บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

พื้นฐานไมโครคอนโทรลเลอร์

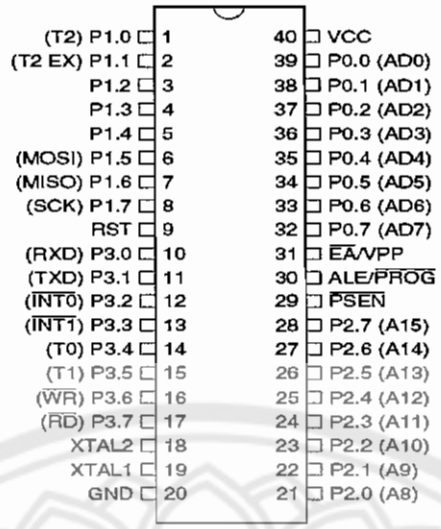
ในปัจจุบันไมโครคอนโทรลเลอร์ได้พัฒนาให้มีประสิทธิภาพสูงขึ้นสามารถประมวลผล ข้อมูลได้เร็วและมีราคาถูกลง มีการออกแบบให้ใช้งานได้ง่าย สามารถเขียนโปรแกรมสั่งงานได้หลายภาษาและมีเครื่องมืออำนวยความสะดวกเพื่อช่วยในการพัฒนาระบบจึงทำให้ไมโครคอนโทรลเลอร์เป็นส่วนประกอบที่สำคัญในวงจรอิเล็กทรอนิกส์ คอมพิวเตอร์ เครื่องคำนวณ เครื่องมือวัดและเครื่องใช้ไฟฟ้าภายในบ้าน

2.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ (Microcontroller) คือชิปประมวลผลอย่างหนึ่งซึ่งจำทำหน้าที่ประมวลผลตามโปรแกรมหรือชุดคำสั่งเหมือนกับไมโครโปรเซสเซอร์ โครงสร้างภายในจะเป็นวงจรรวมขนาดใหญ่ประกอบไปด้วย หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูล บัสควบคุม บัสที่อยู่ พอร์ตขนาน พอร์ตอนุกรม รีจิสเตอร์ หน่วยความจำ วงจรนับ วงจรจับเวลาและวงจรอื่นๆ รวมกันอยู่ภายในชิปหรือไอซี ไมโครคอนโทรลเลอร์ถูกออกแบบมาเพื่อให้ในงานควบคุมสามารถติดต่อกับอุปกรณ์อินพุตและเอาต์พุตได้สะดวกใช้งานง่ายสามารถทำงานได้โดย ใช้ชิปเดียว มีคำสั่งที่สนับสนุนในการเขียนโปรแกรมควบคุมและสามารถเข้าถึงข้อมูลระดับบิดได้

2.2 ไมโครคอนโทรลเลอร์ในตระกูล MCS-51

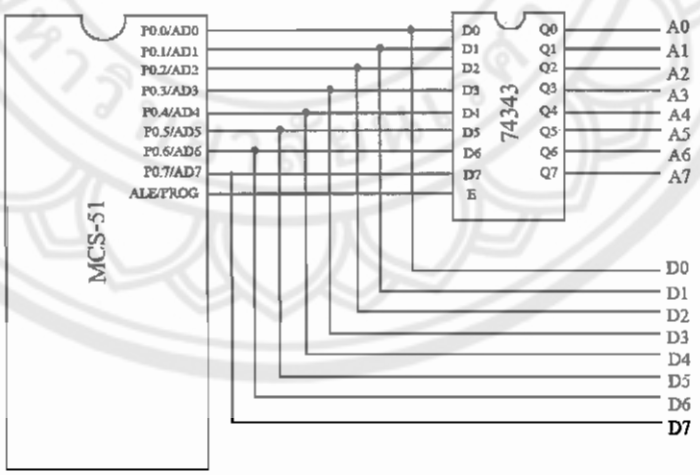
ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เป็นไมโครคอนโทรลเลอร์ที่เป็นที่นิยมในปัจจุบันเนื่องจากใช้งานง่าย ราคาถูก มีซอฟต์แวร์อำนวยความสะดวก สามารถเขียนได้ทั้งภาษาแอสแซมบลีและภาษาซี ใช้งานได้โดยชิปเดียวไม่จำเป็นต้องมีอุปกรณ์ภายนอกประกอบเนื่องจากมีหน่วยความ จำอยู่ภายในตัวไมโครคอนโทรลเลอร์และสามารถต่อใช้งานพอร์ตได้โดยตรง



รูปที่ 2.1 ขาสัญญาณของไมโครคอนโทรลเลอร์ในตระกูล MCS-51

ไมโครคอนโทรลเลอร์ในตระกูล MCS51 เป็นไมโครคอนโทรเลอร์ขนาด 8 บิต มีขาสัญญาณ จำนวน 40 ขา ประกอบด้วยขาสัญญาณต่างๆ ดังนี้

Port 0 เป็นพอร์ตอินพุต 2 ทิศทางขนาด 8 บิตคือ P0.0–P0.7 ใช้งานเป็นพอร์ตอินพุตและเอาต์พุต นอกจากนี้ยังใช้เป็นบัสข้อมูล D0–D7 และบัสที่อยู่ A0-A7 โดยใช้การมัลติเพลกซ์เพื่อสลับการทำงานโดยจะมีขา ALE เป็นสัญญาณควบคุมแสดงดังรูปที่ 2.2 ในการใช้งานพอร์ต P0 เป็นพอร์ตอินพุตและเอาต์พุตต้องต่อความต้านทานพูลอัพ (R Pull Up) ภายนอกด้วย เนื่องจากพอร์ต P0 ไม่มีความต้านทานพูลอัพภายใน



รูปที่ 2.2 การแยกสัญญาณเบัสข้อมูลและบัสที่อยู่

Port 1 เป็นพอร์ตอินพุตและเอาต์พุตขนาด 8 บิต คือ P1.0-P1.7 ใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสามารถเข้าถึงข้อมูลในระดับบิตได้

Port 2 เป็นพอร์ตอินพุตและเอาต์พุตขนาด 8 บิต คือ P2.0-P2.7 ใช้งานเป็นพอร์ตอินพุตและเอาต์พุต นอกจากนี้ยังใช้เป็นแอดเดรสบัส A8-A15 เพื่อใช้ติดต่อกับหน่วยความจำภายนอก

Vcc เป็นขาแหล่งจ่ายแรงดันไฟฟ้ากระแสตรง 5 โวลต์

Gnd ขากราวนด์ของระบบ

XTAL1 และ XTAL2 คือขาสัญญาณที่ใช้ต่อวงจรกำเนิดสัญญาณนาฬิกาเพื่อกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

RST (Reset) คือขาสัญญาณรีเซตการทำงานของไมโครคอนโทรลเลอร์เมื่อได้รับสัญญาณลอจิก 1 นานไม่ต่ำกว่า 2 แมชชีนไซเคิล

ALE (Address Latch Enable) คือขาเอาต์พุตใช้ควบคุมการแลทช์ (Latch) บัสที่อยู่ A0-A7 ของพอร์ต P0 และควบคุมการส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับหน่วยความจำ โดยไมโครคอนโทรลเลอร์จะส่งสัญญาณลอจิก 1 เมื่อต้องการให้พอร์ต P0 แลทช์บัสที่อยู่ A0-A7 จากนั้นจะส่งสัญญาณลอจิก 0 เพื่อให้พอร์ต P0 เป็นบัสข้อมูล

/PSEN (Program Store Enable) เป็นขาสัญญาณเอาต์พุตที่ใช้ควบคุมการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกทำงานที่ลอจิก 0

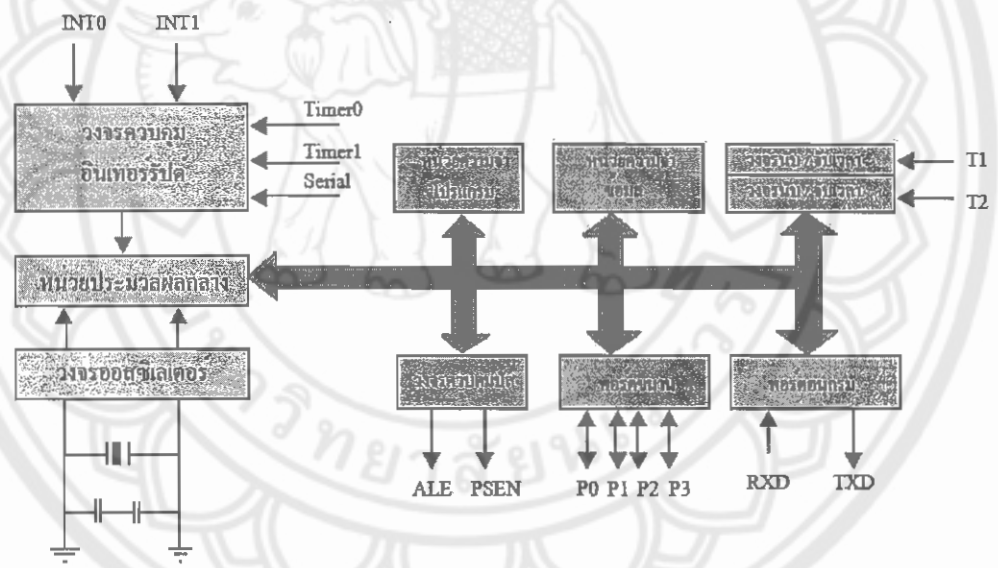
/EA (External Access) เป็นขาสัญญาณอินพุตเพื่อควบคุมให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายในหรือหน่วยความจำโปรแกรมภายนอก ถ้าต่อขา EA เข้ากับกราวนด์หรือลอจิก 0 ไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าต่อขา EA เข้ากับสัญญาณไฟ 5 โวลต์หรือลอจิก 1 ไมโครคอนโทรลเลอร์จะติดต่อกับหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์

Port 3 เป็นพอร์ตอินพุตและเอาต์พุตขนาด 8 บิตคือ P3.0-P3.7 ใช้งานเป็นพอร์ตอินพุตและเอาต์พุตนอกจากนี้พอร์ต P3 ยังมีหน้าที่อื่นๆ แสดงดังตารางที่ 2.1

ตารางที่ 2.1 หน้าที่ของขาสัญญาณของพอร์ต P3

พอร์ต	สัญญาณ	หน้าที่
P3.0	RXD	ขารับสัญญาณของการสื่อสารพอร์ตอนุกรม
P3.1	TXD	ขาส่งสัญญาณของการสื่อสารพอร์ตอนุกรม
P3.2	/INT0	ขารับสัญญาณอินเทอร์รัปต์ภายนอกตัวที่ 0
P3.3	/INT1	ขารับสัญญาณอินเทอร์รัปต์ภายนอกตัวที่ 1
P3.4	T0	ขารับสัญญาณอินพุตภายนอกของวงจรตั้งเวลาตัวที่ 0
P3.5	T1	ขารับสัญญาณอินพุตภายนอกของวงจรตั้งเวลาตัวที่ 1
P3.6	/WR	ขาสัญญาณควบคุมการเขียนข้อมูลในหน่วยความจำข้อมูลภายนอก
P3.7	/RD	ขาสัญญาณควบคุมการอ่านข้อมูลในหน่วยความจำข้อมูลภายนอก

2.3 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ในตระกูล MCS-51



รูปที่ 2.3 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์เป็นวงจรรวมขนาดใหญ่มีวงจรซับซ้อน ดังนั้นเพื่อให้ง่ายต่อการทำความเข้าใจจึงอธิบายโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ ในลักษณะบล็อกไดอะแกรม ซึ่งประกอบไปด้วยส่วนประกอบหลักต่างๆ ดังนี้

1.หน่วยประมวลผลกลางหรือซีพียู เป็นหน่วยประมวลผลขนาด 8 บิตทำหน้าที่ในการประมวลผลข้อมูลตามคำสั่งหรือโปรแกรม

2. วงจรออสซิลเลเตอร์ ทำหน้าที่สร้างสัญญาณนาฬิกาเพื่อกำหนดจังหวะการทำงานของซีพียูและวงจรภายในไมโครคอนโทรลเลอร์

3. วงจรควบคุมการอินเทอร์รัปต์ ทำหน้าที่ควบคุมการอินเทอร์รัปต์หรือการขัดจังหวะการทำงานของซีพียูจากอุปกรณ์ภายนอกและจากอินเทอร์รัปต์ภายใน

