เครื่องควบคุมการทำงาน STEPPING MOTOR

โดยไมโครคอนโทรลเลอร์ 8032

โดย

นาย รุ่งเรือง บุคราช

นาย สมบัติ สว่างอารมย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาครุศาสตรอุตสาหกรรมศาสตรบัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เครื่องควบคุมการทำงาน STEPPINGMOTOR
โดยไมโครคอนโทรลเลอร์ 8032

นายรุ่งเรือง บุตราช
นายสมบัติ สว่างอารมย์

อาจารย์ที่ปรึกษา

ผศ.วิทยา ทิพย์สุวรรณพร

ปีการศึกษา 2538

บทคัดย่อ

ปริญานิพนธ์ฉบับนี้ จะกล่าวถึงการควบคุมสเต็ปปิ้งมอเตอร์(STEPPING MOTOR) ซึ่งจะประกอบด้วยส่วนของซอร์ฟแวร์และฮาร์ดแวร์ โดยส่วนซอร์ฟแวร์จะเป็นโปรแกรมที่ควบคุมการทำงานทั้งหมดที่จะควบคุมให้สเต็ปปิ้งมอเตอร์หมุนตามที่ต้องการ รวมทั้งโปรแกรมกราฟแสดงภาพบนหน้าจอ ส่วนฮาร์ดแวร์แบ่งเป็นภาคต่างๆคือ ภาคควบคุม ภาคดีโค้ดพอร์ทใช้งาน ภาคควบคุมสเต็ปปิ้งมอเตอร์ โครงงานนี้สามารถนำไปประยุกต์ใช้งานต่างๆได้

ABSTRAC

THIS THESIS ABOUT CONTROLLING STEPPING MOTOR. THERE ARE CONSIST OF HARDWARE AND SORFWARE. THE SORFWARE ARE ALL CONTROL PROGRAM AND GRAPHIC PROGRAM. THE HARDWARE ARE SECTION OF CONTROLLER,INTERFACE AND DRIVESTEP PING MOTOR. WE CAN DEVELOPE THIS PROJECT WITH OTHER APPLICATION IN THE WORKS.

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องควบคุมการทำงานของสเต็ปปิ้งมอเตอร์

โดยใช้ไมโครคอนโทรลเลอร์ 8032

ผู้จัดทำ

1.นายรุ่งเรือง บุตราช รหัส 36013301

2.นายสมบัติ สว่าอารมย์ รหัส 36012120

.....อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์วิทยา ทิพย์สุวรรณพร)

สารบัญ

สารบัญต่อ

สารบัญต่อ

หน้า

บทที่ 1

ความมุ่งหมายของรายงาน

1.1 บทนำ

ปัจจุบันเทคโนโลยีทางด้านคอมพิวเตอร์ก้าวหน้าไปอย่างไม่หยุดยั้ง ไม่ว่าจะเป็นทางด้านเทคโนโลยีของการผลิตอุปกรณ์ไมโครโปรเซสเซอร์หรือเทคโนโลยีทางด้านการนำไปใช้งาน และมีหลายบริษัทที่ผลิตอุปกรณ์ออกมาแข่งขันกันในตลาดของไมโครคอมพิวเตอร์ซึ่งเป็นตลาดที่มีการแข่งขันสูงมาก ฉะนั้นเราจะต้องติดตามวิวัฒนาการเหล่านี้เพื่อจะได้ไม่ล้าหลัง การศึกษาวิชาการทางด้านนี้จึงมีความจำเป็นมาก โครงงานนี้ก็เป็นส่วนหนึ่งซึ่งเป็นโครงงานที่ทำการศึกษาและพัฒนาไมโครโปรเซสเซอร์ตระกูล 8032 เพื่อนำมาใช้งานและประยุกต์เข้ากับงานในด้านการควบคุมให้มีความง่ายและสะดวกต่อการใช้งานและรวมไปถึงประสิทธิภาพของงานที่ต้องการควบคุมด้วย

1.2 ทำไมต้องเป็น 8032

ในบรรดาชิปไมโครโปรเซสเซอร์ที่นำมาประยุกต์ใช้งานและพบเห็นกันอยู่เป็นประจำในงานคอนโทรลส่วนใหญ่จะเป็นของบริษัท INTEL และ ZILOG ส่วน MOTOROLA นั้นมีการนำมาใช้กันน้อยมาก ซึ่งในที่นี้จะกล่าวเฉพาะไมโครโปรเซสเซอร์ขนาด 8 บิท INTEL และ ZILOG เท่านั้น

ชิปไมโครโปรเซสเซอร์ (CPU) อาจแบ่งออกได้ 2 ประเภทคือ ไมโครโปรเซสเซอร์ (Micro Processor) และไมโครคอนโทรลเลอร์ (Micro Controller) ไมโครโปรเซสเซอร์ประกอบด้วยหน่วยควบคุม (Control Unit) และหน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU : Arithmatic and Logic) เป็นต้น ส่วนไมโครคอนโทรลเลอร์ประกอบไปด้วย Control Unit, ALU, หน่วยความจำ (Memory), อินพุทเอาท์พุทพอร์ท (I/O Port) เป็นต้น ซึ่งอาจจะกล่าวได้ว่าไมโครโปรเซสเซอร์ก็คือส่วนหนึ่งของไมโครคอนโทรลเลอร์นั่นเอง

ในบรรดาชิปไมโครโปรเซสเซอร์ขนาด 8 บิท INTEL มีชิปที่เป็นไมโครโปรเซสเซอร์ได้แก่เบอร์ 8080, 8085 และไมโครคอนโทรเลอร์ 8048(MCS-48), 8032(MCS-51) ส่วนของ ZILOG นั้นมีไมโครโปรเซสเซอร์ขนาด 8 บิทเพียงเบอร์เดียวคือ Z80 ซึ่งผลิตออกมาเพื่อจะเข้ามาแข่งกับ 8080 ของ INTEL โดยเฉพาะ เพราะไม่เพียงแต่มีชุดคำสั่งของ Z80 ที่ครอบคลุมคำสั่งของ 8080 ได้หมดแล้ว Z80 ยังมีรีจิสเตอร์ภายในมากมายทั้งหมด 16 บิท และ 8 บิท

และชุดคำสั่ง 158 คำสั่งในขณะที่ 8080 มีเพียง 78 คำสั่งเท่านั้น ชุดคำสั่งของ Z80 มีข้อดีตรงที่มีชุดคำสั่งที่ใช้กระทำกับข้อมูลเป็น Block ซึ่งสามารถนำไปใช้ได้ดีกับระบบที่ต้องการฐานข้อมูล (Data Base) ส่วนทางด้านสถาปัตยกรรมของ Z80 สามารถอ้างแอดเดรสของ Rom/Ram ได้ 64 เคไบท์ ต่อ I/O พอร์ทได้ 256 พอร์ทและสามารถรับสัญญาณอินเตอร์รัพท์ได้ 2 แหล่งเท่านั้น คือ NMI และ INT

ส่วน INTEL ก็มี CPU เบอร์ 8032 (MCS-51) ซึ่งพอจะทาบรัศมีกับ Z80 ของ ZILOG ได้และอาจจะมาทดแทน Z80 ก็ได้ 8032 ของ INTEL เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิท มีชุดคำสั่งทั้งหมด 111 คำสั่งมีรีจิสเตอร์ขนาด 8 บิทให้ใช้ถึง 4 Bank (Bank ละ 8 ตัว R0-R7) และมีรีจิสเตอร์ B ใช้ในคำสั่งคูณและหาร ชุดคำสั่งของ 8032 นี้มีข้อดีตรงที่แต่ละคำสั่งใช้เวลาในการทำงานสั้นและที่เด่นชัดมากก็คือ ชุดคำสั่งที่กระทำเกี่ยวกับบิททั้งหมด ซึ่งให้ความคล่องตัวมากกว่า ส่วนทางด้านสถาปัตยกรรมของ 8032 นั้นภายใน CPU นอกจากจะมี Control Unit และ ALU เหมือนที่มีใน Z80 แล้วยังประกอบไปด้วยหน่วยความจำ Rom ภายในขนาด 4 Kbyte, หน่วยความจำ Ram ภายในขนาด 128 ไบท์, ไทเมอร์/เคาน์เตอร์ขนาด 16 บิท 2 แชนแนล, อินพุท/เอาท์พุทพอร์ท 32 บิท (4 Port), พอร์ทอนุกรม (Serial Port) 1 ชุด, สามารถรับสัญญาณอินเตอร์รัพท์ได้ถึง 6 แหล่ง (IE0, TF0, IE1, TF1, RI& TI) สามารถอ้างแอดเดรสของหน่วยความจำโปรแกรม (Program Memory) ได้ถึง 64 Kbyte และสามารถอ้างถึงแอดเดรสของหน่วยความจำข้อมูล (Data Memory) ได้ถึง 64 Kbyte เช่นกัน ส่วน I/O พอร์ทนั้นใน 8032 จะใช้เทคนิค Memory Map I/O ซึ่งก็คือการใช้แอดเดรสของหน่วยความจำบางส่วนเป็นแอดเดรสของ I/O พอร์ท

จากคุณสมบัติที่กล่าวมาทั้งหมดของ Z80 และ 8032 นั้นพอสรุปขอบเขตการใช้งานและความแตกต่างของ CPU ทั้งสองเบอร์นี้ได้คร่าว ๆ ดังในรูปที่ 1.1

Z80 นั้นเป็นชิปไมโครโปรเซสเซอร์ ซึ่งมีชุดคำสั่งมากมายและมีรีจิสเตอร์ให้ใช้มากอีกด้วย ทำให้สะดวกในการใช้งาน แต่ในบางครั้งก็สร้างความสับสนได้ไม่น้อยทีเดียว เพราะทำให้ไม่รู้ว่าจะใช้คำสั่งใดกับรีจิสเตอร์ตัวใดดี อีกทั้งส่วนฮาร์ดแวร์ของ Z80 นั้นระบบพื้นฐานที่ต่อใช้งานได้จะต้องประกอบด้วยชิปไอซีอย่างน้อยอีก 5 ตัว คือ ROM,RAM,I/O PORT,DECODER,CLOCK, OSILLATORME ทำให้ไม่เหมาะกับระบบควบคุมขนาดเล็กเพราะสิ้นเปลืองเนื้อที่และค่าใช้จ่ายด้วย และเนื่องจากระบบบัสของ Z80 เป็นแบบแยก DATA BUS และADDRESS BUS ออกจากกันซึ่งทำให้สะดวกในการ INTERFACE แต่ในด้านเสถียรภาพของระบบแล้ว สามารถทำให้ระบบถูกรบกวน

CPU	8032	Z80
CLOCK	12 MHZ	4 MHZ
INSTRUCTION SET	111 คำสั่ง	128 คำสั่ง
PROGRAMMEMORY(ROM)	64 KBYTE	รวมกันทั้ง ROMและRAM
DATA MEMORY (RAM)	64 KBYTE	ไม่เกิน 64 KBYTE
I/O PORT (8 BIT)	MEMORY MAP I/O	256 PORT
INTERNAL ROM	4 KBYTE	–
INTERNAL RAM	128 KBYTE	–
INTERNAL I/O PORT	32 I/O LINE (4PORT)	–
TIMER/COUNTER	16 BIT 2 CHANNEL	–
SERIAL PORT	FULL DUPLEX	–
CLOCK OSCILLATOR	อยู่ในชิป	–
INTERRUPT	6 SOURCE 2 LEVEL	2 SOURCE

รูปที่ 1.1 ตารางเปรียบเทียบคุณสมบัติของ 8032 กับ Z80

และเกิดการแซกซ้อนของสัญญาณทั้งจากภายนอกและภายใน CPU เองได้มากดังนั้นงานที่เหมาะจะนำ Z80 ไปใช้ส่วนใหญ่จะเป็นงานที่เกี่ยวข้องกับฐานข้อมูล มีความสลับซับซ้อนมาก และเหมาะกับระบบควบคุมขนาดค่อนข้างใหญ่ และจำเป็นต้องมีระบบ Watch Dog (Watch Dog คือวงจรที่ออกแบบขึ้นเพื่อป้องกันการทำงานผิดพลาดของระบบควบคุมที่ใช้ไมโครโปรเซสเซอร์ เช่น การหลุดจากลูปโปรแกรมควบคุมหรือเกิดอาการ Hang หากมีการทำงานผิดพลาดจากโปรแกรมที่กำหนดวอทช์ด็อกจะทำการรีเซ็ทไมโครโปรเซสเซอร์เพื่อให้เริ่มต้นทำงานใหม่ การตรวจสอบการทำงานของระบบไมโคร อาศัยหลักการที่ลูปโปรแกรมของระบบไมโครส่งพัลส์มาทริกวอทช์ด็อกเป็นระยะหากพัลส์ทริกหายไปชั่วขณะหนึ่ง เวลาที่ตั้งไว้แสดงว่าระบบไมโครหลุดจากโปรแกรมควบคุมแล้ววอทช์ด็อกก็จะรีเซ็ทระบบทันที ทำให้ระบบทำงานได้ตามปกติ) เข้ามาร่วมด้วยเพื่อเสถียรภาพการทำงานให้ดีขึ้น

8032 เป็นชิปไมโครคอนโทรลเลอร์มีชุดคำสั่งที่เอื้ออำนวยต่อการใช้งานในระดับบิทได้ดีเลิศ ซึ่งเหมาะสำหรับงานควบคุมโดยเฉพาะ (ชื่อก็บอกอยู่แล้วว่าไมโครคอนโทรลเลอร์) และแต่ละคำสั่งจะใช้เวลาในการทำงานไม่เกิน 2 ไมโครวินาที (ที่ Clock 12 MHz) ทำให้โปรแกรมที่เขียนขึ้นมาทำงานได้รวดเร็วมาก ส่วนฮาร์ดแวร์ของ 8032 นั้นระบบพื้นฐานที่ต่อใช้งานได้นั้นไม่จำเป็นต้องต่อชิปไอซีอื่นใดเพิ่มเติม เพราะภายใน CPU 8032 นี้มีทั้ง Rom, Ram ในตัวอยู่แล้ว และมี I/O Port, Serial Port, Timer/Counter ซึ่งสามารถนำไปใช้งานได้อย่างกว้างขวาง อีกทั้งระบบบัสของ 8032 เป็นแบบ Multiplex Bus คือทั้ง Data และ Address ใช้ Bus ร่วมกัน ซึ่งอาจทำให้ไม่สะดวกต่อการขยายระบบหรือต่อหน่วยความจำหรือ I/O พอร์ทภายนอกเพราะต้องมีชิปไอซีอีก 1 ตัวไว้ทำหน้าที่ Latch Address แต่ด้วยวิธีนี้ที่มีการเปลี่ยนแปลงทั้ง Data และ Address อยู่ตลอดเวลา อีกทั้งเนื้อที่การใช้งานของ Stack ถูกจัดให้อยู่ในหน่วยความจำ Ram ภายในของ CPU รวมทั้ง Clock OScillator ก็มีอยู่ภายใน CPU ทำให้ 8032 มีอัตราเสี่ยงต่อสัญญาณรบกวนลดลงซึ่งมีผลทำให้ระบบมีเสถียรภาพดีเยี่ยม โดยในบางงานระบบ Watch Dog อาจจะไม่มีความจะเป็น แต่หากต้องการความมั่นใจสูงยิ่งขึ้นก็สามารถสร้างระบบ Watch Dog ขึ้นมาโดยไม่ต้องต่อชิปไอซีเพิ่มเติมแต่อย่างใด โดยใช้ Timer/Counter ที่มีอยู่ภายในมาใช้งานแทนและที่สำคัญสามารถต่อหน่วยความจำได้สูงสุด (Rom และ Ram) ถึง 128 Kbyte ส่วน I/O พอร์ท นั้นสามารถขยายได้มากมาย (เกิน 256 พอร์ท) และใช้เทคนิค Memory Map I/O ดังนั้น จะเห็นได้ว่าระบบควบคุมที่ใช้ 8032 เป็น CPU นั้นสามารถทำงานได้คล่องตัวกว่า, เสถียรภาพดีกว่า, มีขนาดเล็กกะทัดรัดและประหยัดค่าใช้จ่ายด้วย

บทที่ 2

8032 ไมโครโปรเซสเซอร์

2.1 บทนำ

8032 ไมโครโปรเซสเซอร์ เป็นไมโครโปรเซสเซอร์ในตระกูล MCS-51 ที่มีขนาด 8 บิท ประกอบด้วยไมโครโปรเซสเซอร์เบอร์ต่างๆ ดังรูปที่ 2.1 ทุก ๆ เบอร์จะมีสถาปัตยกรรมพื้นฐานเหมือนกัน แต่เดิม 8032 ถูกสร้างด้วยวิธี HMOS I แต่ในปัจจุบันได้สร้างด้วยวิธี HMOS II จึงมีชื่อเป็น 8032AH ไมโครโปรเซสเซอร์ในตระกูล 51 นั้นถึงแม้ว่าจะมีหลายเบอร์แต่เราก็จะเรียกว่าเป็น "8032" ซึ่งเราจะใช้ชื่อ 8032 นี้โดยตลอดเมื่อหมายถึงไมโครโปรเซสเซอร์ตระกูล 51 ส่วนเบอร์ 8032 และ 8052 มีหน่วยความจำภายในเพิ่มขึ้นและมีวงจร Timer/Counter ขนาด 16 บิท เพิ่มขึ้นดังแสดงในรูปที่ 2.1

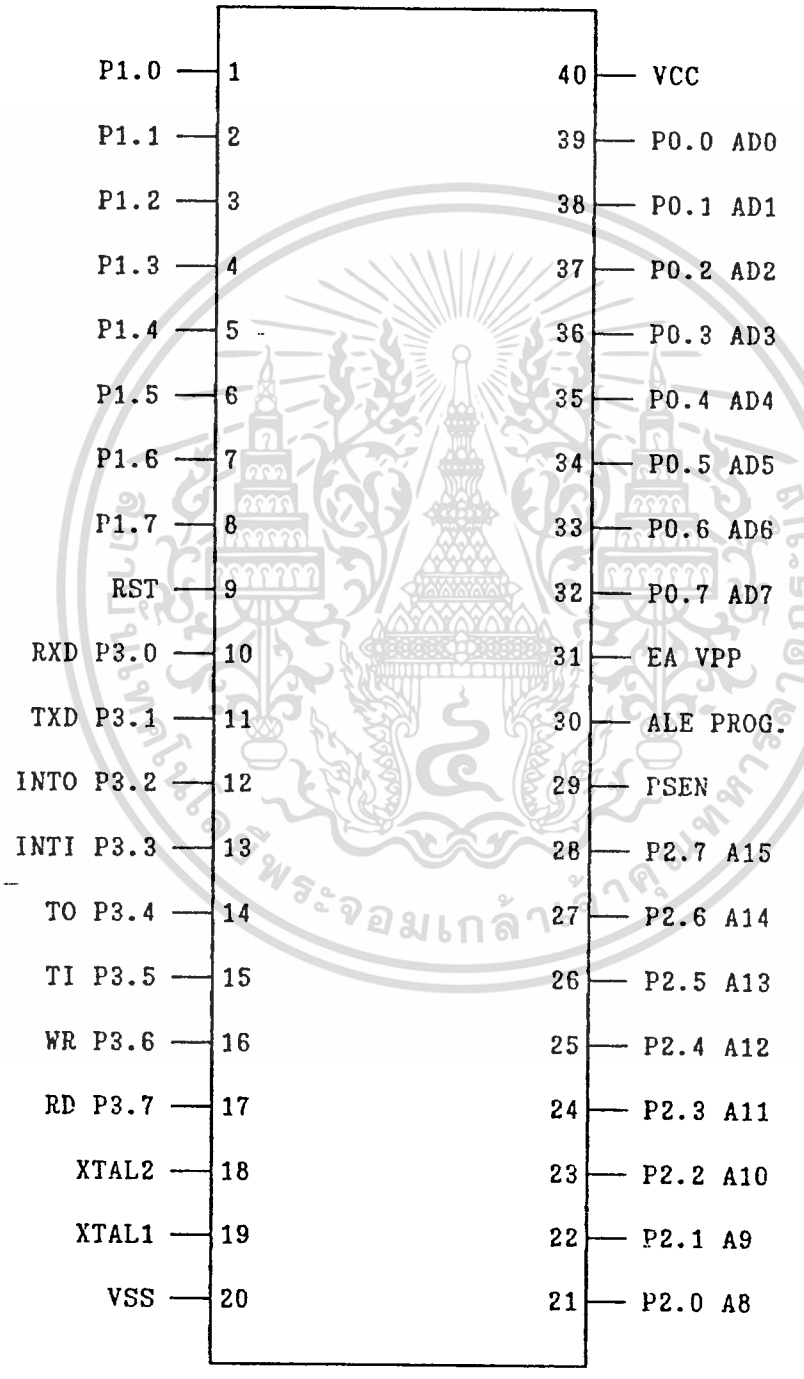
Device	Internal Memoey		Timer/Counters	Interrupt
	Program	Data		
8052AH	8Kx8 ROM	256x8 RAM	3 x 16 Bit	6
8032AH	4Kx8 ROM	128x8 RAM	2 x 16 Bit	5
8051	4Kx8 ROM	128x8 RAM	3 x 16 Bit	5
8032AH	None	256x8 RAM	3 x 16 Bit	6
8031AH	None	128x8 RAM	2 x 16 Bit	5
8031	None	128x8 RAM	2 x 16 Bit	5
8751H	4Kx8 EPROM	128x8 RAM	2 x 16 Bit	5
8751H-8	4Kx8 EPROM	128x8 RAM	2 x 16 Bit	5

รูปที่ 2.1 ไอซีเบอร์ต่าง ๆ ในตระกูล 8032

คุณสมบัติสำคัญของไมโครโปรเซสเซอร์ 8032 จะประกอบด้วย

1. ใช้ HMOS เทคโนโลยีสร้าง และทำงานได้ด้วยแหล่งจ่ายไฟขนาด 5 โวลท์แหล่งเดียว
2. ซีพียู (CPU) มีขนาด 8 บิท
3. มีวงจรออสซิลเลเตอร์ (Oscillator) และวงจรนาฬิกาบนชิป
4. ชุดแบงค์ (Bank) รีจิสเตอร์มีขนาด 4 ชุดด้วยกัน โดยมีชุดละ 8 รีจิสเตอร์
5. ตัวตั้งเวลาและตัวนับขนาด 16 บิทมี 2 ตัว
6. พอร์ทขนาน I/O แบบสองทิศทางพอร์ทละ 8 บิท จำนวน 16 เส้น และอีก 16 เส้น ใช้ในการเข้าถึงทางแอดเดรสและข้อมูล
7. พอร์ทอนุกรมสามารถที่จะโปรแกรมการรับส่งแบบ Full Duplex ที่ความเร็วสูง
8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1 ไมโครวินาที ด้วยการใช้คริสตัล 12 เม็กกะเฮิรตซ์
9. มีหน่วยความจำสำหรับเก็บโปรแกรมภายในขนาด 4 Kbyte
10. แอดเดรสข้อมูลภายนอกได้ 64 Kbyte แอดเดรสโปรแกรมนอกได้ 64 Kbyte
11. มีซอฟต์แวร์แฟลก (Flag Software) สำหรับผู้ใช้ที่กำหนดเองได้ถึง 128 ตำแหน่งบิท
12. โครงสร้างอินเทอร์รัพท์ (Interrupt) ทำได้ 6 แหล่ง พร้อมด้วยการจัดไพรโอริตี้ (Priority) ได้ 2 ระดับ
13. สามารถกำหนดเลขที่อยู่ข้อมูลขนาดไบท์ หรือบิทได้โดยตรง
14. ตัวโปรเซสเซอร์สามารถใช้งานแบบบูลีน (Boolean) ได้ สำหรับการใช้งานควบคุม
15. มีคำสั่งคูณและหารทางฮาร์ดแวร์ทำได้ภายใน 4 ไมโครวินาที
16. ตัวเลขทางคณิตศาสตร์ ใช้ได้ทั้งแบบไบนารีและเดซิมัล
17. การใช้พื้นที่สแต็ก (Stack) สำหรับโปรแกรมย่อยต่าง ๆ ทำได้กว้างขึ้น
18. ชุดคำสั่งของ MCS-51 มี 111 คำสั่ง

2.2 ลักษณะการจัดขาภายนอกของ 8032



รูปที่ 2.2 ลักษณะขาภายนอกและสัญลักษณ์ทางตรรกของ 8032

จากรูปที่ 2.2 เป็นการจัดขาตามลักษณะภายนอกของ 8032 ซึ่งมีรายละเอียดดังนี้คือ

-ขา V_{ss} ขา 20 เป็นขาสำหรับต่อลงดิน

-ขา V_{cc} ขา 40 ต้องการแรงดันไฟกระแสตรงขนาด 5 โวลท์ จะเข้าที่ขานี้ และใช้สำหรับการโปรแกรม

-ขา Port 0 (P0.0-P0.7/AD0-AD7) ขา 32-39 เป็นพอร์ท I/O 8 บิทแบบ Open Drain Bidirectional สามารถที่จะรับโหลด TTL ได้ 8 ตัว การเขียนค่า "1" ไปที่พอร์ทนี้ จะเป็นการลอย (Float) ขาของพอร์ทนี้ ทำให้มันทำงานเป็นอินพุท มีสถานะอิมพีแดนส์สูงในการให้พอร์ทนี้บริการแบบ I/O พอร์ท 0 จพทำงานเป็นมัลติเพล็กซ์ด้วยสัญญาณแอดเดรสไบท์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอกในการใช้งานนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพ พรอ์ท 0 ยังใช้งานเป็นตัวส่งข้อมูลออกทางพอร์ทนี้

-ขา Port 1 (P1.0-P1.7) ขา 1-8 เป็นพอร์ท I/O 8 บิท Open Drain Bidirectional พร้อมด้วยการพูลอัพภายใน ถ้าเป็นพอร์เอาท์พุท บัฟเฟอร์สามารถขับโหลด TTL ตระกูลแอลเอสได้ 4 ตัว พอร์ท 1 เมื่อถูกเขียนค่า "1" ด้วยโปรแกรมมันจะมีสถานะสูงด้วยการพูลอัพภายใน การให้สถานะเช่นนี้จะเป็นการ Initial ใช้งานพอร์ทนี้ให้เป็นอินพุทขณะที่พอร์ท 1 เป็นอินพุทการให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายใน

-ขา Port 2 (P2.0-P2.7) ขา 21-28 เป็นพอร์ท I/O 8 บิทแบบ Open Drain Bidirection ด้วยการพูลอัพภายในพอร์ท 2 ที่ทำหน้าที่เป็นบัฟเฟอร์เอาท์พุทสามารถจ่ายโหลด TTL ตระกูลแอลเอสได้ 4 ตัว พอร์ทจะถูกใช้งานเป็นตัวส่งแอดเดรสไบท์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอกเพื่อให้ได้แอดเดรสได้ถึง 16 บิท ด้วยการใช้งานแบบนี้ มันจะมีพูลอัพภายในที่ช่วยให้การส่งค่า "1" ได้ระดับที่แน่นอน

-ขา Port 3 (P3.0-P3.7) ขา 10-17 เป็นพอร์ท I/O 8 บิท แบบพูลอัพภายในและนอกจากทำเป็นพอร์ท I/O ที่สามารถขับโหลด TTL พวกตระกูลแอลเอสได้ 4 ตัวแล้วยังใช้งานเป็นพิเศษ สำหรับตระกูล MCS-51 คือ

พอร์ท	ขา	การทำงานตามฟังก์ชั่นพิเศษ
P3.0	10	RXD พอร์ทอนุกรมอินพุท
P3.1	11	TXD พอร์ทอนุกรมเอาท์พุท
P3.2	12	INTO อินเทอร์รัพท์ภายนอกตัวที่ 1
P3.3	13	INT1 อินเทอร์รัพท์ภายนอกตัวที่ 2
P3.4	14	TO สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 0
P3.5	15	T1 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 1
P3.6	16	WR สัญญาณควบคุมการเขียน
P3.7	17	RD สัญญาณควบคุมการอ่าน

การที่จะให้ทำงานตามฟังก์ชั่นข้างบน จะต้องเริ่มโปรแกรมด้วยการส่งค่า "1" ไปแลทช์ไว้ก่อนที่จะทำงานตามฟังก์ชั่นบน

-ขา RST ขา 9 ต้องคงสถานะสูงเป็นเวลาประมาณอย่างน้อยสองวัฏจักรระหว่างที่ออสซิลเลเตอร์ทำงาน ขณะที่ต้องการรีเซททั้งระบบงาน โดยจะต่อรีจิสเตอร์พูลดาวน์ (8.2 กิโลโอห์ม) จากขา RST ลงดิน และเพื่อให้ตัวชิปรีเซทได้โดยอัตโนมัติขณะเปิดไฟ จะใช้คาปาซิเตอร์ (10 ไมโครฟารัด) ต่อคร่อมระหว่างขา RST กับขา V_{cc}

-ขา ALE/PROG ขา 30 เป็นขาแอดเดรสแลทช์อีนาเบิ้ลด้วยการส่งพัลซ์ออกไปใช้สำหรับแลทช์ค่าแอดเดรสไบท์ต่ำจากพอร์ท 0 ในระหว่างการเข้าถึงข้อมูลจากหน่วยความจำภายนอก ALE จะถูกส่งสัญญาณนาฬิกาออกมา ในอัตราความเร็วคงที่ที่ 1/6 ของความถี่ออสซิเลเตอร์ ตลอดเวลาแม้ว่าจะไม่มีการเข้าถึงข้อมูลจากภายนอก ดังนั้น จึงสามารถที่จะใช้สัญญาณจากขานี้เป็นตัวตั้งเวลาภายนอก หรือเป็นความถี่สัญญาณนาฬิกา แต่อย่างไรก็ตามความถี่สัญญาณนี้จะลดความถี่ช้าลงไปเท่าหนึ่งระหว่างการทำงานแบบการเข้าถึงของหน่วยความจำข้อมูล

-ขา PSEN ขา 29 Prograr Storage Enable เป็นสโตรบอ่านข้อมูลโปรแกรมหน่วยความจำภายนอกเมื่อชิปทำงานด้วยโปรแกรมภายนอก ขา PSEN จะไม่มีพัลซ์ส่งออก ถ้าชิปทำงานด้วยโปรแกรมหน่วยความจำภายใน

-ขา EA/V_{PP} ขา 31 มีสถานะสูง ตัวชิพภายในชิปจะทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายใน (โดยที่โปรแกรมจะต้องไม่ยาวกว่า 4 กิโลไบท์) การทำให้ EA มีสถานะต่ำจะเป็น

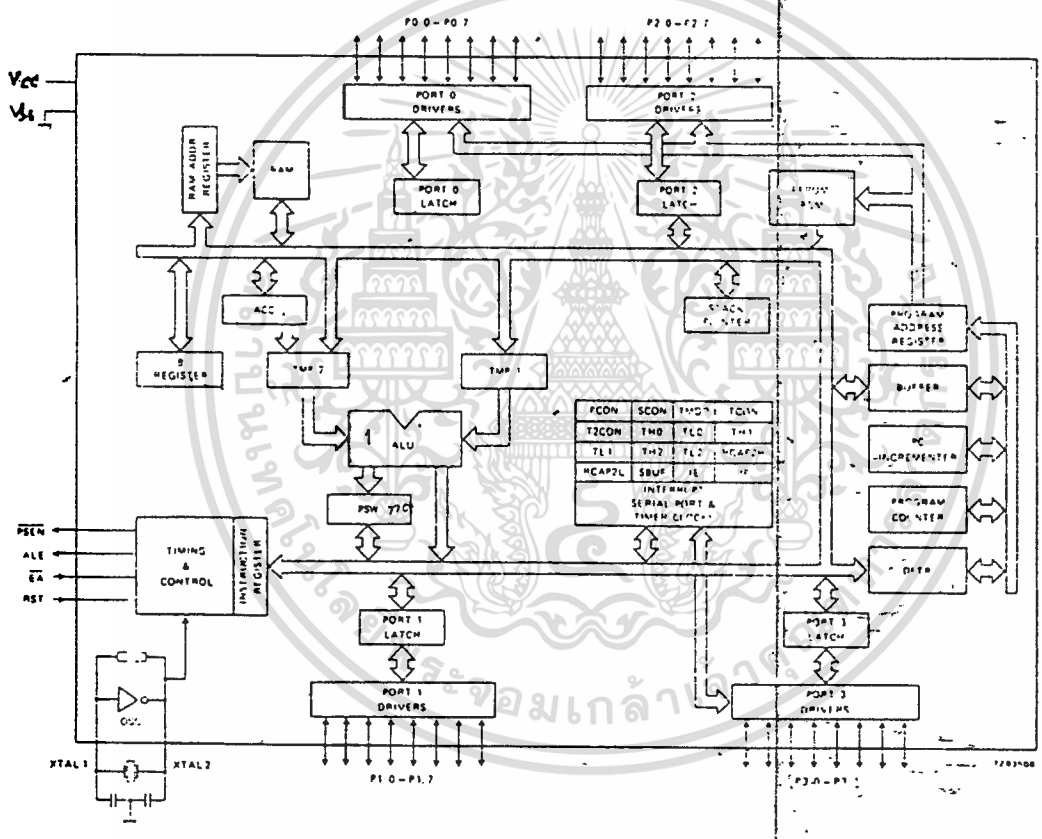
การควบคุมให้ซีพียูทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรแกรมได้ยาวถึง 64 กิโลไบท์ ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี Rom อยู่ภายในก็ตาม

-ขา XTAL 1 ขา 19 ใช้เป็นตัวอินพุทเข้าสู่ตัวออสซิเลเตอร์ขยายแบบ Invert

-ขา XTAL 2 ขา 18 ใช้เป็นตัวเอาท์พุทจากตัวออสซิเลเตอร์ขยายแบบ Invert

รูปที่ 2.3 เป็นบล็อกที่ถูกแบ่งตามลักษณะงานในการจัดการภายในของ MCS-51 โดยชิงเกิ้ลชิปแต่ละตัวของตระกูลนี้จะประกอบด้วยหน่วยศูนย์กลางประมวลผล หน่วยความจำสองชนิดคือแบบ Rom กับโปรแกรม Rom หรือ Eprom, พอร์ทเอาท์พุท อินพุท และโหมดรีจิสเตอร์ สถานะและข้อมูล ส่วนวงจรตรรกในการ Random ที่จำเป็นสำหรับตัวแปรของฟังก์ชั่นการต่อร่วงส่วนต่าง ๆ ที่กล่าวมานี้จะติดต่อกันด้วยบัสข้อมูลขนาด 8 บิท และจะมีบัฟเฟอร์สำหรับการติดต่อข้อมูลกับภายนอกผ่านพอร์ทไอโอ เมื่อต้องการขยายหน่วยความจำหรือพอร์ทไอโอ

2.3 การจัดการทางสถาปัตยกรรม



2.4 หน่วยศูนย์กลางประมวลผลหรือซีพียู

ซีพียูเป็นมันสมองของระบบไมโครคอมพิวเตอร์ การผ่านโปรแกรมและทำงานตามคำสั่งโปรแกรมจะถูกเก็บไว้ในส่วนนี้ โดยการใช้ส่วนคณิตศาสตร์และตรรกศาสตร์ทำงานร่วมกับรีจิสเตอร์ A,B,PSW (Program Status Word) และ SP (Stack Pointer) และรีจิสเตอร์ 16 บิทตัวนับโปรแกรม (PC:Program Counter) และตัวชี้ตำแหน่งข้อมูล (DPTR: Data Pointer) ส่วนคณิตศาสตร์และตรรกศาสตร์ (ALU:Artithmetic Logic Unit) เอแอลยูนี้ทำงานในฟังก์ชั่นทางคณิตศาสตร์และตรรกศาสตร์ด้วยตัวแปรต่าง ๆ ขนาด 8 บิทที่มีลักษณะการทำงานทางคณิตศาสตร์เป็น บวก ลบ คูณ หาร และรวมทั้งทางตรรกศาสตร์เป็น AND OR XOR รวมทั้งการเลื่อนแลวนรอบบิท การเคลียร์ค่าและกลับค่า (Complement) ฯลฯ ALU ยังสามารถที่จะตัดสินใจให้กระโดยไปทำคำสั่งโปรแกรมในส่วนอื่น ๆ ตามข้อกำหนดที่ตั้งขึ้น และยังแบ่งรีจิสเตอร์ชั่วคราวใช้สำหรับให้ข้อมูลเป็นทางผ่านชั่วคราวในการถ่ายเทข้อมูลภายในระบบคำสั่งอื่นที่ถูกสร้างในการใช้ ALU นี้ยังมีความสามารถที่จะให้ค่ารีจิสเตอร์เพิ่มขึ้นครั้งละหนึ่งในลักษณะการบวกด้วยค่าหนึ่ง (Increment) หรือคำนวณเลขที่อยู่ข้อมูลที่จะนำไปเก็บหรือการลดค่าลงครั้งละหนึ่งในลักษณะการลบด้วยค่าหนึ่ง (Decrement) โดยอัตโนมัติ หรือใช้ในการเปรียบเทียบค่าของตัวแปรทั้งสอง

สิ่งที่สำคัญในการทำงานทางสถาปัตยกรรมของ MCS-51 คือสามารถที่ทำได้ที่ขนาดข้อมูล 8 บิท และ 1 บิท การใช้งานในแต่ละบิทในการเซต เคลียร์หรือกลับค่า การเคลื่อนย้าย ทดสอบ และใช้ในการคำนวณทางตรรกขนาด 1 บิท ความสามารถเช่นนี้เหมาะสำหรับการใช้งานควบคุมที่มีการคิดและออกแบบทางตรรกด้วย Boolean ของสัญญาณเข้าและออก ซึ่งโดยปกติทำได้ลำบากสำหรับไมโครโปรเซสเซอร์ทั่ว ๆ ไปด้วย ลักษณะงานเช่นนี้จึงได้ชื่องานอีกอย่างหนึ่งว่า Boolean Processor

2.4.1 แอกคิวมูเลเตอร์ (Accumulator : ACC)

MCS-51 ก็เช่นเดียวกับ MCS-48 ใช้ ACC ขนาด 8 บิท เป็นแอกคิวมูเลเตอร์หลัก โดยคำสั่งส่วนใหญ่จะอ้างถึง ตัวรีจิสเตอร์นี้ ถือเป็นค่าตัวตั้งและรับผลลัพธ์จากคำสั่งทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร เข้ามาเก็บไว้ ตัว ACC ยังสามารถใช้เป็นตัวแหล่งกระทำหรือถูกกระทำในการทำงานทางตรรก และใช้ในการถ่ายเทข้อมูลติดต่อกับอุปกรณ์ภายนอกไอโอและหน่วยความ

จำภายนอก รวมถึงการตรวจสอบตารางข้อมูล

2.4.2 รีจิสเตอร์ B

เป็นรีจิสเตอร์พิเศษที่ใช้งานในคำสั่งของการคูณและหาร ใช้เป็นตัวคูณหรือตัวหาร และเป็นที่เก็บผลลัพธ์ตัวที่สองหลังการคูณและหาร

2.4.3 คำแสดงสถานะโปรแกรม (Program Status Word : PSW)

รีจิสเตอร์ PSW เป็นรีจิสเตอร์ที่แสดงผลที่ได้หลังจากการใช้คำสั่งต่าง ๆ และใช้เป็นตัวเลือกกลุ่มการทำงานของรีจิสเตอร์กลุ่มต่าง ๆ ซึ่งรายละเอียดดังรูปที่ 2.4

(MSB) (LSB)

CY	AC	FD	RS1	RS0	OV	-	P

SYMBOL	POSITION	NAME AND SIGNIFICANCE
CY	PSW.7	CARRY FLAG
AC	PSW.6	ACCURACY CARRY FLAG
F0	PSW.5	FLAG 0
RS1	PSW.4	REGISTER BANK SELECT CONTROL
RS0	PSW.3	0/SET CLAER BY SOFTWAER TO DETERMINE WORKING REGISTER BANK
OV	PSW.2	OVERFLOW FLAG
-	PSW.1	USER DEFINABLE FLAG
P	PSW.0	PARITY FLAG SET/CLAER BY HARDWAER EACH INSTRUCTION CYCLE TO INDICATE

(ต่อ) AN ODD/EVEN NUMBER OF ONE BIT IN THE ACCUMULATOR

NOTE:

THE CONTENTS OF (RS1.RS0) ENABLE THE WORKING REGISTER BANKS AS FOLLOW

(00) BANK 0 (00H - 07H)

(01) BANK 1 (08H - 0FH)

(10) BANK 2 (10H - 17H)

(11) BANK 3 (18H - 1FH)

รูปที่ 2.4 รีจิสเตอร์คำแสดงและสถานะโปรแกรม PSW

2.4.4 ตัวชี้สแตก (Stack Pointer : SP)

MCS-51 จะรวมเอาสแต็กทางฮาร์ดแวร์ที่ใช้ Ram ภายใน สำหรับการเชื่อมต่อระหว่างโปรแกรมหลัก การผ่านพารามิเตอร์ระหว่างงานในแต่ละส่วนโปรแกรม และการใช้สแต็กเก็บตัวแปรข้อมูลชั่วคราว หรือการเก็บสถานะระหว่างการบริการงานอินเทอร์รัพท์ โดยที่ SP จะมีขนาด 8 บิท จะเพิ่มค่าขึ้นโดยอัตโนมัติก่อนที่ข้อมูลจะนำมาเก็บในหน่วยความจำระหว่างการใช้คำสั่ง PUSH และ CALL และจะลดค่าของ SP ลงหลังจากที่ได้ถ่ายเทข้อมูลไปแล้วในคำสั่ง POP หรือ RETURN โดยทฤษฎีทางสถาปัตยกรรม MCS-51 สามารถใช้สแต็กให้มีเนื้อที่ถึง 128 ไบท์ แต่ในทางปฏิบัติสำหรับโปรแกรมทั่วไปจะใช้น้อยกว่านี้ SP จะถูกเริ่มตำแหน่งที่ 07H ดังนั้น สแต็กจะเริ่มบรรจุข้อมูลที่ 08H MCS-51 สามารถเปลี่ยนแปลงค่าใน SP ก็จะเปลี่ยนตำแหน่งสแต็กได้ในที่ใด ๆ ของ Ram ภายใน

2.4.5 ตัวชี้ข้อมูล (Data : Pointer : DPTR)

DPTR รีจิสเตอร์ขนาด 16 บิทที่ประกอบด้วยไบท์สูง (DPH) และไบท์ต่ำ (DPL) ที่สามารถที่จะเลือกแบ่งรีจิสเตอร์ 8 บิทให้ใช้ได้อย่างอิสระหรือจะใช้รวมกันทั้ง 16 บิท ในการ Increment หรือ Decrement เพื่อประโยชน์ใช้เป็นฐานเลขที่อยู่รีจิสเตอร์ในการกระโดดโดยอ้อม การใช้คำสั่งเกี่ยวกับตารางข้อมูลและชี้ตำแหน่งเลขที่อยู่ความจำข้อมูลภายนอก

2.4.6 พอร์ท 0 ถึง 3

รีจิสเตอร์ P0, P1, P2 และ P3 ของกลุ่มรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR) จะเป็นตัวแลทช์ค่าของพอร์ท 0,1,2 และ 3 ตามลำดับ

2.4.7 บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

บัฟเฟอร์ข้อมูลอนุกรมแบ่งเป็นสองรีจิสเตอร์ ตัวหนึ่งเป็นบัฟเฟอร์ส่งและอีกตัวเป็นบัฟเฟอร์รับ เมื่อข้อมูลถ่ายเทเข้า SBUF มันจะถ่ายเข้าบัฟเฟอร์ส่งซึ่งเป็นตัวจัดการส่งข้อมูลอนุกรม วิธีการเคลื่อนย้ายเข้า SBUF ขึ้นอยู่กับการเริ่ม (Initial) การส่ง เมื่อข้อมูลย้ายออกจาก SBUF จะเป็นการรับข้อมูลจากบัฟเฟอร์ตัวรับ

2.4.8 รีจิสเตอร์ควบคุม

กลุ่ม SFR ที่เป็น IP,IE, TMOD, TCON, SCON PCON จะประกอบด้วยบิทที่ใช้ในการควบคุมและแสดงสถานะของการใช้งานในระบบอินเทอร์รัพท์ ตัวจับเวลา/ตัวนับ และพอร์ทอนุกรม

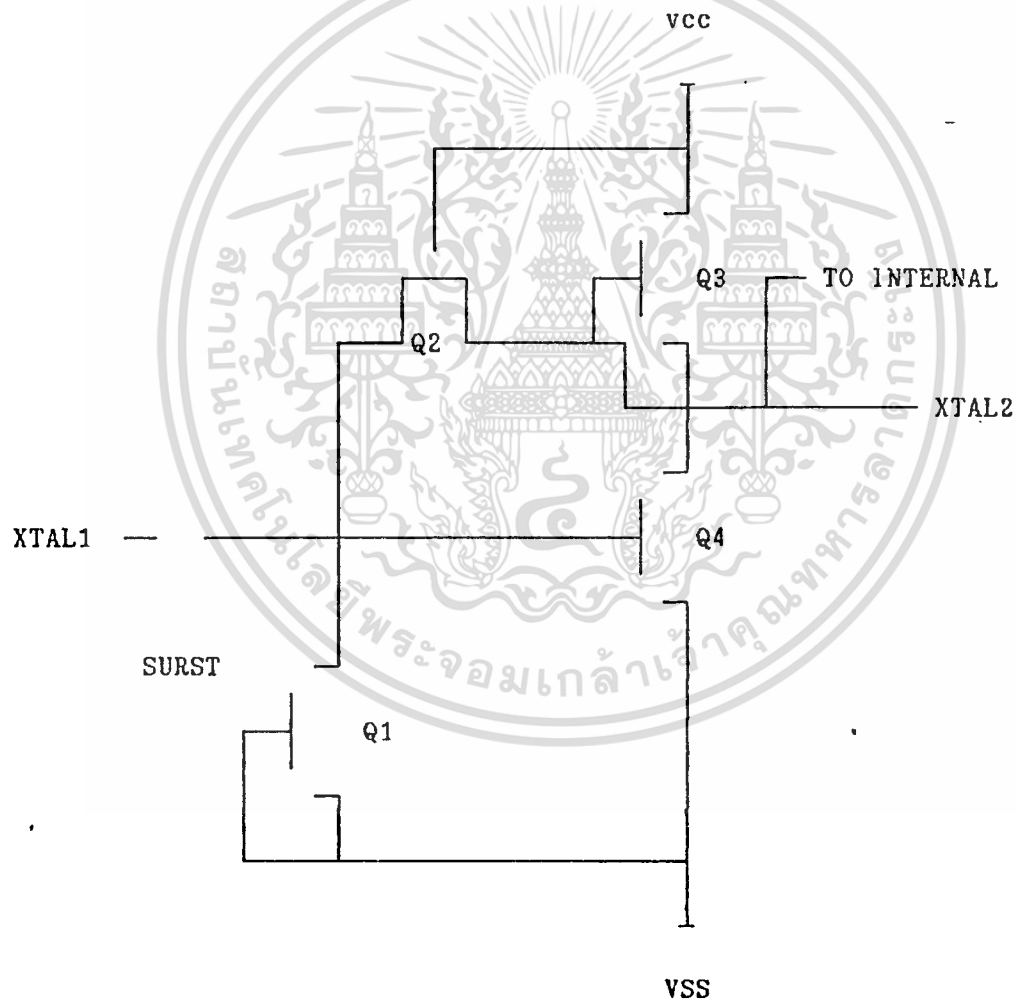
2.5 การจัดการหน่วยความจำ

ตัว MCS-51 จะแยกแอดเดรสสำหรับความจำโปรแกรมและหน่วยความจำข้อมูล โปรแกรมหน่วยความจำขยายได้ถึง 64 กิโลไบท์ ความจำข้อมูลมี 128 ไบท์ และอีก 128 ไบท์ใช้สำหรับรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR) หน่วยความจำข้อมูลภายนอกขยายได้อีก 64 กิโลไบท์และใน 8032 จะมีหน่วยความจำข้อมูลภายในอีก 4 กิโลไบท์

