เครื่องตรวจจับความเร็วในการเคลื่อนที่

INSPECT SPEED MOVING MACHINE

โดย

นายประพันธ์ นามธรรม

นายพงษ์พันธุ์ น้ำสมบูรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	เครื่องตรวจจับการเคลื่อนที่ของวัตถุ
TITLE	**INSPECT SPEED MOVING MACHINE**
นักศึกษา	นายประพันธ์ นามธรรม รหัสประจำตัว 43015731
	นายพงษ์พันธุ์ น้ำสมบูรณ์ รหัสประจำตัว 43015734
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อ.มนชนก ศรีเสือขาม
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2544

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(อ.มนชนก ศรีเสือขาม)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

หัวข้อปริญญานิพนธ์	เครื่องตรวจจับการเคลื่อนที่ของวัตถุ
นักศึกษา	นายประพันธ์ นามธรรม รหัสประจำตัว 43015731
	นายพงษ์พันธุ์ น้ำสมบูรณ์ รหัสประจำตัว 43015734
อาจารย์ผู้ควบคุมปริญญานิพนธ์	อ.มนชนก ศรีเสือขาม
ระดับการศึกษา	ปริญญาอุตสาหกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาการใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 ตรวจจับการเคลื่อนที่ของวัตถุ โดยใช้ลิมิตสวิตช์ในการตรวจจับการเคลื่อนที่ และมีการต่อใช้งานร่วมกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม การควบคุมต่างๆอาศัยไมโครคอนโทรลเลอร์เป็นตัวสั่งการทำงานตามโปรแกรมที่ได้โปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตระกูล MCS-51 การทำงานของตัว ไมโครคอนโทรลเลอร์จะทำการอ่านค่าสัญญาณจากลิมิตสวิตช์เมื่อมีวัตถุเคลื่อนที่ผ่านตัวที่ตัวลิมิตสวิตช์ที่จุด 2 จุด ไมโครคอนโทรลเลอร์จะทำการประมวลผลและควบคุมส่วนจอแสดงผลแอลซีดี และแสดงหน่วยความเร็วออกมาที่หน้าจอคอมพิวเตอร์เป็นกิโลเมตรต่อชั่วโมง

กิตติกรรมประกาศ

การจัดทำโครงงานเครื่องตรวจจับความเร็วได้สำเร็จลุล่วงไปด้วยดีเพราะอาจารย์หลายๆ ท่านและผู้มีพระคุณทุกท่านที่ได้แนะนำสั่งสอนหากโครงงานเล่มนี้มีประโยชน์บ้าง ขอมอบความดีนี้แก่ผู้มีอุปการะคุณทุกท่าน

ในการดำเนินงานจัดทำโครงงานนี้ คณะผู้จัดทำขอขอบพระคุณอาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำทางด้านวิชาการ และในการดำเนินงานปฏิบัติงานตลอดจนแก้ไขปัญหาต่างๆ และขอขอบคุณท่านผู้ให้การสนับสนุน ช่วยเหลืองานทางด้านต่างๆที่มิได้กล่าวนามถึงจนโครงงานสำเร็จลุล่วงไปด้วยดี

ขอกราบขอบพระคุณบิดา-มารดา ที่ให้การสนับสนุนในทุกๆอย่าง ไม่ว่าเป็นเรื่องใดๆก็ตาม และที่สำคัญคือกำลังใจที่มีให้ตลอดอย่างดี ในการทำงานในโครงงานนี้

ผู้จัดทำโครงงาน

นายประพันธ์ นามธรรม รหัสประจำตัว 43015731

นายพงษ์พันธุ์ น้ำสมบูรณ์ รหัสประจำตัว 43015734

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

โครงงานนี้มีชื่อเรียกว่า "เครื่องตรวจจับความเร็ว" ใช้ตรวจจับความเร็วของวัตถุโดยที่วัตถุที่ผ่านเข้ามาจะถูกตรวจจับโดยเซนเซอร์และแสดงผลออกมาที่จอแสดงผลเป็นหน่วยเมตรต่อชั่วโมงโดยโครงงานนี้จะมีชุดเซนเซอร์ไว้ตรวจจับวัตถุที่ผ่านเข้ามาและนำมาแสดงผลที่จอแอลซีดี (LCD) อาศัยการควบคุมผ่านไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ AT-89C52 เป็นตัวไมโครคอนโทรลเลอร์ขนาด 40 ขามีหน่วยความจำโปรแกรมภายใน 8 กิโลไบต์ เป็นหน่วยความจำแบบแฟลช (Flash memory) สามารถโปรแกรมและลบได้นับพันครั้ง มีพอร์ตอินพุทและเอาท์พุทเหมาะสำหรับโครงงานขนาดเล็กและราคาไม่แพง การที่เลือกใช้ไมโครคอนโทรลเลอร์เพราะมีขนาดเล็กใช้งานง่าย มีชุดคำสั่งไม่มากนัก ประยุกต์ใช้งานง่าย เชื่อมต่อกับอุปกรณ์อื่นได้ไม่ยากนัก มีสามารถติดต่อกับคอมพิวเตอร์โดยผ่านพอร์ตอนุกรม (Serial Port) ได้

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อนำตัวไมโครคอนโทรลเลอร์ MCS-51 มาประยุกต์ใช้งานในการตรวจจับการเคลื่อนที่ของวัตถุ

1.1.2 เพื่อนำตัวไมโครคอนโทรลเลอร์ MCS-51 มาต่อใช้งานร่วมกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมเพื่อเป็นตัวจับเวลา

1.2 ขอบเขตโครงงาน

1.2.1 สามารถโปรแกรมตัวไมโครคอนโทรลเลอร์ควบคุมการทำงานของเครื่องตรวจจับความเร็ว โดยโปรแกรมจะทำการตรวจจับวัตถุที่เคลื่อนที่ไปข้างหน้าเท่านั้น ไม่ตรวจจับวัตถุที่เคลื่อนที่ถอยหลัง

1.2.2 ตรวจจับการเคลื่อนที่ของวัตถุโดยใช้ลิมิตสวิตช์

1.2.3 ความเร็วของวัตถุที่เคลื่อนที่ไปข้างหน้าแสดงผลออกมาเป็นตัวเลขที่จอแสดงผลบนจอคอมพิวเตอร์

1.3 วิธีการดำเนินโครงงาน

1.3.1 ศึกษาทฤษฎีและการทำงานของตัวไมโครคอนโทรลเลอร์ MCS-51 ศึกษาการเขียนโปรแกรมที่ใช้ในโครงงาน

1.3.2 ขั้นตอนการทำงานสามารถแบ่งได้เป็นข้อๆดังนี้

1) ศึกษาทฤษฎีของไมโครคอนโทรลเลอร์ MCS-51 และโปรแกรมที่ใช้ในโครงงาน

2) ทดลองใช้งานไมโครคอนโทรลเลอร์ MCS-51 และโปรแกรมที่ใช้ในโครงงาน

3) เขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51

4) ออกแบบลายวงจรทั้งหมดที่ใช้ในโครงงานโดยใช้โปรแกรมโปรเทล

5) นำส่วน Hardware และ Software ประกอบเข้าด้วยกัน

6) ทดลองการทำงานของโครงงาน

7) บันทึกการทดลองและแก้ไขปรับปรุง

8) สรุปผลการทำงานของโครงงาน

บทที่ 2
ทฤษฎีทั่วไป

2.1ไมโครคอนโทรลเลอร์ MCS-51

2.1.1 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ใช้จะพูดถึงไมโครคอนโทรเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำแบบแฟรช(flash memory) ของ Atmel corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ในการเรียนรู้เพื่อใช้วานไมโครคอนเลอร์ ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1.หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟรช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2.ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์และเครื่องโปรแกรมอีพรอม

3.บริษัทผู้ผลิตได้ทำการผลิตไมโครลคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4.ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรเลอร์ ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้อย่างดี

5.ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไมต้องถอดตัวไมโครลคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่ หรือเรียกว่า **การโปรแกรมในวงจร หรือ ในระบบ(In-system programming)**ทำให้การพัฒนาหรือการซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัปเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก

6.ชุดคำสั่งและสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรล MCS-51 ของผู้ผลิตอื่นไม่ว่าจะเป็นอินเทล ซีเมนต์ หรือ ดัสลัส

2.1.2 คุณสมบัติของไมโครลคอนโทรลเลอร์ ตระกูล MCS-51 อนุกรม AT89xx

เป็นไมโครลคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต

ภายในหน่วยความจำโปรแกรมเป็นแบบแฟรชสามารถลบและเขียนใหม่ได้พันครั้ง

หน่วยความจำพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีพรอมเพิ่มเติม

ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานได้ทั้งอินพุตและเอาต์พุต

ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว

สามารถรองรับแหล่งกำเนิดอินเตอร์รัปได้ 6 ประเภท

สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์

มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป

ในรูปที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟรชที่เพิ่มเติมเข้ามา หากเป็นไมโครลคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำโปรแกรมภายในจะเป็นแบบอีพรอม และบางเบอร์โปรแกรมได้เพียงครั้งเดียว

สำหรับในรูปที่ 2-2 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่า มีส่วนประกอบเพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครลคอนโทรลร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบหรือเรียกว่าการโปรแกรมภานในวงจร ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตที่เพิ่มเติมเข้ามาอีกหนึ่งตัวเป็นไทเมอร์ 2 และวงจรวอตช์ด็อกใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู

ในตารางที่ 2-1 แสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แต่ละเบอร์ที่ Atmelผลิตขึ้น และมีใช้งานอยู่ในปัจจุบัน

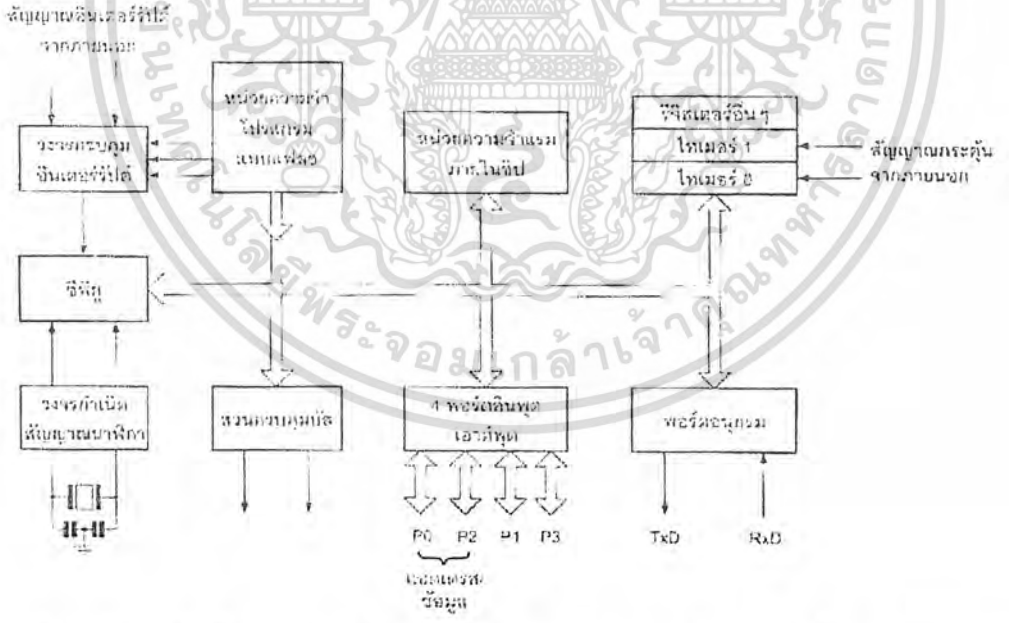
2.1.3 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูปที่ 3 และ 4 โดยมีรายละเอียดขั้นต้น ดังนี้

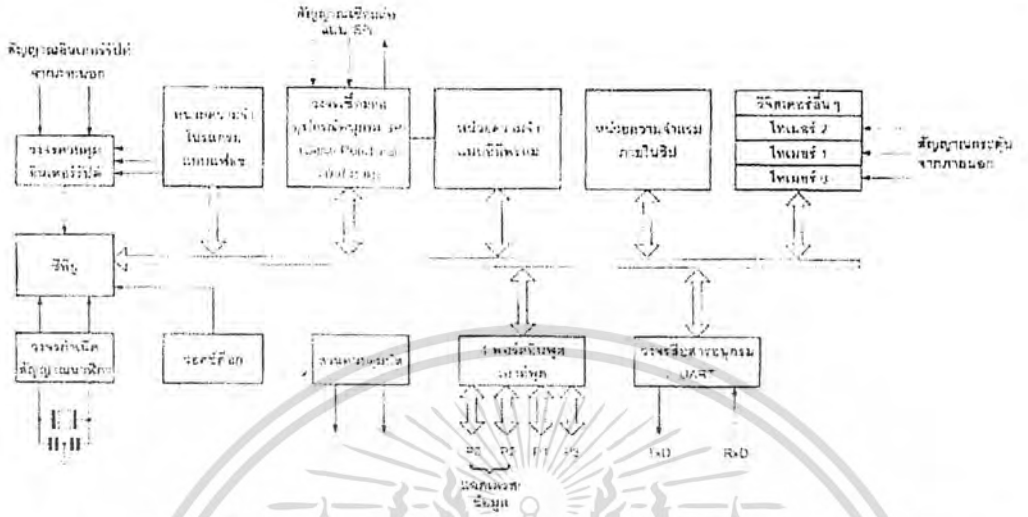
ขา Vcc ใช้ต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 (P0.0-P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำการได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลทำให้ขาพอร์ตนั้นมีสถานะปล่อยลอย(float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก(A0-A7)และขาข้อมูล(D0-D7)โดยใช้กระบวนการมัลติเพล็กเข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล



รูปที่ 2-1 โครสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx



รูปที่ 2-2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Sxx

เบอร์ของไมโครคอนโทรลเลอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	จำนวนไทเมอร์/เคาน์เตอร์ 16 บิต
AT89C1051	แบบแฟลช ขนาด 1 กิโลไบต์	แรม 64 ไบต์	1
AT89C2051	แบบแฟลช ขนาด 2 กิโลไบต์	แรม 128 ไบต์	2
AT89C51	แบบแฟลช ขนาด 4 กิโลไบต์	แรม 128 ไบต์	2
AT89C52	แบบแฟลช ขนาด 8 กิโลไบต์	แรม 256 ไบต์	3
AT89C55	แบบแฟลช ขนาด 20 กิโลไบต์	แรม 256 ไบต์	3
AT89S8252	แบบแฟลช ขนาด 8 กิโลไบต์	แรม 256 ไบต์ ยีอีพรอม 2 กิโลไบต์	3
AT89S53	แบบแฟลช ขนาด 12 กิโลไบต์	แรม 256 ไบต์	3

ตารางที่ 2-1 รายละเอียดโดยสรุปบางส่วนของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่ Atmel ผลิต

ขาพอร์ต 1 (P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำการได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุททริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4-P1.7 เป็นขาเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต 2 (P2.0-P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำการได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลทำให้ขาพอร์ตนั้นมีสถานะปล่อยลอย(float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก(A8-A15)

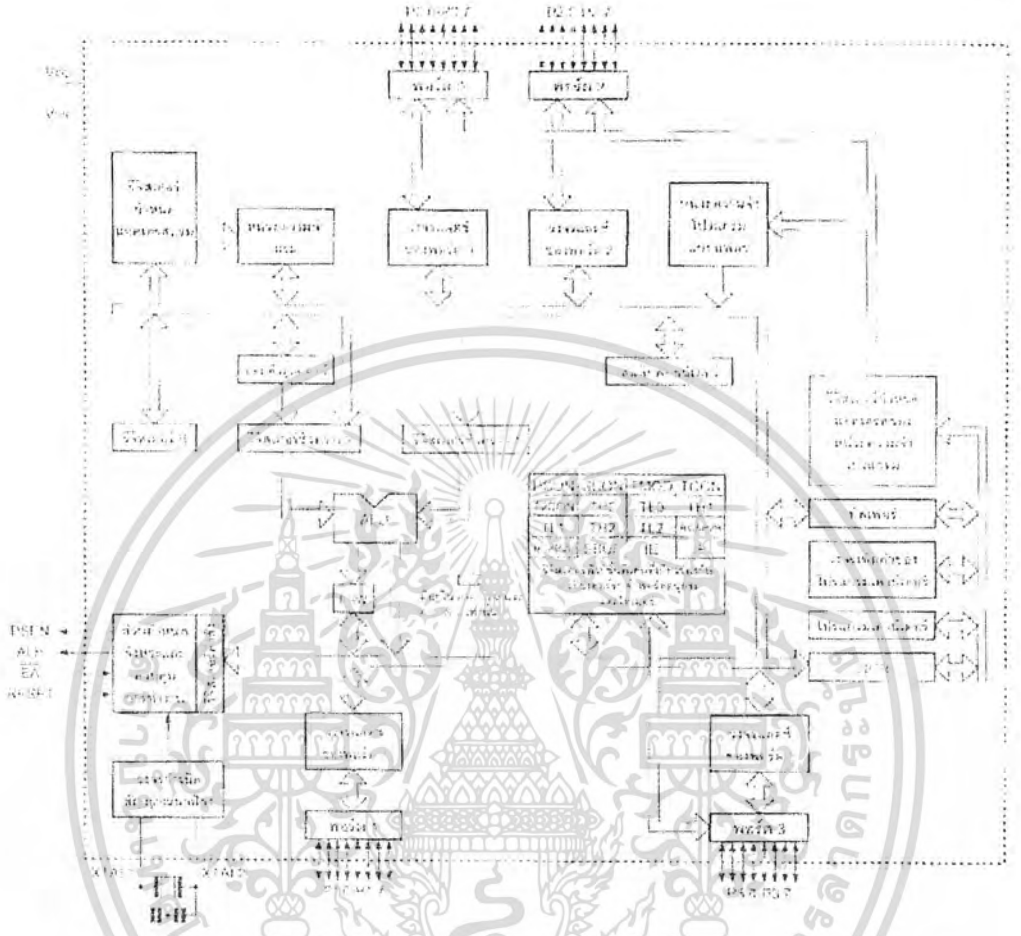
ขาพอร์ต 2 (P2.0-P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำการได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลทำให้ขาพอร์ตนั้นมีสถานะปล่อยลอย(float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่ใช้งานพิเศษ ดังมีรายละเอียดดังต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือ ขา INT0

P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือ ขา INT1



รูปที่ 2-3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmel

รูปที่ 2-4 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

P3.4 ใช้เป็นขาอินพุตรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือ ขา T0

P3.5 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือ ขา T1

P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีใช้กับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีใช้กับหน่วยความจำภายนอก

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE /PROG (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนี้ขานี้ยังใช้เป็นขาสำหรับรับพัลซ์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรกรมเป็นแบบอีพรอม

ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อขอร้องติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปแกรมภายนอกตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆออกมา

ขา EA/Vpp (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอกแต่ถ้าหากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12 V

ขา XTAL1 และ XTAL2 เป็นสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.1.4 โครงสร้างและทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึงพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาท์พุตสำหรับส่งสัญญาณข้อมูลออกทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด ดังสรุปได้ในตารางที่ 2-2

ในรูปที่ 2-5 แสดงวงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยในรูปที่ 2-5 (ก)เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิฟลอป

ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุตเอาท์พุตปกติหรือใช้การติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย

ในรูปที่ 2-5 (ข) เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจร มัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำรายละเอียดของวงจรพูลอัป แสดงในรูปที่ 2-6

ในรูปที่ 2-5 (ค) เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างเพียงมีวงจรพูลอัปเพิ่มเข้ามาส่วนในรูปที่ 2-5 (ง) เป็นวงจรภายในของพอร์ต 3 จะเห็นว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์และวงจรอินพุตเอาท์พุตเมื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา

2.1.5การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดหารทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรงส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น "1" สามารถรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านเข้าไป เมื่อเป็นเช่นนี้อุปกรณ์ภายนอกที่เชื่อมต่อกับอินพุตพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก "0" จะดีและสะดวกที่สุด

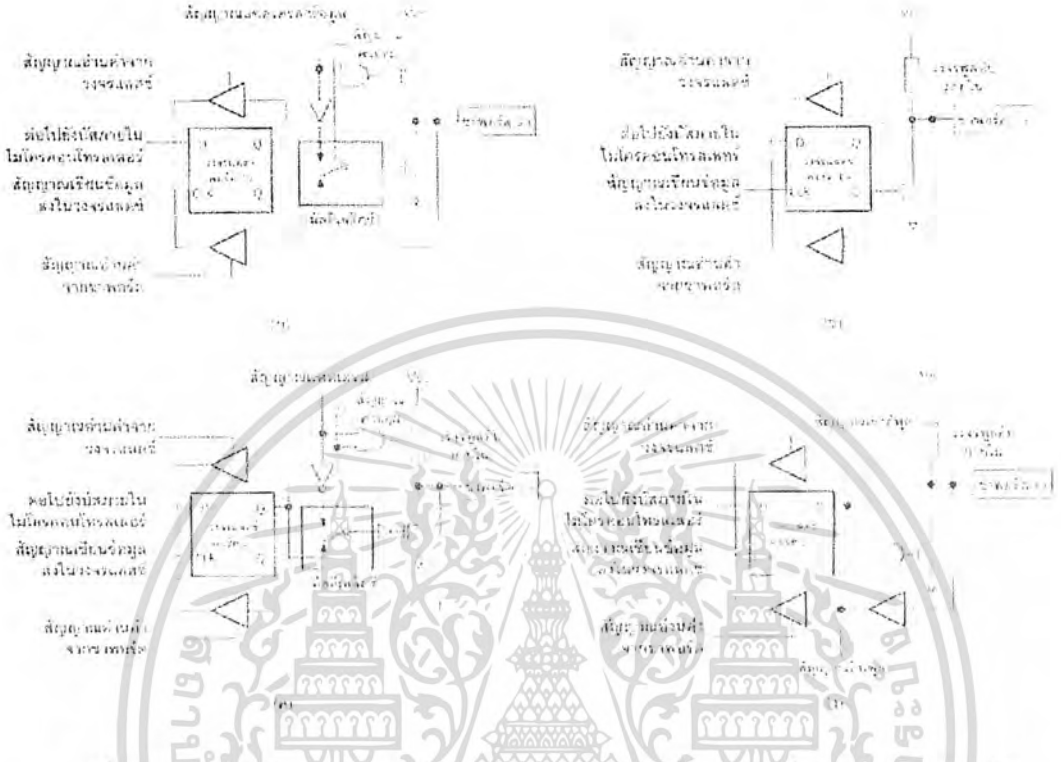
ขา	เบอร์ของไมโครคอนโทรลเลอร์	หน้าที่พิเศษ
P1.0	AT89C52/AT89Sxx	ขา T2 เป็นขาอินพุตนับค่าของไทเมอร์/เคาน์เตอร์ 2 และเป็นขา
P1.1	AT89C52/AT89Sxx	และควบคุมทิศทางของสัญญาณ
P1.4	AT89Sxx	ขา SS (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่ไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟ ในระบบการติดต่อแบบ SPI
P1.5	AT89Sxx	ขา MOSI (Master data output, Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.6	AT89Sxx	ขา MISO (Master data input, Slave data output) ใช้ในการติดต่อกับพอร์ต SPI
P1.7	AT89Sxx	ขา SCK (Master clock output) เป็นขาสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

ตารางที่ 2-2 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmel

2.1.6 การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือเมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุต ก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0" ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไปก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์วงจรก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่กรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุต แต่ละขา (หรือละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส **(source current)** ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสควรต่อวงจรบัฟเฟอร์ทางเอาต์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง



รูปที่ 2-5 วงจรภายในของพอร์ตทุกพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

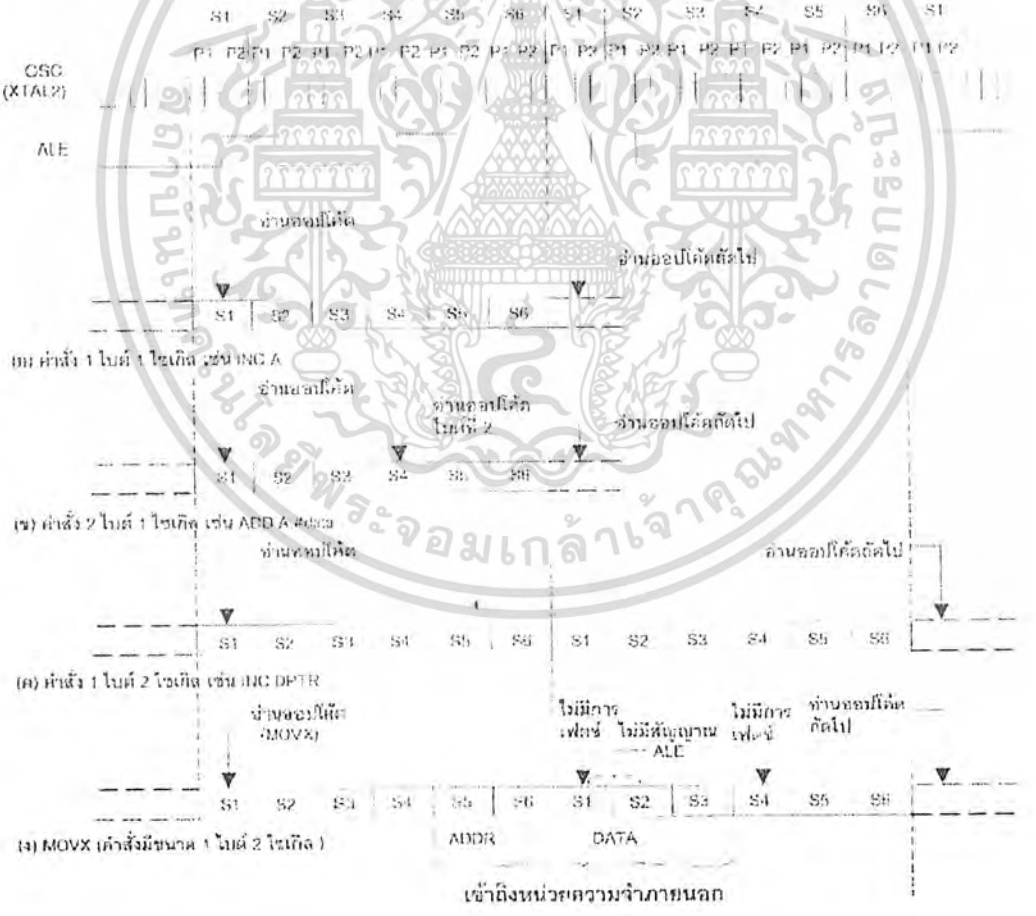
วงจรพูลอัปประกอบด้วยเฟตชนิดพีแชนเนล 3 ตัวคือ PFET1-PFET3 โดย NFET1 จะทำงานเมื่อได้รับลอจิก "1" จากขา Q และจะหยุดทำงานเมื่อได้รับลอจิก "0" วงจรพูลอัปจะเริ่มต้นทำงานเมื่อ NFET1 ได้รับลอจิก "1" PFET1 จะทำงานนานประมาณ 2 คาบเวลาของสัญญาณนาฬิกาภายใน หลังจากนั้นเกิดการเปลี่ยนแปลงจากลอจิก "0" เป็นลอจิก "1" ในขณะที่ PFET1 ทำงาน จะทำให้ PFET3 ทำงานตามไปด้วย ทำให้มีกระแสพูลอัปจากพอร์ต

รูปที่ 2-6 วงจรพูลอัปภายในพอร์ตไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2.1.7 การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะคือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในกรณีที่พอร์ตต่อกับขาเบสของทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงานสถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเสมือนว่าขาพอร์ตนั้นถูกต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่หากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริงดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย



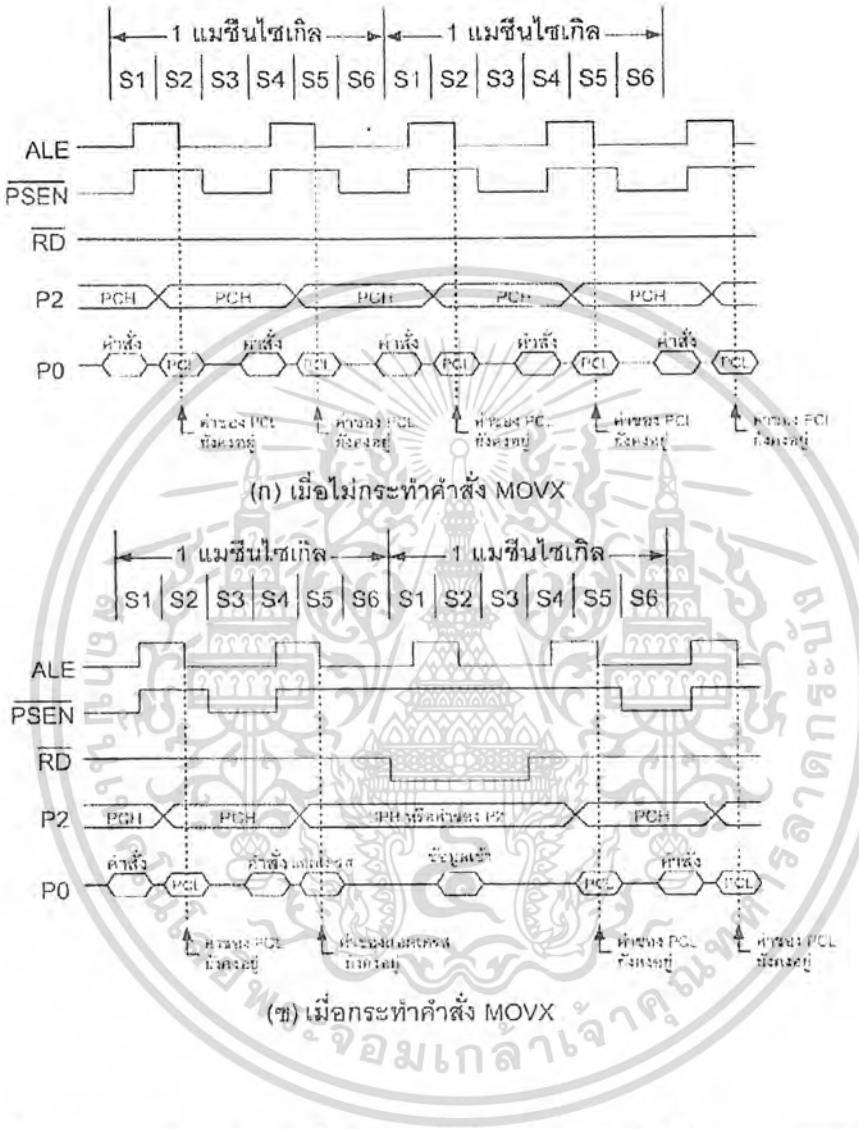
รูปที่ 2-7 ไซเกิลการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2.1.8 จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะการทำงานของซีพียูและลำดับขั้นตอนการประมวณผลคำสั่ง ในการประมวณผลคำสั่งของซีพียูจะมีขั้นตอนหลักๆ 2 ขั้นตอนคือ **กระบวนการเฟตช์ (fetch)** *เป็นการเรียกคำออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวณผล* ขั้นตอนต่อมาคือ **กระบวนการเอ็กซิคิวต์ (execute)** *เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นมาโดยกระบวนการก่อนหน้านี้* เมื่อทำการเอ็กซิคิวต์คำสั่งเรียบร้อยแล้ว ก็จะไปเริ่มกระบวนการเฟตช์คำสั่งใหม่ต่อไป

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์จะเกิดการรีเซตในลักษณะที่เรียกว่า **เพาเวอร์ออนรีเซต (power on reset)** ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม จังหวะการทำงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากรอบการทำงานหรือแมชีนไซเกิล ในรูปที่ 7 เป็นไดอะแกรมเวลาแสดงจังหวะการทำงานของ ไมโครคอนโทรลเลอร์ MCS-51 โดยในหนึ่งรอบการทำงานหรือแมชีนไซเกิลจะแบ่งย่อยออกเป็น 6 สเตต (state) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตตมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms คาบเวลาทั้งสองภายในหนึ่งสเตตจะเรียกว่าเฟส 1 (phase 1) และเฟส 2 (phase 2)

ในรูปที่ 2-7(ก) และ (ข) จะเป็นการเอ็กซิคิวต์คำสั่งที่ใช้เวลา 1 ไซเกิลเริ่มต้นที่สเตต 1 จะเป็นการอ่านค่าออปโค้ด อันเป็นกระบวนการแลตช์ค่าของออปโค้ดส่งไปให้รีจิสเตอร์คำสั่ง(Instruction Register:IR) การเฟตครั้งที่ 2 จะเกิดขึ้นที่สเตต 4 ภายในแมชีนไซเกิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตครั้งที่ 2 ภายในแมชีนไซเกิลเดียวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีใช้เวลา 1 ไซเกิล จะสิ้นสุดการทำงานลงในสเตต 6 ของแมชีนไซเกิลเดียวกัน



รูปที่ 2-8 ไดอะแกรมเวลาแสดงการติดต่อและเข้าถึงหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในกรณีที่คำสั่งใช้เวลา 2 ไซเกิล การทำงานของคำสั่งนั้นจะสิ้นสุดลงในสเตต 6 ของแมชีนไซเกิลที่ 2 ดังในไดอะแกรมรูปที่ 7 (ค) สำหรับในการกระทำคำสั่ง MOVX ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไซเกิล จะไม่มีการเฟตเกิดขึ้นในไซเกิลที่สองของคำสั่ง MOVX นี้ เนื่องจากซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอกดังแสดงในไดอะแกรมรูปที่ 2-7 (ง) จะหิ้นได้ว่าเวลาในการเอ็กซิคิวต์จะไม่ได้ขึ้นอยู่กับว่าทำการติดต่อหน่วยความจำโปรแกรมภายในหรือภายนอก

ในรูปที่ 2-8 แสดงสัญญาณและไดอะแกรมเวลาของการเข้าถึงหน่วยความจำโปรแกรมภายนอก โดยในรูปที่ 2-8 (ก) เป็นไดอะแกรมเวลาในขณะที่ยังไม่มีการกระทำคำสั่ง MOVX สัญญาณที่ขา ALE และ PSEN จะเกิดการแอคตีฟ 2 ครั้งภายในหนึ่งแมชีนไซเกิล ในทุกครั้งที่ ALE เกิดการแอคตีฟที่พอร์ต 0 (P0) จะมีค่าของรีจิสเตอร์ PC ในไบต์ต่ำออกมา ในขณะที่พอร์ต 2 (P2) ก็มีค่าของ PC ในไบต์สูงเพื่อชี้ไปยังแอดเดรสต่อไปที่ต้องไปดำเนินการ สำหรับขา PSEN ก็จะเกิดการแอคตีฟเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ในกรณีที่กระทำคำสั่ง MOVX เพื่อเข้าถึงหน่วยความจำข้อมูลภายนอก ที่ขา PSEN จะไม่เกิดการแอคตีฟ 2 ครั้งภายในหนึ่งแมชีนไซเกิลเนื่องจากบัสแอดเดรสและบัสข้อมูลจะใช้ในการติดต่อหน่วยความจำข้อมูลภายนอกแทน แต่สำหรับสัญญาณ ALE ยังคงแอคตีฟตามจังหวะการทำงานเหมือนเดิม

จากไดอะแกรมเวลาสามารถสรุปได้ว่า ในการทำงาน 1 รอบหรือ 1 แมชีนไซเกิล ซีพียูในไมโครคอนโทรลเลอร์ MCS-51 จะใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไซเกิลมีค่าเท่ากับ 1ms หรือมีความเร็วในการทำงานภายใน 1MHz ในกรณีที่ใช้ความถี่สัญญาณนาฬิกา 12MHz ดังนั้นถ้าต้องการทราบความเร็วของการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถหาได้จาก ค่าความถี่สัญญาณนาฬิกาหารด้วย 12 และถ้าต้องการหาค่าเวลาของ 1 รอบการทำงานหรือ 1 แมชีนไซเกิล สามารถทำได้โดยการหาส่วนกลับของความเร็ว ในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถสรุปเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ

ความถี่ของสัญญาณนาฬิกา(ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2)/12

เวลา 1 แมชีนไซเกิล = 1/ความเร็วในการทำงานภายในของไมโคร-คอนโทรลเลอร์

2.2 การขับโมดูลแสดงผลแบบผลึกเหลว(LCD module)

2.2.1 รายละเอียดเกี่ยวกับโมดูล LCD

โมดูล LCD จะมีส่วนประกอบหลักๆ 3 ส่วน ดังนี้

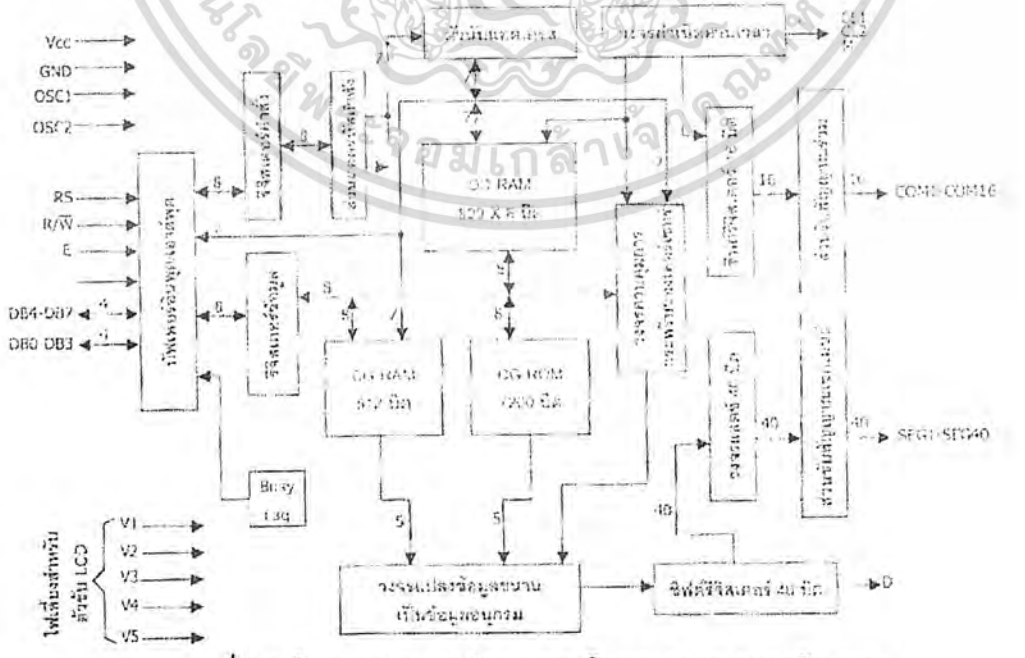
ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวควบคุม (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ทำหน้าที่ทำเป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259

โครงสร้างภายในตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งใช้ในการควบคุมให้ดีเสียก่อน ในหนังสือนี้ขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจง่าย ให้รูปที่ 1 เป็นบล็อกไดอะแกรมภายในของตัวควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย



รูปที่ 2-9 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

บัฟเฟอร์อินพุทเอาต์พุท เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจีสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจีสเตอร์ที่ใช้ในการรับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

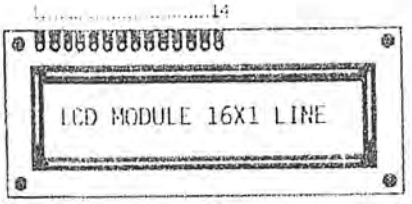
รีจีสเตอร์จข้อมูล (Data Register : DR) เป็นรีจีสเตอร์ที่ใช้ในการรับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำทำหน้าที่เก็บข้อมูลที่มาจากรีจีสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-lable) ของตัวอักษรที่เก็บไว้ในหน่วยความจำของรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM :CGROM) เป็นหน่วยความจำที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงผลได้ มีขนาด 7,200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำที่ใช้เก็บตัวอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGRAM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGRAM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะของการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดั้งนั้นก่อการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน



ขา 1 : GND
ขา 2 : +V
ขา 3 : Brightness ปรับความสว่าง
ขา 4 : RS
ขา 5 : R/W
ขา 6 : E
ขา 7-14 : D0-D7

รูปที่ 2-10 รูปร่างและการจัดขา LCD แบบอักขระ

2.2.2โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด(LCD 16x1)

สำหรับโมดูล LCD ที่ใช้ยกตัวอย่างนี้ เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างมาตฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์ที่แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทร็กซ์ (Cptrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั้นคือเบอร์ HD44750 ของฮิตาชิ

โมดูล LCD ขนาด 16x1 มีขาที่ต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังรูปที่ 2 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

Vss(ขา 1) : ต่อกราวด์

Vdd(ขา 2) : ต่อไฟเลี้ยง +5 โวลต์

Vo(ขา 3) : เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS(ขา 4) : เป็นขาอินพุทใช้ในการแยกชนิดข้อมูลของการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจีสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจีสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลที่ใช้ในการแสดงผล

R/W (ขา 5) : เป็นขาที่ใช้ในการเลือกอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

อนึ่งขา RS, R/W และ E จะใช้งานร่วมกัน โดยมีความสัมพันธ์แสดงในตารางที่ 2-2

RS	R/W	E	
		⌐_	
0	1	⌐¯¬	
1	0	⌐_	
1	1	⌐¯¬	

ตารางที่ 2-2 แสดงความสัมพันธ์ในการทำงานของขา RS, R/W และ E ของโมดูล LCD แบบอักขระ

