ระบบนิ้วหุ่นยนต์ควบคุมแบบ MASTER SLAVE

MASTER SLAVE ROBOT FINGERS SYSTEM

นายสมคิด แยงคำ

นายสราวุธ อิชยาวิโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีโทรคมนาคม ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

MASTER SLAVE ROBOT FINGERS SYSTEM

Mr. SOMKID YANGKÚM

Mr. SARAWUT ITCHAYAVIROJ

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF THE TECHNOLOGY TELECOMMUNICATION

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2000

หัวข้อปริญญานิพนธ์	ระบบนิ้วหุ่นยนต์ควบคุมแบบ MASTER SLAVE	
	MASTER SLAVE ROBOT FINGERS SYSTEM	
ชื่อนักศึกษา	นาย สมคิด แยงคำ	41013344
	นาย สราวุธ อิชยาวิโรจน์	41013350
อาจารย์ที่ปรึกษา	อ. บุณย์ชนะ ภู่ระหงษ์	
ปริญญา	อุตสาหกรรมศาสตร์บัณฑิต	
สาขาวิชา	เทคโนโลยีโทรคมนาคม	
ภาควิชา	เทคนิคอุตสาหกรรม	
ปีการศึกษา	2543	

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

_____ อาจารที่ปรึกษา
(อาจารย์บุณย์ชนะ ภู่ระหงษ์)

_____ กรรมการ
( )

_____ กรรมการ
( )

_____ กรรมการ
( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปริญญานิพนธ์	ระบบนิ้วหุ่นยนต์ควบคุมแบบ MASTER SLAVE
	MASTER SLAVE ROBOT FINGERS SYSTEM
ชื่อนักศึกษา	นาย สมคิด แยงคำ 41013344
	นาย สราวุธ อิชยาวิโรจน์ 41013350
อาจารย์ที่ปรึกษา	อ. บุณย์ชนะ ภู่ระหงษ์
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2543

บทคัดย่อ

โครงงานนี้จะนำเอาหลักการทำงานของ ไมโครคอนโทรลเลอร์ เป็นการควบคุมชุดสเต็ปปิ้งมอเตอร์ มาใช้ในการควบคุมการเคลื่อนที่ของนิ้วมือ ซึ่งเป็นที่มาของโครงงาน " นิ้วมือกล"

ในโครงงานนี้จะใช้สเต็ปปิ้งมอเตอร์เป็นตัวต้นกำลังในการขับเคลื่อนนิ้วโดยที่สเต็ปปิ้งมอเตอร์นี้โดนควบคุมความเร็วโดยไมโครคอนโทรลเลอร์จะเป็นตัวกำหนดพัลซ์ที่จะจ่ายให้กับสเต็ปปิ้งมอเตอร์และไมโครคอนโทรลเลอร์จะถูกควบคุมโดยวงจรอิเล็กทรอนิกส์อีกทีหนึ่ง โดยที่จะใช้ความต้านทานปรับค่าได้เป็นตัวตรวจสอบ และคอยควบคุมการเคลื่อนที่ของนิ้วมือ

Thesis Title	MASTER SLAVE ROBOT FINGERS SYSTEM
Student	Mr. Somkid Yangkum ID 41013344
	Mr. Sarawut Itchayaviroj ID 41013350
Advisor	Mr. Boonchana Phurahong
Academic year	2000

ABSTRACT

Concept of this project, master slave robot fingers system project, is take concept of microcontroller system for control the movement of fingers by stepping motor

In this project we use stepping motor to generate energy for moving the fingers. The stepping motor has controlled speed by microcontroller, This microcontroller to set pulse for the stepping motor and microcontroller has controlled by electronics circuit. We use variable resistor as a checker and control movement of the fingers.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาเกี่ยวกับข้อมูลของปริญญานิพนธ์ จาก อ. บุณย์ชนะ ภู่ระหงษ์ ซึ่งเป็นอาจารย์ที่ปรึกษาและอาจารย์ในภาควิชาเทคนิคอุตสาหกรรม คณะผู้จัดทำรู้สึกทราบซึ้งในความอนุเคราะห์จากทุกท่าน และขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ พี่ยุ้ย ศิษย์เก่าพระจอมเกล้าฯ ลาดกระบังที่ช่วยในการให้ข้อมูลเกี่ยวกับเนื้อหาของปริญญานิพนธ์ และเพื่อนเอ๋ที่เป็น อ. คณะวิทยาศาสตร์ที่ให้ยืมอุปกรณ์ในการทำโครงงานนี้ ซึ่งมีส่วนทำให้โครงงานนี้สำเร็จลุล่วงลงได้

คุณค่าและประโยชน์อันพึงมีจากคู่มือฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

นายสมคิด แยงคำ
นายสราวุธ อิชยาวิโรจน์

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

ปัจจุบันได้มีการนำเอาไมโครโพรเซสเซอร์มาใช้งานกันอย่างกว้างขวาง ส่วนใหญ่จะนำมาใช้ควบคุมอุปกรณ์ต่างๆ อาทิเช่น เตาอบไมโครเวฟ วิทยุติดรถยนต์ วิทยุมือถือ ตลอดจนถึงรีโมทของเครื่องปรับอากาศ เหตุที่นำมาประยุกต์เพราะมีขีดความสามารถสูง ทำงานได้รวดเร็ว มีขนาดเล็ก การเปลี่ยนแปลงฟังก์ชันในการทำงานเพียงแต่เปลี่ยนโปรแกรมในการทำงานเท่านั้นก็สามารถเปลี่ยนแปลงการทำงานได้ โดยไม่ต้องเปลี่ยนแปลงทางด้านวงจรเลย (ฮารด์แวร์) จึงทำให้ไม่สิ้นเปลือง โครงสร้างไมโครโปรเซสเซอร์ ที่นำมาใช้ในการทำงานจะต่อร่วมกับหน่วยความจำ และอุปกรณ์อินพุทเอ้าท์พุทพอร์ทหรือไม่ต่อก็ได้ ขึ้นอยู่กับการใช้งาน

ปัจจุบันได้มีการรวมวงจรทั้งหมดไว้ในชิปเดี่ยวขนาด 40 ขาและ 20 ขา ซึ่งทำให้ลักษณะงานของการควบคุมเฉพาะอย่างของเครื่องมือ หรือเครื่องมือวัดเฉพาะอย่างนั้นไม่จำเป็นต้องใช้องค์ประกอบที่มีอยู่ในไมโครคอมพิวเตอร์ทั้งหมด จึงทำให้มีการนำเอาหน่วยต่างๆที่มีองค์ประกอบที่จำเป็นเฉพาะในการทำงานควบคุมชนิดนั้นๆ มารวมเป็นหน่วยเดียวกัน ทำให้ขนาดลดลงเหลือขนาดเท่ากับชิปไมโครโปรเซสเซอร์ทั่วๆไป ที่มีขนาดตั้งแต่ 20-62 ขา เป็นชิปตัวเดียวที่มีลักษณะการใช้งาน ส่วนใหญ่จะใช้ในด้านการควบคุมงานแบบอัตโนมัติต่างๆ ที่ยังคงมีลักษณะการทำงานเช่นเดียวกับไมโครคอมพิวเตอร์ แต่เมื่อมีการนำไปใช้งานด้านการควบคุมเป็นหลัก จึงมีชื่อเรียกใหม่ตามการใช้งานว่า "ไมโครคอนโทรลเลอร์" (MICROCONTROLLER) หรือโดทั่วๆไปเรียกว่า "ซิงเกิลชิป" (SINGLE CHIP)

1.1 ลักษณะทั่วไปของไมโครคอนโทรเลอร์ (MICROCONTROLLER)

1. สร้างโดยใช้ซีเอ็ชมอส (CHMOS) เทคโนโลยีการทำงานด้วยแหล่งจ่ายไฟขนาด 5 โวลท์ (V) เพียงแหล่งเดียว
2. ซีพียูมีขนาด 8บิท
3. มีวงจรออสซิลเลเตอร์และนาฬิกาบนชิป
4. ชุดแบงค์ (BANK)รีจิสเตอร์มี 4 ชุด แต่ละชุดมีรีจิสเตอร์ 8 ตัว
5. มีตัวจับเวลา/ตัวนับ ขนาด 16 บิต 2 จุด และสำหรับเบอร์ 8032/8052 มี 3 ชุด

6. มีพอร์ตรับข้อมูล และส่งข้อมูลออกแบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิต รวมทั้งหมดเป็น 32 เส้น แต่จะเหลือเพียง 16 เส้น สำหรับเบอร์ 8031 อีก 16 เส้น ใช้ในการเข้าถึงทางแอดเดรสและข้อมูล และ 15 เส้น สำหรับเบอร์ 89C2051 (20 ขา)
7. พอร์ตแบบอนุกรมสามารถที่จะโปรแกรมการรับส่งได้พร้อมกัน (Full Duplex) ที่ความเร็วสูง
8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1 ไมโครวินาที ด้วยการใช้คริสทัส 12 เมกะเฮิรตซ์
9. แอดเดรสข้อมูลภายนอกได้ 64 กิโลไบต์
10. แอดเดรสโปรแกรมภายรนอกได้ 64 กิโลไบต์
11. สามารถกำหนดเลขที่อยู่ข้อมูลขนาดไบต์หรือบิตได้โดยตรง
12. มีซอฟต์แวร์แฟล็กสำหรับสำผู้ใช้ที่จะกำหนดเองได้ถึง 128 ตำแหน่งบิต
13. โครงสร้างอินเตอร์รัพต์ทำได้ 5 แหล่ง และ 6 แหล่ง สำหรับ 8092/8052 พร้อมด้วยการจัดไพรโอริตี้ (Priority) ได้ 2 ระดับ
14. ตัวโพรเซสเซอร์สามารถใช้งานแบบบูลลีน (Boolean) ได้เหมาะสำหรับการใช้ควบคุม
15. มีคำสั่งคูณ และหารทางฮาร์ดแวร์ทำได้ภายใน 4 ไมโครวินาที
16. ตัวเลขทางคณิตศาสตร์ ใช้ได้ทั้งแบบไบนารี และเดซิมอล
17. การใช้พื้นที่สแต็กสำหรับโปรแกรมย่อยต่างๆ ทำได้ง่ายและกว้าง
18. ชุดคำสั่งของ8051 (MCS-51) โดยการกำหนดเลขที่ (ADDRESSING MODE) ได้มากกว่าชุดคำสั่งของ 8051 (MCS-51)

ตระกูล 8051 จะมีทั้งแบบมีหน่วยความจำ (ROM) ในตัว หรือไม่มี (ROM) หรือมี (EPROM) บนชิปเดียวกัน และจะมีตำแหน่งขาที่เหมือนกัน ตารางที่ 1 แสดงถึงรายละเอียดของเบอร์ต่างๆ ในตระกูล 8051 (MCS-51) ที่มีจำหน่ายในท้องตลาด

ตารางที่ 1.1 รายละเอียดของตระกูล 8051 (MCS-51)

เบอร์	หน่วยความจำภายใน		ตัวตั้งเวลา	อินเตอร์รัพต์
	โปรแกรม	ข้อมูล Inter RAM	ตัวนับจำนวน	
8052 AH	8K × 8 ROM	256 × 8-bit	3 × 16 bit	6
8051 AH	4K × 8 ROM	128 × 8-bit	2 × 16 bit	5
8051	4K ×8 ROM	128 × 8-bit	2 × 16 bit	5
8031 AH	ไม่มี ROM	128 × 8-bit	2 × 16 bit	5
8751 H	4K × 8 EPROM	128 × 8-bit	2 × 16 bit	5
89C2051	2K × 8 EPROM	128 × 8-bit	2 × 16 bit	5

1.2 ลักษณะทั่วไปของ มอเตอร์สเต็ป(STEPPING MOTOR)

มอเตอร์สเต็ป (Stepping Motor) เป็นมอเตอร์ที่หมุนที่ละสเต็ปโดยแต่ละสเต็ป มอเตอร์จะหมุนด้วยมุมที่ค่าคงที่ค่าหนึ่ง ซึ่งในการควบคุมการหมุนของมอเตอร์สเต็ป (Stepping Motor) นั้นจะอาศัยวงจรควบคุมทางดิจิตอล โดยวงจรทางดิจิตอลนี้จะทำหน้าที่ในการจัดลำดับการกระตุ้น ในแต่ละเฟสของมอเตอร์สเต็ป (Stepping motor) ซึ่งจะทำให้สามารถกำหนดทิศทางการหมุน ความเร็วในการหมุนและตำแหน่งที่ต้องการจะเลื่อนไปของมอเตอร์สเต็ป (Stepping Motor) ได้อย่างถูกต้องแม่นยำ

เนื่องจากวงจรดิจิตอลที่ใช้ในการควบคุมการหมุนของมอเตอร์สเต็ป (Stepping Motor) สามารถกำหนดความเร็วการหมุน และตำแหน่งที่ต้องการจะเลื่อนไปของมอเตอร์สเต็ป (Stepping Motor) ได้อย่างถูกต้องแม่นยำ ดังนั้นจึงไม่จำเป็นต้องมีการส่งค่ากลับ (Feedback) เพื่อควบคุมความเร็ว และตำแหน่งของมอเตอร์สเต็ป เพราะฉะนั้นอาจจะกล่าวได้ว่าระบบควบคุมการหมุนเป็นแบบระบบเปิดวงจร (open loop control system) แต่ในการหมุนของมอเตอร์สเต็ป นั้นมีข้อเสียคือในบางช่วงของความเร็วในการหมุนอาจจะทำให้เกิดการออสซิเลทที่แกนเพราของมอเตอร์ได้ และอาจจะทำให้มอเตอร์เกิดการออสซิเลทได้

โครงงานนี้ได้นำเอาไมโครคอนโทรลเลอร์เข้ามาควบคุมการหมุนของมอเตอร์สเต็ป (Stepping motor) โดยที่มอเตอร์สเต็ป นี้จะเป็นตัวต้นกำลังในการขับเคลื่อนของนิ้วมือแต่ละนิ้ว โดยการเขียนโปรแกรมเข้าไปที่ไมโครคอนโทรลเลอร์ ให้ทำงานตามที่ต้องการ

1.3 การประยุกต์ใช้งานของมอเตอร์สเต็ป (Stepping motor)

สำหรับการประยุกต์ใช้งานในโครงงานนี้ ได้มีการควบคุมมอเตอร์สเต็ป (Stepping motor) ให้มีการเคลื่อนที่ได้ตามที่เขียนโปรแกรมไว้ในตัวไมโครคอนโทรลเลอร์ (MCS-51) และจะควบคุมการหมุนของมอเตอร์สเต็ป (Stepping motor) เมื่อมอเตอร์หมุนก็จะหมุนแกนเหล็กดันหัวน็อต ทำให้ข้อมืองอลงหรือดันตรงขึ้นตามการควบคุม จึงเกิดการเคลื่อนที่ของนิ้วมือได้

บทที่ 2
ทฤษฎีการทำงาน

2.1 ไมโครคอนโทรลเลอร์ตระกูล 8051 (MCS-51)

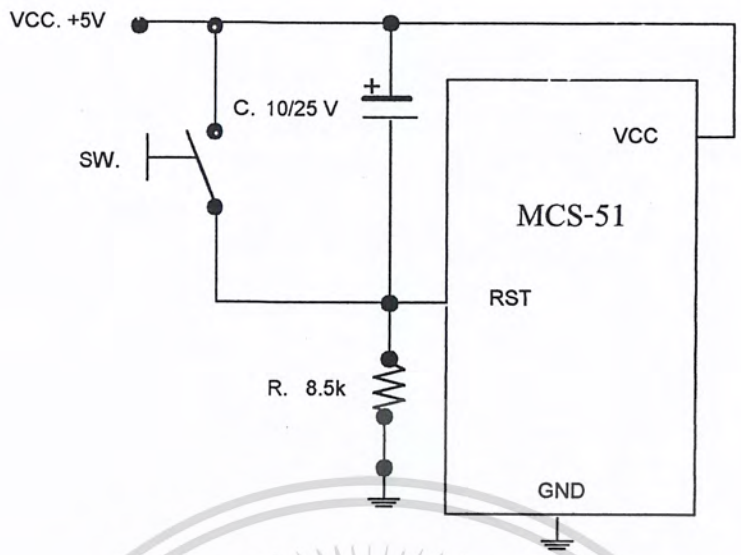
ไมโครคอนโทรลเลอร์เป็นไมโครโปรเซสเซอร์ประเภทหนึ่งที่ได้รับการออกแบบมาเพื่อใช้งานกับระบบควบคุมที่มีขนาดเล็ก โดยที่ภายในไอซีไมโครคอนโทรลเลอร์หนึ่งตัวจะประกอบด้วยหน่วยการทำงานหลักของระบบคอมพิวเตอร์ครบถ้วน เช่น หน่วยประมวลผลกลาง (CPU) หน่วยความจำ (RAM,ROM) พอร์ตในการติดต่อหรือควบคุมอุปกรณ์ต่างๆ (I/O PORT) เป็นต้น หากเป็นการใช้งานไมโครโปรเซสเซอร์ทั่วไปก็จะต้องใช้ไอซีภายนอกมาประกอบเพื่อทำหน้าที่เหล่านี้ ดังนั้นจึงอาจกล่าวได้ว่าไมโครคอนโทรลเลอร์ (MICROCONTROLER) เป็นระบบคอมพิวเตอร์เพื่องานควบคุมที่สมบูรณ์แบบ โดยบรรจุอยู่ภายในไอซีเพียงหนึ่งตัวเท่านั่น บางครั้งจึงมีการเรียกไมโครคอนโทรลเลอร์ว่าไมโครคอมพิวเตอร์ชิปเดี่ยว (Single chip microcomputer)

ไมโครคอนโทรลเลอร์ตระกูล8051 (MCS-51) ประกอบไปด้วย ไมโครคอนโทรลเลอร์หลายรุ่น ซึ่งมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาดหรือจำนวนของหน่วยทำงานภายในที่ต่างกันออกไป เพื่อความเหมาะสมในงานประยุกต์ต่างๆตามความต้องการ โดยจะมีทั้งลักษณะที่ใช้เทคโนโลยีการผลิตไอซีวงจรรวมความจุสูงมาก (LSI) แบบเอ็ชมอส (HMOS) หรือซีเอ็ชมอส (CHMOS) ซึ่งมีคุณลักษณะความจุสูงมากขึ้น และสิ้นเปลืองกำลังไฟฟ้าน้อยกว่ามาก

2.2 โครงสร้างภายในของ8051 (MCS-51)

โครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ ตระกูล 8051 มีดังแสดงในรูปที่ 2.2 โครงสร้างหน่วยความจำภายใน8051 (MCS-51) ไมโครคอนโทรลเลอร์ในตระกูล8051 (MCS-51) ทุกเบอร์จะแบ่งหน่วยความจำออกเป็นสองส่วน คือ

