนสถานะจาก สูง ไป ต่ำ ตัวนับจะเพิ่มค่าขึ้นหนึ่งเสมอ และ TI จะต้องมีสถานะ 0 อย่างน้อย 1 วัฏจักรแมชชีน เพื่อให้แน่ใจว่าการนับจะไม่ขาดหายไปดังนั้นความเร็วของการนับสูงสุด ในแต่ละครั้งจะใช้เวลา 3รอบคำสั่ง หรือทุก ๆ 5.7 ไมโครวินาที เมื่อใช้คริสตอล 8 เมกกะเฮิตซ์และอินพุตเข้า T1 จะต้องมีระดับลอจิกคงที่ 1 อย่างน้อย 1/5 วัฏจักรแมชชีน หลังจากการเปลี่ยนระดับในแต่ละครั้ง

3.10.2 ตัวจับเวลา

เมื่อใช้คำสั่ง START T ตัวจับเวลา/ตัวนับจะทำหน้าที่เป็นตัวจับเวลา ในการใช้คำสั่ง START T นั้นจะทำให้ฐานเวลาภายในถูกหารด้วย 15 และ 32 ตามลำดับ จาก

ผลที่ได้จะเพิ่มค่าของตัวจับเวลาที่ได้ตั้งไว้ก่อนแล้ว เมื่อเกินค่าที่กำหนดจะทำการอินเตอร์รัพต์ของตัวจับเวลา/ตัวนับ เพื่อไปยังโปรแกรมย่อยที่ต้องการ

ตัวอย่างเช่นเมื่อใช้คริสตอลขนาด 11 เมกกะเฮิรตซ์เป็นฐานเวลาให้ MCS-48 เมื่อผ่านตัวหาร 15 จะได้ความถี่ 733 กิโลเฮิรตซ์ และความถี่นี้จะถูกหารด้วย 32 อีก จะได้ 22,917 กิโลเฮิรตซ์ ซึ่งแสดงว่าตัวตั้งเวลาจะเพิ่มขึ้นหนึ่งทุก ๆ 44 ไมโครวินาที รูปที่ 3.12 เป็นการแสดงให้เห็นถึงส่วนประกอบการทำงานของตัวจับเวลา/ตัวนับภายใน MCS-48

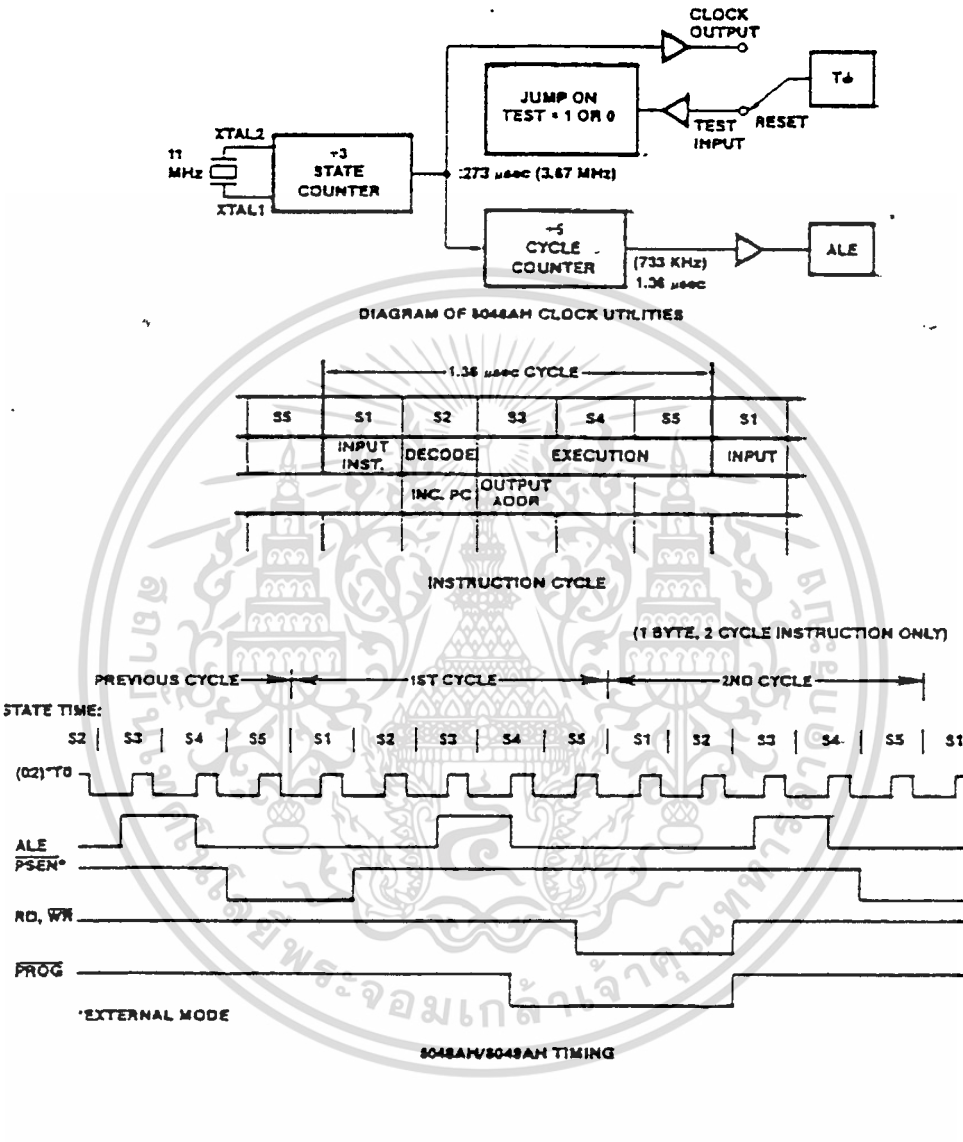
3.11 สัญญาณนาฬิกาและวงจรฐานเวลา (CLOCK AND TIMER CIRCUITS)

แหล่งกำเนิดฐานเวลาสำหรับ MCS-48 สามารถสร้างได้จากอุปกรณ์ภายนอก เช่น. คริสตอล อินดัคเตอร์ หรือแหล่งกำเนิดนาฬิกาภายนอก การทำงานของส่วนสัญญาณนาฬิกาและวงจรฐานเวลาแสดงในรูปที่ 3.13 โดยแยกกล่าวเป็นส่วน ๆ ดังนี้.

ตัวกำเนิดความถี่ (OSCILLATOR) ตัวกำเนิดความถี่เป็นวงจรแบบ SERIES RESONANT ที่มีอัตราขยายสูงโดยมีช่วงความถี่ระหว่าง 1-6 เมกกะเฮิรตซ์ ซึ่งขา X1 เป็นขาอินพุทเข้าสู่วงจรขยาย ขณะที่ขา X2 เป็นขาเอาต์พุต ในการต่อคริตอลหรืออินดัคเตอร์ระหว่าง X1 และ X2 จะทำให้เกิดการป้อนกลับและการเลื่อนเฟสสำหรับการกำเนิดความถี่ ในกรณีที่ไม่ต้องการใช้ความถี่ที่แน่นอน สามารถใช้อุปกรณ์พวกอินดัคเตอร์ แทนคริสตอลได้ ซึ่งจะให้ความถี่อยู่ในช่วง 3-5 เมกกะเฮิรตซ์ได้

ตัวนับสถานะ (STATE COUNTER) เอาต์พุตของตัวกำเนิดความถี่จะถูกหารด้วย 3 ในตัวนับสถานะ เพื่อสร้างสัญญาณนาฬิกาสำหรับการทำงานของ MCS-48 สัญญาณนาฬิกานี้สามารถส่งออกไปยังขา T0 ได้โดยใช้คำสั่ง ENT0 CLK สัญญาณนาฬิกาที่ขา T0 จะหยุดเมื่อถูกรีเซ็ต

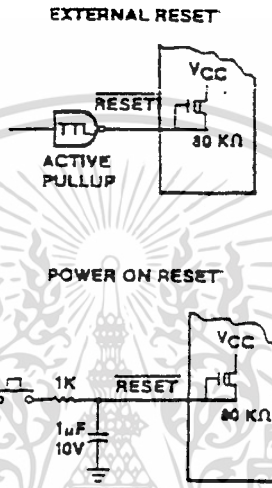
ตัวนับรอบ (CYCLE COUNTER) สัญญาณนาฬิกาจะถูกหารด้วย 5 ในตัวนับรอบเพื่อสร้างสัญญาณนาฬิกาที่ประกอบด้วยรอบการทำงาน 5 สถานะ สัญญาณนาฬิกานี้เรียกว่า ADDRESS LATCH ENABLE (ALE) เพื่อใช้เป็นสัญญาณติดต่อกับหน่วยความจำโปรแกรมภายนอกสัญญาณนี้จะออกมาอย่างต่อเนื่องตลอดเวลาขณะที่ MCS-48 ยังทำงานอยู่



รูปที่ 3.13 แสดงโครงสร้างของส่วนสัญญาณนาฬิกาและจังหวะเวลาวัฏจักร

3.12 การรีเซ็ต (RESET)

การรีเซ็ตจะทำให้ MCS-48 เริ่มการทำงานใหม่เมื่อขารีเซ็ต มีลอจิกศูนย์ซึ่งขานี้จะต้องทำงานแบบ SCHMITT-TRIGGER โดยจะมีความต้านทานที่ใช้พูลอัพอยู่ภายใน MCS-48 และภายนอกจะต่อตัวเก็บประจุค่า 1 ไมโครฟารัดอยู่เพื่อทำให้เกิดช่วงเวลาเพียงพอในการรีเซ็ต การรีเซ็ต MCS-48 สามารถทำได้ดังแสดงในรูปที่ 3.14



รูปที่ 3.14 แสดงการรีเซ็ต MCS-48

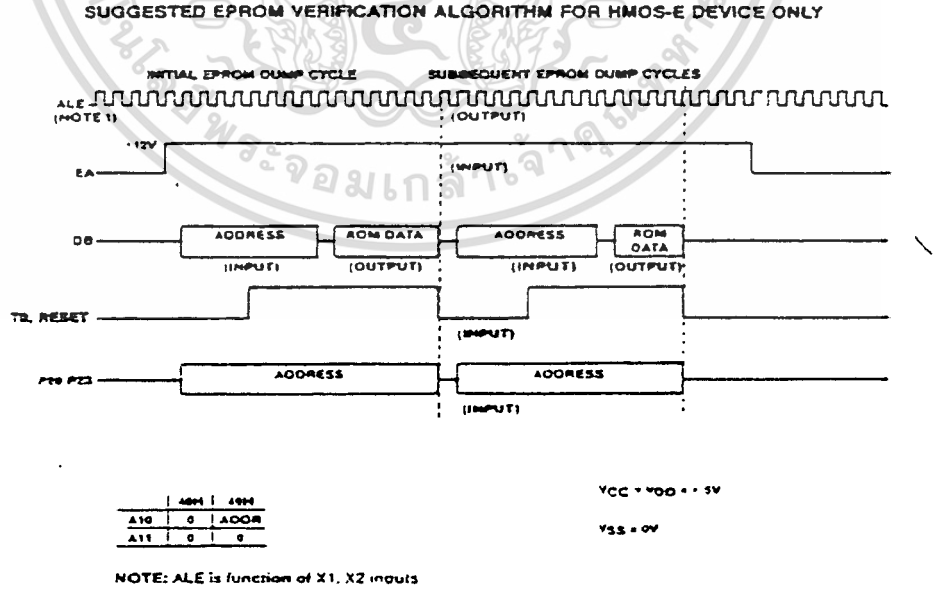
หากมีสัญญาณรีเซ็ตจากภายนอกที่ขารีเซ็ต จะต้องมีค่าลอจิกศูนย์อย่างน้อย 50 มิลลิวินาที หลังจากที่แหล่งจ่ายไฟอยู่ในสภาพที่พร้อมจะใช้งานแล้วเมื่อมีการรีเซ็ตให้ MCS-48 จะทำให้เกิด ตัวนับโปรแกรมถูกเซ็ตให้เป็นศูนย์ ตัวชี้สแตกถูกเซ็ตเป็นศูนย์ รีจิสเตอร์แบงค์ 0 จะถูกเลือก หน่วยความจำโปรแกรมแบงค์ 0 ถูกเลือก บัสจะถูกเซ็ตเป็น HIGH IMPEDANCE ยกเว้นเมื่อขา EA มีสถานะลอจิกหนึ่งพอร์ต 1 และพอร์ต 2 จะอยู่ในกรณีเป็นอินพุต อินเตอร์รัพต์ถูกดิสเอเบิ้ล ตัวจับเวลาจะหยุด แฟลกของตัวจับเวลาถูกทำให้เป็นศูนย์แฟลกศูนย์และแฟลกหนึ่งถูกทำให้เป็นศูนย์

3.13 การอ่านคำสั่งจากหน่วยความจำภายนอก (EXTERNAL ACCESS MODE)

โดยปกติการใช้งานไอซีตระกูล MCS-48 นั้นจะใช้หน่วยความจำโปรแกรมภายในแต่ในบางกรณี เช่นขณะทำการแก้ไขข้อมูลอยู่ หรือทดสอบโปรแกรมที่เขียนขึ้นมา ส่วนมากแล้วจะใช้ความสามารถของไอซีตระกูล MCS-48 จะไม่สามารถใช้หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอกพร้อมกันได้ แต่ก็ยังมีเทคนิคในการใช้งานแบบนี้ได้โดยอาศัยสัญญาณรีเซ็ตเข้าช่วย โดยจะให้เปลี่ยนสถานะลอจิกของ EA ขณะถูกรีเซ็ตเท่านั้น นอกจากนั้นยังสามารถอ่านรหัสโปรแกรมจากหน่วยความจำโปรแกรมภายในโดยอาศัยขา EA ได้อีกด้วย ขั้นตอนของการอ่านเป็นดังนี้

- ขารีเซ็ตจะถูกเซ็ตให้มีสถานะลอจิกศูนย์
- MCS-48 จะอยู่ในโหมดการอ่านหน่วยความจำโปรแกรมภายใน เมื่อขา EA ถูกป้อนแรงดัน 12 โวลต์
- ตำแหน่งของคำสั่งจะถูกเซ็ตที่บัสและ 4 บิตแรกของพอร์ต 2
- ขารีเซ็ตจะถูกทำให้มีสถานะลอจิกหนึ่ง ขณะเดียวกับตำแหน่งของหน่วยความจำภายในจะถูกแลทช์ไว้แล้ว
- หลังจากแลทช์ตำแหน่งของหน่วยความจำโปรแกรมภายในแล้ว สัญญาณรีเซ็ตจะยังคงอยู่ที่สถานะลอจิกหนึ่งอยู่ จะทำให้ข้อมูลที่ตำแหน่งนั้นถูกส่งออกมาที่บัส

แผนผังการอ่านข้อมูลออกจากหน่วยความจำโปรแกรมภายในแสดงในรูปที่ 3.15



รูปที่ 3.15 แผนภูมิจังหวะเวลาการอ่านข้อมูลออกจากหน่วยความจำโปรแกรมภายใน

3.14 ชุดคำสั่งของตระกูล MCS-48

ชุดคำสั่งหรือสำหรับตระกูล MCS-48 สามารถแบ่งตามชนิดการทำงานได้ดังนี้

1. คำสั่งควบคุม (CONTROL INSTRUCTION)

คำสั่งควบคุมจะทำให้โปรแกรมที่ใช้งานสามารถควบคุมการอินเตอร์รัพต์ การเลือกชุดของหน่วยความจำโปรแกรม และการให้สัญญาณนาฬิกาออกมาเพื่อใช้ในงานควบคุมเมื่อ ตระกูล MCS-48 เริ่มทำงานนั้น สัญญาณอินเตอร์รัพต์ที่เข้าทางขา INT จะถูกดิสเอเบิ้ลโดยอัตโนมัติ การอินเตอร์รัพต์จากภายนอกสามารถควบคุมการอินเตอร์รัพต์ได้โดยคำสั่งเกี่ยวกับการอินเตอร์รัพต์ แต่อย่างไรก็ตาม การร้องขอด้วยสัญญาณอินเตอร์รัพต์ที่ขา INT จะไม่ได้รับการตอบรับ เมื่อการทำงานของตระกูล MCS-48 ยังอยู่ในโปรแกรมการอินเตอร์รัพต์อยู่เนื่องจากการอินเตอร์รัพต์ของตระกูล MCS-48 เป็นการอินเตอร์รัพต์ระดับเดียว หรือภายในโปรแกรมที่ทำงานอยู่ไม่ได้สั่งให้มีการรับอินเตอร์รัพต์สำหรับคำสั่งที่ใช้เลือกชุดการทำงาน 4 คำสั่งนั้น 2 คำสั่งจะใช้เลือกรีจิสเตอร์ทำงานที่อยู่ในส่วนของแรมภายในตระกูล MCS-48 และอีก 2 คำสั่งใช้สำหรับเลือกหน่วยความจำโตปรแกรมซึ่งเป็นรอมคำสั่งที่ใช้เลือกชุดของหน่วยความโปรแกรม จะทำหน้าที่เซตค่า MOMERY BANK FLIP-FLOP (DBF) เพื่อใช้เลือกชุดของหน่วยความจำโปรแกรม ค่าของ DBF จะยังค้างค่าเดิมอยู่ จนกว่าจะมีคำสั่งเลือกชุดใหม่เข้ามา บิต 11 ของตัวนับโปรแกรมจะไม่เพิ่มตาม ตามการเพิ่มของบิตอื่น แต่บิตนี้จะเปลี่ยนแปลงตาม DBF เมื่อคำสั่ง CALL และคำสั่ง JUMP ทำงาน ซึ่งจะมีประโยชน์เมื่อมีโปรแกรมย่อยอยู่ในส่วนของหน่วยความจำอีกชุดหนึ่งทำงานได้ทันทีโดยที่ข้อมูลเดิมที่มีอยู่ในรีจิสเตอร์เก่า ยังคงเก็บรักษาอยู่เหมือนเดิม คำสั่งเลือกชุดรีจิสเตอร์นี้มีประโยชน์มากในการใช้อินเตอร์รัพต์ เพราะสามารถใช้รีจิสเตอร์ชุดหนึ่ง สำหรับการอินเตอร์รัพต์ และอีกชุดหนึ่งสำหรับการทำงานปกติสำหรับคำสั่งที่ใช้ควบคุมสัญญาณนาฬิกาที่จะส่งออกไปยังขา T0 นั้น มีประโยชน์สำหรับให้สัญญาณนาฬิกากับอุปกรณ์ภายนอกที่ติดต่ออยู่กับตระกูล MCS-48 ซึ่งสามารถควบคุมสัญญาณนาฬิกาด้วยคำสั่งได้

2. คำสั่งเคลื่อนย้ายข้อมูล (MOVE DATA INSTRUCTION)

ชุดของคำสั่งเคลื่อนยายข้อมูล เป็นชุดคำสั่งแรกในการเคลื่อนย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์ กับรีจิสเตอร์ต่าง ๆ หรือหน่วยความจำของระบบ การเคลื่อนย้ายข้อมูล