4. หน่วยความจำโปรแกรมภายใน ทำหน้าที่เก็บโปรกแกรมเพื่อส่งให้กับซีพียุทำการประมวลผลตามโปรแกรม

5. หน่วยความจำข้อมูลภายใน ทำหน้าที่เก็บข้อมูลจากการประมวลผลของซีพียู

6. วงจรควบคุมบัส ทำหน้าที่ควบคุมการทำงานของบัสข้อมูล บัสควบคุม และบัสที่อยู่

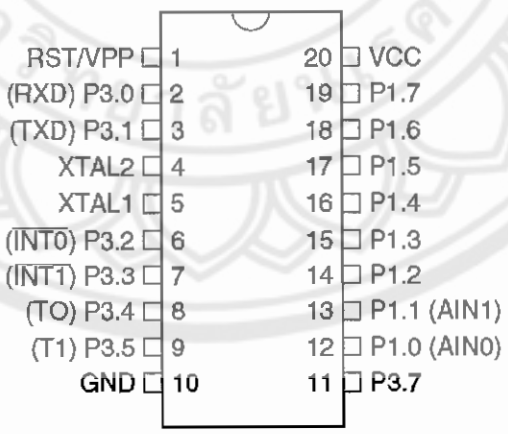
7. พอร์ตขนาน ทำหน้าที่เชื่อมต่อกับอุปกรณ์อินพุตและเอาต์พุตภายนอกแบบขนาน

8. พอร์ตอนุกรม ทำหน้าที่รับส่งข้อมูลแบบอนุกรมระหว่างอุปกรณ์ภายนอกกับไมโครคอนโทรลเลอร์ โดยขา TXD ทำหน้าที่ในการส่งข้อมูลและขา RXD ทำหน้าที่ในการรับข้อมูล

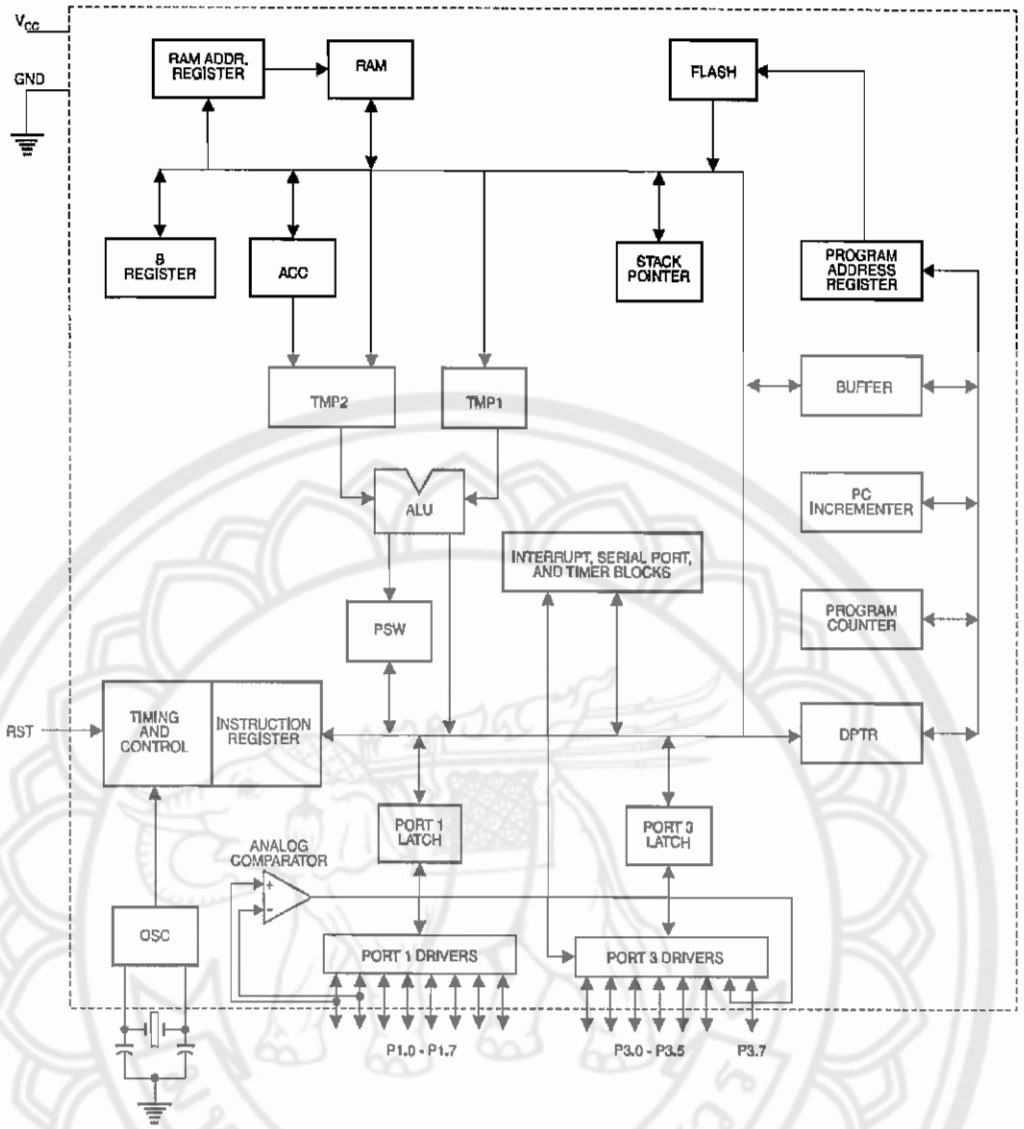
9. วงจรนับ /.จับเวลา ทำหน้าที่นับสัญญาณพัลส์ภายนอกหรือจับเวลาสัญญาณนาฬิกาภายในของระบบเป็นวงจรนับหรือจับเวลาขนาด 16 บิตจำนวน 2 วงจร

2.4 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C2051

โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C2051 มีขาสัญญาณจำนวน 20 ขา มีหน่วยความจำโปรแกรมภายในแบบแฟลชขนาด 2 กิโลไบต์ เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตสามารถต่ออินพุตและเอาต์พุตได้ 15 บิต มีการจัดตำแหน่งขาสัญญาณแสดงดังรูปที่ 2.4 และมีโครงสร้างพื้นฐานดังรูปที่ 2.5



รูปที่ 2.4 การจัดตำแหน่งขาสัญญาณของไมโครคอนโทรลเลอร์ AT89C2051



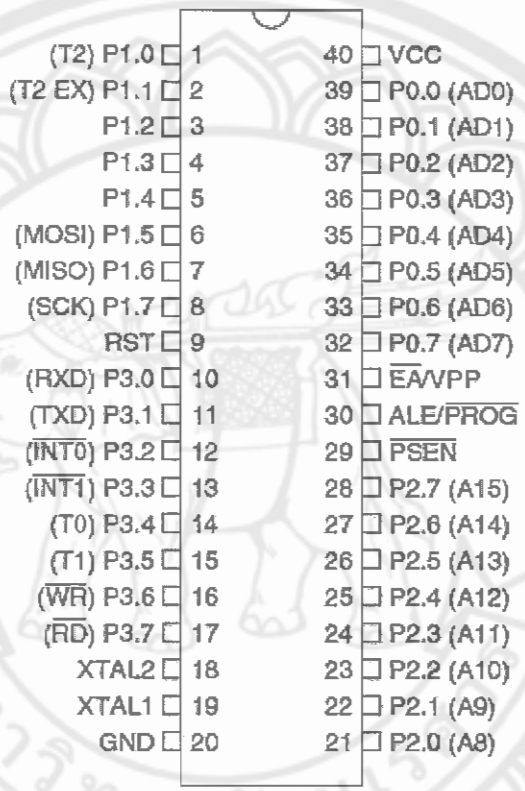
รูปที่ 2.5 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C2051

คุณสมบัติพื้นฐานของไมโครคอนโทรลเลอร์ AT89C2051

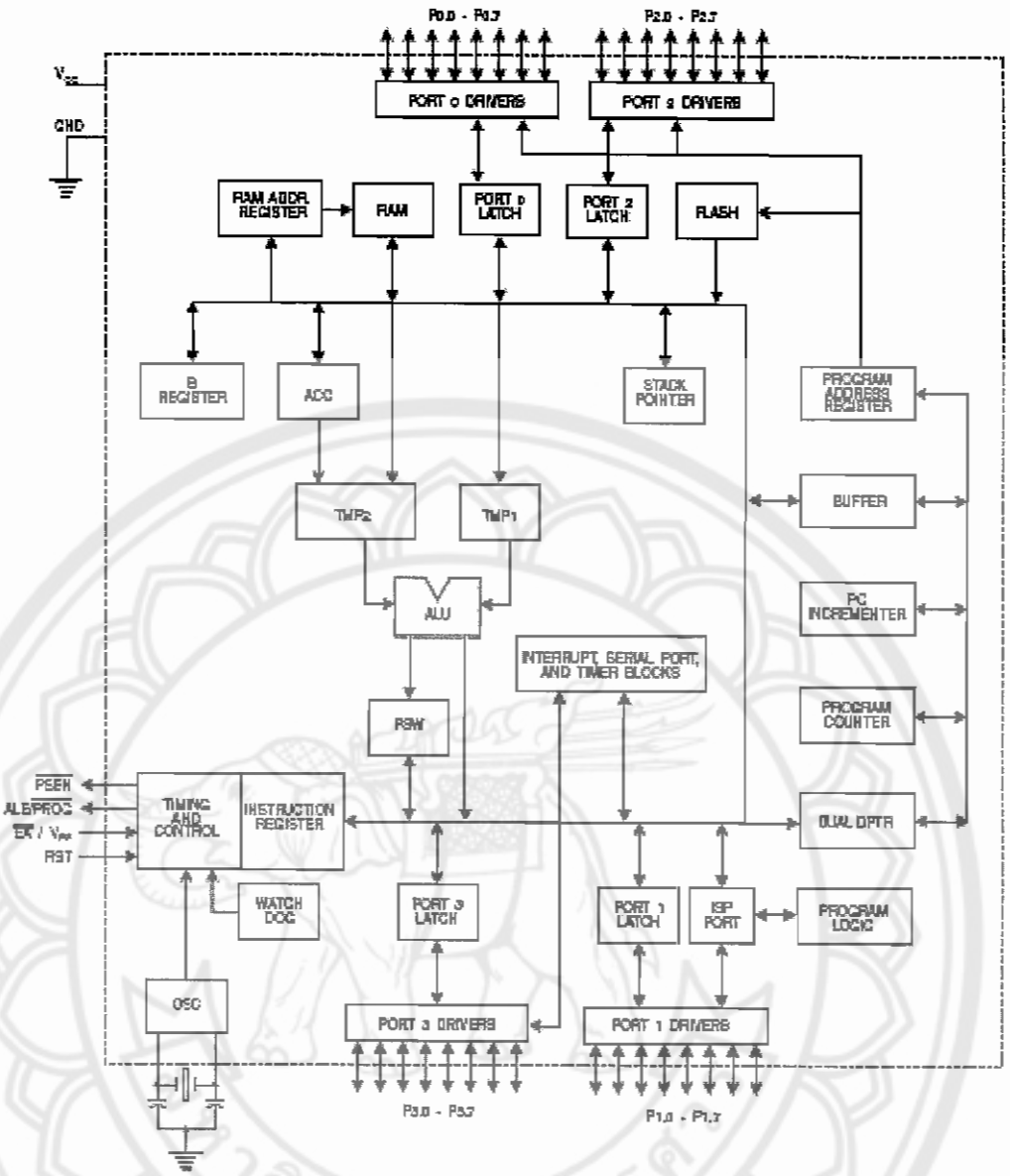
1. เป็นไมโครคอนโทรลเลอร์ที่ใช้งานร่วมกันได้กับไมโครคอนโทรลเลอร์ในตระกูล MCS-51
2. มีหน่วยความจำแบบแฟลชขนาด 2 กิโลไบต์ สามารถลบและเขียนได้ถึง 1,000 ครั้ง
3. ทำงานในช่วงแรงดันไฟฟ้า 2.7 – 6 โวลต์
4. มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์
5. มีอินพุตและเอาต์พุตพอร์ตขนาด 15 บิต
6. มีวงจรนับและวงจรจับเวลาขนาด 16 บิตจำนวน 2 วงจร
7. สามารถอินเทอร์รัปต์ได้จาก 6 แหล่ง