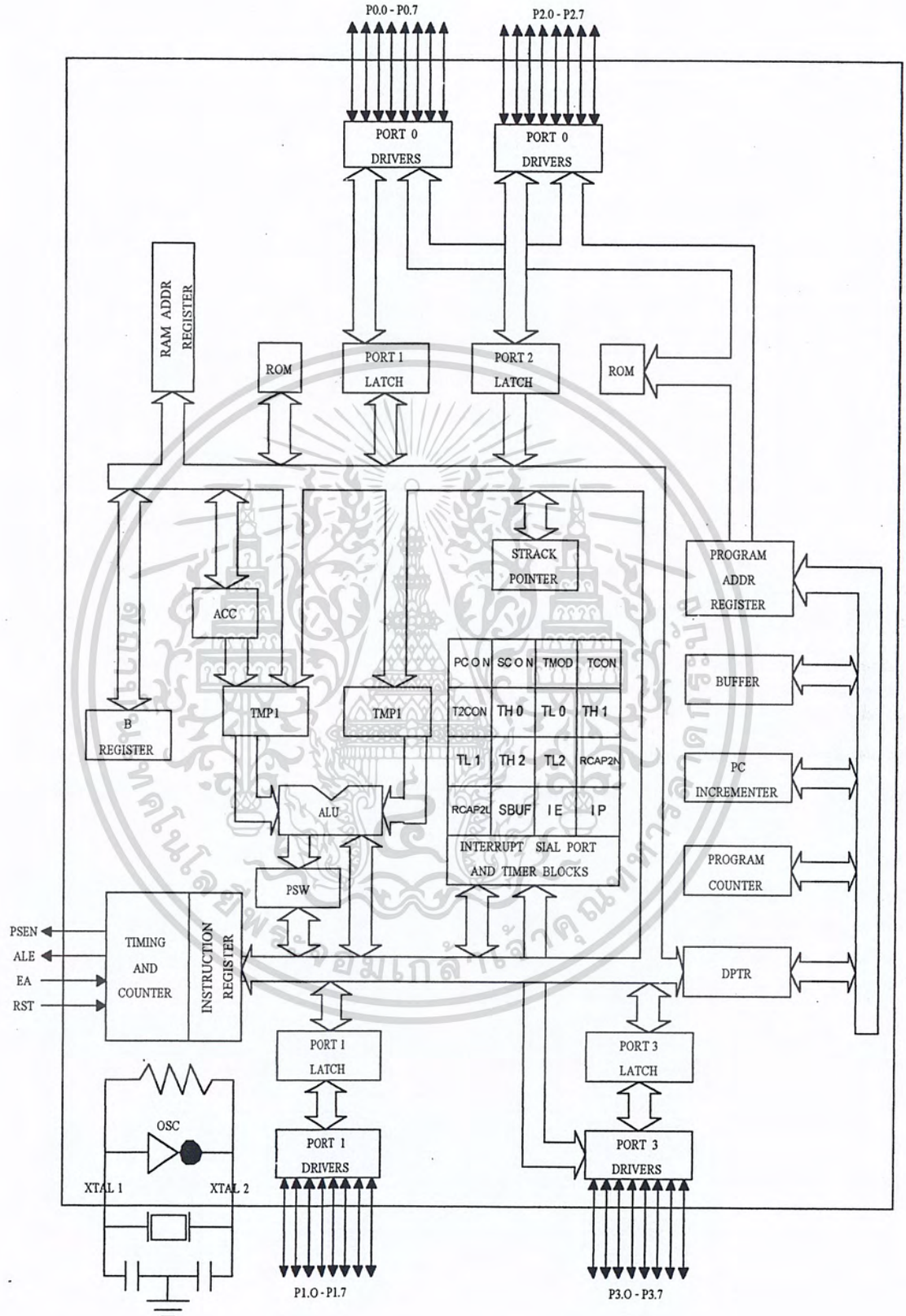
- หน่วยความจำสำหรับเก็บโปรแกรม (program memory)
- หน่วยความจำสำหรับเก็บข้อมูล (data memory)



รูปที่ 2.1 แสดงวงจรสำหรับรีเซตชิปไมโครคอนโทรลเลอร์ 8051 (MCS-51)

หน่วยความจำสำหรับเก็บโปรแกรมจะใช้เก็บโปรแกรมควบคุมการทำงานของชิป8051 (MCS-51) บางเบอร์จะมีหน่วยความจำส่วนนี้อยู่ในชิป แต่บางเบอร์จะไม่มี ทำให้ต้องเก็บโปรแกรมไว้ในหน่วยความจำภายนอกทั้งหมด ส่วนหน่วยความจำหน่วยที่สองคือ หน่วยความจำสำหรับเก็บข้อมูล ซึ่งใช้สำหรับเก็บข้อมูลระหว่างการทำงาน8051 ทุกเบอร์จะมีหน่วยความจำส่วนนี้อยู่ภายในชิปจำนวนหนึ่ง แต่จะมีจำนวนมากน้อยเท่าใดขึ้นกับเบอร์ของชิป

8051 (MCS-51) เป็นตระกูลของไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้นมาจากตระกูล48 (MCS-48) ดังนั้นจึงมีความสามารถเหนือกว่าหลายอย่าง จะเปรียบเทียบข้อดีของ 8051 (MCS-51) เมื่อเทียบกับตระกูล48 (MCS-48) ให้เห็นเป็นบางช่วง เช่น ความเร็วในการประมวลผลของ 8051 สามารถใช้ความถี่ได้ถึง 12 เมกะเฮิรตซ์ หรือสำหรับบางเบอร์ในตระกูลสามารถใช้ได้ถึง 16 เมกะเฮิรตซ์ ทำให้ช่วงเวลาการทำงานแต่ละคำสั่งใช้น้อยมาก เมื่อใช้ความถี่ 12 เมกะเฮิรตซ์ คำสั่งที่ใช้เวลาน้อยที่สุดจะใช้เวลาเพียง 1 ไมโครวินาที ส่วนคำสั่งที่ใช้เวลามากที่สุดจะใช้เวลาเพียง 4 ไมโครวินาทีเท่านั้น



รูปที่ 2.2 แสดงโครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ 8051 (MCS-51)

2.2.1 หน่วยความจำสำหรับเก็บโปรแกรม

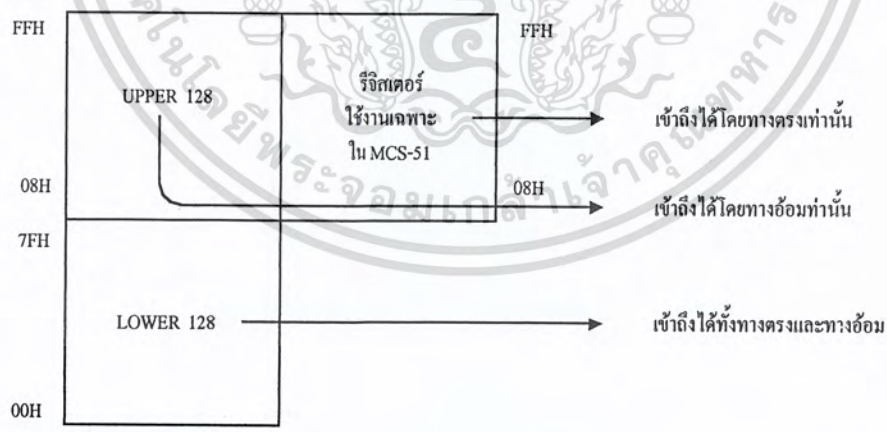
จะแบ่งเป็นสองส่วนคือ หน่วยความจำสำหรับเก็บโปรแกรมภายในชิป (Internal program memory) และหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป (external program memory) ขนาดของหน่วยความจำที่ใช้เก็บโปรแกรมภายในชิปมีได้ตั้งแต่ 0,4,8,16 กิโลไบต์ ขึ้นอยู่กับเบอร์ของชิป

2.2.2 หน่วยความจำสำหรับเก็บข้อมูลของ8051 (MCS-51)

จะแบ่งออกเป็นสองส่วนคือ หน่วยความจำสำหรับเก็บข้อมูลภายในชิป และหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป หน่วยความจำสำหรับเก็บข้อมูลภายในชิปของ 8051 ยังแบ่งออกเป็นสองส่วนย่อยดังนี้

- ส่วนที่ใช้เก็บข้อมูลทั่วไป (Internal ram)
- ส่วนที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะ (Special function register)

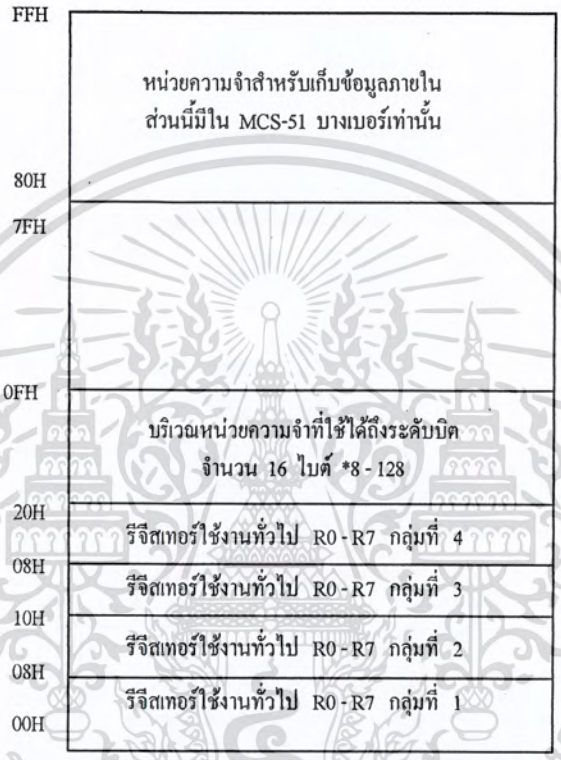
หน่วยความจำส่วนที่ใช้เก็บข้อมูลทั่วไปภายในชิปเป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ภายใน8051 (MCS-51) หน่วยความจำส่วนนี้มีไว้สำหรับเก็บข้อมูลในขณะทำงาน ส่วนหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะเป็นหน่วยความจำสำหรับเก็บข้อมูลภายใน8051 (MCS-51) ซึ่งถูกกำหนอให้เป็นรีจิสเตอร์ใช้งานเฉพาะเพื่อควบคุมการทำงาน



รูปที่ 2.3 แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิป 8051 (MCS-51)

และบอกสถานะของซีพียู (CPU) แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิปทั้งสองบริเวณมีดังในรูปที่ 2.3

8051 (MCS-51) ทุกเบอร์จะมีหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปอย่างน้อย 128 ไบต์ไปจนถึง 256 ไบต์ทั้งนี้ขึ้นอยู่กับเบอร์ของชิป หน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์แรกมีชื่อเรียกว่าโลเวอร์ (lower 128) และในบริเวณ 128 ไบต์หลังที่มีเพิ่มบางเบอร์มีชื่อเรียกว่าอัปเปอร์ (upper 128) ดังแสดงในรูปที่ 2.3



รูปที่ 2.4 แสดงหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปทั้งสองส่วน

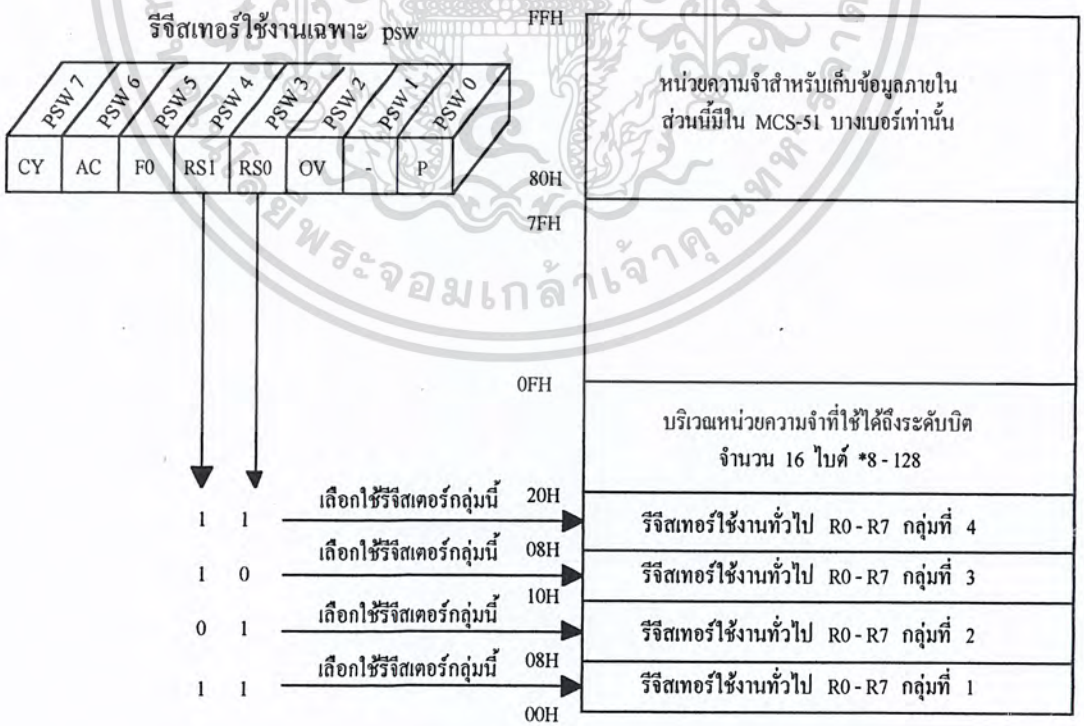
2.2.3 รีจิสเตอร์ใช้งานเฉพาะ

เนื่องจาก8051 (MCS-51) ถูกออกแบบไว้ใช้สำหรับใช้ควบคุมโดยเฉพาะ จึงทำให้มีความสามารถเฉพาะตัวหลายๆ อย่าง ซึ่งจำเป็นต้องอาศัยวงจรภายในชิปที่มีเพิ่มขึ้นจากไมโครโปรเซสเซอร์ทั่วไป การควบคุมการทำงานของวงจรภายในไมโครโพรเซสเซอร์จะกระทำผ่านรีจิสเตอร์ที่ถูกกำหนดหน้าที่ไว้แล้ว ดังนั้นหากต้องการใช้ 8051 ให้มีประสิทธิภาพ จำเป็นต้องทราบหน้าที่การทำงาน ของรีจิสเตอร์ใช้งานเฉพาะแต่ละตัวให้ละเอียด รีจิสเตอร์ใช้งานเฉพาะทั้งหมดจะอยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปบริเวณที่ใช้รีจิสเตอร์ใช้งานเฉพาะดังได้กล่าวมาแล้ว รีจิสเตอร์ใช้งานเฉพาะทั้งหมดใน8051 (MCS-51)

ในส่วนของหน่วยความจำสำหรับเก็บโปรแกรม และหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิป จะเป็นหน่วยความจำส่วนที่อยู่ภายนอกชิป8051 (MCS-51) ซึ่งผู้ใช้ต้องติดตั้งเพิ่มเอง การติดต่อระหว่าง 8051 กับหน่วยความจำทั้งสองส่วนจะใช้ขา 32 ถึง 39 (พอร์ต 0) เป็นตัวส่งค่าแอดเดรสไบต์ต่ำ (A0-A7) และใช้รับส่งข้อมูกับหน่วยความจำด้วย (ใช้เป็นดาต้าบัส) ส่วนค่าแอดเดรสไบต์สูง (A8-A15) จะใช้ขา 21-28 (พอร์ต 2) ดังนั้นเมื่อพอร์ต 0 และพอร์ต 2 ถูกใช้ในการติดต่อกับหน่วยความจำภายนอก (ทั้งหน่วยความจำสำหรับเก็บโปรแกรมและหน่วยความจะสำหรับเก็บข้อมูล) จะทำให้เหลือพอร์ตสำหรับใช้งานอื่นๆน้อยลง

2.2.4 รีจิสเตอร์สำหรับใช้งานทั่วไป

8051 จะมีรีจิสเตอร์ใช้งานทั่วไปที่ผู้เขียนเขียนโปรแกรมสามารถนำมาใช้งานได้คือ รีจิสเตอร์ A, B (อยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะแต่นับเป็นรีจิสเตอร์ใช้งานทั่วไปเพราะไม่ถูกกำหนดหน้าที่ที่ใช้งานโดยตรง) และรีจิสเตอร์ใช้งานทั่วไป (R0-R7) มีอยู่ด้วยกันทั้งหมด 4 กลุ่ม แต่ละกลุ่มประกอบด้วยรีจิสเตอร์จำนวน 8 ตัว (R0-R7) ซึ่งมีชื่อเรียกเหมือนกัน ดังนั้นจำนวนรีจิสเตอร์ทั่วไป กลุ่มใดกลุ่มหนึ่งใน 4 กลุ่มกระทำโดยการเซตหรือเคลียร์บิต (RS0, RS1) ในรีจิสเตอร์ในงานเฉพาะ (PSW) ดังแสดงในรูป ที่ 2.5



รูปที่ 2.5 แสดงการเลือกรีจิสเตอร์ใช้งานทั่วไป (R0-R7) แต่ละกลุ่ม

รีจิสเตอร์ใช้งานทั่วไป (R0-R7) จะมีอยู่ในกลุ่มรีจิสเตอร์ใช้งานทั่วไปทั้ง 4 กลุ่ม ซึ่งจะถูกเลือกใช้งานเพียงกลุ่มเดียวในขณะใดขณะหนึ่ง ค่าที่เปลี่ยนแปลงไปภายในรีจิสเตอร์ใช้งานทั่วไปที่ถูกเลือกใช้งานในขณะนั้นจะไม่มีผลต่อรีจิสเตอร์ใช้งานทั่วไปที่มีชื่อเดียวกันแต่อยู่คนละกลุ่มเลย โครงสร้างเช่นนี้ทำให้มีความสะดวกในการเขียนโปรแกรมเป็นอันมากโดยเฉพาะการเขียนโปรแกรมที่มีการเขียนโปรแกรมย่อย ดังจะได้กล่าวถึงในเรื่องของการเขียนโปรแกรมต่อไป

2.3 รายละเอียดของพอร์ตใน8051 (MCS-51)

จะกล่าวถึงโครงสร้างรวมทั้งรายละเอียดของการทำงานของวงจรที่อยู่ภายใน8051 (MCS-51) ซึ่งประกอบไปด้วย

- โครงสร้างและการทำงานของพอร์ตใน 8051 (port structure)
- โครงสร้างและการทำงานของไทม์เมอร์/เคาน์เตอร์
- โครงสร้างและการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรม
- โครงสร้างอินเตอร์รัปต์ใน 8051 (interrupt structure)

เนื่องจาก 8051 แต่ละเบอร์มีลักษณะโครงสร้างไม่เหมือนกันทีเดียว การอธิบายโดยกล่าวแต่ละเบอร์เฉพาะเจาะจงลงไปจะทำให้เกิดความสับสนและยากในการอ้างชื่อ เพื่อหลีกเลี่ยงปัญหาเหล่านี้จำเป็นต้องกล่าวถึง 8051 เบอร์ที่มีลักษณะคล้ายกันโดยรวม ดังนั้นเราจะแทนชื่อ 8051s และ 8052s ด้วยเบอร์ที่มีลักษณะคล้ายคลึงกันจนกว่าจะมีการกล่าวถึงตัวใดตัวหนึ่งเฉพาะลงไป