2.2.3คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น "0" แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่ง

1.คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D ให้เป็น "1"

2.คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั้นคือ ข้อมูลของคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3.คำสั่งเลือกโหมดการป้อนข้อมูล (Etry mode Set)

มีรายละเอียดของข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	0	0	I/D	S

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น "1" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย ถ้าหากบิตนี้เป็น "0" เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยอ้างบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H-07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของ DDRAM เพิ่มขึ้น

4.คำสั่งควบคุมการแสดงผล

มีรายละเอียดของข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	1	D	C	B

บิต D ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น "1" จะเป็นการเปิดจอแสดงผล ถ้าเป็น "0" จะเป็นการปิดจอแสดงผล

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงบนจอแสดงผล ต้องกำหนอบิตนี้ให้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดั้งนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H-0F(8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ OCH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ OFH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์ และสั่งให้เคอร์เซอร์กระพริบ

5.คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

S/C	R/L	ลักษณะการเลื่อน	ข้อมูลคำสั่ง
0	0	เลื่อนเคอร์เซอร์ไปทางซ้าย	10H-13H
0	1	เลื่อนเคอร์เซอร์ไปทางขวา	14H-17H
1	0	เลื่อนตัวอักษรใหม่ไปทางซ้าย	18H-1BH
1	1	เลื่อนตัวอักษรใหม่ไปทางขวา	1C-1FH

6.คำสั่งกำหนดฟังชั้นการทำงาน

มีรายละเอียดของข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	1	DL	N	F	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น "0" จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น "1" จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น "0" จะแสดงผล 1 บรรทัด ถ้าเป็น "0" จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงผลได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น "1" จะแสดงผลเป็นแบบ 5x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิต แสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5x7 จุด

7.คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น "0" บิต 6 เป็น "1" ส่วน 6 บิตที่เหลือจะแทนด้วยแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านและเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H-3FH

8.คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 ต้องเป็น "1" และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH-0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น "0" แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH ถ้าบิต N เป็น "1" แอดเดรสของ DDRAM จะมี 2 ช่วง คือ 8CH-0FFH และ 0C0H-0C7H

9.คำสั่งอ่าน BUSY และแอดเดรส

มีรายละเอียดของข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
BF	A	A	A	A	A	A	A

เป็นคำสั่งที่ใช้อ่านแฟลก BUSY(BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น "0" แสดงว่าตัวควบคุม LCD พร้อมจะรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการควบคุมภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมจะรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0-บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.3 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นแบ่งออกได้เป็น 2 แบบคือการสื่อสารแบบอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา, ข้อมูล และกราวด์ รูปที่ 1-1 แสดงให้เห็นถึงไทมิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส

2.3.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยกันเหมือนกันกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายถอดข้อมูล หรือ บอดเรต (baudrate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1.บิตเริ่มต้น (start bit) ซึ่งจะมีขนาด 1 บิต

2.บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ 8 บิต

3.บิตตรวจสอบพาริตี้ (Paraty Bit) จะมีขนาด 1 บิตหรือไม่มี

4.บิตปิดท้าย (Stop Bit) จะมีขนาด 1,1.5 หรือ 2 บิต

รูปที่ 2-11 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลที่จะส่งขา DATA จะมีสถานะลอจิก "1" ซึ่งจะแสดงสถานะนี้วาสถานะหยุดรอ (wait stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการใช้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าบิตเริ่มต้นจากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยต่ำสุด (LSB) ก่อน ซึ่งข้อมูลในไบต์ที่จะส่งอาจมีจำนวนบิต 5,6,7 หรือ 8 บิตก็ได้ จากนั้นจะตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะทำให้ขาดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต,1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronus Receive/Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรดมาตฐานที่ใช้สำหรับพอร์ตอนุกรมอนุกรม RS-232 ได้แก่ 110,150,300,600,1200,2400,4800,9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มสามารถกำหนดค่าบอดเรตได้สูงถึง 115200 บิตต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายได้ภายใน 1 วินาที ยกตัวอย่าง ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลที่เท่ากับ 9600 บิตต่อวินาที ก็สามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาราตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd),แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก "1" ภายในข้อมูลที่ส่งไป 1 ไบต์ ว่ามีจำนวนร่วมเป็นเลขคู่หรือเลขคี่โดยร่วมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่ส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ค่าในพาริตี้ จะต้องมีลอจิกเป็น "0" แต่ถ้าพาริตี้เป็นคี่ ค่าที่บิตพาริตี้ต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้มีจำนวนรวมกันเป็นเลขคี่ ในตารางที่ 1-1 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	0	1
11111111	1	0

ตารางที่ 2-3 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ URAT ซึ่งทางภาครับจะต้องทำการกำหนดคุณสมบัติการตรวจสอบพาริตี้ให้ตรงกันว่าจะตรวจสอบพาริตี้คู่หรือคี่ จากนั้นภาครับของ URAT จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งพาริตี้บิตด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดที่เกิดขึ้นในการถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ URAT เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ URAT เบอร์ 8250 URAT ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล(0 และ +5V) แต่เพื่อให้มีแรงดันเป็นไปตามมาตฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก "0" มีระดับแรงดัน +3V ถึง +12 ในขณะที่ลอจิก "1" มีระดับแรงดัน -3V จนถึง -12V

2.3.2 มาตฐานพอร์ตอนุกรมแบบ RS-232

มาตฐานพอร์ตอนุกรมแบบ RS-232 เป็นมาตฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตฐาน RS-232 ในอดีตนั้นออกแบบมาเพื่อใช้ในการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดหนึ่งซึ่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็คทรอนิคส์
(Electronic Industries Association : EIA) ได้วางมาตฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -12V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง(Space)

มาตฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่าอุปกรณ์ DTE จะต้องเป็นอุปกรณ์เป็นอุปกรณ์ที่ต้องมีการประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้

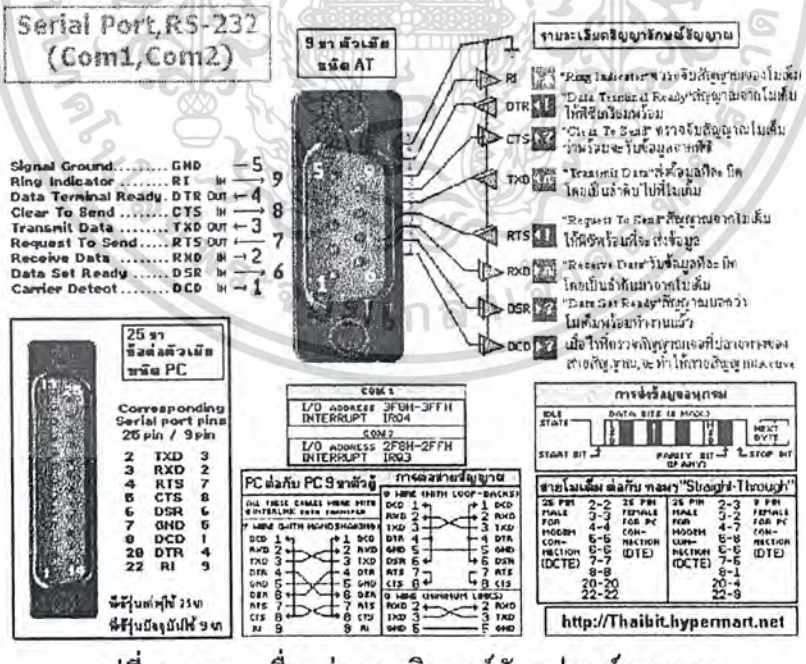
ส่วนอุปกรณ์ DCE จะทำหน้าเป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

2.3.3 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2-11



รูปที่ 2-11 การเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 2-11 ลูกศรในรูปแสดงทิศทางของข้อมูล ในรูปที่ 2-11 เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนเช็กเต็มรูปแบบ ส่วนในรูปที่ 2-11 เป็นการเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูลเช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์

Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลออกไป

Data Terminal Ready : DTR เป็นสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อแบบ Null Modem ซึ่งสายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

Signal Ground : GND ขากราวด์ของระบบ

Data Set Ready : DSR ขานี้ใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับส่งข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

Request To Send : RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

Clear To Send : CTS ขานี้จะคอยรับส่งสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับส่งข้อมูลหรือไม่

Ring Indicate : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.3.4 URAT

URAT มาจากคำว่า Universal Asynchronous Reciever Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั้นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว URAT ถือว่าเป็นหัวใจสำคัญของการสื่อสารแบบอนุกรม

หน้าที่หลักของ URAT คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานกับคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง URAT ให้เป็นแบบขนานก่อนที่จะส่งข้อมูลสู่คอมพิวเตอร์ ซึ่งนอกจาก URAT จะส่งข้อมูลไปยังคอมพิวเตอร์ ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (บอดเรต),รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้,เฟรมข้อมูล,โอเวอร์รัน) เป็นต้น

ภายใน URAT จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (programable baudrate) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ URAT โดยตัวหารมีขนาด 16 บิต ดังนั้นจึงกำหนดตัวหารอยู่ในช่วง 1-65,535 URAT สามารถรับข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (half duflex) และฟูลดูเพล็ก (full duflex) โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟูลดูเพล็กนั้นสามารถรับส่งข้อมูลได้ในคราวเดียวกัน

ชนิดของ URAT

ในเครื่องคอมพิวเตอร์ทั่วไปมี URAT ที่ใช้งานกันอยู่ 2 เบอร์ คือ 8250 ซึ่งเป็น URAT มาตฐานที่มีใช้กันมายาวนาน URAT เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกกำจัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ URAT เบอร์นี้ถือว่าเป็นเบอร์ต้นแบบของ URAT ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุกรุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ URAT เบอร์นี้

URAT อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 115,200 บิตต่อวินาที และเพิ่มรีจีสเตอร์สำหรับพักข้อมูลสำหรับ URAT นอกจากนั้นยังเพิ่มส่วนของชิฟต์ รีจีสเตอร์แบบ FIFO (First In First Out) ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ URAT เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจีสเตอร์แบบ FIFO ขนาด 64 กิโลไบต์ ทำงานได้ที่ระดับแรงดัน +5V และ +3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1
เมกะบิตต่อวินาทีเมื่อใช้สัญญาณนาฬิกา 16 MHz

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ URAT เบอร์ใหม่ๆ ก็ไม่ช่วยให้การรับส่งข้อมูลคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่สัญญาณนาฬิกาในการแปลงข้อมูลเพียง 1.8432 MHz

2.3.5 วงจรภายในและรีจีสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะมีชื่อเรียกเป็น COM1, COM2, COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน URAT ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน

ในรูปที่ 2-11 แสดงไดอะแกรมการทำงานภายในของพอร์ตอนุกรม ซึ่งประกอบไปด้วยรีจีสเตอร์ขนาด 8 บิต 8 ตัวที่ใช้งานร่วมกับ URAT แอดเดรสของรีจีสเตอร์ภายในพอร์ตอนุกรมสามารถคำนวณได้จากค่ารีจีสเตอร์พื้นฐานของพอร์ตอนุกรม ยกตัวอย่าง พอร์ตอนุกรม COM1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจีสเตอร์ต่างๆจะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจีสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

00H รีจีสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูลที่จะส่งออกไป

01H รีจีสเตอร์อีนาเบิลการอินเตอร์รัปต์ใช้ในการเซตโหมดอินเตอร์รัปต์ของพอร์ตอนุกรม

02H รีจีสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดการอินเตอร์รัปต์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

03H รีจีสเตอร์รีจีสเตอร์กำหนดรูปแบบของข้อมูล

04H รีจีสเตอร์ควบคุมโมเด็ม ใช้ติดต่อบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR

05H รีจีสเตอร์แสดงสถานะการรับและการส่งข้อมูลแบบอนุกรม

06H รีจีสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขา DCD,RI,DSR และ CTS

07H รีจีสเตอร์สำหรับเก็บข้อมูลชั่วคราว

รีจีสเตอร์ตำแหน่ง 00H : รีจีสเตอร์บัฟเฟอร์

เป็นรีจีสเตอร์สำหรับเก็บข้อมูลที่รับเข้ามาและข้อมูลที่จะส่งออกไป โดยการติดต่อกับรีจีสเตอร์นี้เพื่อเก็บข้อมูลที่ต้องการจะส่งจะต้องกำหนดให้บิต DLAB ในรีจีสเตอร์กำหนดรูปแบบข้อมูล(03H) จะต้องมีสถานะเป็น 0 ซึ่งการเขียนข้อมูลมายังแอดเดรสนี้ เป็นการส่งข้อมูลไปยังรีจีสเตอร์ส่งข้อมูลและข้อมูลถูกส่งออกไปแบบอนุกรม สำหรับการรับข้อมูล เมื่อมีข้อมูลที่รับเข้ามาเรียบร้อยและแปลงเป็นแบบขนานแล้ว ข้อมูลจะส่งมายังรีจีสเตอร์เก็บข้อมูล หลังจากมีการอ่านรีจีสเตอร์นี้ออกไปรีจีสเตอร์นี้จะถูกเคลียร์ และเตรียมพร้อมสำหรับเก็บข้อมูลในไบต์ต่อไป

รีจีสเตอร์ตำแหน่ง 01H : รีจีสเตอร์อีนาเบิลการอินเตอร์รัปต์

เป็นรีจีสเตอร์สำหรับการอีนาเบิลการอินเตอร์รัปต์ ซึ่งเป็นการกำหนดให้ URAT สร้างสัญญาณอินเตอร์รัปขึ้นมา ฟังชั่นการทำงานในแต่ละบิตมีดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	SINP	ERBK	TBE	RxRD

บิต 4-7 บิตเหล่านี้ไม่ถูกใช้งาน กำหนดให้เท่ากับ "0"

SINP อีนาเบิลการอินเตอร์รัปต์เนื่องเกิดจากการเปลี่ยนสถานะที่อินพุท CTS,DTS,DCD หรือขา RI