2.5 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C51

โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C51 เป็นไมโครคอนโทรลเลอร์ที่มีโครงสร้างและสถาปัตยกรรมคล้ายกับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตมีหน่วยความจำโปรแกรมแบบแฟลช 4 กิโลไบต์ หน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ขาสัญญาณมีจำนวน 40 ขาแสดงดังรูปที่ 2.6 และมีโครงสร้างพื้นฐานดังรูปที่ 2.7



รูปที่ 2.6 การจัดขาสัญญาณของไมโครคอนโทรลเลอร์ AT89C51



รูปที่ 2.7 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ AT89C51

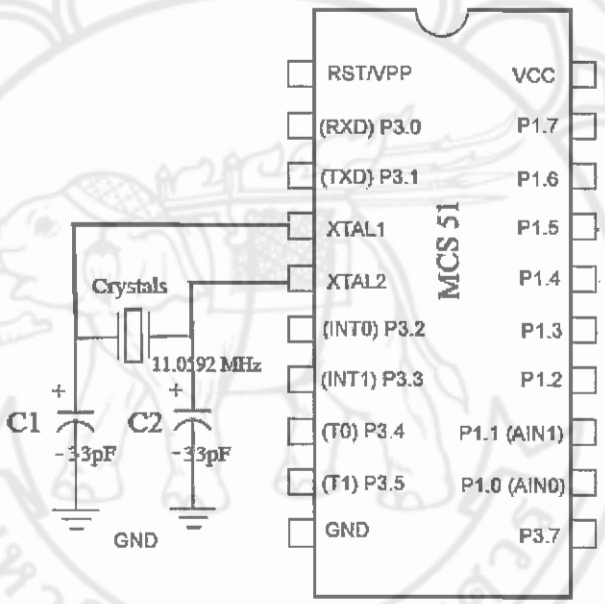
คุณสมบัติของไมโครคอนโทร่ลเลอร์ AT89C51

1. เป็นไมโครคอนโทรลเลอร์ที่ใช้งานร่วมกันได้กับไมโครคอนโทรลเลอร์ในตระกูล MCS51
2. มีหน่วยความจำแบบแฟลชขนาด 4 กิโลไบต์สามารถลบและเขียนได้ถึง 1,000 ครั้ง
3. **ทำงานในช่วงแรงดันไฟฟ้า 4.5 – 5 โวลต์**
4. **สามารถป้องกันการโปรแกรมได้ 3 ระดับ**
5. **มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์**
6. **มีอินพุตและเอาต์พุตพอร์ตขนาด 32 บิต**

7. มีวงจรนับและจับเวลาขนาด 16 บิต 3 วงจร
8. สามารถอินเทอร์รัปต์ได้จาก 8 แหล่ง
9. สามารถโปรแกรมการทำงานได้โดยผ่านพอร์ตอนุกรม

2.6 วงจรกำเนิดสัญญาณนาฬิกา

ไมโครคอนโทรลเลอร์จำทำงานตามสัญญาณนาฬิกาที่ป้อนเข้าระบบเพื่อกำหนดจังหวะในการทำงานให้กับซีพียูและวงจรต่างๆ ภายในตัวไมโครคอนโทรลเลอร์ วงจรกำเนิดสัญญาณนาฬิกา ซึ่งส่วนใหญ่จะใช้คริสตอลเป็นตัวกำเนิดความถี่ และต้องใช้ตัวเก็บประจุ C1 และ C2 ขนาด 20-40 pF แสดงดังรูปที่ 2.8



รูปที่ 2.8 วงจรกำเนิดสัญญาณนาฬิกา

2.7 วงจรรีเซต

การรีเซตการทำงานของไมโครคอนโทรลเลอร์จะต้องป้อนสัญญาณลอจิก 1 เข้าที่ขา Reset นานไม่ต่ำกว่า 2 แมชชีนไซเคิล โดยที่ 1 แมชชีนไซเคิลจะใช้เวลา 12 คาบเวลา ดังนั้นจึงสามารถคำนวณหาค่าเวลาของแมชชีนไซเคิลได้จาก

วิธีทำ

1 คาบเวลา เท่ากับ 1 / ความถี่ของสัญญาณนาฬิกา

กำหนดให้ไมโครคอนโทรลเลอร์ใช้ความถี่ของสัญญาณนาฬิกาเท่ากับ 11.0529 MHz

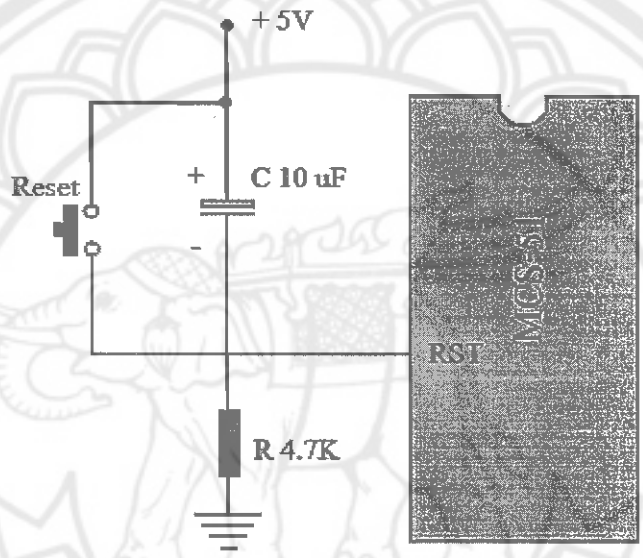
ดังนั้น 1 คาบเวลา = 0.09042 usec

และ 1 แมชชีนไซเคิล = 12 * คาบเวลา

= 1.085 usec

2 แมชชีนไซเคิล = 2.7 uscc

ดังนั้น ในการออกแบบวงจรรีเซตจะต้องให้วงจรค้างสถานะเป็นลอจิก 1 ไม่น้อยกว่า 2.171 usec



รูปที่ 2.9 วงจรรีเซต

2.8 การสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมเป็นการรับหรือส่งข้อมูลในลักษณะกลุ่มของบิต คราวละหนึ่งบิต เรียงลำดับเรื่อยไปจนสิ้นสุด การสื่อสารแบบนี้จะมีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมากเนื่องจากการสื่อสารข้อมูลแบบขนานมีการ โอนย้ายมาพร้อมกัน จึงมีความจำเป็นต้องใช้จำนวนเส้นสัญญาณมากขึ้นตามจำนวนบิตของข้อมูลด้วย ในขณะที่การสื่อสารแบบอนุกรมนั้นต้องการเส้นสัญญาณเพียงสองหรือสามเส้นเท่านั้น ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะในการสื่อสารกับอุปกรณ์ภายนอกเป็นระยะทางไกลๆเพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก

2.8.1 ความเร็วของการสื่อสารข้อมูลอนุกรม เนื่องจากการสื่อสารข้อมูลแบบอนุกรมเป็นการ รับ/ส่ง ข้อมูลในลักษณะกลุ่มของบิตข้อมูล (Bit stream) ดังนั้น จึงต้องให้ความสนใจในการพิจารณาเรื่องอัตราเร็วในการ รับ/ส่งบิตเหล่านี้เป็นอันดับแรก โดยทั่วไปมักจะระบุกันในหน่วยของจำนวนบิตข้อมูลภายในเวลาหนึ่งวินาที เรียกว่า อัตราบอด ตามค่ามาตรฐานเหล่านี้ ได้แก่ 110,150,300,1200,2400,4800,9600,19200 บอด ข้อมูลทั้งแปดบิตนี้หากว่าถูกส่งออกมาด้วยอัตรา 9600 บอด จะใช้เวลาในการส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104 μs และเวลาในการส่งข้อมูลทั้งแปดบิตมีค่าเท่ากับ 8*104 หรือ 832 μs

2.8.2 รูปแบบของการส่งข้อมูลอนุกรม การสื่อสารอนุกรมแบบอะซิงโคนัสจะใช้การแปลงข้อมูลขนานให้เป็นอนุกรมแล้วเพิ่มเติมบิตบางอย่างร่วมไปกับการส่งข้อมูลจริงซึ่งได้แก่

1. บิตเริ่มต้น (Start Bit) มีหน้าที่สำหรับการบ่งบอกให้ทราบถึงตำแหน่งเริ่มต้นก่อนบิตข้อมูล ตามปกติแล้วค่าของบิตเริ่มต้นจะเป็นระดับลอจิกต่ำ

2. บิตแสดงสภาวะความเป็นเลขคู่หรือเลขคี่ (Parity Bit) มีหน้าที่เพื่อการตรวจสอบความถูกต้องของข้อมูลโดยทั่วไปมักเรียกว่า บิตพาริตี้และจะนำไปต่อท้ายบิตของข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของบิตที่เป็น 1 ซึ่งจะเป็นได้สองลักษณะคือ พาริตี้คู่ (Even Parity) หรือพาริตี้คี่ (Odd Parity) ตัวอย่างเช่นระบบที่ติดต่อกันโดยระบุว่าจะใช้พาริตี้คู่ ทางด้านส่งจะนำค่าข้อมูลที่จะส่งมาพิจารณาหาจำนวนของบิตที่มีค่าเป็น 1 หากเป็นเลขจำนวนคู่อยู่แล้ว ค่าของพาริตี้จะมีค่าเป็นศูนย์ แต่หากว่าจำนวนของบิตที่มีค่าเป็น 1 เป็นเลขจำนวนคี่ ค่าของพาริตี้ก็จะมีค่าเป็น 1 การพิจารณาทางด้านรับเป็นการตรวจสอบจำนวนบิตที่มีค่าเป็น 1 ของข้อมูลที่ได้รับมาทั้งหมดรวมทั้งบิตพาริตี้ ถ้ามีค่าเป็นเลขจำนวนคู่ แสดงว่าข้อมูลที่ได้รับเข้ามานี้ถูกต้องแต่หากไม่เป็นเลขจำนวนคู่ แสดงว่าเกิดการผิดพลาดของข้อมูลขึ้น เป็นต้น

3. บิตสิ้นสุด (Stop Bit) เป็นบิตที่เพิ่มขึ้นเพื่อระบุถึงขอบเขตการสิ้นสุดของกลุ่มบิตข้อมูล บิตสิ้นสุดสามารถโปรแกรมได้คือ 1 บิต 1 ½ บิต และ 2 บิต ดังนั้นกรณีของการส่งข้อมูล 8 บิต หากข้อมูลถูกส่งออกไปด้วยอัตราเร็ว 9600 บอด เวลาโดยรวมในการส่งข้อมูลหนึ่งไบต์ จะมีค่าเป็น 12*104 หรือ 1.25 ms

2.9 การใช้งานพอร์ตสื่อสารของไมโครคอนโทรลเลอร์ MCS-51

2.9.1 การใช้งานพอร์ตสื่อสารอนุกรม พอร์ทสื่อสารอนุกรมมีโครงสร้างการทำงานในแบบที่เรียกว่า ฟูลดูเพล็กซ์ (Full Duplex) สามารถรับและส่งข้อมูลอนุกรมได้ในเวลาเดียวกัน

Serial Port Buffer (SBUF) ใช้เป็นบัฟเฟอร์สำหรับรับและส่งข้อมูลอนุกรมโดยมีอยู่ 2 ตัว พอร์ทสื่อสารอนุกรมสามารถโปรแกรมการทำงานได้หลายโหมดด้วยกันโดยเลือกทีละบิต SM1 และ SM0 ซึ่งอยู่ในรีจิสเตอร์ควบคุม SCON การทำงานของทั้ง 4 โหมด ของพอร์ทสื่อสารอนุกรมมีดังตารางที่ 2.2

ตารางที่ 2.2 โหมดการทำงานของพอร์ทสื่อสารอนุกรม

SM0	SM1	โหมด	การทำงาน
0	0	0	Shift Register ความเร็วในการรับหรือส่งข้อมูลเท่ากับ (1/2) ของ CPU Osc
0	1	1	8 Bit UART ความเร็วในการรับหรือส่งข้อมูลกำหนดได้จาก Timer 1,2
1	0	2	9 Bit UART ความเร็วในการรับหรือส่งข้อมูล = (1/32) หรือ (1/64) เท่าของ CPU Osc โดยขึ้นกับบิต SMOD ใน PCON
1	1	3	9 Bit UART ความเร็วในการรับหรือส่งข้อมูลกำหนดที่ Timer 1,2