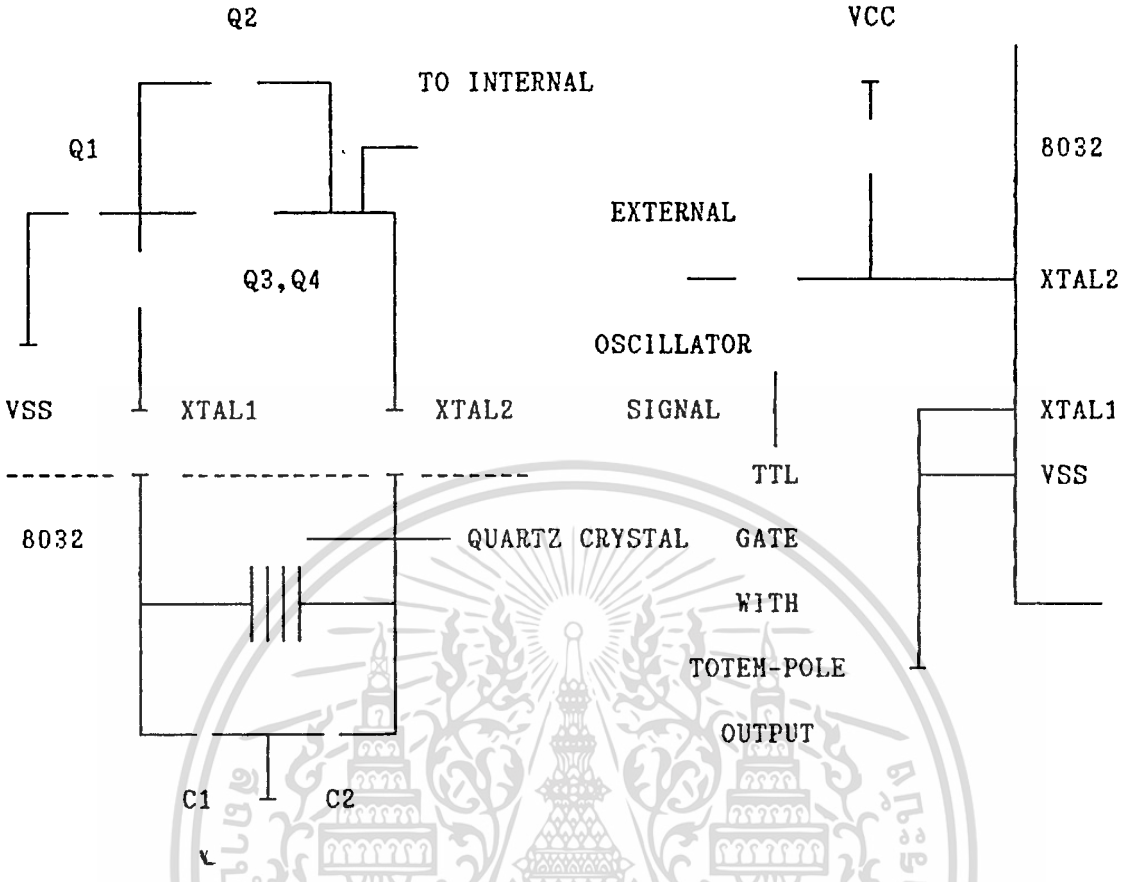
2.6 ออสซิลเลเตอร์และวงจรนาฬิกา

วงจรออสซิลเลเตอร์ที่อยู่ในชิปสำหรับแบบ HMOS จะเป็น Single Linear Inverter ตามรูปที่ 2.5 เพื่อให้ใช้คริสตัลควบคุมเป็นออสซิลเลเตอร์แลรีแอ็กทีฟบวกดังรูป 2.6 ในการ

ใช้งานคริสตัลนี้จะทำงานที่โหมด Fundamental เสมือนเป็น Inductance โดยต่อขนานกับคาปาซิเตอร์ภายนอกกับตัวคริสตัล การกำหนดตัวคริสตัลและค่าคาปาซิเตอร์ C1 และ C2 ในรูป 2.6 ไม่ค่อยวิกฤตนัก อาจจะมีค่าประมาณ 30 pF สำหรับทุกความถี่ของตัวคริสตัลชั้นดี ส่วนการใช้ Ceramic Resonator ค่าคาปาซิเตอร์ที่มาต่อจะมีค่าสูงกว่า โดยมีค่าประมาณ 47 PF การใช้ค่าคาปาซิเตอร์อาจเปลี่ยนแปลงได้ ขึ้นอยู่กับตัว Ceramic Resonator นั้น ๆ การขับตัว HMOS ด้วยสัญญาณนาฬิกาจากภายนอกก็กระทำได้เช่นกัน โดยต่อเข้าที่ขา XTAL2 และต่อลงดินที่ขา XTAL1 ดังรูปที่ 2.7 การใช้พูลอัพตัวต้านทาน ควรจะใช้เพราะระดับสัญญาณที่ XTAL2 ต้องการได้ระดับทีทีแอล



รูปที่ 2.5 วงจรออสซิลเลเตอร์ภายใน MCS-51 แบบ HMOS

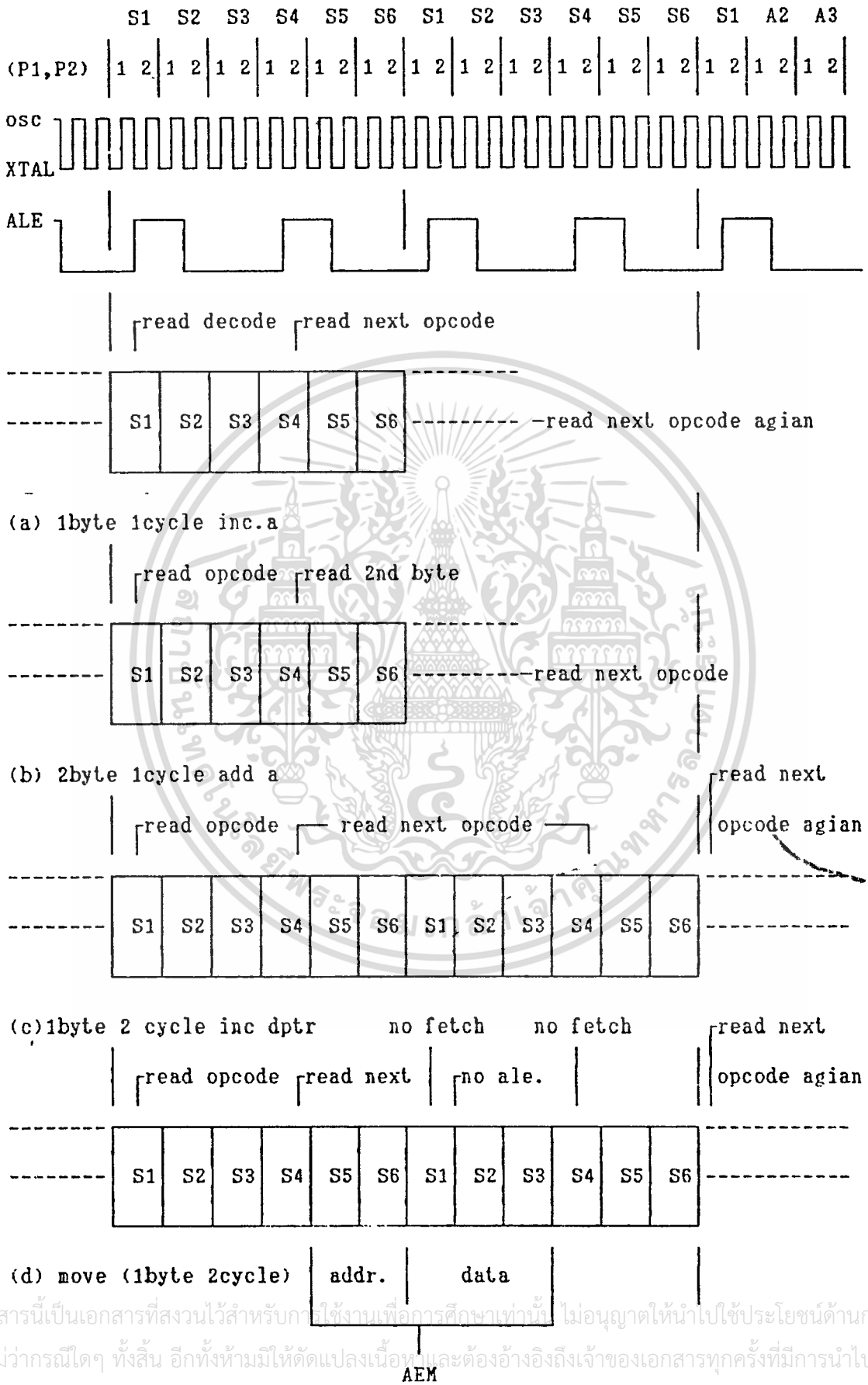


รูปที่ 2.6 การใช้วงจรออสซิลเลเตอร์บน HMOS ชิป

รูปที่ 2.7 การจ่ายสัญญาณนาฬิการภายนอกในการขับ HMOS MCS-51

2.7 ช่วงจังหวะเวลาของ CPU

วัฏจักรแมชชีนประกอบด้วย 6 สถานะหรือเท่ากับ 12 คาบของออสซิลเลเตอร์แต่ละสถานะจะแบ่งเป็นเฟส 1 (P1) ครึ่งหนึ่งเป็นช่วงเฟส 1 แอ็กทีฟ และเฟส 2 (P2) เป็นช่วงเฟส 2 แอ็กทีฟ ดังนั้นในแต่ละวัฏจักรแมชชีนจะประกอบด้วย 12 คาบของออสซิลเลเตอร์เป็นจำนวน S1P1 คือสถานะที่ 1 เฟสที่ 1 ถึง S6P2 คือสถานะที่ 6 เฟส 2 โดยปกติการทำงานแบบคณิตศาสตร์และตรรกศาสตร์จะทำในช่วงเฟส 1 และการถ่ายเทข้อมูลภายในระหว่างรีจิสเตอร์จำทำในช่วงเฟส 2



รูปที่ 2.8 แสดงถึงช่วงจังหวะการเฟทช์ และการทำงานตามลำดับที่อ้างถึงลักษณะภายในและเฟส

จากแผนภูมิในรูปที่ 2.8 แสดงถึงช่วงเวลาการเฟทช์ และจะทำงานอ้างถึงลักษณะภายในและเฟส เนื่องจากสัญญาณนาฬิกาภายในผู้ใช้ไม่สามารถที่จะควบคุมการเข้าถึงภายในได้ โดยปกติ ALE จะแอ็กทีฟ 2 ครั้งในแต่ละวัฏจักรแมชชีน และจะเกิดขึ้นระหว่าง S1P2 ถึง S2P1 ครั้งหนึ่ง และระหว่าง S4P2 ถึง S5P1 อีกครั้งหนึ่ง

การทำงานของแต่ละวัฏจักรคำสั่งจะเริ่มที่ S1P2 เมื่ออ็อปโค้ดเก็บเข้าในรีจิสเตอร์คำสั่งหรืออ่านอ็อปโค้ดเข้ามา ถ้าคำสั่งมีสองไบท์ ไบท์ที่สองจะถูกอ่านในช่วง S4 ภายในวัฏจักรแมชชีนเดียวกันก็เป็น 1 ไบท์คำสั่ง และยังคงเฟทช์ที่ S4 แต่ไบท์ที่ถูกอ่าน (ซึ่งควรจะเป็นไบท์ที่สองของคำสั่งเดียวกัน) จะไม่มีผล และตัวนับโปรแกรม (PC) จะยังไม่เพิ่มค่าไม่ว่ากรณีใด การทำงานจะสมบูรณ์ที่ปลายของ S5P2 ตามรูปที่ 2.8 (A)กับ 2.8 (B) เป็นการแสดงแผนภูมิเวลาสำหรับ 1 ไบท์ใน 1 รอบคำสั่ง กับ 2 ไบท์ใน 1 รอบคำสั่ง

คำสั่ง MCS-51 ส่วนใหญ่จะทำงานในช่วงหนึ่งวัฏจักรยกเว้นคำสั่ง MUL (คูณ) DIV(หาร) ที่ใช้มากกว่าสองวัฏจักรในการที่จะทำงานให้สมบูรณ์ได้จะใช้ถึงสี่วัฏจักร ปกติรหัสสองไบท์จะถูกเฟทช์จากหน่วยความจำโปรแกรมช่วงทุกวัฏจักรแมชชีน ยกเว้นคำสั่งพิเศษคือ MOVX ซึ่งมี 1 ไบท์คำสั่ง แต่จะใช้เวลาสองวัฏจักรในการเข้าถึงหน่วยความจำข้อมูลภายนอก ระหว่างการทำคำสั่ง MOVX การเฟทช์จะถูกสคริปหรือหายไป ขณะที่หน่วยความจำข้อมูลภายนอกจะถูกแอดเดรส และสโตรบหรือกระตุ้นรับเข้าไปในซีพียู รูป 2.8 (C) และ 2.8 (D) เป็นการแสดงแผนภูมิเวลาปกติของคำสั่งประเภท 1 ไบท์แต่ใช้ 2 วัฏจักรแมชชีน

2.8 โครงสร้างพอร์ทและการทำงาน

ทั้งสี่พอร์ทใน MCS-51 เป็นแบบสองทิศทาง แต่ละพอร์ทจะประกอบด้วยแลทช์เป็น P0 ถึง P3 ช่อง SFR จะมีตัวขับเอาท์พุทและบัฟเฟอร์อินพุท ตัวขับเอาท์พุทของพอร์ท 0 และ 2 บัฟเฟอร์อินพุทของพอร์ท 0 จะใช้งานในการเข้าถึงหน่วยความจำภายนอก ในการใช้งานที่เอาท์พุทพอร์ท 0 จะเป็นตัวกำหนดไบท์อันดับต่ำของแอดเดรสหน่วยความจำภายนอก ค่าแอดเดรส และค่าข้อมูลจะถูกมัลติเพล็กซ์ด้วยช่วงจังหวะการเฟทช์และการอ่านหรือเขียนข้อมูล และเอาท์พุทพอร์ท 2 จะเป็นตัวกำหนดส่งไบท์สูงของแอดเดรสในการเข้าถึงหน่วยความจำภายนอก

บางขาของตัวขับเอาท์พุทและบัฟเฟอร์อินพุทของพอร์ท 1 และพอร์ท 3 ทั้งหมดสามารถนำไปใช้งานเป็นหลายฟังก์ชั่น (Multifunction) ได้ดังนี้

ขาพอร์ท	การใช้งานตามฟังก์ชั่น
P3.0	RXD (พอร์ทรับข้อมูลอนุกรม)
P3.1	TXD (พอร์ทส่งข้อมูลอนุกรม)
P3.2	INTO (การใช้อินเทอร์รัพท์ภายนอกตัวที่ 1)
P3.3	INT1 (การใช้อินเทอร์รัพท์ภายนอกตัวที่ 2)
P3.4	TO (Timer/Counter 0 สัญญาณอินพุทภายนอก)
P3.5	I1 (Timer/Counter 1 สัญญาณอินพุทภายนอก)
P3.6	WR (สโตรบการเขียนหน่วยความจำภายนอก)
P3.7	RD (สโตรบการอ่านหน่วยความจำภายนอก)

ตัวขับข้อมูลเอาท์พุทแลทช์ในการที่จะให้ทำงานตามตารางบน จะต้องเริ่มโปรแกรมด้วยการเซตค่า "1" เก็บในแลทช์ก่อน

2.9 การเข้าถึงของหน่วยความจำภายนอก

การเข้าถึงของหน่วยความจำภายนอกมี 2 แบบคือ การเข้าถึงของหน่วยความจำโปรแกรมภายนอกกับของหน่วยความจำข้อมูลภายนอกการเข้าถึงของ หน่วยความจำโปรแกรมภายนอกจะใช้ขาสัญญาณ PSEN (Program Store Enable) แอ็กทีฟต่ำ เป็นสโตรบควบคุมการอ่านและการเข้าถึงของหน่วยความจำข้อมูลภายนอกจะใช้ขา RD หรือ WR แอ็กทีฟต่ำเป็นสัญญาณสโตรบควบคุมหน่วยความจำ

การเฟทช์โปรแกรมภายนอกจะใช้ขาแอดเดรส 16 บิทเสมอ ส่วนการเข้าถึงของหน่วยความจำข้อมูลสามารถที่จะใช้กำหนดเลขที่อยู่ได้ทั้ง 16 บิทแอดเดรส เช่น MOVX @DPTR หรือ 8 บิทแอดเดรสเช่น MOVX @Ri

เมื่อไรที่ใช้ 16 บิทแอดเดรส ไบท์สูงของค่าแอดเดรสจะส่งออกที่พอร์ท 2 และจะคงสถานะค่านั้นตลอดในช่วงวัฏจักรการอ่านและเขียนระหว่างช่วงเวลานี้ตัวแลทช์ของพอร์ท 2 ใน SFR จะไม่ต้องประกอบด้วยค่า "1" และค่าข้อมูลใน SFR จะไม่มีการเซตถ้าช่วงวัฏจักรการใช้หน่วยความจำภายนอกไม่มีการเข้าถึงข้อมูลในวัฏจักรต่อมา ค่าใน SFR ของพอร์ท 2 จะปรากฏค่าเดิมกลับมาใหม่ในวัฏจักรตัวต่อมานี้ OR 70 , DI 6 , OP 31

ถ้าใช้เป็น 8 บิทแอดเดรส ค่าใน SFR ของพอร์ทจะยังคงค่าเดิมที่ขาพอร์ท 2 ตลอดช่วงวัฏจักรการใช้ความจำภายนอกซึ่งลักษณะนี้จะเป็นการใช้งานด้านเพจของหน่วยความจำ

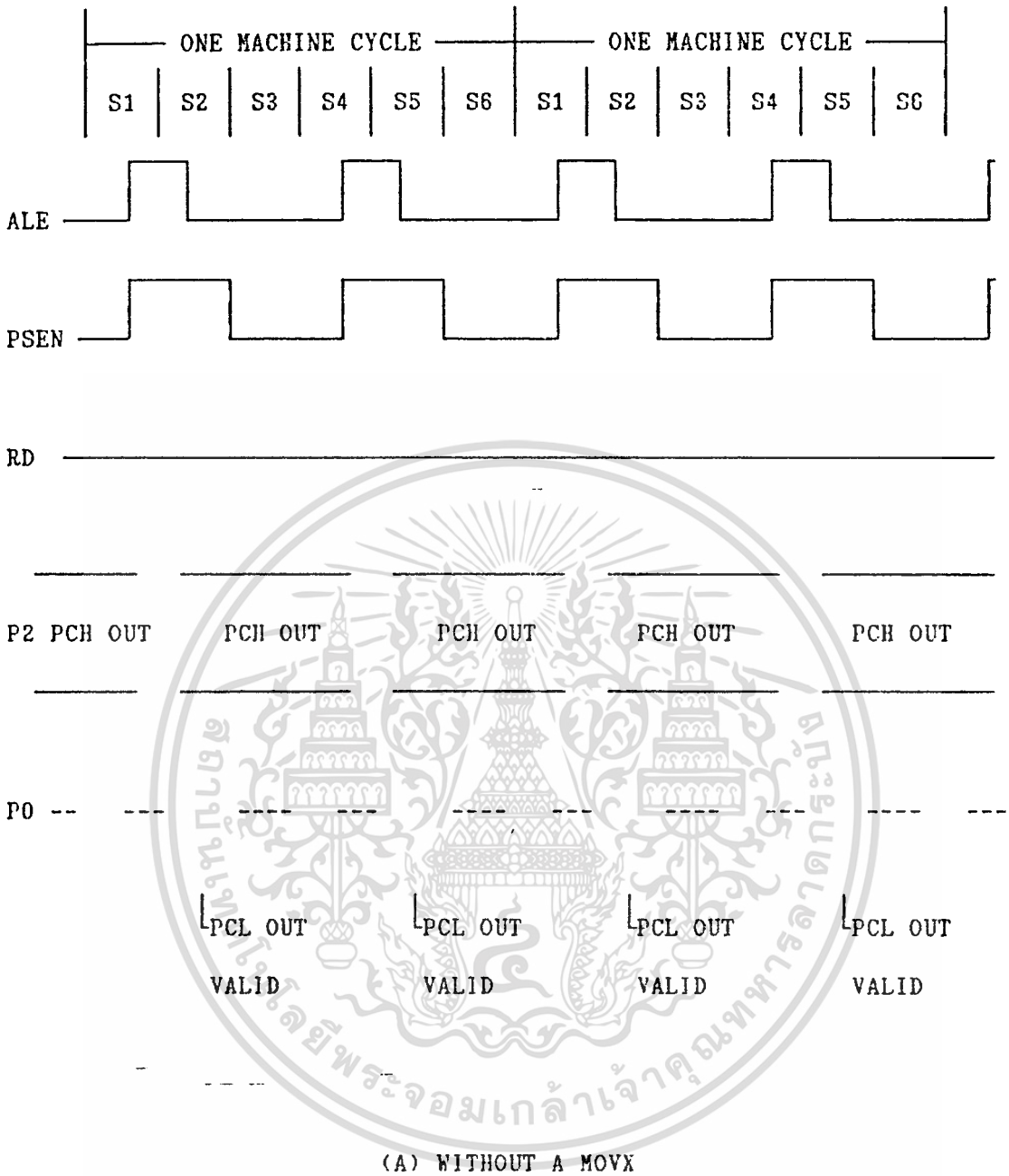
ในกรณีใช้แอดเดรสไบท์ต่ำเป็นช่วงเวลามัลติเพล็กซ์กับข้อมูลของพอร์ท 0 ขาสัญญาณแอดเดรส/ข้อมูล จะขับ FET ทั้ง 2 ตัวในพอร์ท 0 เป็นบัฟเฟอร์ส่งข้อมูลออก ดังนั้นในการใช้งานพอร์ท 0 จะไม่การรับกระแสเข้าจึงไม่จำเป็นต้องพูลอัพจากภายนอก สัญญาณ ALE (Address Latch Enable) ก็จะใช้เป็นขาควบคุมรับไบท์แอดเดรสเก็บไว้ภายนอก ซึ่งค่าแอดเดรสจะคงที่ที่ช่วงขอบขาลงของ ALE ดังนั้นในวัฏจักรการเขียนข้อมูลจะถูกเขียนออกไปที่พอร์ท 0 ก่อนที่ WR จะแอ็กทีฟต่ำ ส่วนวัฏจักรการอ่านข้อมูลจะรับเข้าที่พอร์ม 0 ก่อนสโตรบการอ่านจะปรากฏเล็กน้อย และระหว่างการเข้าถึงของหน่วยความจำภายนอกตัวซีพียูจะส่งค่า OFFH มาเก็บไว้ที่พอร์ท 0 ของ SFR

การใช้หน่วยความจำโปรแกรมภายนอก จะขึ้นอยู่กับสองกรณีคือ

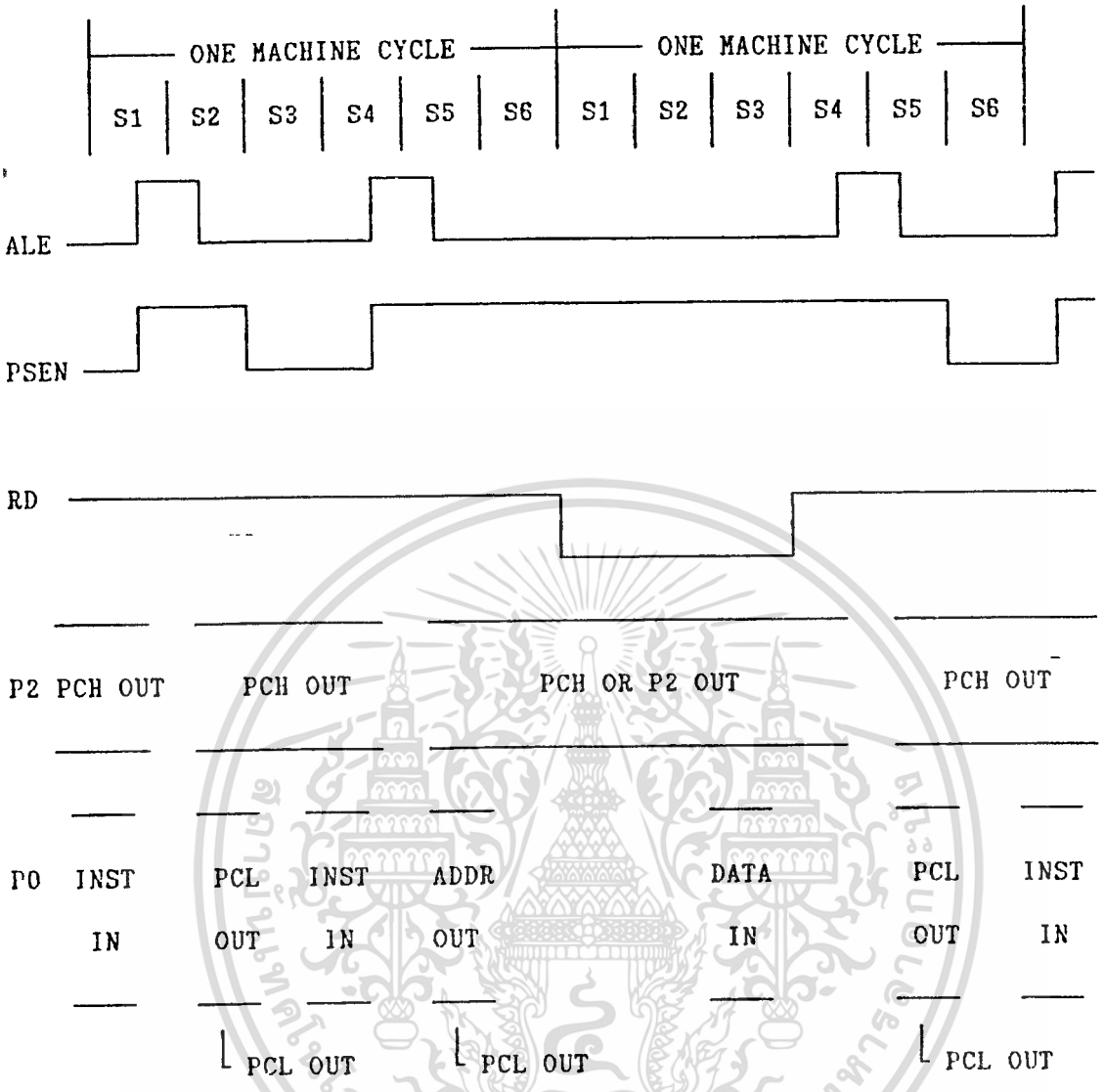
1. เมื่อไรก็ตามที่ EA แอ็กทีฟ หรือ
2. เมื่อไรก็ตามที่ตัวนับโปรแกรม PC ประกอบด้วยตัวเลขที่มีค่า OFFH

ในรุ่นที่ไม่มี Rom ในตัว ให้ใช้ค่าแอ็กทีฟต่ำป้อนที่ขา EA เพื่อกำหนดเฟทช์โปรแกรมภายนอกที่มีค่ากว่า 4 กิโลไบท์ได้

เมื่อซีพียูทำงานด้วยโปรแกรมหน่วยความจำภายนอกทั้ง 8 บิท ค่าพอร์ท 2 จะส่งค่าแอดเดรสออกมาด้วย ทำให้ไม่สามารถจะใช้งานเป็นไอโอได้ในระหว่างการเฟทช์โปรแกรมภายนอกและระหว่างการเข้าถึงของข้อมูลภายนอกพอร์ท 2 จะส่งทั้ง DPH หรือ SFR ขึ้นอยู่กับการใช้คำสั่งว่าใช้แบบให้คำสั่งส่งเอาท์พุทออกที่ DPH ก็จะใช้คำสั่ง MOVX @DPTR หรือใช้แบบให้ข้อมูลส่งข้อมูลออกที่พอร์ท P2 ของ SFR ก็จะใช้คำสั่ง MOVX @Ri



(A) WITHOUT A MOVX



รูปที่ 2.9 จังหวะการเข้าถึงหน่วยความจำภายนอก

2.9.1 สัญญาณ PSEN

ใช้เป็นการควบคุมเฟทช์การอ่านโปรแกรมภายนอก PSEN จะไม่แอ็กทีฟถ้ามีการเฟทช์โปรแกรมภายใน เมื่อซีพียูเข้าถึงการใช้โปรแกรมภายนอก PSEN จะแอ็กทีฟ 2 ครั้ง ในแต่ละช่วงวัฏจักรการเฟทช์ ยกเว้นคำสั่ง MOVX ช่วงเวลาของการที่ PSEN เกิดแอ็กทีฟจะไม่เหมือนกับช่วง RD แอ็กทีฟช่วงวัฏจักร การอ่านที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอ็กทีฟและแอ็กทีฟซ้ำลูกที่สองกับสัญญาณการควบคุม RD ที่เกิดพัลส์ต่ำประกอบเข้าด้วยกันซึ่งจะใช้เวลา 12 คาบสัญญาณนาฬิกา ส่วนช่วงเวลาของ PSEN ที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอ็กทีฟและแอ็กทีฟซ้ำลูกที่สองกับสัญญาณ

ควบคุม PSEN ประกอบเข้าด้วยกัน ซึ่งจะใช้เวลา 6 คาบสัญญาณนาฬิกา ลักษณะทำงานตามลำดับของวัฏจักรการอ่านทั้ง 2 แสดงในรูปที่ 2.9

2.9.2 สัญญาณ ALE

ฟังก์ชั่นหลักของ ALE คือการใช้งานด้านให้จังหวะที่แน่นอนในการแลทช์เอาไบท์ต่ำของแอดเดรสจาก PO ไปเก็บไว้ที่ภายนอก เพื่อใช้ในการถอดระหัสแอดเดรสโปรแกรมภายนอกโดยจะใช้ SLE ทำงานแอ็กทีฟ 2 ครั้ง ในทุก ๆ วัฏจักรแมชชีนสัญญาณนี้จะเกิดขึ้นตลอดแม้ว่าจะไม่ได้เฟทช์จากภายนอก ช่วงเวลาเดียวเท่านั้นที่ ALE ไม่เกิดพัลส์คือระหว่างการเข้าถึงหน่วยความจำภายนอกตามรูปที่ 2.9 (B) จะเห็นว่าพัลส์แรกของ ALE ในวัฏจักรที่สองของคำสั่ง MOVX ขาดหายไป หรือมีพัลส์เพียงพัลส์เดียวในหนึ่งคำสั่ง ลักษณะพัลส์ที่เกิดขึ้นคงที่ในอัตรา 1/6ของสัญญาณความถี่ออสซิลเลเตอร์และสามารถนำมาใช้เป็นสัญญาณนาฬิกาภายนอก หรือกำหนดเวลาได้

2.10 ตัวจับเวลา/ตัวนับ (Timer/Counter)

MCS-51 มีตัวจับเวลา/ตัวนับ 2 ตัวคือ Timer/Counter 0 และ Timer/Counter 1 ขณะที่แต่ละตัวจับเวลา/ตัวนับ (Timer/Counter) สามารถที่จะกำหนดให้ทำงานได้เป็นตัวจับเวลาหรือตัวนับ

2.10.1 ตัว TIMER / COUNTER 0 และ TIMER / COUNTER 1

แต่ละตัวจะถูกกำหนดให้ทำงานเป็นตัวจับเวลาหรือตัวนับ ได้ด้วยการเซตหรือเคลียร์ที่ตัวควบคุมในรีจิสเตอร์ TMOD ในกลุ่ม SFR

ในฟังก์ชั่นตัวจับเวลา ตัวรีจิสเตอร์จะเพิ่มค่าทุก ๆ วัฏจักรแมชชีนดังนั้นตัวเลขในรีจิสเตอร์จะเป็นจำนวนของวัฏจักรแมชชีน เนื่องจากแต่ละวัฏจักรแมชชีนประกอบด้วย 12 คาบของออสซิลเลเตอร์ อัตราการนับแต่ละครั้งจะกินเวลาเป็น 1/12 ของความถี่ออสซิลเลเตอร์

ในฟังก์ชั่นตัวนับ รีจิสเตอร์จะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก "1" เป็น "0" ที่เข้ามาที่ ขา T0 หรือ T1 ในฟังก์ชั่นที่สัญญาณภายนอกที่เข้ามาจะถูกรับแซมปลิ้ง (Sampling) ระหว่างช่วง S5P2 ของทุกวัฏจักรแมชชีน โดยถ้าแซมปลิ้งสัญญาณเข้าเป็นระดับสูงในวัฏจักรหนึ่ง

ดังนั้น ถ้าในวัฏจักรตัวต่อมาของสัญญาณเข้าเป็นระดับต่ำ รีจิสเตอร์จะนับเพิ่มหนึ่งค่าโดยที่ค่าใหม่ของตัวนับจะปรากฏที่รีจิสเตอร์ช่วง S3P1 ของวัฏจักร ซึ่งค่าหนึ่งที่รับเข้าไปจะใช้ช่วง 2 วัฏจักรแมชชีน (เท่ากับ 24 คาบ) ในการรับค่าช่วงการเปลี่ยน 1 เป็น 0 ดังนั้น ค่าสูงสุดในการนับจะมีอัตรา 1/24 ของความถี่ออสซิลเลเตอร์และสัญญาณอินพุทที่นับนั้นจะไม่มีช่วงระยะห่างที่แน่นอนของ Duty Cycle แต่จะถูกนับเมื่อระดับแรงดันที่ถูกแซมปลิ้งในแต่ละครั้งจะต้องมีช่วงคงที่อย่างน้อย 1 วัฏจักรแมชชีนก่อนที่จะเปลี่ยนค่าระดับแรงดันใหม่

ในการเลือกทำงานระหว่างตัวนับเวลากับตัวจับเวลา จะเลือกได้ 4 โหมดคือโหมด 0,1 และ 2 เลือกได้ทั้งสองตัวของ Timer/Counter ส่วนโหมด 3 จะทำงานแตกต่างออกไป

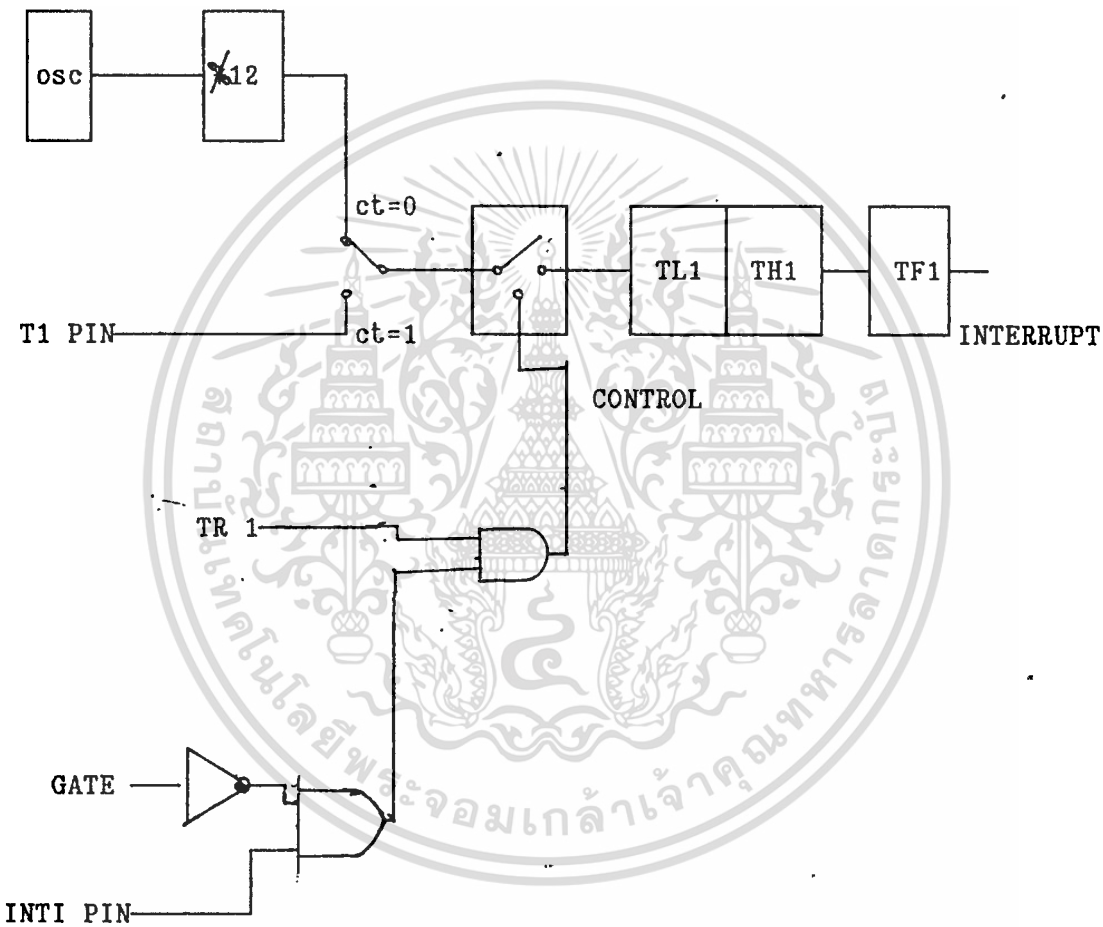
<u>โหมด 0</u>

การใช้ตัวจับเวลา/ตัวนับ 0 หรือ 1 ให้อยู่ในโหมด 0 จะทำงานคล้ายกับของ MCS-48 โดยตัวจับเวลาของ MCS-48 มีขนาด 8 บิท มีตัว Prescaler เป็นตัวหาร 12 รูปที่ 2.10 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวนับ

ในโหมดนี้ รีจิสเตอร์ตัวจับเวลาถูกกำหนดให้มี 13 บิท ด้วยการนับขึ้นเมื่อเป็น "1" หมดทุกบิทจะกลับมาที่ "0" ทุกบิทใหม่ เมื่อกลับเป็น "0" ทุกบิทจะเป็นการเกิด Overflow ไปทดให้แฟลกอินเทอร์รัพท์ TF1 ปรับเป็น "1" การควบคุมให้เริ่มนับตัวอินพุทจะควบคุมด้วยการอีนาเบิ้ล TR1=1; GATE=0 และขา INT1=1 การปรับ GATE=1 เป็นการตั้งตัวนับให้ถูกควบคุมด้วยสัญญาณจากภายนอกเข้าขา INT1 TR1 จะเป็นบิทควบคุมให้รีจิสเตอร์ TCON ของ SFR

รีจิสเตอร์ตัวนับจะมี 13 บิท ประกอบด้วย TH1 8 บิท และ TL1 อีก 5 บิท อันดับต่ำ ส่วนอีก 3 บิทที่เหลือในอันดับสูงของ TL1 จะไม่ใช้ การปรับแฟลก TR1 ให้ทำงานจะไม่เคลียร์ค่าในรีจิสเตอร์

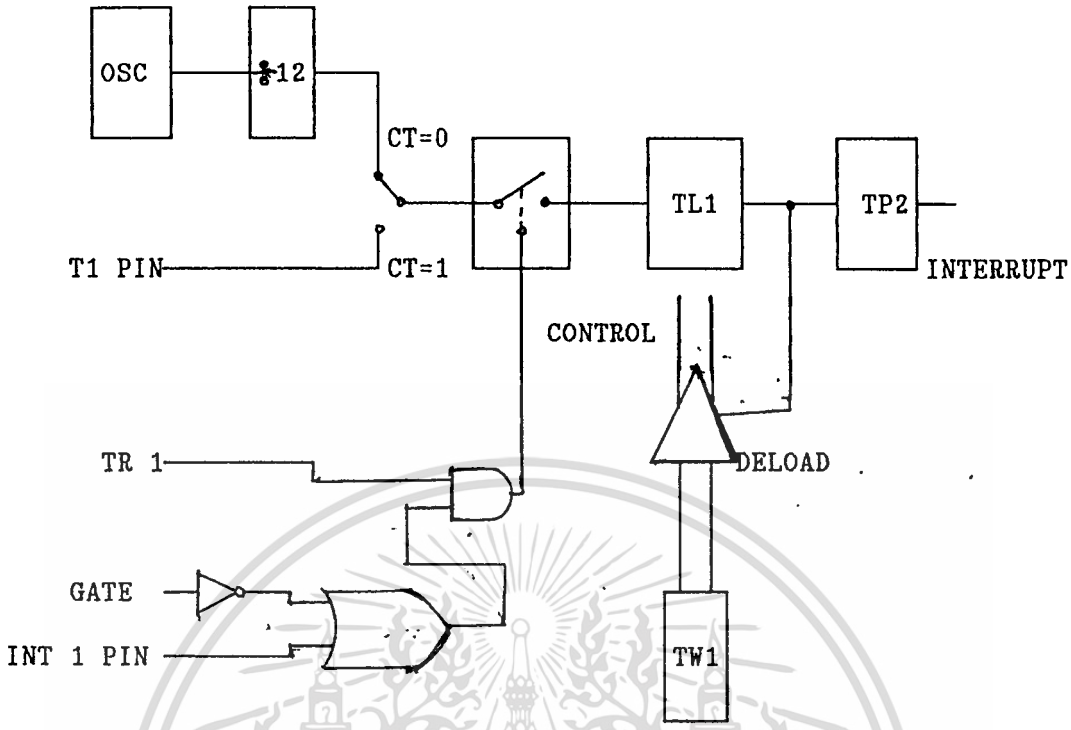
การทำงานในโหมด 0 ในตัวจับเวลา/ตัวนับ จะทำงานเหมือนกับตัวจับเวลา/ตัวนับ 1 โดยใช้ TRO และ INTO รวมกันควบคุมแทนสัญญาณต่าง ๆ ในรูปที่ 2.10 มีความแตกต่างในการควบคุมคือบิทของ GATE ของทั้งสอง ตัวหนึ่งจะแทนตัวจับเวลา/ตัวนับ (TMOD.7) และอีกตัวจะแทนตัวจับเวลา/ตัวนับ (TMOD.3)



รูปที่ 2.10 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวนับ 1 ขนาด 13 บิท

โหมด 1

โหมด 1 ทำงานเหมือนกับโหมด 0 ต่างกันแต่เฉพาะการใช้รีจิสเตอร์ ตัวจับเวลา/ตัวนับจะทำงานด้วยขนาด 16 บิท



รูปที่ 2.11 ตัวจับเวลา/ตัวนับ 1 ทำงานในโหมด 2 แบบโหลดใหม่ 8 บิท

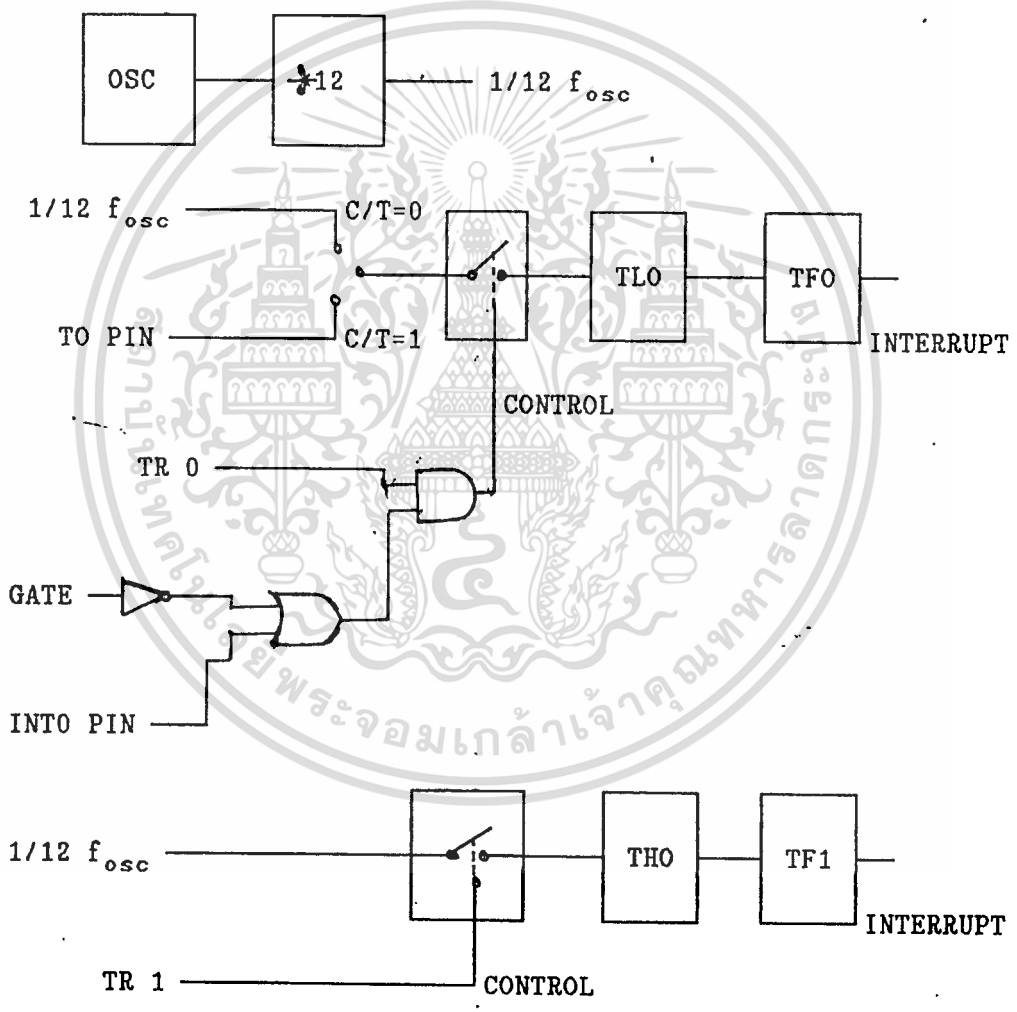
โหมด 2

โหมด 2 มีการทำงานโดยการกำหนดให้ตัวนับ 8 บิทของ TL1 และจะโหมดใหม่โดยอัตโนมัติทุกครั้งเมื่อมีการ Overflow จาก TL1 ดังรูปที่ 2.11 ไม่เพียงแต่ TF1 จะปรับเป็น "1" แต่ TL1 จะถูกโหมดโดยอัตโนมัติจากค่าที่ตั้งไว้ใน TH1 ซึ่งค่าใน TH1 สามารถจะตั้งค่าได้ด้วยซอฟต์แวร์ และบรรจุเข้าไปใหม่ที่ TL1 ทุกครั้งที่เกิด Overflow TH0 และ TF0 จะเป็นตัวร่วมการทำงานในโหมดนี้

โหมด 3

ใช้ตัวจับเวลา/ตัวนับ 1 ในโหมด 3 มีการทำงานเป็นตัวนับ มีผลเช่นเดียวกับการตั้ง TR1=0 และใช้ตัวจับเวลา/ตัวนับ 0 ในโหมด 3 จัดการให้ TL0 และ TH0 เป็นตัวนับสองตัวแรกที่แยกออกจากกัน วงจรตรรกควบคุมสำหรับโหมด 3 ที่ใช้ตัวนับเวลา/ตัวจับเวลา 0 แสดงใน

รูปที่ 2.12 TL0 ใช้ตัวจับเวลา/ตัวนับ 0 เป็นบิทควบคุมของ C/T, GATE, TR0, INT0 และ TF0 และ TH0 จะถูกล็อกให้ทำงานในฟังก์ชั่นตัวจับเวลา (เป็นตัวนับวัฏจักรแมชชีนได้) และจะใช้งานที่ขา TR1 และ TF1 ของตัวจับเวลา 1 เป็นตัวควบคุม ดังนั้นจึงใช้ TH0 เป็นตัวจับเวลาถูกควบคุมอินเทอร์รัพท์ด้วยตัวจับเวลา 1



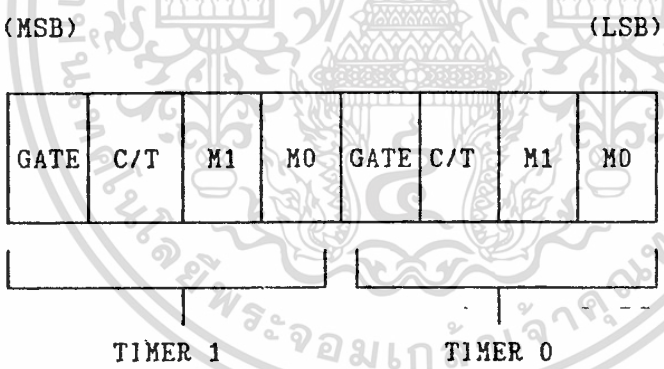
รูปที่ 2.12 ใช้ตัวจับเวลา/ตัวนับ 1 ในโหมด 3 เป็นกลุ่มตัวนับขนาด 8 บิทสองตัว

โดยปกติตัวจับเวลา/ตัวนับ 0 ไม่ใช้ในโหมด 3 ถ้าตัวจับเวลา/ตัวนับ 1 ยังไม่พร้อมที่จะ

ใช้เป็นตัวกำเนิดอัตราบ๊อค (Baud Rate) สำหรับพอร์ทอนุกรม โหมด 3 จะถูกกำหนดแบ่งให้ใช้งานเป็นพิเศษสำหรับความต้องการใช้ตัวจับเวลา/ตัวนับสองตัวเป็นอิสระต่อกัน และจะใช้งานสำหรับการทำงานพอร์ทอนุกรม โดยใช้ตัวจับเวลา 1 เป็นตัวกำเนิดอัตราบ๊อดในโหมด 2 และใช้ตัวจับเวลา 0 ในโหมด 3