หว่างแอคคิวมูเลเตอร์กับรีจิสเตอร์จะเป็นการเคลื่อนย้ายโดยตรง โดยกำหนดทิศทางการเคลื่อนย้ายในคำสั่งได้ แต่การเคลื่อนย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์กับหน่วยความจำนั้น เป็นการเคลื่อนย้ายการอ้อมโดยจะมีตัวชี้ตำแหน่งที่ต้องการเคลื่อนย้ายอยู่ในรีจิสเตอร์ R0 หรือ R1 ชุดรีจิสเตอร์แต่ละชุด การเคลื่อนย้ายระหว่างหน่วยความจำข้อมูลภายในจะใช้เวลาเพียงหนึ่งรอบการทำงาน ขณะที่การเคลื่อนย้ายกับหน่วยความจำข้อมูลภายนอกจะใช้เวลาสองรอบการทำงาน ขณะเดียวกันข้อมูลที่เก็บอยู่ในส่วนหน่วยความจำโปรแกรมสามารถอ่านเข้ามาเก็บไว้ในแอคคิวมูเลเตอร์ได้โดยตรง ซึ่งมีประโยชน์ในการจัดข้อมูลเป็นตารางไว้สำหรับเปรียบเทียบได้ นอกจากนี้ยังสามารถเคลื่อนย้ายข้อมูลของตัวนับ/ตัวจับเวลา หรือรีจิสเตอร์บอกสถานะโปรแกรม เข้าสู่หรือออกจากแอคคิวมูเลเตอร์ได้ ความสามารถในการเคลื่อนย้ายข้อมูลระหว่างตัวแอคคิวมูเลเตอร์กับรีจิสเตอร์บอกสถานะโปรแกรม ทำให้สามารถเปลี่ยนสถานะการทำงานของโปรแกรมได้ หรือเปลี่ยนตัวชี้สแตกที่อยู่ในรีจิสเตอร์บอกสถานะโปรแกรมได้ เมื่อมีความจำเป็นที่จะต้องสลับที่อยู่ข้อมูลในตระกูล MCS-48 จะมีคำสั่งพิเศษอีกคำสั่งต่อ XCHD A เป็นคำสั่งที่ทำงานร่วมกับคำสั่ง SWAP A เพื่อความสะดวกในการย้ายครั้งละ 4 บิต หรือการเคลื่อนย้ายของเลข BCD คำสั่งนี้จะมีการแลกเปลี่ยนระหว่างข้อมูล 4 บิตแรกของแอคคิวมูเลเตอร์กับ 4 บิต แรกของหน่วยความจำข้อมูลภายใน เมื่อทำงานร่วมกับคำสั่ง SWAP A จะทำให้สามารถเก็บรักษาข้อมูลครั้งละ 4 บิตหรือเลข BCD ไว้ในหน่วยความจำข้อมูลภายใน

3. คำสั่งเกี่ยวกับตัวจับเวลา/ตัวนับ (TIMER/COUNTER INSTRUCTION)

ชุดคำสั่งเกี่ยวกับตัวจับเวลา/ตัวนับ จะทำหน้าที่เริ่มการทำงานของตัวจับเวลา/ตัวนับ การทำงานของตัวจับเวลา/ตัวนับสามารถควบคุมได้ด้วยคำสั่งโดยตรงตัวจับเวลา/ตัวนับ สามารถใช้เป็นตัวจับเวลาที่ใช้ฐานเวลาจากสัญญาณนาฬิกาภายในหรือสัญญาณนาฬิกาภายนอก กับตัวนับซึ่งทำหน้าที่นับสัญญาณที่เข้ามาทางขา T1 การเลือกการทำงานทั้งสองแบบเลือกได้โดยคำสั่ง นอกจากนี้ตัวจับเวลา/ตัวนับสามารถอ่านหรือเขียนผ่านทางแอคคิวมูเลเตอร์ได้ไม่ว่าตัวนับหรือตัวจับเวลาทำงานอยู่หรือไม่ ทำให้สามารถตรวจสอบได้ว่าทุกเวลาที่ต้องการ

4. คำสั่งเกี่ยวกับแอคคิวมูเลเตอร์ (ACCUMULATOR INSTRUCTION)

ชุดคำสั่งเกี่ยวกับแอคคิวมูเลเตอร์จะประกอบด้วย การบวกข้อมูลที่มีตัวทด และไม่มีตัวทดการทำงานเกี่ยวกับลอจิก คำสั่งเหล่านี้ คือ ADD,AND,OR,XOR หรือ ROTATE เพื่อ

สร้างรูปแบบของข้อมูลโดยตรง ข้อมูลสำหรับหน่วยความจำหรือการเลือกชุดของรีจิสเตอร์ ข้อมูลสามารถเคลื่อนย้ายระหว่างแอคคิวมูเลเตอร์กับรีจิสเตอร์หรือหน่วยความจำขึ้นกับคำสั่งที่ใช้งานนอกจากนี้ยังมีคำสั่งพิเศษสำหรับแอคคิวมูเลเตอร์คือ SWAP A และ XCHD A ซึ่งได้กล่าวไปแล้วในกรณีที่แอคคิวมูเลเตอร์คำนวณให้อยู่ในรูปแบบของ BCD ได้ นอกจากการทำงานทางคณิตศาสตร์แล้วชุดคำสั่งของแอคคิวมูเลเตอร์ยังประกอบด้วย คำสั่งเกี่ยวกับวนรอบซ้ายหรือขวา รวมหรือไม่รวมบิตตัวทด COMPLEMENT,INCREMENT,DECREMENT หรือการทำให้เป็นศูนย์ เนื่องจากชุดคำสั่งของตระกูล MCS-48 ไม่มีคำสั่งการลบ ดังนั้นเมื่อต้องการลบเลขขึ้นจะต้องใช้คำสั่ง 3 คำสั่งทำหน้าที่การลบโดยผ่านแอคคิวมูเลเตอร์ คือ

```
CPL A  ; COMPLEMENT  THE  ACCUMULATOR
ADD A  ; ADD THE VALUE TO THE ACCUMULATOR
INC A  ;  ADD 1 TO THE ACCUULATOR
```

หลังจากทำคำสั่งดังกล่าวผลลัพธ์จะเก็บไว้ในแอคคิวมูเลเตอร์เมื่อต้องการเปลี่ยนแปลงข้อมูลของรีจิสเตอร์แสดงสถานะโปรแกรมสามารถทำได้โดยการเขียนผ่านแอคคิวมูเลเตอร์ได้

5. คำสั่งกระโดด (BRANCH INSTRUCTION)

ชุดคำสั่งกระโดดนี้ จะทำหน้าที่การกระโดดที่มีเงื่อนไขหรือไม่มีไปยังส่วนของหนวยความจำโปรแกรมใด ๆ เพื่อให้ทำคำสั่งในส่วนของหน่วยความจำโปรแกรมนั้น ๆ ในกรณีของคำสั่งกระโดดแบบไม่มีเงื่อนไขนั้น จะสามารถกระโดดไปยังส่วนใดส่วนหนึ่งของหน่วยความจำโปรแกรมภายในชุดของหน่วยความจำโปรแกรมใดโปรแกรมหนึ่ง หมายความว่าการกระโดดของโปรแกรมจะอยู่ในช่วง 2 กิโลไบต์เท่านั้น สำหรับในกรณีที่ต้องการกระโดดข้ามระหว่างชุดของหน่วยความจำโปรแกรม สามารถทำได้โดยการใช้คำสั่งเลือกชุดของหน่วยความจำโปรแกรมที่ต้องการกระโดดก่อนที่จะใช้คำสั่งกระโดด คำสั่งการเลือกชุดของหน่วยความจำโปรแกรมจะมีผลทำงานเมื่อคำสั่งกระโดดในกรณีของการใช้ชุดคำสั่งกระโดดแบบมีเงื่อนไขนั้น สามารถตรวจสอบสัญญาณอินพุตหรือสถานะการทำงานก่อนทำกระโดดได้ เมื่ออินพุตหรือสถานะการทำงานถูกต้องตามเงื่อนไขหรือไม่ เงื่อนไขที่ใช้กับคำสั่งการกระโดดมีดังนี้

```
T0 = 1  หรือ  0
T1 = 1  หรือ  0
```

INT = 0

แอคคิวมูเลเตอร์ = หรือ <> 0\

แอคคิวมูเลเตอร์ = 1

CARRY = 1 หรือ 0

F0 = 1

F1 = 1

TIMER FLAG = 1

การกระโดดแบบมีเงื่อนไขนี้ จะมีช่วงกระโดดอยู่ในเพจเดียว ขณะที่ทำงานคำสั่งนั้นอยู่หรือภายในช่วง 256 ตำแหน่ง เมื่อเงื่อนไขที่ถูกทดสอบต้องการทำงานจะกระโดดไปยังส่วนที่กำหนดทันที นอกจากเงื่อนไขที่กล่าวมาแล้วยังมีการกระโดเมื่อค่ารีจิสเตอร์ตัวใดตัวหนึ่งไม่เป็นศูนย์หลังจากมีการลดค่ารีจิสเตอร์นั้น ๆ

6. คำสั่งเกี่ยวกับอุปกรณ์ภายนอก (INPUT/OUTPUT INSTRUCTION)

ชุดคำสั่งเกี่ยวกับอุปกรณ์ภายนอกประกอบด้วย การเคลื่อนย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์กับพอร์ต โดยที่พอร์ตจะแลทช์ค่าที่แอคคิวมูเลเตอร์สั่งให้กรณีเป็นเอาท์พุตพอร์ต แต่กรณีเป็นอินพุตพอร์ตนั้นจะไม่มีการแลทช์ นอกจากนี้ยังสามารถทำการ AND หรือ OR โดยตรงกับพอร์ต โดยผลลัพธ์ของการ AND และ OR จะคงเก็บไว้ที่พอร์ตนั้นอยู่จากวิธีการ AND หรือ OR กับพอร์ตได้โดยตรงนี้เอง ทำให้สามารถเซตค่าของแต่ละบิตของพอร์ตได้โดยตรง ในกรณีต้องการเป็นอินพุตพอร์ตนั้น จะต้องทำให้บิตที่ต้องการทำเป็นอินพุตมีลอจิกเป็น 1 เสมอบัสของตระกูล MCS-48 ยังสามารถทำเป็นพอร์ตได้ โดยจะเป็นพอร์ตชนิด BI-DIRECTIONAL และยังสามารถ AND หรือ OR ได้โดยตรงเช่นเดียวกับพอร์ต 1 และ พอร์ต 2 การทำงานของบัส เมื่อทำหน้าที่เป็นพอร์ตจะต่างกับพอร์ต 1 และพอร์ต 2 ตรงที่ว่าการทำงานจะเป็นการอินพุตหรือเอาท์พุตอย่างใดอย่างหนึ่งเท่านั้น เมื่อใช้บัสเป็นอินพุต/เอาท์พุตพอร์ต จะทำหน้าที่เป็นอินพุตตลอดเวลา หลังทำการรีเซ็ตแล้ว และจะเป็นเอาท์พุตเมื่อใช้คำสั่ง OUTL BUS,A เมื่อใช้คำสั่งนี้แล้วจะไม่สามารถทำเป็นอินพุตได้จนกว่าจะใช้เป็นบัภายนอกหรือระบบถูกการรีเซ็ต นอกจากนี้บัสยังทำหน้าที่ติดต่อกับหน่วยความจำข้อมูลภายนอกก็ได้ โดยใช้คำสั่ง MOVX ซึ่งจะทำให้เกิดสัญญาณ RD หรือ WR ขึ้นกับหน้าที่ที่ใช้ เมื่อบัสไม่ได้ใช้งานจะอยู่ในสถานะ TRI-STATE อินพุตเอาท์พุตของตระกูล MS-48 สามารถขยายเพิ่มได้จาก 3 พอร์ตเป็น 7 พอร์ต โดยอาศัย 4 บิตแรก

ของพอร์ต 2 เพื่อทำหน้าที่ในการขยาย พอร์ตที่ขยายออกมานี้จะมีคำสั่งในการ AND หรือ OR แยกออกต่างหาก แต่การ AND หรือ OR ของพอร์ตที่เพิ่มขึ้นนี้ไม่สามารถ AND หรือ OR กับข้อมูลโดยตรง ต้องผ่านแอคคิวมูเลเตอร์เสมอคำสั่ง OUTL BUS,A จะใช้ได้ในกรณีที่ใช้ตระกูล MCS-48 ทำงานเพียงตัวเดียว และคำสั่งนี้สามารถทำงานกับคำสั่ง MOVX ได้ แต่ต้องระวังในการใช้คำสั่ง เพราะจะทำให้รูปแบบของพอร์ตที่ส่งออกไปก่อนจะสูญหายด้

7. คำสั่งเกี่ยวกับรีจิสเตอร์ (REGiSTER INSTRCTION)

ชุดคำสั่งเกี่ยวกับรีจิสเตอร์ใช้สำหรับทำการเพิ่มหรือลดค่าของรีจิสเตอร์ครั้งละหนึ่ง นอกจากนี้ยังสามารถเพิ่มขอมูลในหน่วยความจำข้อมูลภายในครั้งละหนึ่งโดยใช้รีจิสเตอร์ R0 หรือ R1 เป็นตัวชี้ได้อีกด้วย

8. คำสั่งเกี่ยวกับโปรแกรม (SUBROUTINE INSTRUCTION)\

ชุดคำสั่งของโปรแกรมย่อยใช้สำหับการเรียกและกลับจากโปรแกรมย่อย ในการเรียกโปรแกรมย่อย สามารถเรียกได้จากหน่วยความจำโปรแกรมชุดหนึ่งไปยังอีกชุดหนึ่งได้ โดยต้องใช้คำสั่งเลือกชุดหน่วยคามจำโปรแกรมก่อนใช้คำสั่ง CALL ในชุดคำสั่งนี้จะมีคำสั่งการกลับของโปรแกรมย่อยสองชนิดคือ ชนิดกลับโดยไม่เก็บสถานะกลับเข้ารีจิสเตอร์ของสถานะโปรแกรมและชนิดกลับโดยเก็บค่าสถานะรีจิสเตอร์อกสถานะโปรแกรม สำหรับการกลับชนิดหลังนี้ จะใช้สำหรับการกลับจากการอินเตอร์รัพต์

9. คำสั่งเกี่ยวกับแฟลก (FLAG INSTRUCTION)

ชุดคำสั่งเกี่ยวกับแฟลก จะให้ความสะดวกในการที่จะ COMPLEMENT หรือทำให้เป็นศูนย์สำหรับหน้าที่ของแฟลกมีดังนี้

- CARRY ใช้แสดงเมื่อเกิด OVERFLOW เมื่อผลการทำงานของแอคคิวมูเลเตอร์เกินจำนวนบิตของแอคคิวมูเลเตอร์

- AUXILLAR ใช้แสดงเมื่อเกิด OVERFLOW ขึ้นระหว่า 4 บิต ในกรณีทำงานคณิตศาสตร์แบบ BCD เมื่อใช้คำสั่งการปับให้เป็นเลข BCD

- F0 และ F1 เป็นแฟลกที่ใช้ทั่วไปสำหรับกระโดดแบบมีเงื่อนไข

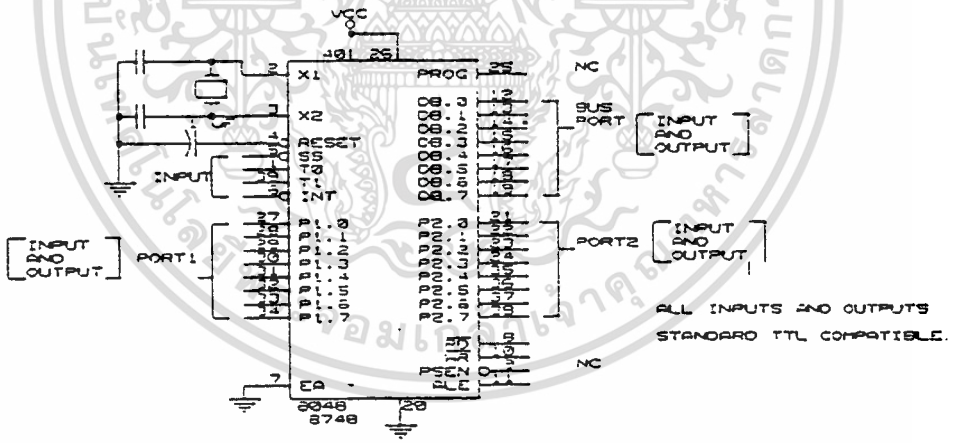
- CARRY และ F0 เป็นส่วนหนึ่งของรีจิสเตอร์บอกสถานะโปรแกรม และจะเก็บรักษาไว้ในสแตกเมื่อมีการเรียกโปรแกรมย่อย

10. คำสั่งเบ็ดเตล็ด (MISCELLANEOUS INSTRUCTION)

คำสั่งเบ็ดเตล็ดเป็นคำสั่งที่นอกเหนือจากที่กล่าวมาแล้วซึ่งได้แก่คำสั่งที่ให้ตระกูล MCS-48 จะไม่มีการทำงานเพียงข้ามไปทำคำสั่งถัดต่อไป

3.5 วงจรการใช้งานและการโปรแกรมตระกูล MCS-48

เนื่องจากตัวไมโครคอนโทรลเลอร์ตระกูล MCS-48 มีระบบวงจรของไมโครคอมพิวเตอร์ที่เกือบจะสมบูรณ์อยู่ในตัวแล้ว ดังนั้น 8040, 8749, 8048, 8748 ในตระกูล MCS-48 ก็จะใช้ตัวคริสตอลและตัวคาปาซิเตอร์ 2 ตัว สร้างสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์และตัวคาปาซิเตอร์ที่ต่อเข้าขารีเซ็ตสำหรับถ่วงเวลาสัญญาณการรีเซ็ตในการจุดให้ตัวไมโครคอนโทรลเลอร์เริ่มทำงานตามโปรแกรมได้ เมื่อมีการจ่ายไฟ 5 V เข้าตัวไมโครคอนโทรลเลอร์ให้ทำงานได้โดยอิสระดังแสดงการต่อวงจรอย่างง่าย ตามรูป 3.16 และสามารถที่จะใช้พอร์ตต่าง ๆ ของไมโครคอนโทรลเลอร์ตัวนี้ไปต่อพ่วงกับวงจรใช้งานต่าง ๆ ได้ตามต้องการจะสังเกตเห็นว่าขา EA จะต่อลงดิน เพื่อที่จะให้ทำงานตามโปรแกรมที่มีอยู่ภายในชิป

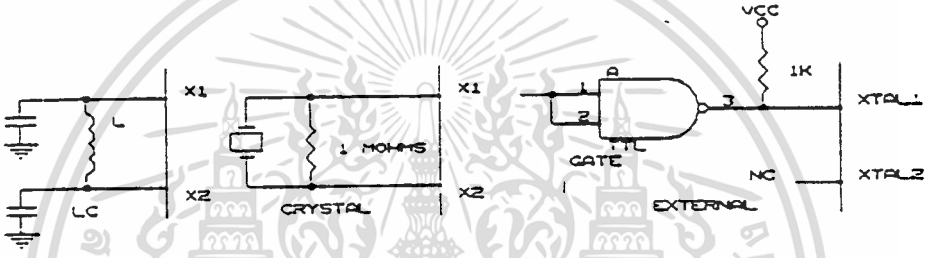


รูปที่ 3.16 การใช้ 8048/ 8049 /8749 /8050 ตัวเดียว

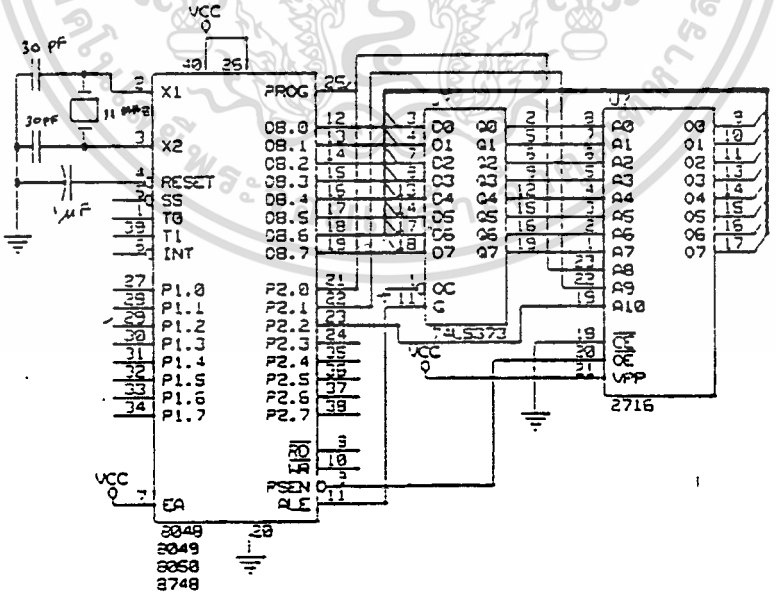
วงจรสำหรับสร้างสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์เบอร์นี้ นอกจากจะใช้คริสตอลแล้วยังสามารถที่จะใช้ตัวอินดัคเตอร์กับคาปาซิเตอร์เป็นตัวสร้างสัญญาณนาฬิกา

ได้อีกด้วย หรือจะใช้สัญญาณนาฬิกาจากภายนอกป้อนเข้าขา XTAL1 โดยผ่านบัฟเฟอร์เกต ดังรูปที่ 3.17