"1" อีนาเบิลการอินเตอร์รัปต์

"0" ไม่มีการอินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

ERBK อีนาเบิลการอินเตอร์รัปต์เนื่องจากเกิดความผิดพลาดขึ้นด้วยสาเหตุจากพาริตี้,โอเวอร์รัน,เฟรมข้อมูล หรือการเบรกข้อมูล

"1" อีนาเบิลการอินเตอร์รัปต์

"0" ไม่มีการอินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

TBE อีนาเบิลการอินเตอร์รัปต์เมื่อรีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"1" อีนาเบิลการอินเตอร์รัปต์

"0" ไม่มีการอินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

RxRD อีนาเบิลการอินเตอร์รัปต์เมื่อรีจีสเตอร์บัฟเฟอร์ได้รับข้อมูลเรียบร้อแล้ว
"1" อีนาเบิลการอินเตอร์รัปต์
"0" ไม่มีการอินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

รีจีสเตอร์ตำแหน่ง 02H : รีจีสเตอร์แสดงโหมดและสถานะการอินเตอร์รัปต์

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	0	ID1	ID0	PND

บิต 3-7 บิดเหล่านี้ไม่ถูกใช้งาน กำหนดให้เท่ากับ "0"

ID1, ID0 ใช้งานร่วมกันเพื่อแจ้งสาเหตุของการเกิดอินเตอร์รัปต์
"00" เกิดการอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงของขาอินพุทขึ้น
การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 4
"01" เกิดการอินเตอร์รัปต์เนื่องจากรีจีสเตอร์บัฟเฟอร์ส่งข้อมูลว่างขึ้น
การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 3
"10" เกิดการอินเตอร์รัปต์เนื่องจากข้อมูลถูกเก็บในกรีจีสเตอร์บัฟเฟอร์สำหรับข้อมูลเรียบร้อยแล้ว
การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 2
"11" เกิดการอินเตอร์รัปต์เนื่องจากความผิดพลาดในการถ่ายทอดข้อมูลหรือเกิดการเบรก (break : เกิดจากการหยุดถ่ายข้อมูลกระทันหัน)
การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 1 หรือมีนัยสำคัญสูงสุด

PND ใช้แสดงสถานะของการเกิดอินเตอร์รัปต์
"1" แสดงว่าไม่มีการอินเตอร์รัปต์
"0" แสดงว่ามีการอินเตอร์รัปต์เกิดขึ้น

เมื่อมีการสร้างสัญญาณอินเตอร์รัปต์ขึ้น จะต้องมีการเคลียร์ค่าก่อนที่จะเกิดการอินเตอร์รัปต์ครั้งต่อไป โดยสามารถทำได้ดังนี้คือ ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงของขาอินพุทจะต้องอ่านค่าของรีจีสเตอร์แสดงสถานะของโมเด็ม (รีจีสเตอร์ตำแหน่ง 06H) เพื่อเคลียร์ค่าการเกิดอินเตอร์รัปต์

ถ้าเกิดอินเตอร์รัปต์เนื่องจากบัฟเฟอร์ส่งข้อมูลว่างจะต้องเขียนข้อมูลไปยังรีจีสเตอร์บัฟเฟอร์ส่งข้อมูล (รีจีสเตอร์ตำแหน่ง 00H) หรืออ่านค่าจากรีจีสเตอร์แสดงสถานะการอินเตอร์รัปต์ (รีจีสเตอร์ตำแหน่ง 02H) เพื่อเคลียร์ค่าการอินเตอร์รัปต์

ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเก็บข้อมูลลงในรีจีสเตอร์บัฟเฟอร์สำหรับข้อมูลเรียบร้อยจะต้องเคลียร์ค่าอินเตอร์รัปต์ในการอ่านข้อมูลจากรีจีสเตอร์บัฟเฟอร์

ถ้าเกิดอินเตอร์รัปต์เนื่องจากความผิดพลาดในการรับส่งข้อมูลหรือเกิดการเบรก จะต้องเคลียร์ค่าการอินเตอร์รัปต์โดยการอ่านค่ารีจีสเตอร์แสดงสถานะการรับและส่งข้อมูลแบบอนุกรม

รีจีสเตอร์ตำแหน่ง 03H : รีจีสเตอร์รีจีสเตอร์กำหนดรูปแบบของข้อมูล

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
DLAB	BRK	PAR2	PAR1	PAR0	STOP	DAB1	DAB0

DLAB ใช้ในการกำหนดหน้าที่การทำงานของรีจีสเตอร์บัฟเฟอร์ (00H) "1" เป็นการเข้าสู่โหมดการหารค่าบอดเรต

"0" เป็นการเข้าถึงรีจีสเตอร์บัฟเฟอร์ (รีจีสเตอร์ตำแหน่ง 00H) และรีจีสเตอร์สำหรับอีนาเบิลการอินเตอร์รัปต์ (รีจีสเตอร์ตำแหน่ง 01H) เมื่อบิต DLAB เป็น "1" รีจีสเตอร์บัฟเฟอร์ (00H) และรีจีสเตอร์อีนาเบิลการอินเตอร์รัปต์ (01H) จะใช้สำหรับโหลดค่าการหารความถี่สำหรับการกำหนดค่าบอดเรต โดยรีจีสเตอร์ 00H เก็บค่าตัวหารไบต์ต่ำ ส่วนรีจีสเตอร์ 01H ใช้เก็บค่าตัวหารไบต์สูง การหาค่าบอดเรตสามารถเขียนเป็นสมการได้ดังนี้

บอดเรต = 115200/ตัวหาร 16 บิต

ค่าตัวเลข 115200 มาจากความถี่ของคริสตอลในวงจร URAT ภายในเครื่องคอมพิวเตอร์ โดยคริสตอลที่ใช้มีความถี่ 1.8432 MHz วงจรภายใน URAT จะทำการหารค่าความถี่นี้ด้วย 16 ทำให้ได้ค่าความถี่ 115200 Hz ออกมา

ค่าตัวหาร 16 บิต = ข้อมูลในรีจีสเตอร์ 00H + (256xข้อมูลในรีจีสเตอร์ 01H)

สมมุติว่าต้องการค่าบอดเรตเท่ากับ 9600 ค่าตัวหารที่ใช้จะต้องมีค่าเท่ากับ 12 ซึ่งค่าเหล่านี้จะถูกโหลดลงในรีจีสเตอร์ 00H และโหลดค่า 0 ลงไปในรีจีสเตอร์ 01H ค่าตัวหารที่ทำให้เกิดค่าบอดเรตสูงสุดที่ 1152000 บิตต่อวินาที คือ ค่า 0001 นั้นคือ

รีจีสเตอร์ 00H มีค่าเท่ากับ 1 และรีจีสเตอร์ 01H มีค่าเท่ากับ 0

BRK ใช้ควบคุมการหยุดถ่ายทอดข้อมูล

"1" สามารถหยุดหรือเบรกได้

"0" ไม่มีการหยุดหรือเบรกได้

PAR2, PAR1, PAR0 ใช้เพื่อกำหนดบิตพาราตี้

"000" ไม่ใช้บิตพาริตี้

"001" กำหนดพาริตี้คี่

"011" กำหนดพาริตี้คู่

"101" มาร์ก (mark)

"111" ช่องว่าง (space)

STOP ใช้กำหนดจำนวนบิตปิดท้าย

"1" มีบิตปิดท้าย 2 บิต

"0" มีบิตปิดท้าย 1 บิต

DAB1, DAB0 ใช้ร่วมกันในการกำหนดจำนวนบิตของข้อมูลที่ต้องการถ่ายทอด

"00" จำนวนบิตข้อมูลเท่ากับ 5 บิต

"01" จำนวนบิตข้อมูลเท่ากับ 6 บิต

"10" จำนวนบิตข้อมูลเท่ากับ 7 บิต

"11" จำนวนบิตข้อมูลเท่ากับ 8 บิต

รีจีสเตอร์ตำแหน่ง 04H : รีจีสเตอร์ควบคุมโมเด็ม

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	LOOP	OUT2	OUT1	RTS	DTR

บิต 5-7 ไม่มีการใช้งาน อ่านค่าได้เท่ากับ 0

LOOP "1" อีนาเบิลการส่งค่ากลับ

"0" ดิสเอเบิล

OUT1, OUT2 "1" อีนาเบิลการใช้งานภายใน

"0" ดิสเอเบิล

R1S ใช้ควบคุมการทำงานของขา RTS (Ready To Send)

"1" อีนาเบิล

"0" ดิสเอเบิล

DTR ใช้ควบคุมการทำงานของขา DTR (Data To Read)

"1" อีนาเบิล

"0" ดิสเอเบิล

รีจีสเตอร์ตำแหน่ง 04H : รีจีสเตอร์แสดงสถานะการรับและส่งข้อมูลแบบอนุกรมของ URAT

ใช้งานร่วมกับรีจีสเตอร์แสดงโหมดและสถานะของการอินเตอร์รัปต์ (รีจีสเตอร์ ตำแหน่ง 02H) เพื่อแสดงสาเหตุของการอินเตอร์รัปต์ มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	TXE	TBE	BREK	FRME	PARE	OVFE	RxRD

TXE(Transmitter Empty) "1" แสดงว่ารีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"0" แสดงว่ายังมีข้อมูล 1 ไบต์เก็บอยู่ในรีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

TBE(Transmitter Buffer Empty) "1" รีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง
"0" ยังมีข้อมูล 1 ไบต์เก็บอยู่ในรีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

BREK (Break) "1" URAT ตรวจพบการเบรก
"0" ไม่มีการเบรก

FRME (Frame Error) "1" URAT ตรวจพบความผิดพลาดด้านเฟรมข้อมูล
"0" ไม่พบความผิดพลาดด้านเฟรมข้อมูล

PARE (Parity Error) "1" URAT ตรวจพบความผิดพลาดทางพาริตี้
"0" ไม่พบความผิดพลาดทางพาริตี้

OVFE (Overrun Error) "1" URAT ตรวจพบความผิดพลาดแบบโอเวอร์รัน
"0" ไม่พบความผิดพลาดแบบโอเวอร์รัน

RxRD (Receive Data Ready) "1" มีการรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์
"0" ไม่มีข้อมูล

รีจีสเตอร์ตำแหน่ง 06H : รีจีสเตอร์แสดงสถานะของโมเด็ม

ใช้เพื่อกำหนดสถานะสัญญาณอินพุท ของพอร์ตอนุกรม RS-232 ซึ่งได้แก่ สัญญาณ DCD,DSR,CTS และ RI สำหรับการเชื่อมต่อใช้งานแบบอเนกประสงค์ ดังมีรายละเอียดหน้าที่ของแต่ละบิตต่อไปนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

DCD ใช้แสดงสภาวะของขา DCD
"1" แสดงว่าที่ขา DCD เป็นลอจิก "1"
"0" แสดงว่าที่ขา DCD เป็นลอจิก "0"

RI ใช้แสดงสภาวะของขา RI
"1" แสดงว่าที่ขา RI เป็นลอจิก "1"
"0" แสดงว่าที่ขา RI เป็นลอจิก "0"

DSR ใช้แสดงสภาวะของขา DSR

"1" แสดงว่าที่ขา DSR เป็นลอจิก "1"

"0" แสดงว่าที่ขา DSR เป็นลอจิก "0"

DCTS (Delta Clear To Send) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต CTS

"1" แสดงว่าบิต CTS (Cler To Send) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DDSR (Delta Data Set Ready) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต DSR

"1" แสดงว่าบิต DSR (Dalta Set Ready) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DRI (Dalta Ring Indicator) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต RI

"1" แสดงว่าบิต RI (Dalta Ring Indicator) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DDCD (Delta Data Carrier Detect) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต DDCD

"1" แสดงว่าบิต CTS (Cler To Send) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DCTS (Delta Data Cler To Send) ใช้แสดงสภาวะของขา CTS

"1" แสดงว่าที่ขา CTS เป็นลอจิก "1"

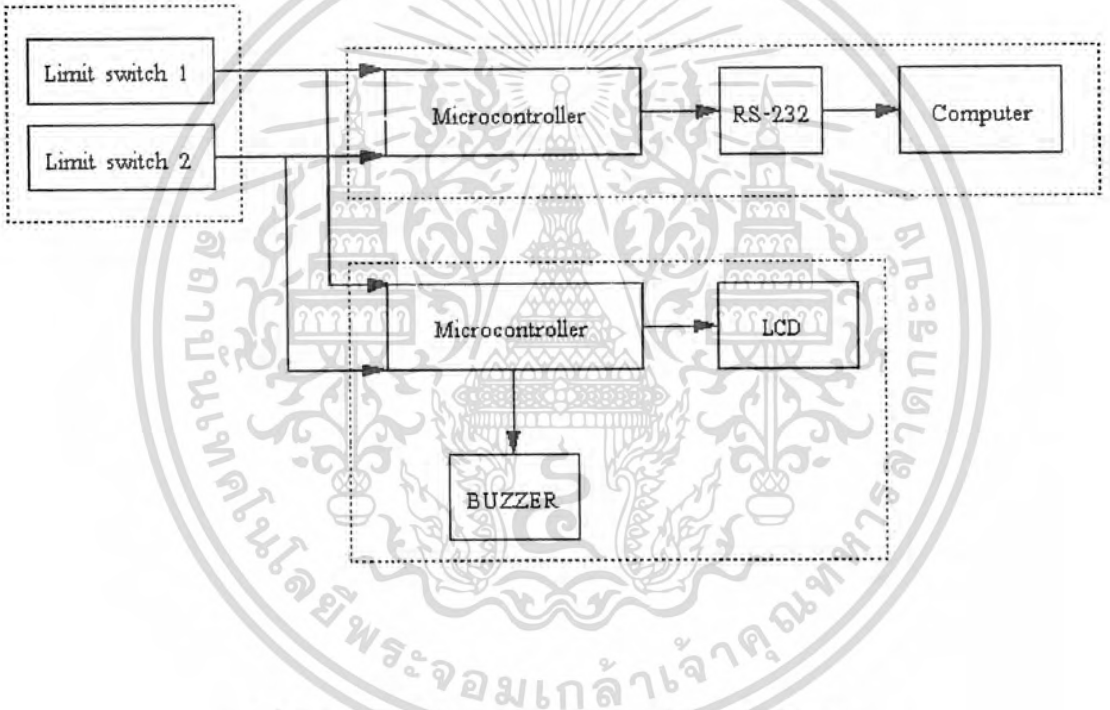
"0" แสดงว่าที่ขา CTS เป็นลอจิก "0"