2.10.2 Timer/Counter Controlและรีจิสเตอร์ Status

การกำหนดโหมดการทำงานและควบคุมฟังก์ชั่นต่าง ๆ ของตัวจับเวลา/ตัวนับ จะควบคุมได้ที่ SFR (Special Function Register), TMOD และ TCON ด้วยซอฟต์แวร์ โดยที่เมื่อมีคำสั่งเปลี่ยนบิทต่าง ๆ ใน TMOD, TCON ค่าที่ถูกเปลี่ยนก็จะถูกแลทช์เข้าไปที่ SFR และเกิดมีผลตามคำสั่งควบคุมที่ช่วง S1P1 ของวัฏจักรตัวแรกของคำสั่งต่อมา ค่ารีจิสเตอร์ต่าง ๆ ที่ใช้มีแสดงดังรูปที่ 2.13, 2.14 ตามลำดับ โดยทุกบิทของรีจิสเตอร์เหล่านี้จะถูกเคลียร์ด้วยการรีเซท



GATE. GETING CONTROL WHEN SET
TIMER/COUNTER "X" IS
ENABLE ONLY

C/T. TIMER OR COUNTER SELECTOR CLAERED FOR TIMER OPERATION SET FOR
COUNTER OPERATION

(ต่อ)

M1	M0	OPERATING MODE
0	0	8 BIT TIMER/COUNTER "THx" WITH "TLx" AS 5 BIT PRESCATER
0	1	16 BIT TIMER/COUNTER "THx" WITH "TLx" ARE CASCADED. THERE IS NO PRESCATER.
1	0	8 BIT AUTO RELOAD TIMER/COUNTER "THx" BOLD A VALUE WIHICH IS TO BE RELOADED INTO "TLx"
1	1	(TIMER 0) TL0 IS AN 8 BIT TIMER/COUNTER CONTROL BY THE STANDARD TIMER 0 CONTROL BIT TH0 IS AN 8 BIT TIMER ONLY CONTROLLED BY TIMER 1 CONTROL BIT.
1	1	(TIMER 1) TIMER/COUNTER IS STOPPED

รูปที่ 2.13 TMOD : Timer/Counter Mode Control Register

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

SYMBOL	POSITION	NAME AND SIGNIFICANCE
TF1	TCON 7	TIMER 1 OVERFLAG SET BY HARDWAER ON TIMER COUNTER OVERFLOW CLAERED BY HARDWAER WHEN PROCESSORVECTOR TO INTERRUPT ROUTIME
TR1	TCON 6	TIMER 1 RUN CONTROL BIT SET/CLAER BY SOFT-WAER TO TURN TIMER/COUNTER ON/OFF
TF0	TCON 5	TIMER 0 OVER FLOW FLAG SET BY HARDWAER ON TIMER/COUNTER OVERFLOW CLAER BY HARDWAER WHEN PROCESSOR VECTOR TO INTERRUPT ROUNTIME
TR0	TCON 4	TIMER 0 RUN CONTROL BIT SET/CLAER BY SOFTWAER TO TURN TIMER/COUNTER ON/OFF
IE1	TCON 3	INTERRUPT 1 EDGE FLAG. SET BY HARDWAER WHEN EXTERNAL INTERRUPT EDGE DETECTED CLAER WHEN INTERRUPT PROCESSOR
IT1	TCON 2	INTERRUPT 1 TYPE CONTROL BIT SET/CLAER BY SOFTWAER TO SPECCITY FILING EDGE/LOW LEVEL EXTERNAL INTERRUPT
IE0	TCON 1	INTERRUPT EDGE 0 FLAG SET BY HARDWAER
IT0	TCON 0	INTERRUPT 1 TYPE CONTROL BIT SET/CLAER BY SOFTWAER TO SPECITY FAILING

รูปที่ 2.14 TCON : Timer/Counter Control Register

2.11 การอินเตอร์รัพท์

โดยทั่วไปความสามารถในการควบคุมการอินเทอร์รัพท์เป็นการทำงานชนิดหนึ่งของซีพียูที่จะต้องศึกษาถึงความสามารถและเทคนิคการทำงาน การอินเทอร์รัพท์ของชิงเกิ้ลชิปมีความสัมพันธ์กันอย่างใกล้ชิดระหว่างอุปกรณ์ต่อพ่วงกับระบบการทำงานของอุปกรณ์ต่อพ่วงเหล่านี้มีระบบฮาร์ดแวร์ที่ช่วยให้การส่งสัญญาณ Real Time กับมอนิเตอร์ได้อย่างต่อเนื่อง โดยปราศจากการรบกวนต่อสัญญาณการทำงานของซีพียู ตัวอย่างเช่นขณะที่มีการรับสัญญาณอนุกรมจากซีอาร์ทีตัวหนึ่ง ก็จะมีการส่งสัญญาณไปยังอุปกรณ์ตัวอื่น และตัวจับเวลา/ตัวนับ ก็จะนับพัลส์การเปลี่ยนแปลงที่เข้ามาอย่างรวดเร็วไปพร้อมกันได้ด้วย ในขณะที่ตัวจับเวลาตัวนับ อีกตัวก็กำลังวัดความกว้างของพัลส์ที่เข้ามา

ซีพียูตัวนี้จะรู้ได้อย่างไรว่า เมื่อใดจะมีการรับ และ ส่งสัญญาณอนุกรมซีอาร์ทีหรือให้ตัวจับเวลา/ตัวนับ มีการนับจำนวนและวัดความกว้างของพัลส์ว่าจะสิ้นสุดลงเมื่อไร

ตัวโปรแกรม MCS-51 สามารถที่จะเลือกการโปรแกรมได้ 3 วิธีด้วยกันคือพิจารณาการโปรแกรมตัวรีจิสเตอร์ TCON และ SCON ที่ประกอบด้วยสถานะบิทที่ถูกเซตทางฮาร์แวร์ เมื่อตัวจับเวลาตัวหนึ่งเกิด Overflow หรือเมื่อการรับส่งข้อมูลที่พอร์ทอนุกรมสิ้นสุดลง

เทคนิคการโปรแกรมวิธีแรกก็โดยการอ่านสถานะของรีจิสเตอร์ควบคุมเข้าไปยังแอกคิวมูเลเตอร์แล้วทดสอบสถานะบิทตามลักษณะการทำงานนั้น ๆ แล้วทำงานกระโดยไปยังโปรแกรมย่อยตามผลที่เกิดขึ้นชนิดนั้น ๆ ลักษณะการทดสอบร่วมกันครั้งละหลายลักษณะงานเช่นนี้เปรียบเทียบเสมือนตัวโปรแกรมใช้ระบบไมโครโปรเซสเซอร์หลายตัวควบคุมชิปอุปกรณ์ต่อพ่วงต่าง ๆ ซึ่งผู้โปรแกรมจะต้องทำความเข้าใจอย่างฉับชิ้งถึงระบบและจังหวะที่จะเกิดในแต่ละงาน และการทดสอบแต่ละครั้งจะใช้คำสั่งไม่น้อยกว่า 3 คำสั่ง

วิธีที่สอง 8032AH สามารถที่จะทำงานด้วยการกระโดดไปตามสถานะของการควบคุม หรือสถานะบิทของรีจิสเตอร์ควบคุมงาน หรือการรับสัญญาณที่เข้ามาตามชาอินพุทแต่ละบิทด้วยการใช้คำสั่งเพียงคำสั่งเดียว ดังนั้นลักษณะงานสี่อย่างก็สามารถที่ใช้คำสั่งตรวจสอบได้ภายในสี่คำสั่ง ซึ่งจะใช้เวลาประมาณภายใน 8 ไมโครวินาที

แต่วิธีทั้งสองที่กล่าวมาแล้วจะต้องใช้ซีพียูมาทำการตรวจสอบบิทสถานะต่าง ๆ อยู่ตลอดเวลาลองเปรียบตัวซีพียูเหมือนกับตัวผู้จัดการของบริษัท ซึ่งจะบริหารงานในบริษัทให้ก้าวหน้าได้อย่างดีนั้น จะต้องมีเวลาทำงานให้กับหน้าที่หลักของตัวเองได้ อย่างต่อเนื่องและใช้เวลาเพียงบางส่วนสำหรับพนักงานที่จะเข้ามาขัดจังหวะ เพื่อขอปรึกษาแก้ไขปัญหาเพียงบางเวลาที่จำเป็นเท่านั้นเช่นเดียวกันแทนที่จะใช้ซีพียูทำงาน ในลักษณะที่ออกไปตรวจสอบสถานะการทำงานของ

อุปกรณ์พ่วงรอบข้างต่าง ๆ ที่ต้องการจะใช้บริการก็จะใช้อุปกรณ์ต่อพ่วงเป็นฝ่ายร้องขอการบริการเข้ามาที่ซีพียูแทน ซึ่งเมื่อซีพียูถูกร้องขอบริการเข้ามาก็ปล่อยงานเดิมและเข้าสู่การบริการที่อุปกรณ์ต่อพ่วงชนิดนั้น ๆได้ร้องขอเข้ามาชั่วระยะเวลาหนึ่งแล้วจึงกลับเข้าทำงานหลักต่อไปเมื่อสิ้นสุดงานบริการนั้นแล้วทำให้รู้สึกว่าตัวไมโครโปรเซสเซอร์ทำงานพร้อมกันได้หลายงานในเวลาเดียวกัน

8032 มีแหล่งการอินเทอร์รัพท์ 5 แหล่ง โดยแต่ละแหล่งสามารถจะโปรแกรมให้ระดับหนึ่งในสองของระดับไพรโอริตี้ (Priority) แหล่งการอินเทอร์รัพท์จะมาจากภายนอก 2 แหล่งที่เข้ามาที่ขา INTO และ INT1 และแต่ละแหล่งจากตัวจับเวลา/ตัวนับ สามตัวในการเกิดแฟลก Overflow

(MSB) (LSB)

EA	–	–	ES	ET1	EX1	ET1	EX0

SYMBOL	POSITION	NAME AND SIGNIFICANCE
EA	IE.7	ENABLE ALL CONTROL BIT CLAERED BY SOFTWAER TO DISABLE ALL INTERRUPT INDEPENDENT OF THE STATE OF IE.4-IE.0
–	IE.6	(RESERVED)
–	IE.5	
ES	IE.4	ENABLE SERIAL PORT CONTROL BIT SET/CLAERED BY SOFTWAER TO ENABLE DISABLE INTERRUPT FORM TI OR RI FLAG
ET1	IE.3	ENABLE TIMER 1 CONTROL BIT SET/CLAER BY SOFTWAER TO ENABLE DISABLE INTERRUPT FROM TIMER/COUNTER 1
EX1	IE.2	ENABLE EXTERNAL INTERRUPT 1 CONTROL BIT SET/CLAER BY SOFTWAER TO ENABLE/DISABLE INTERRUPT FROM INT1.

(ต่อ)

ET0	IE.1	ENABLE TIMER 0 CONTROL BIT SET/CLAER BY SOFTWAER TO ENABLE/DISABLE INTERRUPT FROM TIMER/COUNTER 0
EX0	IE.0	ENABLE EXTERNAL INTERRUPT 0 CONTROL BIT. SET/CLAER BY SOFTWAER TO ENABLE/DISABLE INTERRUPT FROM INTO.

C) E-INTERRUPT ENABLE REGISTER

(MSB) (LSB)

–	–	–	PS	PT1	PX1	PT0	PX0

SYMBOL	POSITION	NAME AND SIGNIFICANCE
–	IP.7	(RESERVED)
–	IP.6	(RESERVED)
–	IP.5	(RESERVED)
PS	IP.4	SERIAL PORT PRIORITY CONTROL BIT SET/ CLAER BY SOFTWAER TO SPECIFY HIGH/LOW PRIORITY INTERRUP FOR SERIAL PORT
PT1	IP.3	TIMER 1 PRIORITY CONTROI BIT SET/CLAER BY SOFTWAER TO SPECIFY HIGH/LOW PRIORITY INTERRUPT FOR TIMER COUNTER 1
PX1	IP.2	EXTERNAL 1 PRIORITY CONTROL BIT. SET/ CLAER BY SOFTWAER TO ENABLE/DISABLE

(ต่อ)

		INTERRUPT FROM INT1
PT0	IP.1	TIMER 0 PRIORITY CONTROL BIT SET/ CLAERED BY SOFTWAER TO SPECIFY HIGH/LOW PRIOITY INTERRUPT FOR TIMER/COUNTER 0
PX0	IP.0	EXTERNAL INTERRUPT 0 PRIORITY CONTROLBIT. SET/CLAERED BY SOFTWAER TO SPECIFY HIGH/ LOW PRIORITY INTERRUPT FOPR INT0

D) P-INTERRUPT PRIORITY CONTROL REGISTER

รูปที่ 2.15 รีจิสเตอร์การอินเทอร์รัพท์อีนาเบิ้ล (IE)และรีจิสเตอร์ลำดับการอินเทอรัรัพท์ (IP)

อินเทอร์รัพท์แต่ละแหล่งสามารถที่จะอีนาเบิ้ลและดิสเอเบิ้ลด้วยการเซท และเคลียร์ค่าบิทต่าง ๆ ในรีจิสเตอร์ IE ซึ่งในรูปที่ 2.15 จะแสดงรายละเอียดของรีจิสเตอร์ IE ในการเซทค่าบิทต่าง ๆ เพื่อควบคุมการอินเทอร์รัพท์แต่ละแบบ

แต่ละแหล่งอินเทอร์รัพท์สามารถที่จะโปรแกรมให้มีระดับไพรโอริตี้สูง หรือต่ำได้ด้วยการเซทหรือเคลียร์ค่าบิทต่าง ๆ ใน IP ของ SFR ตามรูปที่ 2.15 เป็นรายละเอียดของ IP โดยที่ตัวแฟลกอินเทอร์รัพท์ทุกตัวสามารถเซท หรือเคลียร์ได้ด้วยซอฟร์แวร์ ซึ่งจะมีผลเช่นเดียวกับผลที่เกิดขึ้นจากฮาร์ดแวร์

2.11.1 โครงสร้างลำดับความสำคัญการอินเทอร์รัพท์

การอินเทอร์รัพท์ความสำคัญต่ำสามารถที่ถูกอินเทอร์รัพท์ด้วยตัวเอง หรือด้วยการอินเทอร์รัพท์จากความสำคัญสูงแต่ไม่สามารถที่จะถูกอินเทอร์รัพท์จากความสำคัญต่ำตัวอื่นได้ การอินเทอร์รัพท์ความสำคัญสูงไม่สามารถที่จะถูกอินเทอร์รัพท์ได้ด้วยการทำงานตามกฎเหล่านี้ ระบบการอินเทอร์รัพท์จะประกอบด้วยตัวที่ไม่สามารถกำหนดแอดเดรสสองตัวคือ "Priority Level Active" กับ "Flip-Flop" ตัวหนึ่งเป็นตัวแสดงถึงการอินเทอร์รัพท์ความสำคัญสูงกำลังได้รับ

การบริการและการอินเทอร์รัพท์ตัวอื่นจะถูกกันหมด อีกตัวเป็นการแสดงถึงการอินเทอร์รัพท์ความสำคัญต่ำกำลังได้รับบริการและกันตัวอื่นหมด แต่การอินเทอร์รัพท์ความสำคัญสูงยังคงทำงานต่อ

ในเหตุการณ์ที่มีการร้องของระดับความสำคัญเดียวกัน ถูกรับเข้ามาพร้อมกันการหาลำดับการใช่ร่วมกันก่อนหลังภายในเมื่อการร้องขอได้รับบริการ ดังนั้นระดับความสำคัญภายในแต่ละอันจะมีการหาระดับโครงสร้างความสำคัญการอินเทอร์รัพท์ที่มีลำดับการใช้ร่วมกันก่อนหลังดังนี้

แหล่งที่มาการอินเทอร์รัพท์	ลำดับความสำคัญภายใน
การอินเทอร์รัพท์ 0 จากภายนอก	(สูงสุด) 1
การเกิด Overflow ของตัวจับเวลา/ตัวนับ 0	2
การอินเทอร์รัพท์ จากภายนอก	3
การเกิด Overflow ของตัวจับเวลา/ตัวนับ 1	4
พอร์ทอนุกรม	5

แหล่งกำเนิดการอินเทอร์รัพท์ทั้งหมดจะถูกตรวจสอบตามลำดับ ระหว่างช่วงวัฏจักรแต่ละลูก เช่น ถ้าเกิดที่ S6 ของวัฏจักรใด ๆ โดยทุกตัวการร้องขอการอินเทอร์รัพท์ถูกตรวจสอบพบ และมีการจัดลำดับความสำคัญ การตอบสนองการร้องขอของตัวอินเทอร์รัพท์สูงสุดจะถูกให้ทำงานที่สถานะ 1 ของวัฏจักรตัวต่อมา การตอบสนองการอินเทอร์รัพท์ที่กล่าวมานี้จะไม่ถูกกันออกจากเหตุการณ์ที่เกิดขึ้นใด ๆ ต่อไปนี้

1. การอินเทอร์รัพท์ของระดับความสำคัญที่เท่ากัน หรือสูงกว่ากำลังทำงานจะสิ้นสุดแล้วนั่นเอง

2. วัฏจักรแมชชีนที่เกิดระหว่างนี้ จะไม่เป็นวัฏจักรสุดท้ายในการทำงานตามคำสั่งที่กำลังทำอยู่ หรือการอินเทอร์รัพท์ที่ร้องขอมาจะยังได้รับการตอบสนองจนกว่าการทำคำสั่ง ณณะนั้นจะสิ้นสุดสมบูรณ์

3. คำสั่งในขณะนั้นเป็น RETI หรือการเข้าถึงรีจิสเตอร์ IE หรือ IP ของ SFR หรือการร้องขอการอินเทอร์รัพท์จะไม่ได้รับการตอบรับหลังคำสั่ง RETI หรือหลังการอ่านและเขียนเข้ารีจิสเตอร์ IE และ IP จะได้รับการตอบรับจนกว่าจะต้องทำคำสั่งอย่างน้อยหนึ่งคำสั่งไปแล้ว

ถ้ากรณีใด ๆ จากข้างบนนี้เกิดขึ้น ผลของการใช้อินเทอร์รัพท์ร่วมกันจะถูกละเลย ถ้าไม่เกิดกรณีใดจากข้างบนนี้ปรากฏ ผลของการใช้อินเทอร์รัพท์ร่วมกันจะทำงานช่วงวัฏจักรแมชชีนลูกต่อมา

2.11.2 การอินเตอร์รัพท์จากภายนอก

แหล่งกำเนิดภายนอกสามารถที่จะถูกโปรแกรมเลือกระดับการแอ็กทีฟ หรือช่วงการเปลี่ยนแปลงด้วยการเซทหรือเคลียร์บิทที่ IT1 หรือ IT0 ในรีจิสเตอร์ TCON ถ้า ITX=0 การอินเทอร์รัพท์ภายนอก X จะถูกกระตุ้นระดับต่ำที่ขา INTX แต่ถ้า ITX=1 การอินเทอร์รัพท์ภายนอก X เป็นการกระตุ้นใช้ของสัญญาณในโหมดนี้ ถ้าตัวอย่างสัญญาณของเขา INTX แสดงถึงระดับสูงในวัฏจักรลูกหนึ่งและต่ำในวัฏจักรอีกลูกหนึ่ง แฟลกการร้องของอินเทอร์รัพท์ IEX ในรีจิสเตอร์ TCON จะถูกเซท ดังนั้น แฟลกบิทของ IEX จะเป็นการแสดงถึงการร้องขออินเทอร์รัพท์

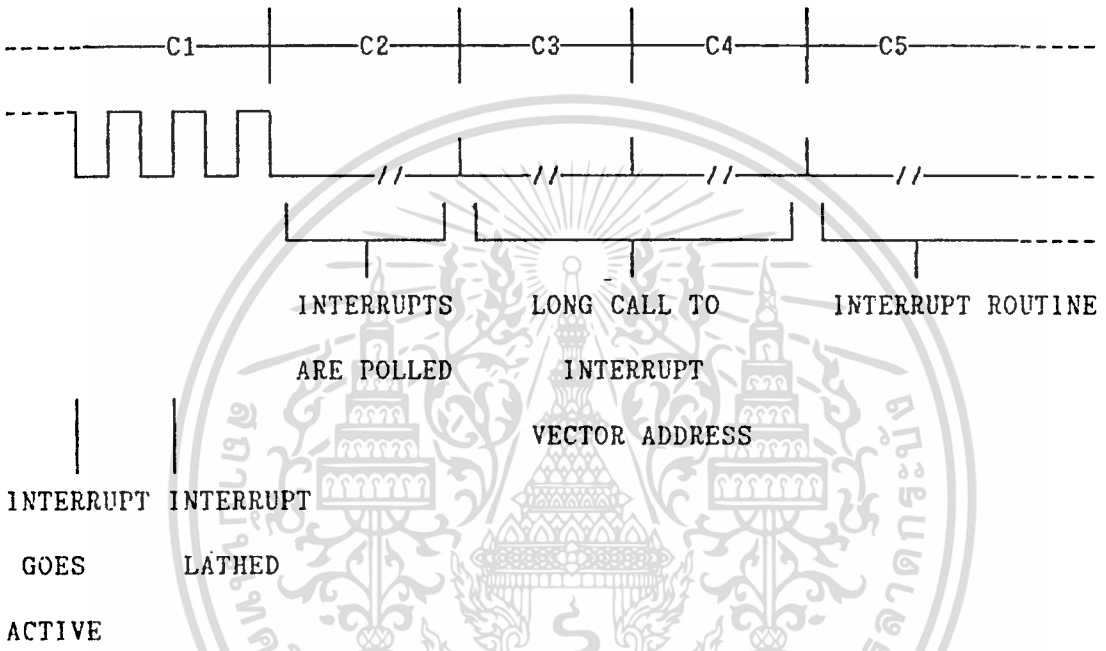
เพราะสัญญาณที่ขาการอินเทอร์รัพท์จะถูกสุ่มตัวอย่างหนึ่งครั้งในแต่ละวัฏจักรแมชชีน สัญญาณที่เข้าจะต้องรักษาระดับสูงหรือต่ำอย่างน้อยภายในช่วง 12 คาบของความถี่ออสซิลเลเตอร์ เพื่อให้มั่นใจในการสุ่มตัวอย่างที่รับเข้าไปได้ค่าแน่นอน ถ้าการอินเทอร์รัพท์ภายนอกถูกแอ็กทีฟเปลี่ยนแปลง แหล่งสัญญาณภายนอกจะต้องรักษาค่าการร้องขอสถานะสูงเป็นเวลาอย่างน้อยหนึ่งลูก และรักษาค่าสถานะต่ำอีกอย่างน้อยเป็นเวลาหนึ่งวัฏจักรเพื่อให้แน่ใจว่าการเปลี่ยนแปลงค่าจะสามารถให้แฟลกการร้องขออินเทอร์รัพท์ของ IEX จะถูกเซท ค่าใน IEX จะถูกเคลียร์โดยอัตโนมัติด้วยซีพียู เมื่อโปรแกรมการบริการอินเทอร์รัพท์ถูกเรียกมาใช้

ถ้าการอินเทอร์รัพท์ภายนอกอยู่ในระดับการแอ็กทีฟ แหล่งภายนอกจะต้องเก็บการแอ็กทีฟการร้องขอไว้จนกว่าสัญญาณการร้องขออินเทอร์รัพท์จะถูกสร้างขึ้นมาเรียบร้อยแล้ว แล้วมันจะต้องกลับมารับแอ็กทีฟการร้องขอใหม่ก่อนที่จะทำงานบริการอินเทอร์รัพท์เดิมจะสิ้นสุดลง หรือการอินเทอร์รัพท์อีกลูกหนึ่งจะถูกสร้างขึ้นมาใหม่

2.11.3 ช่วงเวลาการตอบสนอง

ระดับของ INT0 และ INT1 จะถูกแลทช์เก็บไว้ในรีจิสเตอร์ภายในช่วง S5P2 ของทุก ๆ วัฏจักรแมชชีนค่าที่เก็บจะยังไม่นำมาใช้โดยวงจรจนกว่าจะถึงวัฏจักรแมชชีนลูกใหม่ ถ้าการร้องขอครั้งหนึ่งแอ็กทีฟและข้อแม้ต่าง ๆ ถูกต้องสำหรับการทำให้มีการตอบรับทางฮาร์ดแวร์ก็จะเรียก

โปรแกรมย่อยเพื่อตอบรับการบริการการร้องขอของอินเทอร์รัพท์ในอีกคำสั่งต่อมาจะถูกทำงานการเรียกโปรแกรมตัวเองจะใช้เวลาสองลูกคลื่น ดังนั้น จะต้องใช้อย่างน้อยสามวัฏจักรแมชชีนช่วงระหว่างการเริ่มการร้องขออินเทอร์รัพท์ภายนอกแอ็กทีฟ จนกระทั่งถึงการเริ่มทำงานคำสั่งแรกของโปรแกรมย่อย การบริการอินเทอร์รัพท์ ดังรูปที่ 2.16 จะแสดงช่วงเวลาการตอบสนองการอินเทอร์รัพท์



รูปที่ 2.16 แสดงช่วงเวลาการตอบสนองการอินเตอร์รัพท์

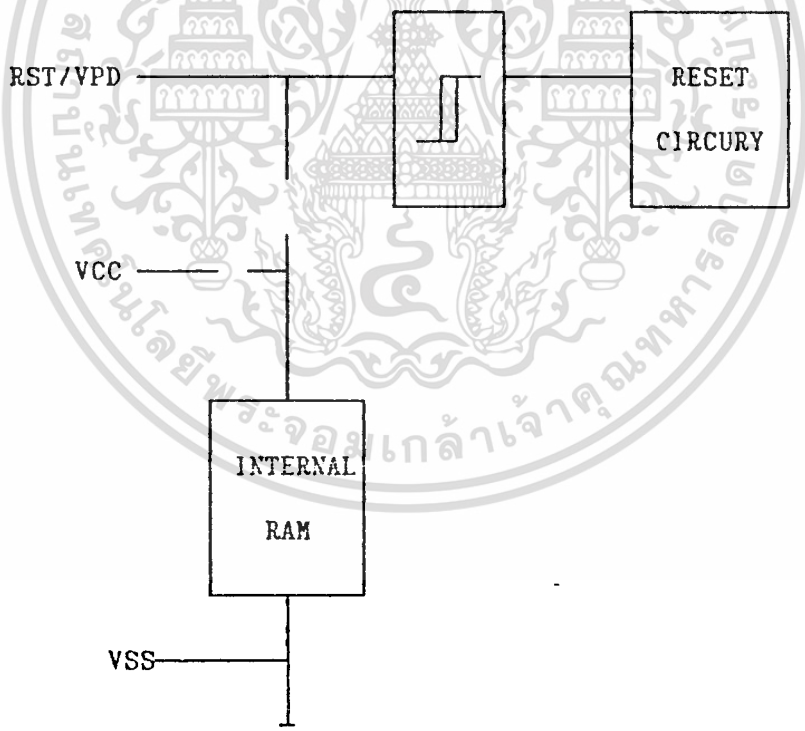
ช่วงเวลาตอบสนองที่ยาวนานกว่าอาจเกิดขึ้นได้ถ้าการร้องขอถูกบล็อกด้วยข้อมูลต่าง ๆ ของการจัดลำดับความสำคัญของการอินเทอร์รัพท์สามกรณีที่กล่าวมาแล้ว ถ้าการอินเทอร์รัพท์ที่มีความสำคัญเท่ากันหรือสูงกว่าได้ทำงานสมบูรณ์ไปแล้ว ช่วงเวลาที่รอทั้งหมดจะขึ้นอยู่กับการใช้เวลาการอินเทอร์รัพท์นั้น ๆ ถ้ากำลังทำคำสั่งอยู่ที่ยังไม่ถึงวงจรรอบสุดท้ายช่วงเวลาที่รอทั้งหมดจะไม่สามารถมีค่าเกินกว่า 3 วัฏจักร เพราะคำสั่งที่ใช้เวลายาวนานที่สุด เช่น MUL และ DIV จะใช้ 4 วัฏจักร

ถ้าอยู่ในช่วงที่กำลังทำคำสั่ง RETI หรือกำลังเข้าถึงรีจิสเตอร์ IE หรือ IP เวลาที่รอทั้งหมดก็จะไม่สามารถมีค่าเกินกว่า 5 วัฏจักร โดยคำสั่งสุดสุดจะคิดที่หนึ่งวัฏจักรในช่วงที่กำลังทำงานคำสั่งที่เกิดการอินเทอร์รัพท์อยู่และบวกกับอีกสี่วัฏจักรเพื่อให้สิ้นสุดคำสั่งต่อมา ถ้าคำสั่งต่อมาคือคำสั่ง MUL หรือ DIV

ดังนั้นในระบบการอินเทอร์รัพท์ครั้งหนึ่ง ช่วงเวลาที่ตอบสนองจะอยู่ระหว่าง 3 วัฏจักรถึง 6 วัฏจักรเสมอ

2.12 รีเซท (RESET)

วงจรรีเซทสำหรับ 8032 จะต่อที่ขารีเซทคือขา RST/VPD ดังแสดงในรูปที่ 2.17 วงจร Schmitt Trigger ถูกใช้เป็นตัว Input สำหรับขจัดสัญญาณรบกวน Noise และ Output ของ Schmitt Trigger จะถูกกลุ่มเก็บเข้าไปด้วยวงจรรีเซทเกิดที่ S5P2 ของทุก ๆ วัฏจักรแมชชีน



รูปที่ 2.17 วงจรการจัดการรีเซท

8032 จะทำงานได้ด้วยการรีเซทสถานะสูงที่ขา RST/VPD เป็นเวลาอย่างน้อย 2 วัฏจักรแมชชีน ขณะที่ออสซิลเลเตอร์กำลังทำงาน ตัวชีพียูจะตอบสนองด้วยการทำงานแบบรีเซทภายในมันจะกำหนดให้ขา ALE และ PSEN เป็นขาอินพุท ซึ่งปกติมันจะเป็น Quasi-Bidirectional การรีเซทภายในจะเริ่มขึ้นระหว่างวัฏจักรที่ 2 ในขณะที่ขา RST มีสถานะสูงและจะซ้ำทุกวัฏจักรแมชชีนจนกว่าขารีเซทมีสถานะต่ำ มันจะทำให้รีจิสเตอร์ภายในมีค่าสถานะดังรูปที่ 2.18

REGISTER	VALUE IN BINARY
.ACC	0000 0000
.B	0000 0000
.PSW	0000 0000
SP	0000 0111
DPTR	
DPH	0000 0000
DPL	0000 0000
.P0	1111 1111
.P1	1111 1111
.P2	1111 1111
.P3	1111 1111
.IP	8032 XXX0 0000
	8052 XX00 0000
.IE	8051 0XX0 0000
	8052 0X00 0000
TMOD	0000 0000
.TCON	0000 0000
+T2CON	0000 0000

TH0	0000 0000
TL0	0000 0000
TH1	0000 0000
TL1	0000 0000
+TH2	0000 0000
+TL2	0000 0000
+RCAP2H	0000 0000
+RCAP2L	0000 0000
.SCON	0000 0000
SBUF	INDETERMINATE
PCON	HMOS OXXX XXXX,OXXX 0000

NOTE. X = UNDERFINED

. = BIT ADDRESSABLE

+ = 805X2 ONLY

รูปที่ 2.18 ค่าต่าง ๆ ในรีจิสเตอร์หลังจาก POWER-ON หรือรีเซท

เป็นที่น่าสังเกตุว่าแรมภายในจะไม่มีผลจากการรีเซท เมื่อมีแรงดันไฟจ่ายที่ Vcc ค่าต่าง ๆ ในแรมจะเป็นค่าที่ไม่แน่นอน ถ้าส่วนนั้นไม่ได้กลับมาจากการใช้งานของโหมดลดพลังงาน (Reduced Power Mode)

2.13 8032 Power down

ในการใช้งานไมโครคอนโทรลเลอร์ 8032 (MCS-51) นอกจากการทำงานในโหมดปกติแล้ว 8032 ยังได้ออกแบบไว้ให้ใช้งานในโหมด Power Reduction การทำงานในโหมดนี้จะกินกระแสต่ำกว่าการใช้งานในโหมดปกติ โหมด Power Reduction แบ่งตามประเภทของชิป 8032 ได้ 2 ประเภทคือ HMOS Version (เบอร์ 8051, 8031, 8751) และ CHMOS

Versions (เบอร์ 80c32, 80C32, 87C32)

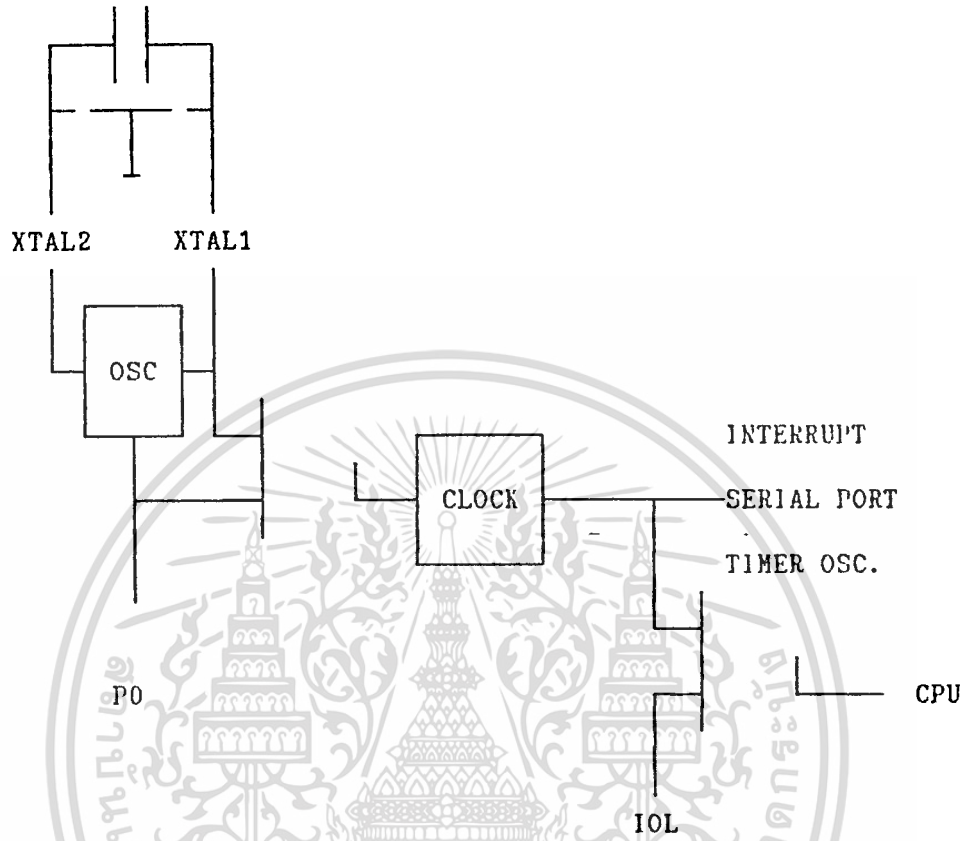
HMOS Version

สำหรับ HMOS Versions มีการทำงานในโหมด Power Reduction เพียงโหมดเดียว คือ โหมด Power Down โหมดนี้ใช้สำหรับ Backup ข้อมูลของ Internal Ram ในระหว่างการทำงานในโหมดปกติ (Normal Mode) Internal Ram จะได้รับไฟเลี้ยงจากขา Vcc และเมื่อเข้าสู่การทำงานในสภาวะ Power Down Mode ดูรูปที่ 2.17 แรงไฟที่จะนำมาจ่ายให้ Internal Ram จะได้มาจากขา TST/VPD ด้วยวิธีนี้ Backup Battery จะถูกใช้เพื่อเก็บข้อมูลใน Ram เมื่อ Power Fail เท่านั้น จากรูปที่ 2.17 ผู้ใช้ไม่สามารถต่อ Backup Battery เข้าที่ขา RST/VPD ของ 8032 ได้เพราะจะทำให้ 8032 ถูกรีเซท ดังนั้นในการใช้งานจริงผู้ใช้จะต้องสร้างวงจรอีกส่วนหนึ่งเพื่อตรวจจับสภาวะ Power Fail และเมื่อเข้าสู่สภาวะPower Fail ให้ส่งสัญญาณไปอินเทอร์รัพท์ CPU ผ่านขา INTO หรือ INT1 เพื่อทำการย้ายข้อมูลที่ต้องการ Backup ลง Internal Ram และทำการต่อ Backup Battery เข้าที่ขา RST/VPD ก่อนที่ไฟ Vcc จะตกลงต่ำกว่าแรงไฟที่ใช้งานต่ำสุด เมื่อแรงไฟ Vcc กลับมาตามปกติ แรงไฟที่ขา VPD ที่ได้จาก Backup Battery จะต้องคงอยู่ชั่วครู่เพื่อใช้รีเซทซีพียู (ออสซิลเลเตอร์จะเริ่มทำงานหลังจากที่ซีพียูได้รับสัญญาณรีเซทไม่น้อยกว่า 2 แมชชีนไซเคิล) หลังจากนั้น 8032 จะกลับเข้าสู่การทำงานในสภาวะปกติ

CHMOS Versions

สำหรับ CHMOS Versions มีโหมด Power Reduction อยู่ 2 โหมดคือ Idle Mode และ Power Down Mode ในการใช้งานทั้ง 2 โหมดนี้ แรงไฟจาก Backup Battery จะจ่ายเข้าที่ขา Vcc ซึ่งต่างกับใน HMOS Versions ที่แรงไฟ Backup จะจ่ายเข้าที่ขา RST รูปที่ 2.19 แสดงวงจรภายในซีพียูในส่วนของ Idle และ Power Down Mode ใน Idle Mode (IDL=1) วงจรออสซิลเลเตอร์ทำงาน Interrupt, Serial Port และ Timer ยังคงทำงานเนื่องจากยังมีสัญญาณ Clock ป้อนอยู่ แต่จะไม่มีสัญญาณ Clock ไปจ่ายให้กับซีพียู ใน Power Down Mode (PD=1) จะทำให้วงจรออสซิลเลเตอร์หยุดทำงาน ทั้ง Idle และ Power Down Mode จะทำงานเมื่อมีการเซ็ทบิทใน PCON แอดเดรสของรีจิสเตอร์ PCON นี้อยู่ที่

แอดเดรส 87H



รูปที่ 2.19 Idle และ Power Down Hardware

Idle Mode

ซีพียูจะเข้าสู่การทำงานในโหมดนี้ได้โดยใช้คำสั่งเซ็ท PCON.0 ซึ่งซีพียูจะกระทำคำสั่งนี้เป็นคำสั่งสุดท้าย แล้วหลังจากนั้นจะเข้าสู่การทำงานใน Idle Mode ใน Idle Mode นี้สัญญาณ Internal Clock จะไม่ถูกจ่ายให้ซีพียูแต่จะจ่ายให้ในส่วนของ Interrupt, Timer และ Serial Port สภาวะต่าง ๆ ของซีพียูจะยังคงเดิมทั้งหมดเช่น Stack Pointer, Program Counter, Program Status Word, Accumulator และรีจิสเตอร์อื่น ๆ ทั้งหมดจะยังคงค่าอยู่เดิมที่ Port ทุกขา (ทุกบิท) จะยังคงสถานะเดิม ส่วนที่ขา ALE และ PSEN มีสภาวะเป็น High

วิธีที่จะออกจาก Idle Mode นี้มีอยู่ 2 วิธีคือเมื่อเกิดการ Interrupt ไม่ว่าจะเกิดจากแหล่งใดก็ตามจะมีผลทำให้ PCON.0 ถูก Clear โดย Hardware ทำให้ซีพียูออกจากการทำงาน

ใน Idle Mode กลับเข้าสู่การทำงานโหมดปกติ หลังจากนั้นซีพียูจะทำงานตามส่วนของโปรแกรมบริการอินเทอร์รัพท์จนถึงคำสั่ง RETI ซึ่งเป็นคำสั่งสุดท้ายของโปรแกรมบริการอินเทอร์รัพท์ ซีพียูจะทำการ Execute คำสั่งที่อยู่ถัดไปจากคำสั่ง Set PCON.0 ซึ่งคำสั่ง Set PCON.0 นี้เป็นคำสั่งที่ซีพียู Execute เป็นคำสั่งสุดท้ายก่อนที่จะเข้าสู่การทำงานใน Idle Mode

แฟลกบิท GF0 และ GF1 อยู่ใน PCON ผู้ใช้สามารถใช้เพื่อบอกให้ทราบว่ามีการอินเทอร์รัพท์เกิดขึ้นระหว่างการทำงานในโหมดปกติ หรือระหว่างการทำงานใน Idle Mode

วิธีที่สองที่จะออกจาก Idle Mode ได้คือ Hardware Reset โดยในขณะที่สัญญาณนาฬิกาออสซิลเลเตอร์ยังคงทำงานอยู่สัญญาณรีเซทจากฮาร์แวร์จะต้องคงอยู่ชั่วครู่ไม่น้อยกว่า 2 แมชชีนไซเคิล เมื่อเกิดสัญญาณรีเซทที่ขา RST จะเป็นผลทำให้บิท JD2 ถูกเคลียร์โดยตรง ในขณะเดียวกันซีพียูจะเริ่มรันโปรแกรมในคำสั่งถัดไป ซึ่งคำสั่งนี้จะเป็นคำสั่งที่อยู่ถัดไปจากคำสั่ง Set PCON.0

Power Down Mode

ซีพียูจะเข้าสู่การทำงานในโหมดนี้โดยใช้คำสั่ง Set PCON.1 ซึ่งคำสั่งนี้จะเป็นคำสั่งสุดท้ายที่ถูก Execute ก่อนที่ซีพียูจะเข้าสู่ Power Down Mode ออสซิลเลเตอร์ที่อยู่บนชิปซีพียูจะหยุดทำงาน (ดูรูปที่ 2.19) เป็นผลทำให้ไม่มีสัญญาณนาฬิกาด้วยฟังก์ชั่นทั้งหมดหยุดการทำงานแต่ Internal Ram และ SFR ยังคงเดิมอยู่ สภาวะต่าง ๆ ที่อาพอร์ทเอาท์พุทยังคงค่าเดิมซึ่งจะถูกเก็บไว้ที่ SFR ที่ขา ALE และ PSEN จะมีสภาวะเป็น Low

วิธีที่จะออกจาก Power Down Mode สำหรับ 8032 คือการรีเซททางฮาร์ดแวร์ เมื่อซีพียูถูกรีเซทจะเป็นผลทำให้ค่าของ SFR ทั้งหมดถูกกำหนดค่าใหม่ ดังรูปที่ 2.18 แต่ค่าใน Internal Ram ยังคงเดิมไม่เปลี่ยนแปลงการทำงานใน Power Down Mode นี้แรงไฟ Vcc สามารถลดลงได้ถึง 2 โวลท์ ซึ่งจะทำให้ซีพียูกินกระแสต่ำสุดการใช้งานในโหมดนี้มีข้อพึงระวังคือต้องแน่ใจว่าแรงไฟ Vcc จะไม่ลดลงก่อนเข้าสู่ Power Down Mode และแรงไฟ Vcc จะกลับเข้าสู่ระดับที่ใช้ในการทำงานปกติก่อนที่จะออกจาก Power Down Mode เมื่อซีพียูถูกรีเซทนอกจากจะเป็นผลให้สิ้นสุดการทำงานใน Power Down Mode แล้วยังทำให้ออสซิลเลเตอร์เริ่มทำงานอีกครั้ง สัญญาณรีเซทนี้จะไม่แอ็กทีฟก่อนที่แรงไฟ Vcc จะกลับเข้าสู่ในระดับที่ใช้ในการทำงานปกติและสัญญาณรีเซทต้องนานพอที่จะทำให้ออสซิลเลเตอร์เริ่มทำงานใหม่อย่างมีเสถียรภาพ

บทที่ 3
การสื่อสารข้อมูล

3.1 บทนำ

การสื่อสารเพื่อถ่ายโอนข้อมูลแบ่งออกเป็นแบบขนานและอนุกรม แต่สำหรับในที่นี้จะเน้นกล่าวเฉพาะการสื่อสารข้อมูลแบบอนุกรม

ถ้าพูดถึงเรื่องของการสื่อสารเกี่ยวกับระบบไมโครคอมพิวเตอร์ ส่วนใหญ่คงจะรู้จักและคุ้นเคยกับการสื่อสารแบบอนุกรม (Serial) ในมาตรฐาน RS-232 เป็นอย่างดี ซึ่งปัจจุบันระบบคอมพิวเตอร์ใช้งานธุรกิจทั่วไปก็มักจะมี RS-232 มาให้พร้อม เพราะ RS-232 สามารถต่อเข้ากับ Modem เพื่อการสื่อสารทางโทรศัพท์และต่อเข้ากับเครื่องมือต่าง ๆ เช่น Mouse รวมทั้งงานในระดับไมโครคอนโทรลเลอร์ก็มักจะมี RS-232 ประกอบเข้ามาด้วย ซึ่งดูได้จาก CPU รุ่นใหม่ ๆ ทั้งหลายที่มักจะมี RS-232 อยู่ภายในระดับชิพเลยทีเดียวเช่น CPU ในตระกูล MCS-51 ต่อไปเราจะมาทำความเข้าใจถึงพื้นฐานการส่งข้อมูลและระหัสที่ใช้

3.2 วิธีของการถ่ายโอนข้อมูล

เพื่อให้เข้าใจระบบการถ่ายโอนข้อมูล เราจะมามองดูก่อนว่าสัญญาณที่ส่งออกมาจากเครื่องและรับเข้าไปในเครื่องไปอย่างไร

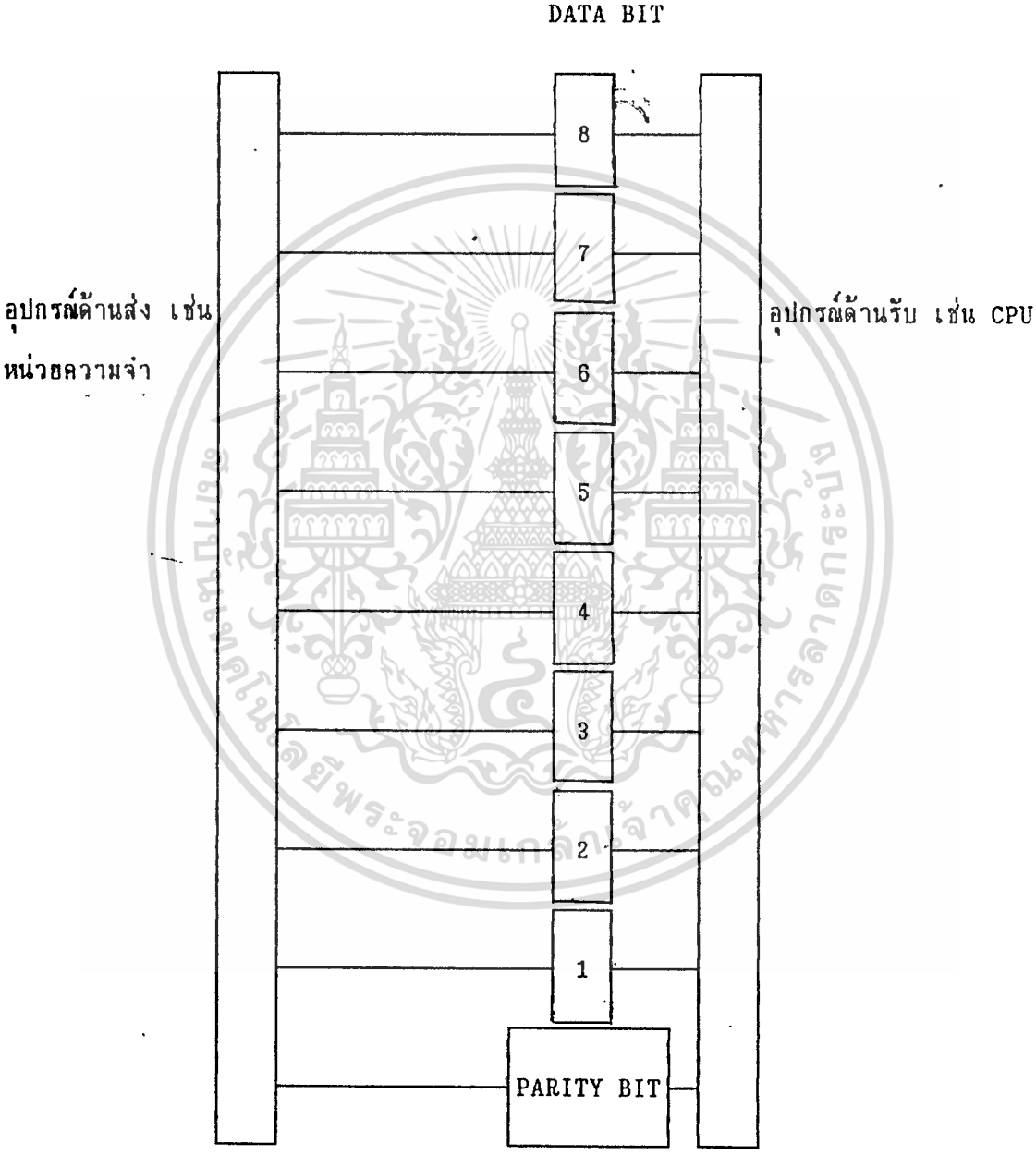
3.2.1 การถ่ายโอนข้อมูลแบบขนาน

ลักษณะของการส่งข้อมูลแบบขนาน ทำได้โดยการส่งข้อมูลออกมาทีละ 1 ไบท์ คือ 8 บิท จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ตัวกลางระหว่าง 2 เครื่องจะต้องมีช่องทางให้ข้อมูลเดินทางน้อย 8 ช่องทาง โดยมากจะเป็นสายขนานให้กระแสไฟฟ้าวิ่งมากกว่าจะเป็นตัวกลางอื่น เนื่องจากมีสัญญาณสูญหายไปกับความต้านทานของสาย ระยะทางระหว่าง 2 เครื่องไม่ควรจะเกิน 100 ฟุต ปัญหาที่เกิดขึ้นหากระยะทางสายมากกว่านี้ก็คือ ระดับของกราวด์ทางไฟฟ้าที่จุดรับก็จะผิดไปจากจุดส่งทำให้เกิดการผิดพลาดในการรับสัญญาณลิจิกทางฝ่ายรับ