บ่อยครั้งที่การพัฒนาโปรแกรมเพื่อให้เครื่องต้นแบบที่ใช้ไมโครคอนโทรลเลอร์ทำงานได้ตามข้อกำหนดต่าง ๆ อย่างไม่ผิดพลาด จำเป็นจะต้องพัฒนาโปรแกรมด้วย การให้ตัวไมโครคอนโทรลเลอร์ตัวนี้ทำงานตามโปรแกรมที่เก็บอยู่ภายนอก ซึ่งส่วนใหญ่จะใช้ตัว EPROM แต่ระหว่างการพัฒนาอาจจะใช้ RAM BACK UP ที่มีแบตเตอรี่จ่ายไฟสำรองหรือ EPROM EMULATOR ก็ได้ ซึ่งจะเพิ่มความสะดวกในการอ่านและเขียนแก้ไขโปรแกรม ลักษณะการต่อวงจรที่ให้ทำงานตามโปรแกรมที่อยู่ภายนอก แสดงในรูปที่ 3.18

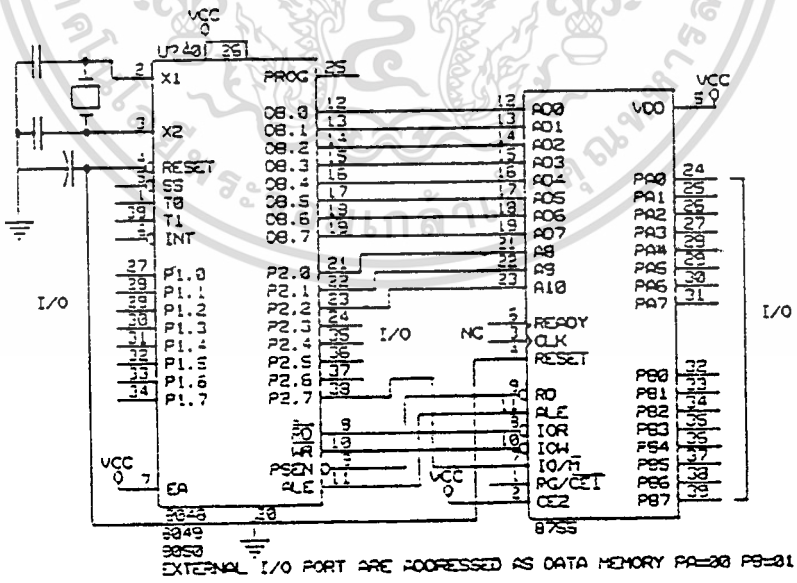


รูปที่ 3.17 วงจรการต่อตัวสร้างสัญญาณนาฬิกาของตระกูล MCS-48



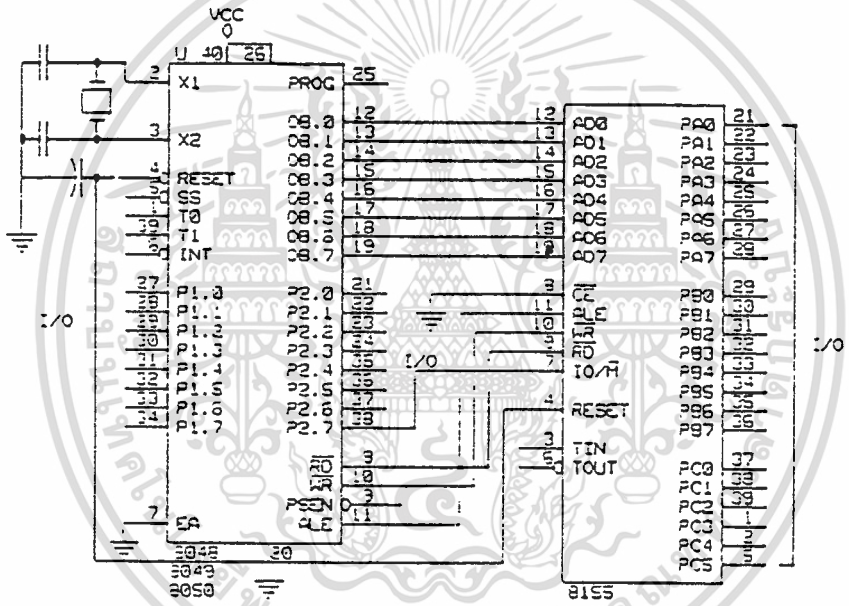
รูปที่ 3.18 วงจรไมโครคอนโทรลเลอร์ที่ใช้ EPROM ภายนอก

โดยที่วงจรจะบังคับให้ขา EA มีสถานะสูง เพื่อเป็นการบอกให้ไมโครคอนโทรลเลอร์ทำงานโปรแกรมภายนอกซึ่งอยู่ใน EPROM ขนาด 2K X 8 ไบต์ ตัวแลทช์ 74LS373 จะทำการแลทช์ข้อมูลแอดเดรสไว้เนื่องจากขา DB0-DB7 เป็นแบบมัลติเพล็กข้อมูลและแอดเดรส ดังนั้นจึงต้องแลทช์แยกเอาค่าแอดเดรสมาแสดงที่ขาเอาต์พุตของตัว 74LS373 ไปถอดรหัสโปรแกรมในตัว EPROM (2716) ขนาด 2K X 8 ไบต์ เพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานตามโปรแกรมภายนอกได้อย่างถูกต้อง ข้อสังเกตุเมื่อเราใช้โปรแกรมจากภายนอกไมโครจะทำให้ตัวไมโครมีขาพอร์ตที่ลดน้อยไปประมาณ 12 ขา คือ ขา DB0-DB7 และขา P20-P23 เหลือพอร์ต P10-P17,P24-P27 ส่วนขา T0-T1,INT จะเป็นตัวตรวจสอบอินพุตอย่างเดียวรูปที่ 3.19 เป็นวิธีการต่อวงจรของชิปที่รวมอาพอร์ตที่ต้องการขยายเพิ่มอีก 2 พอร์ตรวมกับโปรแกรม ROM/EPROM พายนอก แล้วแต่เบอร์ที่ใช้ 8355/8755 มีขา AD0-AD7 ที่ควบคุมด้วยขา ALE และ RD ในการจัดการกับสัญญามัลติเพล็กของแอดเดรสที่ต่อโดยตรงโดยตรงได้กับตัไมโครคอนโทรลเลอร์ MCS-48 เพื่อให้โปรแกรมได้สูงสุด 2K X 8 ส่วนพอร์ตที่ขยายเพิ่มอีก 2 พอร์ตคือ PA0-PA7 และ PB0-PB7 ก็จะถูกควบคุมให้เป็นพอร์ตอินพุตหรือเอาต์พุตได้ด้วยขา ICR และ IOW และกำหนดแอดเดรสของพอร์ต A ดวยการส่งค่าแอดเดรสแบบหน่วยความจำข้อมูลค่า 00 ถ้าเลือกพอร์ต B ก็จะมีค่าเป็น 01



รูปที่ 3.19 การต่อวงจรที่ใช้ชิปพอร์ตและโปรแกรมภายนอกเข้าด้วยกัน

รูปที่ 3.20 เป็นวิธีการต่อวงจรที่ต้องการขยายพอร์ตเพิ่ม 3 พอร์ตและเพิ่มขนาดหน่วยความจำข้อมูลภายนอกได้ 256 X 8 ไบต์ และเพิ่มตัวจับเวลา/ตัวนับ ขนาด 16 บิต ซึ่งรวมอยู่ในชิปเบอร์ 8155 การต่อพ่วงของ 8155 เข้ากับ MCS-48 ก็มีวิธีการเช่นเดียวกับรูปที่ 3.19 คือ AD0-AD7 ต่อโดยตรงเข้ากับ DB0-DB7 ของ MCS-48 การเลือกใช้พอร์ตหรือหน่วยความจำ ส่วนขา RD,WR ใช้ในการอ่าน เขียนข้อมูลเข้าที่หน่วยความจำหรือพอร์ตอินพุต เอาต์พุต ด้วยการเลือกบังคับขา IO/M ด้วยขา P27 ของ MCS-48 ว่าเป็นสูงหรือต่ำ ถ้าเป็นสถานะต่ำก็จะเป็นการเลือกข้อมูลเข้าถึงหน่วยความจำขนาด 256 X 8 ไบต์ของ 8155 การใช้ตัวจับเวลา/ตัวนับ หรือการบังคับพอร์ตต่าง ๆ ทั้งสามให้เป็นอินพุตจะขึ้นอยู่กับการเริ่มต้นโปรแกรมให้ทำงานตามที่กำหนดได้ชิป

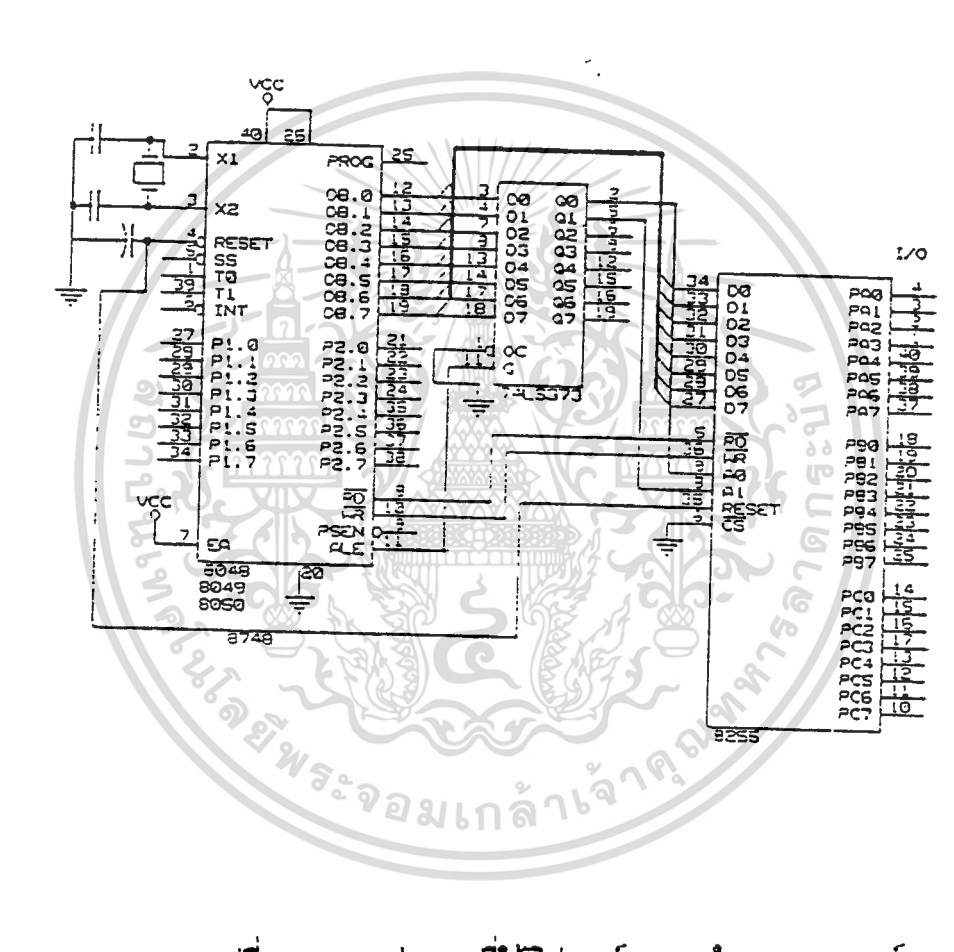


รูปที่ 3.20 การต่อวงจรที่ใช้ชิปเบอร์ 8155

เบอร์ 8255 ชิปที่ทำงานเป็นพอร์ตได้ 3 พอร์ต โดยที่ไม่มี ROM หรือ RAM ในตัวชิปนี้ ซึ่งเป็นชิปที่นิยมใช้กันและราคาถูกจึงได้แสดงวิธีการต่อวงจรการต่อพ่วงกับตัว MCS-48 ดังรูปที่ 3.21

เนื่องจากขาแอดเดรสของ 8255 คือ A0 และ A1 ไม่สามารถจัดการกับการมัลติเพล็กแอดเดรสกับข้อมูลเองได้ จึงต้องต่อกับขาเอาต์พุตของตัวแลทช์ 74LS373 ที่ใช้ใน

การแอดเดรสข้อมูลโปรแกรม EPROM ภายนอกให้ได้ค่าการถอดรหัสแอดเดรส A0,A1 ได้ 4 ตำแหน่ง เพื่อใช้ในการส่งข้อมูลออกและเข้าพอร์ตทั้งสามของ 8255 และอีกหนึ่งแอดเดรสที่เหลือจะใช้ในการจัดการส่งคำควบคุมลักษณะการใช้งานในการเริ่มต้นโปรแกรมเข้าที่ชิป 8255 นี้



รูปที่ 3.21 การต่อวงจรที่ใช้ชิปเบอร์ 8255 ในการขยายพอร์ต

บทที่ 4
รายละเอียดของโครงงานและการออกแบบ

จากรูปที่ 1.1 เป็นบล็อกไดอะแกรมที่แสดงโครงสร้างโดยรวมทั้งหมดของโครงงาน จะเห็นว่าประกอบด้วยส่วนสำคัญ 3 ส่วน คือ ไมโครคอนโทรลเลอร์ ทำหน้าที่ควบคุมการทำงานของระบบถัดมาเป็นวงจรนับ (Driver) ซึ่งมี 3 วงจร แบ่งเป็นวงจรขับแนวแกน X วงจรขับแนวแกน Y และวงจรขับแนวแกน Z ซึ่งวงจรทั้งสามนี้มีลักษณะเหมือนกัน วงจรขับแต่ละวงจรจะต่ออยู่กับสเต็ปปิ้งมอเตอร์แต่ละตัว ซึ่งจะถูกนำไปใช้ในการควบคุมการเคลื่อนที่ในสามทิศทาง

ในบทนี้จะได้กล่าวถึงรายละเอียดของวงจรในแต่ละส่วน วิธีการออกแบบ และวิธีการทำงาน พร้อมกันนั้นยังมีการทดลองเพื่อตรวจสอบว่าโครงงานที่สร้างขึ้นถูกต้องตามทฤษฎีหรือไม่

รูปที่ 4.1 สัญ ลักษณ์ ของสเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2 เฟส

4.1 สเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2เฟส

ในบทที่ 2 ได้กล่าวถึง ทฤษฎีพื้นฐานและหลักการทำงานของสเต็ปปิ้งมอเตอร์มาบ้างแล้ว แต่เป็นรายละเอียดของสเต็ปปิ้งมอเตอร์แบบยูนิโพลาร์ ซึ่งกระแสที่ไหลที่ขดลวดบนสเตเตอร์จะมีทิศทางเดียว หรือขั้วแม่เหล็กที่เกิดขึ้นเป็นขั้วแม่เหล็กเพียงขั้วเดียวเสมอ ข้อ

เสียของสเต็ปปิ้งมอเตอร์แบบยูนิโพลาร์ ก็คือ การเคลื่อนที่ในแต่ละสเต็ปจะอาศัยอำนาจแม่เหล็กจากแม่เหล็กเพียงชุดเดียวทำให้มีกำลังต่ำ

ในโครงงานที่สร้างขึ้นนี้ได้เลือกใช้สเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2 เฟส ซึ่งเป็นที่นิยมใช้กันอย่างแพร่หลาย ข้อดีของสเต็ปปิ้งมอเตอร์แบบนี้ คือ ขดลวดบนขั้วแม่เหล็กที่สเตเตอร์จะมีกระแสไหลอยู่ตลอดเวลา แต่มีทิศทางเปลี่ยนแปลงไปในแต่ละสเต็ป กล่าวคือขั้วแม่เหล็กเป็นได้ทั้งขั้วเหนือและขั้วใต้

จากรูปที่ 4.1 สเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2 เฟส จะมีขดลวดทั้งหมด 2 ชุด แต่ละชุดหมายถึงเฟส ขดลวดชุดที่ 1 เรียกว่าเฟส A ชุดที่สองเรียกว่าเฟส B ที่ขดลวดแต่ละชุดจะมีขั้วที่ใช้ต่อกับวงจรขับ 2 ขั้ว คือ A กับ B ในขณะที่มอเตอร์หมุน จะมีกระแสไหลจาก A ไป B หรือ B ไป A ทำให้เกิดการเปลี่ยนแปลงขั้วแม่เหล็กจาก N ไป S หรือจาก S ไป N

การป้อนกระแสเข้าขดลวดเพื่อให้เกิดการหมุนมีกฎเกณฑ์ตามตารางที่ 4.1 ถ้าต้องการให้มอเตอร์หมุนย้อนทิศก็ทำได้โดยการป้อนกระแสวนกลับจากสเต็ปที่ 4 ไป 1

ตารางที่ 4.1 การป้อนกระแสให้สเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2 เฟส

สเต็ป	กระแสที่เฟส A	กระแสที่เฟส B
1	B ไป A	B ไป A
2	A ไป B	B ไป A
3	A ไป B	A ไป B
4	B ไป A	A ไป B
1	B ไป A	B ไป A

4.2 การขับสเต็ปปิ้งมอเตอร์ให้หมุนแบบครึ่งสเต็ป (Half Step)

การป้อนกระแสตามทิศทางที่แสดงในตารางที่ 4.1 เป็นการขับสเต็ปปิ้งมอเตอร์แบบเต็มสเต็ป (Full Step) จำนวนสเต็ปสูงสุดต่อรอบจะมีค่าเท่ากับจำนวนสเต็ปของ

มอเตอร์ เช่น 200 สเต็ป องศาต่อเสต็ปจะมีค่าเท่ากับ 1.8 องศา การเคลื่อนที่แบบเต็มสเต็ปมีข้อจำกัดคือจำนวนสเต็ปมีไม่มากนัก ทำให้มีความแม่นยำน้อยและอาจเกิดการคลาดเคลื่อนเนื่องจากโครงสร้างของตัวมอเตอร์ ทำให้แต่ละสเต็ปมีองศาไม่เท่ากันนอกจากนั้นยังเกิดการสั่นในขณะเคลื่อนที่ จากสาเหตุดังกล่าว ทำให้เกิดเทคนิคที่ช่วยให้สเต็ปปิ้งมอเตอร์มีจำนวนสเต็ปต่อรอบสูงขึ้นเป็นสองเท่า โดยแบ่ง 1 สเต็ปให้กลายเป็น 2 สเต็ป เรียกว่าแบบครึ่งสเต็ป การเคลื่อนที่ของมอเตอร์มีความราบเรียบมากขึ้น และการเพิ่มจำนวนสเต็ปทำให้ความละเอียดมากขึ้น ส่งผลให้เกิดความแม่นยำและมีความคลาดเคลื่อนน้อยลง การป้อนกระแสแบบครึ่งสเต็ปแสดงดังตารางที่ 4.2

ตารางที่ 4.2 การป้อนกระแสให้เสต็ปปิ้งมอเตอร์เพื่อให้หมุนแบบครึ่งเสต็ป

สเต็ป	กระแสที่เฟส A	กระแสที่เฟส B
1	B ไป A	B ไป A
2	หยุดไหล	B ไป A
3	A ไป B	B ไป A
4	A ไป B	หยุดไหล
5	A ไป B	A ไป B
6	หยุดไหล	A ไป B
7	B ไป A	A ไป B
8	B ไป A	หยุดไหล
1	B ไป A	B ไป A

สังเกตว่าการป้อนกระแสเพื่อให้หมุนแบบครึ่งสเต็ป จะให้กระแสหยุดไหลในบางจังหวะและในบางเฟส

4.3 วงจรขับ (Driver)

ในระบบทั้งหมดเมื่อวงจรขับ 3 ชุด แต่ละชุดมีลักษณะเหมือนกันจึงขออธิบายเพียงชุดเดียว วงจรสมบูรณ์แสดงดังรูปที่ 4.2

จากรูปได้เลือกใช้ไอซีเบอร์ TEA 3718 ซึ่งเป็นไอซีที่ถูกออกแบบมาสำหรับเป็นวงจรขับสเต็ปปิ้งมอเตอร์โดยเฉพาะ ทำให้วงจรไม่ยุ่งยาก ง่ายต่อการสร้างเพราะมีอุปกรณ์ภายนอกเพียงไม่กี่ตัว ในการขับสเต็ปปิ้งมอเตอร์ 1 ตัวจะต้องใช้ไอซี 2 ตัว แต่ละตัวใช้ป้อนกระแสให้ขดลวดแต่ละชุดของมอเตอร์

ขาต่าง ๆที่สำคัญของไอซี TEA 3718 มีดังต่อไปนี้

PHASE เป็นขาที่ใช้ในการเลือกทิศทางการไหลของกระแสที่ออกจากไอซีทาง OUT A และ OUT B โดยมีเงื่อนไขดังนี้ คือ