โหมด 0 : พอร์ทสื่อสารอนุกรม 8 บิต โดยการส่งข้อมูลเลื่อนออกทีละบิตโดยส่งบิต D0 ออกไปก่อน ทางขา RxD เนื่องจากไม่มีการส่งบิตเริ่มต้น แต่จะส่ง Shift Clock ทางขา TxD (ความเร็ว 1/12 เท่าของ CPU Osc)

โหมด 1 : พอร์ทสื่อสารข้อมูลอนุกรม 10 บิต ข้อมูล 8 บิต 1 บิตเริ่มต้น และ 1 บิตสิ้นสุด และสามารถเปลี่ยนแปลงความเร็วในการส่งข้อมูลได้ โดยขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของ ไทเมอร์ 1,2

$$\text{บอดเรทโหมด 1,3} = \frac{2^{\text{SMOD}} \times \text{CPU Osc}}{32 \times 12 \times \left[256 - (\text{TH1})\right]} \quad \text{ไทเมอร์ 1} \qquad (1)$$

$$\text{บอดเรทโหมด1,3} = \frac{\text{CPU Osc}}{32 \times \left[65536 - (\text{RCAP2H,RCAP2L})\right]} \quad \text{ไทเมอร์2} \qquad (2)$$

โหมด 2 : พอร์ทสื่อสารอนุกรม 11 บิต ข้อมูล 9 บิต 1 บิตเริ่มต้น และ 1 บิตสิ้นสุด (TB8 นิยมนำมาใช้ส่งพาริตี้บิต) ความเร็วในการรับส่งข้อมูลเท่ากับ 1/32 หรือ 1/64 เท่าของ CPU Osc โดยขึ้นกับบิต SMOD ใน PCON

$$\text{บอดเรท (โหมด 2)} = \frac{\left(2^{\text{SMOD}}\right)\text{CPU Osc}}{64} \qquad (3)$$

บอดเรท (โหมด 2) = 1/32 CPU Osc เมื่อ SMOD = 1

บอดเรท (โหมด 2) = 1/64 CPU Osc เมื่อ SMOD = 0

โหมด 3 : พอร์ทสื่อสารอนุกรม 11 บิต ข้อมูล 9 บิต 1 บิตเริ่มต้น และ 1 บิตสิ้นสุด เหมือนโหมด 2 ยกเว้นอัตราความเร็วจะขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของ ไทเมอร์ 2

$$\text{บอดเรท โหมด 1,3} = \frac{2^{SMOD} \times \text{CPU Osc}}{32 \times 12 \times \left[256 - (TH1)\right]} \quad \text{ไทเมอร์ 1} \qquad (4)$$

$$\text{บอดเรท โหมด 1,3} = \frac{\text{CPUOsc}}{32 \times \left[65536 - (RCAP2H,RCAP2L)\right]} \quad \text{ไทเมอร์ 2} \qquad (5)$$

2.9.2 กระบวนการรับและส่งข้อมูลอนุกรมของ MCS-51

การส่งข้อมูลออกทางพอร์ทอนุกรมของ MCS-51 จะเริ่มต้นขึ้นภายหลังเมื่อมีการเขียนข้อมูลลงไปใน SBUF ข้อมูลนี้ จะถูกเลื่อนทีละบิต และส่งสัญญาณออกไปภายนอกโดยอัตโนมัติ เมื่อข้อมูลเหล่านี้ได้ส่งออกไปครบถ้วนแล้วจะทำให้ค่าแฟลกซ์ T1 เป็น 1 เพื่อแจ้งให้ทราบว่าขณะนี้ SBUF ว่างและพร้อมที่จะส่งข้อมูลไบต์ต่อไปแล้ว ในกรณีที่ใช้เขียนข้อมูลใหม่ลงในรีจิสเตอร์ SBUF โดยไม่รอให้แฟลกซ์ T1 มีค่าเป็น 1 ก่อน จะมีผลทำให้ข้อมูลที่ส่งไปผิดพลาดได้

สำหรับการรับข้อมูลจากพอร์ทอนุกรมจะต้องเริ่มต้นโดยการกำหนดค่า REN (Receive Enable)ให้มีค่าเป็น 1 ก่อน หลังจากนั้นเมื่อมีข้อมูลภายนอกถูกส่งเข้ามายัง 8051 ทีละบิตจนครบ และเมื่อบิตสุดท้ายเลื่อนเข้ามาเรียบร้อยแล้ว ข้อมูลนั้นจะถูกย้ายมาเก็บไว้ยังรีจิสเตอร์ SBUF และแฟลกซ์ RI ก็จะมีค่าเป็น 1 (ถูกเซต)

1. พอร์ทอนุกรม (โหมด 0)

การทำงานของพอร์ทอนุกรม (โหมด 0) เป็นการรับและส่งข้อมูลอนุกรมจำนวน 8 บิต โดยใช้เพียงขาสัญญาณ RxD เท่านั้น (ขานี้ใช้งาน 2 หน้าที่ใช้ส่งและรับข้อมูล) ส่วนขาสัญญาณ TxD จะนำไปใช้เพื่อเป็นขาสัญญาณนาฬิกาในการให้จังหวะ การเลื่อนข้อมูลกับวงจร เลื่อนบิตภายนอก สำหรับอัตราเร็วจะถูกกำหนดไว้คงที่ที่ค่า 1/12 เท่าของ CPU Osc จากรูปที่ 2.6 แสดงให้เห็นถึงแผนภาพเวลาสัญญาณต่างๆในโหมด 0 เมื่อมีการรับหรือส่งข้อมูล 1 ไบต์ โดยสัญญาณนาฬิกาในการเลื่อนบิตนี้จะเกิดภายในตัว 8051 เอง เนื่องจากโหมดนี้ไม่มีการส่งบิตเริ่มต้นและบิตสิ้นสุด ดังนั้นจึงจำเป็นที่จะต้องส่งสัญญาณ Shift clock ออกไป เพื่อใช้ Synchronize ระหว่างฝ่ายรับและฝ่ายส่ง โดยจะใช้ขา TxD ส่วนการรับข้อมูลจะรับข้อมูลเข้าทางขา RxD และรับ Shift clock เข้าทางขา TxD ถ้า CPU Osc มีค่าเท่ากับ 12 MHz ก็จะส่งได้ถึง 1 ล้านบอด ซึ่งโหมด 0 เป็นโหมดที่ส่งข้อมูลได้เร็วที่สุด รายละเอียดผังเวลาในการรับส่งดังแสดงในรูปที่ 2.10

2. พอร์ทอนุกรม (โหมด 1)

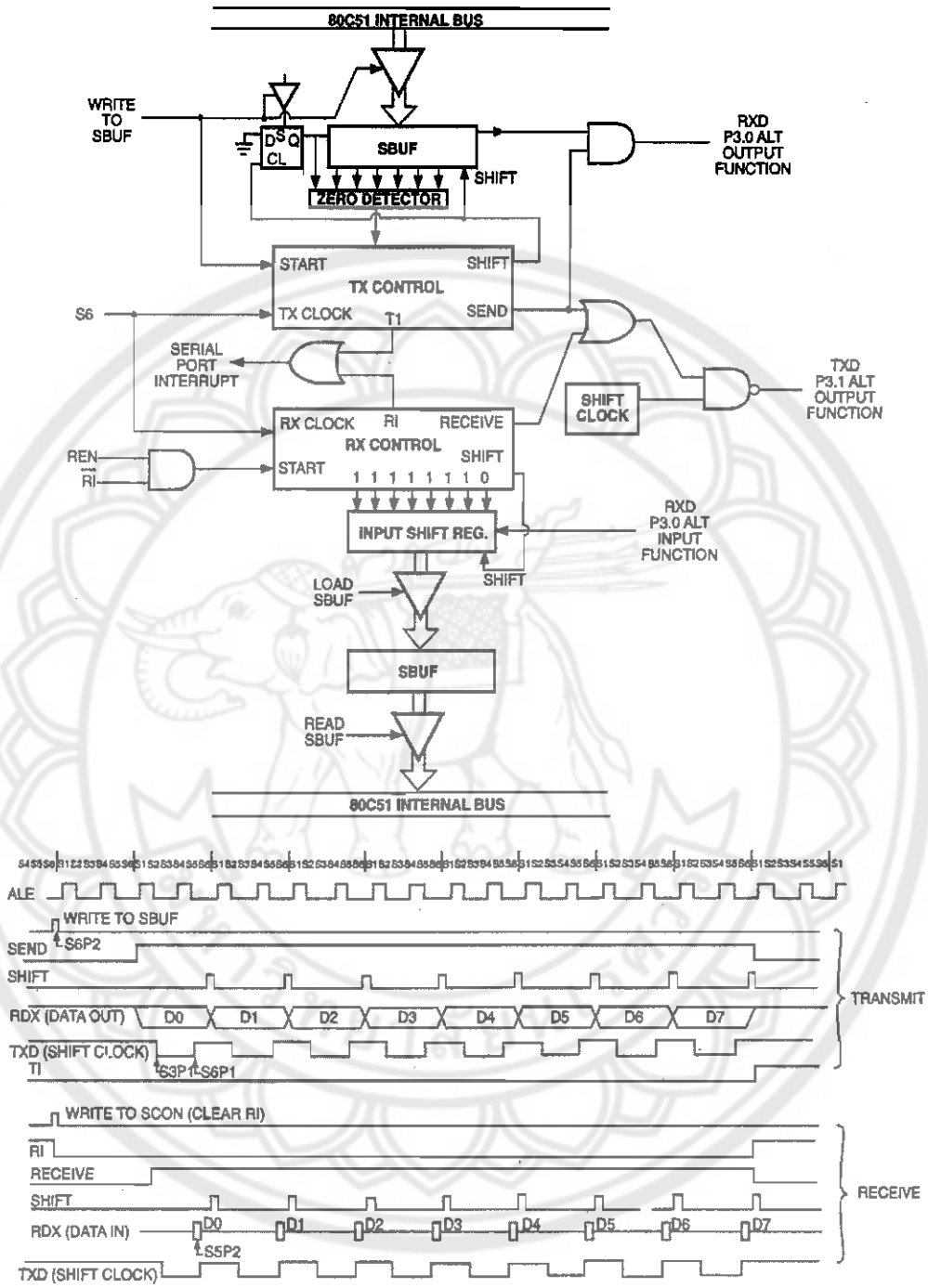
การทำงานในโหมด 1 เป็นการสื่อสารข้อมูลอนุกรมจำนวน 10 บิต ประกอบด้วยบิตเริ่มต้น 1 บิต ข้อมูลจำนวน 8 บิต และบิตสิ้นสุดอีก 1 บิต ดังแสดงในรูปที่ 2.7 โดยข้อมูลจะถูกส่งออกทาง TxD และรับเข้ามาทางขาสัญญาณ RxD ในส่วนของข้อมูล 8 บิต ที่ได้รับหรือทำการส่งออก จะเป็นบิตนัยสำคัญต่ำเป็นลำดับแรก ทางฝ่ายรับค่าของบิตสิ้นสุดจะส่งเข้ามาจัดเก็บไว้ในบิต RB8 ภายในรีจิสเตอร์ SCONสำหรับอัตราเร็วในการส่งข้อมูลของโหมด 1 นั้น สามามารถเลือกได้จากไทเมอร์ 1 ผังเวลาการทำงานแสดงดังในรูปที่ 2.11

3. พอร์ทอนุกรม (โหมด 2)