นอกจากสายที่เป็นทางเดินของข้อมูลแล้วอาจจะมีทางเดินของสัญญาณควบคุมอื่น ๆ อีกเป็น

ต้นว่า บิทที่บอกพาริตี้ของสัญญาณ เพื่อเป็นการตรวจสอบความผิดพลาดของการรับสัญญาณที่ปลายทาง หรือสายที่ควบคุมการโต้ตอบ (Hand-Shake)

จะเห็นว่าการส่งแบบขนานส่วนมากจะทำให้ระยะใกล้ ๆ เนื่องจากจะต้องมีช่องทางเดินของสัญญาณมากกว่า 8 สาย และอุปกรณ์ที่ติดต่อแบบขนานกับคอมพิวเตอร์ก็เห็นจะได้แก่เครื่องพิมพ์



รูปที่ 3.1 การส่งข้อมูลแบบขนาน

3.2.2 การถ่ายโอนข้อมูลแบบอนุกรม

ในการถ่ายโอนข้อมูลแบบอนุกรม ข้อมูลถูกส่งออกมาทีละบิท ระหว่างจุดส่งและจุดรับจะเห็นว่าการส่งข้อมูลแบบนี้จะช้ากว่าแบบขนานที่กล่าวมาแล้วแน่นอนแล้วทำไมต้องส่งแบบนี้ละคำตอบก็คือตัวกลางการสื่อสารต้องการเพียงช่องเดียวหรือสายเพียงคู่เดียว ค่าใช้จ่ายในสื่อกลางจะต้องถูกกว่าแบบขนานแน่นอน สำหรับการส่งระยะทางไกล ๆ โดยเฉพาะเมื่อเรามีระบบสื่อสารทางโทรศัพท์ไว้ใช้งานอยู่แล้วย่อมจะเป็นการประหยัดกว่าที่จะทำการติดต่อสื่อสารทีละ 8 ช่อง เพื่อการถ่ายโอนข้อมูลแบบขนานอย่างแน่นอน

รูปที่ 3.2 แสดงให้เห็นการส่งข้อมูลแบบอนุกรม ข้อมูลจากจุดส่งจะถูกเปลี่ยนให้เป็นอนุกรมเสียก่อนแล้วค่อยทยอยส่งออกทีละบิทไปยังจุดรับ ณ ที่จุดรับจะต้องมีกลไกในการเปลี่ยนข้อมูลมาทีละบิทให้เป็นสัญญาณแบบขนานซึ่งลงตัวพอดีนั่นคือ บิท 1 ลงที่บัสข้อมูลเส้นที่ 1 พอดี การที่จะทำให้การแปลงสัญญาณจากอนุกรมทีละบิทให้ลงพอดีนั้น จำเป็นจะต้องมีกลไกที่เหมาะสม เพื่อป้องกันการผิดพลาดในการรับ กลไกที่ว่านี้ 2 แบบคือ

1. การสื่อสารแบบซิงโครนัส
2. การสื่อสารแบบอะซิงโครนัส

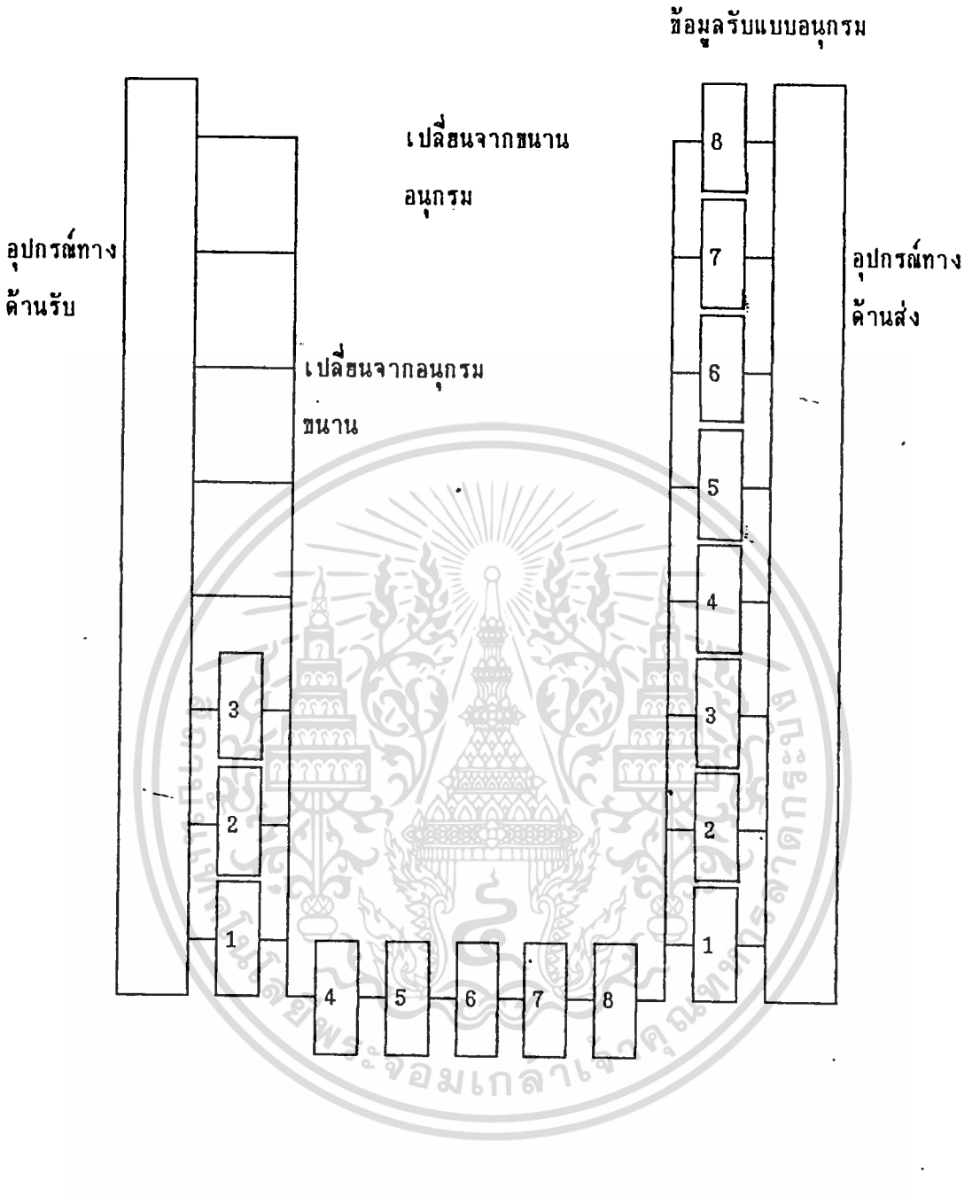
รายละเอียดของทั้งสองแบบจะกล่าวในหัวข้อถัดไป

การติดต่อแบบอนุกรมอาจจะแบ่งตามรูปลักษณะได้ 3 แบบ ตามรูปที่ 3.3 ดังนี้

1. แบบซิมเพล็กซ์ (Simplex) ข้อมูลส่งได้ในทางเดียวเท่านั้น บางครั้งก็เรียกว่าการส่งทิศทางเดียว (Unidirectional Data Bus)

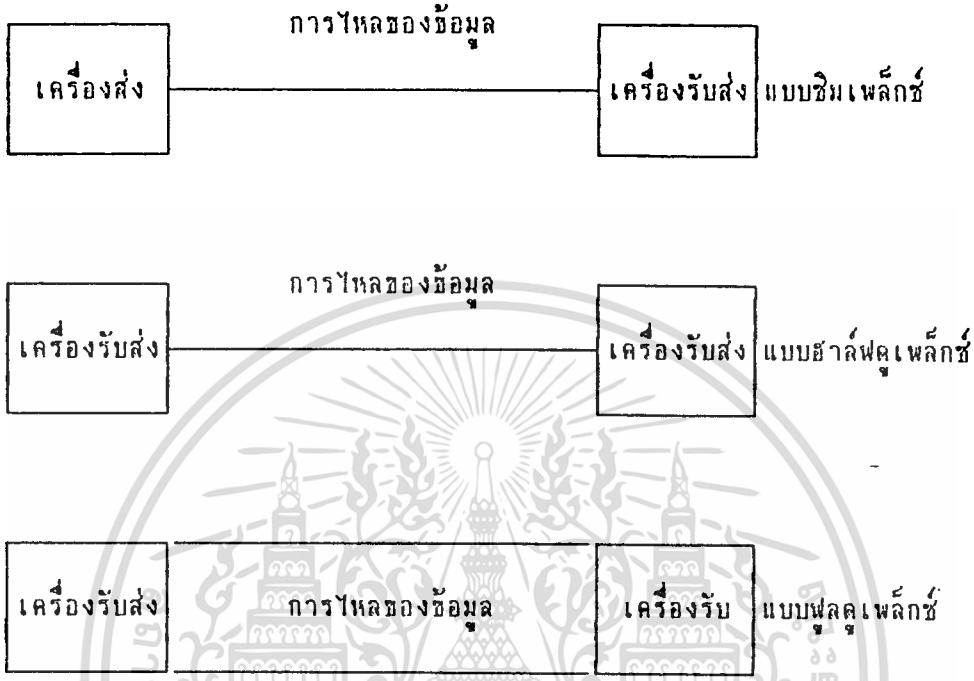
2. แบบฮาล์ฟดูเพล็กซ์ (Half Duplex) ข้อมูลสามารถส่งได้ทั้งสองสถานี แต่จะต้องผลัดกันส่งและผลัดกันรับ จะส่งและรับพร้อมกันไม่ได้

3. แบบฟูลดูเพล็กซ์ (Full Duplex) ทั้งสองสถานีสามารถรับและส่งได้ในเวลาเดียวกัน การส่งแบบฟูลดูเพล็กซ์และฮาร์ฟดูเพล็กซ์ ไม่ขึ้นอยู่กับจำนวนของสายในการติดต่อบางครั้ง คำว่า



รูปที่ 3.2 การส่งข้อมูลแบบอนุกรม

3.3 รูปแบบการติดต่อสื่อสารแบบอนุกรม



รูปที่ 3.3 รูปแบบการติดต่อสื่อสารข้อมูลแบบอนุกรม

ทูไวร์ (Two Wire) หรือสองเส้น และโฟร์ไวร์ (Four Wire) หรือ 4 เส้นใช้ในการบรรยายถึงลักษณะการสื่อสารข้อมูลซึ่งอาจจะทำให้เข้าใจและฮาล์ฟดูเพล็กซ์ สายโทรศัพท์โดยทั่วไปเป็นแบบ 2 เส้น ส่วนในสายที่เป็นแบบเช่า (Lease Line) ส่วนมากจะเป็น 4 เส้น

3.4 ความเร็วในการถ่ายโอนข้อมูลแบบอนุกรม

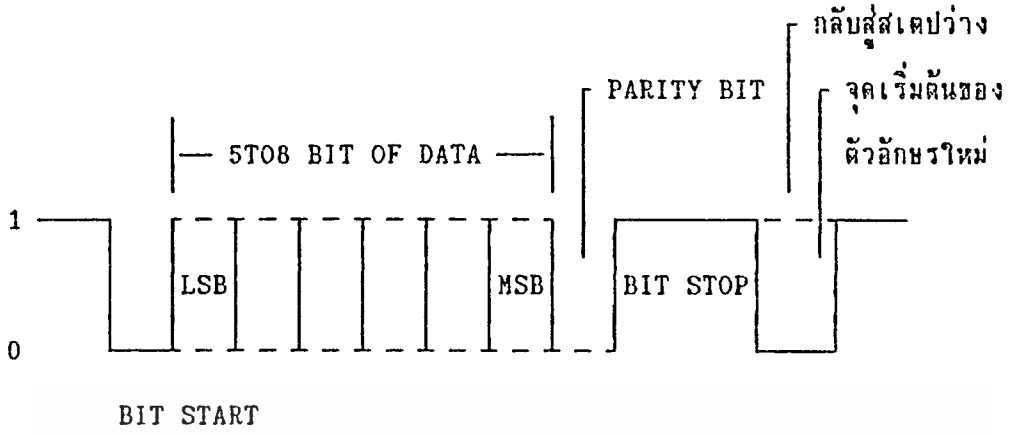
ความเร็วของการถ่ายโอนข้อมูลแลลอนุกรม หน่วยวัดเป็นบิทต่อวินาที (BPS) หน่วยที่บรรยายถึงการเปลี่ยนแปลงของสัญญาณใน 1 วินาที เรียกว่า บอดเรต (Baud Rate) หรืออัตราบอด หลายคนยังเข้าใจสับสนระหว่างอัตราบอดและอัตราบิท (Bit Rate) การเปลี่ยนแปลงของสัญญาณ 1 ครั้ง อาจจะแสดงถึงการส่งข้อมูลแบบอนุกรมมากว่า 1 บิท (รายละเอียดเกี่ยวกับเรื่องนี้จะได้กล่าวถึงอีกครั้งในเรื่องโมเด็ม) ถ้าเขียนในรูปของสมการทางคณิตศาสตร์เราก็จะได้

อัตราบิท (Bit Rate) = อัตราบอด (Bound Rate) * (บิทใน 1 บอด)

3.5 การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission)

การส่งแบบอะซิงโครนัสนี้ พัฒนาการส่งโทรพิมพ์ในสมัยก่อน ลักษณะของสัญญาณแสดงไว้ในรูปที่ 3.4 เพื่อเพิ่มกลไกในการรับส่งอย่างถูกต้อง สัญญาณอะซิงโครนัส จะประกอบด้วยบิทเริ่มต้น หรือบิทสตาร์ท (Start) และบิทสิ้นสุดหรือบิทสตอป (Stop Bit)

ขณะที่สถานะของการส่งเป็นแบบว่าง (Idle) คือ ยังไม่มีสัญญาณส่งออกมาจะมีสัญญาณหรือมีแรงดัน (หรือกระแส) ตลอดเวลา เพื่อความแน่ใจฝ่ายรับยังติดต่ออยู่กับฝ่ายส่งเมื่อเริ่มจะส่งข้อมูล สัญญาณของอะซิงโครนัสจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา บิทนี้เรียกว่าสตาร์บิท ตามหลังของสตาร์บิทก็จะเป็นข้อมูลสำหรับ 1 ตัวอักษร ซึ่งอาจจะมีขนาด 5 บิท จนถึง 8 บิท โดยบิทที่มีค่าน้อยที่สุด (LSB) จะถูกส่งออกมาก่อนไล่ไปจนถึงบิทที่มีค่ามากที่สุด (MSB) การเข้ารหัสอักขระนี้ส่วนมากจะนิยมใช้รหัส ASCII แรกเริ่มที่เดียวในงานของโทรพิมพ์เขาใช้รหัส BAUDOT ซึ่งใช้ 5 บิท ในการแทนอักขระ 1 ตัว ตามหลังข้อมูลก็จะเป็นพาริตี้บิทอาจจะเป็นแบบคู่ (Even) หรือแบบคี่ (Odd) หมายความว่าหากเป็นพาริตี้คู่ จำนวนบิทที่เป็น 1 ในช่วงบิทข้อมูลกับบิทพาริตี้รวมกันแล้วจะต้องเป็นจำนวนคู่ ผู้ส่งจะต้องทำหน้าที่ตรวจสอบข้อมูลแล้วใส่พาริตี้บิทเอง ฝ่ายรับเมื่อรับแล้วก็จะต้องตรวจสอบว่าเป็นจริงดังสถานการณ์ที่ตั้งเอาไว้ หรือไม่หากผิดพลาดก็หมายความว่าสัญญาณที่รับนั้นผิดพลาดไปจากสถานีส่งส่งออกมา ทั้งนี้ทั้งนั้นจะต้องผิดเป็นจำนวนคี่เท่านั้นคือผิดไป 1 บิท 3 บิท หรือ 5 บิท พร้อมกันจึงจะตรวจสอบได้ว่าผิดมองเห็นง่าย ๆ ว่าถ้าผิดเป็นจำนวนคู่ ผลรวมของจำนวนหนึ่งก็ยังเป็นคู่อยู่ดี ทั้งนี้ทั้งนั้นไม่ได้หมายความว่า พาริตี้คู่ (Odd Parity) จะตรวจสอบความผิดพลาดเป็นจำนวนคู่ความจริงแล้วตรวจสอบความผิดพลาดได้เหมือนกับพาริตี้คู่ (Even Parity) แต่แทนที่จะตรวจสอบดูว่าสัญญาณที่รับเข้ามีจำนวนคู่ตรวจสอบดูว่ามีจำนวนคี่หรือเปล่า อย่างไรก็ตามโอกาสที่จะผิดพลาด 2 บิทพร้อมกันมีน้อยมาก



รูปที่ 3.4 ฟอร์แมตการสื่อสารแบบอัซิงโครนัส

ย้อนกลับมาดูสัญญาณอะซิงโครนัสใหม่ หลังจากบิทพาริตี้แล้วก็จะต้องมีสต๊อปบิทซึ่งเป็น 1 ความกว้างของสต๊อปบิทอาจจะเป็น 1, 1.5 หรือ 2 พัลส์ของสัญญาณนาฬิกา แล้วแต่ผู้รับและผู้ส่งจะตกลงใช้กันเอง การเริ่มใช้พอร์ทอนุกรม (ทางออกอนุกรม) จึงจำเป็นจะต้องตั้งค่าต่าง ๆ สำหรับเป็นการ ส่งแบบอนุกรมอันได้แก่

1. ความเร็วในการส่ง
2. ความยาวรหัส 1 อักขระ
3. บิทตรวจสอบ
4. จำนวนสต๊อปบิท

ในการส่งโทรพิมพ์หรือโทรเลขเมื่อก่อนนี้ใช้ความเร็วแค่ 70 บอด และ 110 บอด สำหรับคอมพิวเตอร์ความเร็วในการส่งมีให้เลือกตั้งแต่ 110, 200, 300, 1200, 2400, 4600, 9600 บอด และสูงไปกว่านั้น เนื่องจากมี IC หลายเบอร์ทำหน้าที่รับส่งแบบอะซิงโครนัสให้ใช้การส่งแบบอนุกรมจึงจะสะดวกสบายสำหรับคนออกแบบพอร์ทอนุกรม สำหรับมาตรฐานพอร์ทอนุกรมจะกล่าวถึงในบทต่อไป

จะเห็นว่ากลไกในการซิงโครนัสของการสื่อสารอะซิงโครนัส มีลักษณะเป็นไปทีละตัวอักขระ จำนวนพัลส์ของสัญญาณที่ส่งออกยังมีบางส่วนใช้ในการควบคุมการส่งอยู่อันได้แก่ บิตสตาร์ท บิท

สต๊อป และบิทพาริตี้ ทำให้ความเร็วการส่งอักขระต่อวินาทีน้อยลงไป การส่งสัญญาณด้วยความเร็ว 300 บอด สำหรับการเข้ารหัส 7 บิทไม่ได้หมายความว่าส่งได้ 300 หารด้วย 7 อักขระต่อวินาที

3.6 การสื่อสารแบบซิงโครนัส (Synchonous Transmission)

การส่งผ่านข้อมูลแบบซิงโครนัสจะทำการจัดกลุ่มของข้อมูลเป็นกลุ่ม ๆ และทำการส่งข้อมูลทั้งกลุ่มไปพร้อมกันในทีเดียว เราเรียกกลุ่มของข้อมูลนี้ว่า "Block of Data" และในการส่งผ่านข้อมูลแบบซิงโครนัสนี้ช่วงระยะเวลาของแต่ละบิทที่ทำการส่งจะใช้เวลาเดียวกัน และในการส่งผ่านทั้งตัวอักษร ตัวอักษรตัวแรกและตัวถัดไปจะไม่มีอะไรมาคั่น (ภายในบล๊อคเดียวกัน) ดังนั้นช่วงเวลาระหว่างบิทสุดท้ายของตัวอักษรกับบิทแรกของตัวอักษรตัวถัดไปเวลานี้จะเป็นศูนย์ นั่นเอง หรือกล่าวอีกนัยหนึ่ง การคิดเวลาของการส่งผ่านจะต้องคิดในรูปแบบของเวลารวมทั้งหมดของการส่งผ่านตัวอักษรที่สมบูรณ์เป็นตัว ๆ ไป ถ้าพิจารณากันแล้วเราจะเห็นแนวโน้มว่ารูปแบบของระบบส่งผ่านข้อมูลนี้มีแนวโน้มที่จะยึดหลักการส่งผ่านข้อมูลที่อยู่ในรูปของรหัส 1, 0 หรือเลขฐาน 2 นั่นเอง โดยมีข้อความที่ต้องการส่งนั้นจะอยู่ในรูปของบิทที่แน่นอน

จากรูป 3.5 (A) แสดงให้เห็นการเอากลุ่มตัวอักษร (ข้อความที่ต้องการส่งผ่าน) มาเรียงต่อกันเพื่อเตรียมส่งผ่านข้อมูลแบบซิงโครนัส เราจะเห็นได้ว่าตัวอักษรที่นำมาต่อกันเรียงชนิดกันโดยไม่มีที่ว่างระหว่างตัวอักษรเลย หรือตามที่กล่าวมาแล้วว่าช่วงเวลาระหว่างบิทสุดท้ายของตัวอักษรตัวแรกกับบิทแรกของตัวอักษรถัดไปเท่ากับศูนย์นั่นเอง ในเมื่อรูปแบบการส่งผ่านข้อมูลเป็นเช่นนี้อุปกรณ์รับข้อมูลจะต้องทราบอะไรบ้าง จึงจะระบุว่าส่วนนั้น ๆ เป็นกลุ่มบิทของตัวอักษรที่ถูกเข้ารหัสกันมา สิ่งที่อุปกรณ์รับข้อมูลจะต้องทราบก็คือ บิทใดเป็นบิทแรกของตัวอักษรตัวแรก ขนาดของตัวอักษร (จำนวนบิทที่ใช้แทนหนึ่งตัวอักษร) และอีกประการหนึ่งก็คือ ความเร็วในการส่งผ่านข้อมูล อุปกรณ์รับข้อมูลจึงจะจัดกลุ่มของบิทออกเป็นกลุ่ม ๆ เพื่อแทนค่ากลับเป็นตัวอักษรต่าง ๆ ที่รับเข้ามา อย่างเช่นกรณีข้อมูลที่ส่งผ่านมาอยู่ในรูปของรหัสแอสกี้ (ASCII) ตัวอักษรแต่ละตัวจะถูกเข้ารหัสในรูป 8 บิทแทนหนึ่งตัวอักษรโดยมีบิทแรกเป็นบิทตรวจสอบ ดังนั้น อุปกรณ์รับข้อมูลจะตัดบิทออกเป็นกลุ่มละ 8 บิทเพื่อนำมาตีความเป็นอักษรแต่ละตัวนั่นเอง

สำหรับวิธีการที่จะระบุลงไปได้ว่า บิทใดเป็นบิทแรกของตัวอักษรตัวแรกนั้น มีวิธีการดังที่ได้กล่าวมาแล้วว่า ข้อมูลที่ถูกส่งผ่านโดยวิธีการซิงโครนัสจะถูกจับมารวมกันเป็นกลุ่มของข้อมูล (Block of Data) และที่ส่วนต้นของบล๊อคเราใส่ตัวอักขระ SYN ซึ่งเป็นอักขระพิเศษใช้ในการควบคุมการส่งผ่านข้อมูลโดยที่อักขระซิง (SYN Character) มีรูปแบบของบิทคือ 00010110

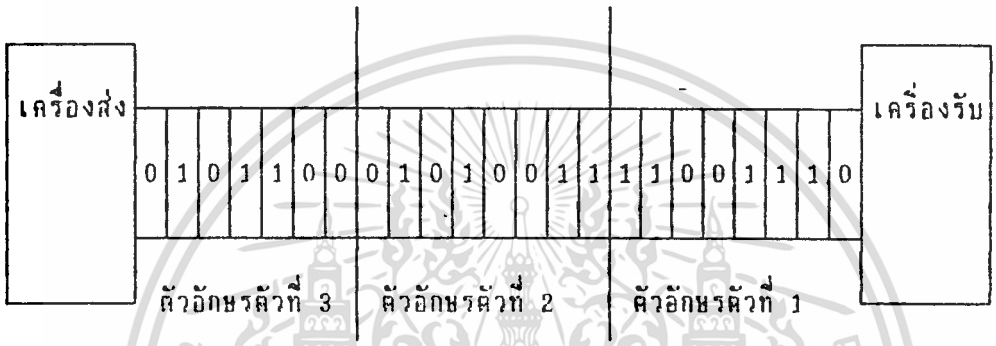
มีบิทตรวจสอบแบบเลขคี่ (Odd Parity) และอุปกรณ์รับข้อมูลจะคอยตรวจดูขบวนบิทที่วิ่งมามีส่วนใดตรงกับอักขระซิงบ้าง เมื่อพบอักขระซิงแล้วอุปกรณ์รับข้อมูลจะทราบได้ทันทีว่าถึงจุดเริ่มต้นที่จะจัดกลุ่มของบิทกลุ่มละ 8 บิท เพื่อแทนตัวอักษรได้ และตัวอักษรหลาย ๆ ตัวที่ดีความได้ก็คือข้อความที่ส่งมา ในแต่ละบล็อคแค่การใช้ตัวอักขระซิงเพียงตัวเดียวใส่ไว้ที่ส่วนต้นของบล็อคยังเป็นวิธีการที่ไม่ถูกต้อง เพราะในบางกรณีขบวนของบิทที่แทนตัวอักษรมีบางช่วงที่ไปตรงกับรูปแบบของบิทของอักขระซิงได้ ถ้าพิจารณาจากรูปจะเห็นว่า ถ้าส่งข้อความที่มีตัวอักษร B และ A ติดกัน 4 บิทท้ายของตัวอักษร B ต่อกับ 4 บิทแรกของตัวอักษร B ตรงกับอักขระซิงพอดี จะทำให้อุปกรณ์รับข้อมูลตีความผิดได้ ดังนั้น วิธีการแก้ไขข้อผิดพลาดที่จะเกิดได้จากกรณีเช่นนี้ โดยการใช้อักขระซิง (SYN Character) 2 ตัว ใส่ไว้ที่ส่วนต้นของบล็อคและอุปกรณ์รับข้อมูลด้วยก็ต้องทราบข้อตกลงนี้เป็นอย่างดี โดยทันทีที่ตรวจพบอักขระซิงจะดูอีก 8 บิทถัดไปว่าเป็นอักขระซิงด้วยหรือไม่ ถ้าใช่จะเริ่มต้นรับว่าทุก ๆ บิทที่ตามมาคือ ตัวอักษรแต่ละตัว กรณีไม่ใช่ก็จะเริ่มตรวจหาอักขระตัวต่อไป หรือกล่าวได้ว่าเครื่อง ๆ จะปรับตัวเข้าสู่ภาคการค้นหาอักขระซิง (Look For Sync Mode) และเมื่อพลอักขระซิงอย่างน้อย 2 ตัวก็จะเริ่มเข้าสู่ขบวนการจัดกลุ่มบิทกลุ่มละ 8 บิทแทนตัวอักษรหรือข้อมูลที่ได้รับ ในบางระบบการใช้อักขระซิง น้ำหนักกลุ่มข้อมูลอาจใช้อักขระซิงถึง 3-4 ตัวก็ได้ เพื่อความแน่นอนในการส่งผ่านข้อมูลแบบซิงโครนัสที่สมบูรณ์แบบยิ่งขึ้น

ถ้าจะกล่าวถึง การส่งผ่านข้อมูลแบบซิงโครนัสนั้นมีอยู่ด้วยกันหลายหัวเรื่องที่จะต้องกล่าวถึง ในบางกรณีจะใช้ตัวอักษรที่แตกต่างกัน 2 ตัว เป็นรูปแบบการส่งแบบซิงโครนัส ดังนั้นเครื่องข้อมูลจะต้องจำลักษณะของตัวอักษรตัวแรก ซึ่งมีลักษณะเป็นเอกลักษณ์ไว้หนึ่งตัวก่อนจากนั้นจะต้องเตรียมตรวจดูกลุ่มของบิทกลุ่มต่อไป เพื่อให้รู้ว่าเป็นตัวอักษรอะไร ในระบบการส่งผ่านข้อมูลที่เป็นบิทล้วน ๆ แฟล็ค (Flag) ความหมายก็คือ มีสัญญาณหรือไม่มีสัญญาณเช่นเดียวกับการใช้สัญญาณการโบกธง แต่ในทุกกรณีจะเห็นได้ว่า หลักของการส่งสัญญาณแบบซิงโครนัสนั้น จะต้องให้เครื่องรับข้อมูลที่ปลายทางเข้าใจถึงลักษณะของบิทพิเศษที่ส่งมาเพื่อให้รู้ว่านั่นคือ จุดเริ่มต้นของกลุ่มตัวอักษร (Block) ที่กำลังส่งเรียงกันเข้ามา

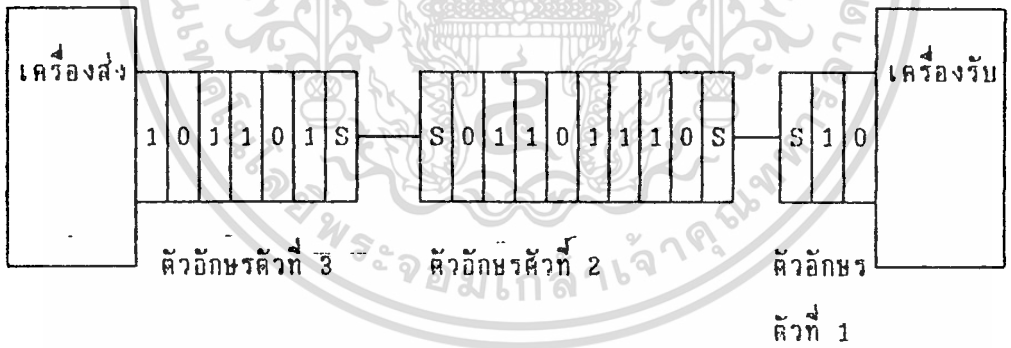
3.7 ประสิทธิภาพของการส่งผ่านข้อมูล (Efficiency of Transmission)

จากรูป 3.5 เป็นการเปรียบเทียบประสิทธิภาพการส่งผ่านข้อมูลแบบซิงโครนัส กับอะซิงโครนัส โดยที่รูป 3.5 (A) แสดงให้เห็นว่าการส่งผ่านข้อมูบแบบซิงโครนัสกับอะซิงโครนัสส่วนมาแล้วตลอดทางของสายส่ง จะใช้ส่งผ่านข้อมูลเต็มตลอดทั้งสายส่วนรูป 3.5 (B) แสดงให้เห็น

แต่ละส่วนของการส่งผ่านตัวอักษรแต่ละตัว และระหว่างตัวอักษรที่ส่งผ่านไปก็จะเสียช่องว่างระหว่างตัวอักษรไปด้วยในกรณีที่ส่งผ่านข้อมูลที่อยู่ในรหัสของแอสกี้ องค์ประกอบของตัวอักษรก็คือ 8 บิท ถ้ารวมสัญญาณเริ่มและสัญญาณหยุดไปด้วยก็จะรวมเป็น 10 บิท และสมมติว่าไม่มีช่องว่างระหว่างการส่งตัวอักษรแต่ละตัวเลย (คือว่าระยะเวลาสิ้นสุดตัวอักษรตัวแรกกับจุดเริ่มต้นตัวอักษรที่ 2 อยู่ติดกันหรือเวลาเสียไป = 0) ประสิทธิภาพในการส่งผ่านข้อมูลแบบอะซิงโครนัส ก็จะมีได้สูงสุดเพียง 80% ข้อเปรียบเทียบนี้ไม่รวมถึงการส่งผ่านข้อมูลแบบซิงโครนัส



(A) การส่งผ่านข้อมูลแบบซิงโครนัส



(B) การส่งผ่านข้อมูลแบบอะซิงโครนัส

240 ตัวอักษร ด้วยการใช้รหัสแอสกี้ ในบางกรณีของการส่งผ่านแบบซิงโครนัสโดยถือว่าตัวอักษรนี้จัดเข้าเป็นขบวนหรือกลุ่มเดียวกัน (Block) และหน้าขบวนมีรหัส SYS สามตัวอักษรเป็นตัวนำ ดังนั้น องค์ประกอบของขบวนข้อมูลนี้ทั้งขบวนจะประกอบด้วย

240 ตัวอักษร x 8 บิท/ตัวอักษร = 1920 บิท

SYS 3 ตัว เท่ากับ 3 x 8 บิท = 24 บิท

ผลรวมของบิทที่ต้องส่งทั้งหมด = 1944 บิท

ดังนั้น อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง ๆ กับจำนวนบิทที่จำเป็นต้องส่งคือ

1920/1944 = 99%

จากที่ทราบแล้วว่า ค่าประสิทธิภาพสูงสุดของการส่งข้อมูลแบบอะซิงโครนัสคือ 80% ฉะนั้นในกรณีนี้การส่งผ่านข้อมูลแบบซิงโครนัส จะมีประสิทธิภาพสูงกว่าแบบอะซิงโครนัสอยู่ 19% (และในทางปฏิบัติแล้วจะมากกว่า 19% อีก เพราะการส่งแบบอะซิงโครนัส ยังมีช่องว่างระหว่างการส่งผ่านตัวอักษร แต่ละตัวรวมอยู่ด้วย) ในกรณีที่ใช้สัญญาณหยุด (Stop Pulse) กินสองบิท ประสิทธิภาพของการส่งผ่านข้อมูลแบบอะซิงโครนัสก็ย่อมจะลดลงไปกว่าเดิมอีก

โดยทั่วไปแล้วการส่งผ่านข้อมูลแบบซิงโครนัสจะใช้ช่องทางการสื่อสาร (Communication Channel) มีประสิทธิภาพสูงกว่าการใช้ช่องทางการสื่อสารของการส่งผ่านข้อมูลแบบอะซิงโครนัส สมมติว่าช่องทางการสื่อสารที่ใช้อยู่สามารถส่งผ่านข้อมูลได้ 4800 บิท/วินาที ซึ่งจะทำให้การส่งผ่านข้อมูลแบบซิงโครนัสส่งผ่านตัวอักษรในรูปแบบของแอสกี้ได้ 600 ตัวอักษร/วินาที ในระบบอะซิงโครนัส (โดยสมมติว่าใช้ Stop Pulse เพียง 1 บิท)

ส่วนข้อดีของระบบอะซิงโครนัสก็คือ เสียค่าใช้จ่ายน้อยและอุปกรณ์ที่ใช้ก็ไม่จำเป็นต้องใช้อุปกรณ์ที่มีความละเอียดอ่อนสูงนัก สามารถใช้ได้กับการส่งผ่านด้วยความเร็วต่ำ ตัวอย่างเช่น ใช้ในกรณีผู้ใช้เครื่อง (USER) หรือผู้ควบคุมเครื่อง (OPERATOR) พิมพ์ข้อความส่งผ่านไปตามสายส่งข้อมูล เพราะอัตราการส่งผ่านข้อมูลจะขึ้นอยู่กับความเร็วในการพิมพ์ของผู้ใช้ด้วย และช่องว่างระหว่างตัวอักษรก็ไม่แน่นอนอีก เนื่องจากคนเป็นผู้พิมพ์แต่ในหลาย ๆ กรณีบางระบบมีการกำหนดเวลาห่างระหว่างการส่งตัวอักษรแต่ละตัวไว้ว่าจะมีเวลาห่างได้สูงสุดเท่าใด โดยทั่ว ๆ ไปแล้ว ถ้าสร้างอุปกรณ์รับส่งข้อมูลแบบอะซิงโครนัสย่อมเสียค่าใช้จ่ายน้อยกว่าอุปกรณ์รับส่งข้อมูลแบบซิงโครนัส เพราะอย่างน้อยอุปกรณ์ในประเภทหลังนี้ต้องใช้หน่วยความจำเพื่อรวมข้อมูลให้ได้เป็นกลุ่มพอสมควร แล้วจึงส่งผ่านข้อมูลแบบซิงโครนัส หน่วยความจำประเภทนี้เรียกว่า บัฟเฟอร์ (Buffer) แต่ความแตกต่างในด้านค่าใช้จ่ายจะลดลง เมื่อความเจริญทางเทคโนโลยีสูงขึ้น (เช่นทำให้ราคาของหน่วยความจำที่นำมาทำบัฟเฟอร์ถูกลงแต่ประสิทธิภาพสูงขึ้น) ในไม่ช้าทั้งสองแบบอาจมีราคาเท่ากันก็ได้

3.8 มาตราฐาน RS 232C

มาตรฐานการส่งข้อมูลมีด้วยกันหลายระบบหลายรูปแบบเช่นมาตรฐาน RS-232C RS-423 หรือ RS-449 เป็นต้น แต่ที่นิยมมากในปัจจุบันคือมาตรฐาน RS-232C

แนวโน้มที่ทำให้มาตรฐาน RS-232C ได้รับความนิยมอย่างมากมายและคำสั่งจะเป็นเส้นทางสำคัญของการพัฒนาในระดับไมโครคอนโทรลโลอร์ด้วย ซึ่งพอจะรวมตามจุดเด่นเป็นข้อ ๆ ได้ดังนี้

1. ความเรียบง่าย ซึ่งหมายถึงการใช้จำนวนสายเพียง 3 เส้นเท่านั้น (Local Connecting) ก็สามารถสื่อสารได้แบบ Full Duplex ถึงแม้จะกล่าวกันว่าการส่งแบบอนุกรมช้ากว่าการส่งแบบขนาน ซึ่งก็เป็นจริงตามหลักการแต่ความช้าของอนุกรมก็ยังเร็วมากในแง่การใช้งานจริง ความเรียบง่ายของจำนวนสายของ RS-232 นับว่าเป็นจุดเด่นซึ่งจะทำให้ทุกอย่างที่ตามมาเรียบง่ายไปด้วย ไม่ว่าจะเป็นขั้วต่อ ราคาของสายและยังต่อได้ไกลอย่างเพียงพออีกด้วย

2. การสื่อสารแบบ Full Duplex คือความสามารถในการสื่อสารแบบโต้ตอบกันได้ในเวลาเดียวกันซึ่งลักษณะนี้จะตอบสนองการทำงานในแบบที่เรียกกันว่า Interactive (เช่นการใช้ภาษา Basic-52 ของ 8052 นั่นเอง) อีกทั้งยังรองรับการใช้งานอื่น ๆ ในทำนองเดียวกัน และเนื่องจากความเป็นอนุกรมนี่เองที่ทำให้การสื่อสารเกิดช่วงเวลาว่างระหว่าง Byte ซึ่งเวลาว่างที่จุดนี้เป็นจุดที่ทำให้เราสามารถใช้ CPU ได้เต็มประสิทธิภาพมากขึ้น เพื่อให้เกิดการทำงานอย่างอื่นได้

3. ความยืดหยุ่นสูง เนื่องจาก RS-232 สามารถกำหนด Parameter ต่าง ๆ ในการสื่อสารได้เช่น Buad Rate, Parity, Data Bit และตัวอื่น ๆ อีก จึงทำให้เกิดความยืดหยุ่นสูงเช่นสมมติเกิดปัญหาเรื่องความยาวของสายเราอาจสามารถลด Buad Rate ลงได้เพื่อให้ได้ความยาวมากขึ้น หรือเช่นถ้าข้อมูลที่ส่งเป็นข้อมูลแบบ 7 บิท เราก็สามารถลด Data Bit ให้เป็น 7 บิท เพื่อประหยัดเวลาได้

RS-232C เป็นมาตรฐานเผยแพร่โดยสมาคมอุตสาหกรรมอิเล็กทรอนิคส์ (Electronic Industry Association, EIA) ซึ่ง RS ย่อมาจาก "Recommend Standard" และชื่อที่เป็นทางการของมาตรฐานคือ "Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange" หรือเรียกสั้น ๆ ว่า RS-232 การส่งข้อมูลมาตรฐาน RS-232 จะเป็นการส่งข้อมูลอนุกรมแบบ Asychronous ดังได้อธิบายมาแล้วในตอนต้น

3.8.1 ข้อกำหนดบางประการเกี่ยวกับ RS 232C

1. สถานะ (State)

- Mark หรือ "1" เมื่อแรงดันเกิน 3 V แต่ไม่เกิน 12 V
- Space หรือ "0" เมื่อแรงดันต่ำกว่า -3 V แต่ไม่เกิน -12 V

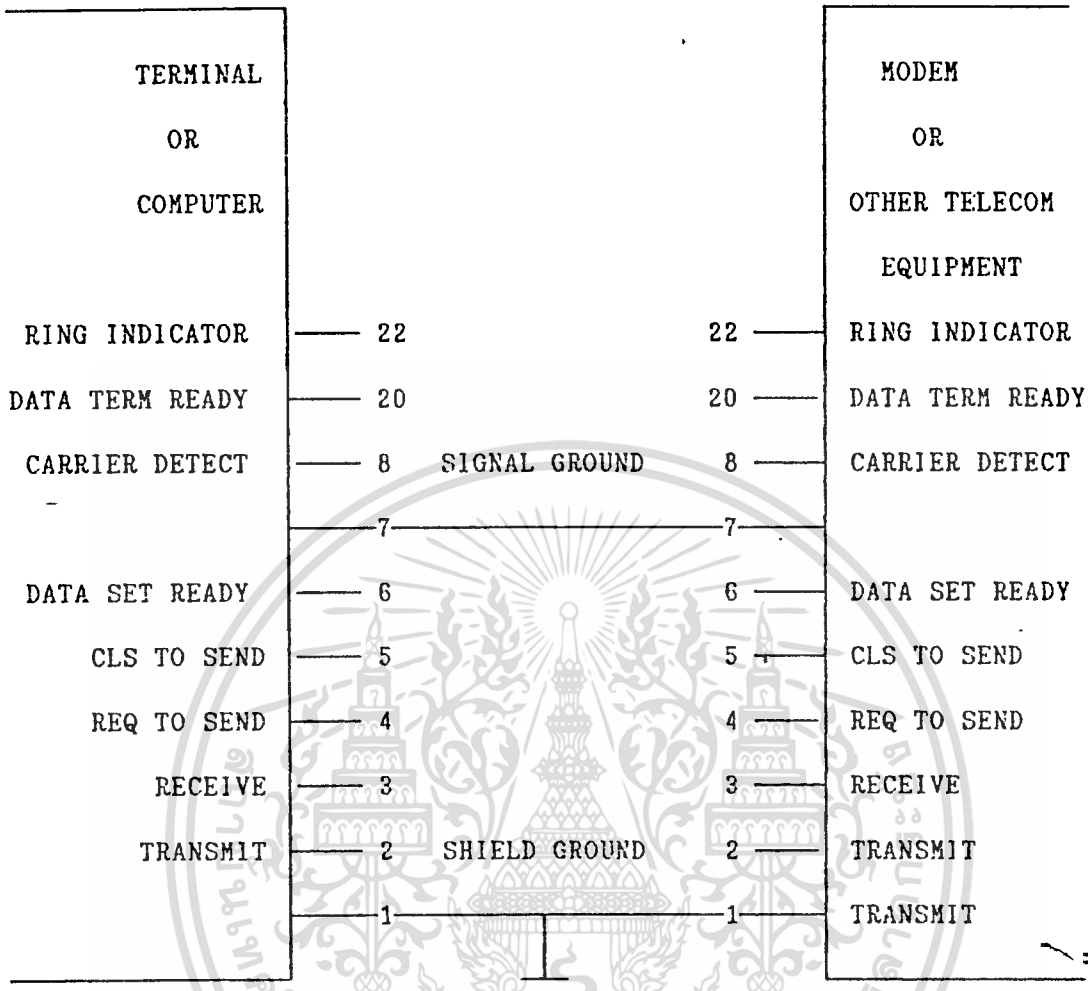
ดังนั้นแรงดันในช่วง -3 V ถึง 3 V เป็นแรงดันช่วงเปลี่ยนสถานะ และแรงดันสูงสุดในวงจร DTE ต้องไม่เกิน 25 V

2. วงจรขับสถานะ (Driver Circuit)

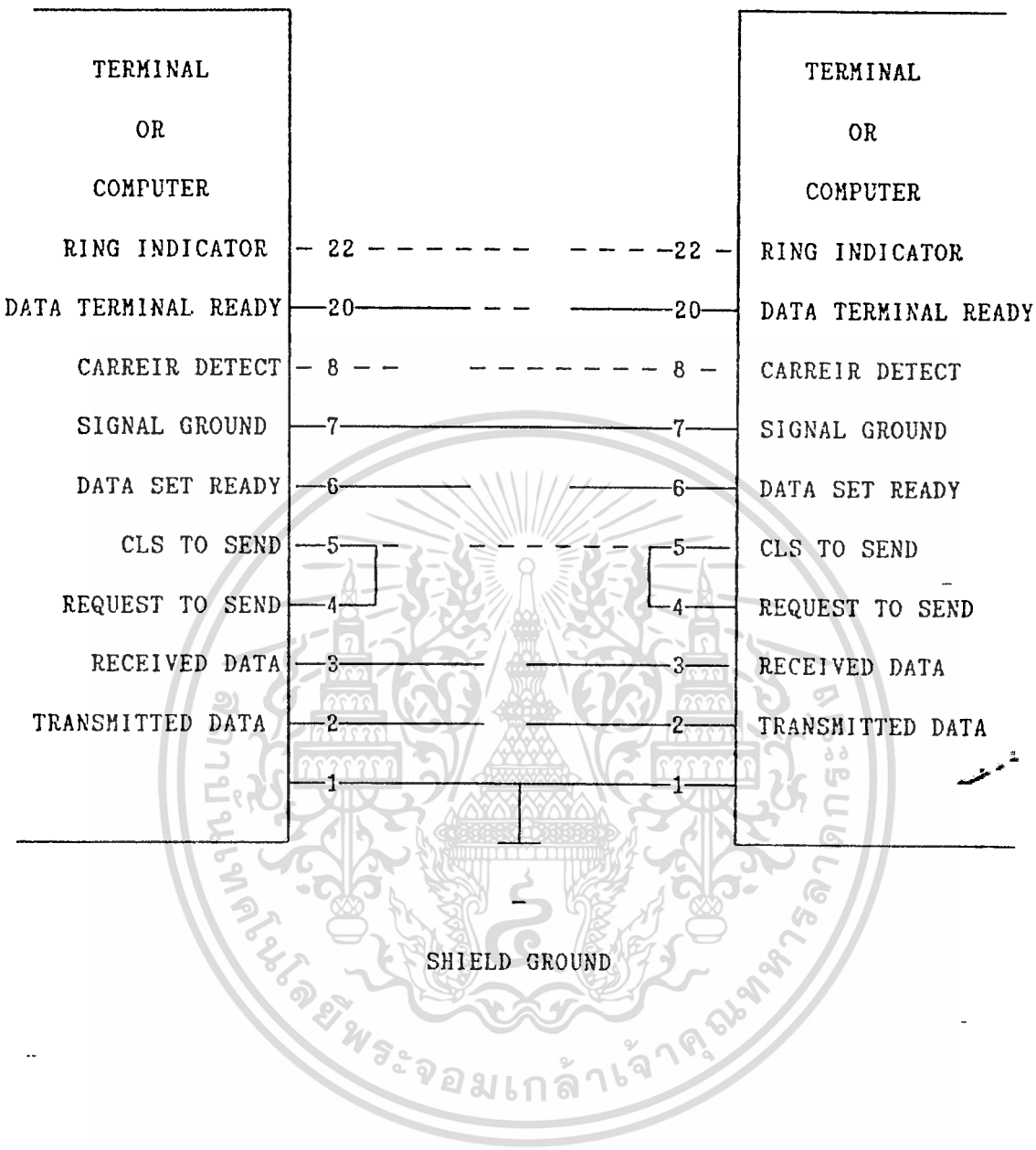
จะออกแบบให้เป็นแบบ Current Source หากเกิดสัญญาณลัดวงจรของสายเชื่อมต่อคู่ใด ๆ จะต้องมีกระแสลัดวงจรไม่เกิน 0.5 A และในขณะที่เปิดวงจรหรือขณะถอด Connecter ออก แรงดันในวงจรจะไม่เกิน 25 V

3. ความต้านทาน (Resisrance)

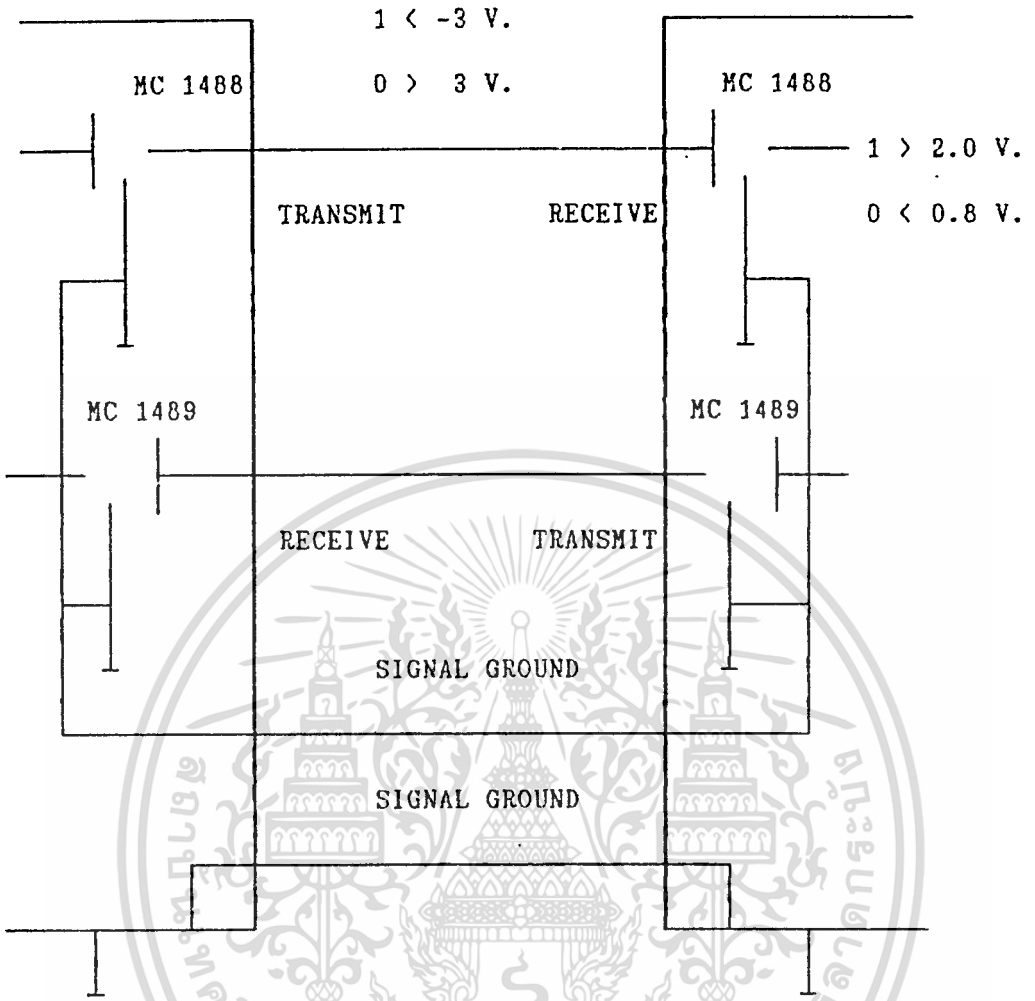
เมื่อมองจากด้านจุดอินเตอร์เฟสออกไปต้องอยู่ในช่วง 3,000 - 7,000 Ohm และ Capacitance จะต้องไม่เกิน 2,500 pF



รูปที่ 3.6 RS-232-C Interface with communications equipment.