รีจีสเตอร์ตำแหน่ง 07H : รีจีสเตอร์สำหรับเก็บข้อมูลชั่วคราว

ทำหน้าที่เป็นหน่วยความจำแรมขนาด 1 ไบต์ การอ่านและเขียนข้อมูลที่รีจีสเตอร์ตัวนี้ไม่ส่งผลใดๆต่อการใช้งาน URAT

บทที่ 3
การออกแบบซอฟท์แวร์และฮาร์ดแวร์

3.1 ส่วนประกอบหลักของโครงงาน



รูปที่ 3-1 บล็อกไดอะแกรมแสดงการทำงานของโครงงาน

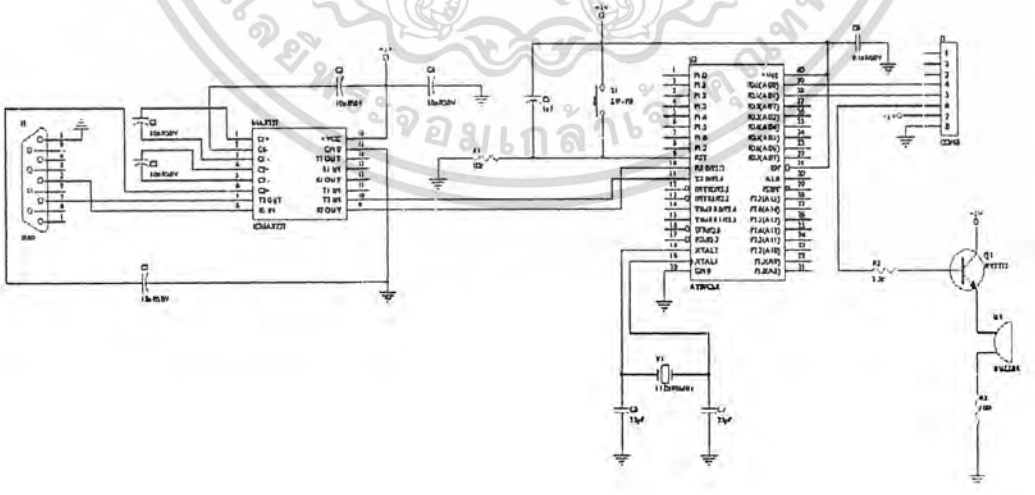
อธิบายบล็อคไดอะแกรมของการตรวจจับความเร็ว

วงจรเราจะประกอบไปด้วยชุดลิมิตสวิตช์ทั้งหมด 2 ชุด โดยมีการวางในตำแหน่งที่ตรงกับเส้นทาง โดยเส้นทางละสองชุด โดยเป็นจุดต้นและจุดปราย เมื่อได้รับสัญญาณเข้ามาก็จะทำการส่งสัญญาณไปที่ไมโครคอนโทรลเลอร์ที่พอร์ทอินพุทของ MCS AT89C51 และส่งผ่านไปยัง IC LM 324 โดยการอินเตอร์เฟส RS 232 โดยคำนวณผ่านคอมพิวเตอร์พร้อมกับแสดงผลเป็นความเร็ว โดยมีความเร็วอยู่ที่ 8 มิลลิเซ็คโดยเราจะเขียนโปรแกรมโดยมีหลักการเขียนโฟรชาร์ตคร่าวๆ ดังในรูป และสัญญาณส่วนหนึ่งจาก IC LM324 จะถูกส่งไปยัง MCS AT89C51 ตัวที่สองที่ทำหน้าที่ควบคุม LCD ซึ่ง LCD ตัวนี้จะทำหน้าที่แสดงจังหวะในการทำงานของโปรเจ็ค แล้วเราจะส่งไปแสดงผลที่ LCD ต่อไป โดย LCD ที่ใช้นี้มีขนาด 16 ตัวอักษร 1 บรรทัด ซึ่งจะแสดงรายละเอียดต่อไปพร้อมกับ BUZZER ให้จังหวะในการทำงานเป็นเสียง

3.2 ส่วนของฮาร์แวร์ทที่ใช้ในโครงงาน

ฮาร์แวร์ที่ใช้ในโครงงานจะมีส่วนของลิมิตสวิตช์ ส่วนของตัวไมโครคอนโทรลเลอร์ 2 ชุด ซึ่งประกอบด้วยส่วนควบคุมจอแสดงผล และส่วนที่ใช้ทำการจับเวลา

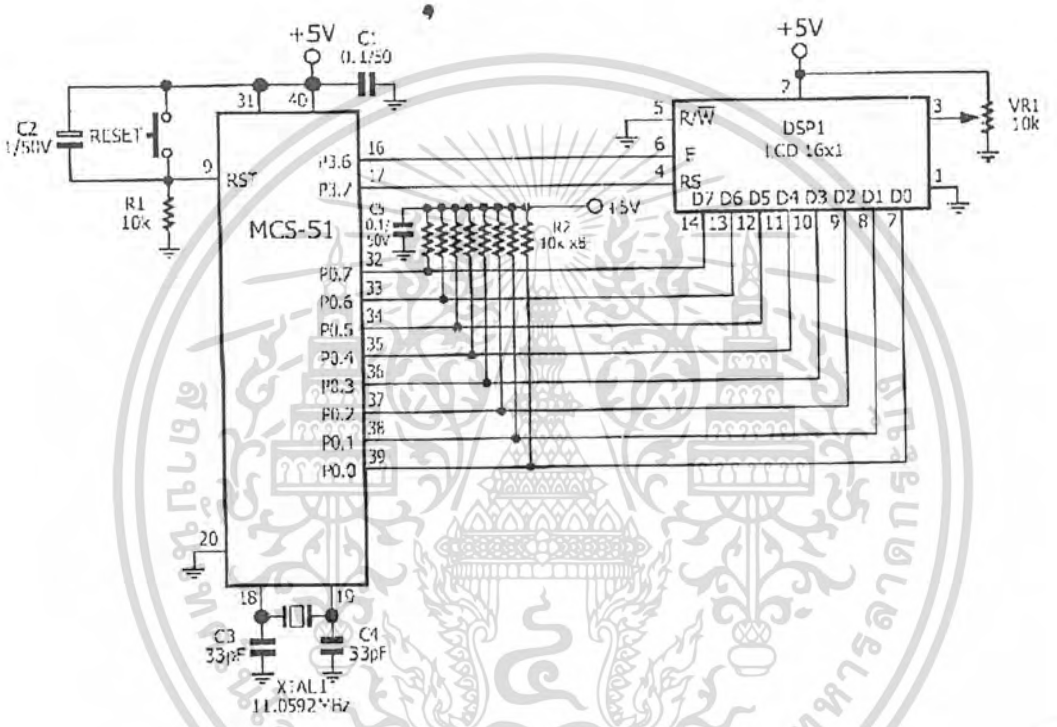
3.2.1 ส่วนของตัวตรวจจับเวลา



รูปที่ 3-2 วงจรตรวจจับเวลาโดยใช้ MCS-51 ต่อกับ MAX232 เพื่อต่อใช้งานพอร์ตอนุกรม

ส่วนของวงจรภาคนี้เป็นการต่อ MCS-51 กับ MAX232 เพื่อต่อใช้งานกับพอร์ตอนุกรมของคอมพิวเตอร์เพื่อใช้ในการจับเวลาให้มีความแน่นอนมากยิ่งขึ้น

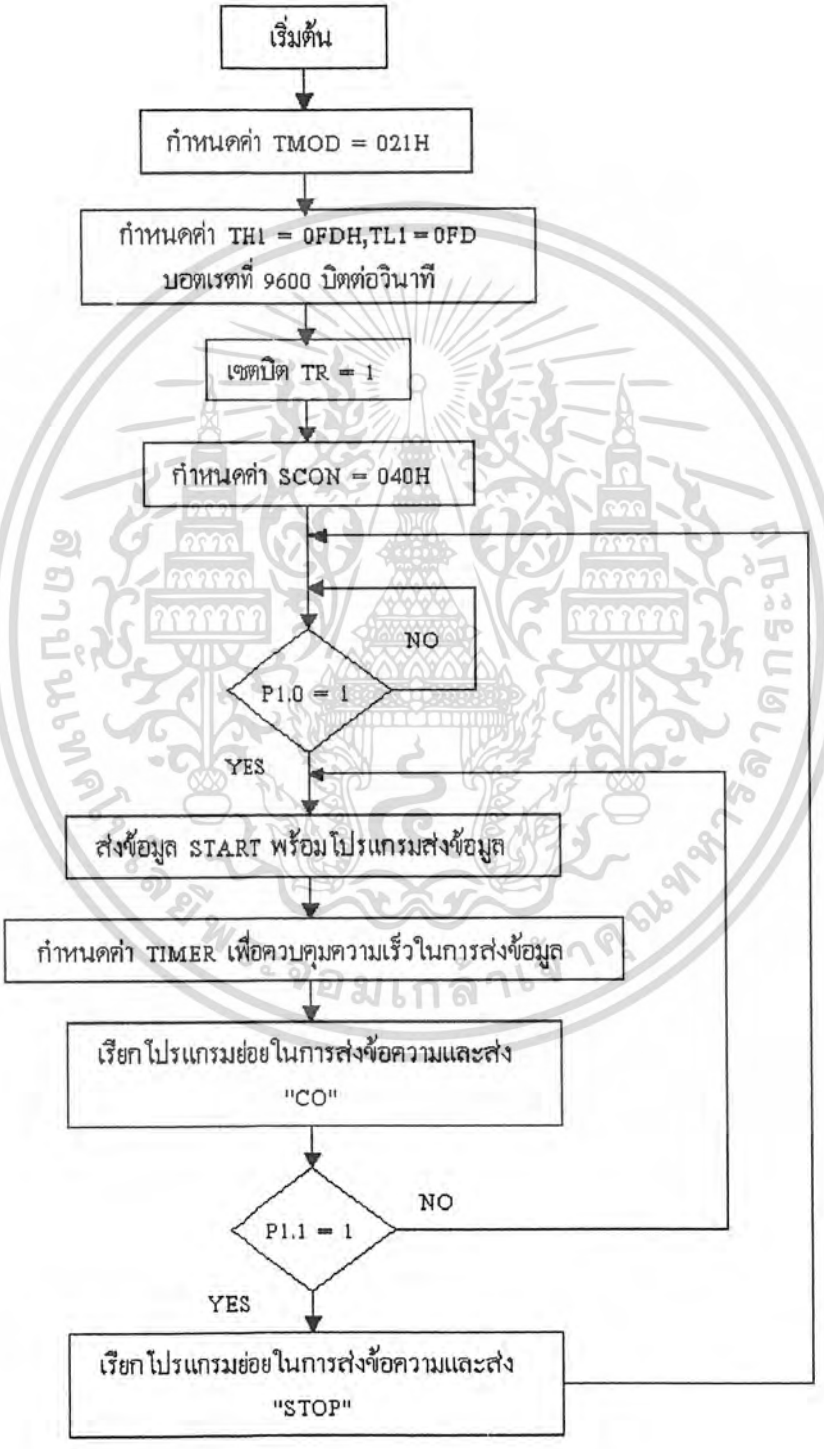
3.2.2 ส่วนของตัวควบคุมจอแสดงผลแบบแอลซีดี



รูปที่ 3-3 ส่วนของตัวควบคุมจอแสดงผลแบบแอลซีดี

ส่วนนี้เป็นการต่อใช้งาน MCS-51 ร่วมกันกับจอแสดงผลแบบแอลซีดี จากรูปที่ 3-5 เป็นการต่อใช้งานแบบง่ายๆ ของส่วนตัวควบคุมการแสดงผลบนจอแอลซีดี

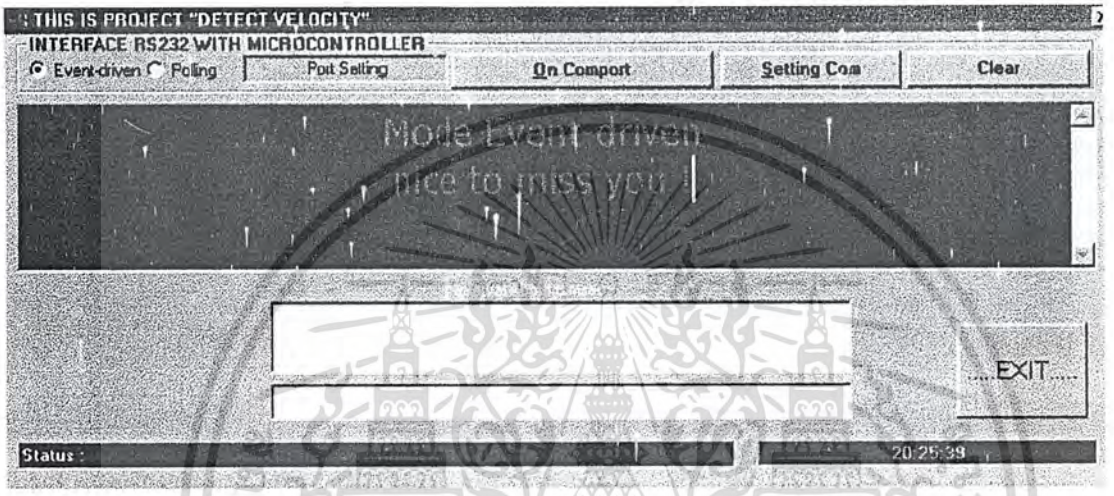
3.3 การออกแบบซอฟแวร์



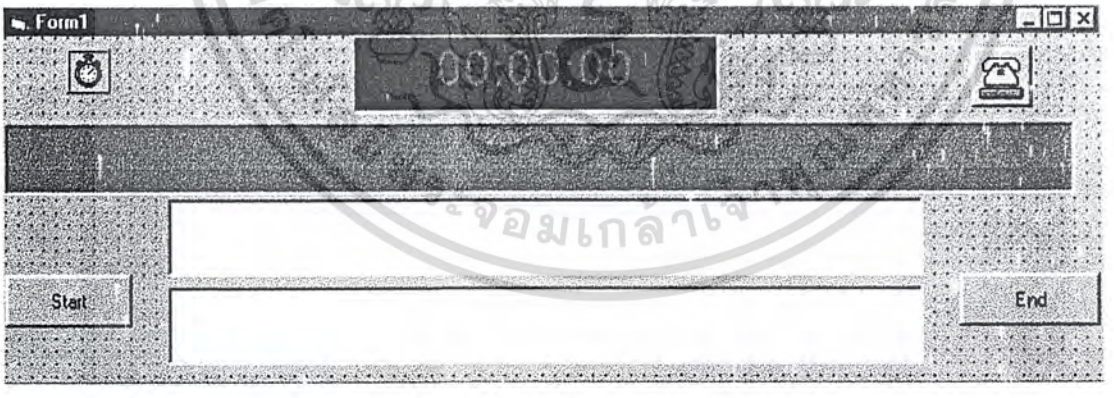
รูปที่ 3.4 โฟลว์ชาร์แสดงการทำงานของโปรแกรม