- ถ้าป้อนลอจิก 0 กระแสจะไหลจาก B ไป A
- ถ้าป้อนลอจิก 1 กระแสจะไหลจาก A ไป B

IN0 และ IN1 เป็นขาที่ใช้กำหนดกระแสที่ไหลออกไปยังขดลวด

ในกรณีของการขับแบบเต็มสเต็ปและครึ่งสเต็ป ปริมาณกระแสที่ใช้มี 2 ค่า คือ สูงสุด กับ หยุดไหล ลอจิกมีเงื่อนไขดังนี้

ถ้าป้อน 00 กระแสไหลสูงสุด

ถ้าป้อน 11 กระแสหยุดไหล

Vss เป็นขาที่ใช้รับไฟเลี้ยงวงจร 5 โวลต์

REF เป็นขาที่ใช้ต่อกับแรงดันอ้างอิงจากภายนอก เพื่อใช้เป็น Vref ของวงจร เปรียบเทียบแรงดัน (Voltage Comparator)

GND ใช้ต่อลงกราวด์

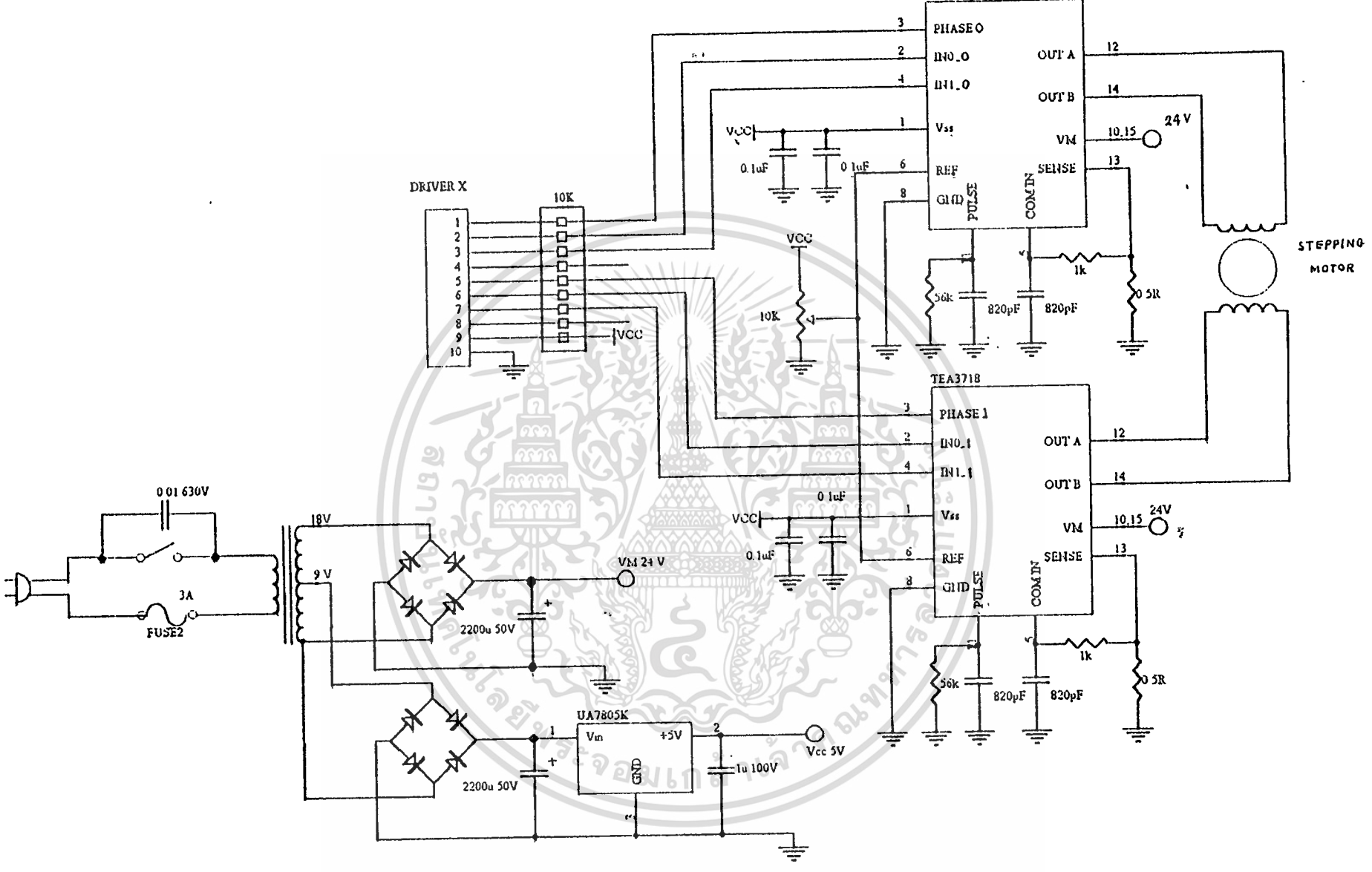
PULSE เป็นขาที่ใช้ต่อกับอุปกรณ์เพื่อสร้างพัลส์สำหรับวงจร PWM (Pulse Width Modulation)

COM IN เป็นขาคอมมอนของวงจรเปรียบเทียบแรงดัน

SENSE เป็นขาคอมมอนของวงจรขับแบบไฮบริดจ์ภายในตัวไอซี

VM เป็นขาสำหรับป้อนไฟเลี้ยงวงจรขับ 24 โวลท์

OUT A และ OUT B เป็นขาที่ต่อกับขดลวดของมอเตอร์เพื่อป้อนกระแส



รูปที่ 4.2 วงจรขับและแหล่งจ่ายไฟ

จากตารางที่ 4.2 และ 4.2 ทำให้เราทราบว่าควรจะป้อนลอจิกที่ขา PHASE 0 , PHASE 1, IN0.0 , IN0.1 , IN1.0 และ IN1.1 อย่างไร ซึ่งค่าต่าง ๆ เป็นไปตามตารางที่ 4.3 และ 4.4

ตารางที่ 4.3 ลอจิกที่ขาต่าง ๆของ IC สำหรับการหมุนแบบเต็มสเต็ป

สเต็ป	PHASE 0	PHASE 1	IN0.0 ,IN1.0	IN0.1 , IN1.1
1	0	0	0,0	0,0
2	1	0	0,0	0,0
3	1	1	0,0	0,0
4	0	1	0,0	0,0

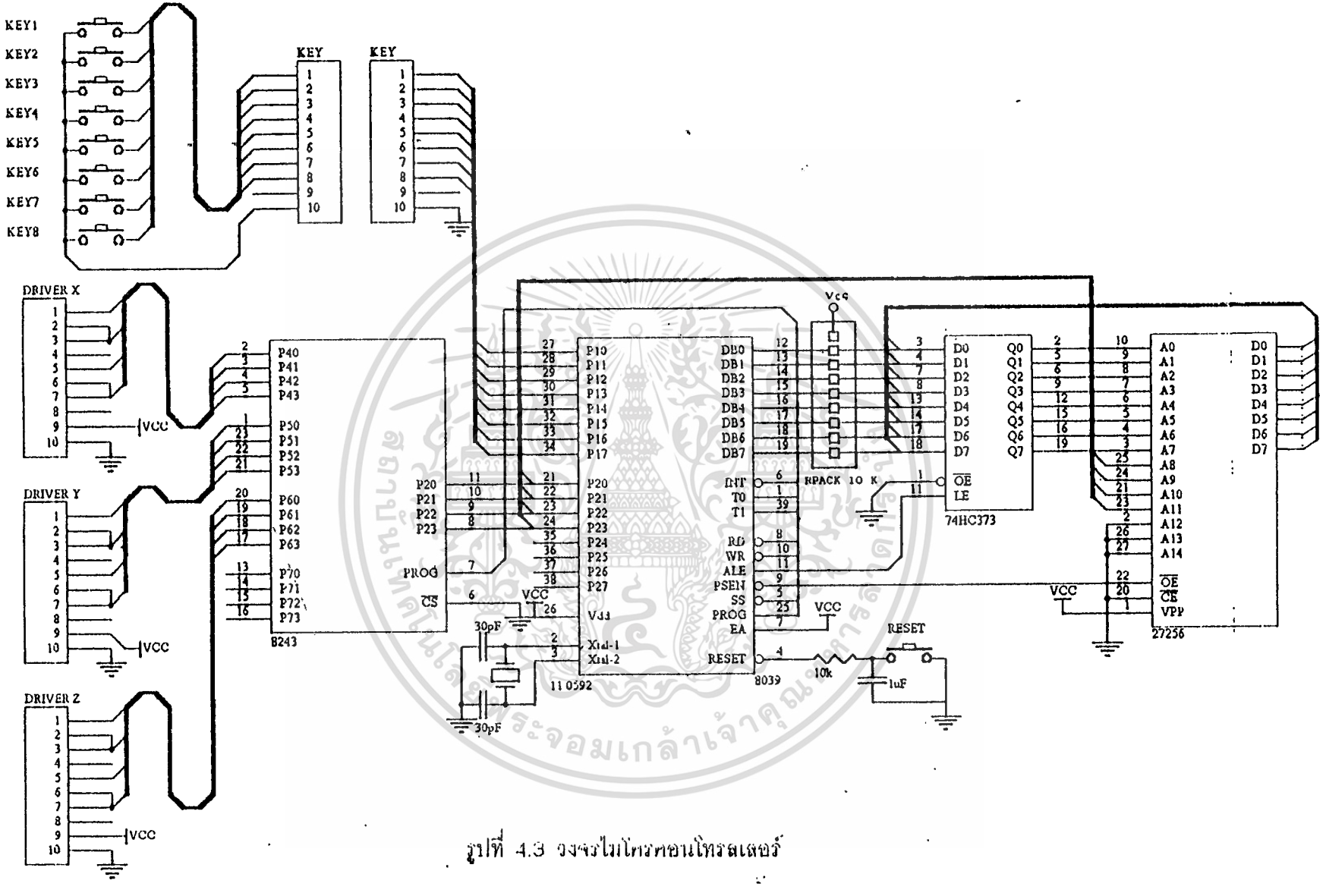
ตารางที่ 4.3 ลอจิกที่ขาต่าง ๆ ของ IC สำหรับการเคลื่อนที่ของครึ่งสเต็ป

สเต็ป	PHASE 0	PHASE 1	IN0.0 , IN1.0	IN0.1 , IN1.1
1	0	0	0,0	0,0
2	0	0	1,1	0,0
3	1	0	0,0	0,0
4	1	0	0,0	1,1
5	1	1	0,0	0,0
6	1	1	1,1	0,0
7	0	1	0,0	0,0
8	0	1	0,0	1,1

4.4 วงจรไมโครคอนโทรลเลอร์

ระบบควบคุมโดยใช้ไมโครคอนโทรลเลอร์ ดังแสดงดังรูปที่ 4.3ไมโครคอนโทรลเลอร์ที่เลือกใช้คือเบอร์ 8039 อยู่ในตระกูล MCS- 48 ซึ่งรายละเอียดได้อธิบายในบทที่ 3 แล้ว ในที่นี้จะพูดถึงการเชื่อมต่อกับอุปกรณ์อื่น ๆ คือ หน่วยความจำโปรแกรมและพอร์ต

ไมโครคอนโทรลเลอร์เบอร์ 8039 ภายในมีหน่วยความจำข้อมูล (RAM) ขนาด 128 ไบต์ แต่ไม่มีหน่วยความจำโปรแกรมจึงจำเป็นต้องต่อเพิ่มภายนอกโดยในโครงงานนี้เลือกใช้ EPROM เบอร์ 27256 ในขณะเดียวกันขา EA ของ 8039 ก็ต้องต่อไว้กับ Vcc เพื่อแสดงถึงการใช้หน่วยความจำโปรแกรมภายนอก ที่พอร์ต DB0-DB7 ของ 8039 จะต้องทำการมัลติเพล็กระหว่างบัสแอดเดรส กับบัสข้อมูล จึงต้องต่อไอซีแลทช์เบอร์ 74HCT373 ไว้ .พอร์ต P1 ต่ออยู่กับ คีย์ 1-8 ส่วนพอร์ต P2 ไบต์ต่ำ (P20-P23) ทำหน้าที่เป็นบัสแอดเดรส และต่ออยู่กับพอร์ตเบอร์ 8243 เพื่อขยายจำนวนพอร์ตให้เพิ่มขึ้นอีก 4 พอร์ต คือ พอร์ต 4 - พอร์ต 7 โดยที่ พอร์ต 4 , พอร์ต 5 และ พอร์ต 6 จะต่อกับอินพุทของไดรเวอร์ X , ไดรเวอร์ Y , และ ไดรเวอร์ Z ตามลำดับ



รูปที่ 4.3 วงจรไมโครคอนโทรลเลอร์

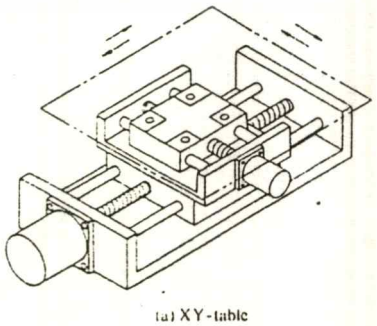
4.5 การนำระบบควบคุมสเต็ปปิ้งมอเตอร์ 3 แนวแกนมาประยุกต์ใช้งาน

นอกจากการศึกษาระบบการควบคุมสเต็ปปิ้งมอเตอร์ และในโครงงานชิ้นนี้ได้ทดลองสร้างส่วนกลไกที่สามารถใช้งานในเครื่องเจาะ 3 แนวแกน เพื่อสาธิตให้เห็นถึงการนำระบบควบคุมดังกล่าวไปใช้งาน

การเชื่อมต่อสเต็ปปิ้งมอเตอร์เข้ากับโหลดเพื่อเปลี่ยนการเคลื่อนที่เชิงมุมให้กลายเป็นการเคลื่อนที่ในแนวระนาบสามารถทำได้หลายวิธี ในที่นี้เลือกใช้วิธีการเชื่อมต่อโดยใช้สกรูว์ดังที่แสดงในรูปที่ 4.4

รูปที่ 4.4 แสดงการเชื่อมสเต็ปปิ้งมอเตอร์ด้วยสกรูว์

ที่ระนาบจะมีน็อตที่สวมเข้ากับสกรูว์ยึดไว้ เมื่อเกิดการหมุนระนาบจะเคลื่อนที่ การออกแบบกลไกให้ครอบคลุมพื้นที่ใช้งานได้ทั้งหมดทำได้ โดยการสร้างกลไกของระนาบ Y ซ้อนกับระนาบ X ดังรูปที่ 4.5

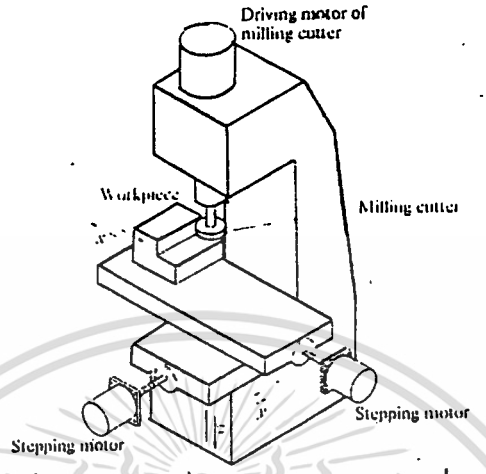


(a) XY-table

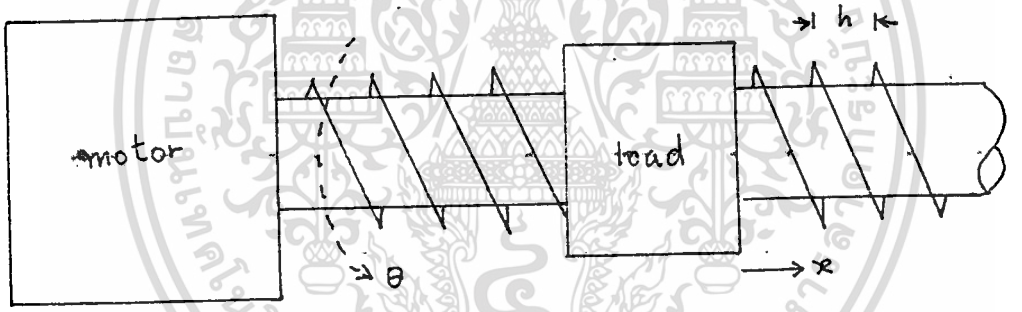
รูปที่ 4.5 แสดงกลไกการเคลื่อนที่ในแนว X และ Y

แท่งก

ส่วนในแนวแกน Z จะทำหน้าที่เคลื่อนดอกสว่านเพื่อเจาะรูและถูกออกแบบให้ยึดแผ่นอยู่กับที่รูปร่างของเครื่องเจาะ 3 แนวแกนโดยสมบูรณ์เป็นดังรูปที่ 4.6



รูปที่ 4.6 ภาพเค้รื่องเจาะสามแนวแกนที่สร้างขึ้น



รูปที่ 4.7 การหาความละเอียดของตำแหน่ง

$$\theta / 2\pi = x / h$$

การหาความละเอียดของตำแหน่ง

มอเตอร์หมุน 1 รอบ (200 สเต็ป) โหลดเคลื่อนที่เป็นระยะ h ถ้ามอเตอร์หมุน 1 สเต็ป โหลดเคลื่อนที่เป็นระยะ h / 200 เมื่อออกแบบให้ h = 1.43 มิลลิเมตร ความละเอียดจะเท่ากับ 7.15 ไมโครเมตร

แทรก

คุณสมบัติของเครื่องเจาะ

- ความละเอียดของตำแหน่ง 7.15 ไมโครเมตร
- ขนาดของแท่น 45 X 60 เซนติเมตร
- พื้นที่ทำงานไม่เกิน 15 X 15 เซนติเมตร
- ระยะเคลื่อนที่ของสว่าน 2 เซนติเมตร
- ตัวขับเคลื่อน สเต็ปปิ้งมอเตอร์ไฮบริดจ์ 2 เฟส
- การหมุนแบบเต็มสเต็ป
- ระบบควบคุมใช้ไมโครคอนโทรลเลอร์ 8039
- การควบคุมเป็นแบบ manual

บทที่ 5
การทดลอง

จุดประสงค์

1. เพื่อทดลองเขียนโปรแกรมด้วยภาษาแอสเซมบลีของ MCS-48 เพื่อควบคุมสเต็ปปิ้งมอเตอร์

2. ทดสอบสัญญาณที่ตำแหน่งต่าง ๆ ที่สำคัญ เพื่อตรวจสอบดูว่าถูกต้องตามทฤษฎีหรือไม่

3. เพื่อเปรียบเทียบสัญญาณที่จุดต่าง ๆ ระหว่างการขับสเต็ปปิ้งมอเตอร์แบบเต็มสเต็ปกับแบบครึ่งสเต็ป

อุปกรณ์

1. วงจรไมโครคอนโทรลเลอร์
2. วงจรขับ 1 ชุด
3. สเต็ปปิ้งมอเตอร์ 1 ตัว
4. แหล่งจ่ายไฟ
5. ออสซิลโลสโคป

การทดลองที่ 1 การหมุนแบบเต็มสเต็ป (Full Step)

วิธีการทดลอง

1. เขียนโปรแกรมขับสเต็ปปิ้งมอเตอร์แบบเต็มสเต็ปตามตัวอย่างในโปรแกรมเอดิเตอร์ ใช้โปรแกรมแอสเซมเบลอร์ เพื่อเปลี่ยนรูปแบบของโปรแกรมให้เป็นภาษาแอสเซมบลี และเปลี่ยนภาษาแอสเซมบลีให้อยู่ในรูป Hex File เพื่อโหลดลงอีพรอมอีมูเลเตอร์