8051s จะประกอบด้วยเบอร์ 8051, 8051AH และ 80C51BH (เป็น MCS-51 เบอร์ที่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป) และหน่วยความจำ (EPROM version) คือเบอร์ 8751 (เป็น MCS-51 เบอร์ที่มีหน่วยความจำสำหรับเก็บโปรแกรมเป็น EPROM อยู่ภายในชิป)

8052s ประกอบด้วย 8052AH, 8032AH และ 8752BH

2.3.1 โครงสร้างและการทำงานของพอร์ต

โครงสร้างภายใน 8051 ที่เราจะศึกษาเป็นอันดับแรกคือพอร์ตใน 8051 ซึ่งมีพอร์ตขนาด 8 บิตรวมทั้งสิ้น 4 พอร์ต พอร์ตแต่ละพอร์ตมีหน้าที่ติดต่อกับวงจรภายนอก โดยสามารถรับและส่งข้อมูลกับวงจรภายนอกได้ กล่าวคือ ทั้ง 4 พอร์ตจะเป็นพอร์ตจะเป็นพอร์ตสองทิศทาง (bidirectional port) โครงสร้างของแต่ละพอร์ตจะประกอบด้วย

1. วงจรแลตช์ซึ่งสถานะของเอาต์พุตของวงจรแลตช์จะปรากฏที่รีจิสเตอร์ใช้งานเฉพาะ P0 ถึง P3
2. วงจรเอาต์พุตไดรเวอร์
3. วงจรอินพุตบัฟเฟอร์

เอาต์ไดรเวอร์ของพอร์ต 0 และพอร์ต 2 กับอินพุตบัฟเฟอร์ของพอร์ต 0 จะถูกใช้ติดต่อกับหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป โดยมีการทำงานดังนี้

พอร์ต 0 ใช้ส่งค่าแอดเดรสไบต์ต่ำ (Low byte – [A0-A7]) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอกชิป) และยังถูกใช้เป็นดาต้าบัส (data bus) ซึ่งจะใช้ในการรับส่งข้อมูลกับหน่วยความจำภายนอก โดยจะผลัดกันใช้คนละเวลา (time multiplex)

พอร์ต 2 ใช้ส่งค่าแอดเดรสไบต์สูง (High byte – [A8-A15]) ของตำแหน่งหน่วยความจำภายนอก (หน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอกชิป) เมื่อมีการใช้แอดเดรสขนาด 16บิต และถ้าพอร์ต 2 ไม่ถูกใช้งาน (ติดต่อหน่วยความจำภายนอกน้อยกว่า 256 ไบต์) คือถูกใช้ไม่

ครบ 16 บิต (ใช้หน่วยความจำภายนอกไม่ครบ 64 กิโลไบต์) พอร์ต 2 บิตที่ไม่ได้ใช้ก็จะส่งค่าภายในรีจิสเตอร์ใช้งานเฉพาะ พอร์ต 2 (P2) ออกมาที่แต่ละขา

พอร์ต 3 ทั้งหมดรวมถึงพอร์ต 1 ของ 8052 อีก 2 ขา แต่ไม่เพียงจะใช้เป็นพอร์ตอินพุตเอาต์พุตเท่านั้น เพราะแต่ละบิตของพอร์ต 3 และพอร์ต 1 บางบิตอาจมีหน้าที่พิเศษอย่างอื่นอีก ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แสดงหน้าที่พิเศษอื่นๆของพอร์ต 1 และพอร์ต 3

พอร์ต	หน้าที่
P1.0 (T2)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทม์เมอร์
P1.1(T2EX)	ใช้เป็นสัญญาณควบคุมการทำงานของไทม์เมอร์ 2 รายละเอียดจะกล่าวต่อไป
P3.0(RXD)	ใช้เป็นอินพุตของพอร์ตสื่อสารอนุกรม
P3.1(TXD)	ใช้เป็นเอาต์พุตของพอร์ตสื่อสารอนุกรม
P3.2(INT0)	ใช้เป็นอินพุตของอินเตอร์รัปต์ภายนอกชนิด 0
P3.3(INT1)	ใช้เป็นอินพุตของอินเตอร์รัปต์ภายนอกชนิด 1
P3.4(T0)	ใช้เป็นอินพุตของเคาน์เตอร์ของไทม์เมอร์ 0
P3.5(T1)	ใช้เป็นอินพุตให้เคาน์เตอร์ของไทม์เมอร์1
P3.6($\overline{WR}$)	ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอก
P3.7($\overline{RD}$)	ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำภายนอกเข้ามา

หน้าที่พิเศษเหล่านี้จะใช้ได้ก็ต่อเมื่อ ค่าในบิตแลตช์ของพอร์ตในบิตนั้นๆ มีค่าเป็น 1 มิฉะนั้นที่ของพอร์ตภายนอกชิปจะมีค่าเป็น 0

2.4 กลุ่มคำสั่ง

ก่อนที่จะเริ่มศึกษากลุ่มคำสั่งของ 8150 จำเป็นต้องทราบถึงวิธีการเข้าถึงข้อมูล ใน 8051 ก่อน เพราะคำสั่งส่วนใหญ่ต้องการข้อมูลประกอบการทำงาน และ 8051 เองก็มีวิธีการเข้าถึงข้อมูลได้หลายวิธี ดังนั้นสิ่งแรกที่ต้องการทราบก่อนจะเริ่มศึกษากลุ่มคำสั่งของ 8051 คือ วิธีการเข้าถึงข้อมูลดังมีรายละเอียดดังต่อไปนี้

คำสั่งต่างๆที่มีใน 8051 ส่วนใหญ่จะเป็นคำสั่งที่ต้องการข้อมูลมาประมวลผล เพื่อต้องการให้ได้ผลลัพธ์ แต่ยังมีคำสั่งอีกประเภทหนึ่งซึ่งไม่จำเป็นต้องอาศัยข้อมูลที่จะมาประมวลผล คำสั่งประเภทนี้จะเป็นคำสั่งที่ซีพียูสามารถปฏิบัติงานได้ด้วยตัวเอง กลุ่มคำสั่งทั้งหมดนั้นจึงสามารถแยกออกเป็นประเภทตามจำนวนข้อมูลที่ต้องการนำมาประกอบคำสั่งดังนี้

- คำสั่งที่ไม่ต้องการข้อมูลประกอบการทำงาน เช่น RET,RETI
- คำสั่งที่ต้องการข้อมูลเพื่อประมวลผล 1 ตัว เช่น INC Rn
- คำสั่งที่ต้องการข้อมูลเพื่อประมวลผล 2 ตัว เช่น MUL AB,DIV AB,ADD A,Rn
- คำสั่งที่ต้องการข้อมูลเพื่อประมวลผล 3 ตัว เช่น CJNE A,#data,rel

กลุ่มคำสั่งทั้งหมดใน8051 (MCS-51) สามารถแบ่งออกได้เป็น 5 กลุ่มย่อยตามลักษณะของการทำงานได้ดังนี้

- กลุ่มคำสั่งทางคณิตศาสตร์ (arithmetic instructions)
- กลุ่มคำสั่งทางตรรกศาสตร์ (logical instructions)
- กลุ่มคำสั่งเคลื่อนย้ายข้อมูล (data transfer instructions)
- กลุ่มคำสั่งควบคุมลำดับการทำงานของโปรแกรม (program control instructions)
- กลุ่มคำสั่งประมวลผลแบบบูลีน (boolean instructions)

ข้อมูลที่ถูกนำมาประมวลผลในคำสั่งที่ต้องการนำมาโอเปอร์แรนด์มีด้วยกันหลายชนิด แต่ละชนิดก็มีวิธีการเข้าถึงข้อมูลแตกต่างกันออกไป วิธีการในการเข้าถึงข้อมูล (addressing mode) เพื่อนำมาประกอบคำสั่งมีด้วยกันหลายวิธีดังนี้

2.4.1 วิธีเข้าถึงข้อมูลในคำสั่ง

1. วิธีการเข้าถึงข้อมูลโดยตรง วิธีนี้จะระบุค่าตำแหน่งหน่วยความจำที่เก็บข้อมูลโดยตรงในคำสั่ง ข้อมูลที่จะประมวลผลโดยวิธีนี้จะเป็นค่าของข้อมูลในหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไปที่ใช้บริเวณ 128 ไบต์ล่างและหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะเท่านั้นและเนื่องด้วยหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เก็บข้อมูลทั่วไปที่ใช้ในบริเวณ 128 ไบต์ล่างกับหน่วยความจำทีร่ใช้เก็บข้อมูลที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะมีขนาดรวมกันทั้งสิ้น 256 ไบต์

สิ่งที่ควรระวังในการเข้าถึงข้อมูลวิธีนี้คือ หน่วยความจำที่ใช้เก็บข้อมูลทั่วไปบริเวณ 128 ไบต์บนไม่สามารถใช้วิธีการเข้าถึงแบบนี้ได้

2. วิธีการเข้าถึงข้อมูลโดยทางอ้อม ค่าตำแหน่งหน่วยความจำที่ต้องการติดต่อจะเก็บไว้ในรีจิสเตอร์เฉพาะของคำสั่ง ดังนั้นวิธีนี้จึงถือเป็นวิธีการเข้าถึงข้อมูลโดยทางอ้อม นั่นคือ แทนที่ผู้เขียนโปรแกรมจะระบุค่าตำแหน่งของข้อมูลโดยตรง วิธีนี้จะใช้ค่าที่เก็บไว้ในรีจิสเตอร์ที่ระบุในรหัสคำสั่งให้ชี้ไปยังรหัสตำแหน่งของหน่วยความจำแทน รีจิสเตอร์ที่สามารถนำมาใช้เป็นตัวชี้ (Pointer) ตำแหน่งของหน่วยความจำมีดังต่อไปนี้

- รีจิสเตอร์ใช้งานทั่วไป (R0,R1) ของแต่ละกลุ่มรีจิสเตอร์ทั้ง 4
- รีจิสเตอร์ใช้งานเฉพาะ (SP:stack pointer)
- รีจิสเตอร์ใช้งานเฉพาะ DPTR (data pointer)

รีจิสเตอร์แต่ละตัวจะมีที่ใช้ในแต่ละคำสั่งไม่เหมือนกัน ดังจะได้กล่าวดังต่อไปนี้

2.4.1.1 รีจิสเตอร์ใช้งานทั่วไป (R0,R1)

เป็นรีจิสเตอร์ขนาด 8 บิต ดังนั้นรีจิสเตอร์ทั้งสองสามารถชี้ตำแหน่งหน่วยความจำได้เพียง 256 ตำแหน่งเท่านั้น ส่วนหน่วยความจำสำหรับเก็บข้อมูลที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะไม่สามารถใช้รีจิสเตอร์ทั้งสองชี้ตำแหน่งได้ และต้องมีเครื่องหมาย @ นำหน้ารีจิสเตอร์ R0 หรือ R1 ด้วย

ตัวอย่างคำสั่งที่ใช้ในรีจิสเตอร์ใช้งานทั่วไปเป็นตัวชี้ตำแหน่งข้อมูลมีดังนี้

ADD A,@R0 นำค่าในหน่วยความจำตำแหน่งที่ตรงกับค่าในรีจิสเตอร์ R0 มาบวกกับค่าในรีจิสเตอร์ A ผลลัพธ์ที่ได้เก็บไว้ในรีจิสเตอร์ A

2.4.1.2 รีจิสเตอร์ที่ใช้งานเฉพาะ (SP)

มีขนาด 8 บิต เหมือนรีจิสเตอร์ใช้งานเฉพาะทั่วไป R0, R1 แต่รีจิสเตอร์ใช้งานเฉพาะ (SP) ไม่สามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิป 8051 ได้ ในสภาวะถูกรีเซตหรือเริ่มจ่ายพลังงานให้แก่ 8051 ค่าในรีจิสเตอร์ใช้งานเฉพาะ (SP) มีค่าเท่ากับ 07H เสมอ

บริเวณของหน่วยความจำในรีจิสเตอร์ใช้งานเฉพาะ (SP) ชื้อยู่จะมีชื่อเรียกว่า สแตก ตัวรหัสคำสั่งของคำสั่งที่ใช้รีจิสเตอร์ใช้งานเฉพาะ (SP) เป็นตัวชี้ข้อมูลไม่จำเป็นต้องระบุตำแหน่งของรีจิสเตอร์ใช้งานเฉพาะ (SP) เพราะ 8051 ทราบเองจากรหัสคำสั่งจากตัวอย่าง

PUSH ACC นำค่าในรีจิสเตอร์ A ไปเก็บไว้ในสแตก สมมติในขณะนั้นรีจิสเตอร์ใช้งานเฉพาะ (SP) มีค่า 15H

2.4.1.3 รีจิสเตอร์ใช้งานเฉพาะ (DPTR)

เป็นรีจิสเตอร์ใช้งาน 16 บิต ดังนั้นรีจิสเตอร์ตัวนี้สามารถชี้ตำแหน่งหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิปได้มากถึง 64 กิโลไบต์

MOVX A,@DPTR นำค่าในหน่วยความจำตำแหน่งที่เท่ากับค่าในรีจิสเตอร์ (DPTR) ในขณะนั้นมาไว้ในรีจิสเตอร์ A

2.5 สเต็ปปิ้งมอเตอร์ (STEPPING MOTER)

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ชนิดหนึ่งที่ทำหน้าที่เปลี่ยนสัญญาณดิจิตอลทางไฟฟ้าไปเป็นการเคลื่อนที่ทางกล ดังนั้นการติดต่อกับอุปกรณ์ดิจิตอลเป็นไปโดยง่าย และวงจรขยายกำลังจากสัญญาณดิจิตอล (DIGITAL POWER AMPLIFIER) ที่ใช้ก็มีราคาถูกกว่าวงจรขยายกำลังเชิงเส้นอีกด้วย อีกทั้งการออกแบบวงจรควบคุมสเต็ปปิ้งมอเตอร์สามารถทำได้ง่ายกว่าวงจรควบคุมมอเตอร์แบบเซอร์โว และยังสามารถออกแบบวงจรสเต็ปปิ้งมอเตอร์ให้หยุดการทำงานได้อย่างทันทีทันใดอีกด้วย

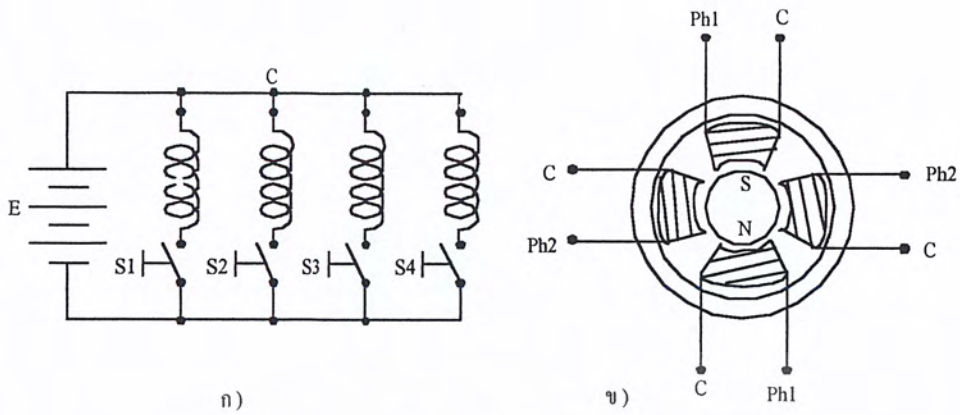
2.5.1 ชนิดของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์แบ่งตามมาตรฐานได้ 3 แบบ คือ

1. วาริเอเบิลรีลักแตนซ์ (VARIABLE RELUCTANCE : VR)
2. เพอร์มาเนนต์แมกเน็ต (PERMANENT MAGNET : PM)
3. ไฮบริด (HYBRID)

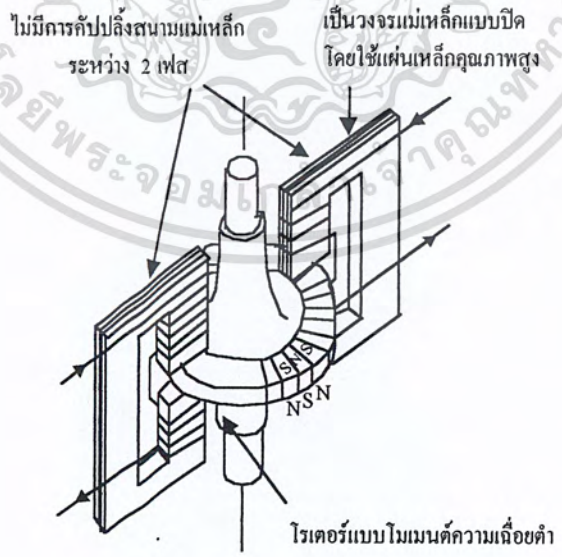
ชนิดวาริเอเบิลรีลักแตนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (MULTI TOOTH) ทำจากเหล็กอ่อนเราทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมากคือ ใช้นิ้วหมุนเพลาของมอเตอร์และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์ไม่เกิดปรากฏการณ์ทางแม่เหล็ก (MAGNATISM) มันจึงหมุนได้ตลอดโดยไม่ติดขัดแตกต่างจากชนิด PM และไฮบริดซึ่งมีสนามแม่เหล็กที่โรเตอร์เมื่อหมุนจะรู้สึกขัดๆ เหมือนเป็นฟันเฟือง สเต็ปในการหมุนสูง

ชนิดเพอร์มาเนนต์แมกเน็ตมีโครงสร้างแบบเรียบ ไม่มีซี่ขั้วแม่เหล็กและบนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยการป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้ว ซึ่งคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กถาวรจะถูกแรงดึงดูดจากขั้วแม่เหล็กที่อยู่บนสเตเตอร์ เมื่อถูกป้อนกระแสไฟฟ้าเข้าสู่ขดลวด และโรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป จึงทำให้เกิดแรงยึดหน่วงขึ้น สเต็ปปิ้งมอเตอร์ชนิดนี้จึงมีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น



รูปที่ 2.6 ก) วงจรกระตุ้นเฟสพื้นฐานสำหรับ PM มอเตอร์ 4 เฟส
ข) ภาพหน้าตัดของ PM มอเตอร์แบบ 4 เฟส

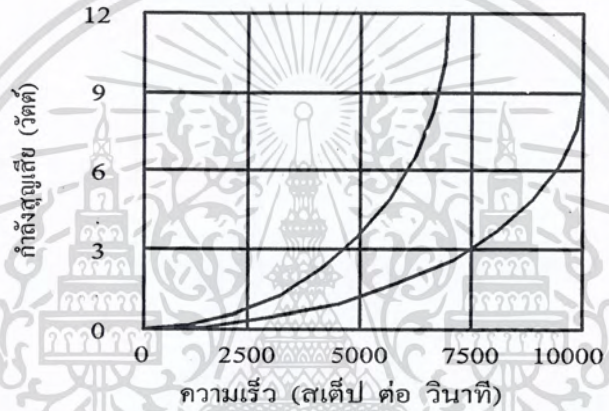
ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานมากที่สุด โดยเฉพาะนำมาใช้งานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ ชนิดมีโครงสร้างภายในซึ่งได้จาก การรวมเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากเพอร์มาเนนต์แมกเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ที่มีแรงยึดหน่วงสูง มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง



รูปที่ 2.7 แสดงโครงสร้างพื้นฐานของชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต

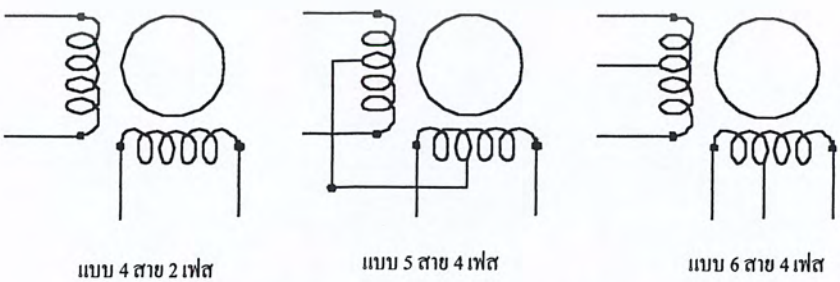
สเต็ปปิ้งมอเตอร์แบบใหม่อีกชนิดหนึ่งที่จะกล่าวถึงอีกเล็กน้อย คือ ชนิดที่ปรับปรุงมาจากชนิดเพอร์มาเนนต์แมกเน็ตนั่น คือ ชนิดแรเอิร์ธเพอร์มาเนนต์แม็กเน็ต ดังแสดงโครงสร้างในรูปที่ 2.7 หรือที่เรียกกันว่าชนิดดิสก์แมกเน็ตสเต็ปปิ้งมอเตอร์

โครงสร้างของโรเตอร์ของมอเตอร์ชนิดนี้มีลักษณะเป็นแผ่นซึ่งยึดติดกับเพลาของมอเตอร์ การทำงานของมอเตอร์ยังเป็นเช่นเดิม แต่ด้วยโครงสร้างที่เป็นแบบใหม่นี้จะทำให้เกิดโมเมนต์ของความเฉื่อยต่ำมาก, มีอัตราเร่งสูง และมันก็มีประสิทธิภาพสูงอีกหลายอย่างเช่น แรงบิดดึง, กำลังทางกลที่ได้ของมอเตอร์, ความถูกต้องของตำแหน่งสูงมาก และความเร็วในการเริ่มหมุนและหยุดสูง อีกทั้งยังมีความสูญเสียของกำลังต่ำ ดังแสดงในรูปที่ 2.8



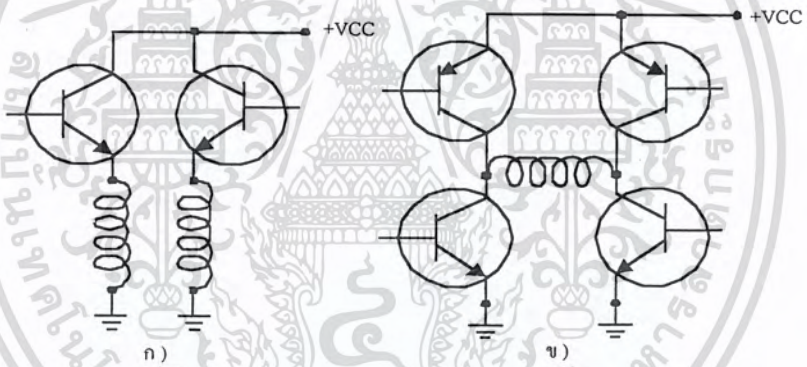
รูปที่ 2.8 แสดงกราฟข้อมูลการสูญเสียกำลังงานไฟฟ้าและความเร็วในการหมุนโดยเปรียบเทียบระหว่างสเต็ปปิ้งมอเตอร์ชนิดไฮบริดและชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต

การพันขดลวดหรือคอยล์บนสเต็ปปิ้งมอเตอร์มีอยู่ 2 วิธีคือ แบบไบโพลาร์ (BIPOLAR) และแบบยูนิโพลาร์ (UNIPOLAR) ดังแสดงในรูปที่ 2.9



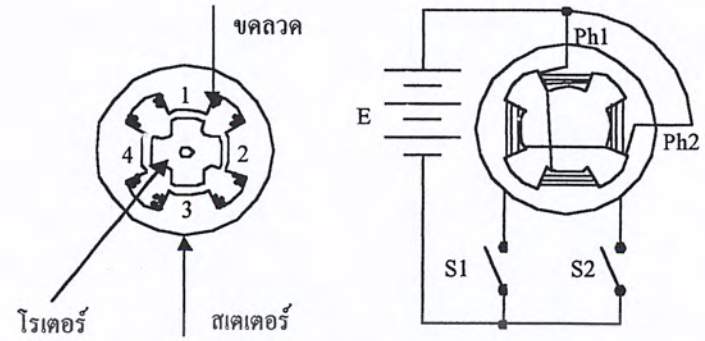
รูปที่ 2.9 แสดงการพันขดลวดของสเตเตอร์ด้านซ้ายเป็นแบบไบโพลาร์ และที่เหลือเป็นแบบยูนิโพลาร์

สเต็ปปิ้งมอเตอร์แบบไบโพลาร์มีการพันขดลวด 1 ขดแต่ละขั้วแม่เหล็กของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้า และสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศทางการไหลของกระแสไฟฟ้า ซึ่งในการกำหนดทิศทางการไหล และกลับทิศทางของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิตช์ชิ่งกลับขั้วไฟฟ้า สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปเปลี่ยนมาทำได้โดยการสวิตช์กระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดลวดหนึ่งแทนเท่านั้น ซึ่งปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วม เพื่อลดจำนวนของสายไฟที่ต่อกับมอเตอร์ วงจรกำลังไฟฟ้าของมอเตอร์แบบยูนิโพลาร์ทำได้ง่ายกว่าไบโพลาร์ เพราะมอเตอร์แบบยูนิโพลาร์ต้องการเพียงสวิตช์ธรรมดา ในการเปิดปิดกำลังไฟฟ้าขดลวดบนสเตเตอร์ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 2.10 แสดงวงจรการจ่ายไฟฟ้าซึ่งทรานซิสเตอร์ทำหน้าที่เป็นสวิตชิ่งให้กับสเต็ปปิ้งมอเตอร์ที่มีการพันขดลวดทั้งสองแบบ จะเห็นว่าแบบยูนิโพลาร์เป็นวงจรที่ง่ายและไม่ซับซ้อน



รูปที่ 2.10 แสดงการจ่ายไฟฟ้าให้กับสเต็ปปิ้งมอเตอร์
ก) แบบยูนิโพลาร์ซึ่งใช้ทรานซิสเตอร์เพียงตัวเดียวต่อ 1 คอยล์
ข) แบบไบโพลาร์ ซึ่งต้องใช้ทรานซิสเตอร์ 4 ตัวต่อ 1 คอยล์

การทำงานของสเต็ปปิ้งมอเตอร์แบบวีอาร์ (VR) จะเป็นพื้นฐานสำคัญในการทำงานของสเต็ปปิ้งมอเตอร์ ซึ่งจะช่วยให้เข้าใจการทำงานของสเต็ปปิ้งมอเตอร์ชนิดอื่น ๆ ได้ง่ายยิ่งขึ้น ดังในรูปจะเป็นภาพแสดงถึงการพันขดลวดแบบวีอาร์ (VR) มอเตอร์แบบ 2 เฟส มีขั้วเหนือและขั้วใต้อยู่ตรงข้ามกัน 2 คู่ โดยจะพันขดลวดแบบอนุกรมกันในแต่ละชุด ถ้ามีการกระตุ้นเฟสเกิดขึ้น 1,2 จะเป็นขั้วใต้ และ 1,2 จะเป็นขั้วเหนือ ทั้งโรเตอร์และสเตเตอร์จะทำจากเหล็กซิลิกอน ซึ่งเป็นวัสดุที่มีความซึมซับสูง สามารถให้เส้นแรงแม่เหล็กไหลผ่านได้มาก



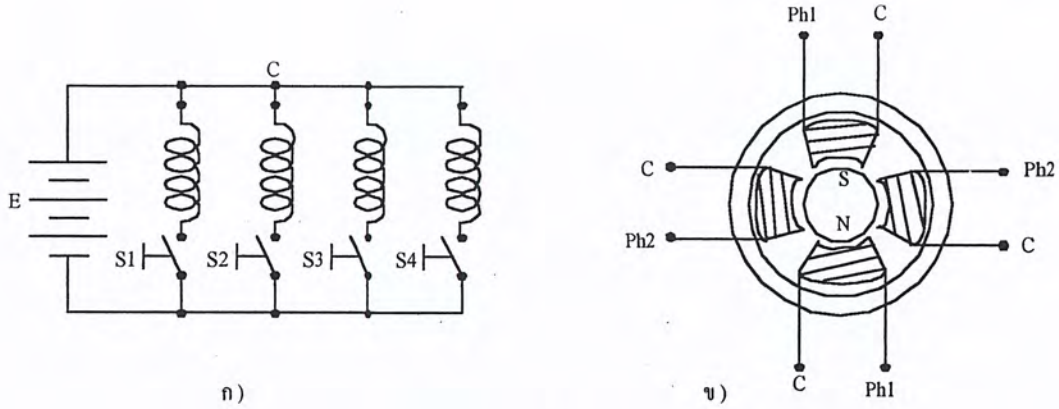
รูปที่ 2.11 ภาพหน้าตัดและการพันขดลวดของวาริเอเบิลรีลักแตนซ์สเต็ปมอเตอร์ 2 เฟส

การทำงานจะมีการกระตุ้นที่เฟส 1 ก่อน (S "ON") ซึ่งจะทำให้เส้นแรงแม่เหล็กเกิดขึ้นดังรูป ตัวโรเตอร์จะพยายามวางตำแหน่งของตัวเองให้อยู่ในทิศทางที่ทำให้เกิดค่าความต้านทานแม่เหล็กน้อยที่สุด ในขณะที่เริ่มต้นที่กระตุ้นที่เฟส 2 (S1 "OFF" , S2 "ON") เส้นแรงแม่เหล็กจะเดินทางไม่สะดวก จึงทำให้ค่าความต้านทานแม่เหล็กมีค่าสูง ตัวโรเตอร์ก็พยายามปรับตัวเองให้มีค่าความต้านทานแม่เหล็กน้อยที่สุด ด้วยการหมุนในทิศทางทวนเข็มนาฬิกาซึ่งแรงบิดที่ใช้หมุนเกิดจากแรงของเส้นแรงแม่เหล็ก ก็จะหมุนไปหยุดในตำแหน่งที่ความต้านทานแม่เหล็กน้อยที่สุด นั่นคือจะหมุนไป 1 สเต็ป หรือ 30 องศานั่นเอง ความสัมพันธ์ระหว่างจำนวนสเต็ปของการหมุนของโรเตอร์ไป 1 รอบ (S) มุมที่เปลี่ยนไป 1 สเต็ป จำนวนเฟสของสเตเตอร์ (m) และจำนวนฟันของโรเตอร์ (Nr) ดังแสดงไว้ในสมการที่ 1

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

ตัวอย่างเช่น สเต็ปปิ้งมอเตอร์ตัวหนึ่งมี m = 2 , Nr = 4 ก็จะได้ S = 2 * 4 = 8 สเต็ป และมุมในการหมุน θ_s = 360 / 8 = 4.5 องศา ซึ่งจากสมการที่ 1 ทำให้เราทราบอีกว่าถ้าจะลดค่าของ θ_s ให้น้อยลง อาจทำได้โดยการเพิ่มค่า m หรือค่า Nr ให้สูงขึ้น และลดช่องว่างระหว่างโรเตอร์กับสเตเตอร์ให้มีค่าน้อย ๆ เพื่อให้เกิดแรงบิดสูงสุด และยังมีผลต่อความเที่ยงตรงของตำแหน่งมากยิ่งขึ้นด้วย

สำหรับสเต็ปปิ้งมอเตอร์ชนิดเพอร์มาเนนต์แนกเน็ต หรือ PM จะมีข้อแตกต่างสำคัญจาก VR สเต็ปปิ้งมอเตอร์ คือ โรเตอร์จะเป็นแบบแม่เหล็กถาวรการพันขดลวดจึงจะต้องมีความแตกต่างกันออกไปซึ่งแสดงดังรูป



รูปที่ 2.12 ก) วงจรกระตุ้นเฟสพื้นฐานสำหรับ PM มอเตอร์ 4 เฟส
ข) ภาพหน้าตัดของ PM มอเตอร์แบบ 4 เฟส

จากรูป ก) จะเห็นว่าสเตเตอร์ในแต่ละขั้วจะมีขดลวดพันอยู่ ซึ่งถือว่าแต่ละขั้วคือหนึ่ง ดังนั้นจากรูปจึงมีทั้งหมด 4 เฟส ด้วยกัน สำหรับการต่อวงจรกระตุ้นเฟสอย่างง่ายแสดงไว้ในรูป ข) จะเห็นว่าปลายขดลวด (c) จะต่อร่วมกันถึงขั้วบวกของแหล่งจ่ายไฟดังนั้นเมื่อเกิดการกระตุ้นที่เฟสใดแล้วขั้วสเตเตอร์ที่เฟสนั้นจะกลายเป็นขั้วเหนือ และจากรูปจะเห็นว่าเป็นการแสดงตำแหน่งของโรเตอร์แต่ละสเต็ป หลังจากถูกกระตุ้นที่เฟส 1 , 2 , 3 และ 4 ตามลำดับ และจะหมุนไปในทิศทางตามเข็มนาฬิกา ทุก 90 องศา ต่อสเต็ป ถ้าต้องการให้หมุนองศาต่อสเต็ปมีค่าลดลงหรือมีความละเอียดในตำแหน่งมากขึ้น จะต้องเพิ่มจำนวนเฟสของสเตเตอร์และจำนวนขั้วแม่เหล็กของโรเตอร์ให้มากขึ้น ข้อเสียของ PM มอเตอร์ คือ ราคาแพง และความหนาแน่นของเส้นแรงแม่เหล็กจะถูกจำกัดที่เส้นแรงแม่เหล็กภายใน ทำให้ไม่สามารถผลิตแรงบิดได้มาก

2.5.2 โหมดการทำงานของสเต็ปปิ้งมอเตอร์

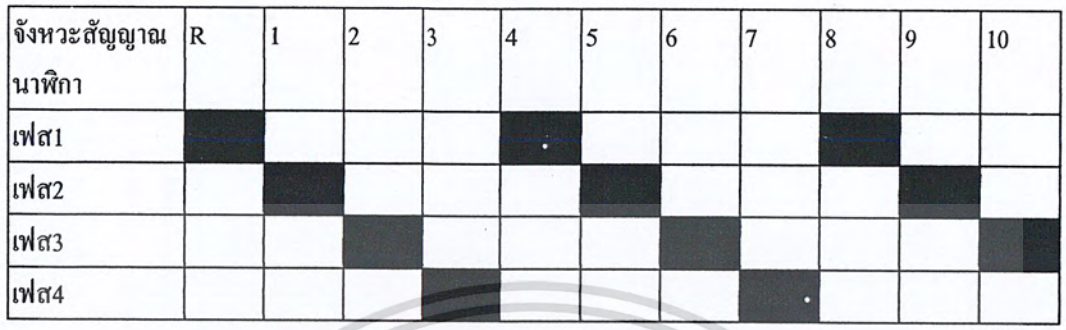
ถ้าจะแบ่งการทำงานของสเต็ปปิ้งมอเตอร์ตามอัตราเร็วของสเต็ปแต่ละสเต็ปจะแบ่งออกได้เป็น 2 โหมด คือหมุนเป็นสเต็ป และหมุนแบบต่อเนื่อง โดยถ้าหมุนแบบเป็นสเต็ปและมีเวลาหยุดนิ่งก่อนที่จะเปลี่ยนเป็นสเต็ปถัดไป ก็จะเรียกการทำงานในโหมดนี้ว่าการหมุนเป็นสเต็ป ดังแสดงในรูปที่ (ก) ซึ่งเป็นตัวอย่างการนำไปใช้เป็นเครื่องตอกบัตร

ถ้าเพิ่มความเร็วของการหมุนของสเต็ปปิ้งมอเตอร์ให้เร็วขึ้นและเป็นไปอย่างติอเนื่องไม่มีการหยุดนิ่งเราเรียกการทำงานนี้ว่า การหมุนแบบต่อเนื่องซึ่งหาความสัมพันธ์ของความเร็วรอบของมอเตอร์ (n) กับอัตราการเร็วของสเต็ป (f) และจำนวนสเต็ปทั้งหมด (s)

$$n = 60\ f/s$$

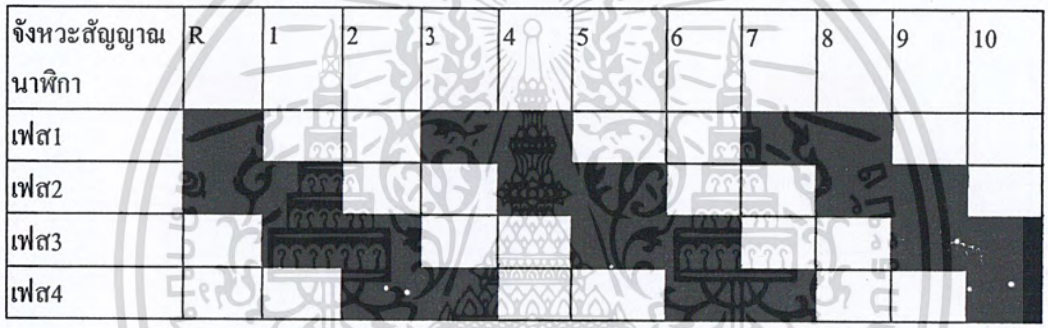
2.5.3 วิธีการกระตุ้นเฟส

การที่จะทำให้สเต็ปปิ้งมอเตอร์หมุนได้อย่างต่อเนื่องเหมือนการหมุนของมอเตอร์ไฟฟ้ากระแสตรงนั้นต้องมีการจ่ายกระแสพัลซ์เป็นลำดับอย่างต่อเนื่อง วิธีการที่จะกระตุ้นแบบเฟสมี



จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส1											
เฟส2											
เฟส3											
เฟส4											

ก)



จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส1											
เฟส2											
เฟส3											
เฟส4											

ข)

จังหวะสัญญาณนาฬิกา	R	1	2	3	4	5	6	7	8	9	10
เฟส1											
เฟส2											
เฟส3											
เฟส4											

ค)

รูปที่ 2.13 ตารางการกระตุ้นเฟส

ก) แบบเฟสเดียว Single Phase

ข) แบบเฟสคู่ Two Phase

ค) แบบกึ่งสเต็ป Half Phase

ด้วยกันหลายวิธีแต่จะอธิบายวิธีที่ใช้กันมากเริ่มจากแบบแรก คือ การกระตุ้นแบบเฟสเดียว (Single Phase Excitation) เป็นการกระตุ้นเฟสเพียงเฟสเดียวเท่านั้นที่สัญญาณนาฬิกาหนึ่ง ๆ แบบที่ 2 เป็นการกระตุ้น แบบเฟสคู่ (Two Phase Excitation) ก็จะมีการกระตุ้นเฟส 2 เฟส พร้อมกันในจังหวะสัญญาณนาฬิกาหนึ่ง ๆ สำหรับแบบสุดท้ายเป็นการกระตุ้นแบบกึ่งสเต็ป (Half Step Excitation) จะเป็นการรวมเอาทั้งสองแบบเข้าด้วยกันโดยจะทำการกระตุ้นเฟสทั้งสองแบบสลับกันไป ซึ่งลักษณะของการกระตุ้นเฟสทั้งสามแบบสามารถดูได้จากรูปที่ 2.13