รูปที่ 3.7 RS-232-C interface, terminal/computer to terminal/computer



รูปที่ 3.8 RS-232-C interface showing gates for changing voltage levels.

PIN #	SIGNAL NAME
1	SHIELDED GROUND
2	TXD (TRANSMIT DATA)
3	RXD (RECEIVE DATA)
4	RTS (REQUEST TO SEND)
5	CTS (CLS TO SEND)
6	DRS (DATA SET READY)
7	SG (SIGNAL GROUND)
8	DCD (DATA CARREIR DETECT)
18	NOT CONNECTED
20	DTR (DATA TERMINAL READY)
22	RI (RING INDICATOR)
25	NOT CONNECTED

รูปที่ 3.9

3.8.2 ขาที่ใช้งาน (ดูรูปที่ 3.9)

ขา 1 Protective Ground, PGND เป็นกราวด์ของตัวเครื่อง

ขา 2 Transmit Data, TXD เป็นเอาท์พุทของสัญญาณที่จะส่ง

ขา 3 Teceived Data, RXD เป็นอินพุทของสัญญาณที่รับเข้ามา

ขา 4 Request to Send, RTS เป็นขาเอาท์พุทที่ออกจาก DTE ไป DCE

On หมายถึง Transmitting Mode

Off หมายถึง Receiving Mode

ขา 5 Clear to Send, CTS เป็นขาอินพุทของ DCE หรือโมเด็ม หมายความว่า ข้อมูลถูกส่งออกไป

ขา 6 Data Set Ready, DSR เป็นอินพุทจาก DCE หรือโมเด็ม แสดงการพร้อมที่จะรับข้อมูล

ขา 8 Data Carrier Detecter, DCD เป็นสัญญาณจาก DCE ไป DTE

ขา 20 Data Terminal Ready, DTR เป็นขาเอาท์พุทจาก DTE ไปยัง DCE ให้เตรียมเชื่อมต่อหรือรักษาช่องทางติดต่อไว้

ขา 22 Ring Indicater, RI จาก DCE ไปให้ยัง DTE เมื่อรับสัญญาณเรียกของโทรศัพท์ว่ามีสัญญาณเข้ามาให้ต่อสายเข้ากับการสื่อสาร

3.9 รหัสแอสกี้ (ASCII Code)

รหัสที่ใช้ในการส่งสัญญาณ (Transmission Code) จะมีอยู่ด้วยกันหลายรหัสแต่รหัสที่ใช้ในการสื่อสารข้อมูลทั้งหมดจะอยู่ในรูปแบบของระบบไบนารีตัวอย่างรหัสเช่นรหัสบอโด (BAUDO Code)รหัสแอสกี้ (ASCII Code) รหัสบีซีดี (BCD Code) หรือรหัสเอ็บซีดิค (EBCDIC Code) ซึ่งรหัสแอสกี้นับว่าเป็นรหัสที่ใช้กันมากในการส่งสัญญาณของข้อมูล ฉะนั้นเราจะกล่าวเน้นที่การส่งข้อมูลในรหัสแอสกี้เท่านั้น

คำว่า ASCII ย่อมาจากคำเต็มว่า American Standard Code for Information Interchange หรืออีกนัยหนึ่งก็คือ รหัสมาตรฐานอเมริกันที่ใช้ในการส่งข่าวสารซึ่งเป็นรหัสขนาด 8 บิท โดยใช้ 7 บิทเข้ารหัสแทนตัวอักษรและอีก 1 บิท เป็นบิทที่คอยตรวจสอบ (PARITY BIT Check) และถือว่าเป็นรหัสแบบหนึ่งที่ใช้ในการสื่อสารข้อมูลกันอย่างกว้างขวาง และได้รับ

มาตรฐานของ CCITT โดยถือว่าเป็นมาตรฐานของ CCITT (The Consultative Committee on International Telephones and Telegraphs) หมายเลข 5 (CCITT Alphabet No.5) ในบางครั้งก็เรียกว่า "International Standards Organization หรือ ISO" ก็มี และกลุ่มอักขระที่สร้างขึ้นด้วยรหัสแอสกี้ก็มีชื่อเต็มเรียกว่า "ISO Seven Bit Code Character Set for Information Processing Interchange" นอกจากนี้ยังมีส่วนที่ให้แต่ละประเทศเลือกกำหนดใช้รหัสบางส่วนได้ ด้วยตามความเหมาะสมของประเทศนั้น ๆ ตัวอย่างเช่น ในประเทศอังกฤษก็จะกำหนดรหัสที่เป็นอักขระพิเศษซึ่งอังกฤษไม่ใช้แล้วเอาเครื่องหมายเงินปอนด์ไปใส่แทนก็ได้

								0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
b7	b6	b5	b4	b3	b2	b1	b0	0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	(TC$_7$) DLE	SP	0	@	P	.	p
			0	0	0	1	1	(TC$_1$)SOH	DC$_1$	!	1	A	Q	a	q
			0	0	1	0	2	(TC$_2$)STX	DC$_2$	"	2	B	R	b	r
			0	0	1	1	3	(TC$_3$)ETX	DC$_3$	#	3	C	S	c	s

(ต่อ)

0	1	0	0	4	(TC_4)EQT	DC_4	$	4	D	T	d	t
0	1	0	1	5	(TC_5)ENQ	(TC_8)NAK	%	5	E	U	e	u
0	1	1	0	6	(TC_6)ACK	(TC_9)SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	(TC_{10})ETB	,	7	G	W	g	w
1	0	0	0	8	FE_0 (HT)	CAN	(	8	H	X	h	x
1	0	0	1	9	FE_1 (LF)	EM	)	9	I	Y	i	y
1	0	1	0	10	FE_2 (VT)	SUB	.	:	J	Z	j	z
1	0	1	1	11	FE_3 (FF)	ESC	+	;	K	[	k	|
1	1	0	0	12	FE_4 (CR)	IS_4 (FS)	,	<	L	/	l	:
1	1	0	1	13	FE_5	IS_3 (GS)	-	=	M	]	m	|
1	1	1	0	14	SO	IS_2 (RS)	.	>	N		n	
1	1	1	1	15	SI	IS_1 (GS)	/	?	C	-	o	DEL

รูปที่ 3.10 ตารางแสดงรหัสแอสกีตามมาตรฐาน CCITT No.5

เนื่องจากรหัสแอสกี้เป็นรหัส 7 บิท แทน 1 ตัวอักษร จึงสามารถสร้างตัวอักษรได้ถึง 128 ตัว (เท่ากับ 2 ยกกำลัง 7) ดังนั้น จึงช่วยให้เข้ารหัสได้ทั้งตัวอักษรตัวใหญ่และตัวอักษรตัวเล็กได้เพียงพอและยังใส่ลักษณะของกราฟฟิค และตัวอักษรพิเศษที่ใช้ในการควบคุม (Control Character) เข้าไปด้วย ตัวอย่างของการเข้ารหัสแอสกี้ตามแบบของอเมริกัน ดังแสดงในรูป 3.10 ซึ่งประกอบด้วยกลุ่มตัวอักษร ซึ่งมี 8 แถวตามแนวดิ่ง และ 16 บรรทัดตามแนวนอนโดยบิทที่ 1 ถึง 4 อยู่ทางด้านซ้ายส่วนค่าของบิทที่ 5 ถึง 7 อยู่ด้านขวา ดังนั้น การดูจะต้องดูประกอบกันทั้งด้านซ้ายและด้านขวาตัวอย่างเช่น เราต้องการดูว่า W มีรหัสแอสกี้เป็นเช่นไร จะดูจากบิทที่ 4 ถึง 1 จะเห็นประกอบด้วย 0111 จากนั้นดูค่าบิทที่ 7 ถึง 5 จากส่วนขวาซึ่งประกอบด้วย 101 ดังนั้น W นะมีรหัสแอสกี้เป็น 1010111 จะสังเกตได้ว่าจะเรียงจากบิทสูงมาหาบิทต่ำ

สำหรับการอ้างอิงอักษรแต่ละตัวด้วยตารางนี้ สามารถทำได้ดังนี้ เช่น เราต้องการอ้างอิงถึงตัวอักษร R ซึ่งมีรหัสแอสกี้เป็นเลขฐาน 2 คือ 1010010 สามารถเขียนแทนด้วยการอ้างถึงตารางว่า 5/02 ซึ่งหมายถึงแถว (Column) ที่ 5 บรรทัด (Row) ที่ 2 เป็นต้น

จากรูป 3.10 จะเห็นได้ว่าส่วนที่อยู่ในแถว 0 และแถว 1 (Column 0 and 1) เป็นส่วนของตัวอักขระพิเศษที่ใช้ในการควบคุมเสียเป็นส่วนใหญ่ ซึ่งตัวอักขระพิเศษเหล่านี้ใช้ในการควบคุมการส่งผ่านข้อมูล (Control The Transmission of Data), ควบคุมรูปแบบของข้อมูล (Control The Format of Data) ควบคุมความสัมพันธ์ทางตรรกของข้อมูล และควบคุมความสัมพันธ์ทางกายภาพเครื่องรับปลายทาง และตั้งแต่แถวที่ 6 เป็นต้นไปเป็นส่วนที่เราจะนำเอาตัวอักขระที่เราจำเป็นต้องใช้ในกิจการของเรา หรือของท้องถิ่นเข้ารหัสแทนเข้าไปในตารางได้ยกเว้นของ DEL ซึ่งย่อมาจาก Delete หรือ Rubout ซึ่งใช้ในการลงตัวอักษรซึ่งมีตำแหน่งในตารางคือ 7/15 ถือเป็นตัวอักขระพิเศษในการควบคุม ซึ่งเข้าเป็นรหัสมาตรฐานมาแล้ว

3.10 อักขระพิเศษใช้ในการควบคุม (Control Characters)

จากรูป 3.10 เราจะพบว่ามีอักขระพิเศษนี้อยู่ 32 ตัว โดยที่อักขระ 32 ตัวนี้ถูกแบ่งหน้าที่การทำงานออกเป็น 4 กลุ่ม โดยที่แต่ละกลุ่มมีหน้าที่ดังต่อไปนี้

Transmission Controls มีหน้าที่ควบคุมการเดินทางของข้อมูลและเดินทางไปตามสายส่งสัญญาณ

Format Effectors มีหน้าที่ควบคุมรูปแบบทางกายภาพของข้อมูล

Device Controls มีหน้าที่เป็นส่วนเริ่มต้นอุปกรณ์ช่วยของเครื่องรับส่งข้อมูลปลายทาง

Information Separators มีหน้าที่กำหนดส่วนต่าง ๆ ของข้อมูลเช่นข้อมูลที่ส่งมาตั้งแต่จุดไหนถึงไหนเป็นต้น

สำหรับ Transmission Controls จัดได้ว่าเป็นส่วนสำคัญอย่างยิ่ง ซึ่งต้องการอธิบายกันโดยละเอียด ดังนั้น จึงขออธิบายการควบคุมอื่น ๆ ทั้ง 3 แบบเสียก่อน เพื่อให้มองเห็นภาพพจน์อย่างคร่าว ๆ ก่อนแล้วจึงจะนำเอา Transmission Controls มาอธิบายโดยละเอียดอีกครั้งหนึ่ง Format Effectors หมายถึง ตัวอักขระพิเศษที่ควบคุมรูปแบบของข่าวสารซึ่งมีอยู่ด้วยกันทั้งหมด 6 ตัวคือ FE_0 - FE_5 (FE ย่อมาจาก Format Effector) ซึ่งมาจากตารางตามรูป 3.10 จะอยู่ในแถวที่สูงและบรรทัดที่ 8-13 อักขระตัวแรกได้แก่ FE0 ซึ่งตรงกับการใช้ปุ่ม Back Space (BS) บนแท่นพิมพ์ของเครื่องส่งข้อมูล ทันทีที่ปุ่มนี้ถูกกดหัวพิมพ์บนเครื่องพิมพ์หรือเคอร์เซอร์ ซึ่งเป็นไฟบ่งตำแหน่งบนจอภาพกรณีเครื่องส่งข้อมูลมีจอภาพจะถอยหลังกลับมา 1 ตำแหน่ง ตัวอักขระต่อไปได้แก่ FE_1 ซึ่งตรงกับการเลื่อนออกไปตามแนวนอน (Horizontal Tabulation หรือ HT) เมื่อใช้จะเกิดการเลื่อนหัวพิมพ์หรือเคอร์เซอร์ไปทางแนวนอนให้ตรงกับตำแหน่งที่ตั้งไว้ล่วงหน้า อักขระตัวต่อไปได้แก่ FE_2 ซึ่งตรงกับการเลื่อนขึ้นบรรทัดใหม่ (Line Feed หรือ LF) ทันทีที่ใช้อักขระนี้จะเลื่อนหัวพิมพ์หรือเคอร์เซอร์ไปยังบรรทัดถัดไปทันทีแต่ยังคงรักษาตำแหน่งการพิมพ์ (หลักของบรรทัด) ให้คงที่ไว้ตำแหน่งเดิม อักขระตัวต่อไปได้แก่ FE_3 ซึ่งตรงกับการเลื่อนตามแนวดิ่ง (Vertical Tabulation หรือ VT) เมื่อใช้จะทำให้พิมพ์ หรือเคอร์เซอร์เลื่อนข้ามไปกี่บรรทัดก็แล้วแต่เราตั้งไว้ล่วงหน้า แต่ยังคงรักษาการพิมพ์ตามแนวนอน (หลัก) ให้คงเดิมเหมือนก่อนก่อนเลื่อนแต่การใช้ VT นี้จะเลื่อนข้ามไปกี่บรรทัดก็ตาม จะเป็นการใช้งานในการพิมพ์ข่าวสาร ซึ่งยังอยู่ภายในหน้าเดียวกันอักขระตัวต่อไปคือ FE_4 ซึ่งตรงกับการเลื่อนขึ้นหน้าใหม่ (From Feed หรือ FF) เมื่อใช้หัวพิมพ์หรือเคอร์เซอร์จะเลื่อนขึ้นหน้าใหม่ของข่าวสารต่อไปโดยให้หัวพิมพ์หรือเคอร์เซอร์อยู่ตรงตำแหน่งการพิมพ์ตัวอักขระตัวแรกของบรรทัดที่จัดไว้เป็นบรรทัดแรกของหน้าทันที ส่วนอักขระตัวสุดท้าย คือ FE_5 จะตรงกับการปัดแคร่ (Carriage Return หรือ CR) จะทำให้หัวพิมพ์หรือเคอร์เซอร์เลื่อนกลับมายังตำแหน่งแรกของการพิมพ์ในบรรทัดที่กำลังพิมพ์อยู่

แต่มีอุปกรณ์รับส่งข้อมูลปลายทาง (Terminal) บางชนิดจะใช้อักขระพิเศษนี้ทำงาน 2 หน้าที่พร้อมกัน เช่น ในการควบคุมการเลื่อนขึ้นบรรทัดใหม่ (LF) อยู่ภายใต้อักขระควบคุมตัวเดียวกับการปัดแคร่ (CR) และเรียกหน้าที่การทำงานนี้เสียใหม่ว่า ขึ้นบรรทัดใหม่ (New Line หรือ NL) และส่วนมากให้อยู่ในภายใต้การควบคุมของ FE_c

จากการทำงานของอักขระควบคุมที่กล่าวมาแล้วนี้ จะเห็นได้ว่าทั้งหมดมีหน้าที่ในการจัดรูป

แบบการพิมพ์ข่าวสารที่จะส่งไปยังสถานีปลายทาง ไม่ว่าการส่งจะเป็นแบบเครื่องพิมพ์ธรรมดาหรือแบบมีจอภาพก็ตาม ทำให้รูปแบบข่าวสารวางตัวตามความต้องการของผู้ส่งได้

Device Control หมายถึง ตัวอักขระพิเศษที่ใช้ในการควบคุมอุปกรณ์ ซึ่งมีอยู่ด้วยกันทั้งหมด 4 ตัว คือ ตั้งแต่ DC_1 - DC_4 (DC ย่อมาจาก Device Control) อักขระพิเศษชุดนี้จะช่วยในการควบคุมอุปกรณ์ต่าง ๆ ณ อุปกรณ์รับส่งข้อมูลปลายทาง เช่น DC_1 จะตรงกับการสั่งให้สวิทช์บนเครื่องบันทึกเทปคลับเปิด กรณีเมื่อมีเครื่องบันทึกเทปคลับพ่วงติดอยู่กับอุปกรณ์รับส่งข้อมูลปลายทาง (Terminal) ส่วน DC_2 จะทำหน้าที่ตรงกันข้ามกับ DC_1 คือการสั่งปิดเครื่องบันทึกเทปคลับและอักขระพิเศษ DC_3 เมื่อใช้จะเอาข้อความทั้งหมดที่อยู่บนจอภาพขณะนั้นพิมพ์ลงยังเครื่องพิมพ์ในกรณีเทอร์มินอลของมีเครื่องพิมพ์ติดอยู่ด้วย สำหรับอักขระพิเศษตัวสุดท้ายคือ DC_4 เมื่อใช้จะทำการล็อคแป้นพิมพ์ที่พ่วงอยู่กับจอภาพ ทำให้ไม่สามารถป้อนข้อมูลผ่านแป้นพิมพ์ได้แต่การใช้อักขระพิเศษกลุ่มนี้อาจมีข้อแตกต่างกันไปบ้าง ขึ้นอยู่กับบริษัทผู้ผลิตเทอร์มินอล

Information Separators หมายถึง อักขระพิเศษที่ใช้แบ่งแยกข่าวสารออกเป็นสัดส่วนตามที่ต้องการมีอยู่ด้วยกัน 4 ตัวคือ IS_1-IS_4 (IS ย่อมาจาก Information Separator) เมื่อมีการใช้อักขระพิเศษกลุ่มนี้ขึ้นในข่าวสารที่ส่งเรียงต่อกันเป็นขบวนเข้าไปได้ แต่เครื่องคอมพิวเตอร์จะทำให้คอมพิวเตอร์ทราบได้ว่า จุดแบ่งแยกอยู่ตรงส่วนไหน เช่น แยกเป็นเร็คคอร์ด (Record) และเครื่องจะได้จัดเก็บข้อมูลเหล่านี้เป็นเร็คคอร์ด ๆ ไป อักขระพิเศษตัวแรกได้แก่ IS_1 ใช้เป็นตัวแบ่งแยกข่าวสารออกเป็นส่วนย่อย เช่น อาจใช้ในการแบ่งเร็คคอร์ดออกเป็นฟิลด์ (Field) ต่าง ๆ ก็ได้ ดังนั้น อักขระพิเศษตัวนี้จึงมีชื่อว่า ตัวแบ่งหน่วยย่อย (Unit Separator หรือ US) ลำดับถัดไปได้แก่ IS_2 ใช้เป็นตัวแบ่งแยกข่าวสารออกเป็นเร็คคอร์ด โดยที่ภายในเร็คคอร์ดหนึ่ง ๆ ก็จะประกอบด้วยหน่วยย่อยของข่าวสารหลายหน่วยและอักขระพิเศษนี้ได้รับชื่อว่า ตัวแบ่งแยกเร็คคอร์ด (Record Separator หรือ RS) อักขระตัวต่อไปได้แก่ IS_3 ซึ่งใช้เป็นตัวแบ่งข่าวสารออกเป็นกลุ่ม โดยที่ภายในกลุ่มของข่าวสารนี้จะประกอบด้วย ข่าวสารจำนวนหลายเร็คคอร์ดบางครั้งอาจเรียกว่าบล็อคก็ได้ ดังนั้น อักขระพิเศษตัวนี้จึงได้ชื่อว่า ตัวแบ่งกลุ่มข่าวสาร (Group Separator หรือ GS) ส่วนตัวอักขระพิเศษตัวสุดท้ายสำหรับชุดนี้ได้แก่ IS_4 ใช้เป็นตัวแบ่งแยกข่าวสารออกเป็นแฟ้มข้อมูล หรือไฟล์และเรียกอักขระพิเศษตัวนี้ว่า ตัวแบ่งไฟล์ (File Separator หรือ FS)

3.11 อักขระควบคุมการสื่อสาร

ANSI ได้ให้คำนิยามในการใช้อักขระควบคุมในการสื่อสาร เพื่อเป็นที่เข้าใจได้ตรงกันดังนี้

NUL

เป็นอักขระว่างคือ ไม่มีอะไรเลยอย่าสับสนกับ Blank เพราะ Blank ยังมีอักขระอยู่คือ " " ในภาษาเบสิกส่วน NUL = " " ในภาษาเบสิก NUL อาจจะแทรกหรือคิงออกจากสายต่อเนื่องของอักขระโดยไม่มีผลกระทบกระเทือน การแทรกหรือคิง NUL เข้าหรือออก เพียงเพื่อต้องการผลในแบบแปลนในการส่งข้อมูลเท่านั้น (เช่น ทำให้ได้ความยาวของข้อมูลตามต้องการ) อักขระนี้มีประโยชน์สำหรับอุปกรณ์เครื่องพิมพ์ซึ่งต้องการเวลาที่แน่นอน ในการเลื่อนหัวพิมพ์เครื่องพิมพ์บางตัวต้องการอักขระ NUL อย่างน้อยสองตัวตามหลัง CR (Carriage Return) เพื่อเป็นการตั้งหัวพิมพ์ให้เข้าทางซ้ายก่อนที่จะรับอักขระต่อไปได้

SOH

Start of Heading (SOH) ใช้ในการควบคุมในสายข้อมูลของ Bisyn ซึ่งหมายถึงจุดเริ่มต้นของส่วนนำหน้าของข้อมูล สถานีในย่ายการสื่อสารเราจะตรวจสอบอักขระที่ตามหลัง SOH สำหรับตีความว่าจะเป็นผู้รับข้อความที่ตามหาข้างล่างนี้หรือไม่ หรือง่าย ๆ ก็หมายความว่า "ฟังซิว่าชื่อเธอถูกเรียกหรือเปล่า"

SOH บางครั้งก็ใช้ในแบบอะซิงโครนัส สำหรับการถ่ายโอนข้อมูลหลาย ๆ แฟ้มติดต่อกันโดยไม่แยกแฟ้มส่ง SOH ใช้สำหรับจุดเริ่มต้นของแต่ละแฟ้ม ในการสื่อสารแบบอะซิงโครนัส มีผู้รับอยู่สถานีเดียวจึงไม่จำเป็นต้องใช้ตำแหน่งที่อยู่ของสถานีรับ เพียงแต่ใช้ชื่อแฟ้มตามหลังเท่านั้น การรับส่งแบบนี้ใช้เฉพาะในโปรแกรมสำเร็จรูปที่เหมือนกันเท่านั้น ยังไม่ได้เป็นมาตรฐานของการสื่อสาร

SOH ใช้ในโปรแกรมโปรโตคอล Xmodem เพื่อเป็นการบอกให้รู้ว่า กลุ่มของข้อมูลอีก 128 ไบต์ กำลังตามมา

STX

Start of Text (STX) ใช้ใน Bisyn เป็นการเตือนให้ฝ่ายรับได้รู้ว่าข้างหลังที่ตามมาเป็นข้อความที่ต้องการจะส่งให้ และเป็นการบอกให้รู้ว่าส่วนหัวได้สิ้นสุดลงแล้ว

ETX

End of Text (ETX) ใช้ใน Bisyn เหมือนกัน เป็นการบอกฝ่ายรับว่าข้อความที่ต้องการส่งในกลุ่มนี้หมดแล้ว

EOT

End of Transmission (EOT) เป็นอักขระควบคุมการสื่อสารเพื่อเป็นการบอกให้รู้ว่าสิ้นสุดการส่งข้อมูลไปยังอุปกรณ์ที่บ่งบอกแล้ว อักขระนี้ยังเป็นการบอกให้อุปกรณ์อื่น ๆ ที่อยู่ในย่าย

สื่อสาร คอยตรวจสอบว่าจะมีข้อความส่งมาถึงตัวเองหรือเปล่า EOT เป็นตัวชี้ถึงจุดสุดท้ายของเฟรมที่เริ่มต้นโดยอักขระ SOH ในโปรโตคอล Xmodem ใช้อักขระตัวนี้คอยบอกการสิ้นสุดของการโอนแฟ้มข้อมูล

ETB

End of Transmission Block (ETB) เป็นอักขระควบคุมการสื่อสารเป็นตัวบ่งชี้ถึงการสิ้นสุดกลุ่มข้อมูลที่ส่ง Bisyn ใช้อักขระตัวนี้แทน ETX เมื่อข้อมูลที่ส่งมากกว่า 2 กลุ่ม

ENQ

Enquiry (ENQ) อักขระควบคุมตัวนี้ใช้ในการเรียกการตอบสนองจากฝ่ายรับ อาจจะใช้เป็นการเรียกใช้อุปกรณ์ในข่ายการสื่อสารแสดงตัวเอง หรืออาจจะใช้ในการบอกสถานภาพของการสื่อสาร โปรแกรสำเร็จรูปที่ใช้กับ IBM PC บางตัวใช้อักขระตัวนี้ในโปรโตคอลการส่งแฟ้มข้อมูลเพื่อให้แสดงการตอบสนองด้วย ENQ ฝ่ายรับอาจจะต้องการที่จะให้ส่งกลุ่มข้อมูลที่รับไปหลังสุด อย่างไรก็ตามนี้เป็นการใช้ ENQ อย่างไม่เป็นมาตรฐาน

ACK

Acknowledge (ACK) อักขระควบคุมใช้ในการบ่งบอกถึงการสื่อสาร และถ่ายโอนข้อมูลอย่างไม่ผิดพลาด ระหว่างสถานีรับและสถานีส่ง หลังจากรับกลุ่มของข้อมูลอย่างถูกต้องแล้ว ฝ่ายรับจะต้องส่ง ACK ให้ฝ่ายส่งได้รับรู้ว่าข้อความที่ส่งไปได้รับอย่างถูกต้องแล้ว

NAK

Negative Acknowledge (NAK) ใช้ระหว่างฝ่ายรับกับฝ่ายส่งบ่งบอกว่ามีการผิดพลาดเกิดขึ้นในสื่อสาร โดยทั่วไป NAK จะส่งโดยฝ่ายรับไปยังฝ่ายส่งเมื่อมีความผิดพลาดเกิดขึ้น และเป็นการร้องขอให้ฝ่ายส่ง ส่งข้อความเก่ามาใหม่ ENQ, ACK และ NAK ส่วนมากจะใช้ด้วยกันในโปรโตคอลการรับส่ง แต่ผู้ใช้จะไม่ยุ่งเกี่ยวกับอักขระเหล่านี้

BEL

BEL เป็นอักขระควบคุมที่ทำงานเหมือนชื่อ คือสั่นกระดิ่งเรียกร้องความสนใจจากผู้ใช้อาจจะรวมอยู่ในแฟ้มหรือรับส่งระหว่างการสื่อสารที่เป็นลักษณะสนทนาใน IBM PC อาจจะทำได้ทันทีโดยกด Ctrl และ G พร้อมกัน หรือในภาษาเบสิกใช้คำสั่ง Lprint CHR4$ (7)

BS

Back Space (BS) เป็นอักขระสำหรับจัดรูปร่างเพื่อควบคุมเครื่องพิมพ์และการแสดงผลบนหน้าจอ โดยจะมีผลทำให้เคอร์เซอร์หรือหัวพิมพ์เลื่อนไปทางซ้าย 1 ตำแหน่ง ในขณะที่เคอร์เซอร์ไม่ได้อยู่ในตำแหน่งที่ 1 อักขระที่อยู่ในตำแหน่งที่เคอร์เซอร์เลื่อนไปทับจะหายไปบางครั้ง

เราเรียกอักขระนี้ว่าอักขระถอยหลังและลบ

HT

Horizontal Tabulation (HT) เป็นอักขระสำหรับการจัดรูปร่าง ทำให้เครื่องพิมพ์และการแสดงผลบนหน้าจอ เลื่อนไปตำแหน่งที่ตั้งค่าไว้ก่อนแล้ว ก่อนที่จะพิมพ์อักขระตัวต่อไป

LF

Line Feed (LF) เป็นอักขระควบคุมทำให้ตำแหน่งที่พิมพ์เลื่อนไป 1 บรรทัดนับต่อไปในแถวตั้งอันเดียวกันอาจจะสับสนกัน Carriage Return (CR) แต่ว่า LF ไม่ได้เลื่อนหัวพิมพ์หรือเคอร์เซอร์ไปยังแถวตัวที่ 1 ของบรรทัดถัดไปนอกจากจะตามด้วย CR

VT

Vertical Tabultion (VT) เป็นอักขระควบคุมการจัดรูปร่าง ซึ่งจะทำให้เครื่องพิมพ์หรือการแสดงผลบนหน้าจอเลื่อนไปยังแถวตั้งเดิมแต่จำนวนบรรทัดห่างออกไปเท่ากับจำนวนบรรทัดที่ตั้งไว้ก่อนหน้านี้

FF

From Feed (FF) เป็นอักขระควบคุมรูปร่าง ใช้ในการเลื่อนหัวพิมพ์ไปยังส่วนบนของแบบฟอร์มที่ตั้งเอาไว้ของหน้าถัดไป พูดง่าย ๆ ก็คือการขึ้นหน้าใหม่ของเครื่องพิมพ์

CR

Carriage Return (CR) เป็นอักขระควบคุมการจัดรูปร่างซึ่งจะทำให้เครื่องพิมพ์หรือเคอร์เซอร์ของการแสดงผลบนหน้าจอเลื่อนไปยังตำแหน่งที่ 1 ของบรรทัดเดียวกัน

SO

Shift Out (SO) เป็นอักขระควบคุมพิเศษ ซึ่งทำหน้าที่ในการขยายอักขระมาตรฐานของ ASCII ออกไป สำหรับเครื่องพิมพ์อาจจะใช้รหัสควบคุมตัวนี้ทำให้เปลี่ยนโหมดการพิมพ์เป็นพิมพ์ตัวใหญ่หรือความหนาเป็นสองเท่า

SI

Shift In (SI) เป็นอักขระพิเศษซึ่งอาจจะใช้ในการตั้งค่าใหม่ของเครื่องรับไปยังอักขระมาตรฐานของ ASCII ในเครื่องพิมพ์อาจจะใช้อักขระตัวนี้ในการโคดกลับเข้ามาพิมพ์ตัวขนาดมาตรฐานหลังจากที่ใช้ SO พิมพ์ตัวหนาสองเท่า

DLE

Data Link Escape (DLE) เป็นอักขระควบคุมการสื่อสารใช้ในการดัดแปลงแก้ไขความหมายของอักขระที่ตามหลังมาใช้ใน Bisyn สำหรับหมายถึงการเริ่มต้นของข้อมูล Transparent

DC1

Device Control 1 (DC1) เป็นอักขระสำหรับการโยกสวิทช์ทางอิเล็กทรอนิกส์การใช้งานจริง ๆ อาจจะแตกต่างกันไปสำหรับบริษัทผู้ผลิตอุปกรณ์สื่อสาร หรือซอฟต์แวร์ของแต่ละบริษัท แต่โดยทั่วไปแล้วใช้ในการควบคุมอุปกรณ์ที่เกี่ยวโยงเอาไว้สำหรับการใช้แสดงผลอักขระตัวนี้มีค่าเท่ากับ Ctrl-Q จะทำให้เกิดการเริ่มต้นการแสดงผลใหม่หลังจากหยุดไปเพราะ DC3 หรือ Ctrl-S ในทางการสื่อสารข้อมูลอักขระตัวนี้ส่วนมากจะมอบหมายให้มีค่าเท่ากับ XON และใช้สำหรับการเริ่มถ่ายโอนใหม่หลังจากหยุดไปเพราะ XOFF

DC2

Device Control 2 (DC2) เช่นเดียวกับ DC1 ความหมายอาจจะแตกต่างกันออกไปแล้วแต่ผู้ผลิตจะนำไปใช้

DC3

Device Control 3 (DC3) เป็นอักขระสำหรับโยกสวิทช์อิเล็กทรอนิกส์อีกอันหนึ่งส่วนมากจะใช้ร่วมกับ DC1 เสมอ สำหรับ (Speed Matching) ปรับความเร็วให้เข้ากัน DC3 ก็คือ XOFF และใช้ในการหยุดส่งชั่วคราว

DC4

Device Control 4 (DC4) ก็เช่นเดียวกับ DC2 ซึ่งความหมายอาจจะแตกต่างกันออกไปแล้วแต่ผู้จะใช้

DYN

Synchrounous Idle (SYN) เป็นอักขระควบคุมการสื่อสารใช้ในโปรโตคอล Bisyn สำหรับเริ่มกับการสื่อสาร หรือเมื่อยังไม่มีข้อมูลจะส่ง สำหรับให้ฝ่ายรับปรับสัญญาณนาฬิกาให้เข้ากับฝ่ายส่งเสมอ

CAN

Cancel (CAN) เป็นอักขระควบคุมพิเศษอาจจะนำไปใช้งานต่าง ๆ กันสำหรับผู้ใช้ในแต่ละราย แต่ส่วนมากจะหมายถึงเกิดมีข้อผิดพลาดในการส่งข้อมูลอักขระตัวนี้เป็นตัวบ่งบอกว่าข้อมูลที่ได้รับเข้าก่อนหน้านี้ผิดพลาด (เนื่องจากฝ่ายส่งเอง) ก็ทิ้งไปได้

EM

End of Medium (EM) เป็นอักขระควบคุมพิเศษใช้ในการบ่งบอกการสิ้นสุดของตัวกลางในทางฟิสิกส์ (หน่วยบันทึกข้อมูล อันเป็นตัวแทนของตัวกลางการสื่อสาร) หรือสิ้นสุดส่วนของตัวกลางที่มีข้อมูลอยู่

SUB

Substitute(SUB) อักขระนี้ใช้ในการสื่อสารเพื่อควบคุมความแม่นยำ เครื่องรับใช้ SUB แทนอักขระที่ถูกตีความว่าผิดพลาด ไม่ถูกต้อง หรือเป็นไปไม่ได้ในการส่งให้เครื่องพิมพ์หน่วยแสดงผล

ESC

Escape (ESC) เป็นอักขระควบคุมที่ใช้กันอย่างแพร่หลายในการสื่อสารระหว่างเครื่องพิมพ์กับคอมพิวเตอร์ ปกติจะส่งออกไปก่อนที่จะตามด้วยอักขระหรือตัวเลขที่เป็นรหัสเสริมต่อหรืออักขระควบคุมเสริมต่อ ซึ่งเปลี่ยนแปลงไปตามผู้ผลิตเครื่องพิมพ์ที่อักขระที่เสริมต่อนั้นจะมีความหมายอย่างไร

FS

File Separator (FS) เป็นตัวรับข่าวสารที่ใช้แสดงถึงขอบเขตในทางลอจิกระหว่างแฟ้มข้อมูลที่ถูกถ่ายโอน

GS

Group Separator (GS) เป็นอักขระคั่นกลางที่ใช้แสดงขอบเขตในทางลอจิกของกลุ่มข้อมูลที่ส่งออกไป

RS

Record Separator (RS) เป็นอักขระที่คั่นกลางเช่นกันแต่คั่นกลางระหว่างเร็คคอร์ด

US

Unit Separator (US) ใช้ในการคั่นระหว่างข้อมูลที่แตกต่างกัน

DEL

Delete (DEL) ใช้ในการลบอักขระ สำหรับเครื่องพิมพ์ใช้ในการลบอักขระที่ไดรัตัวท้ายสุด ถ้าเป็นการลบอักขระใต้เคอร์เซอร์ออก

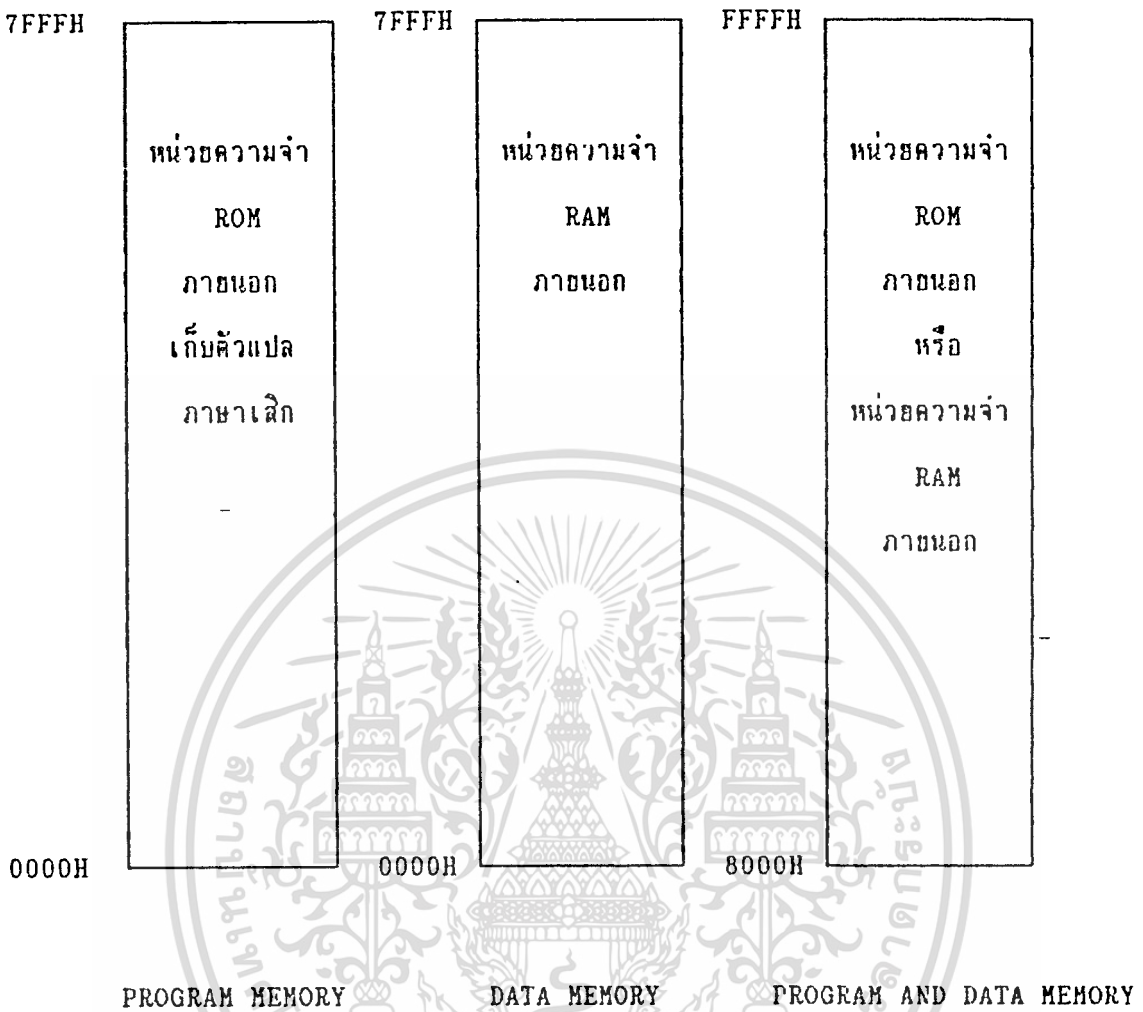
บทที่ 4

การจัดหน่วยความจำ

4.1 บทนำ

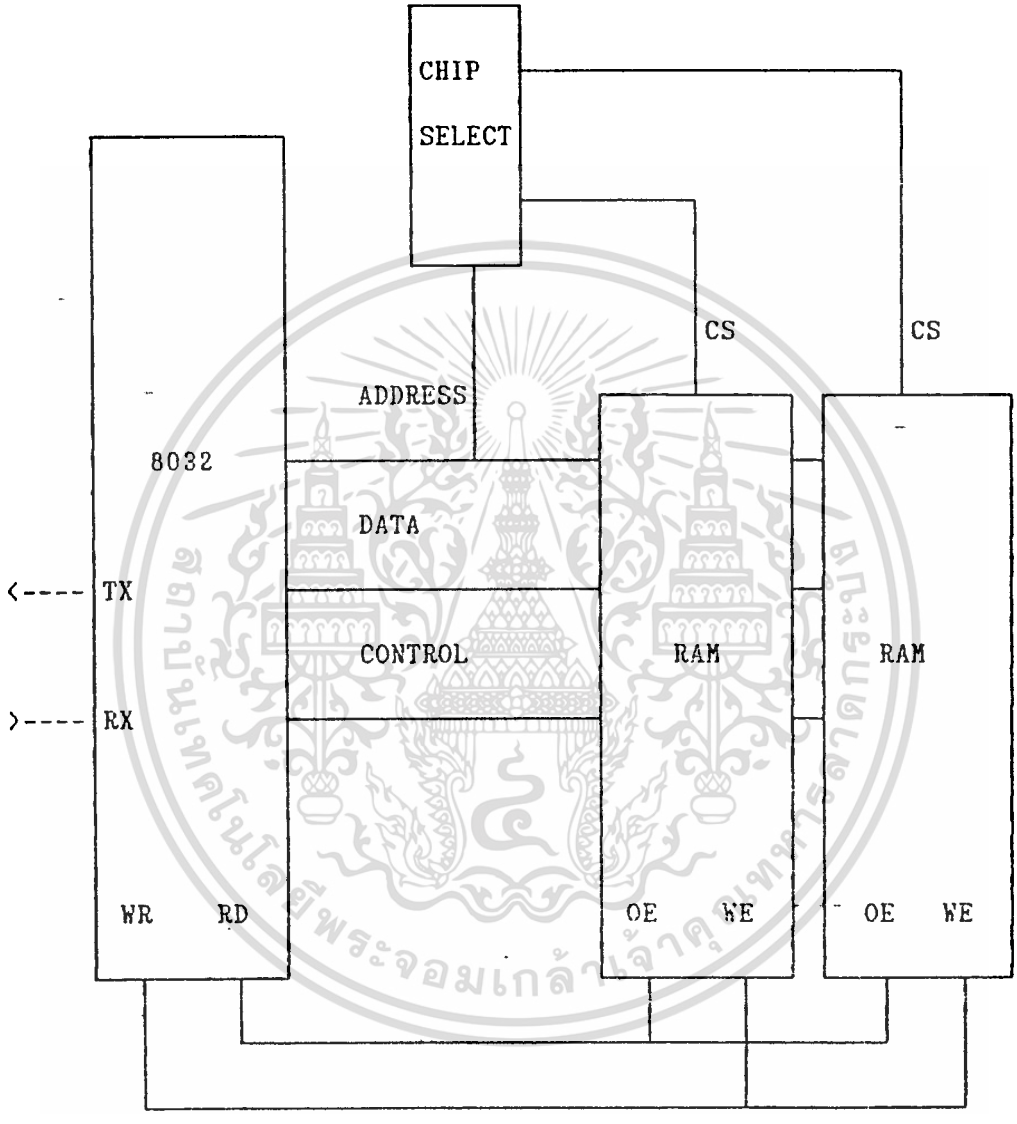
การจัดการหน่วยความของ MCS-51 ในตัวมันเองแล้วจะมีตัวแปลภาษาเบสิกบรรจุอยู่ภายใน ซึ่งในโครงงานนี้จะใช้เบอร์ 8032 ซึ่งไม่มี ROM ภายในชิพตัวมันเองแต่มีราคาถูกจะใช้ ROM ภายนอก ซึ่งบรรจุตัวแปลภาษาเบสิกอยู่ระหว่างแอดเดรส 0000H-1FFFH (PROGRAM MEMORY) ในการทำงานของ 8032 ต้องการหน่วยความจำเป็น RAM ภายนอกอย่างน้อย 1 กิโลไบต์

หลังจากทำการรีเซ็ต ชุดคอนโทรลเลอร์ 8032 แล้ว มันจะหาขนาดของหน่วยความจำของแรมภายนอกโดยเริ่มตั้งแต่แอดเดรส 0000H-0DFFH (DATA MEMORY) ข้อมูลในหน่วยความจำที่ผ่านการทำสอบเพื่อหาขนาดมีค่าเป็น 00H ถ้าขนาดของหน่วยความจำภายนอกไม่ถึง 1 กิโลไบต์ 8032 จะไม่สามารถทำงานได้ ส่วนแอดเดรส 0E00H-0FFFFH (DATA MEMORY) สงวนไว้ให้ผู้ใช้สำหรับ INPUT OUTPUT และโปรแกรมแอสเซมบลี้ ข้อมูลในแอดเดรสระหว่างนี้จะไม่ถูกเคลียเป็น 00H