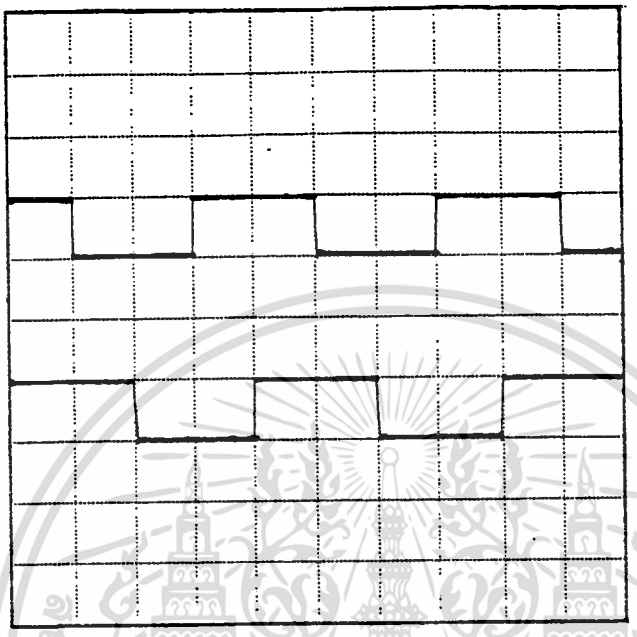
2. ต่อสเต็ปปิ้งมอเตอร์เข้ากับวงจรขับ และต่อวงจรขับเข้ากับวงจรไมโครคอนโทรลเลอร์จากนั้นป้อนไฟที่จุดต่าง ๆ

3. ใช้ออสซิลโลสโคปวัดแรงดันที่จุดต่าง ๆ ดังต่อไปนี้ พร้อมกับบันทึกรูปร่างของสัญญาณ เพียงแต่เปลี่ยนมาใช้โปรแกรมตัวอย่างสำหรับการหมุนแบบครึ่งสเต็ป

```
;*****************************************************************

        ;THIS IS AN EXAMPLE PROGRAM FOR FULL STEP
        ;STEPPING MOTOR DRIVE
        CPU    "8048.TBL"      ;CPU CODE
        HOF    "INT8"          ;HEX FORMAT
;*****************************************************************

               ORG    000H
        BEGIN: MOV    A,#00H ;OUTPUT FOR 1'ST STEP
               MOVD   P4,A
               CALL   DELAY
        ;
               MOV    A,#01H ;OUTPUT FOR 2'ND STEP
               MOVD   P4,A
               CALL   DELAY
        ;
               MOV    A,#05H ;OUTPUT FOR 3'RD STEP
               MOVD   P4,A
               CALL   DELAY
        ;
               MOV    A,#04H ;OUTPUT FOR 4'ST STEP
               MOVD   P4,A
               CALL   DELAY
        ;
               JMP    BEGIN
```

```
;*****************************************************************

        ;ROUTINE FOR DELAY

;*****************************************************************

DELAY: MOV   R7,#03H

LOOP1: MOV   R6,#0FEH

LOOP2: DJNZ  R6,LOOP2

       DJNZ  R7,LOOP1

       RETR

;*****************************************************************

       END
```

รูปที่ 5.1 รูปสัญญาณที่จุด Phase 0 **และ** Phase 1



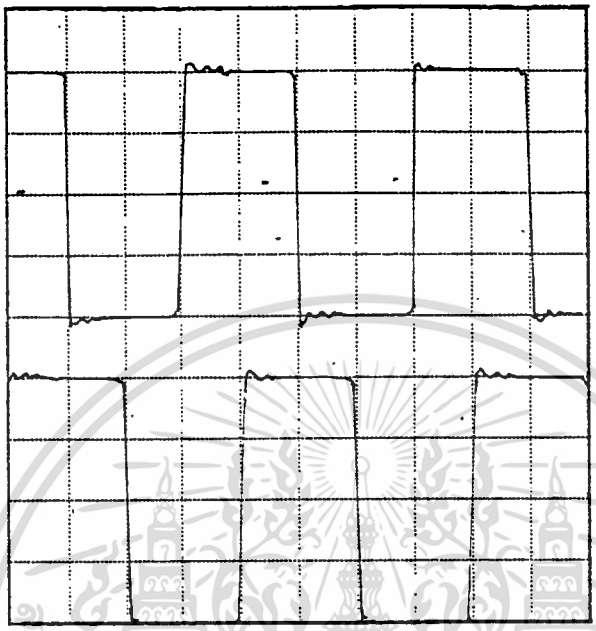
Voit / div..... 5 Volt / div .....

Time / div..... 5 Volt / div .....

CH 1 **แรงดันที่จุด** Phase 0 ..... 5 Vpp .....

CH 2 **แรงดันที่จุด** Phase 1 ..... 5 Vpp .....

รูปที่ 5.2 รูปสัญญาณที่จุด OUTA (Phase 0) และ OUTA (Phase 1)



Volt / div..... 10 Volt / div .....

Time / div..... 5 msec / div .....

CH 1 แรงดันที่จุด OUT A (Phase 0)..... 40 V_{pp} .....

CH2 แรงดันที่จุด OUT A (Phase 1)..... 40 V_{pp} .....

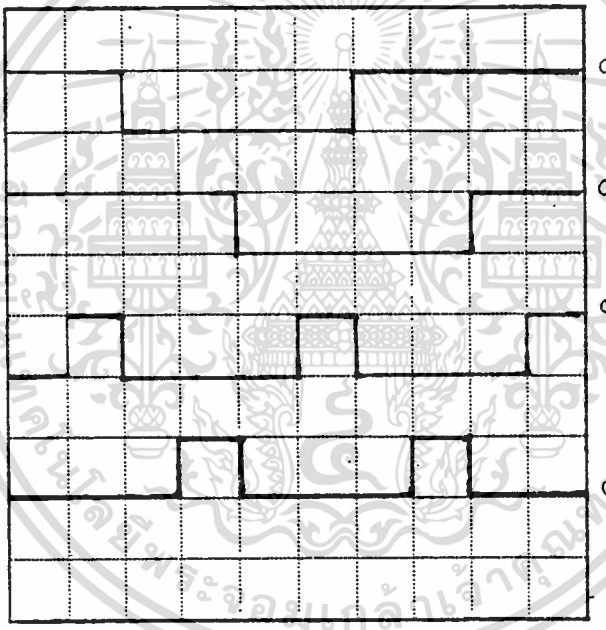
การทดลองตอนที่ 2 การหมุนแบบครึ่งสเต็ป (Half Step)

วิธีการทดลอง

1. วิธีทดลองเช่นเดียวกับตอนที่ 1 เพียงแต่เปลี่ยนโปรแกรมเป็นโปรแกรมตัวอย่างสำหรับการหมุนแบบครึ่งสเตป

2. ใช้ออสซิลโลสโคปวัดแรงดันที่จุดต่าง ๆ ดังต่อไปนี้พร้อมกับบันทึกรูปร่างของสัญญาณ

รูปที่ 5.3 รูปสัญญาณที่จุด Phase 0, Phase 1, IN0.0 และIN0.1



Volt / div.....5 Volt / div.....

Time / div.....5 msec / div.....

CH 1 Phase 0.....5 V_{PP}.....

CH 2 Phase 1.....5 V_{PP}.....

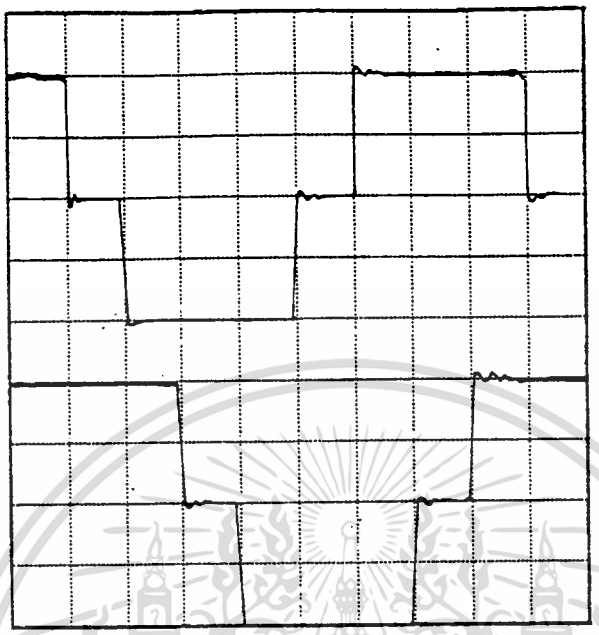
CH1 / IN0.0.....5 V_{PP}.....

CH2 / IN0.1.....5 V_{PP}.....

```
;*****************************************************************
;THIS IS AN EXAMPLE PROGRAM FOR  HALF STEP
;STEPPING MOTOR DRIVE
CPU    "8048.TBL"      ;CPU CODE
HOF    "INT8"          ;HEX FORMAT
;*****************************************************************
       ORG    000H
START: MOV    A,#00H ;OUTPUT FOR 1'ST STEP
       MOVD   P4,A
       CALL   DELAY
;
       MOV    A,#08H ;OUTPUT FOR 2'ND STEP
       MOVD   P4,A
       CALL   DELAY
;
       MOV    A,#04H ;OUTPUT FOR 3'RD STEP
       MOVD   P4,A
       CALL   DELAY
;
       MOV    A,#06H ;OUTPUT FOR 4'TH STEP
       MOVD   P4,A
       CALL   DELAY
;
       MOV    A,#05H ;OUTPUT FOR 5'TH STEP
       MOVD   P4,A
       CALL   DELAY
;
```

```
        MOV   A,#0DH;OUTPUT FOR 6'TH STEP
        MOVD  P4,A
        CALL  DELAY
;
        MOV   A,#01H ;OUTPUT FOR 7'TH STEP
        MOVD  P4,A
        CALL  DELAY
;
        MOV   A,#03H ;OUTPUT FOR 4'TH STEP
        MOVD  P4,A
        CALL  DELAY
;
        JMP   START
;*****************************************************************
    ;ROUTINE FOR DELAY
;*****************************************************************
    DELAY: MOV   R7,#03H
    LOOP1: MOV   R6,#0FEH
    LOOP2: DJNZ  R6,LOOP2
           DJNZ  R7,LOOP1
           RETR
;*****************************************************************
           END
```

รูปที่ 5.4 รูปสัญญาณที่จุด OUT A (Phase 0) และ OUTA (Phasse1)



Volt / div..... 10 Volt / div .....

Time / div..... 5 msec / div .....

CH 1 OUT A (Phase0) ..... 40 Vpp .....

CH 2 OUT A (Phase1)..... 40 Vpp .....

สรุปผลการทดลอง

1. จากโปรแกรมตัวอย่างทั้งสองโปรแกรม คือโปรแกรมการขับแบบเต็มสเต็ป และแบบครึ่งสเต็ป ทำให้เข้าใจในการเขียนโปรแกรมเพื่อควบคุมการทำงาน และสามารถนำไปเป็นแนวทางเพื่อพัฒนาโปรแกรมที่มีความซับซ้อนขึ้นได้

2. สัญญาณที่ปรากฏ ณ จุดต่าง ๆ ถูกต้องตามทฤษฎี

บทที่ 6
บทสรุป

สเต็ปปิ้งมอเตอร์ เป็นมอเตอร์ที่มีความสามารถในการควบคุมตำแหน่งและความเร็วได้แม่นยำ เพราะฉะนั้นปัจจุบันนี้จึงมีผู้นิยมหันมาใช้สเต็ปปิ้งมอเตอร์ในโครงงานต่าง ๆ ที่ต้องใช้มอเตอร์เป็นส่วนประกอบมากขึ้นแทนการใช้เซอร์โวมอเตอร์

โครงงานที่สร้างขึ้น คือ การสร้างระบบควบคุมสเต็ปปิ้งมอเตอร์ 3 แนวแกน เพื่อใช้ประโยชน์ในการขับเคลื่อน 3 ทิศทาง สามารถนำไปใช้เป็นส่วนประกอบของเครื่องมือได้หลายชนิด เช่น เครื่องเจาะ เป็นต้น

โครงสร้างหลักของโครงงานแบ่งเป็น 3 ส่วน คือ

1. ส่วนควบคุมที่ใช้ไมโครคอนโทรลเลอร์
2. ส่วนวงจรขับ 3 ชุด
3. สเต็ปปิ้งมอเตอร์แบบไบโพลาร์ 2 เฟส 3 ตัว

1. ส่วนควบคุมเลือกใช้ไมโครคอนโทรลเลอร์เบอร์ 8039 ซึ่งมีคุณสมบัติเหมาะสมกับโครงงานที่สร้างขึ้น เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิท มีหน่วยความจำข้อมูลภายใน 128 ไบต์ มีพอร์ต 3 พอร์ต เพื่อใช้ต่อกับหน่วยความจำโปรแกรมภายนอก ใช้ต่อกับคีย์บอร์ดและต่อกับพอร์ต เพื่อเชื่อมเข้ากับวงจรขับ โดยที่ไมโครคอนโทรลเลอร์จะทำหน้าที่ควบคุมการเคลื่อนที่ทุกอย่างของสเต็ปปิ้ง เช่น จำนวนรอบของการหมุน ความเร็วรอบ จังหวะการหมุน โดยการส่งสัญญาณพัลส์ที่มีลักษณะเฉพาะ

2. ส่วนวงจรขับใช้ไอซีเบอร์ TEA 3718 ซึ่งเป็นไอซีสำหรับขับสเต็ปปิ้งมอเตอร์โดยเฉพาะ ในการขับสเต็ปปิ้งมอเตอร์แต่ละตัวจะใช้ไอซี 2 ตัว แต่ละตัวจะต่ออยู่กับขดลวด แต่ละชุดของมอเตอร์ การขับมีทั้งแบบเต็มสเต็ป และแบบครึ่งสเต็ป โดยวงจรขับจะทำหน้าที่ขยายสัญญาณลอจิกที่มาจากส่วนควบคุมให้มีขนาดใหญ่เพียงพอที่จะขับมอเตอร์ให้หมุนได้ นอกจากนั้นยังมีความสามารถในการควบคุมปริมาณกระแสที่จ่ายให้ขดลวดของมอเตอร์ ซึ่งมีประโยชน์ในกรณีต้องการให้มอเตอร์หมุนแบบครึ่งสเต็ป หรือ มินิสเต็ป

3. สำหรับสเต็ปปิ้งมอเตอร์ที่ใช้เป็นแบบไบโพลาร์ 2 เฟส จำนวนสเต็ปต่อรอบ 200 สเต็ป องศาต่อสเต็ป 1.8 องศา เมื่อควบคุมให้หมุนแบบครึ่งสเต็ปจำนวนสเต็ปต่อรอบจะเพิ่มขึ้นเป็น 400 สเต็ป องศาต่อสเต็ปจะมีค่า 0.9 องศา

ภาคผนวก

SGS-THOMSON MICROELECTRONICS

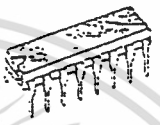
TEA3718
TEA3718S

STEPPER MOTOR DRIVER

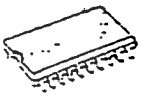
ADVANCE DATA

- HALF-STEP AND FULL-STEP MODE
- BIPOLAR DRIVE OF STEPPER MOTOR FOR MAXIMUM MOTOR PERFORMANCE
- BUILT-IN PROTECTION DIODES
- WIDE RANGE OF CURRENT CONTROL 5 TO 1500 mA
- WIDE VOLTAGE RANGE 10 TO 50 V
- DESIGNED FOR UNSTABILIZED MOTOR SUPPLY VOLTAGE
- CURRENT LEVELS CAN BE SELECTED IN STEPS OR VARIED CONTINUOUSLY
- THERMAL OVERLOAD PROTECTION
- ALARM OUTPUT OR PRE-ALARM OUTPUT (see internal table)

Powerdip 12 + 2 + 2



SO-20L



ORDERING NUMBERS : TEA3718SDP TEA3718DP

ORDERING NUMBER : TEA3718SFP

MULTIWATT-15

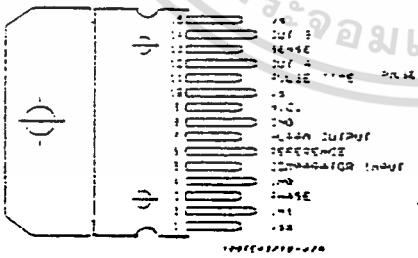
ORDERING NUMBER : TEA3718SP

DESCRIPTION

The TEA3718 and TEA3718S are bipolar monolithic integrated circuits intended to control and drive the current in one winding of a bipolar stepper motor. The circuits consist of an LS-TTL compatible logic input, a current sensor, a monostable and an output stage with built-in protection diodes. Two TEA3718 or TEA3718S and a few external components form a complete control and drive unit for LS-TTL or microprocessor-controlled stepper motor systems.

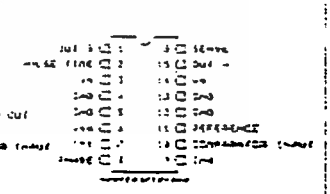
PIN CONNECTIONS (top views)

TEA3718SP (Multiwatt-15)



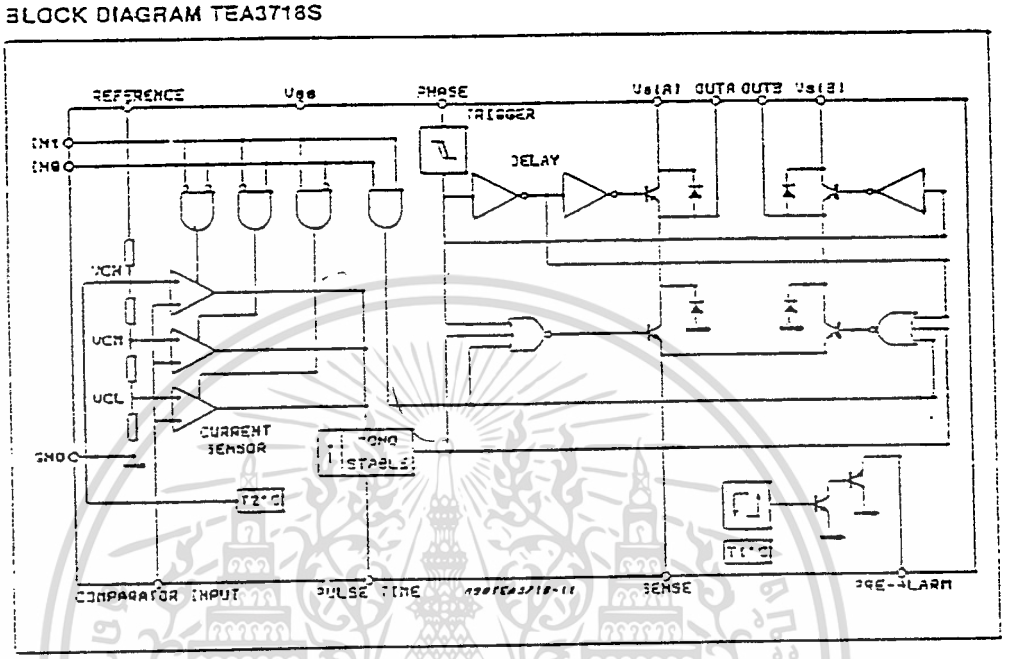
TEA3718SFP (SO-20L)

TEA3718DP TEA3718SDP (Powerdip 12+2+2)