ในการกระตุ้นเฟสแบบเฟสคู่จะมีทิศทางของเส้นและแรงแม่เหล็กไม่เป็นเส้นตรงเหมือนกับการกระตุ้นแบบเฟสเดียว ดังแสดงในรูป ก) แต่ถึงกระนั้นค่าของมุมที่เปลี่ยนไปในหนึ่งสเต็ปก็ยังคงมีค่าเท่าเดิมเหมือนกับการกระตุ้นแบบเฟสเดียว และสำหรับการกระตุ้นแบบครึ่งสเต็ปค่ามุมที่เปลี่ยนไปในหนึ่งสเต็ปจะมีค่าลดลงครึ่งหนึ่ง การกระตุ้นแบบเฟสคู่จะมีแรงบิดที่มากกว่าการกระตุ้นแบบเฟสเดียว ขณะเดียวกันยังสามารถเข้าถึงตำแหน่งได้รวดเร็วกว่าดังแสดงไว้ในรูป ข)

ข้อดีอีกอย่างหนึ่งของการกระตุ้นเฟสแบบกึ่งสเต็ปก็คือสามารถลดผลกระทบจากความถี่ในย่านรีโซแนนซ์ได้ แต่ที่ความถี่ต่ำ ๆ จะมีแรงบิดลดลง ดังแสดงไว้ในรูป ค) และสำหรับทิศทางการหมุนของมอเตอร์ขึ้นอยู่กับทิศทางของลำดับเฟสที่ถูกกระตุ้นด้วย

บทที่ 3

การออกแบบและกระบวนการสร้าง

ในส่วนของการออกแบบและกระบวนการสร้าง ชิ้นงานในการปฏิบัติงานครั้งนี้ จะแบ่งส่วนหลัก ๆ ได้เพียงสองส่วนใหญ่ ๆ คือ ทางด้าน

-Software (MCS51)

-Hardware

3.1 Software

ในการใช้โปรแกรม (Software) ไปควบคุมการตอบสนองต่ออินพุตเอาต์พุตของ 8051 ได้โดยออกแบบโปรแกรมโดยใช้ภาษาแอสเซมบลี (Assembly) โดยทำการเปลี่ยนรูปแบบโปรแกรมไปเป็นตัวเลขฐานสิบหก (File .Hex) แล้วผ่านเข้าเครื่องโปรแกรม ทำการโปรแกรมข้อมูลลงในชิป 8051 โดยโปรแกรมนี้จะเป็นตัวควบคุมการทำงานของพอร์ตและขบวนการทำงานต่าง ๆ ของ 8051 เพื่อให้พอร์ตนี้ไปควบคุมการทำงานของอุปกรณ์อิเล็คทรอนิคส์ภายนอก

ชุดคำสั่งและความหมาย

รหัสนีโมนิก	ลักษณะการทำงาน	ไบต์	ช่วงเวลาออสซิลเลเตอร์
กลุ่มคำสั่งทางคณิตศาสตร์			
ADD A,Rn	Add register to Accumulator	1	12
ADD A,direct	Add direct byte to Accumulator	2	12
ADD A,@Ri	Add indirect RAM to Accumulator	1	12
ADD A,# data	Add immediate data to Accumulator	2	12
ADDC A,Rn	Add register to Accumulator with Carry	1	12
ADDC A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry	1	12

รหัสนีโมนิก	ลักษณะการทำงาน	ไบต์	ช่วงเวลาออสซิลเลเตอร์
ADDC a,# data	Add immediate data to Acc with Carry	2	12
SUBB A,Rn	Subtract register from Acc with borrow	1	12
SUBB A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB A,@ Ri	Subtract indirect RAM from Acc with borrow	1	12
SUBB A,# data	Subtract immediate data from Acc with borrow	2	12
INC A	Increment Accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment direct RAM	1	12
กลุ่มคำสั่งทางคณิตศาสตร์			
DEC A	Decrement Accumulator	1	12
DEC Rn	Decrement register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12
INC DPTR	Increment data pointer	1	24
MUL AB	Multiply A B	1	48
DIV AB	Divide A by B	1	48
DA A	Decimal Adjust Accumulator	1	12
กลุ่มคำสั่งทางตรรกศาสตร์			
ANL A,Rn	AND register to Accumulator	1	12
ANL A,direct	AND direct byte to Accumulator	2	12
ANL A,@Ri	AND indirect RAM to Accumulator	1	12

รหัสนีโมนิก		ลักษณะการทำงาน	ไบต์	ช่วงเวลาออสซิลเลเตอร์
ANL	A,# data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
AND	direct, data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR direct RAM to Accumulator	1	12
ORL	A,# data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct, #data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,# data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct, data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator left	1	12
RLC	A	Rotate Accumulator left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through Carry	1	12

รหัสนีโมนิก	ลักษณะการทำงาน	ไบต์	ช่วงเวลาออสซิลเลเตอร์
SWAP A	Swap nibbles with in the Accumulator	1	12
กลุ่มคำสั่งการเคลื่อนย้ายข้อมูล			
MOV A,Rn	Move register to Accumulator	1	12
MOV A,direct	Move direct byte to Accumulator	2	12
MOV A,@Ri	Move indirect Ram to Accumulator	1	12
MOV A,# data	Move immediate data to Accumulator	2	12
MOV Rn,A	Move register to Accumulator	1	12
MOV Rn,direct	Move direct byte to Accumulator	2	12
MOV Rn,# data	Move immediate data to register	2	12
MOV direct, A	Move Accumulator to direct byte	2	12
MOV direct,Rn	Move register to direct byte	2	12
MOV direct,direct	Move direct byte to direct	3	24
MOV direct,@Ri	Move indirect RAM to direct byte	2	24
MOV direct,# data	Move immediate data to direct byte	3	24
MOV @Ri,A	Move Accumulator to indirect RAM	1	12
MOV @Ri,direct	Move direct byte to indirect RAM	2	24
MOV @Ri,# data	Move immediate data to indirect RAM	2	12
MOV DPTR,# data16	Load data Pointer with A 16-bit constant	3	24
MOVC A,@A+DPTR	Move code byte relative to DPTR to ACC	1	24
MOVC A,@A+PC	Move code byte relative to PC to ACC	1	24
MOVX A,@Ri	Move External RAM (8-bit addr) to Acc	1	24

ตัวอย่าง คำสั่งที่ใช้เขียนโปรแกรมหน่วงเวลา

```
        ORG     0000H
DELAY:  MOV     R0,#0FFH
D1:     MOV     R1,#00H
D2:     DJNZ    R1,D2
        DJNZ    R0,D1
        END
```

วิธีการคำนวณ ระยะเวลาในการหน่วงเวลาทำได้ดังนี้

เช่น ถ้าเราใช้ตัวกำเนิดสัญญาณนาฬิกาด้วยความถี่ 12 MHz และ 8051 ทำงาน 1 ช่วง (State) ใช้เวลา 12 ลูกคลื่นนาฬิกา

คำสั่ง	Time State	จำนวนครั้งในการใช้	State ทั้งหมด
MOV R0,#0FFH	1	1	1
MOV R1,#00H	1	255	255
DJNZ R1,D2	2	65,280	130,560
DJNZ R0,D1	2	255	510
รวม	6	65,791	131,326

เพราะฉะนั้นเราจะได้เวลาการหน่วงดังนี้

$$T = 1/f = 1/12 \text{ MHz}$$

$$= 83.33 \text{ n sec.}$$

เนื่องจาก 1 ช่วงเวลา (Time State) เราใช้สัญญาณลูกคลื่นนาฬิกา 12 ลูก ดังนั้นเราได้ $T = 1\mu$ sec.

$1\mu \text{ sec} \times 131{,}326 = 0.131326 \text{ sec}$

เป็นการหน่วงเวลา0.131326 sec

โปรแกรมที่ใช้ในการควบคุมนิ้วมือกลชุดนี้มีดังต่อไปนี้

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology

Compile By ASM.EXE

Date : 19 ,October,2000:

MASTER SLAVE ROBOT FINGERS SYSTEM.asm

```
0090              1              p1 equ 090h
00B0              2              p3 equ 0b0h
00B5              3              s1 equu p3.5
00B4              4              s2 equ p3.4
0000              5              org 0000h
0000 7411         6              mov a,#11h
0002 7F50         7              mov r7,#50h
0004 03           8        az:   rr a
0005 F590         9              mov p1,a
0007 114D         10             acall de
0009 DFF9         11             djnz r7,az
000B D2B5         12       pb:   setb s1
000D 00           13             nop
000E 20B5FA       14             jb s1,pb
0011 7E00         15             mov r6,#00h
0013 23           16       cc:   rl a
0014 F590         17             mov p1,a
0016 114D         18             acall de
0018 DEF9         19             djnz r6,cc
001A D2B4         20       dt:   setb s2
001C 00           21             nop
001D 30B411       22             jnb s2,ff
0020 D2B5         23             setb s1
0022 00           24             nop
0023 30B5F4       25             jnb s1,dt
0026 7E00         26             mov r6,#00h
0028 03           27       eo:   rr a
```

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology

Compile By ASM.EXE

Date : 19 ,October,2000:

MASTER SLAVE ROBOT FINGERS SYSTEM.asm

```
0029 F590          28              mov p1,a
002B 114D          29              acall de
002D DEF9          30              djnz r6,eo
002F 010B          31              ajmp pb
0031 7C00          32      ff:     mov r4,#00h
0033 23            33      g:      rl a
0034 F590          34              mov p1,a
0036 114D          35              acall de
0038 DCF9          36              djnz r4,g
003A D2B4          37      h:      setb s2
003C 00            38              nop
003D 114D          39              acall de
003F 30B4F8        40              jnb s2,h
0042 7C00          41              mov r4,#00h
0044 03            42      i:      rr a
0045 F590          43              mov p1,a
0047 114D          44              acall de
0049 DCF9          45              djnz r4,i
004B 011A          46              ajmp dt
                   47
004D 78A0          48      de:     mov r0,#0a0h
004F 790A          49      d1:     mov r1,#00ah
0051 7A0A          50      d2:     mov r2,#00ah
0053 DAFE          51      d3:     djnz r2,d3
0055 D9FA          52              djnz r1,d2
0057 D8F6          53              djnz r0,d1
0059 22            54              ret
                   55              end
```

8051 Cross-Assembler (1.3) (C) 1987, 1989 Binary Technology

Compile By ASM.EXE

Date : 19 ,October,2000:

MASTER SLAVE ROBOT FINGERS SYSTEM.asm

AZ ..... C ADDR 0004H

CC ..... C ADDR 0013H

D1 ..... C ADDR 004FH

D2 ..... C ADDR 0051H

D3 ..... C ADDR 0053H

DE ..... C ADDR 004DH

DT ..... C ADDR 001AH

EO ..... C ADDR 0028H

FF ..... C ADDR 0031H

G. ..... C ADDR 0033H

H. ..... C ADDR 003AH

I. ..... C ADDR 0044H

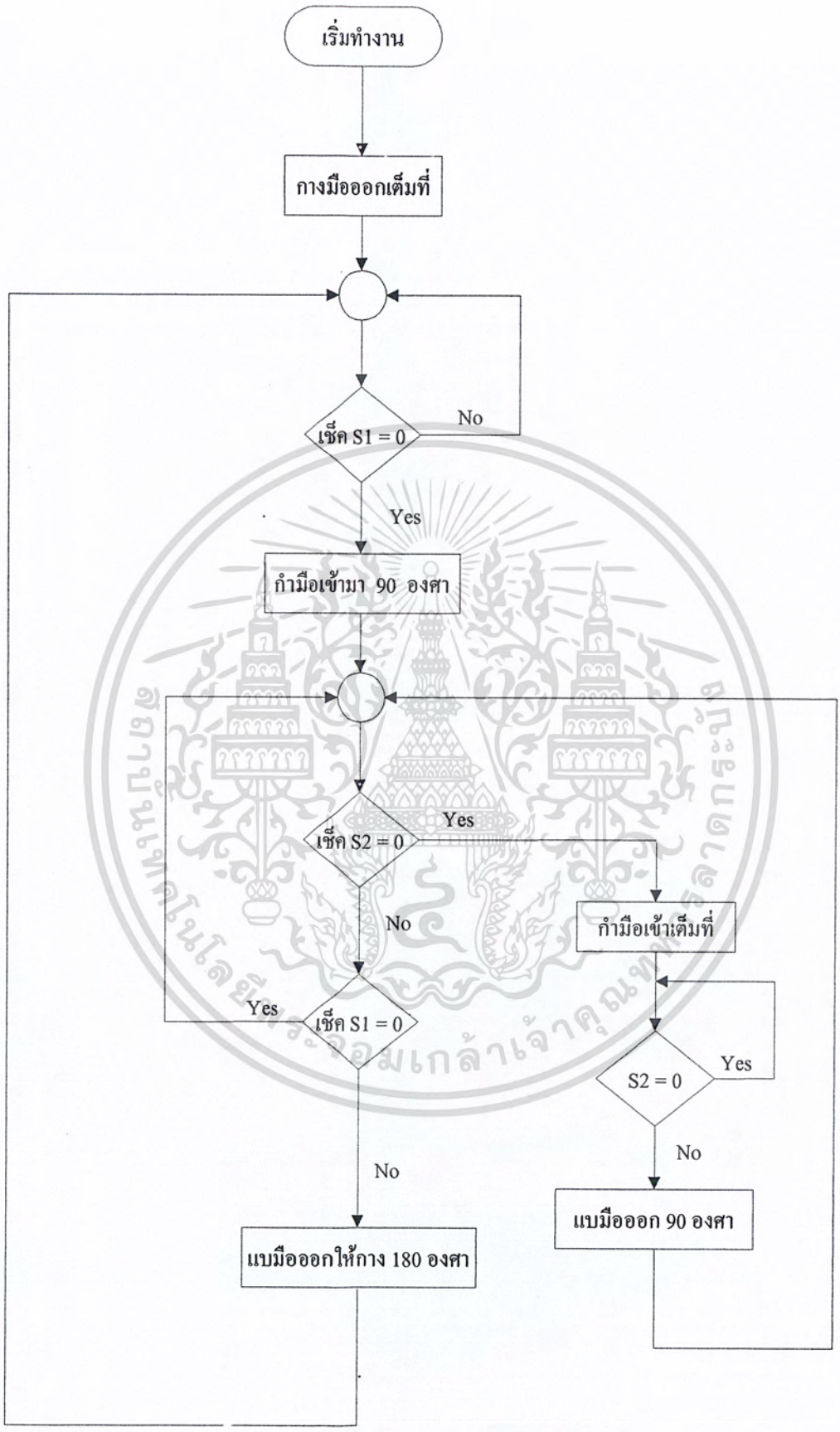
P1 ..... NUMB 0090H

P3 ..... NUMB 00B0H

PB ..... C ADDR 000BH

S1 ..... NUMB 00B5H

S2 ..... NUMB 00B4H



รูปที่ 3.1 Flow chart การทำงาน

ตารางที่ 3.1 แสดงสถานะการทำงาน

S1	S2	สถานะการทำงาน
1	1	แบมือเต็มที่
1	0	กำมือเข้ามา 90 องศา
0	0	กำมือเข้ามา 180 องศา

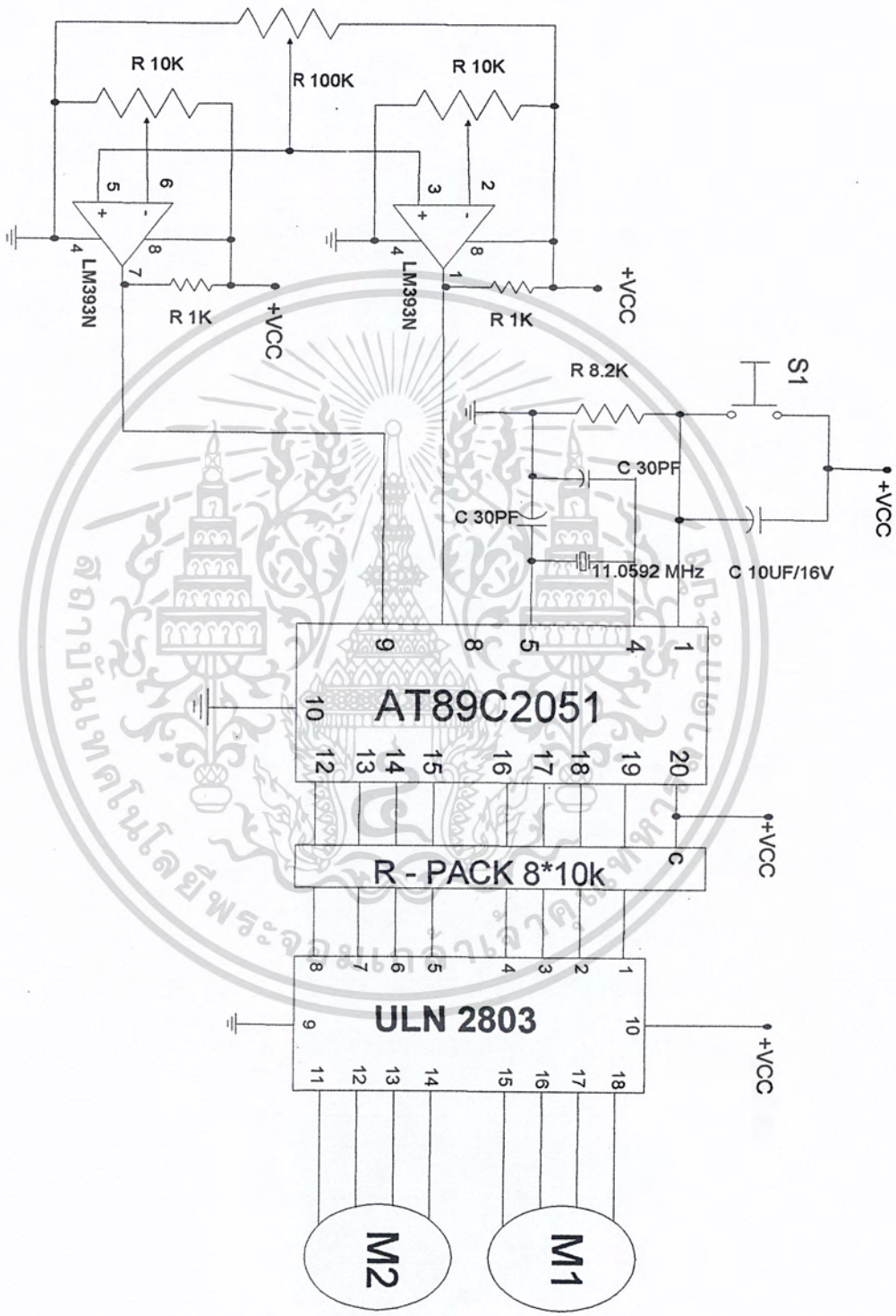
3.2 การออกแบบโปรแกรมและ ลำดับขั้นการทำงาน (Flow chart)