รูปที่ 4.1 การจัดหน่วยความจำของ 8032

4.2 RAM ONLY MODE

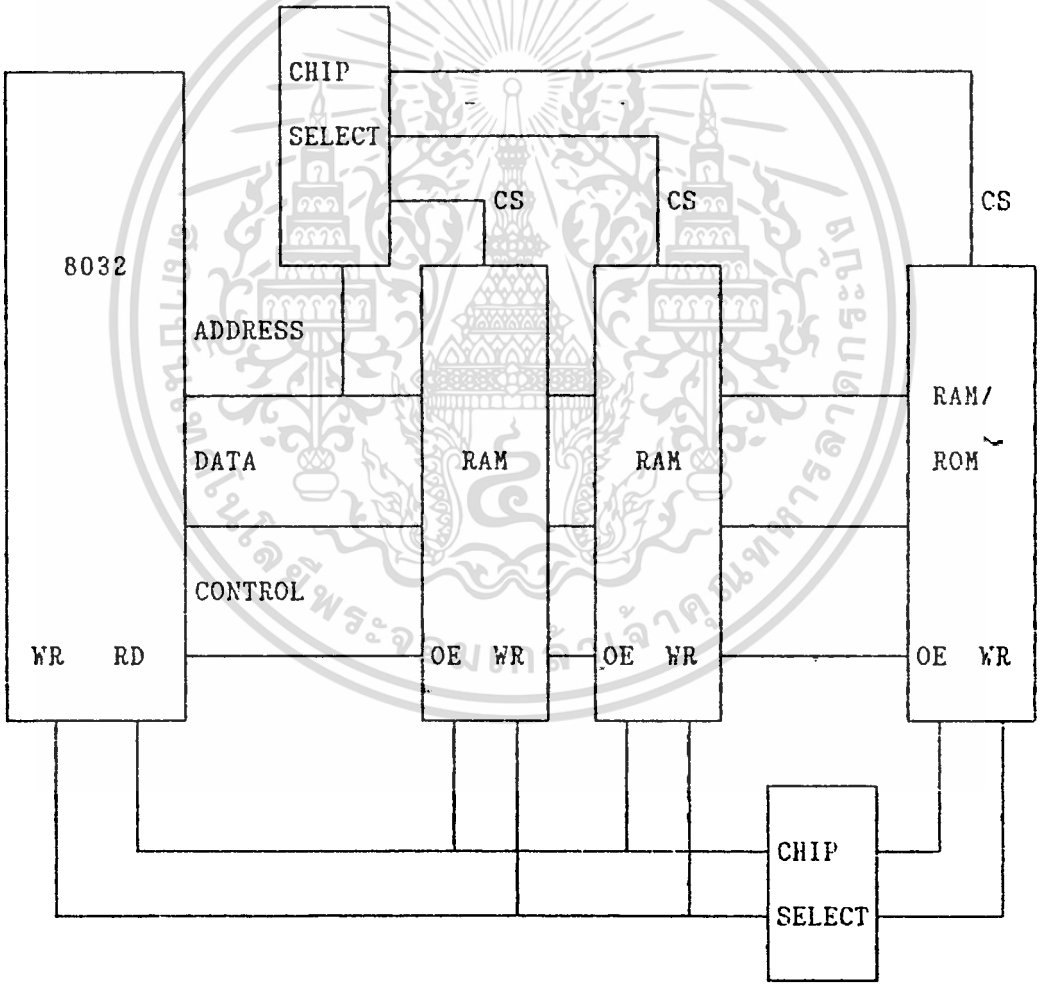


รูปที่ 4.2 RAM ONLY MODE

จากรูปจะเห็นว่าการทำงานในโหมดนี้ สามารถต่อแรมภายนอก (EXTERNAL RAM) ได้ถึง 64 โลไบต์ เริ่มต้นที่แอดเดรส 0000H ถึง 0FFFFH ในโหมดนี้จะใช้สัญญาณติดต่อกับแรมภายนอกดังต่อไปนี้

- CS ดีโค้ดแอดเดรสจา 8032 เป็นสัญญาณสำหรับแรมภายนอ
- ขา RD ใช้เป็นสัญญาณ OE สำหรับแรมภายนอก
- ขา WR ใช้เป็นสัญญาณ WR สำหรับแรมภายนอก

4.3 RAM/ROM MODE



รูปที่ 4.3 RAM/ROM MODE

การทำงานในโหมดนี้เป็นการทำงานที่สมบูรณ์ของระบบ 8032 โหมดนี้ต้องการ การจัดหน่วยความจำภายนอกให้อยู่ในแบบที่แน่นอนในโหมดนี้จะใช้สัญญาณคิดต่อกับหน่วยความจำภายนอกทั้ง RAM และ ROM ดังต่อไปนี้

- ขา RD และ WR ของ 8032 ใช้เป็นสัญญาณ OE และ WR สำหรับแรมภายนอกตามลำดับ โดยมีแอดเดรส ตั้งแต่ 0000H-7FFFH และมีสถานะเป็น DATA MEMORY ส่วนดีโค้ดแอดเดรสจาก 8032 ใช้เป็นสัญญาณ CS สำหรับแรมภายนอก

- ขา PSEN ของ 8032 ใช้เป็นสัญญาณ OE สำหรับ EPROM โดยมีแอดเดรสตั้งแต่ 2000H ถึง 7FFFH และมีสถานะเป็น PROGRAM MEMORY ดีโค้ดแอดเดรสจาก 8032 ใช้เป็นสัญญาณ CS สำหรับ EPROM ภายนอก

- ขา RD และ PSEN ของ 8032 นำมา AND กันใช้เป็นสัญญาณ OE สำหรับ RAM หรือ ROM โดยมีแอดเดรสตั้งแต่ 8000H-0FFFH โดยมีสถานะเป็นได้ทั้ง DATA และ PROGRAM MEMORY ดีโค้ดแอดเดรสจาก 8032 ใช้เป็นสัญญาณ CS สำหรับ RAM และ ROM ภายนอกและขา WR ของ 8032 ใช้เป็นสัญญาณ WR หรือ OE สำหรับแรมภายนอก

บทที่ 5

สเต็ปปิ้งมอเตอร์ไดร์ฟ

(STEPPING MOTOR DRIVE)

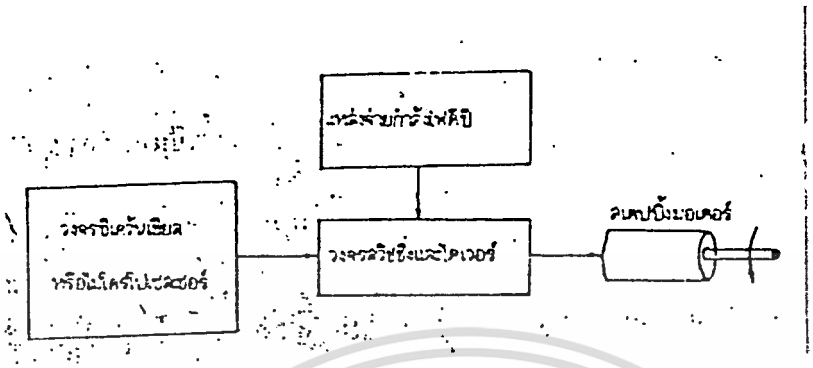
โครงสร้างและการทำงานของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ เป็นอุปกรณ์จำพวกเชิงกลทางไฟฟ้า ที่มีอินพุทเป็นกลุ่มของไบนารีโวลท์เดจและเอาท์พุทเป็นลักษณะของการเคลื่อนที่แบบเชิงมุมหรือหมุนไปเป็นสเต็ป(แต่ละสเต็ปอยู่ในช่วง 0.1 ถึง 30 องศา ขึ้นอยู่กับโครงสร้างของสเต็ปปิ้งมอเตอร์)ตามสัญญาณพัลซ์ที่ป้อนให้กับขดสเตเตอร์ซึ่งจะเกิดแรงผลักให้โรเตอร์หมุนไป แต่ลักษณะของ สเต็ปปิ้งมอเตอร์ จะมีขดของสเตเตอร์อยู่หลายขด ซึ่งเรียกว่า " เฟส " ฉะนั้นเมื่อป้อนสัญญาณที่เป็นพัลซ์ในลักษณะชีเควันของเลขไบนารีโดยผ่าน วงจรไดรเวอร์ (Driver) จะทำให้โรเตอร์หมุนได้อย่างต่อเนื่องดังบล็อคไดอะแกรม รูปที่ 1

คุณประโยชน์ของสเต็ปปิ้งมอเตอร์

ในระบบควบคุมตำแหน่งที่ใช้สเต็ปปิ้งมอเตอร์นั้นมีข้อดีอยู่หลายประการ คือ

1. เป็นลักษณะการควบคุมแบบไม่ต้องการการป้อนกลับ ไม่ว่าจะเป็นการควบคุมตำแหน่งหรือความเร็ว
2. ความผิดพลาดเกี่ยวกับตำแหน่งแทบไม่มีเลย เนื่องจากการ เคลื่อนที่ของสเต็ปปิ้งมอเตอร์นั้นเคลื่อนที่เป็นสเต็ป ด้วยจำนวนองศาที่มีค่าแน่นอน
3. สเต็ปปิ้งมอเตอร์จะถูกนำมาใช้กับเครื่องมือ ที่ต้องการความละเอียดแม่นยำและใช้อยู่ใน เครื่องมือประเภทดิจิตอล เช่น เครื่องวาดรูป เครื่องคอมพิวเตอร์ นิวเมอริคอลคอนโทรล (Computer Numerical Control) หรือ CNC
4. ไม่จำเป็นต้องใช้วงจรแปลงดิจิตอลเป็นอนาลอกเมื่ออินเตอร์เฟสกับไมโครคอมพิวเตอร์

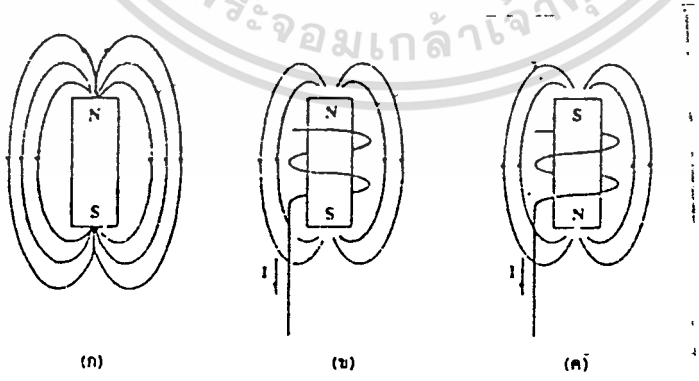


รูปที่ 5.1 บล็อกไดอะแกรมแสดงการควบคุมสเต็ปปิ้งมอเตอร์

จากบล็อกไดอะแกรม สเต็ปปิ้งมอเตอร์ จะทำงานเมื่อเราป้อน

- สัญญาณพัลซ์นาฬิกา (Clock Pulses)
- อินพุทสำหรับควบคุมทิศทางการหมุน

<u>หลักการทำงานของสเต็ปปิ้งมอเตอร์ทั่ว ๆ ไป</u>



รูปที่ 5.2 แสดงถึงสนามแม่เหล็กที่เกิดขึ้นในลักษณะต่าง ๆ

ในรูป 5.2 (ก) สนามแม่เหล็กที่เกิดขึ้นจากแม่เหล็กถาวร

5.2 (ข) สนามแม่เหล็กของแม่เหล็กไฟฟ้าที่เกิดจากกระแส I

5.2 (ค) ขั้วแม่เหล็กกลับทิศทางเมื่อขดลวดถูกพันกลับทิศทาง และทิศทางการไหลของกระแสไม่เปลี่ยนแปลง

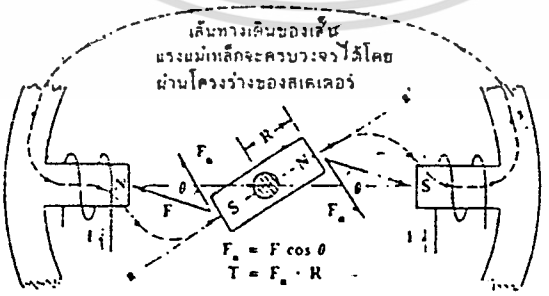
ในรูป 5.3 แท่งแม่เหล็กถาวรติดอยู่กับเพลาและหมุนได้อิสระเหมือนอาร์เมเจอร์ มีขั้วแม่เหล็กไฟฟ้า 2 ขั้วซึ่งเป็นส่วนหนึ่งของโครงโลหะที่เป็นสเตเตอร์ (Stator)

ในรูป 5.3 ตำแหน่งแกนของอาร์เมเจอร์แม่เหล็กคือ a-a' ซึ่งต่างไปจากตำแหน่งแกนขั้วของแม่เหล็กไฟฟ้าเล็กน้อยเป็นมุม θ

แรงแม่เหล็กที่เกิดจากการดึงดูดของขั้วแม่เหล็กที่ต่างกัน ทำให้เกิดส่วนของแรงปกติ

$Fn = F \cos \theta$ (แรงนี้ตั้งฉากกับแกน a-a')

ทอร์คผลรวม $T = Fn\ R$ (ทำให้อาร์เมเจอร์หมุนไปทิศทาง CW จนกว่าแกนของอาร์เมเจอร์ a-a' จะอยู่ในแนวเดียวกับแกนขั้วของสเตเตอร์)



รูปที่ 5.3 แสดงแรงดึงดูดทำให้เกิดทอร์คที่หมุนอาร์เมเจอร์ให้ไปอยู่ในตำแหน่งสมดุลย์

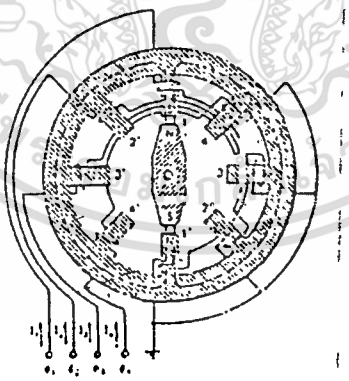
ถ้าหากมีคู่ขั้วแม่เหล็กไฟฟ้าหลาย ๆ คู่ขั้วรอบ ๆ สเตเตอร์ และถ้าหากขั้วเหล่านั้นถูกกระตุ้น ด้วยกระแสพัลซ์ในรูปแบบ ที่เรียงลำดับกัน ไปอาร์เมเจอร์ก็จะหมุนในรูปลักษณะของสเต็ปที่เป็นไปตามการหมุนของสนามแม่เหล็ก ที่เกิดจากการสวิทช์ที่เรียงลำดับ ของขดลวดขั้วแม่เหล็กไฟฟ้าของสเตเตอร์

2.1 สเต็ปมอเตอร์แบบแม่เหล็กถาวร

โครงสร้างของสเต็ปมอเตอร์แบบแม่เหล็กถาวรแสดงได้ในรูปที่ 5.3

ในรูปที่ 5.4 เป็นสเต็ปมอเตอร์แบบ 4 เฟส แต่ละเฟสเป็นขดลวดอยู่บน 2 ขั้ว ของสเตเตอร์จะต้องมี 8 ขั้ว

โรเตอร์ทำจากแม่เหล็กถาวรและอยู่ในแนวของขั้วสเตเตอร์ 1 และ 1' มันหยุดอยู่ที่ ตำแหน่งนี้ได้ด้วยกระแส I1 ที่ไหลอยู่ในเฟส 1



รูปที่ 5.4 โครงสร้างของสเต็ปมอเตอร์แบบแม่เหล็กถาวรมี 4 เฟส และแต่ละเฟสพันด้วยขดลวดบน 2 ขั้วของสเตเตอร์ มุมสเต็ปเท่ากับ 45 องศา

ขดลวดของเฟส $\phi 1$, $\phi 4$, $\phi 3$ และ $\phi 2$ (1-4-3-2 ตามลำดับ) จะได้รับพลังงานด้วยกระแสพัลซ์ที่สอดคล้องกัน I1, I2, I3 และ I2 กระแสแต่ละเฟสจะไหลในทิศทางที่แสดงในไดอะแกรม) แต่ละสเต็ปโรเตอร์จะหมุนไปตาม ทิศทางตามเข็มนาฬิกา45 องศา (360/8)

เมื่อขั้วเหนือของโรเตอร์ (แม่เหล็กถาวร) หมุนไปถึงขั้วของสเตเตอร์ หมายเลข 2 ลำดับ การขับขดลวดเฟสของสเต็ปมอเตอร์คือ 1-4-3-2 จะต้องกระทำเหมือนเดิม (เพื่อให้มอเตอร์หมุนไปตามเข็มนาฬิกาอีก 180 องศา) ยกเว้นเราต้องให้หมุนกลับทิศทางใน 180 องศา ที่เหลือ ด้วยการป้อนกระแสกลับทิศทางเพื่อให้เกิดการเหนี่ยวนำเป็นขั้วใต้ที่ขั้ว สเตเตอร์ 1', 4' 3' และ 2' ตามลำดับ (ทิศทางของกระแสแสดงในรูปที่ 5.4)

2.2 <u>สเต็ปมอเตอร์แบบค่ารีลัคแตนซ์แปรค่าได้ที่มีสแต็คเดียว</u>

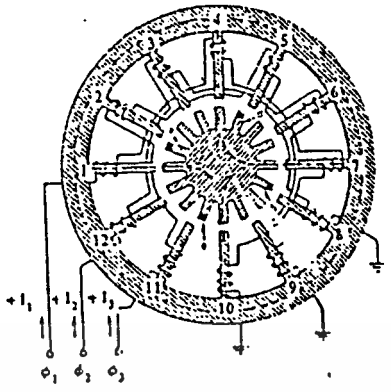
ตัวอย่างโครงสร้างของสเต็ปมอเตอร์ แบบค่ารีลัคแตนซ์แปรค่าได้ที่มีสแต็คเดียวหรือที่เรียกสั้นว่า VR สเต็ปมอเตอร์ที่สแต็คเดียวแสดงได้ในรูปที่ 5.5

VR สเต็ปมอเตอร์ที่มีสแต็คเดียวจะมีโรเตอร์เดียวเมื่อเทียบกับ VR สเต็ปมอเตอร์แบบที่หลายสแต็คหมายถึงมีหลายโรเตอร์ โรเตอร์และสเตเตอร์ทำจากสารแม่เหล็ก

สเต็ปมอเตอร์ในรูปที่ 5.5 มี 3 เฟสแต่ละเฟสใช้ขดลวดพันบน 4 ขั้ว หรือชี่ฟันของมอเตอร์

ตัวอย่าง เฟสที่ 1 พันอยู่บนขั้วที่ 1,4,7 และ 10 ของสเตเตอร์ ดังนั้นสเตเตอร์จะมี 12 ชี่ฟัน และในที่นี้กำหนดให้โรเตอร์มี 16 ชี่ฟัน

ขั้วของสเตเตอร์ที่อยู่ตรงกันข้ามจะพันด้วยขดลวดลักษณะที่ต่างกัน เพื่อให้มีความสมดุลย์ระหว่างเส้นแรงแม่เหล็กเข้าและออกจากโรเตอร์



รูปที่ 5.5 VR สเต็ปมอเตอร์แบบมีสแต็คเดียวมีรายละเอียดโครงสร้าง ดังนี้

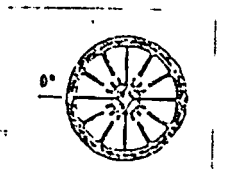
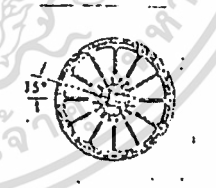
Nr=16, Ns=12, X=4 โพล/เฟส, θs=7.5องศา, Rs=48 สเต็ป/รอบ

สมมุติว่ากระแส I1 ป้อนให้กับเฟสที่ 1 ดังแสดงในรูปที่ 5.5 และโรเตอร์ทั้ง4 ซี่ฟันจะอยู่ในแนวซี่ฟันที่ 1,4 7 และ 10 ของสเตเตอร์ เส้นแรงแม่เหล็กจะเข้าสู่โรเตอร์จากสเตเตอร์ซี่ฟันที่ 4 และ 10 และออกจากโรเตอร์ไปยังซี่ฟันของสเตเตอร์ ที่ 1และ 7 ซึ่งเป็นทางเดินของเส้นแรงแม่เหล็กที่ครบวงจรโดยผ่านโครงร่างของสเตเตอร์เราจะสังเกตได้ว่าปลายของซี่ฟันของสเตเตอร์ที่ 4 จะถูกเหนี่ยวนำเป็นขั้วเหนือ (เนื่องจากเส้นแรงออกจากซี่ฟันที่ 4) และปลายของซี่ฟันโรเตอร์ซึ่งอยู่ในแนวเดียวกับซี่ฟันที่ 4 ของสเตเตอร์ จะเป็นเส้นทางผ่านเข้าไปยังโรเตอร์ของเส้นแรงแม่เหล็ก และเหนี่ยวนำให้ปลายของซี่ฟันของโรเตอร์นั้นเป็นขั้วใต้ การทำให้เกิดลักษณะเป็นแม่เหล็กนี้ จะทำให้มีเส้นแรงแม่เหล็กอย่างต่อเนื่องผ่านช่องว่าง (gap) ระหว่างซี่ฟันทั้งสอง ที่อยู่ในแนวเดียวกันส่วนซี่ฟันของสเตเตอร์และโรเตอร์ที่เหลืออีก 3 คู่ ก็เกิดลักษณะของแม่เหล็กในทำนองเดียวกัน

ในสภาวะต่อไปเราจะให้โรเตอร์หมุนไปหนึ่งสเต็ปในทิศทาง CW เราจะต้องจ่ายพลังงาน ให้กับเฟส 3 ที่ขดลวดพันอยู่บนซี่ฟันที่ 2, 5, 8 และ 11 ของสเตเตอร์ด้วยด้วยกระแส I3 หลังจากหยุดจ่ายกระแส I1 แล้ว ในคอนนี้เส้นแรงแม่เหล็กจะหาทางเดินที่ต่าง ไปจากเดิมเพื่อทำให้วงจรแม่เหล็กครบวงจร (เหมือนกับกระแสในวงจรไฟฟ้าจะหาเส้นทางไหลในส่วนที่มีความต้านทานต่ำที่สุด) ในทำนองเดียวกันเส้นแร แม่เหล็กในวงจรแม่เหล็กก็จะหาเส้นทางเดินที่มีค่าริลัคแต้นซ์ต่ำที่สุด (ช่องว่างอากาศระหว่างซี่ฟันจะทำให้เกิดค่ารีลัคแต้นซ์ต่อเส้นแรงแม่เหล็กช่องว่างกว้างมากค่ารีลัคแต้นซ์ก็จะมีค่ามาก) ด้วยเหตุผลดังกล่าวเส้นแรงแม่เหล็กจะออกจากขั้วที่ 2 และ 8 ของสเตเตอร์ซึ่งถูกเหนี่ยวนำให้เป็นขั้วเหนือ และเส้นแรงแม่เหล็กนี้ก็จะกระโดดผ่านช่องว่างไปยังซี่ฟัน ของโรเตอร์ที่ใกล้ที่สุด ซี่ฟัน a และ b ของโรเตอร์ที่อยู่ใกล้ที่สุดและจะ

ถูกเหนี่ยวนำให้เป็นขั้วใต้ เส้นแรงแม่เหล็กจะออกจากชี่ฟัน d และ e ของโรเตอร์ผ่านช่องว่างอากาศเข้าสู่ฟันที่ 5 และ 11 ของสเตเตอร์ ดังนั้น ส่วนที่เหลือของวงจรแม่เหล็กจะสมบูรณ์โดยผ่านโครงร่างของสเตเตอร์ ในระหว่างเวลานั้นแรงของแม่เหล็กหรือแรงดึงดูดจะเกิดขึ้นระหว่างชี่ฟันที่ 2 ของสเตเตอร์ (ถูกเหนี่ยวนำเป็นขั้วเหนือ) และชี่ฟัน a ของโรเตอร์ (ถูกเหนี่ยวนำเป็นขั้วใต้)แรงดึงดูดจะเกิดขึ้นระหว่างคู่ขั้ว (11,e) (8,6) และ (5,d) ด้วย ดังที่อธิบายในรูปที่5.3 ผลที่เกิดขึ้นนี้ จะทำให้เกิดทอร์ค กระทำต่อโรเตอร์หมุนไปจนกระทั่งชี่ฟัน a, d, b และ e ของโรเตอร์อยู่ในแนวเดียวกับชี่ฟัน 2,5,8 และ 11 ของสเตเตอร์ตามลำดับขณะเวลาดังกล่าวช่างช่องว่างระหว่างชี่ฟันตามลำดับจะมีค่าน้อยที่สุด ผลลัพธ์ของค่ารีลัค -แตันซ์ จะมีค่าต่ำที่สุดและเส้นแรงแม่เหล็กจะมีค่าสูงสุดผ่านวงจรแม่เหล็ก ที่ตำแหน่งนี้เป็นตำแหน่งที่สมดุลย์ของการขับเฟส 3 ในกระบวนการที่กล่าวมาแล้วโรเตอร์จะเคลื่อนที่ในทิศทาง CW หนึ่งสเต็ปเป็นมุม 7.5 องศา

ลำดับการทำงานที่สมบูรณ์แสดงได้ในตารางที่ 5.1 เมื่อตำแหน่งเริ่มต้นของชี่ฟันของโรเตอร์ จะเป็นสีดำเพื่อให้เราทำความเข้าใจได้ชัดเจนถึงการหมุนของโรเตอร์ในทิศทาง CW เมื่อเฟสถูกขับในลักษณะเรียงลำดับ 1-3-2-1 ชี่ฟันของโรเตอร์ที่เป็นสีดำจะเคลื่อนที่ไป 3 สเต็ป คิดเป็นมุมได้เท่ากับ 22.5 องศา เราจะขับเฟสในลักษณะเรียงลำดับเดิมขึ้นใหม่อีกเมื่อต้องการให้โรเตอร์หมุนต่อเนื่องในทิศทาง CW แต่ถ้าเราต้องการให้ โรเตอร์หมุนในทิศทาง CCW เราต้องกลับการเรียงลำดับเฟสเป็น 1-2-3-1

การเรียงลำดับเฟส	ตำแหน่งของโรเตอร์และเส้นแรงแม่เหล็ก
ตำแหน่งโรเตอร์เริ่มต้น : - เฟส ϕ 1 ได้รับพลังงาน - ขี้ฟันของโรเตอร์จะอยู่ในแนวขี้ฟัน ที่ 1,4,7,10 ของสเตเตอร์	
สเต็ปที่ 1: เฟส ϕ 3 ได้รับพลังงาน - ขี้ฟันของโรเตอร์จะอยู่ในแนวขี้ฟัน ที่ 2,5,8,11 ของสเตเตอร์ - โรเตอร์จะเคลื่อนที่ไปในทิศทาง CW เป็นมุม 7.5 องศา (1/3 ช่วงห่าง ระหว่างขี้ฟันของโรเตอร์)	
สเต็ปที่ 2 : เฟส ϕ 2 ได้รับพลังงาน - ขี้ฟันของโรเตอร์จะอยู่ในแนวขี้ฟัน ที่ 3,6,9,12 ของสเตเตอร์ - โรเตอร์ที่จะเคลื่อนที่ไปในทิศทาง CW รวมเป็นมุม 7.5 องศา	

(ต่อ)

การเรียงลำดับเฟส	ตำแหน่งของโรเตอร์และเส้นแรงแม่เหล็ก
สเต็ปที่ 3 : เฟส ϕ 1 ได้รับพลังงาน - ซี่ฟันของโรเตอร์จะอยู่ในแนวซี่ฟัน ที่ 1,4,7,10 ของสเตเตอร์ - โรเตอร์จะเคลื่อนที่ไปในทิศทาง CW รวมเป็นมุม 22.5 องศา (เคลื่อนไปได้ 1 ช่วงห่างระหว่างซี่ฟันของโรเตอร์)	

ตารางที่ 5.1แสดงลำดับการสวิทช์ 3 สเต็ปของ VR สเต็ปมอเตอร์แบบสแต็คเดียว และแสดงกึ่งตำแหน่งของโรเตอร์และเส้นทางของเส้นแรงแม่เหล็กเมื่อโรเตอร์เคลื่อนที่ไปในแต่ละสเต็ป

สัญลักษณ์ต่าง ๆ ของ VR สเต็ปมอเตอร์

Nr = จำนวนซี่ฟันของโรเตอร์

Ns = จำนวนซี่ฟันของสเตเตอร์

Np = จำนวนเฟส

Pr = ความห่างระหว่างปลายซี่ฟันของโรเตอร์ (องศา)

Ps = ความห่างระหว่างปลายซี่ฟันของสเตเตอร์ (องศา)

θs = มุมสเต็ป (องศา)

Rs = อัตราการสเต็ปหรือความเร็วในการสเต็ป (สเต็ป/รอบ)

X = Ns/Np = จำนวน ซี่ฟันของสเตเตอร์ต่อเฟส

พารามิเตอร์ต่าง ๆ ของสเต็ปมอเตอร์

1. ความห่างระหว่างปลายฟันของโรเตอร์และสเตเตอร์ (tooth pitch)

$$Pr = \frac{360}{Nr} \quad \text{และ} \quad Ps = \frac{360}{Ns}$$

2. มุมสเต็ป (step angle)

ในตารางที่ 5.1 โรเตอร์จะเคลื่อนที่ในขนาดมุม Pr ได้เท่ากับ Np สเต็ป ดังนั้นเราจะหามุมสเต็ปได้

$$\bullet s = \frac{Pr}{Np} = \frac{360}{Nr\ Np} \quad \text{องศา/สเต็ป}$$

มุมสเต็ปจะเท่ากับความแตกต่างระหว่าง Pr และ Ps ดังนั้นเราหามุมสเต็ปได้เป็น

$$\bullet s = \left| Pr-Ps \right| \quad \text{องศา/สเต็ป}$$

3. อัตราการสเต็ป (stepping rate)

ความเร็วในการสเต็ปต่อรอบ (360 องศา) หาได้เป็น

$$Rs = \frac{360}{\bullet s} = Nr\ Np \quad (\text{สเต็ป/รอบ})$$

4. ความเร็วของสเต็ปมอเตอร์ (speed of step motor)

เมื่อเราป้อนอินพุทพัลซ์ที่มีความถี่ (f) สเต็ปต่อพัลซ์ให้กับสเต็ปมอเตอร์ มอเตอร์จะสเต็ปไปด้วยความเร็ว $\frac{(\text{สเต็ป})}{\text{พัลซ์}} \times f \frac{(\text{พัลซ์})}{\text{วินาที}}$

$$\frac{1\ (\text{รอบ})}{Rs\ \text{สเต็ป}} \times f \frac{(\text{พัลซ์})}{\text{วินาที}} \frac{(\text{สเต็ป})}{\text{พัลซ์}} \times 60 \frac{(\text{วินาที})}{\text{นาที}}$$

$$\text{ความเร็วของมอเตอร์ } (W) = \frac{60f}{Rs} = \frac{60f}{NpNr} = \frac{\bullet sf}{6} \ (\text{rpm})$$

5. จำนวนโพลของสเตเตอร์ต่อเฟส (number of stator poles per phase)

$$\text{จำนวนโพลของสเตเตอร์ต่อเฟส } (X) = \frac{Ns}{Np}$$

หรือ $X = \frac{Rs}{Np\ (Np \pm 1)} = \frac{Nr}{(Np \pm 1)}$

จำนวนโพลของสเตเตอร์ต่อเฟส (X) จะสัมพันธ์กับอัตราการสเต็ปหรือจำนวนซี่ฟันของโรเตอร์

สเต็ปมอเตอร์ในรูปที่ 5.5 เราสามารถสรุปการเลือกพารามิเตอร์บางตัวของสเต็ปมอเตอร์ได้ดังตาราง

ตารางที่ 5.2 แสดงการเลือกพารามิเตอร์ของสเต็ปมอเตอร์

Np	Rs	Nr	X	Ns
3	48	16	4	12
			8	24
4	48	12	4	16
4	64	16	?	?

ตัวอย่าง การหาพารามิเตอร์ของสเต็ปมอเตอร์

ขั้นแรกเรากำหนดความต้องการของมุมสเต็ป = 9 องศา

มุมสเต็ปจะเป็นตัวจำกัดอัตราการสเต็ป $= \frac{360}{9} = 40$ สเต็ป/รอบ

ในเงื่อนไขเหล่านี้เราอาจจะต้องใช้สเต็ปมอเตอร์ที่มี 4 หรือ 5 เฟส ที่มีสเตเตอร์ 2 โพล ต่อเฟส

ถ้า Np = 4

$Nr = \frac{Rs}{Np} = \frac{40}{4} = 10$

$Ns = NpS = 4 \times 2 = 8$

ถ้า Np = 5

$$Nr = \frac{40}{5} = 8$$

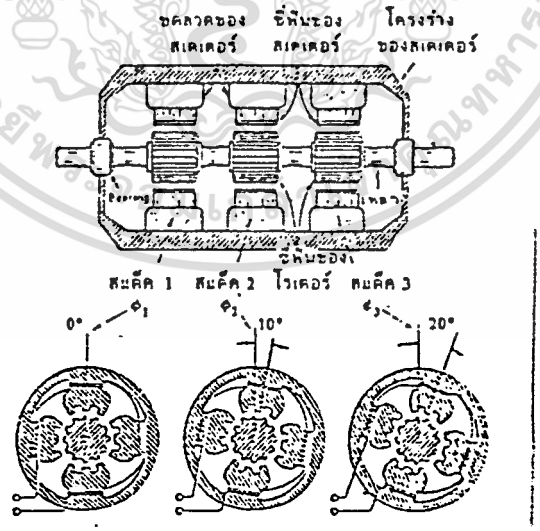
$$Ns = 5 \times 2 = 10$$

2.3 สเต็ปมอเตอร์แบบรีลัคแตนซ์แปรค่าได้และมีหลายสแต็ค

สเต็ปมอเตอร์แบบรีลัคแตนซ์แปรค่าได้ (VR) และมีหลายสแต็คหรือมากกว่า 1สแต็คขึ้นไป สแต็คในที่นี้หมายถึงเฟสซึ่งประกอบด้วยโรเตอร์ ที่เป็นซี่ฟันและโครงร่างของสเตเตอร์อยู่รอบนอก

สเตปมอตอร์แบบ VR ที่มี 3 สแต็ค (หมายถึง 3 เฟส)มีโครงสร้างดังแสดงในรูปที่ 5.6

สเต็ปมอเตอร์ในรูปที่ 5.6 ได้ถูกออกแบบให้สเตเตอร์ของแต่ละสแต็คประกอบด้วย 4 โพล และแต่ละโพลจะมีซี่ฟัน 3 ซี่ ซึ่งต่างจาก VR สเต็ปมอเตอร์แบบสแต็คเดียว(แต่ละโพลจะมีซี่ฟันเดียว) ข้อสังเกตในแต่ละสแต็คจำนวนซี่ฟันของโรเตอร์และสเตเตอร์จะมีจำนวนเท่ากัน ซึ่งต่างกับ VR สเต็ปมอเตอร์แบบสแต็คเดียวคือจำนวนซี่ฟันของโรเตอร์และสเตเตอร์จะเท่ากันไม่ได้ ถ้าหากมีจำนวนซี่ฟันเท่ากันมันจะไม่ทำงาน



รูปที่ 5.6 แสดงโครงสร้างของสเต็ปมอเตอร์แบบ VR ที่มี 3 เฟส โรเตอร์และสเตเตอร์ของแต่ละเฟส (สแต็ค) จะมี 12 ซี่ฟันและมุมสเต็ป (θs)=10 องศาแต่ละเฟสของสเตเตอร์ที่เรียงลำดับต่อเนื่องกันจะถูกจัดตำแหน่ง ให้ต่างกันเท่ากับ 1/3 ของช่องห่างระหว่างซี่ฟันของโรเตอร์ (10 องศา)

การทำงานของ VR สเต็ปมอเตอร์ที่มี 3 สแต็ค

ไดอะแกรมส่วนล่างของรูปที่ 5.6 แสดงถึงโครงสร้างของโรเตอร์และสเตเตอร์ของ VR สเต็ปมอเตอร์ที่มี 3 สแต็ค

แต่ละ สแต็คจะมี Nr = Ns

แต่ละสแต็คจะมีตำแหน่งของสเตเตอร์แตกต่างจากตำแหน่ง ของสเตเตอร์ในสแต็คถัดไปเท่ากัน 10 องศา

ส่วนชี่ฟันของโรเตอร์ทั้ง 3 อันจะประกอบอยู่บนแกนเดียวกันและได้รับการปรับแต่งให้อยู่แนวเดียวกันอย่างสมบูรณ์

ตามปกติเราจะหาค่ามุมสเต็ป (หรือ index angle) ได้จากสมการ

ในที่นี้เราจะหา θi (index angle) ได้จากสมการเดียวกันคือ

$$\theta i = \frac{Pr}{Np} = \theta s$$

ในกรณีนี้ Nr = Ns = 12 ดังนั้นเราหา Pr = 360/12 = 30 องศา

และค่า θi = 30/3 = 10 องศา

สเต็ปมอเตอร์แบบ 3 สแต็ค ถึงแม้ว่าโรเตอร์ทั้ง 3 อันจะติดอยู่บนเพลาอันเดียวกันสแต็คทั้ง 3 สแต็คจะมีวงจรแม่เหล็กที่แยกกันดังนี้

ถ้าเฟสที่ 1 ถูกจับด้วยกระแสเป็นเฟสแรกเริ่มต้นให้ชี่ฟันของ โรเตอร์สเตเตอร์อยู่ในแนวเดียวกันส่วนชี่ฟันของโรเตอร์และสเตเตอร์ในสแต็คที่ 2 ในขณะนั้นจะมีตำแหน่งต่างกัน 10 องศา และชี่ฟันของโรเตอร์และสเตเตอร์ในสแต็คที่ 3 จะมีตำแหน่งต่างกัน 20 องศาต่อจากนั้นเราหยุดจ่ายกระแส (กระแสยคลวดสเตเตอร์) ในสแต็คที่ 1 และป้อนกระแสให้กับสแต็คที่ 2 อยู่ในแนวเดียวกับในขณะนี้ชี่ฟันของโรเตอร์และสเตเตอร์ในสแต็คที่ 3 จะมีตำแหน่งต่างกัน 10 องศา ต่อจากนั้นเราหยุดจ่ายกระแสในสแต็คที่ 2 และป้อนกระแสให้กับสแต็คที่ 3 โรเตอร์จะหมุนไปอีก 10 องศา ซึ่งจะทำให้ชี่ฟันของโรเตอร์และสเตเตอร์ในสแต็คที่ 3 อยู่ในแนวเดียวกัน ส่วนชี่ฟันของโรเตอร์และสเตเตอร์ ในสแต็คที่ 1 จะมีตำแหน่งต่างกัน 10 องศา

ลำดับการสวิทช์กระแสให้แต่ละสแต็คแสดงได้ในตารางที่ 5.3 ซึ่งแสดงให้เห็นว่าเพลาของสเต็ปมอเตอร์จะเคลื่อนที่ไปเท่ากับหนึ่งช่องของระยะระหว่าง ชี่ฟันของโรเตอร์ (30 องศา) ภายใน 3 สเต็ป

	สแต็คที่ 1	สแต็คที่ 2	สแต็คที่ 3
ตำแหน่งเริ่มต้นของโรเตอร์ : - เฟส ϕ 1 ได้รับพลังงาน			
สเต็ปที่ 1 : - เฟส ϕ 2 ได้รับพลังงาน - โรเตอร์จะเคลื่อนที่ไป 10 องศา			
สเต็ปที่ 2 : - เฟส ϕ 3 ได้รับพลังงาน - โรเตอร์จะเคลื่อนที่ไป 20 องศา			
สเต็ปที่ 3 : - เฟส ϕ 3 ได้รับพลังงาน - โรเตอร์จะเคลื่อนที่ไป 30 องศา หรือเท่ากับหนึ่งช่องของระยะห่างระหว่างซี่ฟันของโรเตอร์			

ตารางที่ 5.3 แสดงลำดับการสเต็ปของ VRSM แบบ 3 เฟส Nr = Ns = 12, Pr = 30 องศา และ θs = 10 ซี่ฟันของโรเตอร์สีดำจะเคลื่อนที่ไปในทิศทาง CW 10 องศา ในแต่ละสเต็ปรวมทั้งหมด 30 เมื่อ สเต็ปไปครบ 3 สเต็ป สำหรับการหมุนในทิศทาง CW ตามลำดับ การขับเฟส 1-2-3-1 และเมื่อต้องการให้หมุนในทิศทาง CCW ลำดับการขับเฟสก็ต้องกลับเป็น 1-3-2-1

ตามปกติเพลาของมอเตอร์จะเคลื่อนที่ไปหนึ่งช่อง ของระยะห่างระหว่างซี่ฟันของโรเตอร์ (rotor tooth pitch) ด้วยการสเต็ปไป Np สเต็ป เมื่อ Np คือจำนวนสแต็คที่ใช้(หรือเท่ากับจำนวนเฟส)

ลำดับการสวิทช์ที่แสดงในตารางที่ 5.3 เราสามารถนำมาเขียนเป็นตารางได้ดังในรูปที่ 5.7 วงจรสวิทช์ประกอบด้วย VRSM แบบ 3 เฟส (สัญลักษณ์ของสเต็ปมอเตอร์) การขับเฟสแสดงได้ด้วยสวิทช์และแหล่งการกำเนิดดีซี

สเตป	S_1	S_2	S_3
1	×		
2		×	
3			×
1	×		

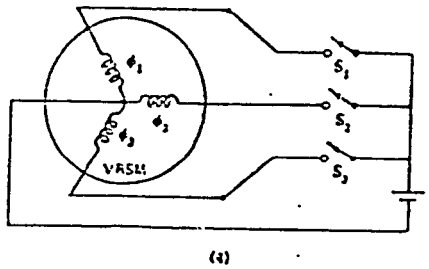
(ก)

สเตป	S_1	S_2	S_3
1	×	×	
2		×	×
3	×		×
1	×	×	

(ข)

สเตป	S_1	S_2	S_3
1	×	×	
2		×	
3		×	×
4			×
5	×		×
6	×		
1	×	×	

(ค)



(ง)

รูปที่ 5.7 แสดงถึง VRSM แบบ 3 เฟส (ก) ตารางแสดงลำดับการขับแบบเฟสเดียวในทิศทาง CW (ข) ตารางแสดงลำดับการขับแบบ 2 เฟสในทิศทาง CW (ค) การขับแบบ ครึ่งสเต็ปในทิศทาง CCW เราจะต้องกลับลำดับของการของก็อให้อ่านตาราง(ก) (ข) และ (ค) จากข้างล่างขึ้นไปข้างบน

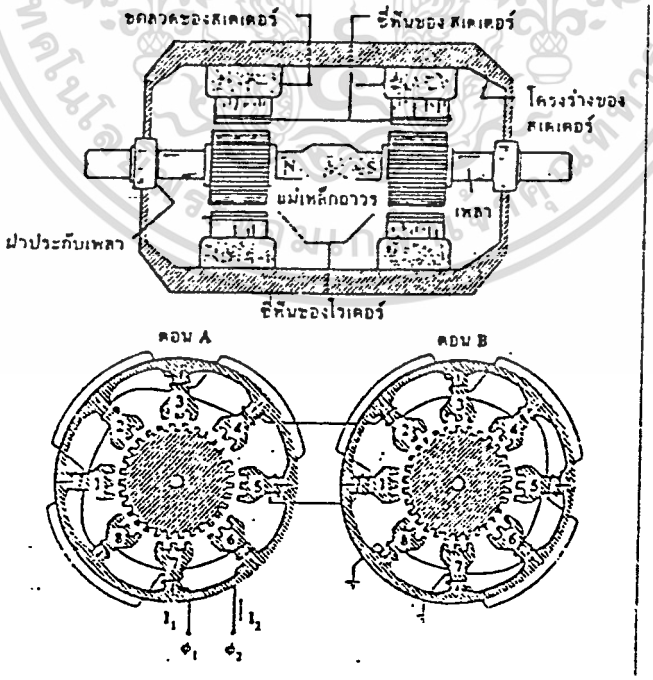
จากตาราง (ก) ถ้าเราจับเฟสที่ 1 และเฟสที่ 2 เรียงตามลำดับมอเตอร์จะหมุนไปหนึ่งสเต็ป

จากตาราง (ข) ถ้าเราจับเฟสที่ 1 และเฟสที่ 2 พร้อมกันเพลาของมอเตอร์จะหมุนไป 1/2 สเต็ป ต่อจากนั้นเราจับเฟสที่ 2 และเฟสที่ 3 พร้อมกันอีก ก็จะทำให้มอเตอร์หมุนไปครบเต็มหนึ่งสเต็ป ดังนั้นการจับแบบ 2 เฟส เราเรียงลำดับการจับได้ดังนี้ 1-2, 2-3, 3-1 และ 1-2 กระทำซ้ำแบบเดิมไปเรื่อย ๆ

อย่างไรก็ตามการจับแบบ 2 เฟส หรือ 1 เฟสจะให้การหมุนเป็นมุมสเต็ปเท่ากันที่ต่างกัน ก็คือการจับแบบ 2 เฟสจะให้การหมุนของโรเตอร์นำหน้าการจับแบบเฟสเดียวด้วย ขนาด 1/2 สเต็ป นอกจากนั้นการจับแบบ 2 เฟส จะต้องการกระแสเป็น 2 เท่าของการจับแบบเฟสเดียว

ตาราง (ค) แสดงการจับแบบ 2 เฟสสลับกับการจับแบบ 1 เฟสซึ่งจะทำให้โรเตอร์หมุนไป 1/2 สเต็ปเท่านั้น การจับแบบนี้จะทำให้จำนวนสเต็ปต่อรอบเพิ่มขึ้นเป็น 2 เท่าจากเดิม

2.4 สเต็ปมอเตอร์แบบไฮบริด



รูปที่ 5.8 โครงสร้างของไฮบริดสเต็ปมอเตอร์ : Nr = 30, Ns = 24 ซี่ฟันของสเตเตอร์ทั้ง 2 คอนจะอยู่ในแนวเดียวกันส่วนซี่ฟันของโรเตอร์ทั้งสองตัวจะมีตำแหน่งต่างกัน 1/2 Pr (=6 องศา), θ_s = 3 องศา

ไฮบริดสเต็ปมอเตอร์ (HSM) มีคุณลักษณะผสมของ PM และ VR สเต็ปมอเตอร์ในรูปที่ 5.8 แสดงถึงโครงสร้างของ HSM ประกอบด้วย 2 คอนกับแกนแม่เหล็กอยู่ระหว่าง2 คอน แต่ละคอนประกอบด้วย ชี่ฟันของโรเตอร์และ โพลของสเตเตอร์ที่มีชี่ฟันเช่นกัน และพันด้วยขดลวด รายละเอียดโครงสร้างของสเตเตอร์และโรเตอร์ของแต่ละคอนแสดงในไดอะแกรมของรูปที่ 5.8

ลักษณะโครงสร้างของไฮบริดสเต็ปมอเตอร์

- จำนวนชี่ฟันของโรเตอร์และของสเตเตอร์ไม่เท่ากัน
- คอน A และคอน B มีโครงสร้างเหมือนกัน
- ชี่ฟันของสเตเตอร์ทั้ง 2 คอนจะอยู่ในแนวเดียวกันอย่างถูกต้อง
- ส่วนชี่ฟันของโรเตอร์ทั้ง 2 คอนจะมีตำแหน่งที่แตกต่างกัน 1/2 Pr (ในรูปที่ 2.8 กำหนดให้ Pr = $\frac{360}{30}$ = 12 องศา ดังนั้นตำแหน่งชี่ฟันของโรเตอร์ทั้ง 2 คอนจะแตกต่างกัน 6 องศา)
- สเตเตอร์ของแต่ละคอนมี 8 โพลแบ่งออกเป็น 2 สเตเตอร์เฟส
- เฟสที่ 1 จะพันขดลวดบนสเตเตอร์โพลหมายเลข 1,3,5 และ 7 ของทั้งในคอน A และคอน B
- เฟสที่ 2 จะพันขดลวดบนสเตเตอร์โพลหมายเลข 2,4,6 และ 8 ของทั้งในคอน A และคอน B
- แกนแม่เหล็กถาวรจะเหนี่ยวนำโรเตอร์ในคอน A ให้เป็นแม่เหล็กขั้วเหนือ และโรเตอร์ในคอน B ให้เป็นแม่เหล็กขั้วใต้ ความซับซ้อนจะเพิ่มมากขึ้น เนื่องจาก การแบ่งส่วนของขดลวดเฟสใน 2 คอนทำให้ได้วงจรแม่เหล็กที่ซับซ้อนและได้เส้นทางเดินของเส้นแรงแม่เหล็ก ที่แตกต่างกันเป็นวงกลม ทิศทางเดินของสนามแม่เหล็กของสเตเตอร์โพล จะขึ้นอยู่กับทิศทางการไหลของกระแสเฟส ดังแสดงด้วยลูกศรในรูปที่ 5.8

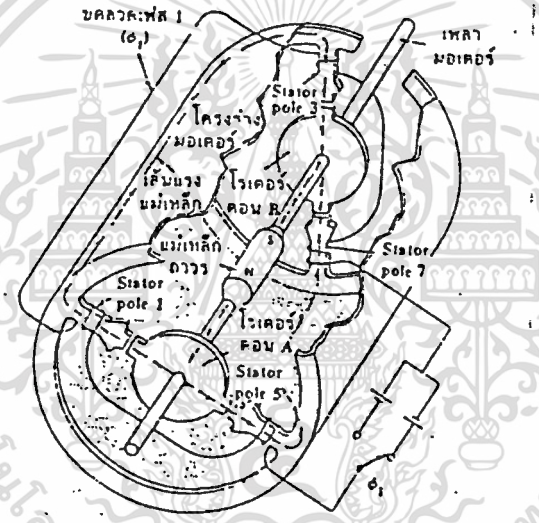
การทำงานของไฮบริดสเต็ปมอเตอร์

ชี่ฟันของโรเตอร์ในคอน A จะอยู่ในแนวเดียวกับชี่ฟันของสเตเตอร์ของโพลที่ 1 และโพลที่ 3 ส่วนของคอน B จะอยู่ในแนวเดียวกับชี่ฟันของโพลที่ 3 และโพลที่ 7 ดังแสดงในรูปที่ 5.9