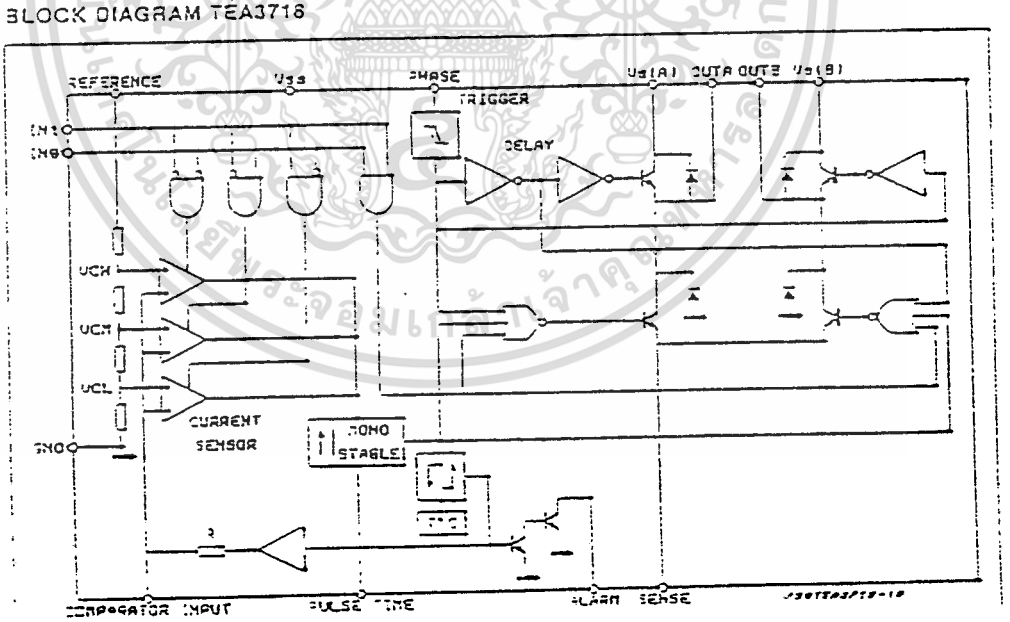


Note: HEATSINK SURFACE CONNECTED TO PIN 8

BLOCK DIAGRAM TEA3718S



BLOCK DIAGRAM TEA3718



PIN FUNCTIONS

Name	Function
OUT B	Output Connection (with pin OUTA). The output stage is a "H" bridge formed by four transistors and four diodes suitable for switching applications.
PULSE TIME	A parallel RC network connected to this pin sets the OFF time of the lower power transistors. The pulse generator is a monostable triggered by the rising edge of the output of the comparators (t_{off} = 0.69 RtCt).
V_S(B)	Supply Voltage Input for Half Output Stage
GND	Ground Connection. In SO-20L and Powerdip these pins also conduct heat from die to printed circuit copper.
V_{SS}	Supply Voltage Input for Logic Circuitry
IN1	This pin and pin IN0 are logic inputs which select the outputs of three comparators to set the current level. Current also depends on the sensing resistor and reference voltage. See truth table.
PHASE	This TTL-compatible logic input sets the direction of current flow through the load. A high level causes current to flow from OUT A (source) to OUT B (sink). A Schmitt trigger on this input provides good noise immunity and a delay circuit prevents output stage short circuits during switching.
IN0	See INPUT 1
COMPARATOR INPUT	Input connected to the three comparators. The voltage across the sense resistor is fedback to this input through the low pass filter RcCc. The lower power transistor are disabled when the sense voltage exceeds the reference voltage of the selected comparator. When this occurs the current decays for a time set by RtCt. T_{off} = 0.69 RtCt.
REFERENCE	A voltage applied to this pin sets the reference voltage of the three comparators. Reference voltage with the value of R_S and the two inputs IN0 and IN1 determines the output current.
V_S(A)	Supply voltage input for half output stage
OUT A	See pin OUT B
SENSE RESISTOR	Connection to lower emitters of output stage for insertion of current sense resistor
ALARM	When T_j reaches T1°C the alarm output becomes low (TEA3718SP)
PRE-ALARM	When T_j reaches T2°C the prealarm output becomes low (T2<T1) (TEA3718SFP)

TEA3718-TEA3718S

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameters	Value	Unit
V_{SS}	Supply Voltage	7	V
V_S		50	V
V_I	Input Voltage:		
	Logic Inputs	6	V
	Analog Inputs	V_{SS}	V
	Reference Input	15	V
I_I	Input Current		
	Logic Inputs	-10	mA
	Analog Inputs	-10	mA
I_O	Output Current	±1.5	A
T_j	Junction Temperature	+150	°C
T_{op}	Operating Ambient Temperature Range	0 to 70	°C
T_{stg}	Storage Temperature Range	-55 to +150	°C

THERMAL DATA

Symbol	Parameter	SO-20L	Powerdip	Multiwatt	Unit
$R_{th(j-c)}$	Maximum Junction-case Thermal Resistance	15	11	3	°C/W
$R_{th(j-a)}$	Maximum Junction-ambient Thermal Resistance	50 *	45 *	40	°C/W

* Soldered on a 35 µm thick 4 cm² PC board copper area.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{SS}	Supply Voltage	4.75	5	5.25	V
V_S	Supply Voltage	10	–	45	V
I_O	Output Current	0.020	–	1.2	A
T_{amb}	Ambient Temperature	0		70	°C
t_r	Rise Time Logic Inputs	–	–	2	µs
t_f	Fall Time Logic Inputs	–	–	2	µs

COMPARISON TABLE

Device	Current	Package	Alarm	Pre-Alarm
TEA3718SDP	1.5A	Powerdip 12+2+2		not connected
TEA3718SFP	1.5A	SO-20L		x
TEA3718SP	1.5A	Multiwatt-15	X	
TEA3718DP	1.5A	Powerdip 12+2+2	not connected	

MAXIMUM POWER DISSIPATION

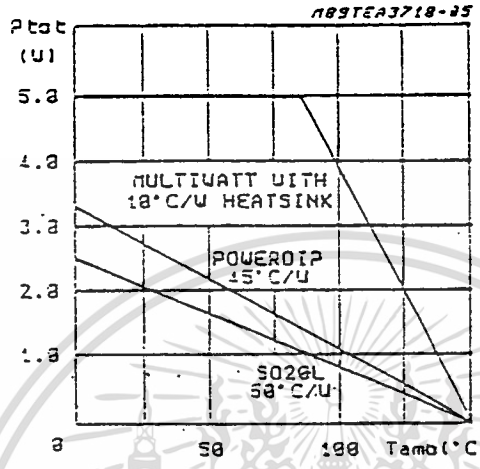


Figure 1.

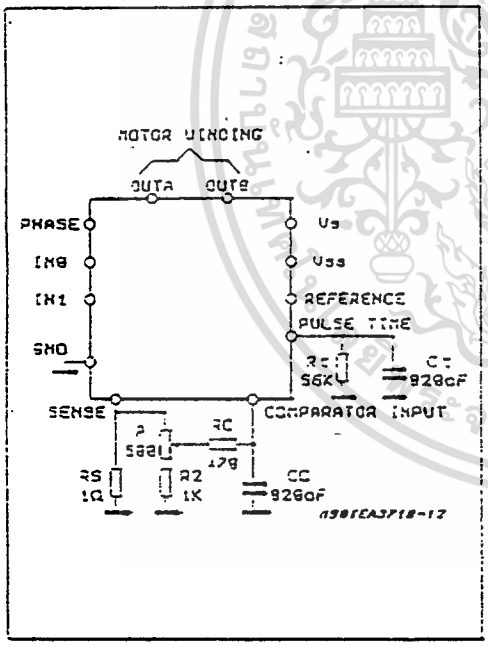
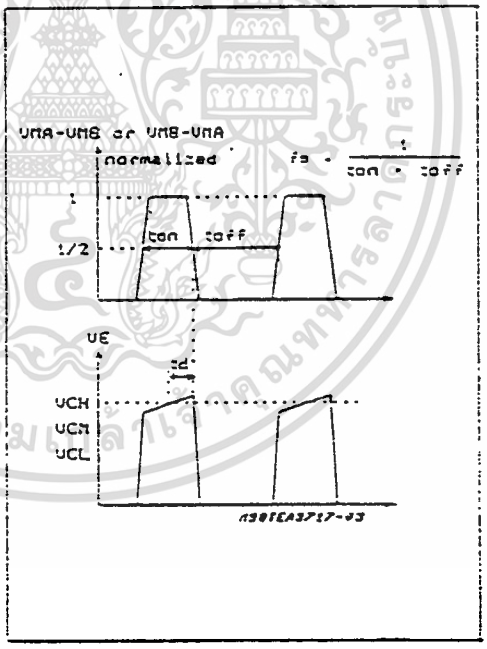


Figure 2.



R_S = 1 Ω INDUCTANCE FREE
R_C = 470 Ω
C_C = 820 pF CERAMIC
R_t = 56 kΩ
C_t = 820 pF CERAMIC
P = 500 Ω
R_2 = 1 K

TEA3718-TEA3718S

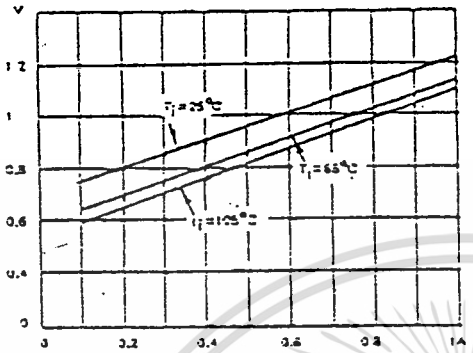
ELECTRICAL CHARACTERISTICS (V_{CC} = 5V, ± 5%, V_{mm} = 10V to 45V, T_{amb} = 0 to 70°C (T_{amb} = 25°C for TEA3718FP/SFP) unless otherwise specified)

Symbol	Parameter			Min.	Typ.	Max.	Unit
I_{CC}	Supply Current			-	-	25	mA
V_{IH}	High Level Input Voltage - Logic Inputs			2	-	-	V
V_{IL}	Low Level Input Voltage - Logic Inputs			-	-	0.8	V
I_{IH}	High Level Input Current - Logic Inputs			-	-	20	uA
I_{IL}	Low Level Input Current - Logic Inputs (VI = 0.4V)			-0.4	-	-	mA
V_{CH}	Comparator Threshold Voltage (VR = 5V)	I_O = 0	I_I = 0	390	420	440	mV
V_{CM}		I_O = 0	I_I = 0	230	250	270	mV
V_{CL}		I_O = 0	I_I = 1	65	80	90	mV
I_{CO}	Comparator Input Current			-20	-	20	uA
I_{off}	Output Leakage Current (I_O = 0, I_I = 1 T_{amb} = 25°C			-	-	100	uA
V_{sat}	Total Saturation Voltage Drop (Im = 1A)		SO20/Powerdip	-	-	2.3	V
			Multiwatt	-	-	3.2	V
P_{tot}	Total Power Dissipation - I_m = 1A, f_s = 30KHz			-	3.1	3.6	W
t_{off}	Cut off Time (see figure 1 and 2, V_{mm} = 10V, V_{on} > 5us			25	30	35	ms
t_d	Turn off Delay (see fig. 1 and 2, T_{amb} = 25°C, dVC/dt>50mV/us)			-	1.5	-	us
V_{sat}	Alarm Output Saturation Voltage - I_O = 2mA		(Multiwatt)	-	0.8	-	V
I_{ref}	Reference Input Current, VR = 5V			-	0.4	1	mA
V_{sat}	Source Diode Transistor Pair Saturation Voltage	Powerdip I_m = 0.5A		-	1.05	1.2 (1.3)	V
		Powerdip I_m = 1A		-	1.35	1.5 (1.7)	V
		Multiwatt I_m = 0.5A		-	-	1.3	V
		Multiwatt I_m = 1A		-	-	1.7	V
V_f	Diode Forward Voltage		I_f = 0.5A	-	1.1	1.5 (1.5)	V
			I_f = 1A	-	1.25	1.7 (1.9)	V
I_{sub}	Substrate Leakage Current		I_f = 1A	-	-	5	mA
V_{sat}	Sink Diode Transistor Pair Saturation Voltage	Powerdip I_m = 0.5A		-	1	1.2 (1.3)	V
		Powerdip I_m = 1A		-	1.2	1.3 (1.5)	V
		Multiwatt I_m = 0.5A		-	-	1.3	V
		Multiwatt I_m = 1A		-	-	1.5	V
V_f	Diode Forward Voltage		I_f = 0.5A	-	1	1.4 (1.5)	V
			I_f = 1A	-	1.1	1.5 (1.9)	V

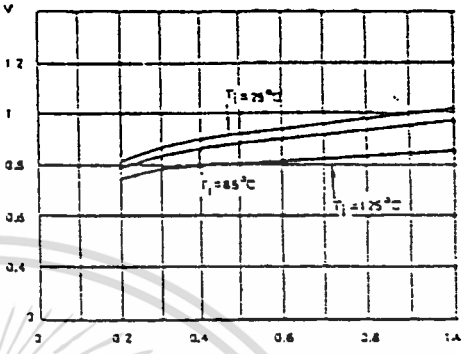
Notes:

(...) Only for TEA3718SFP mounted in SO-20L package.

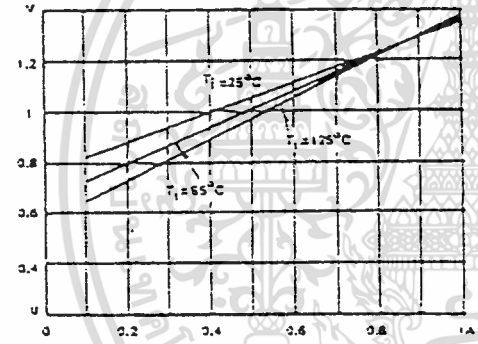
SINK DRIVER $V_{CE\ sat}$ VS I_{OUT} and T_j



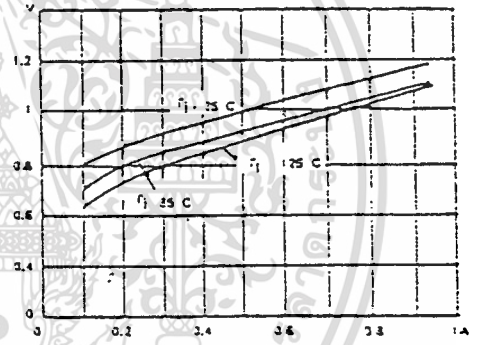
LOWER DIODE V_F VS I_{OUT} and T_j



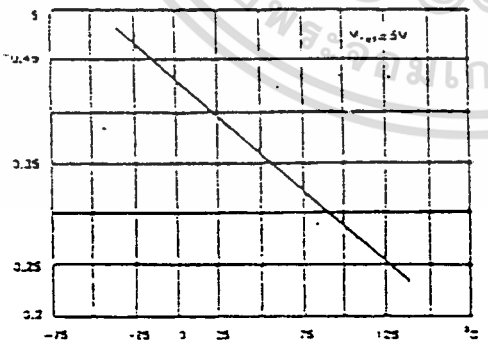
SOURCE DRIVER $V_{CE\ sat}$ VS I_{OUT} and T_j



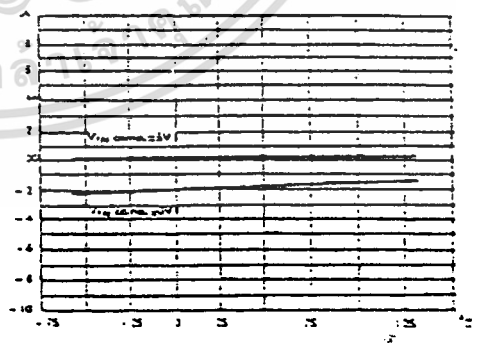
UPPER DIODE V_F VS I_{OUT} and T_j



I_{src} VS JUNCTION TEMPERATURE



COMPARATOR INPUT CURRENT VS T_j and V_C



National Semiconductor

Preliminary
February 1980

INS8048-Series Microcomputer/Microprocessor Family

General Description

The INS8048/49/50-Series microcomputers and the INS8035/39/40-Series microprocessors (hereinafter referred to as the 48-Series) are self contained, 8-bit parallel, 40-pin, dual in-line devices fabricated using National Semiconductor's scaled N-channel, silicon gate MOS process, XMOS. The 48-Series devices contain the system timing, control logic, ROM (where applicable) program memory, RAM data memory and 27 I/O lines necessary to implement dedicated control functions. All 48-Series devices are pin compatible, differing only in the size of on-board ROM (where applicable) and RAM as shown below:

DEVICE	RAM ARRAY	ROM ARRAY
INS8048	64 x 8	1K x 8
INS8049	128 x 8	2K x 8
INS8050	256 x 8	4K x 8
INS8035	64 x 8	N/A
INS8039	128 x 8	N/A
INS8040	256 x 8	N/A

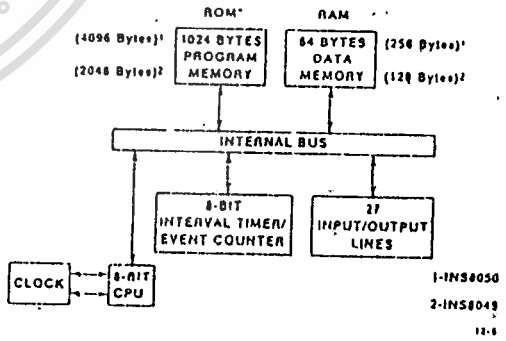
The devices are designed to be efficient controllers. They have extensive bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory is derived from an instruction set comprised predominantly of single bytes. The remaining instructions are two bytes in length. Additional external memory may be added up to a maximum of 4K bytes of program memory and 256 bytes of data memory without paging.

Features

- 8-Bit CPU, RAM, ROM, I/O in Single Package
- 2.5 μsec Cycle, 6 MHz Clock; 1.36 μsec Cycle, 11 MHz Clock
- On-Chip Oscillator Circuit and Clock (or External Source)
- 27 I/O Lines
- Expandable Memory and I/O
- 8-Bit Timer/Counter
- Single Level Interrupt
- Interrupt has Schmitt Trigger with Hysteresis*
- Over 90 Instructions (Most Single Byte)
- Binary and BCD Arithmetic
- Single +5V Power Supply
- Low Standby Power Mode*
- Memory on Standby is Programmable*
- Low Voltage Standby (2.2V Min)*
- On-Chip Battery Charging*
- Counter Increment, Non-Increment Option

*NOTE Transparent improvements over industry standard part.

48-Series Block Diagram



*Not Applicable to INS8035/39/40

Absolute Maximum Ratings

Temperature Under Bias ..... 0° C to +70° C
Storage Temperature ..... -65° C to +150° C
All Input or Output Voltages with respect V_{SS} ..... -0.5V to +7.0V
Power Dissipation ..... 1.5 Watt

NOTE: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under DC Electrical Characteristics.*

DC Electrical Characteristics

T_A = 0° C to +70° C, V_{CC} = +5V ±10%, V_{SS} = 0V unless otherwise specified.

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (All except XTAL1, XTAL2)	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	(1)
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage All except ports	3.0		V_{CC}	V	I_{OH} = 100 µA
V_{OH1}	Port Option 1, Port TTL (Std. Drain)	2.4			V	$I_{OH} \geq$ 125 µA (1)
V_{OH2}	Port Option 2 (Open Drain)			-100	µA	$V_{CC} \geq V_{OUT} \geq$ 0.40
I_P Max	Port Current (Max) (2)				mA	Each port output from 0V to V_{CC}
I_{IL}	Input Leakage Current (T1, EA, INT)			±10	µA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OL}	Output Leakage Current (BUS T0) (High Impedance State)			-100	µA	$V_{CC} \geq V_{IN} \geq V_{SS}$ + 0.45
I_{DD} (32)	32 words of Standby Current (2)			1.5	mA	at 2.2V supply
I_{DD} (64)	64 words of Standby Current (2)			2.5	mA	
I_{DD} (96)	96 words of Standby Current (2)			3.5	mA	8049, 8050 only
I_{DD} (128)	128 words of Standby Current (2)			4.5	mA	8049, 8050 only
I_{DD} (160)	160 words of Standby Current (2)			5.5	mA	8050 only
I_{DD} (192)	192 words of Standby Current (2)			6.5	mA	8050 only
I_{DD} (224)	224 words of Standby Current (2)			7.5	mA	8050 only
I_{DD} (256)	256 words of Standby Current (2)			8.5	mA	8050 only
I_{DD} +I_{CC}	Total Supply Current 8048		30	65	mA	T_A = 25° C
I_{DD} +I_{CC}	Total Supply Current 8049		32	70	mA	T_A = 25° C
I_{DD} +I_{CC}	Total Supply Current 8050		35	75	mA	T_A = 25° C
I_{DDC}	Battery Charging Current			TBD	mA	See *Figure 5*
V_{DD}	Standby Power Supply	2.2		V_{CC}	V	See *Figure 5*

Notes: 1 Port pullup current is mask programmable
2 Location and number of words in RAM on stand-by are factory programmable. Current with 2.2V (Nickel Cadmium cells)

AC Electrical Characteristics - INS80XX-6 (1-6 MHz part)

T_A = 0° C to +70° C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
t_{LL}	ALE Pulse Width	400			ns	Note 1
t_{AL}	Address Setup to ALE	150			ns	Note 1
t_{LA}	Address Hold from ALE	80			ns	Note 1
t_{CC}	Control Pulse Width $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$	700			ns	Note 1
t_{DW}	Data Set-Up Before $\overline{WR}$	500			ns	Note 1
t_{WD}	Data Hold After $\overline{WR}$	120			ns	C_L = 20 pF
t_{CY}	Cycle Time	2.5		15.0	µs	1 to 6 MHz XTAL
t_{DR}	Data Hold	0		200	ns	Note 1
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In			500	ns	Note 1
t_{AW}	Address Setup to $\overline{WR}$	230			ns	Note 1
t_{AD}	Address Setup to Data In			950	ns	Note 1
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$	0			ns	Note 1
t_{CA}	Control Pulse to ALE	10			ns	Note 1

Port 2 Timing

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
t_{CP}	Port Control Setup before Falling Edge of PROG	110			ns	Note 1
t_{PC}	Port Control Hold after Falling Edge of PROG	140			ns	Note 1
t_{PR}	PROG to Time P2 Input must be Valid			810	ns	Note 1
t_{DP}	Output Data Setup Time	250			ns	Note 1
t_{PD}	Output Data Hold Time	65			ns	Note 1
t_{PF}	Input Data Hold Time	0		150	ns	Note 1
t_{PP}	PROG Pulse Width	1510			ns	Note 1
t_{PL}	Port 2 I/O Data Setup	400			ns	Note 1
t_{LP}	Port 2 I/O Data Hold	150			ns	Note 1

AC Electrical Characteristics - INS80XX-11 (4-11 MHz Part)

T_A = 0° C to +70° C, +5V ±10%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
t_{LL}	ALE Pulse Width	150			ns	Note 1
t_{AL}	Address Setup to ALE	70			ns	Note 1
t_{LA}	Address Hold from ALE	50			ns	Note 1
t_{CC}	Control Pulse Width $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$	300			ns	Note 1
t_{DW}	Data Set-Up Before $\overline{WR}$	250			ns	Note 1
t_{WD}	Data Hold After $\overline{WR}$	40			ns	C_L = 20 pF
t_{CY}	Cycle Time	1.36		15.0	µs	4 to 11 MHz XTAL
t_{DR}	Data Hold	0		100	ns	Note 1
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In			200	ns	Note 1

AC Electrical Characteristics - INS80XX-11 (Cont'd.)