การออกแบบโปรแกรมจะต้องรู้ถึงสิ่งที่ต้องการหรือจุดประสงค์ก่อน จึงจะทำการออกแบบควบคุมการทำงานให้ได้ตามที่ต้องการ ดังนั้นจึงควรออกแบบ ลำดับขั้นการทำงาน (Flow chart) ก่อน จากนั้นจึงทำการออกแบบโปรแกรม

การออกแบบโปรแกรมจะทำการออกแบบโดยการคำนวนการดีเลย์เพื่อหาว่าต้องการเวลาเท่าใดที่จะกระตุ้นไปแล้วทำให้มอเตอร์มีความเร็วสูงสุดที่ทำให้ได้กำลังขับของมอเตอร์สูงด้วย และต้องทำการคำนวนพัลซ์ในการขับมอเตอร์เพื่อให้นิ้วมือเคลื่อนที่ได้ตามต้องการ จากการทดลองจะได้ค่าดังนี้

มอเตอร์หมุน 1 รอบ ต้องกระตุ้นพัลซ์ 48พัลซ์ (48 Pulse) ถ้าต้องการให้น๊อตหมุนไปได้ระยะ 1 เซนติเมตร หรือหมุนรอบน๊อต 7 รอบ ซึ่งเท่ากับกระตุ้นจำนวนพัลซ์เท่ากับ 48*7 พัลซ์ โดยใช้พัลซ์รวมกันทั้งหมด 336 พัลซ์ ต่อการเลื่อนน๊อต 1 ซ.ม. ซึ่งจะทำให้การเคลื่อนที่ของนิ้วมือช้า ถ้าเพิ่มความถี่สูงขึ้นจะทำให้มอเตอร์หมุนเร็วขึ้น แต่จะทำให้กำลังขับของมอเตอร์ลดลงด้วยจึงไม่เหมาะที่จะเพิ่มความถี่ของมอเตอร์ให้สูงขึ้น แต่จะมีวิธีแก้อีกวิธีหนึ่งคือ ทำการแก้ได้โดยใช้น๊อตที่มีเกลียวหยาบกว่านี้ก็จะสามารถทำให้นิ้วขยับได้เร็วมากขึ้น

3.3 Hardware



รูปที่ 3.2 วงจรที่ใช้ควบคุมการทำงาน

3.3.1 รายการอุปกรณ์ในการสร้างวงจร

- จากกระบวนการสร้างในโครงงานนี้จะสรุปเป็นค่าอุปกรณ์ต่างๆได้ดังนี้
- ในอุปกรณ์ 1 ชุด จะเท่ากับอุปกรณ์ที่ใช้ควบคุมนิ้วมือ 1 นิ้ว

รายการอุปกรณ์	ค่าอุปกรณ์	จำนวน
VR	10 kΩ	2 ตัว
VR	100 kΩ	1 ตัว
R	8.2kΩ	1ตัว
R- PACK	8*10 kΩ	1 ตัว
C	10 μF/16V	1 ตัว
C	30 pF	2 ตัว
Switch		1 ตัว
Cystral	11.059 MHz	1 ตัว
IC	LM 393N	1ตัว
IC	AT 89C2051	1 ตัว
IC	ULN 2803	1 ตัว
Motor	stepping	2 ตัว

3.4 ลำดับตอนการทำโครงสร้างและส่วนประกอบของนิ้วมือ

1. นำลวดทองแดงมาดัดขึ้นรูปเป็นโครงสร้างของนิ้วมือ โดยที่แต่ละนิ้วมือจะมีจุดเชื่อมต่อโดยใช้การบัดกรีเพื่อทำการเชื่อมต่อแต่ละจุด โดยการดัดแต่ละนิ้วมือนั้นจะต้องมีขนาดที่เท่ากับตัวสเต็ปปิ้งมอเตอร์ที่สามารถจะประกอบติดไปได้กับโครงสร้างของนิ้วมือ

2. ส่วนของมอเตอร์ที่ประกอบติดอยู่กับโครงของนิ้วมือ แกนโรเตอร์ของมอเตอร์นั้นจะยึดติดอยู่กับตัวน๊อตเพื่อนำไปหมุนน๊อตทำให้นิ้วมือขยับได้ โดยการยึดมอเตอร์ติดกับลวดทองแดงจะใช้ซิลิโคลเป็นตัวประสาน

3. การทำฐานเพื่อรองรับนิ้วมือนั้นจะใช้ปูนปลาสเตอร์ในการหล่อฐาน โดยทำการฝังสายไฟของมอเตอร์สเต็ป (Stepping motor) ไว้ที่ฐานรองและลวดทองแดงแต่ละนิ้วมือเพื่อขึ้นรูปนิ้วมือ ให้นิ้วมือติดกับฐานอย่างมั่นคง

4. ทำการทาสีที่ฐานของนิ้วมือเพื่อเคลือบปูนปลาสเตอร์ให้เรียบขึ้น และเคลือบด้วยแลกเกอร์อีกที

3.5 ลำดับขั้นตอนการทำถุงมือเพื่อสวมให้กับมือ

1. นำโครงสร้างของมือที่ทำสำเร็จแล้วมาทำการทาทับด้วยยางพารา แล้วนำไปตากแดดให้แห้งทำลักษณะนี้หลายรอบจนกว่าจะได้ความหนาของยางพาราที่พอเหมาะ แล้วถุงมือยางออกจากโครงของมือ แต่ถุงมือยางที่ได้จะยังไม่สามารถใช้งานจริงได้

2. นำถุงมือยางที่ได้จากขั้นแรกมาเป็นแบบอีกทีหนึ่ง โดยจะนำปูนปลาสเตอร์เทใส่ในแบบเพื่อหล่อทำรูปมือ ก็จะได้มือที่ได้จากการหล่อของปูนปลาสเตอร์

3. นำแบบมือที่ได้จากการหล่อจากปูนปลาสเตอร์มาทาด้วยยางพาราอีกรอบ ตามแบบขั้นตอนแรก ก็จะได้ถุงมือยางที่ใช้ในการครอบโครงสร้างมือ

3.6 การออกแบบแผ่นวงจร

1. เมื่อทราบถึงวงจรเป็นลักษณะใดก็ทำการเขียนลายวงจร โดยเขียนใส่กระดาษ
2. ทำการเขียนเรียบร้อยแล้วนำไปถ่ายเอกสารใส่แผ่นใส และตัดตามขนาดที่ต้องการ
3. ตัดแผ่นทองแดงตามขนาดที่ต้องการ
4. ตัดแผ่นใสตามขนาดที่ต้องการ และทำความสะอาดผิวทองแดง
5. ลอกแผ่นพลาสติก (Cover Sheet) ที่ติดอยู่กับแผ่นฟิล์ม (Dry Film) แล้นำไปติดลงบนแผ่นทองแดง โดยไม่ให้มีฟองอากาศ
6. นำผ้ามาวางทับบนชิ้นงาน จากนั้นนำเตารีดไฟฟ้าที่มีความร้อนประมาณ 110 องศาเซลเซียส รีดทับบนผ้าให้ทั่ว ระวังอย่าให้ความร้อนมากเกินไป
7. นำหลอดไฟขนาด 100 วัตต์ มาฉายบนชิ้นงานโดยให้ห่างจากชิ้นงานประมาณ 1 ฟุต
8. นำแผ่นใสที่เตรียมไว้มาวางทับบนชิ้นงานแล้วนำกระจกใสมาวางทับอีกชั้น จากนั้นเปิดไฟให้กับหลอดไฟเพื่อทำการถ่ายขึ้นลายโดยใช้เวลาประมาณ 15 นาที
9. ลอกแผ่นพลาสติก (Cover Sheet) ที่ติดอยู่กับแผ่นฟิล์ม (Dry Film) อีกชั้นออก แล้วนำมาล้างในน้ำยา โซเดียมคาร์บอเนต ที่ผสมกับน้ำในอัตราส่วน 10 กรัมต่อน้ำ 1 ลิตรแล้วใช้ฟองน้ำลูบเบาจนแผ่นฟิล์มที่ไม่ต้องการละลายออกหมด แล้วนำไปล้างด้วยน้ำสะอาดและรอจนฟิล์มแห้ง
10. นำชิ้นงานไปละลายกับกรดกัดทองแดง และทำการเขย่าน้ำยาไปมาพร้อมทำการตรวจสอบจนทองแดงหลุดในส่วนที่ไม่ต้องการออก แล้วนำไปล้างน้ำสะอาด
11. นำชิ้นงานไปลงในภาชนะที่มีทินเนอร์ ทิ้งไว้สักครู่แผ่นฟิล์มก็จะหลุดออก จะได้ลายทองแดงที่ต้องการจากนั้นจึงนำไปทำความสะอาด
12. เจาะรู และทำความสะอาดแล้วเช็ดลายแผ่นปริ้นซ์ก็จะแผ่นวงจรตามที่ต้องการ

บทที่ 4
การทดลองและผลการทดลอง

โครงงานนี้เป็นการนำไมโครคอนโทรลเลอร์มาใช้ควบคุมการเคลื่อนที่ของ มอเตอร์สเต็ป (Stepping Motor) และ มอเตอร์ตัวนี้เป็นตัวที่นำไปควบคุมการเคลื่อนที่ของนิ้วมือแต่ละนิ้ว โดยนำเอามอเตอร์ แต่ละตัวควบคุมข้อนิ้วแต่ละข้อโดยแยกการควบคุมอย่างอิสระจากกัน โดยใช้ 8051 (MCS-51) เป็นตัวควบคุมการจ่ายพัลซ์ให้กับมอเตอร์สเต็ป (Stepping Motor) ซึ่งการทดลองนั้นจะต้องประกอบกันหลายส่วน เริ่มจากส่วนโครงสร้างของชิ้นงาน กำลังขับของมอเตอร์สเต็ป โดยการใช้พัลส์จากไมโครคอนโทรลเลอร์มาควบคุมการทำงานซึ่งการทดลองการทำงานที่ประกอบด้วย ส่วนต่างๆดังนี้

4.1 โครงสร้างของชิ้นงาน

ชิ้นงานที่สร้างขึ้นขึ้นมานั้นจะประกอบไปด้วยส่วนต่างๆดังนี้

4.1.1 โครงร่าง

เป็นการใช้เส้นลวดทองแดงนำมาดัดขึ้นรูปให้เป็นโครงร่างของมือ และนิ้วโดยใช้เส้นลวดทองแดงนี้ดัดเป็นโครงให้เป็นแต่ละข้อนิ้วมือ เพื่อให้แยกเป็นอิสระต่อการควบคุม หลังจากทำการดัดลวดทองแดงขึ้นรูปแล้ว ผลที่ได้คือ ข้อนิ้วมือแต่ละข้อสามารถขยับขึ้นลงได้อย่างอิสระ แต่เกิดปัญหาคือบางข้อของนิ้วมือเกิดความไม่แน่นอนของการดัดลวดเพราะในการดัดนั้นไม่ใช้อุปกรณ์ในการขึ้นรูปที่ทันสมัยพอ

4.1.2 ถุงมือยาง

เป็นการนำเอายางพารามาหล่อกับแบบที่เป็นนิ้วมือ แล้วเป่าด้วยลมร้อนให้ถุงมือแห้งและใช้แสงแดดช่วยในการทำให้ผิวยางพาราแห้ง

ผลที่ได้จากการหล่อยางพารา จะได้ถุงมือที่ยังไม่สมบูรณ์เท่าที่ควร เพราะการหล่อยังใช้อุปกรณ์ในการหล่อที่ยังไม่ทันสมัยพอ

4.1.3 ถุงมือควบคุม

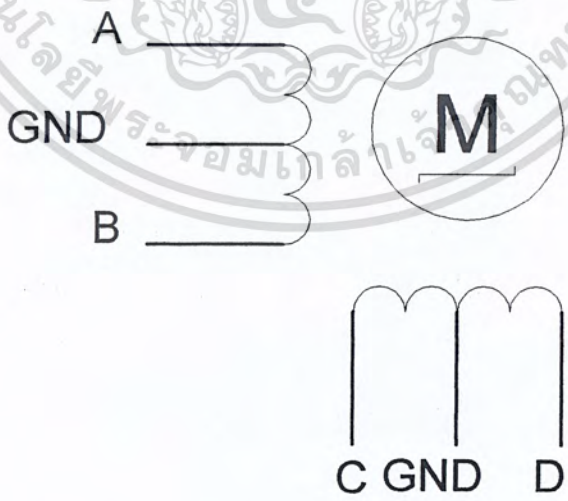
ในการควบคุมนิ้วมือกลจะถูกสั่งการผ่านถุงมือควบคุมนี้โดยตรง จะใช้วิธีการขยับนิ้วมือขึ้นลงเป็นการปรับความต้านทาน เพื่อให้ได้แรงดันไฟฟ้าที่เปลี่ยนแปลงตามการเคลื่อนไหวของถุงมือควบคุม เพื่อนำไปทำการควบคุมสัญญาณอินพุต

ผลที่ได้จากการทดลอง คือ เมื่อขยับถุงมือควบคุมจะได้การเปลี่ยนแปลงของแรงดันและสามารถนำการเปลี่ยนแปลงของแรงดันนี้ ไปทำการควบคุมการทำงานของ 8051 ได้

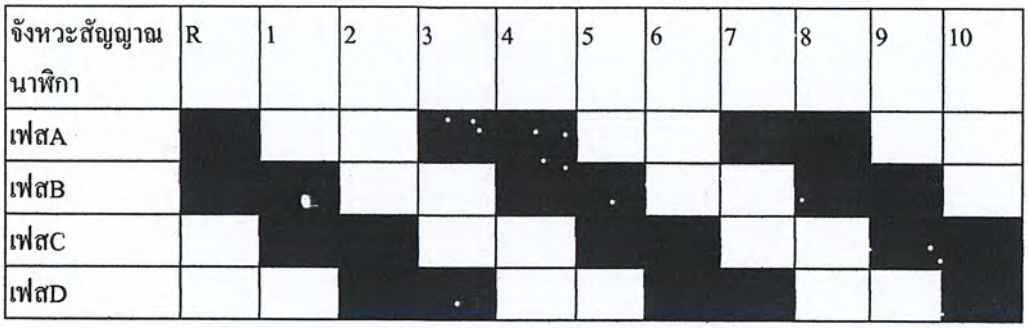
4.2 การทดสอบป้อนพัลซ์ให้ มอเตอร์สเต็ป (Stepping Motor)

มอเตอร์สเต็ป (Stepping motor) ที่ใช้ในโครงงานนี้เป็นมอเตอร์แบบ 6 สาย 4 เฟส ซึ่งมีลักษณะดังรูปที่ 4.1 ซึ่งจะมีเฟสทั้งหมด 4 เฟส โดย 2 เฟส จะอยู่ในขดลวดเดียวกัน ดังนั้นในการต่อจึงต้องต่อขดลวดกราวนด์ (ground) เข้าด้วยกัน และทำการกระตุ้นโดยการเรียงตามลำดับเฟส A-B-C-D ซึ่งในโครงงานนี้จะทำการกระตุ้นแบบเฟสคู่ (Double phase) ซึ่งจะมีลักษณะดังรูปที่ 4.2

จากการกระตุ้นแบบนี้จะให้ค่าแรงดันพัลซ์ 1 จังหวะ สลับการทำงานไปคราวละ 2 เฟส (phase) และจะพบว่าในการใช้แรงดันพัลส์จากไมโครคอนโทรลเลอร์นั้น ค่าของกระแสไฟจะมีกำลังงานไม่เพียงพอในการจ่ายให้แก่มอเตอร์สเต็ป (stepping motor) จึงต้องมีวงจรขับสำหรับการจ่ายกำลังงานให้แก่มอเตอร์สเต็ป (stepping motor)



รูปที่ 4.1 แสดงลักษณะโครงสร้างของ มอเตอร์สเต็ป (Stepping motor)



รูปที่ 4.2 แสดงการกระตุ้นแบบสองเฟส (Double phase)

ผลที่ได้จากการกระตุ้นเฟสในกรณีที่ต่อจุดร่วมเข้ากับไฟบวก คือ เมื่อกระตุ้นเฟส A + B ต่อด้วย กระตุ้นเฟส B + C ต่อด้วย กระตุ้นเฟส C + D ต่อด้วย กระตุ้นเฟส D + A จะทำให้มอเตอร์หมุนตามเข็มนาฬิกา แต่ถ้ากระตุ้นจากเฟส A + D ต่อด้วย กระตุ้นเฟส D + C ต่อด้วย กระตุ้นเฟส C + B ต่อด้วย กระตุ้นเฟส B +A จะทำให้มอเตอร์หมุนทวนเข็มนาฬิกา

ผลที่ได้จากการกระตุ้นเฟสในกรณีที่ต่อจุดร่วมเข้ากับไฟกราวนด์ (GND) คือ เมื่อกระตุ้นเฟส A + B ต่อด้วย กระตุ้นเฟส B + C ต่อด้วย กระตุ้นเฟส C + D ต่อด้วย กระตุ้นเฟส D + A จะทำให้มอเตอร์หมุนทวนเข็มนาฬิกา แต่ถ้ากระตุ้นจากเฟส A + D ต่อด้วย กระตุ้นเฟส D + C ต่อด้วย กระตุ้นเฟส C + B ต่อด้วย กระตุ้นเฟส B +A จะทำให้มอเตอร์หมุนตามเข็มนาฬิกา

ในการทำงานจริงจะต้องใช้การหมุนทั้งตามเข็มและทวนเข็ม เพื่อนำไปใช้ควบคุมการในเคลื่อนที่ขยับขึ้นลงของนิ้วมือ

4.3 การออกแบบวงจรการทำงาน

4.3.1 วงจรด้านรับสัญญาณ (I/P)

การออกแบบด้านรับของสัญญาณมีแนวความคิด คือ ใช้การเปลี่ยนแรงดันไฟฟ้าตามการเคลื่อนไหวโดยใช่ความต้านทานเปลี่ยนค่า (VR) เป็นตัวเช็ค แล้วจึงนำแรงดดันไฟฟ้านี้ไปทำการเปลี่ยนเป็นสัญญาณดิจิตอลโดยผ่านวงจรเปรียบเทียบแรงดันโดยใช้ ออปแอมป์เป็นตัวเปรียบเทียบแรงดันกับค่าที่ตั้งไว้ ถ้าให้แรงดันที่เข้ามาสูงกว่าแรงดันที่ตั้งไว้จะได้สัญญาณดิจิตอล “1” แต่ถ้าแรงดันที่เข้ามาต่ำกว่าแรงดันที่ตั้งไว้จะได้สัญญาณดิจิตอล “0” ในการควบคุมนิ้วมือ 1 นิ้วจะใช้สัญญาณดิจิตอลจำนวน 2 บิต จึงจะทำให้ความละเอียดมากกว่าจึงต้องมีวงจรเปรียบเทียบแรงดัน 2 ชุด แต่ปรับตั้ง Vref ให้แตกต่างกันไปจะได้ผลการเปรียบเทียบแรงดันดังนี้