เพื่อให้เพลาของมอเตอร์หมุนไปหนึ่งสเต็ปในทิศทาง CW เราจะต้องหยุดป้อนกระแส I1 และป้อนกระแส I2 ให้กับเฟสที่ 2 ($\phi 2$)

ในตารางที่ 2.4 ซี่ฟันของโรเตอร์ที่เป็นสีดำใช้สำหรับอ้างอิง ซี่ฟันสีดำจะอยู่ใกล้แนวซี่ฟัน ของสเตเตอร์โพลที่ 4 และโพลที่ 8 ในตอน A และโพลที่ 2 และโพลที่ 6 ในตอน B มากที่สุด (ซี่ฟันของโรเตอร์ที่เป็นสีดำอยู่ห่างจากแนวซี่ฟันของสเตเตอร์เท่ากับ 1สเต็ปพอดี)

เราจะต้องป้อนกระแส I2 ในทิศทางที่ถูกต้องคือจะต้องทำให้โพลที่ 4 และโพลที่ 8 และโพลที่ 2 และโพลที่ 6 ถูกเหนี่ยวนำเป็นแม่เหล็กในทิศทางที่ถูกต้อง (เกิดวงจรแม่เหล็กที่สมบูรณ์) ด้วย I2 ในกรณีนี้ I2 จะต้องเป็นลบ



รูปที่ 5.9 วงจรแม่เหล็กของ HSM แสดงถึงเส้นทางเดินของเส้นแรงแม่เหล็กเมื่อเฟสที่ 1 ได้รับพลังงานและเส้นแรงแม่เหล็กเกิดขึ้นในคอน A และผ่านโพลที่ 1 และโพลที่ 5 เข้าไปยังโรเตอร์ของคอน B ผ่านโพลที่ 3 และโพลที่ 7 เข้าสู่ขั้วใต้ (S) ของแม่เหล็กถาวร

ในตารางที่ 5.4 แสดงถึงลำดับการสวิทช์ (ให้กระแสไหล) ให้มอเตอร์หมุนไปในทิศทาง CW 4 สเต็ป ซึ่งแสดงถึงตำแหน่งของโรเตอร์และทิศทางการเป็นแม่เหล็กของสเตเตอร์โพลในแต่ละคอนด้วยการกำหนดทิศทางของการไหลของกระแสเฟส สำหรับการหมุนในทิศทาง CW (ดังแสดงในรูป) เราจะต้องกำหนดลำดับของกระแสเฟสดังนี้ 1, +2,-1, -2 และ 1+ ตามลำดับ ถ้าต้องการหมุนในทิศทาง CCW ลำดับเหล่านั้นก็กลับไปเป็น1+, 2+, 1-, 2- และ 1+

เพลาของมอเตอร์หมุนไปได้หนึ่งช่องห่างระหว่างซี่ฟันภายใน 4 สเต็ป ดังนั้นมุมสเต็ปจะต้องเท่ากับ 1/4 Pr หรือมีค่าเท่ากับ $|Ps - Ps|$ ดังนั้น

$$\theta s = \frac{Pr}{4} = \frac{360}{4Nr} = \frac{90}{Nr}$$

$$\theta s = |Ps - Pr|$$

สเตป	θ_1 I_1	θ_2 I_2	เส้นแรงออกจากคอน A	เส้นแรงเข้าสู่คอน B	คอน A	คอน B
1	+		1,5	3,7		
2		-	4,8	2.6		
3	-		3,7	1,5		
4		+	2,6	4,8		
1	+		1,5	3,7		

ตารางที่ 2.4 ลำดับ 4 สเต็ปของ HSS แบบ 2 เฟส ในแต่ละสเต็ปสะท้อนตำแหน่งของโรเตอร์และทิศทางของเส้นแรงแม่เหล็ก Nr = 30, = 24, θs = 3 องศา ซี่ฟันของโรเตอร์ที่เป็นสีดำจะหมุนในทิศทาง CW ไป 3 องศา ในแต่ละสเต็ปได้เป็น 12 องศา เมื่อครบตามจำนวนลำดับ (หนึ่งช่องห่างระหว่างซี่ฟันของโรเตอร์) สำหรับการหมุนในทิศทาง CW จะต้องจัดลำดับการยับเป็น 1+, 2-, 1-, 2+, 1+

ตารางที่ 5.4 ลำดับ 4 สเต็ปของ HSM แบบ 2 เฟส ในแต่ละสเต็ปแสดงถึงตำแหน่งของโรเตอร์และทิศทางของเส้นแรงแม่เหล็ก Nr = 30, = 24, Os = 3 องศา ชี่ฟันของโรเตอร์ที่เป็นสีดำจะหมุนในทิศทาง CW ไป 3 องศา ในแต่ละสเต็ปได้เป็น 12 องศา เมื่อครบตามจำนวนลำดับ (หนึ่งช่องห่างระหว่างชี่ฟันของโรเตอร์) สำหรับการหมุนในทิศทาง CW จะต้องจัดลำดับการขับเป็น 1+, 2-, 1-, 2+, 1+

ในรูป Nr = 30 และ Ns = 24

ดังนั้น $\phi s = \frac{90}{30} = 3$ องศา

$= \frac{360}{24} - \frac{360}{30} = 3$ องศา

ไฮบริดสเต็ปมอเตอร์ (HSM) จะทำงานด้วยกระแสเฟสที่มีการไหล ได้สองทิศทาง ดังนั้นเราจำเป็นต้องใช้เพาเวอร์ซัพพลาย 2 ตัว (bipolar drive)

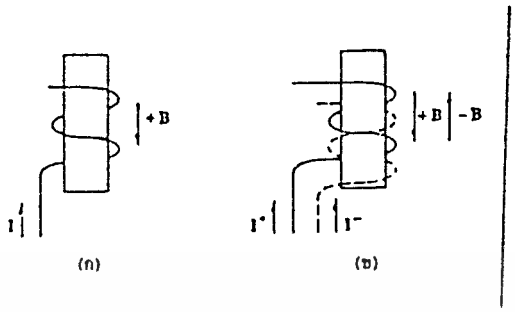
การแก้ปัญหาเพื่อจะขับไฮบริดสเต็ปมอเตอร์ให้ทำงาน ด้วยเพาเวอร์ซัพพลาย เพียงตัวเดียว (unipolar drive) ได้โดยดัดแปลงโครงสร้างการพันขดลวดเฟส ของสเตเตอร์

การพันขดลวดเฟสของสเตเตอร์แบบ bifilar (การพันแบบสองแกวสลับกัน) สามารถขับได้ด้วย ยูนิโพล่าร์ (unipolar drive)

ขดลวดแบบ unifilar แสดงดังในรูปที่ 5.10 (ก) จะต้องกลับทิศทางของกระแสเพื่อกลับทิศทางของเส้นแรงแม่เหล็ก B

ขดลวดแบบ bifilar แสดงในรูปที่ 5.10 (ข) ถ้าเราต้องการกลับ ทิศทางของเส้นแรงแม่เหล็กเป็น -B สามารถทำได้โดย ป้อนกระแสขนาดเดิมจากเพาเวอร์ซัพพลายตัวเดิมเข้าที่ขดลวดที่เป็นเส้นประในรูปที่ 5.10(ข) ก็จะทำให้ทิศทางการเหนี่ยวนำแม่เหล็กและทิศทางของเส้นแรงแม่เหล็ก (-B) กลับทิศทางได้

ถ้าหาก HSM ในรูปที่ 5.8 มีขดลวดเฟสของสเตเตอร์เป็นแบบ bifilar ขดลวดเฟส $\phi 1$ เดิมจะถูกแบ่งตัวออกเป็นสองขดลวดเฟส $\phi 1+$ และเฟส $\phi 1-$ ขดลวดเฟส $\phi 2$ เดิมจะถูกแบ่งตัวออกเป็นสองขดลวดเฟส $\phi 2+$ และเฟส $\phi 2-$

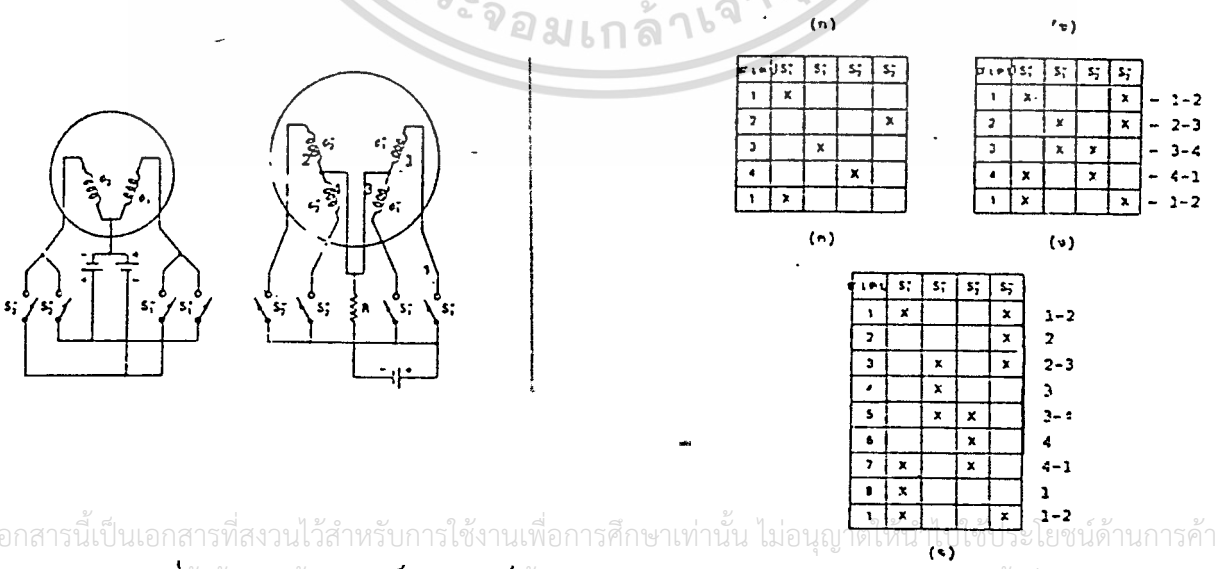


รูปที่ 5.10 การพันขดลวดเฟสของสเตเตอร์

(ก) แบบ unifilar (ข) แบบ bifilar

ในตอนนี้ก็จะทำให้เรา ได้ขดลวดเฟสกิง 4 เฟส และแต่ละเฟสสามารถขับได้ด้วยกระแสที่ไหลใน ทิศทางเดียว ส่วนเครื่องหมาย + และ - ใช้สำหรับแสดงกิงทิศทางการเกิดสนามแม่เหล็กของสเตเตอร์โพล

ในรูปที่ 5.11 (ก) (ข) แสดงวงจรการสวิทช์ 2 วงจร สำหรับ HSM ชนิด 2 เฟส และแบบ 4 เฟส รูป (ค) (ง) และ (จ) แสดงตารางลำดับการขับแบบทีละเฟส และ แบบทีละ 2 เฟส และการขับแบบครึ่งสเต็ปตามลำดับ



รูปที่ 5.11 ไฮบริดสเต็ปมอเตอร์

(ก) HSM แบบ 2 เฟส unifilarจะต้องขับแบบไบโพล่าร์

(ข) HSM แบบ 4 เฟส bifilar ใช้การขับแบบยูนิโพล่าร์

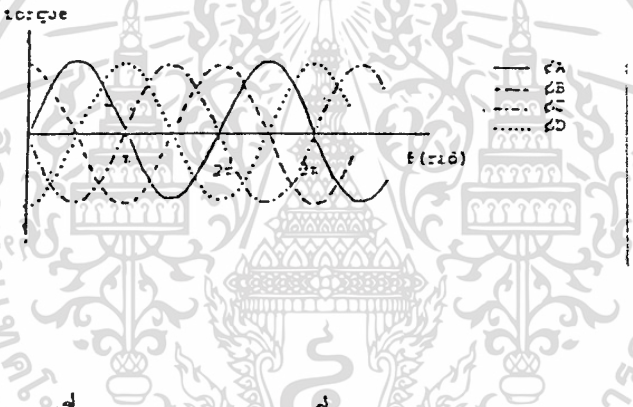
(ค) ตารางแสดงลำดับการขับทีละ 1 เฟส

(ง) ตารางแสดงลำดับการขับทีละ 2 เฟส

(จ) ตาราง แสดงลำดับการขับ แบบครึ่งสเต็ปในทิศทาง CW

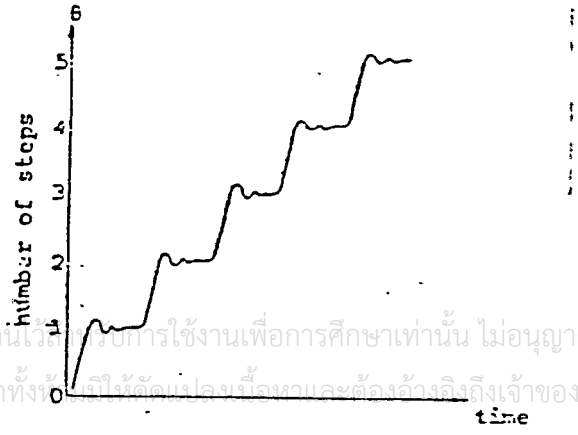
การเพิ่มความละเอียดในแต่ละ STEP ของมอเตอร์

สำหรับ Stepping motor เมื่อให้สัญญาณไฟฟ้าผ่านขดลวดเส้นใดเส้นหนึ่งจะเกิด torque ขึ้น ซึ่ง torque ที่ได้จะมีการเปลี่ยนแปลงไปตามมุมของแกนมอเตอร์ ดังรูปที่ 5.12



รูปที่ 5.12 แสดงการเปลี่ยนแปลงของ torque

จากรูปจะพบว่าเมื่อจ่ายกระแสให้เฟส A,B,C และ D ตามลำดับจะเกิดจุดสมดุลย์ที่มุม (2n-1), (2n-1/2), (2n) และ (2n+1/2) ตามลำดับเช่นกันจากผลดังกล่าวทำให้แกนมอเตอร์มีการหมุนไปทีละ Step และมีโอกาสเกิด osoillate ได้ดังแสดงดังรูปที่ 5.13



รูปที่ 5.13 แสดง single-step response

ผลลัพธ์ที่ได้จากรูปที่ 5.13 จะสังเกตได้ง่ายเมื่อ stepping motor มีขนาดแรงเฉื่อยของโหลดสูงขึ้น โดยเฉพาะ stepping motor ที่มีค่าองศาต่อ step มากเช่น15/step

จากผลดังกล่าว สามารถสรุปผลเบื้องต้นได้ว่า

1. stepping motor มีการหมุนของแกนที่ไม่ต่อเนื่อง เนื่องจากเป็น step
2. stepping motor เกิดการ damp ได้ง่ายและคาบการ damp จะยาวขึ้นเมื่อมีแรงเฉื่อยมาก

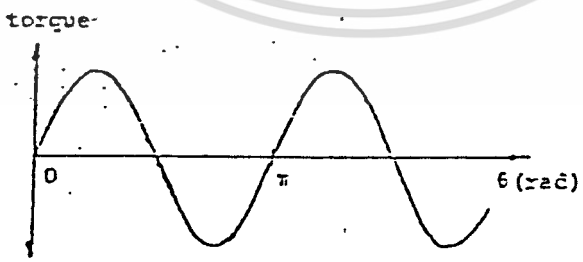
ในการแก้ปัญหานี้ สามารถแก้ได้โดยการลดขนาด องศา/step ของ stepping motor ให้น้อยลง ซึ่งสามารถกระทำได้ 2 วิธี คือ

1. พัฒนาทางด้านโครงสร้างของ stepping motor
2. พัฒนาทางด้าน control system

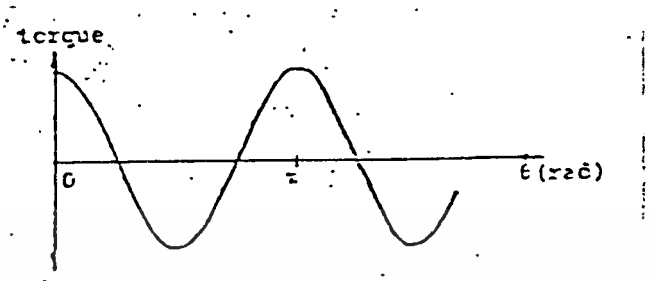
ในกรณีนี้ การพัฒนาทางด้าน control system จะสะดวกกว่าอีกทั้งสามารถนำไปประยุกต์ใช้กับ stepping motor ขนาดและชนิดอื่น ๆ ได้

<u>หลักการ</u>

สำหรับ permanent and hybrid stepping motorนั้นเมื่อมีการจ่ายกระแสไฟฟ้า Ix ผ่านเฟส A จะได้ความสัมพันธ์ดังรูปที่ 5.14



รูปที่ 5.14 แสดงความสัมพันธ์ระหว่าง torque กับ degree เมื่อมีกระแสไหลผ่านเฟส A เช่นเดียวกันเมื่อมีกระแส Ix ผ่านเฟส B จะได้ความสัมพันธ์ใหม่ดังรูปที่ 5.17



รูปที่ 5.15 แสดงความสัมพันธ์ระหว่าง torque กับ degree
เมื่อมีกระแส Im ผ่านเฟส B

จากความสัมพันธ์ดังกล่าวสามารถเขียนรูปแบบสมการเบื้องต้นได้ว่า

สำหรับเฟส A torque A = - pn ϕ i_a Sin pθ ----(1)

สำหรับเฟส B torque B = - pn ϕ i_b Sin p(θ-) ----(2)

pn = number of pairs of magnetic poles

ϕ = peak fluk linkage

ia, ib = กระแสที่ไหลผ่านเฟส A และ B ตามลำดับ

θ = ตำแหน่งมุมปัจจุบันของแกนโรเตอร์

= มุมระหว่างเฟส A และ B

จากข้อมูลที่ได้จากกราฟในรูปที่ 5.14 และ 5.15 นั้นจะพบว่าจุดสมดุลย์จะมีการเคลื่อนตำแหน่งไปทีละ step ซึ่งเป็นผลให้เกิดการเคลื่อนที่เป็น step ขึ้น ดังนั้นการที่จะ ทำให้แกนมอเตอร์ มีการหมุนที่ต่อเนื่องมากขึ้น จึงต้องหาทางปรับปรุง ช่วงกว้างระหว่างจุดสมดุลย์ที่อยู่ใกล้กัน 2 จุด ให้มีขนาดแคบลง ซึ่งจะเป็นการลดความกว้างของstep เช่นกัน ผลที่ได้จะทำให้จำนวน step/รอบ ของมอเตอร์มีมากขึ้น ทำให้มอเตอร์สามารถอ้างอิงตำแหน่งของมอเตอร์ได้ละเอียดขึ้น

พิจารณา สมการที่ 1 และ 2 นำมาประยุกต์เขียนรูปแบบสมการง่ายๆ ขึ้นใหม่โดยกำหนดค่า p = 1 และ = 90 องศา จะได้ว่า

torque A = K ia sin θ ----(3)

torque B = K ib sin (θ-π/2) ----(4)

$K = -pn\phi$ (ซึ่งเป็นค่าคงที่)

ถ้าให้ i_a และ i_b มีค่าเท่ากับกระแส imax จะได้ผลลัพธ์ของ torqueดังสมการ

torque รวม = torque A + torque B

$= K(i_a \sin\theta + i_b \sin(\theta - \pi/2)) \quad ----(5)$

นำสมการ 5 มาเขียนกราฟจะได้ดังรูปที่ 5.16

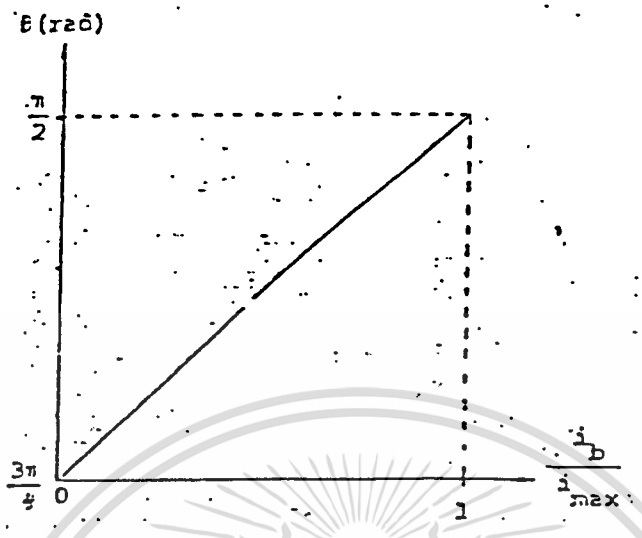
รูปที่ 5.16 แสดงความสัมพันธ์ระหว่าง torque กับ deree

เมื่อ $i_a = it = i_{max}$

จากรูปที่ 5.16 จะพบว่า torque จะมีค่า peak สูงขึ้นและจุดสมดุลย์จะอยู่ระหว่าง $\pi/2$ กับ π คือ $3\pi/4$ ซึ่งเป็นตำแหน่งกึ่งกลางพอดี ดังนั้นแสดงว่าการขับ มอเตอร์ แบบ one-two excitiation จะเพิ่มจำนวน step/รอบ ขึ้นเป็น 2 เท่า ซึ่งเรียกกันว่า "การขับแบบ half step"

พิจารณา สมการที่ 5 จะพบว่า ถ้า i_a มีค่าคงที่เท่ากับ i_{max} และ i_b มีค่าระหว่าง o ถึง i_{max} จะพบว่า จุดสมดุลย์จะมีการเปลี่ยนแปลงระหว่าง θ กับ $\theta - \frac{\pi}{2}$ ซึ่งสามารถหาจุดสมดุลย์ที่เปลี่ยนแปลงตาม i_b ได้ดังนี้

จากสมการที่ 5 มาเขียนกราฟโดยปรับปรุงค่า θ ที่ได้ให้อยู่ในช่วง $\pi/2$ ถึงจะได้ดังรูปที่ 5.17



รูปที่ 5.17 แสดงความสัมพันธ์ระหว่าง ib/imax กับ θ

จากกราฟ จะพบว่าจุดสมดุลย์มีการเปลี่ยนแปลงตำแหน่งมุมตาม ib/imax แสดงว่าในทางปฏิบัติ หากสามารถควบคุมค่า ib/imax ได้ ย่อมมีทางที่จะควบคุมตำแหน่งจุดสมดุลย์ได้เช่นกัน

บทที่ 6

การนำมาประยุกต์ใช้งาน

6.1 บทนำ

การนำมาประยุกต์ใช้งานของ 8032AH ในที่นี้จะนำเอา ไมโครคอนโทรลเลอร์เบอร์ 8032 AH BASIC มาใช้งานร่วมกับสเต็ปปิ้งมอเตอร์ ทั้งหมด 8 ตัว โดยแบ่งออกเป็น 2 ภาคคือ ภาค HARD WARE และ ภาค SOFT WARE ของทั้งหมด

ถ้าจะกล่าวถึง การนำเอา 8032 มาควบคุมการทำงานของ สเต็ปปิ้งมอเตอร์แล้วความสามารถในการควบคุมการทำงานนั้น ก็มีความสามารถพอ ๆกับ Z-80 เพราะสามารถสั่งงานให้สเต็ปปิ้งมอเตอร์หมุนซ้าย หมุนขวา และกำหนดจำนวนรอบในการหมุนก็ได้ ภาษาที่ใช้ในการเขียนโปรแกรมควบคุมนั้นก็ง่ายต่อการศึกษาและทำความเข้าใจ เพราะใช้ภาษา BASIC เป็นภาษาที่ใช้ในการควบคุมซึ่งต่างจาก Z-80 จะใช้ภาษาแอสเซมบลี้ในการเขียนโปรแกรมซึ่งทำให้เกิดความยุ่งยากในการตรวจสอบแก้ไข เมื่อมีข้อผิดพลาดเกิดขึ้นสำหรับผู้ที่ไม่เคยศึกษาภาษาแอสเซมบลี้มาก่อนก็จะไม่สามารถเขียนโปรแกรมได้ แต่ภาษาเบสิกนั้นง่ายแก่การศึกษาและทำความเข้าใจถึงไม่ต้องมีพื้นฐานมาก่อนเลยก็สามารถที่จะเขียนโปรแกรมได้

6.2 ภาค HARD WARE

ภาคนี้จะเป็นวิธีการนำเอาชุดไดนโทรลเลอร์มาต่อร่วมกับวงจรควบคุมการติดต่อ และวงจรควบคุมสเต็ปปิ้งมอเตอร์

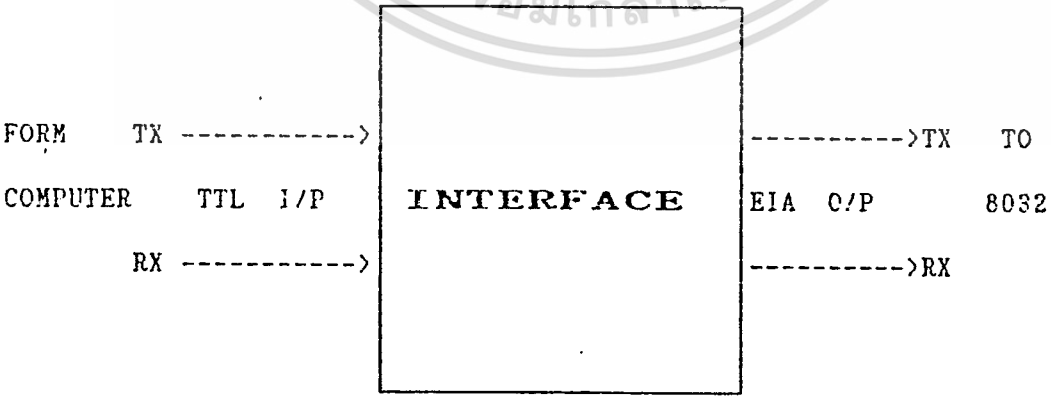
6.2.1 วงจรของชุดโคนโทรลเลอร์ 8032 AH BASIC

จากรูปจะเห็นว่าในวงจรของชุดคอนโทรลเลอร์นี้จะมีพอร์ทที่ใช้งานอยู่ทั้งหมด 4 พอร์ทแต่เราใช้แค่พอร์ท 1 เพียงแค่ พอร์ทเดียวเราก็สามารถนำเอามาควบคุมสเต็ปปิ้งมอเตอร์ถึง 8 ตัว ที่ใช้พอร์ท 1 ก็เพราะการเขียนโปรแกรมจะง่ายกว่า CPU ในตระกูล MCS-51 นี้จะไม่มีสัญญาณ IORQ เหมือนกับ Z-80 ดังนั้นในการออกแบบจึงต้องทำการสร้างสัญญาณ IORQ เทียมขึ้นมาใช้งาน โดยการทำพอร์ทให้เป็นเหมือนหน่วยความจำ ในการติดต่อกับพอร์ทนั้นให้อ้างไปที่หน่วยความจำตำแหน่ง E000H จะทำให้เกิดสัญญาณ IORQ เทียมขึ้น เบอร์พอร์ทจะถูกดีโค้ดด้วยไบท์ต่ำ ฉะนั้น ชื่อพอร์ทที่ MCS-51 มองเห็นจริงมีชื่อเป็น E0xxH ซึ่ง xx ก็คือแอดเดรสนั่นเอง

6.2.2 การรับ-ส่งข้อมูล

รูปแบบของข้อมูลที่ 8032 ใช้ในการติดต่อรับ-ส่งข้อมูลกับเทอร์มินอลอยู่ในโหมด 8-BIT UART (Universal Synchronous Asynchronous Receiver Transmitter) ซึ่งมี 8 ดาต้าบิต,1 สตาร์ทบิต และ 1 สต็อปบิต ไม่มีพาริตี้บิต

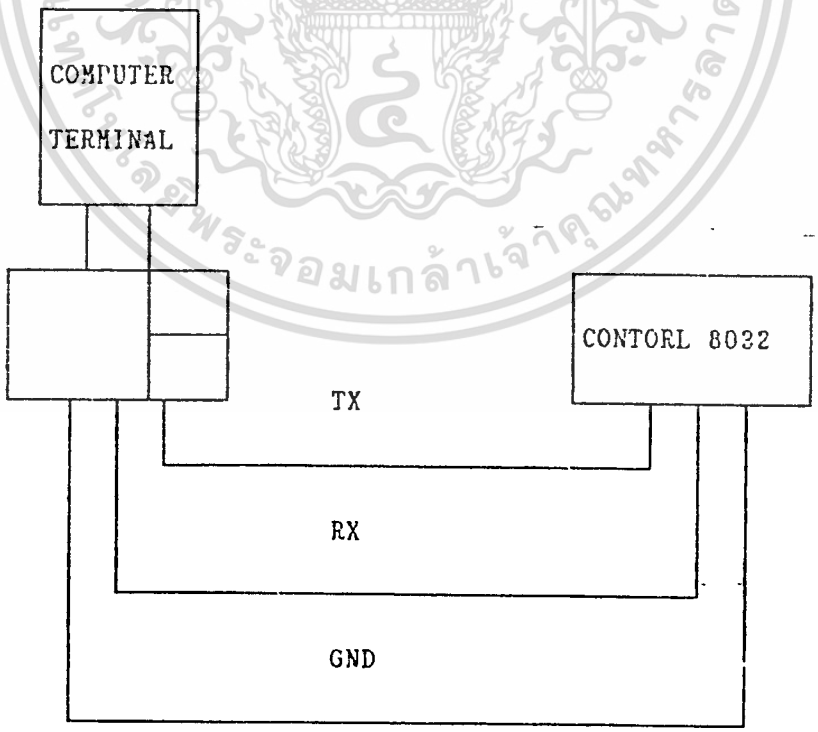
พอร์ทอนุกรมของ 8032 เป็นสัญญาณแบบ TTL จึงไม่สามารถนำมาต่อเข้าโดยตรงกับพอร์ทอนุกรมมาตรฐาน RS 232C ของเทอร์มินอลได้ จึงต้องมีวงจรอินเตอร์เฟสเพื่อแปลงสัญญาณ TTL เป็นสัญญาณ EIA ตามมาตรฐาน RS 232C



รูปที่ 6.2 Inerface Translator Signal

6.2.3 การเชื่อมต่อกับเทอร์มินอล (PC)

การเขียนโปรแกรมภาษาเบสิกจำเป็นต้องมี จอภาพ และ คีย์บอร์ด เพื่อใช้ในการเขียน แก้ไข และ ทดสอบโปรแกรม ในการทดลองใช้งานระบบ 8032 โดยใช้เครื่องคอมพิวเตอร์ IBM PC/XT เป็นเทอร์มินอลให้กับระบบ การสื่อสารข้อมูลโดยผ่านพอร์ทสื่อสารอนุกรม RS 232C ซึ่งอยู่ในการ์ดมัลติ I/O ที่ติดตั้งอยู่บนสล็อตของเครื่องไมโครคอมพิวเตอร์ และใช้โปรแกรมสื่อสารข้อมูลคือ PROCOM PLUS เป็นโปรแกรมควบคุมการสื่อสารข้อมูล โดยสามารถนำข้อมูล ที่ถูกส่งมาจากชุดคอนโทรล 8032 มาแสดงที่หน้าจอภาพได้ และสามารถส่งข้อมูลจากคีย์บอร์ดของเครื่องไมโครคอมพิวเตอร์มาให้กับชุดคอนโทรล 8032 เช่นกัน ข้อมูลที่รับ-ส่งนั้นจะอยู่ในรูปของรหัส ASCII การเชื่อมต่อระหว่างพอร์ทของ RS 232C ของเครื่องไมโครคอมพิวเตอร์กับชุดคอนโทรล 8032 นั้นจะมีสายสัญญาณเพียง 3 เส้นเท่านั้น คือ TX,RX และ GND ดังรูป 6.3



รูป 6.3 การต่อชุดคอนโทรล 8032 เข้ากับ ไมโครคอมพิวเตอร์

6.2.4 การทดลองการทำงานชุดคอนโทรล 8032

ในการทดลองการทำงานนี้จะทดลองระหว่างชุดคอนโทรล 8032 กับไมโครคอมพิวเตอร์ ก่อนโดยการต่อเข้าด้วยกันทางพอร์ท RS 232C หลังเครื่องไมโครคอมพิวเตอร์ จะเป็นขั้วต่อแบบ DB 25 ตัวผู้ เมื่อต่อเข้ากันเรียบร้อยแล้วก็ลองเปิดเครื่องไมโครคอมพิวเตอร์ เริ่มใช้โปรแกรม PROCOM PLUS โดยคีย์คำว่า PCP เข้าไป

```
A:\>PCP      <ENTER>
```

เมื่อเข้าสู่โปรแกรมแล้ว เครื่องไมโครคอมพิวเตอร์ก็พร้อมที่จะทำหน้าที่เป็นเทอร์มินอลให้กับชุดคอนโทรล 8032 ที่บอร์ดชุดคอลโทรลจะมีสวิทช์ RESET อยู่ให้กดสวิทช์ RESET ที่บอร์ดและจากนั้นให้กด SPACE BAR ที่คีย์บอร์ด ข้อความนี้จะปรากฏที่มุมซ้ายบนสุดของหน้าจอทันทีให้ไม่มีอะไรเกิดขึ้นให้ทำการตรวจเช็คสายสัญญาณระหว่างบอร์ดชุดคอนโทรล กับเครื่องไมโครคอมพิวเตอร์ ว่าสายสัญญาณสลับกันหรือเปล่าถ้าทุกอย่างถูกต้องแสดงว่า พอร์ทที่ต่อออกมาใช้งานนั้นไม่ใช่พอร์ทที่ชุดคอนโทรลต้องการเพราะในการ์ดมัลติ I/O นั้นมีพอร์ทอนุกรมอยู่ 2 พอร์ทคือ 2F8 และ 3F8 พอร์ทที่ชุดคอนโทรลต้องการคือ 2F8 ให้ทำการสลับสายพอร์ทที่การ์ดมัลติ I/O ในเครื่องไมโครคอมพิวเตอร์แล้วลองทำใหม่อีกครั้งถ้ายังไม่ออกให้ดูที่บอร์ดว่ามี RAM หรือเปล่าถ้ามีให้ตรวจเช็คอุปกรณ์ตัวอื่นต่อไป เช่น ROM ตัวแปลภาษาเบสิกอาจจะเสียก็ได้ให้ลองเปลี่ยนอุปกรณ์ตัวอื่นๆดู

```
A:\>PCP      <ENTER>

* MCS-51(tm) BASIC V1.1 *
READY
>
>
>
```

จากนั้นให้ทดลองเขียนโปรแกรมสั้นๆเพื่อทดสอบการทำงานของชุดคอนโทรล 8032 ดู

```
READY
>10 FOR I=1 TO 5
>20 PRINT I
>30 NEXT I
>40 END
```

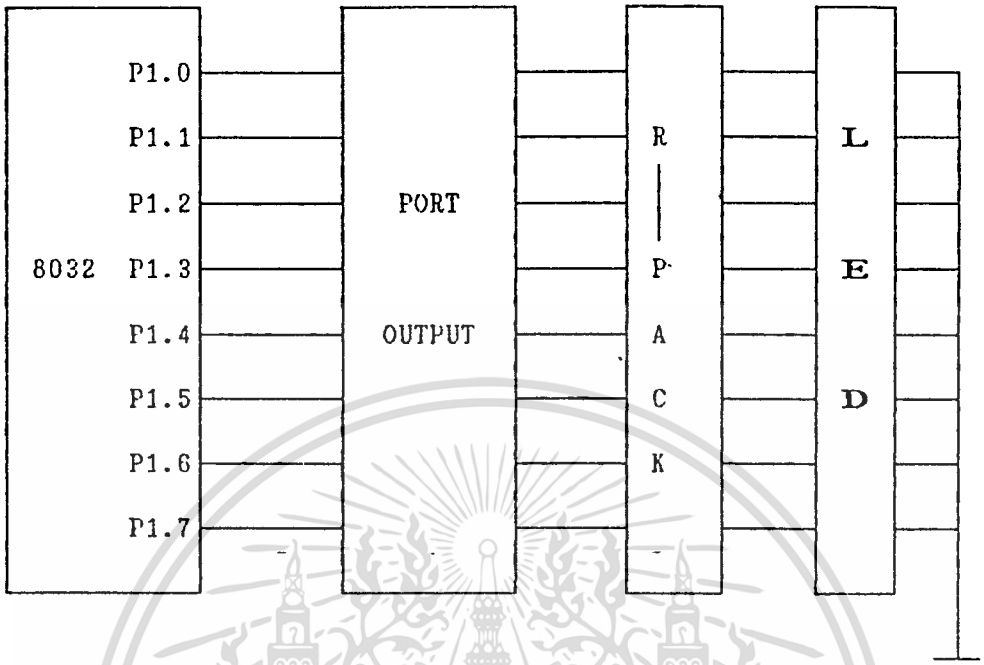
ระหว่างการเขียนโปรแกรมถ้าเขียนผิดจะต้องเขียนบรรทัดนั้นใหม่ทั้งหมด เมื่อเขียนเสร็จแล้วให้ลองทดสอบ โดยใช้คำสั่ง RUN แล้วดูผลที่หน้าจอ

```
>RUN
1
2
3
4
5
READY
>
```

ถ้าผลออกมาดังตัวอย่างที่เห็นแสดงว่าการเชื่อมต่อชุดคอนโทรล 8032 กับ ไมโครคอมพิวเตอร์เรียบร้อยดีและ ชุดคอนโทรลพร้อมที่จะใช้งานได้แล้ว

6.2.5 การทดลองใช้งานพอร์ทของชุดคอนโทรล

ในตัวของไมโครคอนโทรลเลอร์ตระกูล MCS-51นั้นจะแบ่งพอร์ทที่ใช้งานออกเป็น 4 พอร์ท คือ PORT0, PORT1, PORT2, PORT3 แต่ในที่เราจะใช้งานที่ PORT1 เพียงพอร์ทเดียวเท่านั้น ซึ่งมีคำสั่งที่ใช้งานคือ PORT1=n โดยให้ n เป็นจำนวนตัวเลขที่ต้องการจะส่งค่าออกพอร์ท หลังจากที่เราทำการกดสวิทช์ RESET ที่ชุดคอนโทรลและกด SPACE BAR แล้ว ที่PORT1 จะมีค่าเท่ากับ 255 ฐานสิบ หรือ FFH ฐานสิบหก คือทุกบิทของ PORT1 จะมีค่าเป็นลอจิก "1" หมด ถ้าต้องการจะให้ PORT1 เป็นลอจิก "0" ก็กำหนดได้โดยเขียนคำสั่งลงไปคือ PORT1=0 แล้วกด ENTER ที่ PORT1 ก็จะมีค่าเป็น "0" ทุกบิท เราสามารถทดลองและตรวจสอบการส่งค่าต่างๆ ออกพอร์ทได้ โดยให้ต่อชุด LED เพิ่มดังรูป



รูป 6.4 การนำเอา LED มาต่อที่ PORT1 เพื่อทดลองส่งค่าออกพอร์ท

```
>PORT1=00              <ENTER>
```

เป็นการเขียนค่า 00H (00000000b)ให้แก่พอร์ท 1 ผลที่ได้คือ LED จะดับหมดทุกดวง

```
>PORT1=170   (หรือ 0AAH ก็ได้ค่า)        <ENTER>
```

การเขียนค่า AAH (10101010b)ให้แก่พอร์ท 1 ผลที่ได้คือ LED จะติดดวงเว้นดวง

```
>PRINT PORT1           <ENTER>
170
```

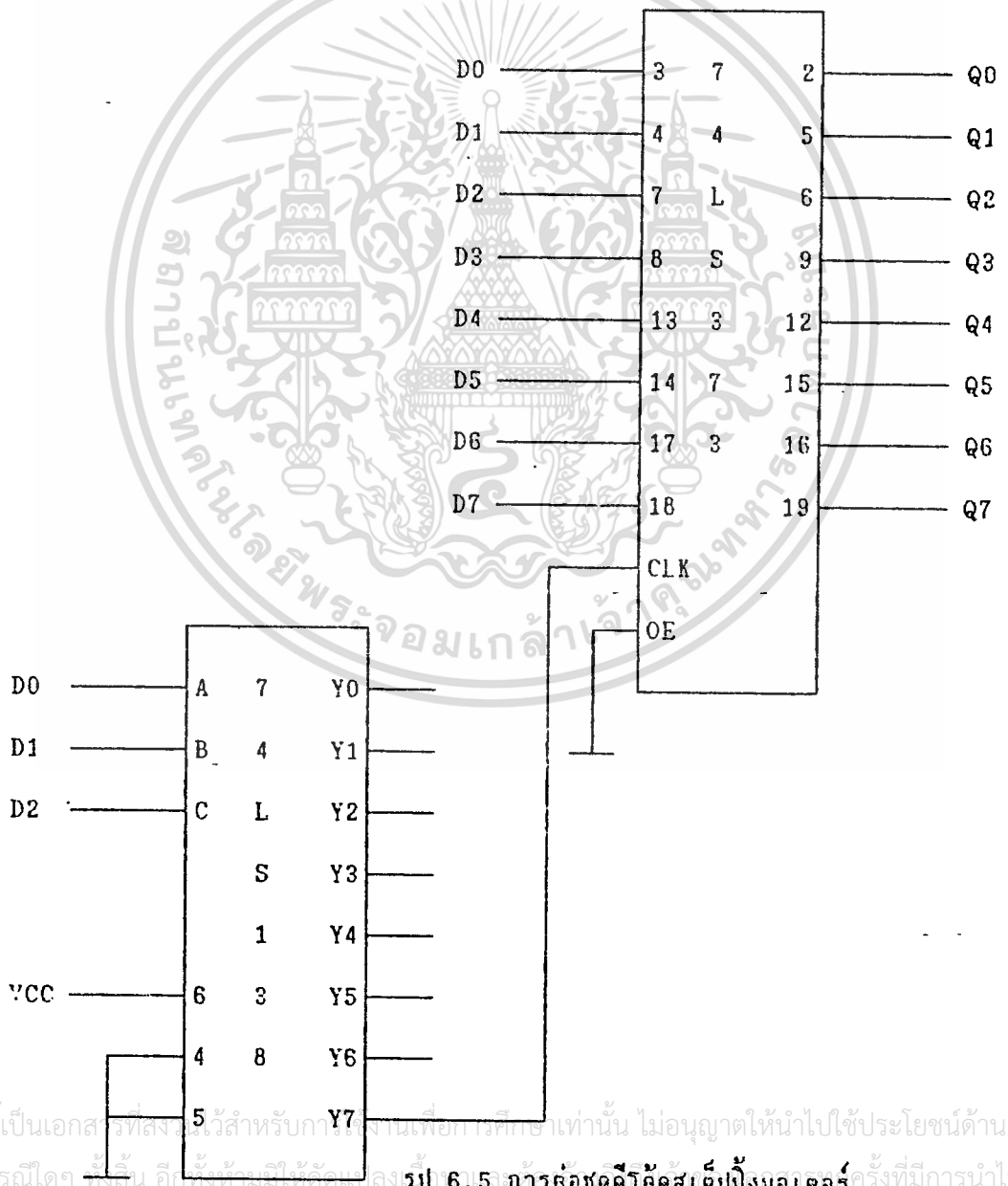
เป็นการอ่านค่าจากพอร์ท 1 แสดงค่าเป็นเลขฐานสิบ

```
>PH0.PORT1             <ENTER>
AAH
```

เป็นการอ่านค่าจากพอร์ท 1 แสดงค่าเป็นเลขฐานสิบหก

6.2.6 การออกแบบชุดดีโค้ดสเต็ปปิ้งมอเตอร์

ที่ต้องทำการออกแบบ ชุดดีโค้ดสเต็ปปิ้งมอเตอร์ก็เพราะว่า ในโครงงานนี้มีสเต็ปปิ้งมอเตอร์ทั้งหมด 8 ตัวจึงต้องการเอามาเลือกการทำงานของสเต็ปปิ้งมอเตอร์แต่ละตัวแต่ถ้าใช้เพียงตัวเดียว ก็ไม่ต้องทำชุดดีโค้ดนี้ เพราะมีการสั่งงานทางโปรแกรมที่ง่ายขึ้น ชุดนี้จะเป็นชุดที่ทำการติดต่อระหว่าง ไมโครคอนโทรลเลอร์ 8032 กับชุดขับสเต็ปปิ้ง มอเตอร์ซึ่งจะทำหน้าที่เป็น พอร์ทเอ้าท์พุท และ อิทพุท โดยจะรับข้อมูลจาก 8032 เป็น DATA แล้วส่งออกไปที่ชุดขับให้ทำตามโปรแกมที่เขียนดังรูป 6.5



รูป 6.5 การต่อชุดดีโค้ดสเต็ปปิ้งมอเตอร์

การทำงานของวงจรชุดดีโค้ดนี้จะใช้ 74LS138 มาเป็นตัวดีโค้ดไปทริกขา CLOCK ของ 74LS373 ทั้ง 8 ตัว

6.2.7 การออกแบบและทดลองชุดขับสเต็ปปิ้งมอเตอร์

รูป 6.6 วงจรจ่ายไฟของทั้ง ชุดคอนโทรล 8032 ชุดอิเลอร์เฟส และชุดขับสเต็ปปิ้งมอเตอร์

รูป 6.7 วงจรขับสเต็ปปิ้งมอเตอร์

รูป 6.8 วงจรอินเตอร์เฟสสเต็ปปิ้งมอเตอร์

การออกแบบชุดขับสเต็ปปิ้งมอเตอร์จะเลือกเอา TRANSISTOR เบอร์ 2N2222 มาใช้ร่วมกับ เบอร์ 2SC1061 ซึ่งขับกระแสได้สูงถึง 800 มิลลิแอมป์ และตัวสเต็ปปิ้งมอเตอร์ที่ใช้ จะเลือกเอาที่ใช้แรงดันเพียงแค่ 12 โวลท์ 500 มิลลิแอมป์เพราะเป็นเพียงแค่ชุดทดลองต้นแบบเท่านั้น ถ้าหากจะนำมาใช้งานจริงก็ต้องทำการออกแบบชุดขับใหม่เพื่อที่จะนำไปขับสเต็ปปิ้งมอเตอร์ตัวใหญ่ๆให้ทำงานได้โดยที่ไม่ทำให้ TRANSISTOR ขับนั้น ร้อนเกินไป และในการออกแบบควรติดแผ่นระบายความร้อนให้กับ TRANSISTOR ด้วย

เมื่อต่อวงจรขับสเต็ปปิ้งมอเตอร์จากรูป 6.6 แล้ว ก็ทำการทดลองของวงจรด้วยวิธีง่ายๆ โดยการเอามือแตะที่ R INPUT ของแต่ละแกน R1 แล้วเปลี่ยนมาที่ R2,R3 และ R4 ตามลำดับ แล้วสังเกตุที่ LED จะเห็นว่าติดเรียงกันไป เมื่อเอามือแตะที่ R INPUT ตัวไหน LED ตัวนั้นจะติดตามที่เราเอามือแตะ

6.3 การออกแบบและเขียนโปรแกรมควบคุมการทำงาน

ในส่วนนี้จะเป็นเรื่องของซอร์ทแวร์ทั้งโดยจะใช้ ภาษาเบสิกมาเขียนเป็น โปรแกรมควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ทั้ง 8 ตัว การเขียนโปรแกรมจำเป็นจะต้องใช้คู่มือของไมโครคอนโทรลเลอร์ คือ MCS BASIC-52 จะมีคำสั่งและตัวอย่างการใช้คำสั่งแต่ละคำสั่งรวมไปถึงมีโปรแกรมตัวอย่างด้วยเพื่อความสะดวกในการศึกษาและเขียนโปรแกรม โดยจะเก็บไว้ในรูปของ TOKEN CODE:

TOKEN	KEYWORD
80H	LET
81H	CLEAR
82H	PUSH
83H	GOTO
84H	PWM
85H	PHO
86H	UI
87H	UO

TOKEN	KEYWORD
88H	POP
89H	PRINT
89H	P.
89H	7 (V1.1 ONLY)
8AH	CALL
8BH	DIM
8CH	STRING
8DH	BAUD
8EH	CLOCK
8FH	PH1
90H	STOP
91H	ONTIME
92H	ONE X1
93H	RET I
94H	DO
95H	RESTORE
96H	REM
97H	NEXT
98H	ONERR
99H	ON
9AH	INPUT
9BH	READ
9CH	DATA
9DH	RETURN
9EH	IF
9FH	GOSUB
0A0H	FOR