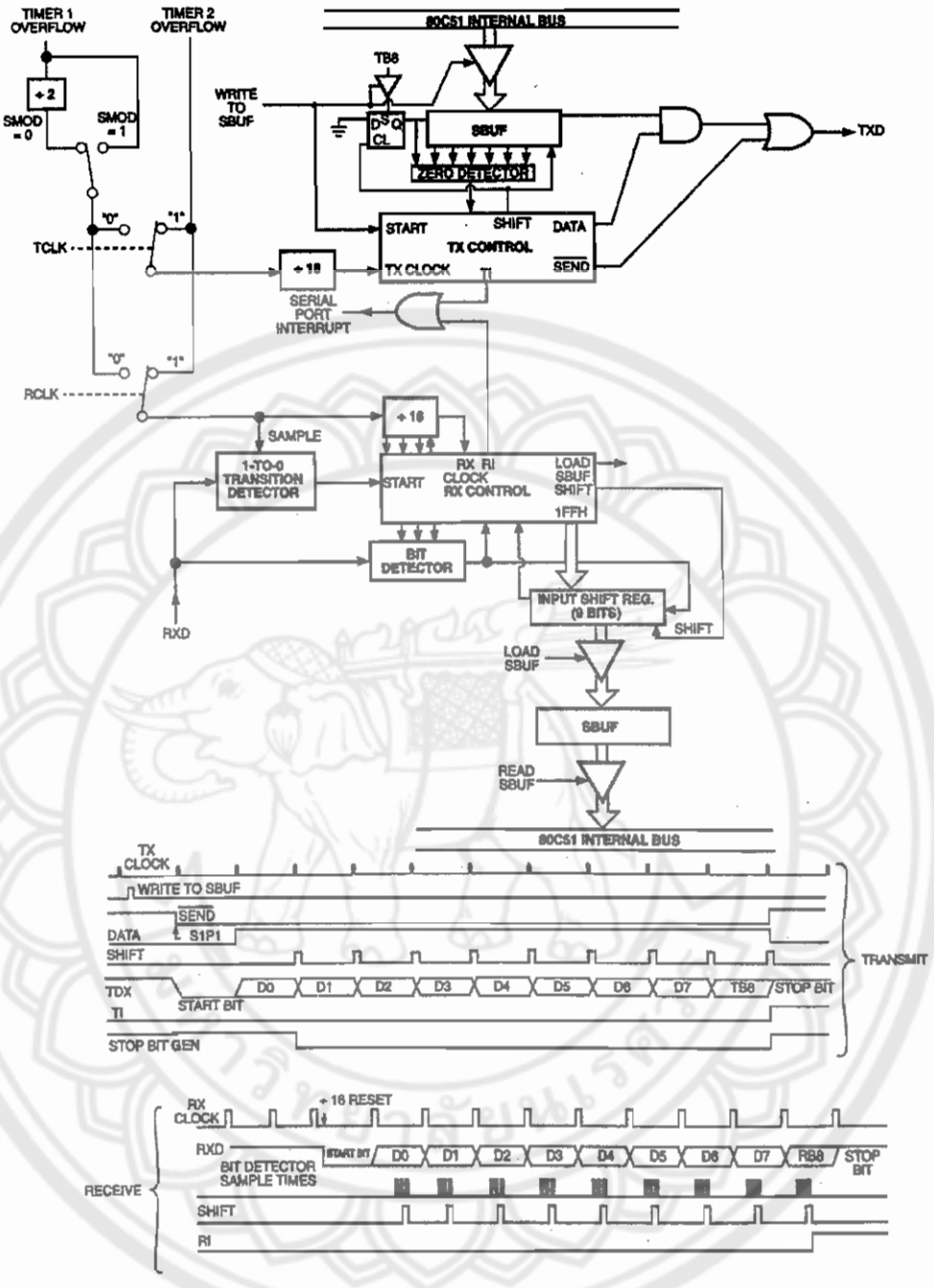
โหมด 2 ใช้ทั้งหมด 11 บิต โดยแบ่งเป็น บิตเริ่มต้น 9 บิตข้อมูล และบิตสิ้นสุด โดยบิตที่ 9 ผู้ใช้สามารถกำหนดค่าเองได้ว่าจะส่งค่าอะไรออกไป โดยจะต้องนำไปใส่ไว้ในบิต TB8 ในรีจิสเตอร์ SCON ส่วนมากผู้ใช้มักจะนำบิตนี้มาใช้เป็นพาริตี้บิต โดยรับค่ามาจากพาริตี้แฟลกซ์ใน PSW ส่วนทางด้านรับ บิตที่ 9 จะถูกนำไปเก็บไว้ใน RB8 อัตราเร็วในการส่ง/รับข้อมูลกับ CPU Osc และค่า SMODซึ่งอยู่ในบิต 7 ใน SCON ผังเวลาการทำงานแสดงดังในรูปที่ 2.12

4. พอร์ทอนุกรม (โหมด 3)

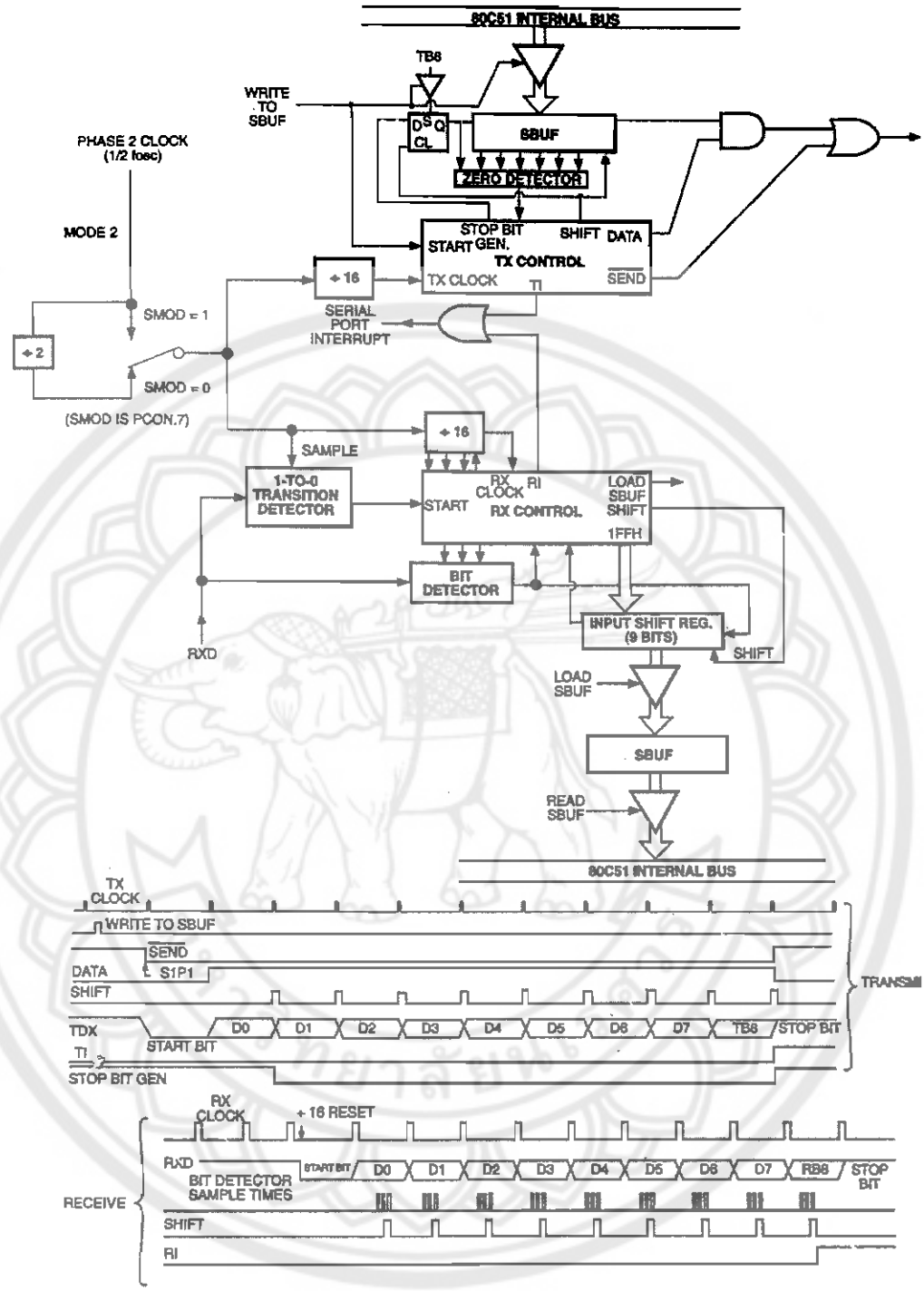
การทำงานเหมือนกับโหมด 2 ทุกอย่าง จะแตกต่างกันที่ความเร็วในการรับส่งข้อมูลจะขึ้นอยู่กับอัตราโอเวอร์โฟลว์ของไทเมอร์ 1 หรือ 2 โดยมีผังการทำงานแสดงดังในรูปที่ 2.13



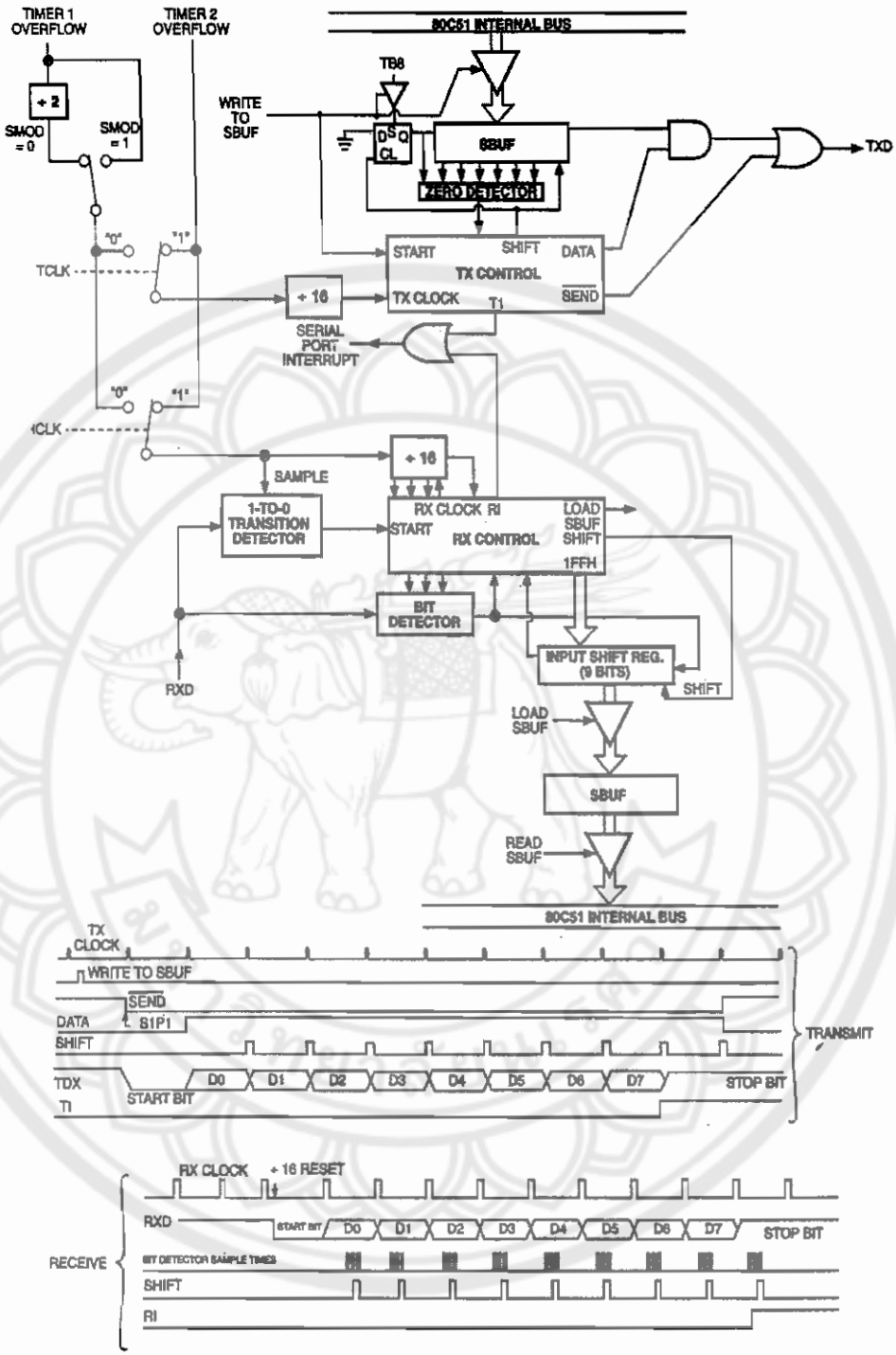
รูปที่ 2.10 ผังการทำงานโหมด 0



รูปที่ 2.11 ผังการทำงานโหมด I



รูปที่ 2.12 ผังการทำงานโหมด 2



รูปที่ 2.13 ผังการทำงานโหมด 3

2.9.3 รีจิสเตอร์ที่ใช้ควบคุมการรับส่งข้อมูลอนุกรม SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0, SM1 : บิตเลือกโหมดการทำงาน

SM2 : บิตเลือกการทำงานแบบ Single Processor Environment หรือ Multiprocessor Environment

1 : เลือก Multiprocessor Environment ใช้ได้กับโหมด 2, 3

0 : เลือก Single Processor Environment ใช้ได้กับทุกโหมด REN (Receive Enable): บิตควบคุมให้รับหรือไม่ให้รับข้อมูล