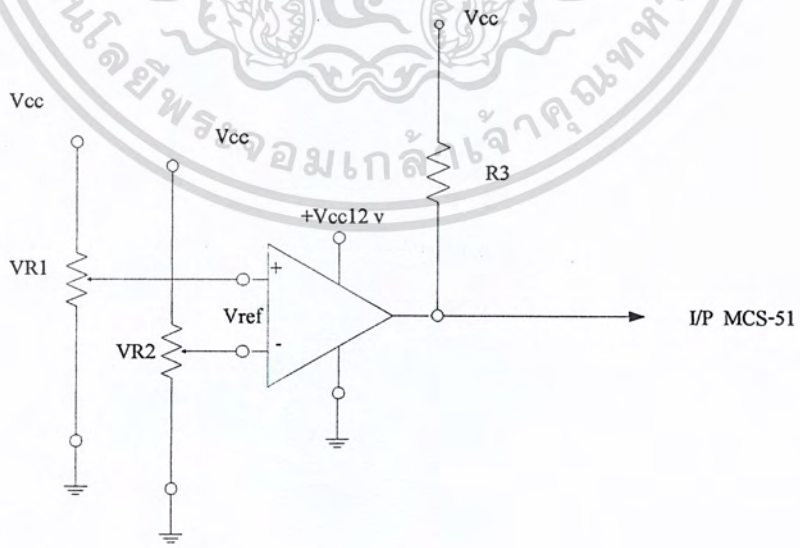
แต่ในการทดลองจริง เมื่อทดสอบเฉพาะวงจรเปรียบเทียบแรงดันจะให้ผลตามที่ต้องการจริง แต่วงจรนี้จะไม่สามารถส่งกระแสสูงได้ถึงประมาณ 100 mA ไม่เพียงพอที่จะไปกระตุ้นการ

เปลี่ยนแปลงให้ 8051 ด้วยสัญญาณดิจิตอล "1" ได้ ดังนั้นจึงต้องมีวงจรขยายกระแสต่อเข้าไปอีก เพื่อช่วยในการกระตุ้น 8051 ให้รู้ถึงการเปลี่ยนแปลงของแรงดัน

ตารางที่ 4.1 ตารางเปรียบเทียบแรงดัน

Vin	Vref	สภาพ O/P
0	0.5	0
0	1	0
1	0	Vcc
1.5	1	Vcc
2	1	Vcc
3	1	Vcc

เมื่อใช้วงจรขยายดังรูปที่ 4.3 จะทำให้สามารถขับกระแสได้สูงสุดถึง 800 mA แต่ในการกระตุ้นสัญญาณใช้งานในวงจรจะใช้กระแส 100 mA ก็พอ จึงสามารถใช้งานได้ตามต้องการ แต่ปัญหานี้สามารถแก้ได้อีกวิธีหนึ่งคือ ทำการแก้ที่โปรแกรมการทำงานของ 8051 ให้มีการตรวจเช็คสัญญาณที่เป็นบิตต่ำ "0" (Lower) ก็จะไม่จำเป็นต้องใช้วงจรขยายสัญญาณเพื่อส่งเข้าไปใน 8051



รูปที่ 4.3 วงจรภาคอินพุต

4.3.2 ส่วนที่ทำการควบคุมการทำงานต่าง ๆ ของ 8051 (MCS-51)

การทำนิ้วมือกลจะใช้ไอซี AT 89C2051 เป็นตัวประมวลผล และสั่งการของระบบเพราะ AT 89C2051 เป็นไอซีตัวเล็กใช้แทน AT 89C51 ได้ แต่มีจำนวนอินพุตและเอาต์พุตจำนวน 15 บิต ก็เพียงพอต่อการทำงาน และสั่งการให้แต่ละนิ้วมือทำงานตามต้องการได้ แต่ถ้าทั้ง 5 นิ้วใช้ไอซี AT 89C2051 เพียงตัวเดียวก็จะทำให้เคลื่อนที่ได้ช้าถ้ามีการเคลื่อนที่พร้อม ๆ กันดังนั้นในการสร้างจึงแยกใช้ไอซี AT 89C2051 หนึ่งตัวต่อหนึ่งนิ้วมือ

ไอซี AT 89C2051 จะต้องการชุดสัญญาณนาฬิกาที่จะเป็นตัวกำหนดความเร็วในการประมวลผล ซึ่งให้ใช้ได้สูงสุดประมาณ 24 MHz โดยจ่ายแรงดันบวก (Vcc) ในช่วง 4.0 โวลท์ ถึง 6.0 โวลท์ โดยถ้าใช้คริสตอลจะต้องใช้ C3 และ C4 = 30 pF หรือถ้าใช้เซรามิครีโซเรเตอร์จะต้องใช้ C3 และ C4 = 40 pF

วงจรรีเซตต้องการไฟบวก (Vcc) เข้าที่ขา 1 ของไอซี AT 89C2051 ในวงจรจะต่อสวิทช์เป็นตัวจ่ายไฟบวก (Vcc) เพื่อจ่ายให้ไอซี AT 89C2051 เพื่อทำการรีเซต C1 ของไอซี AT 89C2051 ลงกราวนด์เมื่อไม่มีการรีเซต

จากวงจรได้ทำการต่อวงจรลงในแผงทดลอง โดยใช้ไดโอดเรืองแสง (LCD) เป็นตัวแสดงผล และใช้สวิทช์เป็นตัวเป็นตัวควบคุมทางด้านอินพุต เมื่อทำการทดลองสามารถทำให้ไดโอดเรืองแสง (LCD) กระพริบสลับไปสลับมาได้ และเมื่อนำมอเตอร์สเต็ปมาต่อแทนไดโอดเรืองแสง (LCD) สามารถทำให้มอเตอร์หมุนตามเข็มนาฬิกา และทวนเข็มนาฬิกาได้ตามที่กำหนดไว้

4.3.3 ส่วนวงจรขยายสัญญาณและการต่อเข้ามอเตอร์

ในส่วนของวงจรขยายจะใช้ไอซี ULN 2803 ที่สามารถขยายสัญญาณให้กระแสเอาต์พุตได้ 500 mA โดยมีการขยายแบบอินเวอร์ส เช่น ถ้าได้รับไฟบวก (Vcc) "1" เข้าขา 1 ก็จะออกที่ขา 18 เป็นกราวนด์ "0" แต่ถ้าจ่ายแรงดัน 0 โวลท์ "0" ให้แก่ขา 1 ก็จะออกที่ขา 18 เป็นแรงดันไฟบวก(Vcc) "1"

ในการทดลองครั้งแรกไม่ต่อขา 10 เข้าไฟบวก (Vcc) จะทำให้มีกระแสไหลผ่านมอเตอร์น้อยมาก แต่เมื่อต่อขา 10 ของ ULN 2803 เข้าไฟบวก (Vcc) จะทำให้ได้กระแสสูงขึ้นและสามารถขับมอเตอร์ได้

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1 สรุปผลการทดลอง

สรุปผลที่ได้จากการทดลองที่ได้นั้นพบว่า สามารถที่จะนำ8051 (MCS-51) ไปประยุกต์ใช้งานในการควบคุมการทำงานของระบบนี้ได้ โดยที่นิ้วมือกลจะเคลื่อนที่โดยที่การใช้มอเตอร์สเต็ป (stepping motor) เป็นตัวขับเคลื่อน เพื่อปรับให้นิ้วมือกลเคลื่อนที่ตามที่ต้องการได้ โดยใช้น๊อตช่วยในการเคลื่อนที่ และใช้ 8051 เป็นตัวคอนโทรลมอเตอร์ ส่วนของวงจรควบคุมจะใช้วงจรเปรียบเทียบแรงดันเป็นตัวกำหนดการทำงานของมอเตอร์ และส่วนของตัวควบคุมจะใช้ความต้านทานปรับค่าได้เป็นตัวปรับแรงดันที่อินพุต ถ้าแรงดันอินพุตมีค่ามากกว่าแรงดันเปรียบเทียบ (Vref) จะได้บิตออกมาเป็น "1" (Vcc) แต่ถ้าแรงดันอินพุตมีค่าน้อยกว่าแรงดันเปรียบเทียบ (Vref) จะได้บิตออกมาเป็น "0" (GND) แล้วนำค่าที่ได้ส่งไปประมวลผลที่ 8051 เพื่อทำการประมวลผลตามที่โปรแกรมที่ได้กำหนดไว้ เมื่อมีการเปลี่ยนแปลงของอินพุต 8051 จะทำการตรวจสอบและควบคุมการทำงานของนิ้วมือได้เองโดยอัตโนมัติ

5.2 วิจารณ์ผลการทดลอง

จากการทดลองพบข้อผิดพลาดหลายประการที่พบระหว่างการทำโครงงานนี้

5.2.1 ส่วนของวงจร

ส่วนของวงจรเกิดการผิดพลาด คือ เมื่อทำการทดลองในแผงวงจรแล้วให้ผลตามที่ต้องการ แต่เมื่อนำมาใช้งานจริง การกระตุ้นของสัญญาณอินพุตไม่สามารถที่จะจ่ายกระแสได้สูงเพียงพอต่อความต้องการของ 8051 จึงต้องทำการต่อทรานซิสเตอร์ช่วยในการจ่ายกระแสอีกที

5.2.2 ส่วนของโปรแกรม

ส่วนของโปรแกรมเกิดการผิดพลาด คือ การออกแบบและการทดลองจริงให้ผลที่ไม่ตรงตามต้องการจึงต้องทำการแก้ไขโปรแกรมซ้ำหลายครั้ง แล้วทำการโปรแกรมลงใน 8051 (MCS-51) แล้วนำมาทดสอบว่าได้ผลตามต้องการหรือไม่ โดยทำการโปรแกรมและทดลองจนกว่าจะได้ผลตามที่ต้องการ

ส่วนข้อผิดพลาดของโปรแกรมที่สำคัญอีกอย่างหนึ่ง คือ การคำนวนในการหน่วงเวลาของความถี่เพื่อไปควบคุมความเร็วในการหมุนของมอเตอร์ เพื่อให้ได้ความเร็วรอบและแรงบิดสูงสุด

5.3 การพัฒนา

ส่วนที่จะพัฒนาออกไปอีก

5.3.1 ส่วนของถุงมือตรวจจับ

ส่วนของถุงมือตรวจจับจะต้องมีการใช้อุปกรณ์ตรวจจับที่มีความสวยงามในการประกอบที่ดีกว่านี้ เช่นการใช้ตัวต้านทานสแตนเกต ที่สามารถเปลี่ยนแปลงค่าความต้านทานไปเมื่อถูกทำให้โก่งงอ จึงจะทำให้การทำถุงมือตรวจจับได้สวยงามขึ้น

5.3.2 ส่วนของวงจรเปลี่ยนสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอล

ในส่วนนี้ควรจะใช้วงจรเปลี่ยนสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอลที่มีจำนวนบิตของสัญญาณออก หลายบิตกว่านี้เพื่อทำให้การขยับของนิ้วมือกลเป็นไปได้อย่างละเอียดมากกว่านี้

5.3.3 ส่วนของวงจรควบคุมและโปรแกรม

ในส่วนนี้ควรใช้โปรแกรมที่มีการรับสัญญาณเข้า (Input) ที่เป็นลอจิก "0" ซึ่งการรับสัญญาณเข้าแบบนี้จะไม่ต้องใช้กระแสในการทริก 8051 ที่สูงมากเกินไป ซึ่งจะส่งผลให้ไม่ต้องใช้วงจรขยายกระแสเพื่อช่วยในการทริก ดังนั้นวงจรควบคุมก็จะเปลี่ยนไปด้วย

บรรณานุกรม

1.รศ.สมยศ จุณณะปิยะ. 2541. **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล Mcs51**. คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,พิมพ์ครั้งที่ 2.

2.รศ.ดร.โยธิน เปรมปราณีรัตต์. 2532. "Steppingmotor". **วารสารวิศวสารลาดกระบัง**. ปีที่9 ฉบับที่ 4 : หน้า 46-56.

3.ผศ. พิพัฒน์ เลาหสงคราม. 2532. "ไมโครคอนโทรเลอร์ mcs51". วารสารวิศวสารลาดกระบัง. ปีที่ 9 : หน้า 60-68

4.Neil J. Rubenking. 1997. **DELPHI PROGRAMMING FOR DUMMIES.** พิมพ์ครั้งที่ 2

5.ปรเมษฐ์ ประนยานันทน์. ปิยพงศ์ เผ่าวณิช. "คู่มือประยุกต์การใช้งาน ไมโครคอนโทรลเลอร์ Mcs-51"

ภาคผนวก

โปรแกรมที่ใช้ในโครงงาน . Asm

```
p1 equ 090h
        p3 equ 0b0h
        s1 equ p3.5
        s2 equ p3.4
        org 0000h
      mov a,#11h
      mov r7,#50h
az:     rr a
        mov p1,a
        acall de
        djnz r7,az
pb:     setb s1
        nop
      jb s1,pb
      mov r6,#00h
cc:     rl a
        mov p1,a
        acall de
        djnz r6,cc
dt:     setb s2
        nop
        jnb s2,ff
        setb s1
        nop
        jnb s1,dt
      mov r6,#00h
eo:     rr a
        mov p1,a
        acall de
        djnz r6,eo
        ajmp pb

ff:   mov r4,#00h
  g:      rl a
          mov p1,a
          acall de
          djnz r4,g
  h:      setb s2
          nop
          acall de
          jnb s2,h
        mov r4,#00h
  i:      rr a
          mov p1,a
          acall de
          djnz r4,i
          ajmp dt

de:   mov r0,#0a0h
d1:   mov r1,#00ah
d2:   mov r2,#00ah
   d3:    djnz r2,d3
          djnz r1,d2
          djnz r0,d1
          ret
          end
```

โปรแกรมที่ใช้ในโครงงาน . Hex

:1000000074117F5003F590114DDFF9D2B50020B582

:10001000FA7E0023F590114DDEF9D2B40030B41110

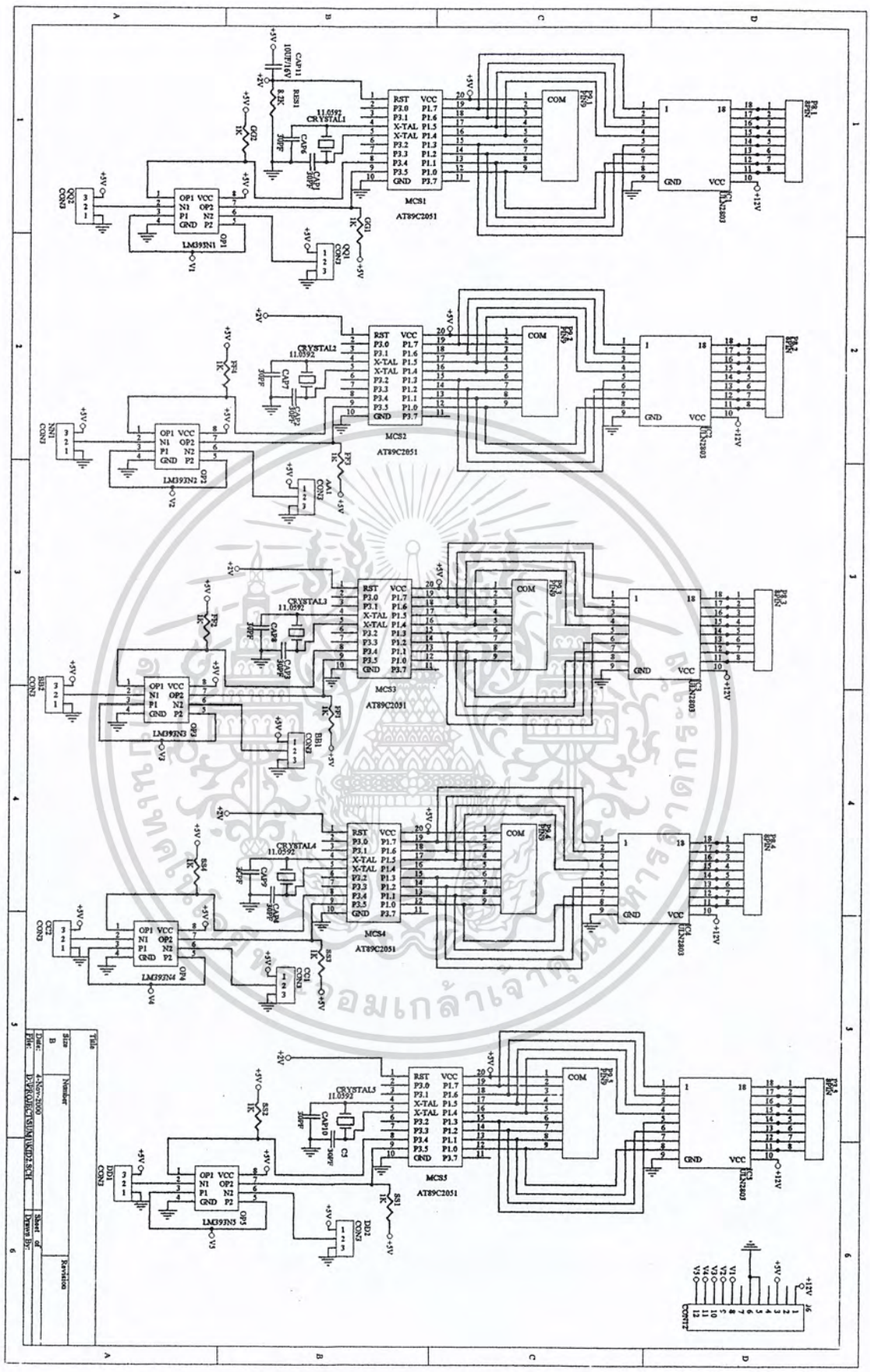
:10002000D2B50030B5F47E0003F590114DDEF90134