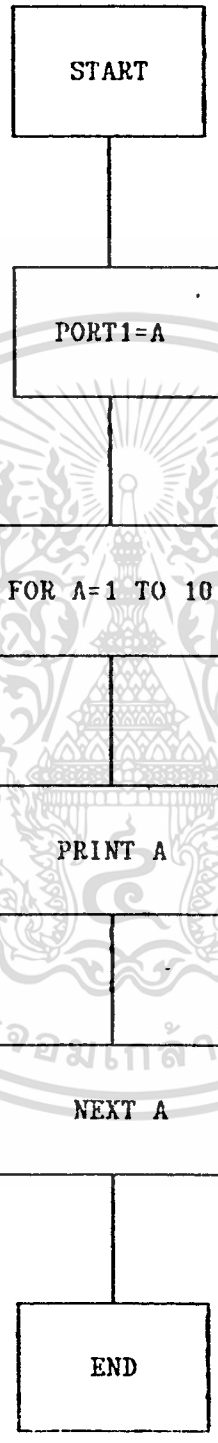
TOKEN	KEYWORD
0A2H	WHILE
0A3H	UNTIL
0A4H	END
0A5H	TAB
0A6H	TO
0A7H	STEP
0A8H	ELSE
0A9H	SPC
0AAH	CR
0ABH	IDLE
0ACH	ST@ (V 1.1 ONLY)
0ADH	LD@ (V 1.1 ONLY)
0AEH	PGM (V 1.1 ONLY)
0AFH	PROM (V 1.1 ONLY)

6.4 การเขียนโปรแกรมทดสอบการทำงาน

หลังจากที่เราได้ทำการออกแบบชุดฮาร์ดแวร์ทั้งหมดแล้วต่อไปนี้ก็จะเป็นวิธีการทดลองเขียนโปรแกรมเพื่อทดสอบการทำงานของฮาร์ดแวร์ทั้งหมด

ตัวอย่างโปรแกรมที่ 1

FLOW CHART



PROGRAM

```
10 PORT1 = A
20 FOR A = 1 TO 10
30 PRINT A
40 NEXT A
50 END
```

อธิบายโปรแกรม

10 กำหนดพอร์ทใช้งานที่จะส่งค่าออก

20 กำหนดค่าที่จะส่งออกตั้งแต่ 1 ถึง 10 โดยส่งทีละ 1

30 แสดงค่าออกที่หน้าจอ

40 ส่งค่าออกพอร์ทต่อไปจนกว่าจะครบถึง 10

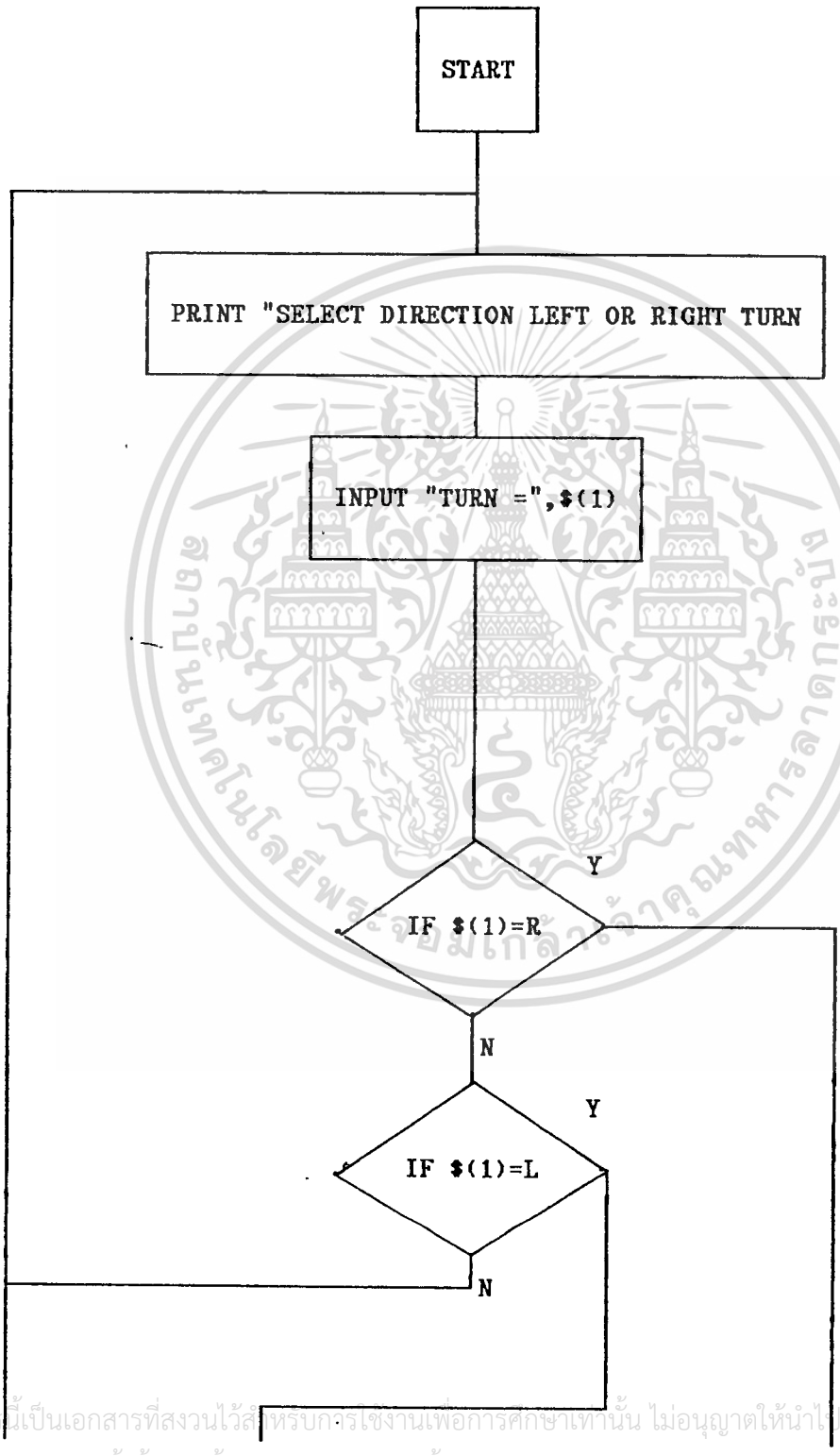
50 จบโปรแกรม

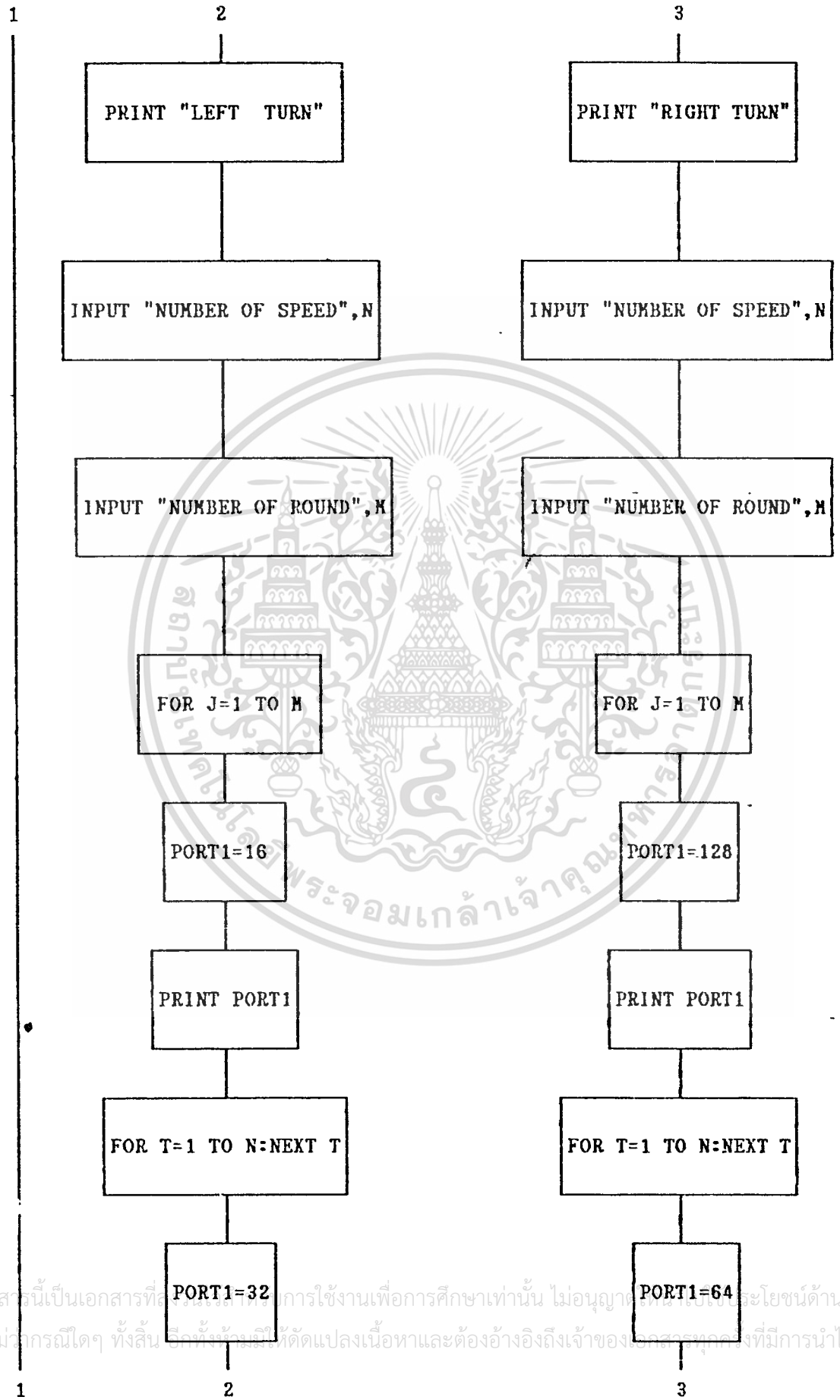
ผลที่ได้หลังจากรัน

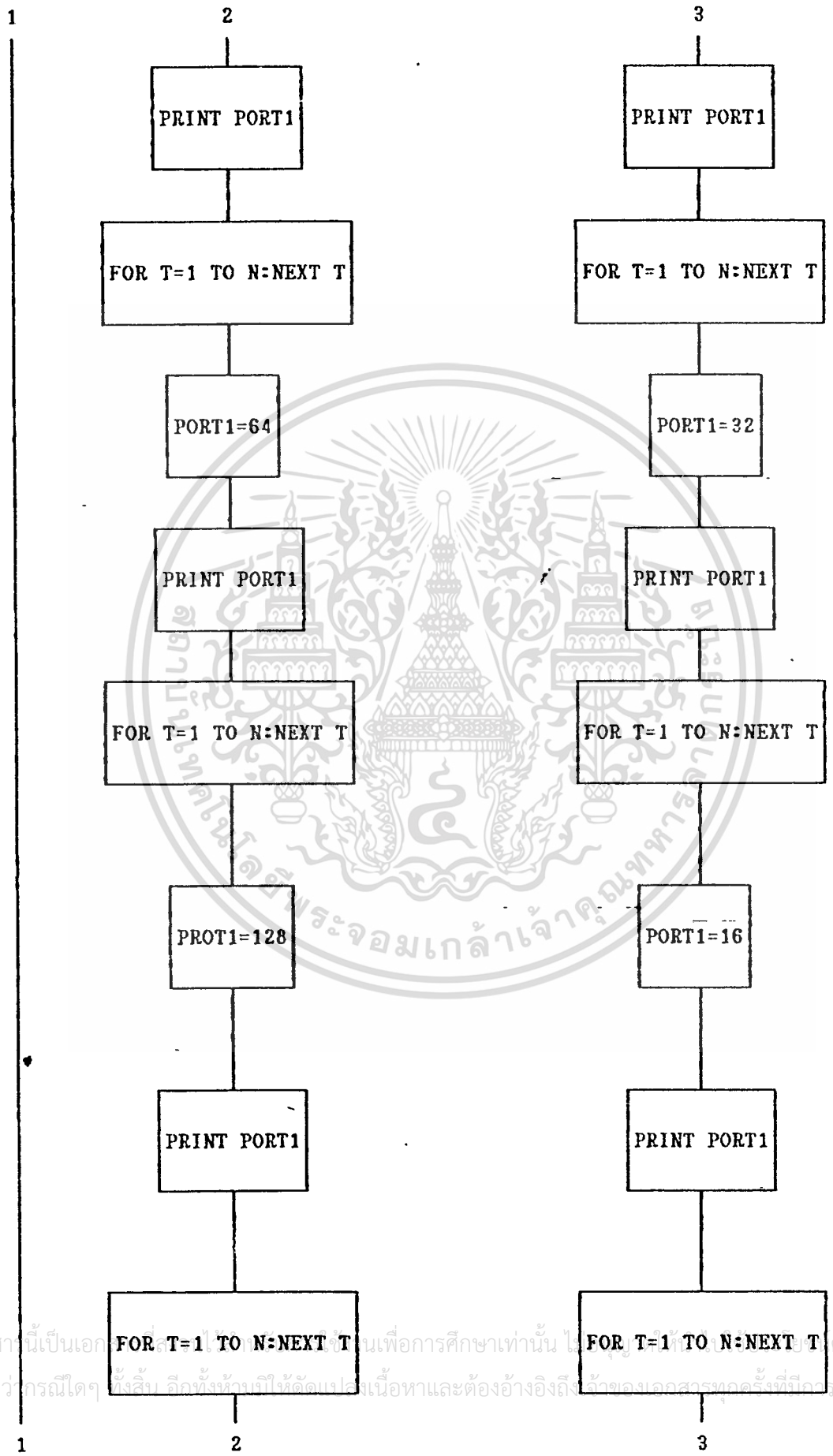
```
1
2
3
4
5
6
7
8
9
10
```

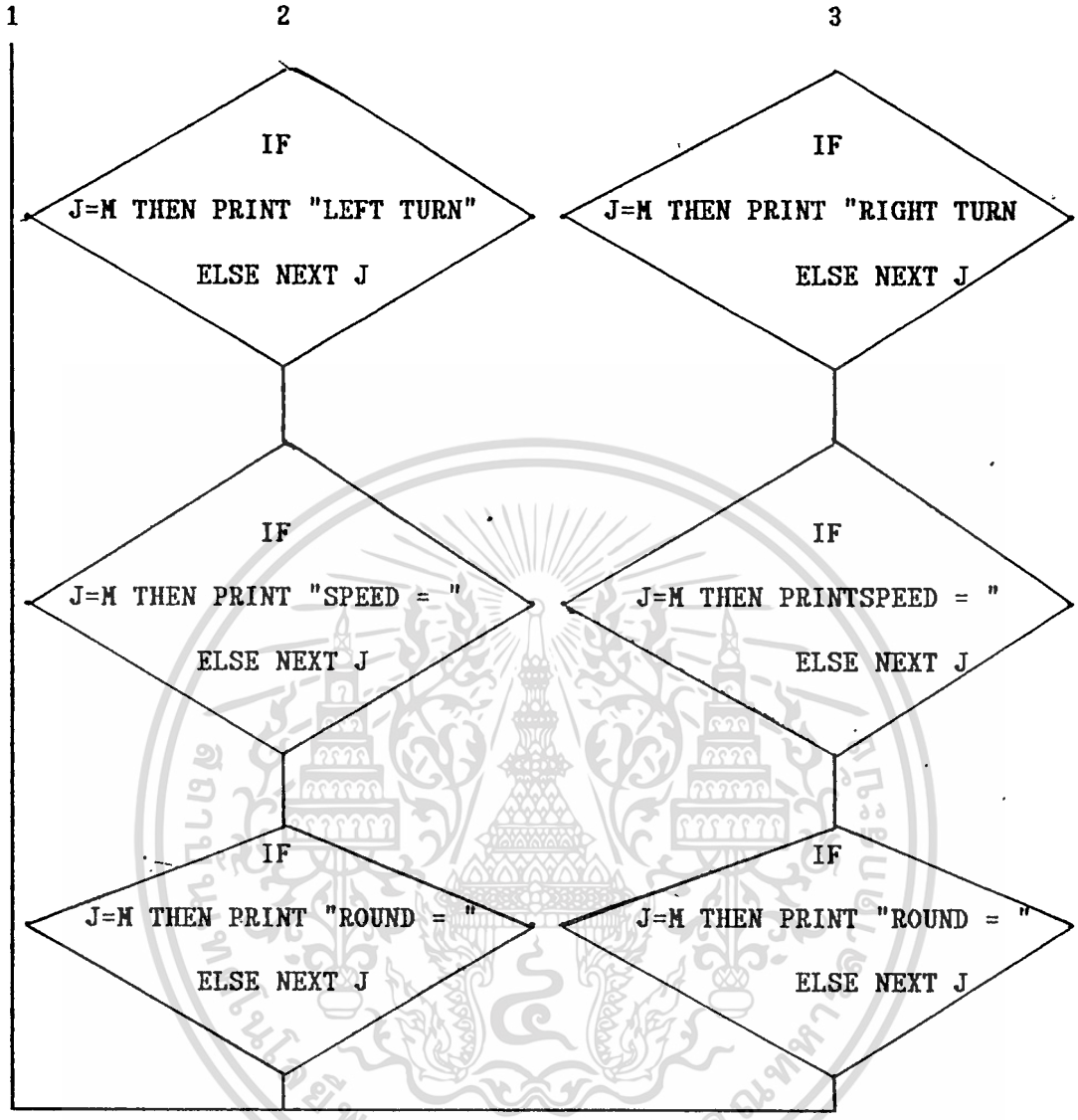
ตัวอย่างโปรแกรมที่ 2

FLOW CHART










```
PROGRAM
10 CLS
20 PRINT "SALECT DIRECTION LEFT OR RIGHT TURN"
30 STRING 100,5
40 INPUT "TURN = ",$(1)
50 IF ASC (($(1),1) = ASC (R) THEN GOTO 1000 ELSE GOTO 60
60 IF ASC (($(1),1) = ASC (L) THEN GOTO 2000 ELSE GOTO 10
70 GOTO 10
1000 REM "STEP MOTOR IS RIGHT TURN"
1010 PRINT "STEP MOTOR RIGHT TURN"
1020 INPUT "NUMBER OF SPEED = ",N
```

```
1030 INPUT "NUMBER OF ROUND = ",M
1040 FOR J=1 TO M
1050 PORT1=128
1060 PRINT PORT1
1070 FOR T=1 TO N:NEXT T
1080 PORT1=64
1090 PRINT PORT1
1100 FOR T=1 TO N:NEXT T
1110 PORT1=32
1120 PRINT PORT1
1130 FOR T=1 TO N:NEXT T
1140 PORT1=16
1150 PRINT PORT1
1160 FOR T=1 TO N:NEXT T
1170 PRINT
1180 IF J=M THEN PRINT "RIGHT TURN" ELSE NEXT J
1190 IF J=M THEN PRINT "SPEED = " ELSE NEXT J
1200 IF J=M THEN PRINT "ROUND = "ELSE NEXT J
1210 GOTO 10
1220 PRINT
2000 REM "STEP MOTOR IS LEFT TURN"
2010 PRINT "STEP MOTOR LEFT TURN"
2020 INPUT "NUMBER OF SPEED = ",N
2030 INPUT "NUMBER OF ROUND = ",M
2040 FOR J=1 TO M
2050 PORT1=16
2060 PRINT PORT1
2070 FOR T=1 TO N:NEXT T
2080 PORT1=32
```

```
2090 PRINT PORT1
2100 FOR T=1 TO N:NEXT T
2110 PORT1=64
2120 PRINT PORT1
2130 FOR T=1 TO N:NEXT T
2140 PORT1=128
2150 PRINT PORT1
2160 FOR T=1 TO N:NEXT T
2170 PRINT
2180 IF J=M THEN PRINT "LEFT TURN" ELSE NEXT J
2190 IF J=M THEN PRINT "SPEED = " ELSE NEXT J
2200 IF J=M THEN PRINT "ROUND = "ELSE NEXT J
2210 GOTO 10
```

อธิบายโปรแกรม

10 เคลียหน้าจอ

20 แสดงที่หน้าจอให้เลือกว่าต้องการหมุนซ้ายหรือขวา

30 กำหนดค่าและขอบเขตของอินพุทตัวอักขระ

40 รับค่าอินพุทเป็นตัวอักขระว่าจะให้หมุนไปทางไหน

50 เช็คค่าอินพุทว่าใช่อักขระตัว R หรือไม่ถ้าใช่ไปบรรทัดที่ 1000 ถ้าไม่ใช่ไปบรรทัดที่60

60 เช็คค่าอินพุทว่าใช่อักขระตัว L หรือไม่ถ้าใช่ไปบรรทัดที่ 2000 ถ้าไม่ใช่ไปบรรทัดที่10

70 กลับไปบรรทัดที่ 10 เพื่อรับค่าใหม่ตามที่กำหนดไว้

1000 หมายเหตุ ว่านี่คือโปรแกรมสั่งให้หมุนขวา

1010 แสดงที่หน้าจอว่าสั่งให้สเต็ปปิ้งมอเตอร์หมุนขวา

1020 รับค่าอินพุทเพื่อกำหนดความเร็ว

1030 รับค่าอินพุทเพื่อกำหนดจำนวนรอบ

1040 เริ่มนับจำนวนรอบ

1050 ส่งค่าออกพอร์ท

```
1060 แสดงค่าที่ส่งออกไปบนหน้าจอ
1070 เริ่มนับความเร็ว
1080 ส่งค่าออกพร์ท
1090 แสดงค่าที่ส่งออกไปบนหน้าจอ
1100 เริ่มนับความเร็ว
1110 ส่งค่าออกพอร์ท
1120 แสดงค่าที่ส่งออกไปบนหน้าจอ
1130 เริ่มนับความเร็ว
1140 ส่งค่าออกพอร์ท
1150 แสดงค่าที่ส่งออกไปบนหน้าจอ
1160 เริ่มนับความเร็ว
1170 เว้นหนึ่งบรรทัด
1180 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "หมุนขวา" ที่หน้าจอ
     ถ้ายังไม่ครบให้ทำต่อไป
1190 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "ความเร็ว ="ที่หน้า
     จอ ถ้ายังไม่ครบให้ทำต่อไป
1200 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "รอบ ="ที่หน้าจอถ้า
     ยังไม่ครบให้ทำต่อไป
1210 กลับไปทำบรรทัดที่ 1 ใหม่
2000 หมายเหตุ ว่านี่คือโปรแกรมสั่งให้หมุนซ้าย
2010 แสดงที่หน้าจอว่าสั่งให้สเต็ปปิ้งมอเตอร์หมุนซ้าย
2020 รับค่าอินพุทเพื่อกำหนดความเร็ว
2030 รับค่าอินพุทเพื่อกำหนดจำนวนรอบ
2040 เริ่มนับจำนวนรอบ
2050 ส่งค่าออกพอร์ท
2060 แสดงค่าที่ส่งออกไปบนหน้าจอ
2070 เริ่มนับความเร็ว
2080 ส่งค่าออกพร์ท
2090 แสดงค่าที่ส่งออกไปบนหน้าจอ
```

2100 เริ่มนับความเร็ว

2110 ส่งค่าออกพอร์ท

2120 แสดงค่าที่ส่งออกไปบนหน้าจอ

2130 เริ่มนับความเร็ว

2140 ส่งค่าออกพอร์ท

2150 แสดงค่าที่ส่งออกไปบนหน้าจอ

2160 เริ่มนับความเร็ว

2170 เว้นหนึ่งบรรทัด

2180 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "หมุนซ้าย" ที่หน้าจอ ถ้ายังไม่ครบให้ทำต่อไป

2190 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "ความเร็ว ="ที่หน้าจอ ถ้ายังไม่ครบให้ทำต่อไป

2200 เช็คค่าว่าครบจำนวนรอบที่ตั้งไว้หรือยังถ้าครบแล้วให้แสดงคำว่า "รอบ ="ที่หน้าจอถ้ายังไม่ครบให้ทำต่อไป

2210 กลับไปทำบรรทัดที่ 1 ใหม่

ผลที่ได้หลังจากรัน

```
SELECT DIRECTION LEFT OR RIGHT TURN
TURN = ?R
STEP MOTOR RIGHT TURN
NUMBER OF SPEED = ? 10
NUMBER OF ROUND = ? 5

128
64
32
16
```

128

64

32

16

128

64

32

16

128

64

32

16

128

64

32

16

RIGHT TURN

SPEED = 10

ROUND = 5

SELECT DIRECTION LEFT OR RIGHT TURN

TURN = ? L

STEP MOTOR LEFT TURN

NUMBER OF SPEED = ? 25

```
NUMBER OF ROUND = 7 3

16
32
64
128

16
32
64
128

16
32
64
128

LEFT TURN
SPEED = 25
ROUND = 3
```

จากผลการทดลองเขียนโปรแกรมจะเห็นได้ว่าเราสามารถสั่งงานให้ สเต็ปปิ้งมอเตอร์ทำงานได้ตามที่เราต้องการไม่ว่าจะให้หมุนซ้าย-ขวา หรือ กำหนดจำนวนรอบ กำหนดความเร็ว กำหนดตัวสเต็ปปิ้งที่จะให้ทำงานตั้งแต่ 1-8

ดังนั้นเราจะเห็นได้ว่าประโยชน์ในการนำเอาไมโครคอนโทรลเลอร์ตระกูล MCS-51 มาประยุกต์ใช้งานสามารถที่จะ นำไปใช้งานได้หลายทางขึ้นอยู่กับความต้องการของงานแต่ละชนิดว่าต้องการจะนำไปใช้ในด้านไหน ร่วมกับอะไร

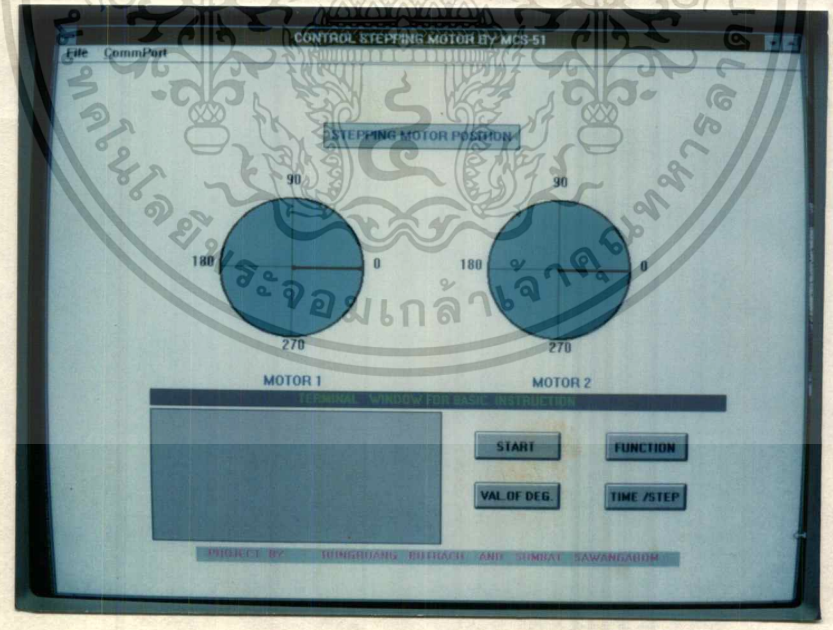
การแสดงกราฟฟิก

การแสดงกราฟฟิกเป็นการโชว์รูปออกมาทางหน้าจอ เพื่อที่จะทำให้มองเห็นภาพว่ามอเตอร์ตัวใดกำลังทำงานอยู่ โปรแกรมที่ใช้ในการเขียนเพื่อให้กราฟฟิกทำงานคือ VISUAL BASIC 3.00

ลำดับการสั่งงานของกราฟฟิก

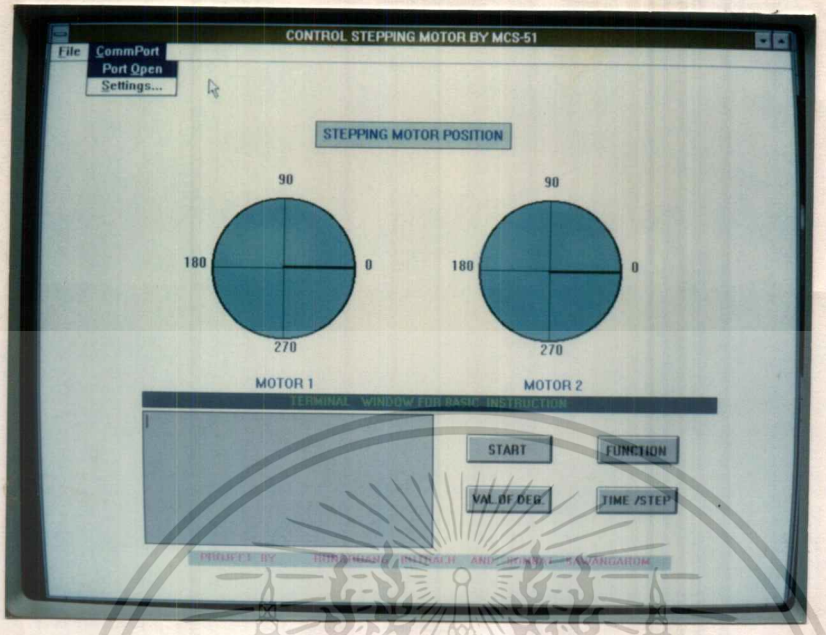
-รันวินโดว์

-ดับเบิ้ลคลิกที่ Stepping Control ไลคอน จอภาพจะโชว์รูปมอเตอร์พร้อมทั้งเมนูต่างๆในการที่จะสั่งรัน ดังแสดงในรูปที่ 6.9



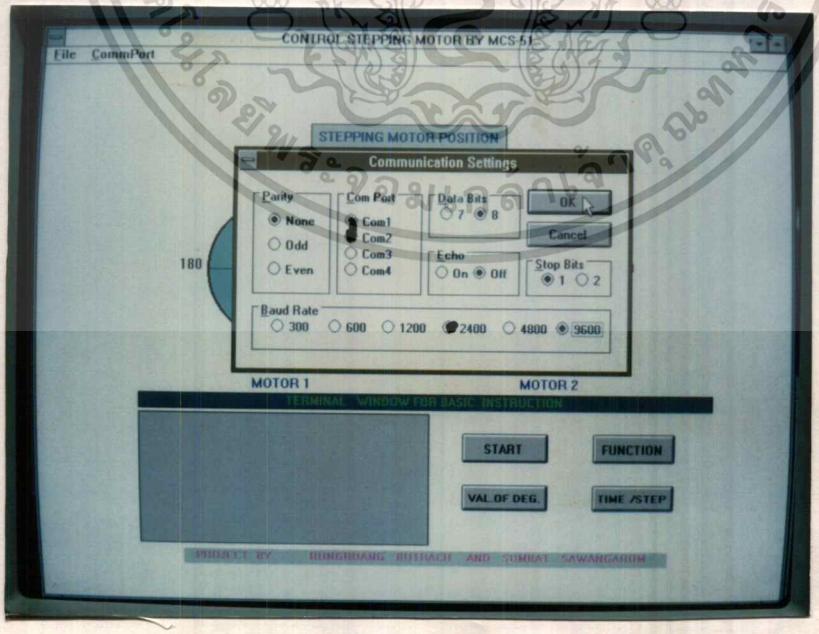
รูปที่ 6.9 แสดงผลจากการดับเบิ้ลคลิกที่ Stepping Control

-ดับเบิ้ลคลิกที่ Comm Port จอภาพจะโชว์ภาพดังรูปที่ 6.10



รูปที่ 6.10 แสดงผลของการคลิกที่ Comm Port

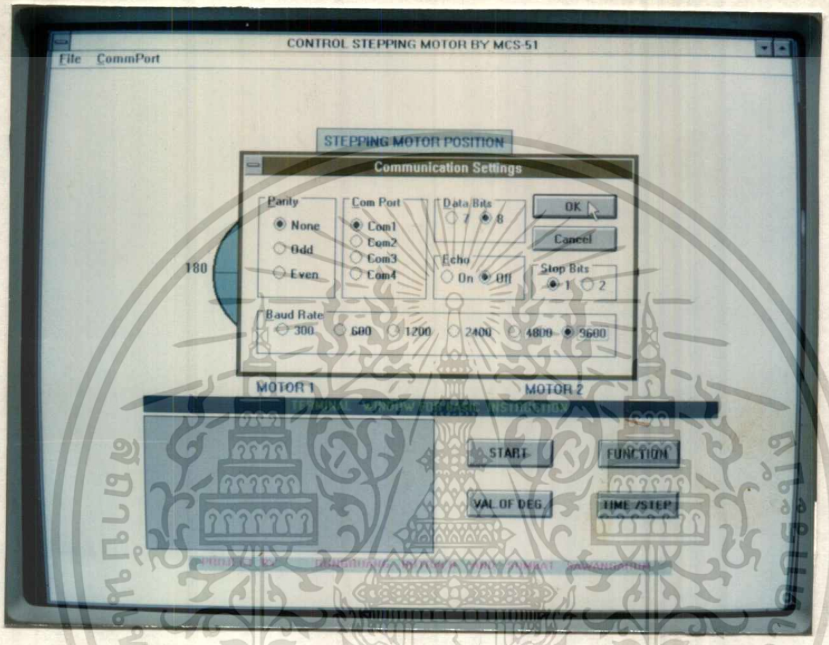
-คลิกที่ Settings.... จอภาพจะแสดงดังรูปที่ 6.11



รูปที่ 6.11แสดงผลของการคลิกที่ Setting....

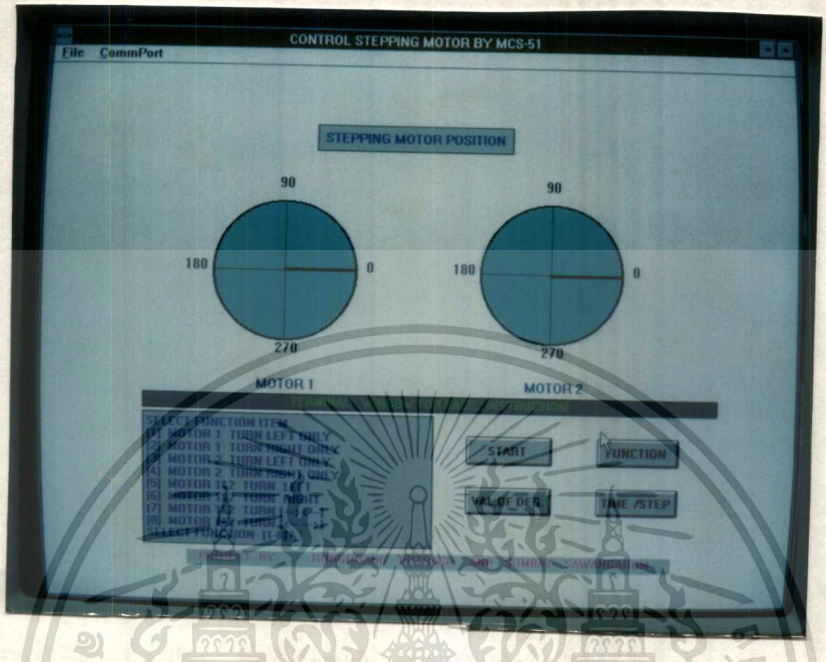
-ทำการเซ็ทพอร์ทโดยคลิกที่ Com1 นั่นแสดงว่าเราเลือก Serial Port

-เลือกอัตราบ๊อดโดยการคลิกที่ 9600 ดังแดงในรูปที่ 6.12 แล้วคลิกที่ OK



รูปที่ 6.12 แสดงการเลือกเซ็ทพอร์ทและการเลือกอัตราบ๊อด

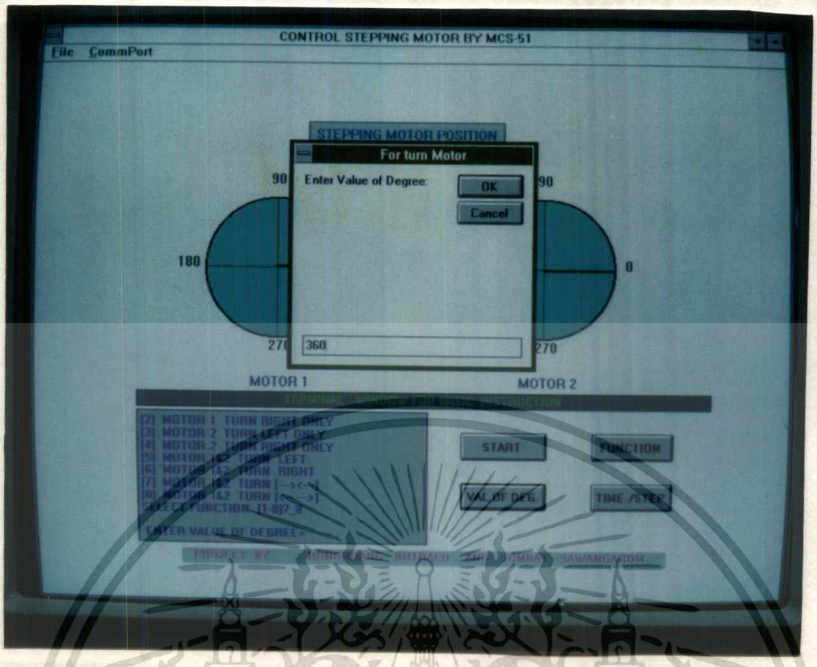
-คลิกที่ START จอภาพจะโชว์ฟังก์ชั่นการทำงาน 8 ฟังก์ชั่น ดังแสดงในรูปที่ 6.13



รูปที่ 6.13 แสดงผลจากการคลิกที่ START

-คลิกที่ FUNCTION จอภาพจะมีข้อความบอกให้เลือกฟังชั่น SELECT FUNCTION [1-8]?

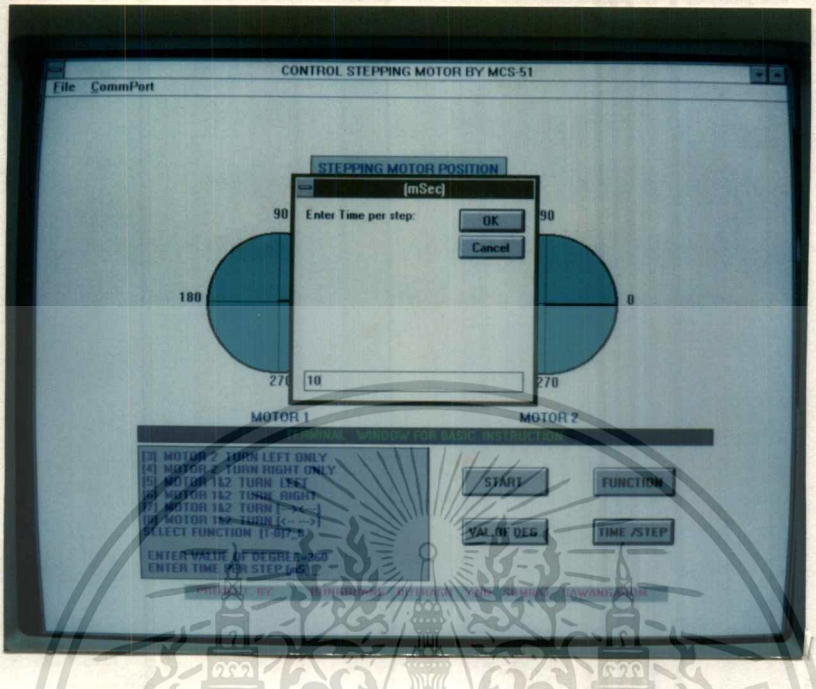
-คลิกที่ VAL. OF DEG. จอภาพจะมีข้อความให้เราเลือกจำนวนองศาที่จะให้มอเตอร์หมุนไป ดังแสดงในรูปที่ 6.14



รูปที่6.14 แสดงผลจากการคลิกที่ VAL.OF DEG.

ค่าของจำนวนองศา ควรจะเป็นค่าที่ 1.8 หารลงตัวเพราะมอเตอร์เป็นแบบ 1.8 องศาต่อสเต็ป เช่นค่า 9,18,90,... เป็นต้น เมื่อทำการใส่ค่าองศาเรียบร้อยแล้วทำการคลิกที่ OK

-คลิกที่ TIME/STEP จอภาพจะแสดงข้อความให้เราใส่ค่าเวลาต่อหนึ่งสเต็ป ดังแสดงในรูปที่ 6.15



รูปที่ 6.15 แสดงผลจากคลิกที่ TIME/STEP

การใส่ค่าเวลาต่อหนึ่งสเต็ปก็คือการให้ความเร็วรอบแก่มอเตอร์นั่นเอง โดยค่านี้มีตั้งแต่ 0 ถึงค่าอนันต์ ค่าที่มากจะทำให้มอเตอร์หมุนช้าเพราะใช้เวลาในการเคลื่อนที่หนึ่งสเต็ปมาก ในทางตรงกันข้ามหากว่าใส่ค่าเวลาต่อหนึ่งสเต็ปมีค่าน้อยจะทำให้มอเตอร์หมุนเร็ว ความเร็วหาได้จากสูตร

$$N = 300/T$$

T คือค่าเวลาต่อหนึ่งสเต็ป

N คือความเร็วรอบ (รอบ/นาที)

การทดลอง

ลำดับขั้นการทดลอง

-ทำการรันตามขั้นตอนข้างต้น

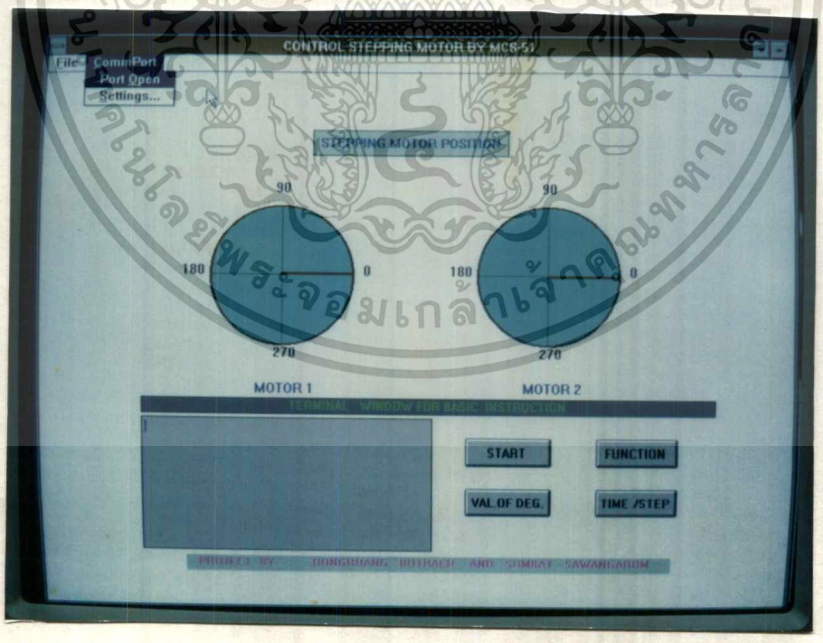
เลือกฟังก์ชั่น 6

จำนวนองศา 360

เวลาต่อสเต็ป 10

ผลการทดลอง1

ผลการทดลองจะเห็นว่ามอเตอร์หมุนไป 1 รอบ โดยเข็มมาหยุดที่ตำแหน่ง 0 องศา ดังแสดงในรูปที่ 6.16



รูปที่ 6.16 แสดงผลการทดลองที่1

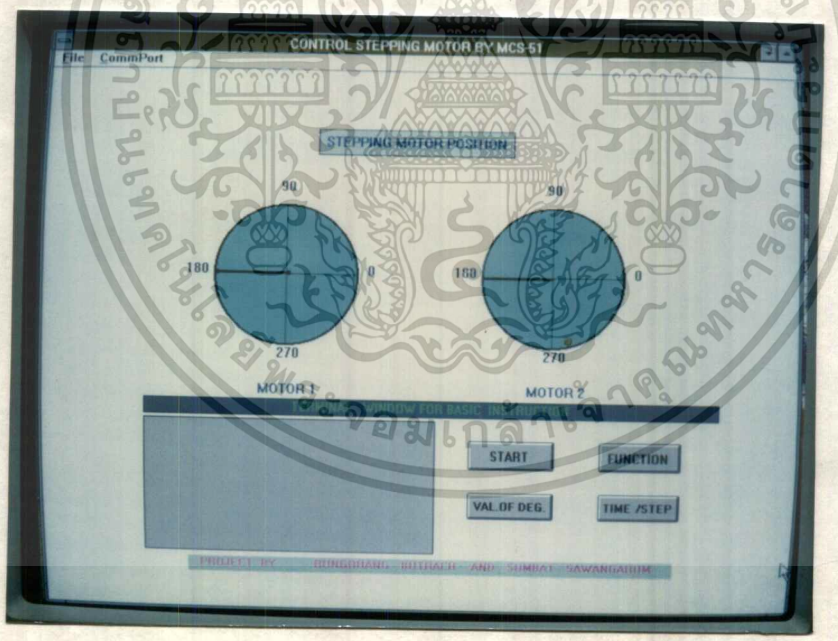
ผลการทดลองที่ 2

เลือกฟังก์ชั่น 6

จำนวนองศา 180

เวลาต่อสเต็ป 10

ผลการทดลอง จะเห็นว่ามอเตอร์จะหมุนไปทางขวาทั้งสองตัว และไหยุดที่ 180 องศา ดังแสดงในรูป 6.17



รูปที่ 6.17 แสดงผลการทดลองที่ 2

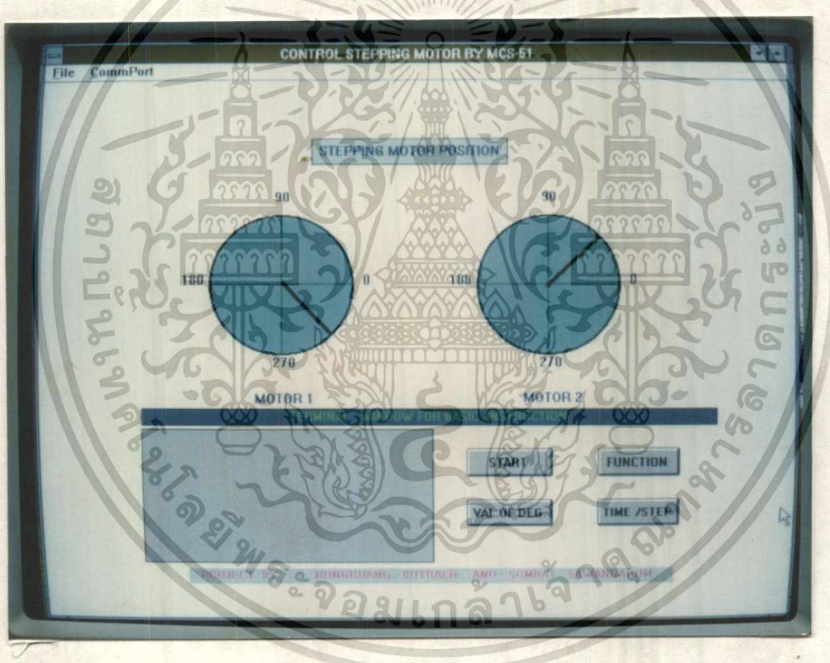
ผลการทดลองที่ 3

เลือกฟังชั่น 7

จำนวนองศา 45

เวลาต่อสเต็ป 30

ผลการทดลองพบว่ามอเตอร์ 1 หมุนขวาไป 45 องศา มอเตอร์ 2 หมุนซ้ายไป 45 องศา ดังแสดงในรูปที่ 6.18



รูปที่ 6.18 แสดงผลการทดลองที่ 3

บทสรุป

จากโครงงานนี้จะเห็นว่าการควบคุมการทำงานของสเต็ปปิ้งมอเตอร์นั้นสามารถทำได้โดยใช้ไมโครคอมพิวเตอร์ โดยการต่อพ่วงคอนโทรลเลอร์เข้าไปแล้วสั่งงานจากคีย์บอร์ดของคอมพิวเตอร์ เพื่อให้สเต็ปปิ้งทำงานตามวัตถุประสงค์ที่ได้ทำการโปรแกรมไว้ และสามารถมองเห็นการทำงานของสเต็ปปิ้งมอเตอร์ทางมอนิเตอร์ จากหลักการนี้สามารถนำไปประยุกต์ใช้ในงานต่างๆได้อย่างกว้างขวาง อาทิเช่นเครื่องพล็อตเตอร์, เครื่องถ่ายเอกสาร, แขนกล และในงานแมกคานิก ซึ่งสามารถพัฒนาโปรแกรมได้มากกว่าที่เป็นอยู่ในปัจจุบัน

บรรณานุกรม

1. TAKASHI KENJO.STEPPING MOTOR AND THEIR MICROPROCESSOR CONTORLS

2. ผู้ช่วยศาสตราจารย์พิพัฒน์ เลาหสงคราม ไมโครคอนโทรลเลอร์ 8051 กรุงเทพฯ

3. ETT CO., LTD CP-32BASIC EXPANISON COMMAND V2.0 กรุงเทพฯ

4. PHILIPS,SIGNETICS MICROCONTROLLER USERS GUIDE,PHILIPS ELECTRONIC AND COMPONENTS AND MATERIALS DIVISION,PHUILIPS CO

5. พิเชษฐ์ ฤทธิ์สุนทร. เทคนิคการรับส่งข้อมูล ดิจิตอล กรุงเทพฯ

6. รัชนัย อินทุใส. ไมโครคอนโทรลเลอร์ 8051 กรุงเทพฯ