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
t_{AW}	Address Set-Up to $\overline{WR}$	200			ns	Note 1
t_{AD}	Address Set-up to Data In			400	ns	Note 1
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$	-10			ns	Note 1
t_{CA}	Control Pulse to ALE	10			ns	Note 1

Port 2 Timing

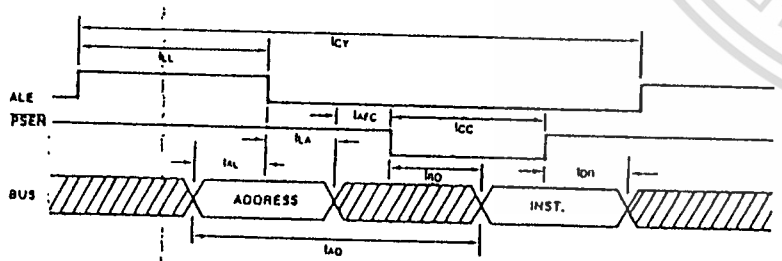
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
t_{CP}	Port Control Setup before Falling Edge of PROG	100			ns	Note 1
t_{PC}	Port Control Hold after Falling Edge of PROG	60			ns	Note 1
t_{PR}	PROG to Time P2 Input must be Valid			650	ns	Note 1
t_{DP}	Output Data Setup Time	200			ns	Note 1
t_{PD}	Output Data Hold Time	20			ns	Note 1
t_{PF}	Input Data Hold Time	0		150	ns	Note 1
t_{PP}	PROG Pulse Width	700			ns	Note 1
t_{PL}	Port 2 I/O Data Setup	150			ns	Note 1
t_{LP}	Port 2 I/O Data Hold	20			ns	Note 1

Note 1 Control outputs C_L 80 pF; Bus outputs C_L = 150 pF

Capacitance $T_A = 25°C$, $V_{CC} = V_{SS} = 0V$

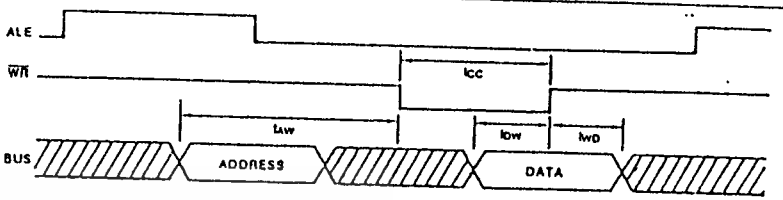
Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
C_{IN}	Input Capacitance		6	10	pF	f_c = 1 MHz
C_{OUT}	OUTPUT AND $\overline{RESET}$ Capacitance		10	20	pF	Unmeasured pins returned to V_{SS}

Timing Waveforms

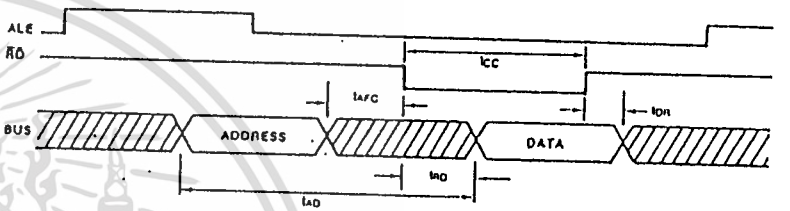


Instruction Fetch from External Program Memory

NOTE: Diagonal lines indicate interval of high impedance.

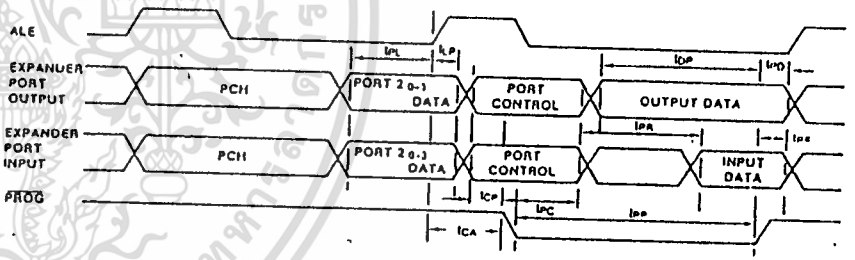


Write to External Data Memory

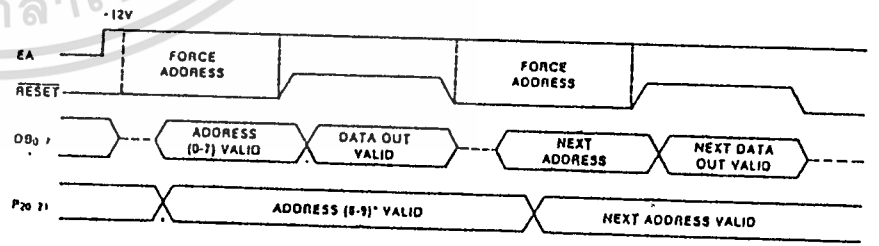


Read from External Data Memory

NOTE: Diagonal lines indicate interval of high impedance.



Port 2 Timing



*8049 = 8-10
8050 = 8-11

Verify Mode Timing

Functional Pin Description

INPUT SIGNALS

Reset ($\overline{RESET}$): An active low (0) input that initializes the processor and is used to verify program memory. (See note #1.)

Single Step ($\overline{SS}$): Active low (0) input which, in conjunction with ALE, can single step the processor through each instruction.

External Access (EA): An active high (1) input that forces all program memory fetches to reference external program memory.

Testable Input 0 (T_0): Testable input pin using conditional branch functions JT0 (T_0 = 1) or JNT0 (T_0 = 0). T_0 can be designated as the clock output using instruction ENT0 CLK.

Testable Input 1 (T_1): Testable input pin using conditional branch functions JT1 (T_1 = 1) or JNT1 (T_1 = 0). T_1 can be designated as the Timer/Counter input from an external source using instruction EN TCNTI.

Interrupt ($\overline{INT}$): An active low input that initiates an interrupt when interrupt is enabled. Interrupt is disabled after a reset. Also can be tested with instruction JNI ($\overline{INT}$ = 0). (See Note 2.)

OUTPUT SIGNALS

Read Strobe ($\overline{RD}$): An active low output strobe activated during a Bus read. Can be used to enable data onto the BUS from an external device. Used as a Read Strobe to External Data Memory.

Write Strobe ($\overline{WR}$): An active low output strobe activated during a Bus write. Used as a Write Strobe to External Data Memory.

Program Store Enable ($\overline{PSEN}$): An active low output that occurs only during an external program memory fetch.

Address Latch Enable (ALE): An active high output that occurs once during each cycle and is useful as a clock output. The negative going edge of ALE strobes the address into external data or program memory.

Program ($\overline{PROG}$): This output (active high) provides the output strobe for INS8243 I/O Expander.

INPUT/OUTPUT SIGNALS

Crystal Input (XTAL1, XTAL2): These two pins connect the crystal for internal oscillator operation. XTAL1 is the timing input for external source.

Port 1 (P10-P17): 8-bit quasi-bidirectional port.

Port 2 (P20-P27): 8-bit quasi-bidirectional port. During an external program memory fetch, the four high-order program counter bits occur at P20-P23. They also serve as a 4-bit I/O expander bus when the INS8243 I/O Expander is used. (See note 3).

BUS (DB_0-DB_7): True bidirectional port, either statically latched or synchronous. Can be written to using $\overline{WR}$ Strobe, or Read from using $\overline{RD}$ Strobe. During an external program memory fetch, the 8 lower order program counter bits are preset at this port. The addressed instruction appears on this bus when $\overline{PSEN}$ is low. During an external RAM data store instruction, this port presents address and data under control of ALE, $\overline{RD}$, and $\overline{WR}$.

V_{SS}: Processor Ground potential.

V_{DD}: V_{DD} functions as the Low Power Stand-by Voltage and can vary from 2.2V to 5.5V.

V_{CC}: Pin 40: Primary Power Source for 48-Series Devices.

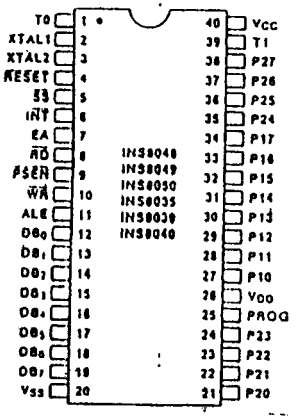
NOTES:

1. The $\overline{Reset}$ input has a pullup resistor which may be disconnected by a mask program to provide greater flexibility for a Power Reset. (See option 26.)

2. The $\overline{INT}$ input does not have a pullup resistor, but a mask programmable resistor can be provided to increase chip flexibility (See option 27.)

3. The Port 2 pins may be mask programmed to provide TTL drive or open drain capabilities (see options 10-25). (Refer to paragraph on Transparent Improvements.)

Pin Configuration



Functional Description

The following paragraphs contain the functional description of the major elements of the 48-Series microcomputer/microprocessor. *Figure 1* is a block diagram of the 48-Series devices. The data paths are illustrated in simplified form to show how the various logic elements communicate with each other to implement the instruction set common to all devices.

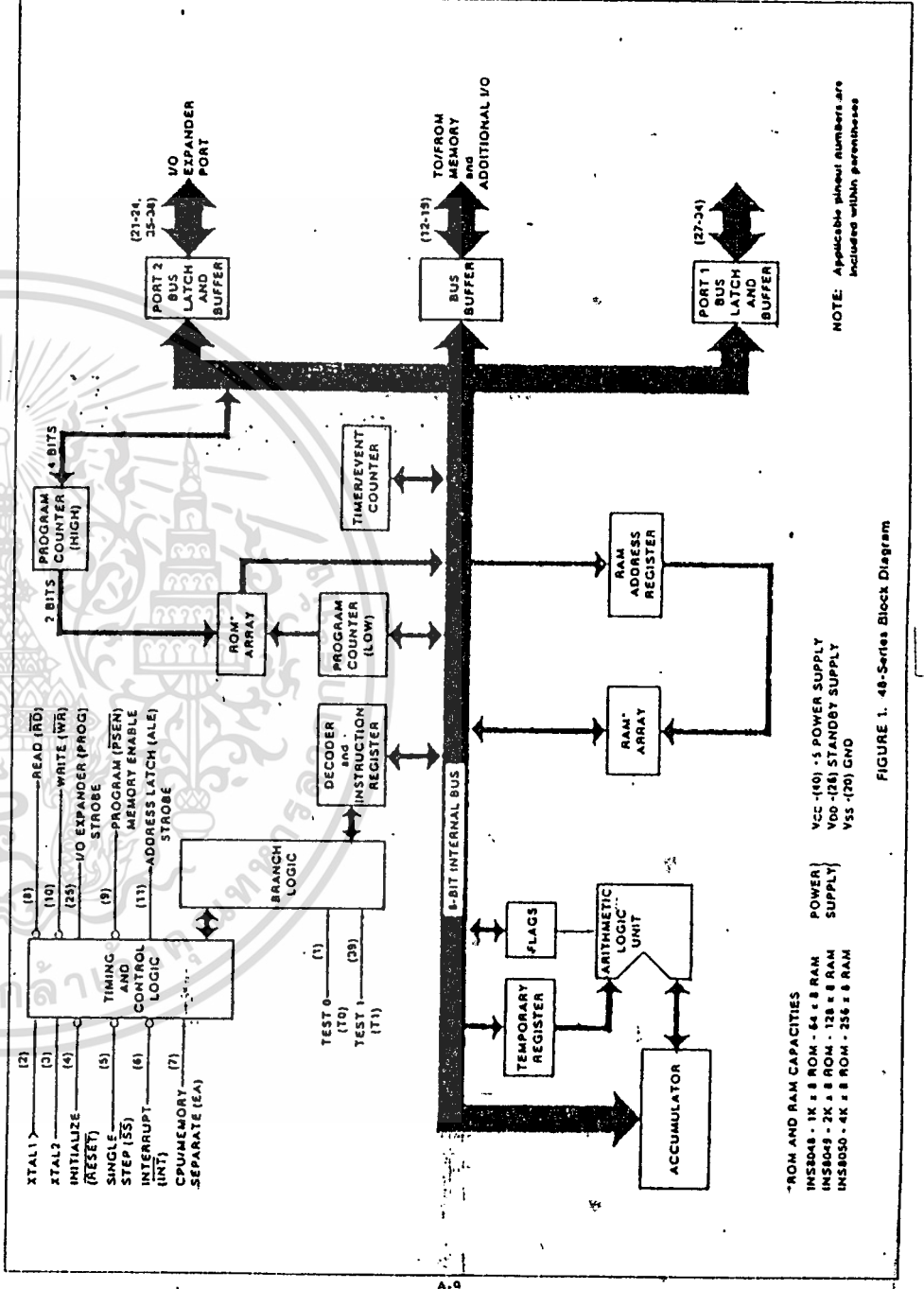


FIGURE 1. 48-Series Block Diagram

Program Memory

The Program Memory (ROM) contained on the INS8048/49/50 devices is comprised of 1024, 2048 or 4096 8-bit bytes, respectively. As it is seen by examining the 48-Series instruction set, these bytes may be program instructions, program data or ROM addressing data. The ROM for the above devices must be mask programmed at the National Semiconductor factory. The ROMless microprocessors, INS8035, INS8039 and INS8040 use external program memory. This makes program development straightforward using standard UV erasable PROMs to emulate a possible future single chip (using the on-board ROM) system. ROM addressing, up to a maximum of 4K, is accomplished by a 12-bit Program Counter (PC). The INS8048 and INS8049 will automatically address external memory when the boundary of their internal memories, 1K and 2K respectively, are exceeded. The binary value of the address selects one of the 8-bit bytes contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential binary count value.

With reference to the Program Memory Map (see *Figure 2*) there are three ROM addresses which provide for the control of the microcomputer.

1. Memory Location 0000 - Resetting the Reset (negative true) input to the microcomputer forces the first instruction to be fetched from address 0000.
2. Memory Location 0003 - Asserting the Interrupt (negative true) input to the microcomputer (when Interrupt is enabled) forces a jump to subroutine.
3. Memory Location 0007 - A timer/counter interrupt that results from timer/counter overflow (when enabled) forcing a jump to subroutine.

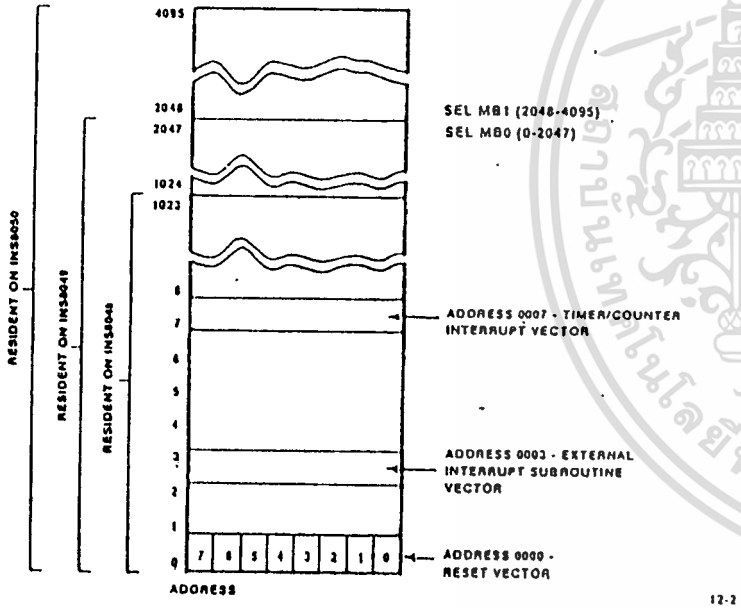


FIGURE 2. INS8048/49/50 Resident ROM Program Memory Map

Data Memory (RAM)

The resident RAM data memory is arranged as 64 (INS8035/8048), 128 (INS8039/8049) or 256 (INS8040/8050) bytes. RAM addressing is implemented indirectly via either of two 8-bit RAM pointer registers R0 and R1. These pointer registers are essentially the first two locations in the RAM (see *Figure 3*), addresses 000 and 001. RAM addressing may also be performed directly by 11 direct register instructions. The pointer register area of the RAM array is made up of eight working registers that occupy either the first bank (0), locations (0 to 7), or the second bank (1), locations 24-31. The second bank of working registers is selected by using the Register Bank Switch instruction (SEL RB). If this bank is not used for working registers, it can be used as user RAM.

There is an 8-level stack after Bank 0 that occupies address locations 8 to 23. These RAM locations are addressed indirectly through R0, R1 or the 3-bit Stack Pointer (SP). The stack pointer keeps track of the return address and pushes each return address down into the stack. There are 8 levels of subroutine nesting possible in the stack because each address occupies 10 bits or more using two bytes in RAM. When the level of subroutine nesting is less than 8, the stacks not used may be utilized as user RAM locations.

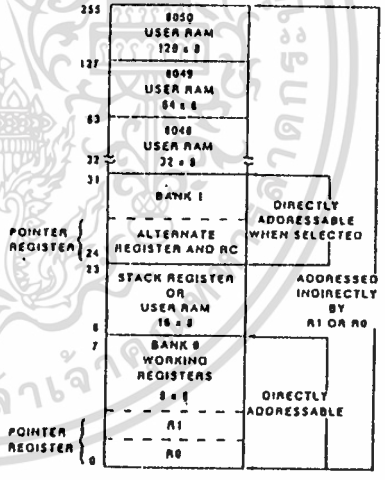


FIGURE 3. 48-Series Resident RAM Data Memory Map

Input/Output

The 48-Series devices have 27 lines of input/output organized as three, 8-bit ports plus three test inputs. The three ports may be used as inputs, outputs or bidirectional ports. Ports 1 and 2 differ from port 3 (Bus Port) in that they are quasi-bidirectional ports. Ports 1 and 2 can be used as input and output while being statically latched. If more I/O lines are required, Port 1 and/or Port 2 can also serve as a 4-bit I/O bus expander when used in conjunction with the INS8243 I/O Expander.

National has designed two options for the port current drive: the standard TTL drive of 100 μA at 2.4 volts (see *Figure 4*) and:

Open Drain - When a logic 1 is written, the output will supply less than 10 μA. When a logic 0 is written, the output will supply at least 2mA at 0.4V to ground.

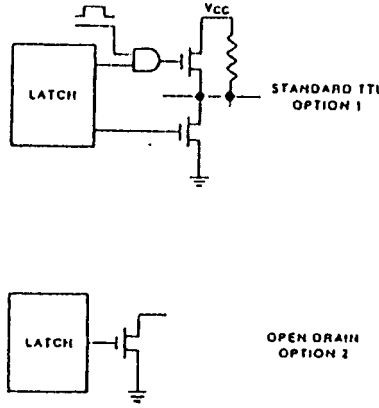


FIGURE 4. Input/Output Options

The bus port is a true bidirectional port and is either statically latched or synchronous. It can be written to using $\overline{WR}$ strobe or read from using $\overline{RD}$ strobe. During an external program memory fetch, the 8 lower order program counter bits are preset at this port. The addressed instruction appears on this bus when $\overline{PSEN}$ is low. During an external RAM data store instruction, this port presents address and data under control of ALE, $\overline{RD}$, and $\overline{WR}$.