1 : ให้รับข้อมูล

0 : ห้ามรับข้อมูล

TB8 (Transmit bit D8): ข้อมูลบิตที่ 9 ที่จะส่งออกไปในโหมด 2, 3 ให้ใส่ในบิตนี้

RB8 (Receive bit D8): ข้อมูลบิตที่ 9 ที่รับเข้ามาจะมาเก็บในบิตนี้

TI: แฟลกซ์ TI จะเป็น 1 เมื่อสิ้นสุดการส่งข้อมูล 1 ไบต์

RI: แฟลกซ์ RI จะเป็น 1 เมื่อรับข้อมูลเสร็จ 1 ไบต์

2.9.4 หน่วยความจำและการเชื่อมต่อกับ MCS – 51

การสร้างระบบไมโครคอนโทรเลอร์โดยการใช้ชิป MCS–51 จำเป็นต้องมีการขยายหน่วยความจำภายนอกสำหรับเก็บโปรแกรมและเก็บข้อมูลจากการประมวลผลเพราะว่าชิป MCS 51 บางรุ่นไม่มีหน่วยความจำโปรแกรมภายใน บางเบอร์มีหน่วยความจำภายในขนาดเล็กซึ่งไม่พอต่อการใช้งานหน่วยความจำที่ใช้เก็บโปรแกรมเรียกว่าหน่วยความจำโปรแกรม (Program Memory)

หน่วยความจำโปรแกรม (Program Memory)

เป็นหน่วยความจำที่สามารถเก็บข้อมูลได้แม้ว่าไม่มีไฟเลี้ยง ข้อมูลก็จะยังคงอยู่หน่วยความจำประเภทนี้มักใช้เก็บโปรแกรมระบบที่มีการเปลี่ยนแปลงอีก เช่น BIOS ในคอมพิวเตอร์ โปรแกรมมอนิเตอร์ในบอร์ดควบคุมต่างๆ หน่วยความจำประเภทนี้เรียกรวมๆว่า ROM

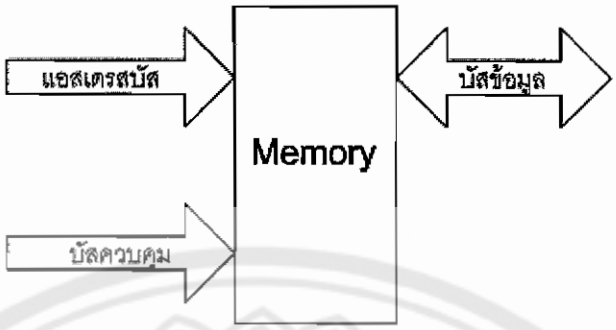
EEPROM (Electrically Erasable PROM)

หน่วยความจำชนิดนี้จะคล้ายกับ EPROM แต่สามารถลบได้โดยใช้กระแสไฟฟ้าโดยไม่ต้องใช้แสง UV

การอ่านและเขียนข้อมูลกับหน่วยความจำ

ไอซีหน่วยความจำโดยทั่วไปจะประกอบด้วยกลุ่มสัญญาณ คือ แอดเดรสบัส, บัสข้อมูล และบัสควบคุม โดยแอดเดรสบัสจะใช้ในการอ้างตำแหน่ง ถ้ามีแอดเดรสบัส 10 เส้น คือ A0 – A9 สามารถอ้างตำแหน่งได้ 2^{10} หรือ 1 กิโลไบต์ บัสข้อมูลจะเป็นตัวบอกว่าในแต่ละตำแหน่งจะเก็บ

ข้อมูลได้อีกกี่บิต เช่น ถ้ามีข้อมูล 8 เส้น คือ D0 – D7 สามารถเก็บข้อมูลได้ตำแหน่งละ 1 ไบต์ ส่วนบัสควบคุมจะใช้ในการอ่านเขียนหน่วยความจำ ระบบบัสทั้ง 3 แสดงได้ดังรูปที่ 2.14



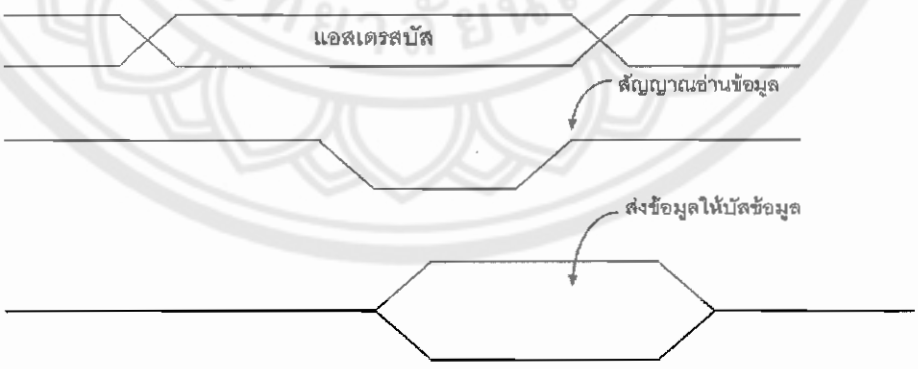
รูปที่ 2.14 แสดงกลุ่มสัญญาณที่ใช้ในการติดต่อกับหน่วยความจำ

โดยทั่วไปไอซีหน่วยความจำจะมีขา CE (Chip Enable) ไว้สำหรับเลือกให้ไอซีตัวที่ต้องการใช้ทำงาน ในกรณีที่มีไอซีหน่วยความจำหลายๆ ตัว ขา OE (Output Enable) จะใช้ต่อกับไอซีถอดรหัสสำหรับการอ่านข้อมูล ถ้าขานี้แอคทีฟ ข้อมูลที่อยู่ในไอซีหน่วยความจำจะส่งออกมาทางบัสข้อมูลได้

การอ่านข้อมูลจากหน่วยความจำ มีลำดับขั้นดังนี้

1. ส่งตำแหน่งที่จะอ่านไปก่อนทางแอดเดรสบัส
2. ส่งสัญญาณควบคุมว่าต้องการจะอ่าน
3. ข้อมูลในไอซีหน่วยความจำถูกส่งออกมาทางบัสข้อมูล

การอ่านข้อมูลจากหน่วยความจำสามารถเขียนเป็นไดอะแกรมเวลาได้ดังรูปที่ 2.15

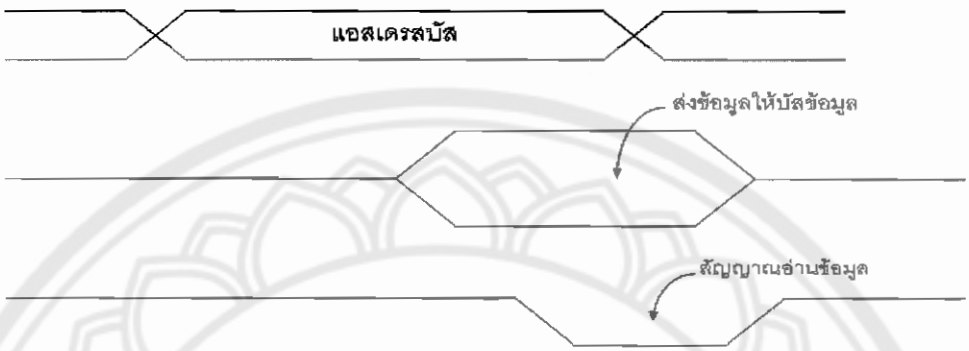


รูปที่ 2.15 แสดงสัญญาณในการอ่านข้อมูล

การเขียนข้อมูลลงหน่วยความจำ มีลำดับขั้นดังนี้

1. ส่งตำแหน่งที่จะเขียนข้อมูลออกไปก่อนทางแอดเดรสบัส
2. ส่งข้อมูลที่จะเขียนไปทางบัสข้อมูล
3. ส่งสัญญาณเขียนข้อมูล

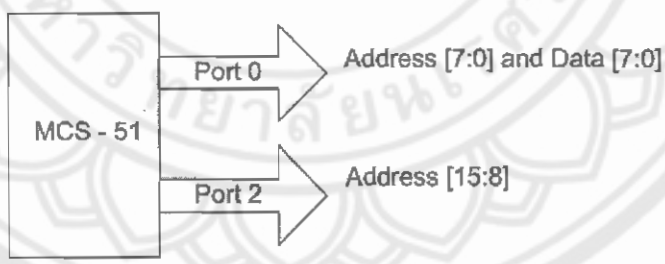
การเขียนข้อมูลให้กับหน่วยความจำสามารถเขียนเป็นไดอะแกรมเวลาได้ดังรูปที่ 2.16



รูปที่ 2.16 แสดงสัญญาณในการเขียนข้อมูล

การเชื่อมต่อหน่วยความจำกับ MCS – 51

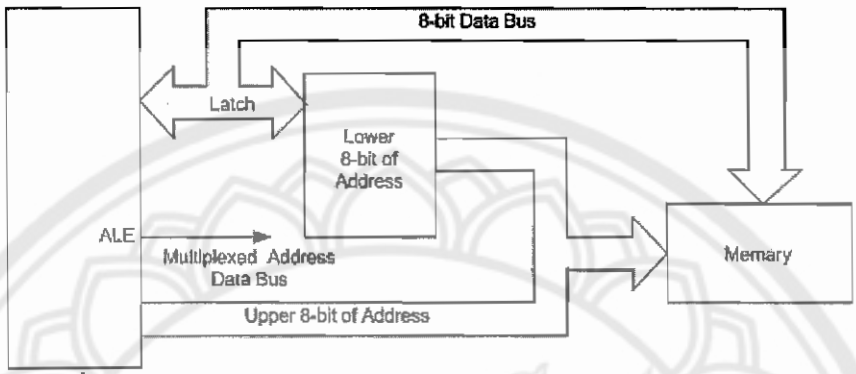
สัญญาณที่ใช้ในการติดต่อกับหน่วยความจำประกอบด้วย แอดเดรสบัส บัสข้อมูล และบัส -ควบคุม ใน MCS – 51 สามารถติดต่อกับหน่วยความจำภายนอกได้ 64 กิโลไบต์ โดยจะใช้สายสัญญาณแอดเดรสบัส 16 เส้น ส่งออกมาทาง พอร์ท 0 และ พอร์ท 2 โดยพอร์ท 0 จะใช้ Multiplexed ระหว่างแอดเดรสบัส และบัสข้อมูล โดยแอดเดรสบัสจะเป็นบิตต่ำ A0–A7 ดังรูป



รูปที่ 2.17 แสดงสัญญาณของ MCS 51 ที่ใช้ติดต่อกับตำแหน่งของหน่วยความจำ

ในการเชื่อมต่อกับหน่วยความจำจะต้องมีอุปกรณ์ภายนอกมา Latch สัญญาณแอดเดรสบัสที่ได้จากพอร์ท 0 เพื่อที่จะใช้พอร์ท 0 เป็นบัสข้อมูลต่อไปนี้ ในรูปที่2.14 จะแสดงการต่อหน่วยความจำประเภท ROM และ RAM กับ MCS – 51 โดยมีอุปกรณ์ภายนอกมา Latch ค่าแอดเดรสไบต์ต่ำเอาไว้ เรียกว่า"Address Latching"

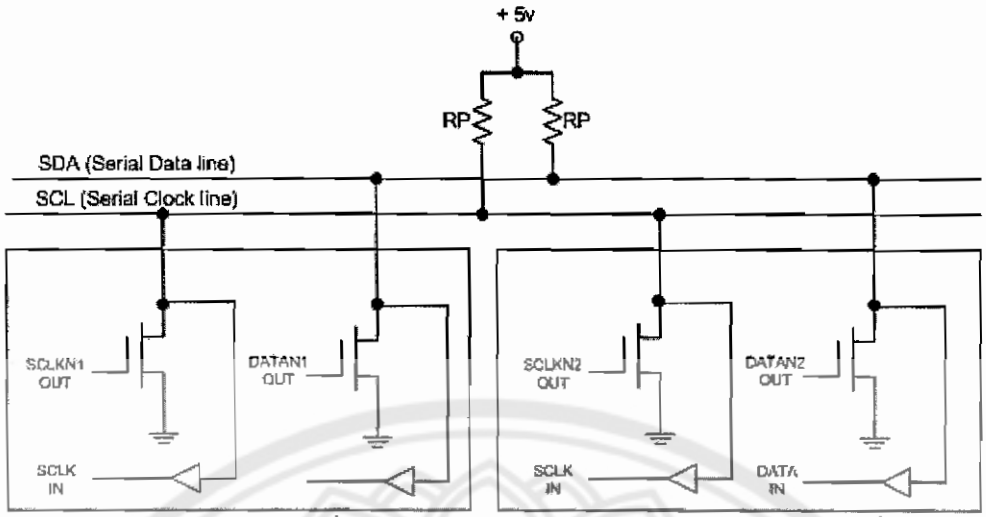
อุปกรณ์ที่นิยมใช้ได้แก่ ไอซี TTL โดยใช้สัญญาณที่ใช้ Latch คือสัญญาณ ALE