บทที่ 4
การทดลองการทำงานของโครงงาน

4.1 โปรแกรมที่ใช้งานกับตัวจับเวลา



รูปที่ 4.1 โปรแกรมทดลองการทำงานของตัวตรวจจับเวลา



รูปที่ 4.2 แสดงการจับเวลาของตัวจับเวลาของตัวจับเวลา

4.2 ผลการทดลอง

เมื่อมีวัตถุเคลื่อนที่ผ่านลิมิตสวิตช์ 1 โปรแกรมจากรูป 4.2 จะเริ่มจับเวลาเริ่มต้น จนกระทั่งวัตถุเคลื่อนที่ผ่านลิมิตสวิตช์ 2 โปรแกรมจากรูป 4.2 จะหยุดนับ และจะนำค่าเวลาที่จับได้ไปคำนวณแสดงเป็นค่าความเร็วออกมาตามรูปที่ 4.3 โดยแสดงเป็นหน่วยเมตรต่อวินาที และกิโลเมตรต่อชั่วโมง

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

หลักการทำงาน

จากบล็อกไดอะแกรม จะประกอบด้วย ลิมิตสวิตช์ มี 2 ชุด แต่ละชุดจะประกอบไปด้วยตัวรับและตัวส่ง จะทำงานตลอดเวลา เมื่อมีวัตถุตัดผ่านก็จะได้ระดับสัญญาณที่เปลี่ยนไป จากนั้นส่งต่อไปให้กับ ไมโครคอลโทรลเลอร์ ซึ่งมี 2 ตัว ตัวหนึ่งจะทำหน้าที่ ตรวจสอบวัตถุที่วิ่งตัดผ่าน ลิมิตสวิตช์ แล้วส่งสัญญาณสู่เครื่องคอมพิวเตอร์ โดยใช้การอินเตอร์เฟส RS232 ICL232 ทางพอร์ตอนุกรมสู่เครื่องคอมพิวเตอร์ โดยคอมพิวเตอร์จะทำหน้าที่คำนวณความเร็วออกมาเป็น km / L โดยใช้สูตรดังต่อไปนี้

$$v = \frac{s}{t} \times \frac{3600}{1000}$$

สำหรับการคำนวณบนาคอมพิวเตอร์ โดยอาศัยสูตร

$$v = \frac{s}{t}$$

s คือ ระยะทาง หน่วย เมตร

t คือ เวลา หน่วย วินาที

โดยเราจะกำหนดให้ระยะห่างระหว่างเซ็นเซอร์ห่างกันเป็นระยะเท่าไรก็ได้ ซึ่งกำหนดเป็นค่าคงที่จากนั้นทำการเปลี่ยนแปลงทางเวลา จากความเร็วที่แตกต่างกันของวัตถุที่วิ่งตัดผ่าน มาเข้าสูตร $v = s / t$ โดยได้ความเร็วแล้วนำมาเปลี่ยนหน่วยเป็น km / sec สำหรับสเกลในการจับเวลา ความละเอียด จะอยู่ที่ความเร็ว 8 ms อาจคลาดเคลื่อนเล็กน้อย ซึ่งถ้าหากเราสามารถที่จะทำให้สเกลลดลงจาก 8 ms ได้เท่าใดก็จะทำให้ความสามารถในการจับเวลาได้ถูกต้องยิ่งขึ้น หรือเราสามารถเพิ่มระยะห่างระหว่างเซ็นเซอร์ทั้งสองชุดให้มากขึ้น ก็จะทำให้สามารถจับความเร็วได้สูงขึ้น

ส่วนไมโครคอลโทรลเลอร์อีกตัวหนึ่งนั้นจะใช้ในการควบคุมการแสดงผลของ แอลซีดี 16×1 โดย แอลซีดี จะแสดงในส่วนของการเซ็ทอัพเครื่องและบอกสถานะในการทำงานของวงจร และพร้อมกันนั้นวงจรก็จะมี Buzzer ซึ่งจะแสดงจังหวะในการทำงานโดยเสียง

สำหรับ ซอฟต์แวร์ ในการคำนวณและแสดงผล ถูกพัฒนาขึ้นโดยใช้โปรแกรม Visual Basic 6 ส่วนของความละเอียดในการจับที่ 8 ms ได้มาจาก ไมโครคอลโทรลเลอร์ โดยอาศัย Time ส่ง Signal มายังคอมพิวเตอร์โดยผ่านการ Interface RS232 และใช้การตรวจจับสัญญาณที่ส่งโดยการตรวจ Port open โดยได้ความเร็วเฉลี่ยสูงสุดอยู่ที่ 8 ms โดยทำงานได้อย่างมีเสถียรภาพ และมีการเปรียบเทียบกับเวลาปกติด้วย

เอกสารอ้างอิง

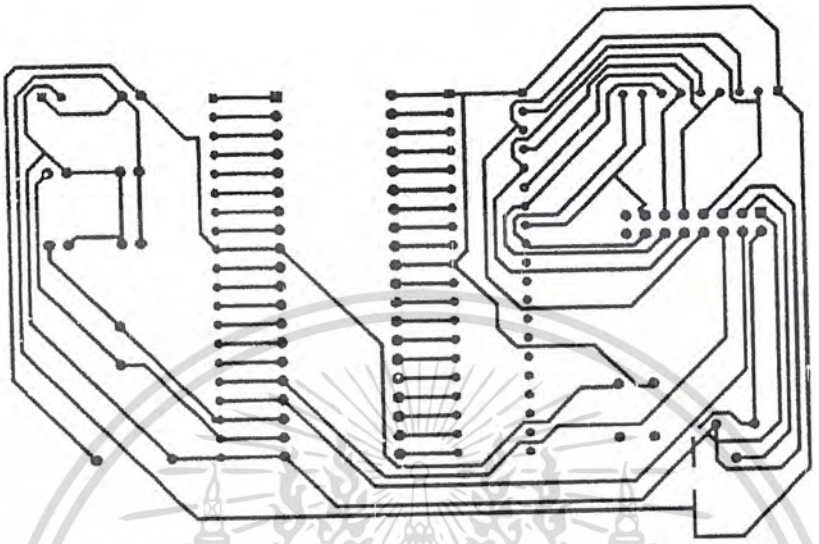
รองศาสตราจารย์ สมยศ จุณณะปิยะ Microcontroller Application MCS-51.พิมพ์ครั้งที่ 3 : ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2543

วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรวิจิตรวิไล. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์.พิมพ์ครั้งที่ 3.กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์. 2543

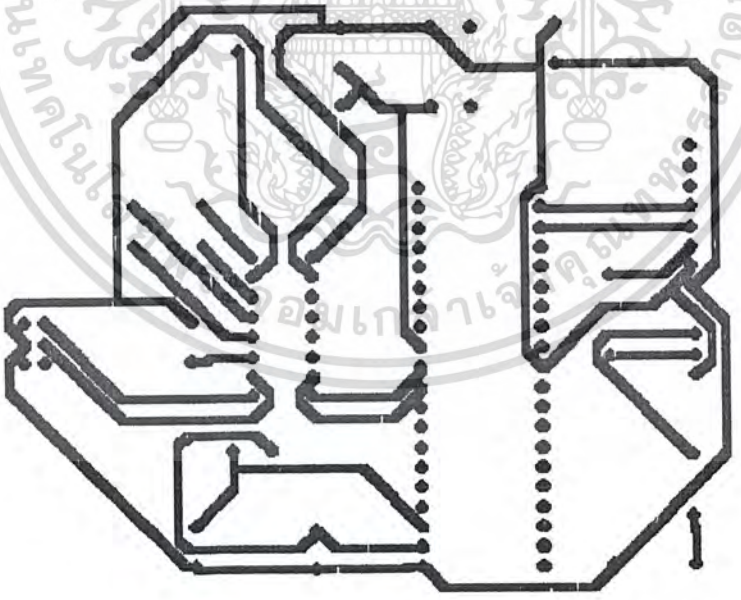
ภาคผนวก

รายการอุปกรณ์

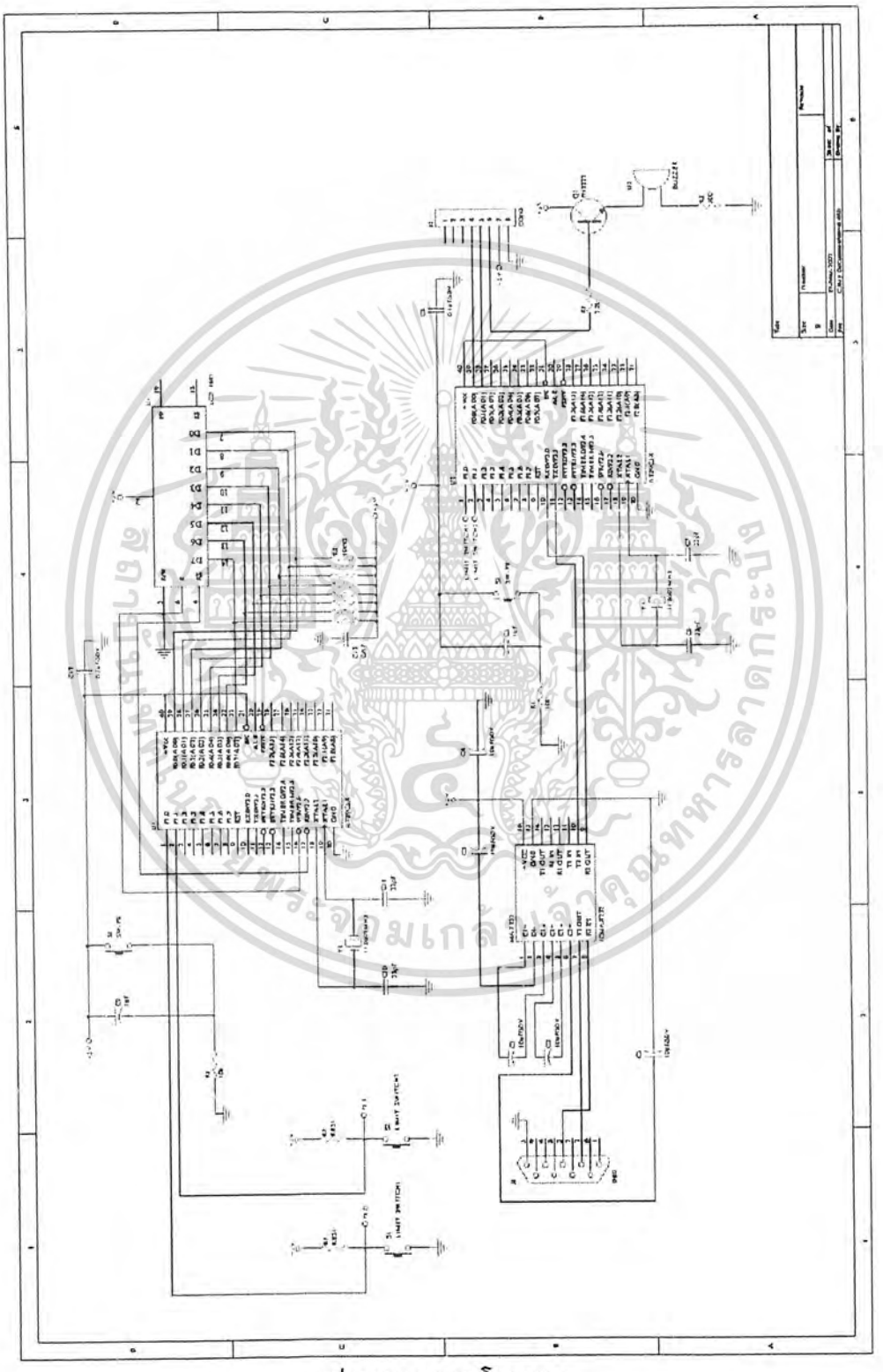
1. MCS-AT89C52A
2. LCD 16 X 1
3. SOCKET 8 PIN ; 40 PIN
4. TRANSFORMER 0.25A 9V
5. CRISTAL 11.059 MHZ.
6. ZENER DIODE 9.1V 0.5W
7. VR 10K
8. R-NETWOEK 10K
9. อินฟราเรด
10. ตัวต้านทาน 4.7K,1K,22K,5K,,100K,330K,1K,10K,
11. ตัวเก็บประจุ 0.047UF,0.1UF,47UF,1000UF,10UF,1UF,33PF
12. ไดโอด 1N4148,1N4002,1N4001
13. TRANSISTOR BC549,H1061
14. LED
15. LIMIT SWITCH
16. IC. 7805
17. BOARD PLASTIC
18. HEAT SINK
19. COPPER PRINT