Transparent Improvements

National has made some additional improvements to the standard industry parts. These include a battery charging circuit, programmable RAM on standby, interrupt pin with hysteresis, and programmable current drive. Also, these improvements are transparent to the user and, as noted, some options are programmed at the factory.

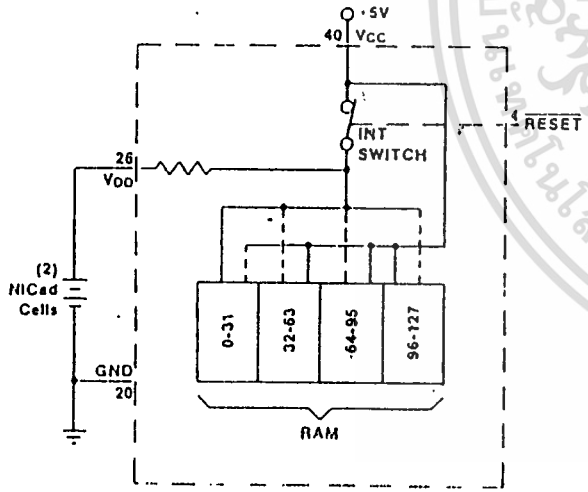
Programmable RAM on Standby During POWER Down Mode (see *Figure* 5)

Battery backed applications are accomplished by factory programming the minimum amount of RAM kept alive during standby. At 32 words, the National 48-Series consumes only 3 mA at a minimum 2.2 volts (See DC Electrical Parameters). The banks of RAM are individually mask programmable in groups of 32 words. Each of the 32 byte sections can be factory programmed to be connected to V_{DD} in any combination.

During the power down mode, V_{DD} which normally maintains the RAM cells, is the only pin that receives power. V_{CC}, which serves the CPU and ports is dropped from nominal 5 volts to 0 volt, after the CPU is reset, so that the RAM cells are unaltered by the loss of power. When power is restored, the processor goes through the normal power-on procedure.

Battery Charging Circuit

All 48-Series devices contain a circuit to provide external battery charging capabilities. Power for all on-board circuits are provided by V_{CC} (pin 40). As shown in *Figure* 5 under normal operating conditions the $\overline{RESET}$ input is a logic high holding the internal switch in the closed position. V_{CC} is supplied to the program selectable portion of the RAM array through the closed contact of the internal switch. The normally closed contacts of the switch also provide charging power to the external NiCad cells. In the event of power failure, the $\overline{RESET}$ pin must be pulled low before V_{CC} drops below 4.5 volts in order to guarantee the RAM will not loose data. When the $\overline{RESET}$ pin becomes a logic low (0V) the internal switch is forced to the open condition. DC power to sustain the desired RAM data is provided by the two NiCad batteries (approximately 2.2 volts). Normally, approximately 5 volts are required to provide RAM data protection in the event of a power failure. National's innovative advances in NMOS technology provide the user with a RAM that requires 50% less voltage and 10% of the power to protect data during power failure. The on-chip charging circuit and lower RAM power requirements provide the user with a twofold saving; no external circuits required for the battery charging and only 2 NiCad cells as opposed to the normal requirement of 4 to 5 NiCad cells.



12-5

FIGURE 5. INS8049 Battery Charging Circuit

Counter Increment, Nonincrement Option (Option 28)

National Semiconductor has designed the 48-Series to conform with either of the two versions presently available. The user may select the device suitable for his application. The difference in the devices is as follows:

1. One or more pulses applied to the input of the stopped counter causes the counter to increment by one when restarted.
2. The aforementioned incrementing of the stopped counter by input pulses is eliminated when the counter is restarted.

National Semiconductor accomplishes either option by mask programming at the factory.

Instruction Set

Table 1 details the 95 instructions common to both the microcomputers and the microprocessors. The table provides the mnemonic, function and description, instruction code, number of cycles and, where applicable, flag settings.

Table 1 Instruction Set

MNEMONIC	FUNCTION	DESCRIPTION	CYCLES	BYTES	FLAGS C	AC	F0	F1
		CONTROL						
EN I		Enable the External Interrupt Input.	1	1				
DIS I		Disable the External Interrupt Input	1	1				
ENTO CLK		Enable T0 as the Clock Output	1	1				
SEL MB0	(DBF) ← 0	Select Bank 0 (locations 0 - 2047) of Program Memory.	1	1				
SEL MB1	(DBF) ← 1	Select Bank 1 (locations 2048 - 4095) of Program Memory.	1	1				
SEL RB0	(BS) ← 0	Select Bank 0 (locations 0 - 7) of Data Memory.	1	1				
SEL RB1	(BS) ← 1	Select Bank 1 (locations 24 - 31) of Data Memory	1	1				
		DATA MOVES						
MOV A, #data	(A) ← data	Move Immediate the specified data into the Accumulator	2	2				
MOV A, Rr	(A) ← (Rr); r = 0 - 7	Move the contents of the designated registers into the Accumulator.	1	1				
MOV A, @ Rr	(A) ← ((Rr)); r = 0 - 1	Move Indirect the contents of data memory location into the Accumulator.	1	1				
MOV A, PSW	(A) ← (PSW)	Move contents of the Program Status Word into the Accumulator	1	1				
MOV Rr, #data	(Rr) ← data; r = 0 - 7	Move Immediate the specified data into the designated register	2	2				
MOV Rr, A	(Rr) ← (A); r = 0 - 7	Move Accumulator contents into the designated register	1	1				
MOV @ Rr, A	((Rr)) ← (A); r = 0 - 1	Move Indirect Accumulator contents into data memory location	1	1				
MOV @ Rr, #data	((Rr)) ← data; r = 0 - 1	Move Immediate the specified data into data memory.	2	2				
MOV PSW, A	(PSW) ← (A)	Move contents of Accumulator into the Program Status Word.	1	1	•	•	•	
MOVP A, @ A	(PC 0 - 7) ← (A) (A) ← ((PC))	Move the content of program memory location in the current page addressed by the content of accumulator into the accumulator	2	1				
MOVP3 A, @ A	(PC 0 - 7) ← (A) (PC 8 - 10) ← 011 (A) ← ((PC))	Move the content of program memory location in page 3 address by the content of accumulator into the accumulator.	2	1				
MOVX A, @ R	(A) ← ((Rr)); r = 0 - 1	Move Indirect the contents of external data memory into the Accumulator	2	1				

Table 1. Instruction Set (Cont'd.)

MNEMONIC	FUNCTION	DESCRIPTION	CYCLES	BYTES	FLAGS C	AC	F0	F1
		DATA MOVES (Cont'd.)						
MOVX @ R A	((Rr)) ← (A); r = 0 - 1	Move indirect the contents of the Accumulator into external data memory	2	1				
XCH A, Rr	(A) ↔ (Rr); r = 0 - 7	Exchange the Accumulator and designated register's contents	1	1				
XCH A, @ Rr	(A) ↔ ((Rr)); r = 0 - 1	Exchange indirect contents of Accumulator and location in data memory	1	1				
XCHD A, @ Rr	(A0 - A3) ↔ (((Rr)) 0 - 3); r = 0 - 1	Exchange indirect 4-bit contents of Accumulator and data memory.	1	1				
		TIMER/COUNTER						
EN TCNTI		Enable Internal Interrupt Flag for Timer/Counter output	1	1				
DIS TCNTI		Disable Internal Interrupt Flag for Timer/Counter output	1	1				
MOV A, T	(A) ← (T)	Move contents of Timer/Counter into Accumulator	1	1				
MOV T, A	(T) ← (A)	Move contents of Accumulator into Timer/Counter	1	1				
STOP TCNT		Stop Count for Event Counter/Timer	1	1				
STRT CNT		Start Count for Event Counter	1	1				
STRT T		Start Count for Timer	1	1				
		ACCUMULATOR						
ADD A, #data	(A) ← (A) + data	Add immediate the specified Data to the Accumulator	2	2	•	•		
ADD A, Rr	(A) ← (A) + (Rr) for r = 0 - 7	Add contents of designated register to the Accumulator	1	1	•	•		
ADD A, @ Rr	(A) ← (A) + ((Rr)) for r = 0 - 1	Add indirect the contents the data memory location to the Accumulator	1	1	•	•		
ADDC A, #data	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the Accumulator	2	2	•	•		
ADDC A, Rr	(A) ← (A) + (C) + (Rr) for r = 0 - 7	Add with carry the contents of the designated register to the Accumulator	1	1	•	•		
ADDC A, @ Rr	(A) ← (A) + (C) + ((Rr)) for r = 0 - 1	Add indirect with carry the contents of data memory location to the Accumulator	1	1	•	•		
ANL A, #data	(A) ← (A) AND data	Logical AND specified Immediate Data with Accumulator	2	2				
ANL A, Rr	(A) ← (A) AND (Rr) for r = 0 - 7	Logical AND contents of designated register with Accumulator	1	1				
ANL A, @ Rr	(A) ← (A) AND ((Rr)) for r = 0 - 1	Logical AND indirect the contents of data memory with Accumulator	1	1				
CPL A	(A) ← NOT (A)	Complement the contents of the Accumulator	1	1				
CLR A	(A) ← 0	CLEAR the contents of the Accumulator	1	1				
DA A		DECIMAL ADJUST the contents of the Accumulator	1	1	•			
DEC A	(A) ← (A) - 1	DECREMENT by 1 the accumulator's contents	1	1				
INC A	(A) ← (A) + 1	Increment by 1 the accumulator's contents	1	1				
ORL A, #data	(A) ← (A) OR data	Logical OR specified Immediate data with Accumulator	2	2				
ORL A, Rr	(A) ← (A) OR (Rr) for r = 0 - 7	Logical OR contents of designated register with Accumulator	1	1				
ORL A, @ Rr	(A) ← (A) OR ((Rr)) for r = 0 - 1	Logical OR indirect the contents of data memory location with Accumulator	1	1				

Table 1. Instruction Set (Cont'd.)

MNEMONIC	FUNCTION	DESCRIPTION	CYCLES	BYTES	FLAGS C	AC	F0	F1
		ACCUMULATOR (Cont'd.)						
RLA	(An + 1) ← (An) for n = 0 - 6 (A0) ← (A7)	Rotate Accumulator left by 1-bit without carry	1	1				
RLC A	(An+1) ← (An); n = 0-6 (A0) ← (C) (C) ← (A7)	Rotate Accumulator left by 1-bit through carry.	1	1				
RR A	(An) ← (An+1); n = 0-6 (A7) ← (A0)	Rotate Accumulator right by 1-bit without without carry.	1	1				
RRC A	(An) ← (An+1); n = 0-6 (A7) ← (C) (C) ← (A0)	Rotate Accumulator right by 1-bit through carry	1	1				
SWAP A	(A4-A7) ↔ (A0 - A3)	Swap the 2, 4-bit nibbles in the Accumulator	1	1				
XRL A, #data	(A) ← (A) XOR data	Logical XOR immediate specified data with Accumulator.	2	2				
XRL A, Rr	(A) ← (A) XOR (Rr) for r = 0 - 7	Logical XOR contents of designated register with Accumulator	1	1				
XRL A, @ Rr	(A) ← (A) XOR ((Rr)) for r = 0 - 1	Logical XOR indirect the contents of data memory location with Accumulator	1	1				
		BRANCH						
DJNZ Rr, addr	(Rr) ← (Rr) -1; r = 0-7 if (Rr) ≠ 0; (PC 0-7) ← addr	Decrement the specified register and test contents	2	2				
JBb addr	(PC 0-7) ← addr if Bb = 1 (PC) ← (PC) + 2 if Bb = 0	Jump to specified address if Accumulator bit is set	2	2				
JC addr	(PC 0-7) ← addr if C = 1 (PC) ← (PC) +2 if C = 0	Jump to specified address if carry flag is set	2	2				
JF0 addr	(PC 0-7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address if Flag F0 is set	2	2				
JF1 addr	(PC 0-7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address if Flag F1 is set.	2	2				
JMP addr	(PC 8-10) ← addr 8-10 (PC 0-7) ← addr 0-7 (PC 11) ← DBF	Direct Jump to specified address within the 2K address block.	2	2				
JMPP @ A	(PC 0-7) ← ((A))	Jump indirect to specified address pointed to by the accumulator in current page	2	1				
JNC addr	(PC 0-7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump to specified address if carry flag is low	2	2				
JNI addr	(PC 0-7) ← addr if I = 0 (PC) ← (PC) + 2 if I = 1	Jump to specified address if interrupt is low	2	2				
JNT0 addr	(PC 0-7) ← addr if T0 = 0 (PC) ← (PC) + 2 if T0 = 1	Jump to specified address if Test 0 is low	2	2				
JNT1 addr	(PC 0-7) ← addr if T1 = 0 (PC) ← (PC) + 2 if T1 = 1	Jump to specified address if Test 1 is low	2	2				
JNZ addr	(PC 0-7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump to specified address if accumulator is non-zero	2	2				
JTF addr	(PC 0-7) ← addr if TF = 1 (PC) ← (PC) + 2 if TF = 0	Jump to specified address if Timer Flag is set to 1	2	2				
JT0 addr	(PC 0-7) ← addr if T0 = 1 (PC) ← (PC) + 2 if T0 = 0	Jump to specified Address if Test 0 is a 1	2	2				
JT1 addr	(PC 0-7) ← addr if T1 = 1 (PC) ← (PC) + 2 if T1 = 0	Jump to specified address if Test 1 is a 1	2	2				
JZ addr	(PC 0-7) ← addr if A = 0 (PC) ← (PC) + 2 if A = 1	Jump to specified address if Accumulator is 0	2	2				

TYPICAL APPLICATIONS (Cont'd)

Figure 7 shows a typical remote data acquisition system with an INS8250 Programmable Asynchronous Communication System which can receive commands or update information from a supervisory computer. The figure also shows an INS8294 CMOS DVM that receives data at V_{IN} and displays the data on the 7-segment local display unit. Data are transferred from the INS8294 to the INS8049 via National's MICROBUS™

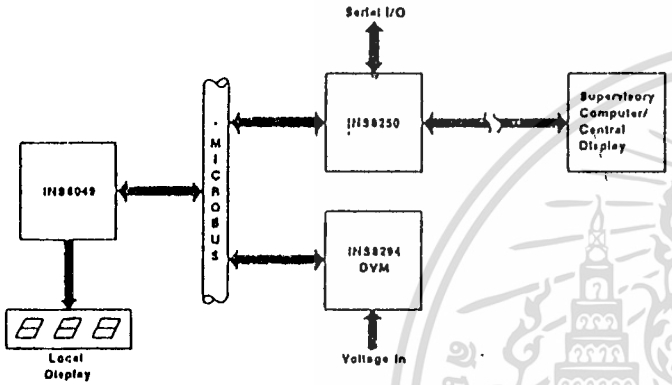


FIGURE 7. Data Acquisition System

TYPICAL APPLICATIONS (Cont'd)

Figure 8 shows a typical way to add a Input/Output Expander and Programmable Interval Timer to the 48-Series Microprocessors

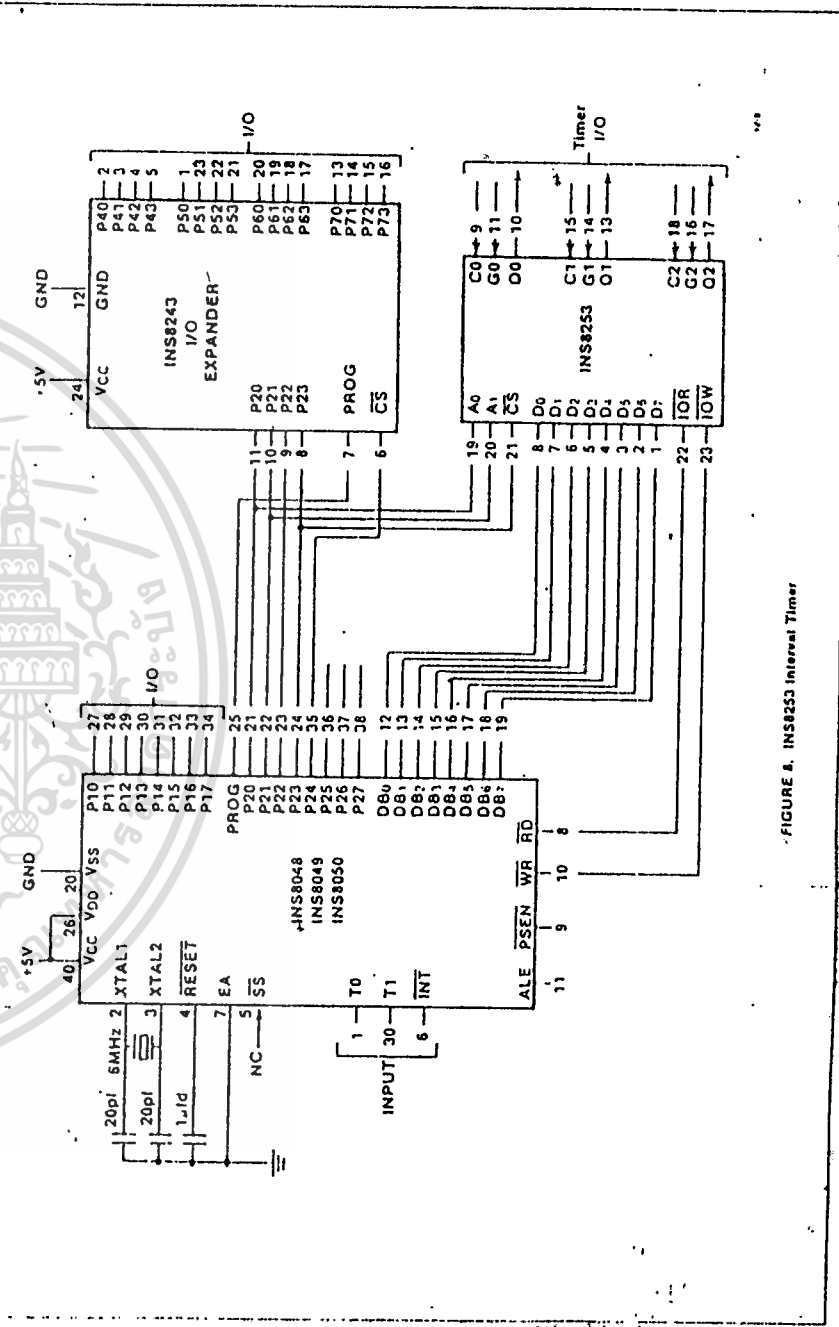


FIGURE 8. INS8253 Interval Timer

กิตติกรรมประกาศ

ปริญญานิพนธ์และโครงงานที่ได้จัดทำขึ้นในครั้งนี้สำเร็จลุล่วงลงได้ด้วยดีก็ด้วยความเมตตาจากอาจารย์ ขนิษฐา แซ่ตั้ง อาจารย์ที่ปรึกษา ที่ให้แนวคิดในการทำโครงงาน และให้คำแนะนำปรึกษา ตลอดจนช่วยจัดหาเครื่องมือที่จำเป็นในการทำโครงงาน รวมทั้งเป็นธุระในด้านจัดหางบประมาณ กระผมขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี่ด้วย

ด้วยความซาบซึ้งอย่างยิ่ง

นาย ชาย แซ่เล้า

นาย ทรงพล น้ำแก้ว

บรรณานุกรม

(1) TaKashi Kenjo, " steppping Motor and their Microprocessor Control ", Elarendon Press, Oxford, 1986

(2) Barry B. Brey , " The Z80 Microprocessor ", Prentice-Hall International , 1988

(3) สุนทร วิทูสุรพจน์ , " การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051 ", ซีเอ็ดยูเคชั่น , 2537

(4) ไชยันต์ สุวรรณชีวะศิริ , " สเต็ปปิ้งมอเตอร์และการควบคุม " , วารสาร เซมิคอนดัคเตอร์ ฉบับ 116 หน้า 102-110

(5) สุพันธ์ อมรเชิดชูกิจ , " สเต็ปปิ้งมอเตอร์ " , วารสาร เซมิคอนดัคเตอร์ ฉบับ126 หน้า 83-89

(6) พิพัฒน์ เลาหสงคราม , " ไมโครคอนโทรลเลอร์ MCS-48 MCS-51 " , 2537