รูปที่ 2.18 แสดงการต่อหน่วยความจำประเภท ROM และ RAM กับ MCS 51

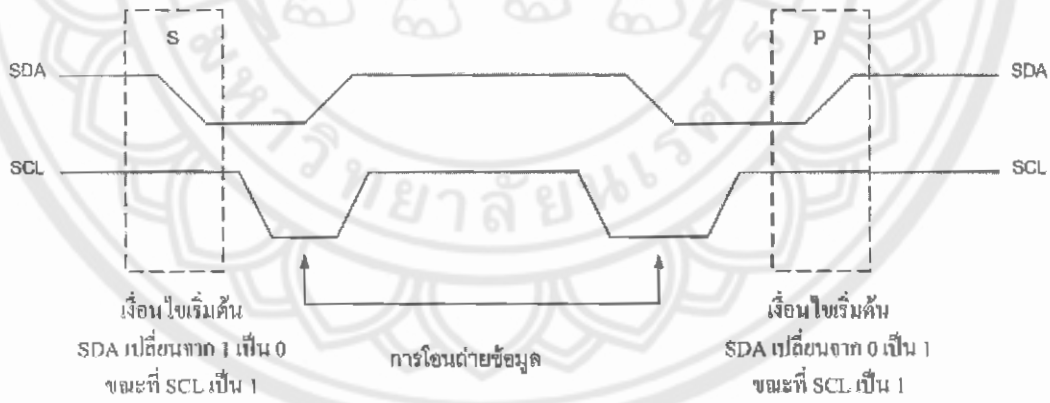
2.9.5 ระบบบัสแบบ I^2C และการเชื่อมต่อ I^2C ย่อมาจาก Inter – IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือต้องการให้ไอซีหรือโมดูลสามารถติดต่อสั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูลอนุกรม หรือ SDA (Serial Data line) อีกเส้นหนึ่งคือ สัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock line) ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานกันไป ส่วนการกำหนดแอดเดรสสำหรับติดต่ออุปกรณ์แต่ละตัวใช้รหัสข้อมูลร่วมกับการกำหนดสภาวะลอจิกที่ขาแอดเดรส

อัตราการส่งข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (Standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (Fast mode) การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่าคือ 7 บิต (7 – bit addressing) หรือ 10 บิต (10 – bit addressing) ในรูปที่2.19 แสดงการเชื่อมต่อมาตรฐานบนระบบบัส I^2C



รูปที่ 2.19 รูปแบบการเชื่อมต่อมาตรฐานของระบบบัส I^2C

การติดต่อสื่อสารกับอุปกรณ์แต่ละตัวจะมีรูปแบบการรับส่งข้อมูลหรือโปรโตคอล (Protocol) ที่อุปกรณ์ทุกตัวรู้จัก ถ้าหากต้องการให้อุปกรณ์ตัวใดเป็นตัวรับข้อมูล ตัวส่งจะส่งแอดเดรสของอุปกรณ์ตัวนั้นไปก่อนถ้าหากอุปกรณ์ตัวใดมีแอดเดรสตรงกันก็จะรับข้อมูลนั้นไป สำหรับตัวอุปกรณ์ที่ต้องการส่งข้อมูลจะเรียกว่ามาสเตอร์ (master) ซึ่งเป็นตัวที่สร้างจังหวะสัญญาณต่างๆ บนระบบบัส ส่วนอุปกรณ์ที่ถูกควบคุมหรือเป็นตัวรับข้อมูลจะเรียกว่า สเลฟ (slave)



รูปที่ 2.20 จังหวะสัญญาณเวลาบนบัส

สภาวะที่เกิดขึ้นบนบัส I^2C

1. บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2. เริ่มต้นส่งข้อมูล (Start data transfer) สภาวะนี้สาย SCL จะเป็นลอจิกสูง แต่สาย SDA จะเปลี่ยนจากลอจิกสูงไปเป็นลอจิกต่ำ เรียกว่าสภาวะเริ่มต้น (Start)

3. สภาวะหยุด (Stop) สภาวะนี้สาย SCL จะเป็นลอจิกสูง แต่สาย SDA จะเปลี่ยนจากลอจิกต่ำไปเป็นลอจิกสูง

4. สภาวะมีข้อมูล (Data valid) สภาวะนี้จะอยู่ระหว่างสภาวะเริ่มต้นและสภาวะหยุด โดยการรับส่งข้อมูลต่างๆ จะเกิดในสภาวะนี้ การรับส่งข้อมูลแต่ละบิตจะใช้สัญญาณนาฬิกาหนึ่งลูก โดยข้อมูลบน SDA จะต้องคงที่ ขณะที่ SCL เป็นลอจิกสูง และบิตข้อมูลใน SDA จะเปลี่ยนแปลงได้ขณะที่ SCL เป็นลอจิกต่ำ ถ้าหากบิตบน SDA มีการเปลี่ยนแปลงขณะที่ SCL เป็นลอจิกสูง ระบบจะตีความว่าเป็นสภาวะเริ่มต้นส่งข้อมูลหรือสภาวะหยุดแทน

5. สภาวะตอบรับ (Acknowledge) เมื่ออุปกรณ์มาสเตอร์ส่งข้อมูลมาครบหนึ่งไบต์แล้ว ในช่วงสัญญาณ SCL ลูกที่ 9 มาสเตอร์จะส่งข้อมูลลอจิกสูงออกมา และถ้าตัวรับได้รับข้อมูลครบแล้วมันจะส่งสัญญาณตอบรับ ACK โดยทำให้ระดับลอจิกสูงบนสัญญาณ SDA ให้กลับเป็นลอจิกต่ำ แต่ถ้าตัวรับได้ รับข้อมูลไม่ถูกต้อง ตัวรับจะบังคับให้ตัวส่งหยุดในสภาวะรอ

การอ้างถึงแบบ 7 บิต (7 bit addressing)

ก่อนเริ่มส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงอุปกรณ์เสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้วก็จะเริ่มต้นการส่งข้อมูลต่อไป รูปแบบของข้อมูลมาตรฐานที่ใช้ในการเข้าถึง

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ ใน 7 บิตรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์ สเลฟ ที่ต้องการติดต่อ โดยแบ่งเป็น

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
X	X	X	X	A2	A1	A0	R/$\overline{W}$
← บิตกำหนดแอดเดรสคงที่ →				← บิตกำหนดแอดเดรสโปรแกรมได้ →			

รูปที่ 2.21 รูปแบบข้อมูลมาตรฐานในการอ้างแอดเดรสของการเชื่อมต่อบนระบบบัส I^2C

1. บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้

2. บิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) จำนวน 3 บิต โดยกำหนดสถานะลอจิกให้แก่ขา A0 – A2 ของอุปกรณ์ที่เชื่อมต่อบัส I^2C

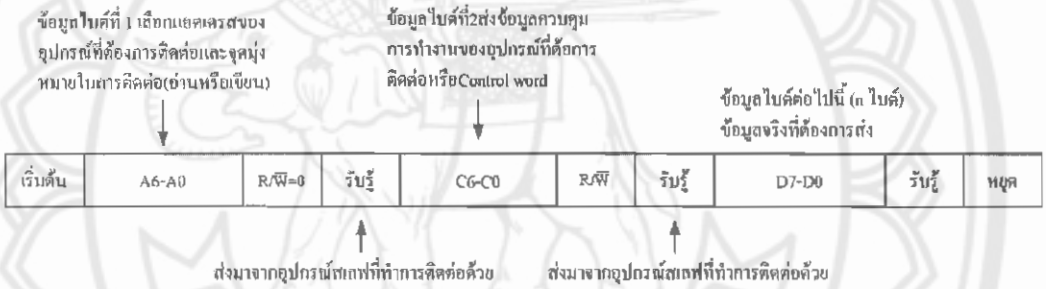
3. บิตกำหนดการอ่านหรือเขียนข้อมูลอุปกรณ์ สเลฟ (เป็นบิต LSB)

เป็น "0" หมายถึง ต้องการเขียนข้อมูลไปยังอุปกรณ์สเลฟ

เป็น "1" หมายถึง ต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (Control Byte) ข้อมูลไบต์นี้จะขึ้นอยู่กับอุปกรณ์แต่ละประเภท เพื่อกำหนดการทำงานต่างๆ ของตัวมัน อุปกรณ์บางประเภทอาจจะไม่ต้องมีการเขียนไบต์นี้ก็ได้

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการส่งจริง (Data) หลังจากที่มีการส่งข้อมูลในแต่ละไบต์ อุปกรณ์ สเลฟ ที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง ในรูปที่ 2.22 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I^2C การอ้างถึงข้อมูล 7 บิต

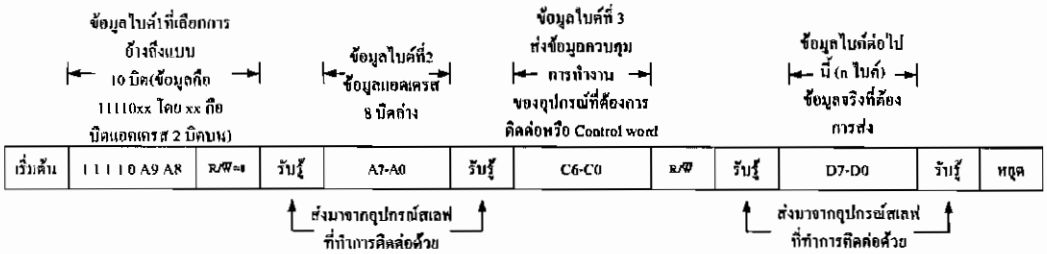


รูปที่ 2.22 รูปแบบของข้อมูลในการอ้างแอดเดรสแบบ 7 บิตบนระบบบัส I^2C

การอ้างถึงแบบ 10 บิต

ในรูปที่ 2.23 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ 10 บิตจะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดได้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการจะติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์ สเลฟ ตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่เราต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกันกับการอ้างถึงแบบ 7 บิต หลังจากส่งข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการส่งข้อมูลสามารถดำเนินต่อไปได้



รูปที่ 2.23 รูปแบบของข้อมูลในการอ้างแอดเดรสแบบ 10 บิตบนระบบบัส I^2C

ฟังก์ชั่นที่ช่วยในการเขียนโปรแกรมภาษา C เพื่อติดต่ออุปกรณ์บนระบบบัส I^2C

1. i2c_delay ทำหน้าที่หน่วงเวลาเพื่อสร้างสภาวะเงื่อนไขของระบบบัส I^2C

2. i2c_clk ทำหน้าที่สร้างสัญญาณนาฬิกาที่ขา SCL เพื่อกำหนดจังหวะในการสื่อสารข้อมูลบน ระบบบัส

3. i2c_start ทำหน้าที่สร้างสภาวะเริ่มต้นในระบบบัส

4. i2c_stop ทำหน้าที่สร้างสภาวะสิ้นสุดในระบบบัส

5. i2c_wrdata ทำหน้าที่ส่งข้อมูล 1 ไบต์ ไปยังอุปกรณ์ สเลฟ ที่ต่อร่วมกับบัส โดยข้อมูลที่ต้องการส่ง มีพารามิเตอร์ dat ทำหน้าที่รับค่า และคืนค่าออกมาเป็น"0" เมื่อการส่งข้อมูลไปยังอุปกรณ์ สเลฟเกิดขึ้นอย่างสมบูรณ์ แต่ถ้าไม่ จะคืนค่า "1" ออกมาแทน

6. i2c_rddata ทำหน้าที่อ่านข้อมูล 1 ไบต์จากอุปกรณ์ สเลฟ ที่ต่อบนบัส โดยจะคืนค่าข้อมูลขนาด 1 ไบต์ ออกมา