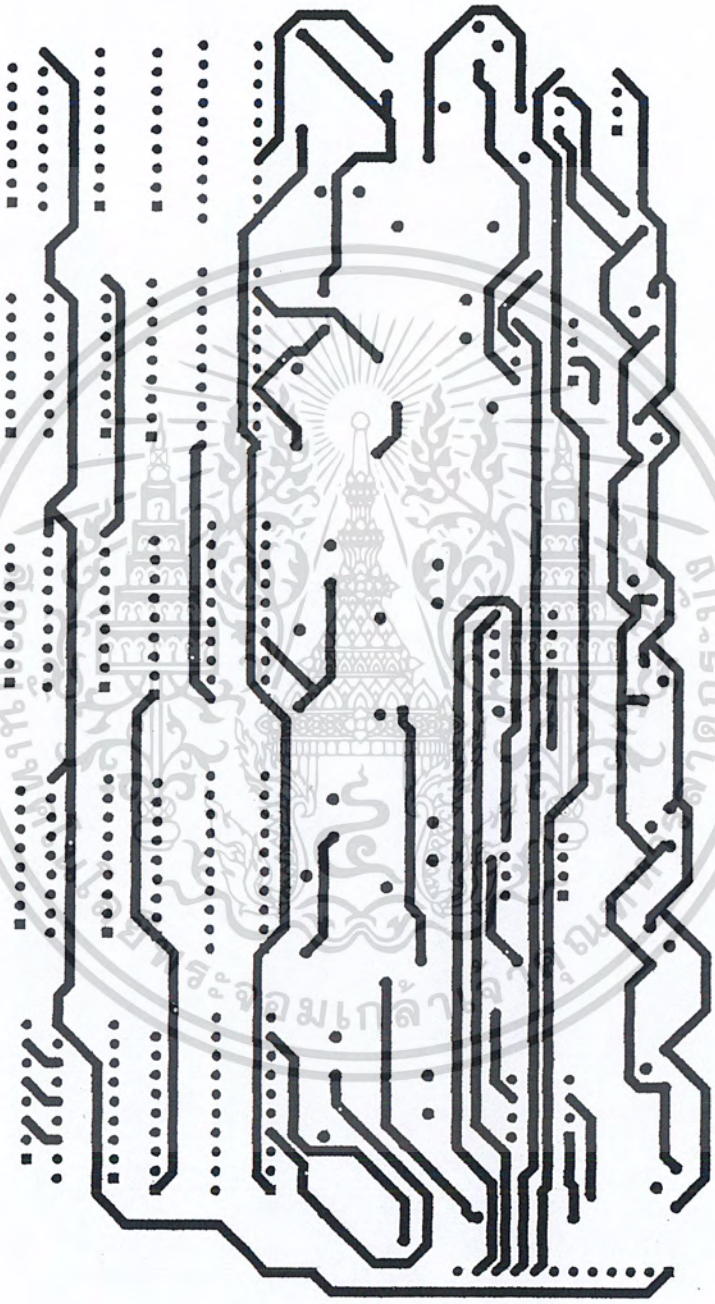
:100030000B7C0023F590114DDCF9D2B400114D304A

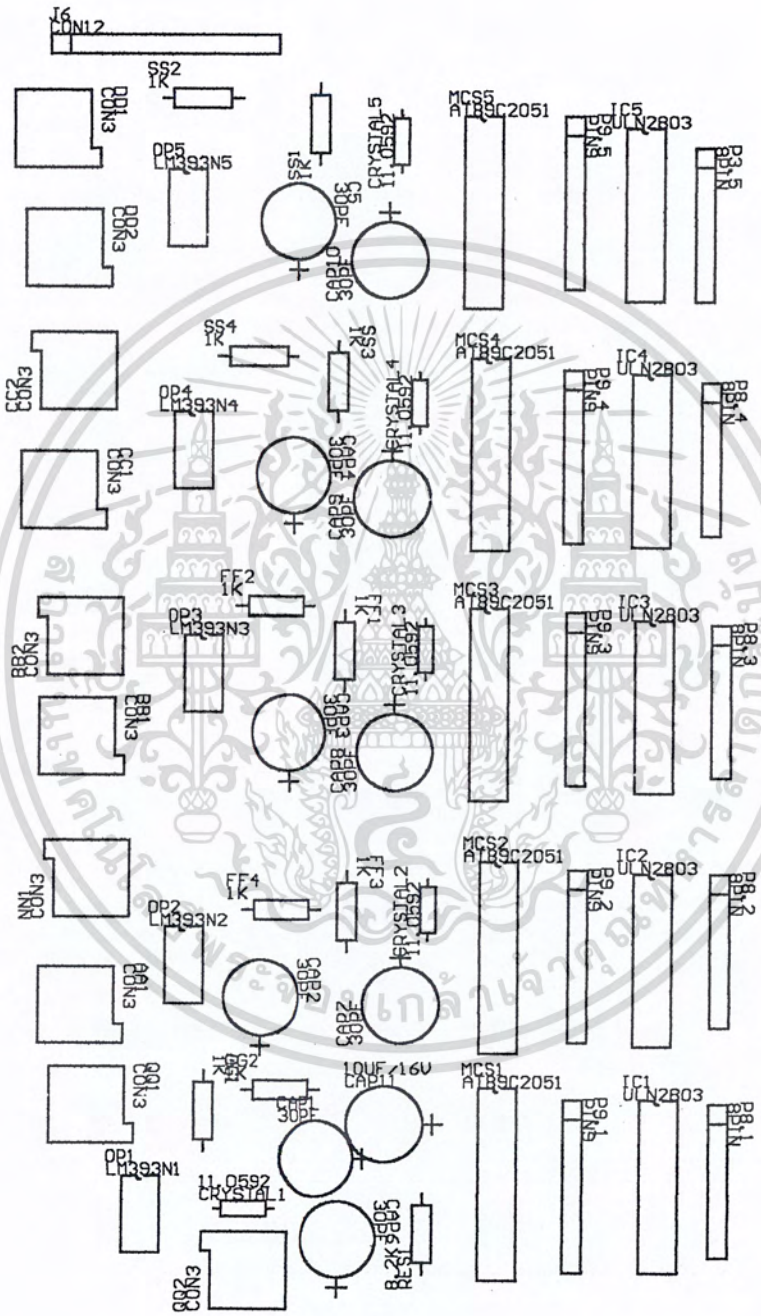
:10004000B4F87C0003F590114DDCF9011A78A07921

:0A0050000A7A0ADAFED9FAD8F6227D

:00000001FF







Features

- Compatible with MCS-51™ Products
- 2K Bytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-chip Analog Comparator
- Low-power Idle and Power-down Modes

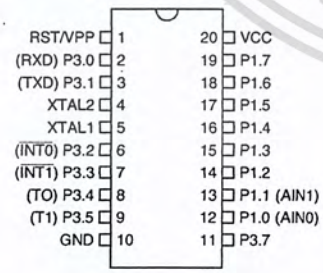
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

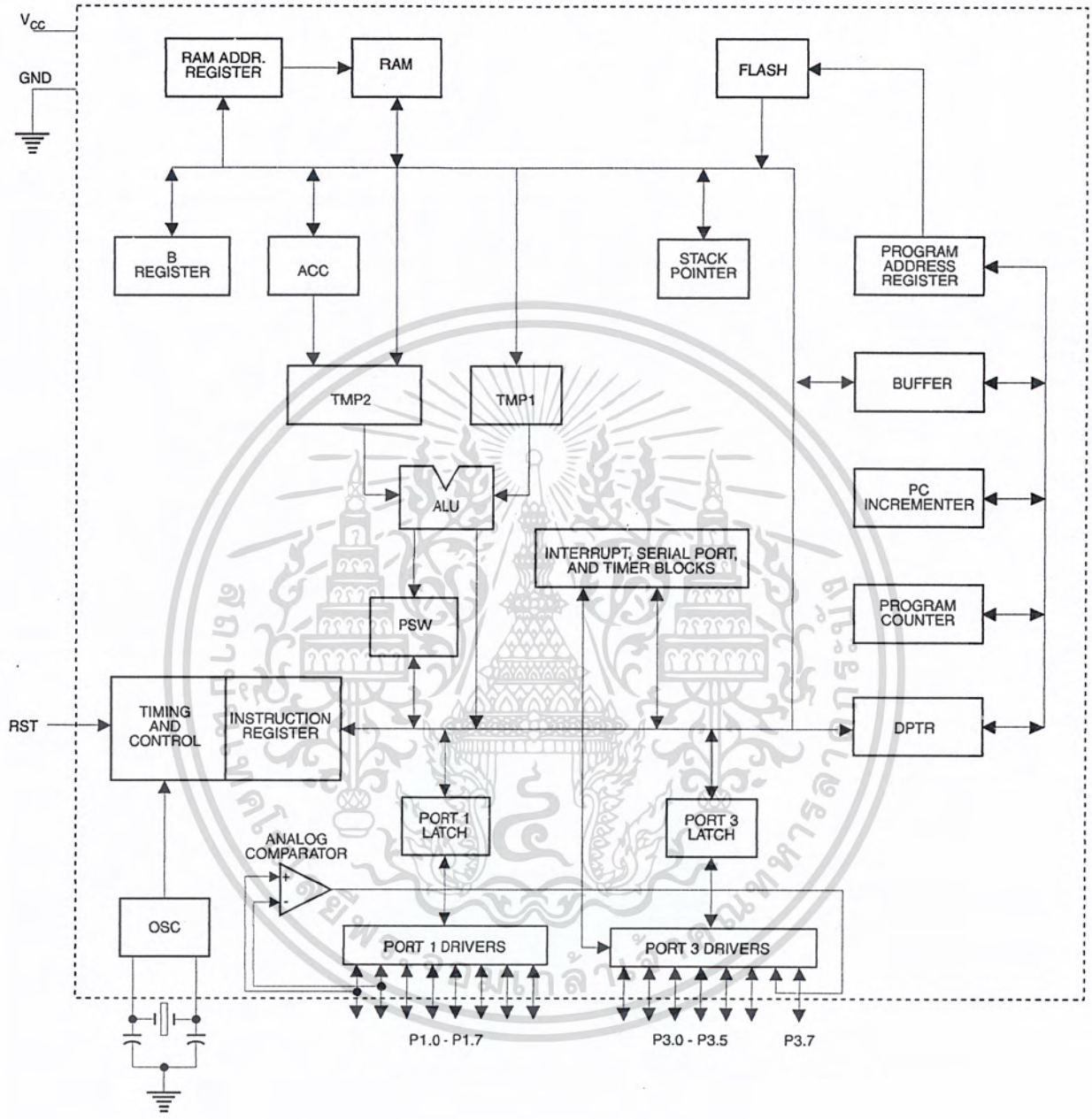


8-bit Microcontroller with 2K Bytes Flash

AT89C2051

Rev. 0368E–02/00

Block Diagram



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 1

Port 1 is an 8-bit bi-irectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3

Port 3 pins P3.0 to P3.5, P3.7 are seven bi-irectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST

Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

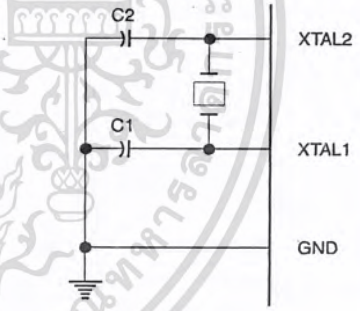
XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

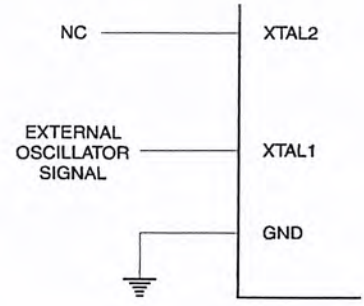
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Restrictions on Certain Instructions

The AT89C2051 and is an economical and cost-effective member of Atmel's growing family of microcontrollers. It contains 2K bytes of flash program memory. It is fully compatible with the MCS-51 architecture, and can be programmed using the MCS-51 instruction set. However, there are a few considerations one must keep in mind when utilizing certain instructions to program this device.

All the instructions related to jumping or branching should be restricted such that the destination address falls within the physical program memory space of the device, which is 2K for the AT89C2051. This should be the responsibility of the software programmer. For example, LJMP 7E0H would be a valid instruction for the AT89C2051 (with 2K of memory), whereas LJMP 900H would not.

1. Branching instructions:

LCALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR

These unconditional branching instructions will execute correctly as long as the programmer keeps in mind that the destination branching address must fall within the physical boundaries of the program memory size (locations 00H to 7FFH for the 89C2051). Violating the physical space limits may cause unknown program behavior.

CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ With these conditional branching instructions the same rule above applies. Again, violating the memory boundaries may cause erratic execution.

For applications involving interrupts the normal interrupt service routine address locations of the 80C51 family architecture have been preserved.

2. MOVX-related instructions, Data Memory:

The AT89C2051 contains 128 bytes of internal data memory. Thus, in the AT89C2051 the stack depth is limited to 128 bytes, the amount of availabie RAM. External DATA memory access is not supported in this device, nor is external PROGRAM memory execution. Therefore, no MOVX [...] instructions should be included in the program.

A typical 80C51 assembler will still assemble instructions, even if they are written in violation of the restrictions mentioned above. It is the responsibility of the controller user to know the physical features and limitations of the device being used and adjust the instructions used correspondingly.

Program Memory Lock Bits

On the chip are two lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

Lock Bit Protection Modes[1]

Program Lock Bits			
	LB1	LB2	Protection Type
1	U	U	No program lock features.
2	P	U	Further programming of the Flash is disabled.
3	P	P	Same as mode 2, also verify is disabled.

Note: 1. The Lock Bits can only be erased with the Chip Erase operation.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

P1.0 and P1.1 should be set to "0" if no external pullups are used, or set to "1" if external pullups are used.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Power-down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

P1.0 and P1.1 should be set to "0" if no external pullups are used, or set to "1" if external pullups are used.

Programming The Flash

The AT89C2051 is shipped with the 2K bytes of on-chip PEROM code memory array in the erased state (i.e., contents = FFH) and ready to be programmed. The code memory array is programmed one byte at a time. *Once the array is programmed, to re-program any non-blank byte, the entire memory array needs to be erased electrically.*

Internal Address Counter: The AT89C2051 contains an internal PEROM address counter which is always reset to 000H on the rising edge of RST and is advanced by applying a positive going pulse to pin XTAL1.

Programming Algorithm: To program the AT89C2051, the following sequence is recommended.

1. Power-up sequence:
   Apply power between V_{CC} and GND pins
   Set RST and XTAL1 to GND
2. Set pin RST to "H"
   Set pin P3.2 to "H"
3. Apply the appropriate combination of "H" or "L" logic levels to pins P3.3, P3.4, P3.5, P3.7 to select one of the programming operations shown in the PEROM Programming Modes table.

To Program and Verify the Array:

4. Apply data for Code byte at location 000H to P1.0 to P1.7.
5. Raise RST to 12V to enable programming.
6. Pulse P3.2 once to program a byte in the PEROM array or the lock bits. The byte-write cycle is self-timed and typically takes 1.2 ms.
7. To verify the programmed data, lower RST from 12V to logic "H" level and set pins P3.3 to P3.7 to the appropiate levels. Output data can be read at the port P1 pins.
8. To program a byte at the next address location, pulse XTAL1 pin once to advance the internal address counter. Apply new data to the port P1 pins.
9. Repeat steps 5 through 8, changing data and advancing the address counter for the entire 2K bytes array or until the end of the object file is reached.
10. Power-off sequence:
    set XTAL1 to "L"
    set RST to "L"
    Turn V_{CC} power off

$\overline{\text{Data}}$ Polling: The AT89C2051 features $\overline{\text{Data}}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P1.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{\text{Busy}}$: The Progress of byte programming can also be monitored by the RDY/$\overline{\text{BSY}}$ output signal. Pin P3.1 is pulled low after P3.2 goes High during programming to indicate BUSY. P3.1 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed code data can be read back via the data lines for verification:

1. Reset the internal address counter to 000H by bringing RST from "L" to "H".
2. Apply the appropriate control signals for Read Code data and read the output data at the port P1 pins.
3. Pulse pin XTAL1 once to advance the internal address counter.
4. Read the next code data byte at the port P1 pins.
5. Repeat steps 3 and 4 until the entire array is read.

The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire PEROM array (2K bytes) and the two Lock Bits are erased electrically by using the proper combination of control signals and by holding P3.2 low for 10 ms. The code array is written with all "1"s in the Chip Erase operation and must be executed before any non-blank memory byte can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 001H, and 002H, except that P3.5 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(001H) = 21H indicates 89C2051

Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	25.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

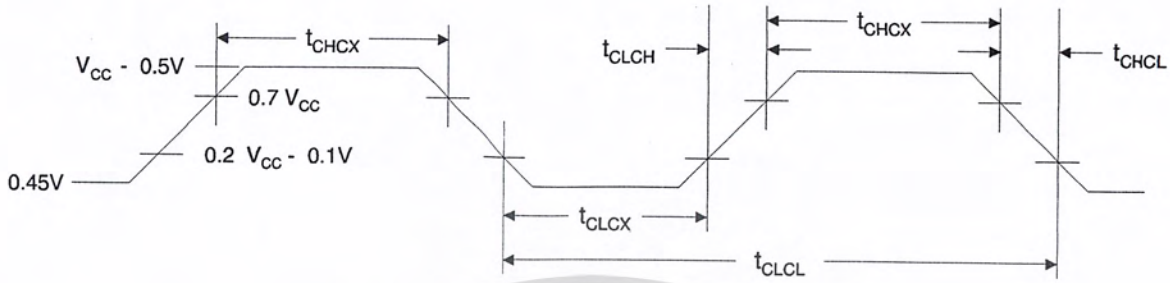
DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.0V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.2 V_{CC} - 0.1	V
V_{IH}	Input High-voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low-voltage[1] (Ports 1, 3)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V		0.5	V
V_{OH}	Output High-voltage (Ports 1, 3)	I_{OH} = -80 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -30 µA	0.75 V_{CC}		V
		I_{OH} = -12 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1, 3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-750	µA
I_{LI}	Input Leakage Current (Port P1.0, P1.1)	0 < V_{IN} < V_{CC}		±10	µA
V_{OS}	Comparator Input Offset Voltage	V_{CC} = 5V		20	mV
V_{CM}	Comparator Input Common Mode Voltage		0	V_{CC}	V
RRST	Reset Pull-down Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz, V_{CC} = 6V/3V		15/5.5	mA
		Idle Mode, 12 MHz, V_{CC} = 6V/3V P1.0 & P1.1 = 0V or V_{CC}		5/1	mA
	Power-down Mode[2]	V_{CC} = 6V P1.0 & P1.1 = 0V or V_{CC}		100	µA
		V_{CC} = 3V P1.0 & P1.1 = 0V or V_{CC}		20	µA

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 20 mA
   Maximum total I_{OL} for all output pins: 80 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

External Clock Drive Waveforms

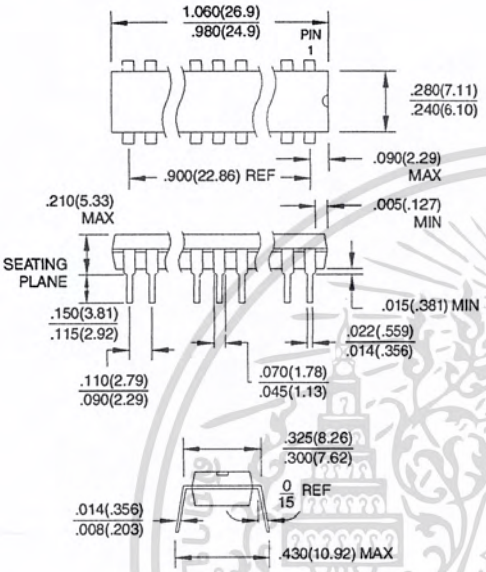


External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	12	0	24	MHz
t_{CLCL}	Clock Period	83.3		41.6		ns
t_{CHCX}	High Time	30		15		ns
t_{CLCX}	Low Time	30		15		ns
t_{CLCH}	Rise Time		20		20	ns
t_{CHCL}	Fall Time		20		20	ns

Packaging Information

20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-001 AD



20S, 20-lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)
Dimensions in Inches and (Millimeters)