ลายวงจรของตัวไมโครคอนโทรลเลอร์



ลายวงจรของพอร์ต RS-232 ต่อกับไมโครคอนโทรลเลอร์



รูปวงจรรวมของโครงงาน

```
;*******************************************************************************
; Program               : LCD Display 01
; Description    : LCD Display Show , 8 Char from ROM , Shift Display
; For                   : MCS-51 Microcontroller Training System
; Filename              : lab1501.asm
; Assembler             : RAD51
; Copyright (C) 2000 Innovative Experiment Co.,Ltd.
;*******************************************************************************

;-------------------------------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------------------------------
LCD_EN          BIT     P3.6
LCD_RS          BIT     P3.7
;-------------------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------------------
LCD_ADDR        EQU     030H
LCD_DATA        EQU     031H
;-------------------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------------------

                ORG     0000H
                MOV     P0,#00000000B
                CLR     LCD_EN
                CLR     LCD_RS
                CLR     P3.0
                MOV     R3,0FFH
```

```
MAIN:   ACALL   INIT_LCD
LOOP:   MOV     LCD_ADDR,#000H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#TITLE_1
        ACALL   WRLINE_LCD
        MOV     LCD_ADDR,#040H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#TITLE_2
        ACALL   WRLINE_LCD
        ACALL   DELAY_1s
        ACALL   DELAY_1s
        LCALL   BEEP2
        LCALL   BEEP2
        LCALL   BEEP1
        ACALL   LCD_CLR
        MOV     LCD_ADDR,#000H
        MOV     DPTR,#NAME_1
        ACALL   WRLINE_LCD
        MOV     DPTR,#NAME_3
        ACALL   WRLINE_LCD
        MOV     LCD_ADDR,#040H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#NAME_2
        ACALL   WRLINE_LCD
        MOV     DPTR,#NAME_4
        ACALL   WRLINE_LCD
        ACALL   LCD_OFF
        ACALL   DELAY_1s
```

```
                ACALL   LCD_ON
                ACALL   DELAY_1s
                ACALL   LCD_OFF
                ACALL   DELAY_1s

                ACALL   LCD_ON
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                LCALL   BEEP2
                LCALL   BEEP2
                LCALL   BEEP1
                MOV     R4,#8
LOOP_LCD_L_SHF: ACALL LCD_LSHF
                ACALL   DELAY_100ms
                ACALL   DELAY_100ms
                DJNZ    R4,LOOP_LCD_L_SHF
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_5
                ACALL   WRLINE_LCD
                MOV     LCD_ADDR,#040H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_6
                ACALL   WRLINE_LCD
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                LCALL   BEEP2
```

```
                    MOV     R4,#8
LOOP_LCD_R_SHF:     ACALL   LCD_RSHF
                    ACALL   DELAY_100ms
                    ACALL   DELAY_100ms
                    DJNZ    R4,LOOP_LCD_R_SHF
                    ACALL   DELAY_1s
                    ACALL   DELAY_1s
                    LCALL   BEEP2
                    LCALL   BEEP2
                    LCALL   BEEP1
                    ACALL   LCD_CLR
                    ACALL   LCD_CLR
                    MOV     LCD_ADDR,#000H
                    MOV     DPTR,#NAME_1
                    ACALL   WRLINE_LCD
                    MOV     DPTR,#PC3
                    ACALL   WRLINE_LCD
                    MOV     LCD_ADDR,#040H
                    ACALL   SET_ADDR_LCD
                    MOV     DPTR,#NAME_2
                    ACALL   WRLINE_LCD
                    MOV     DPTR,#PC4
                    ACALL   WRLINE_LCD
                    MOV     R4,#0H
                    MOV     R4,#8

LOOP_LCD_L_SHF2:    ACALL   LCD_LSHF
                    ACALL   DELAY_100ms
                    ACALL   DELAY_100ms
```

```
                DJNZ    R4,LOOP_LCD_L_SHF2
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#PC1
                ACALL   WRLINE_LCD
                MOV     LCD_ADDR,#040H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#PC2
                ACALL   WRLINE_LCD
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                MOV     R4,#0H
                MOV     R4,#8
LOOP_LCD_R_SHF2: ACALL  LCD_RSHF
                ACALL   DELAY_100ms
                ACALL   DELAY_100ms
                DJNZ    R4,LOOP_LCD_R_SHF2
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                LCALL   BEEP1
                LCALL   BEEP1
                LCALL   BEEP1
                LCALL   BEEP1
                ACALL   DELAY_1s
                LCALL   BEEP1
                LCALL   BEEP1
                LCALL   BEEP1
                LCALL   BEEP1
```

```
        ACALL   DELAY_1s
        LCALL   BEEP1
        LCALL   BEEP1
        LCALL   BEEP1
        LCALL   BEEP1
        ACALL   DELAY_1s
        LCALL   BEEP1
        LCALL   BEEP1
        LCALL   BEEP1
        LCALL   BEEP1
        ACALL   DELAY_10Ms
        LCALL   BEEP2
        LCALL   BEEP2
S1:     JB      P1.0,$
        ACALL   BEEP1
        ACALL   BEEP1
        ACALL   BEEP1
SA:     ACALL   LCD_CLR
        MOV     LCD_ADDR,#000H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#ON1
        ACALL   WRLINE_LCD
        MOV     LCD_ADDR,#040H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#ON2
        ACALL   WRLINE_LCD
        JB      P1.1,$
        ACALL   BEEP2
        LCALL   DELAY_10ms
```

```
        ACALL   BEEP2
        ACALL   LCD_CLR
        MOV     LCD_ADDR,#000H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#OFF1
        ACALL   WRLINE_LCD
        MOV     LCD_ADDR,#040H
        ACALL   SET_ADDR_LCD
        MOV     DPTR,#OFF2
        ACALL   WRLINE_LCD
        AJMP    S1

BEEP1:  SETB    P3.0
        DJNZ    R3,BEEP1
        ACALL   DELAY_10ms
        ACALL   DELAY_10ms
        CLR     P3.0
        RET
BEEP2:  ACALL   DELAY_10ms
        ACALL   DELAY_10ms
        ACALL   BEEP1
        ACALL   BEEP1
        ACALL   BEEP1
        ACALL   BEEP1
        RET
```

```
;-------------------------------------------------------------------
; LCD Initialize
;-------------------------------------------------------------------
INIT_LCD:       ACALL   DELAY_100ms
                CLR     LCD_RS
                MOV     P0,#00111000B
                ACALL   LCD_CLK
                ACALL   DELAY_10ms
                MOV     P0,#00111000B
                ACALL   LCD_CLK
                ACALL   LCD_OFF
                ACALL   LCD_CLR
                MOV     P0,#00000110B
                ACALL   LCD_CLK
                ACALL   LCD_HOME
;-------------------------------------------------------------------
; LCD Clear Display
;-------------------------------------------------------------------
LCD_CLR:        CLR     LCD_RS
                MOV     P0,#00000001B
                ACALL   LCD_CLK
```

```
;-------------------------------------------------------------------
; LCD Return Home
;-------------------------------------------------------------------
LCD_HOME:       CLR     LCD_RS
                MOV     P0,#00000010B
                ACALL   LCD_CLK
                RET

;-------------------------------------------------------------------
; LCD Display Off
;-------------------------------------------------------------------
LCD_OFF:        CLR     LCD_RS
                MOV     P0,#00001000B
                ACALL   LCD_CLK
                RET

;-------------------------------------------------------------------
; LCD Clk
;-------------------------------------------------------------------
LCD_CLK:        SETB    LCD_EN
                ACALL   LCD_DELAY
                CLR     LCD_EN
                ACALL   LCD_DELAY
                RET
```

```
;-------------------------------------------------------------------
; LCD Display On
;-------------------------------------------------------------------
LCD_ON:         CLR     LCD_RS
                MOV     P0,#00001100B
                ACALL   LCD_CLK
                RET

;-------------------------------------------------------------------
; LCD Cursor On
;-------------------------------------------------------------------
LCD_BLINK:      CLR     LCD_RS
                MOV     P0,#00001111B
                ACALL   LCD_CLK
                RET

;-------------------------------------------------------------------
; LCD Left Shift Display
;-------------------------------------------------------------------
LCD_LSHF:       CLR     LCD_RS
                MOV     P0,#00011000B
                ACALL   LCD_CLK
                RET
```

```
DELAY_10ms_1:   MOV     R6,#0E6H
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
                RET
DELAY_1s:       MOV     R5,#100
DELAY_1s_1:     ACALL DELAY_10ms
                DJNZ  R5,DELAY_1s_1
                RET
```

```
;-------------------------------------------------------------------------
; LCD Right Shift Display
;-------------------------------------------------------------------------
LCD_RSHF:       CLR     LCD_RS
                MOV     P0,#00011100B
                ACALL   LCD_CLK
                RET

;-------------------------------------------------------------------------
; Set LCD Address
; I/P:          LCD_ADDR
;-------------------------------------------------------------------------
SET_ADDR_LCD:   CLR     LCD_RS
                MOV     A,LCD_ADDR
                SETB    ACC.7
                MOV     P0,A
                ACALL   LCD_CLK
                RET
```

```
;-------------------------------------------------------------------------
; Write Character to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------------------
WRCHAR_LCD:     SETB    LCD_RS
                MOV     P0,LCD_DATA
                ACALL   LCD_CLK
                ACALL   LCD_ON
                RET

;-------------------------------------------------------------------------
; Write Line of 8 Character from ROM
; I/P:          DPTR : Locate ROM Address
;-------------------------------------------------------------------------
WRLINE_LCD:     MOV     R0,#0
WRLINE_LCD_1:   SETB    LCD_RS
                CLR     A
.               MOVC    A,@A+DPTR
                MOV     P0,A
                ACALL   LCD_CLK
                INC     DPTR
                INC     R0
                CJNE    R0,#8,WRLINE_LCD_1
                ACALL   LCD_ON
                RET
```

```
;-------------------------------------------------------------------------
; Dummy Delay time LCD_DELAY, 10m, 100m, 1s
;-------------------------------------------------------------------------
LCD_DELAY:          MOV         R7,#002

LCD_DELAY_1:        MOV    R6,#0E6H
LCD_DELAY_2:        NOP
                    NOP
                    DJNZ        R6,LCD_DELAY_2
                    DJNZ        R7,LCD_DELAY_1
                    RET

DELAY_10ms:         MOV         R7,#010
DELAY_10ms_1:       MOV         R6,#0E6H
DELAY_10ms_2:       NOP
                    NOP
                    DJNZ        R6,DELAY_10ms_2
                    DJNZ        R7,DELAY_10ms_1
                    RET

DELAY_100ms:        MOV         R7,#100
DELAY_100ms_1:      MOV         R6,#0E6H
DELAY_100ms_2:      NOP
                    NOP
                    DJNZ        R6,DELAY_100ms_2
                    DJNZ        R7,DELAY_100ms_1
                    RET
```

```
DELAY_1s:       MOV     R5,#100
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5,DELAY_1s_1
                RET

;-------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------
;                                01234567
TITLE_1:        DB      'Hi Are y'
TITLE_2:        DB      'ou mary?'

TITLE_3:        DB      'Test Pro'
TITLE_4:        DB      'ject ...'
NAME_1:         DB      'My name '
NAME_2:         DB      'is COMBO'
NAME_3:         DB      'This is '
name_4:         DB      'Version1'
TITLE_5:        DB      'Check co'
TITLE_6:        DB      'nnector?'
PC1:            DB      'PC READY'
PC2:            DB      '...OK...'

PC3:            DB      'Test You'
PC4:            DB      'r PC_?OK'

ON1:            DB      'SENSOR1_'
ON2:            DB      'START_OK'
```

```
OFF1:       DB      'FINISH_S'
OFF2:       DB      'ee at PC'

SPACE:      DB      'SENSOR_2'
```

```
                        PROJECT 2N
;                       MR. PHONGPUN      NAMSOMBOON
                        MR. PRAPAN        NAMTAM
                        PROGRAM CONTROL TIMER
;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
            BUFFER      EQU   030H
;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------
            ORG         0000H

MAIN:       MOV         TMOD,#021H
            MOV         TH1,#0FDH
            MOV         TL1,#0FDH
            SETB        TR1
            MOV         SCON,#040H
            MOV         R2,#0H
            MOV         R3,#0H
            AJMP        SPEED1
SPEED1:     JB          P1.0,SPEED1
            MOV         DPTR,#SENSOR1
            ACALL       TX_TEXT
            ACALL       DELAY_10MS
```

```
count:      CLR     TF0
            CLR     TR0
            MOV     TMOD,#021H
            MOV     TH0,#0FFH
            MOV     TL0,#071H
            SETB    TR0
            MOV     TH0,#0FFH
            MOV     TL0,#071H
            CLR     TF0
            SETB    TR0
            ACALL   SEND
            MOV     DPTR,#SENSOR3
            ACALL   TX_TEXT
PEED2:      JB      P1.1,COUNT
            ACALL   DELAY_10MS
            MOV     DPTR,#SENSOR2
            ACALL   TX_TEXT
            LJMP    SPEED1
;-------------------------------------------------------------------
; TX Serial Text from ROM Pointer
;-------------------------------------------------------------------
TX_TEXT:    CLR     TI
TX_LOOP:    CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
            CJNE    A,#0FFH,TX_CHAR
            RET
```

```
TX_CHAR:    MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            AJMP    TX_LOOP


SEND:       CLR     TF0
            CLR     TR0
            MOV     TMOD,#021H
            MOV     TH1,#0FDH
            MOV     TL1,#0FDH
            SETB    TR1
            MOV     SCON,#040H
            RET
;-------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------


GOT_TEXT1:      DB      '';,0FFH
GOT_TEXT2:      DB      '';,00AH,00DH,0FFH
SENSOR1:        DB      'START',0FFH;
SENSOR2:        DB      'STOP',0FFH;
SENSOR3:        DB      'CO',0FFH;DELAY_10ms:
                MOV     R7,#010
```