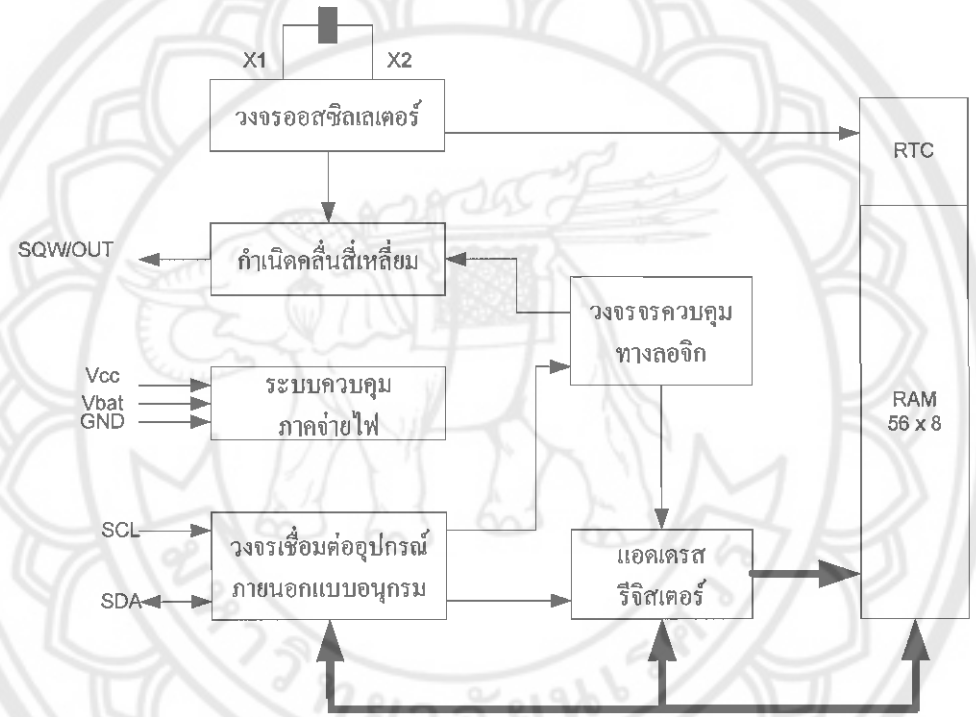
7. i2c_ACK ทำหน้าที่สร้างสภาวะรับรู้ในการรับข้อมูลจากอุปกรณ์ สเลฟ

8. i2c_NACK ทำหน้าที่สร้างสภาวะหยุดรับรู้ในการรับข้อมูลจากอุปกรณ์ สเลฟ

2.9.6 การสร้างฐานเวลา ระบบที่ใช้ไมโครคอนโทรลเลอร์ในการควบคุมบางระบบจะต้องมีเวลาเข้ามาเกี่ยวข้องด้วย โดยจะมีการบอกเวลาเป็นชั่วโมง นาที และวินาที หรือมีการบอกวัน เดือน และปีให้กับระบบด้วย ดังนั้นการสร้างฐานเวลาจะต้องถูกต้องและแม่นยำ แม้ว่าระบบหยุดทำงานและเริ่มทำงานใหม่ค่าเวลาต่างๆจะต้องถูกต้องด้วย ในปัจจุบันการทำงานประเภทนี้จะใช้ชิปไอซีที่ทำหน้าที่สร้างฐานเวลาจริงให้กับระบบ (RTC หรือ Real Time Clock) ไอซีประเภทนี้จะมีวงจรจัดการด้านเวลาจริงอยู่ภายใน การใช้งานเพียงต่อคริสตอลให้กับมัน การทำงานด้านเวลาก็จะ**เป็นไปอย่างอัตโนมัติ เสมือนว่าตัวมันเป็นนาฬิกาและปฏิทินให้กับระบบ ถ้าหากเมื่อใดต้องการทราบเวลาเราสามารถอ่านจากหน่วยความจำภายในของมันได้โดยตรง**

ไอซีฐานเวลาจริง DS1307

ชิปเบอร์ DS1307 ของบริษัท Dallas Semiconductor ซึ่งเป็นชิปที่มีความแม่นยำมาก โดยเป็นไอซีแบบ 8 ขา การเชื่อมต่อกับไมโครคอนโทรลเลอร์จะใช้การอินเตอร์เฟสแบบอนุกรม 2 สาย หรือแบบ I^2C ไอซีเบอร์นี้กินพลังงานต่ำมาก พร้อมทั้งมีปฏิทินเวลาแบบ BCD สามารถใช้ข้อมูลเกี่ยวกับเวลา เช่น วินาที นาที ชั่วโมง (ทั้งแบบ 24 ชั่งโมง / 12 ชั่วโมง พร้อมทั้งระบุค่า AM/PM) และบอกวัน เดือน ปี ได้โดยจะมีการปรับวันที่โดยอัตโนมัติ ในแต่ละเดือนจะแสดงวันได้สูงสุดไม่เกิน 31 วัน และจะปรับวันต่างๆ อย่างถูกต้องเมื่อครบปี การใช้งานกับไมโครคอนโทรเลอร์จะใช้การส่งข้อมูลและแอดเดรสของค่าต่างๆ แบบอนุกรม โดยมีขาหนึ่งเป็นขาสัญญาณ อีกขาหนึ่งเป็นตัวกำหนดสัญญาณนาฬิกา โครงสร้างภายในของไอซี DS1307 แสดงได้ดังรูปที่ 2.24



รูปที่ 2.24 ไดอะแกรมภายในของไอซี DS1307

จากรูปที่ 2.24 ไอซี DS1307 ไอซีแบบ 8 ขา ขาต่างๆ ของ DS1307 เป็นดังต่อไปนี้

ขา SDA (Serial Data Input/Output) เป็นขารับส่งข้อมูลแบบอนุกรม ในการอินเตอร์เฟสจะต้องมีตัวต้านทานต่อพูลอัพภายนอกด้วย

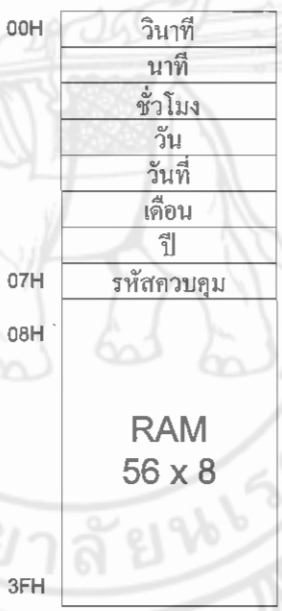
ขา SCL (Serial Clock Input) เป็นขาอินพุตสัญญาณนาฬิกาอนุกรม เพื่อให้เกิดการซิงโครไนซ์ในการรับส่งข้อมูลแบบอนุกรม

ขา SQW/OUT เป็นขาส่งสัญญาณคลื่นสี่เหลี่ยมออกมาทางเอาต์พุต เมื่อเริ่มทำงานบิตนี้จะถูกเซตเป็นลอจิก "1" สัญญาณเอาต์พุตที่ได้รับจะมีอยู่ 4 ค่า คือ 1 Hz , 4 kHz, 8 kHz และ 32 kHz โดยเราสามารถเลือกได้ในการใช้งานจะต้องมีตัวต้านทานต่อพูลอัพภายนอกด้วย ในการใช้งานบางประเภทไม่จำเป็นต้องใช้ขานี้

ขา X1,X2 เป็นขาที่ใช้ต่อกับคริสตอลภายนอก โดยใช้ความถี่ 32.768 kHz วงจรสร้างสัญญาณนาฬิกาภายในออกแบบให้ทำงานรวมกับคริสตอลที่มีตัวประจุ 12.5 พิโกฟารัดต่อร่วมอยู่ด้วย

ขา Vcc และ GND เป็นขาที่ใช้ต่อกับไฟเลี้ยง โดยทั่วไปแล้วจะต่อกับแรงดันไฟ 5 V ถ้าหากแรงดันไฟต่อลงต่ำกว่า 1.25 x V_{BAT} การรักษาเวลาสำรองภายในจะทำงานต่อ โดยจะรับพลังงานจากแหล่งจ่ายไฟสำรอง

ขา V_{BAT} เป็นขาต่อกับแรงดันไฟเลี้ยงสำรอง โดยทั่วไปแล้วจะใช้แหล่งจ่ายไฟแรงดัน 3 V หน่วยความจำ RAM ภายใน และส่วนที่ใช้เป็น RTC มีการจัดแอดเดรสดังรูปที่ 2.28



รูปที่ 2.25 แผนผังหน่วยความจำ

จากรูปที่ 2.25 โดยตำแหน่ง 00H ถึง 07H จะใช้เป็น RTC ส่วนตำแหน่ง 08H ถึง 3FH จะเป็นหน่วยความจำ RAM จากรูปจะเห็นว่าที่ตำแหน่ง 00H จะเก็บเวลาเป็นวินาที ตำแหน่ง 01 จะเก็บเวลาเป็นวินาที โดยข้อมูลที่เก็บจะอยู่ในรูปแบบรหัส BCD สำหรับการอ่านและเขียนค่าเวลากับไอซีตัวนี้ทำได้โดยการเขียนข้อมูลในลักษณะตัวเลข BCD กับรีจิสเตอร์ที่เก็บค่าต่างๆ ตามที่กำหนดเอาไว้ รายละเอียดของแอดเดรสตำแหน่งต่างๆ แสดงในรูป

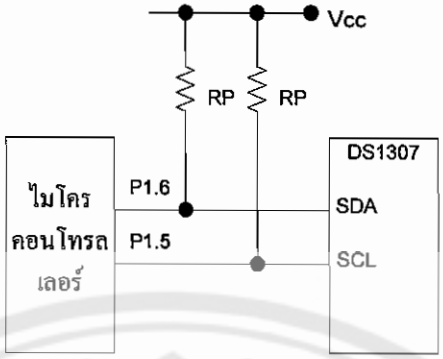
	บิต 7							บิต 0	
00H	CH	วินาที หลักสิบ			วินาที				00-59
01H	X	นาที หลักสิบ			นาที				00-59
02H	X	12.24	A/P	10 HP	ชั่วโมง				01-12,00-23
03H	X	X	X	X	X	วัน			0-7
04H	X	X	วันที่ หลักสิบ		วันที่				01-31
05H	X	X	เดือน หลักสิบ		เดือน				01-12
06H	ปี หลักสิบ				ปี				00-99
07H	OUT	X	X	SQWE	X	X	RS1	RS2	

รูปที่ 2.26 แผนผังรีจิสเตอร์ตำแหน่งต่างๆ ในการจัดการด้านฐานข้อมูลเวลา

จากรูปที่ 2.26 ในรีจิสเตอร์ตำแหน่ง 00H ซึ่งเป็นรีจิสเตอร์วินาที ถ้าบิตที่ 7 ถูกเซตให้เป็นลอจิก “1” วงจรออสซิลเลเตอร์จะหยุดทำงาน ถ้าเป็น “0” วงจรออสซิลเลเตอร์จะทำงานต่อไป สำหรับรีจิสเตอร์ตำแหน่ง 01H เป็นรีจิสเตอร์นาที ซึ่งเวลาหลักสิบของวินาทีจะมีค่าตั้งแต่ 0 ถึง 5 ดังนั้นบิตที่ 7 จะไม่ใช้ สำหรับรีจิสเตอร์ตำแหน่ง 02H ซึ่งเป็นรีจิสเตอร์ชั่วโมง สามารถเซตได้ว่าจะให้เก็บข้อมูลแบบ 12 ชั่วโมง และยังบอกค่า AM/PM ได้อีกด้วย ส่วนรีจิสเตอร์ตำแหน่ง 07H จะเป็นรีจิสเตอร์ควบคุมเพื่อกำหนดว่าจะให้มีเอาต์พุตออกมาทางขา SQW/OUT หรือไม่ และเอาต์พุตที่ออกมาจะให้มีความถี่เท่าใด รายละเอียดต่างๆ สามารถศึกษาได้จากคู่มือของไอซีได้โดยตรง

การจัดหน่วยความจำภายในของชิปไอซีฐานเวลาในลักษณะนี้ทำให้ผู้ใช้สามารถเลือกอ่านหรือเขียนข้อมูลใดๆ ในตำแหน่งที่กำหนดได้ เช่นถ้าหากอ่านค่าในตำแหน่ง 01H ออกมาได้เป็น 0011 0010 หรือ 32 ในระบบเลข BCD จะหมายความว่าเป็นเวลา 32 นาที ถ้าอ่านค่าในตำแหน่ง 02H ออกมาได้เป็น 0010 0010 ในระบบเลข BCD หมายความว่าเป็นเวลา 22 ชั่วโมงเป็นต้น

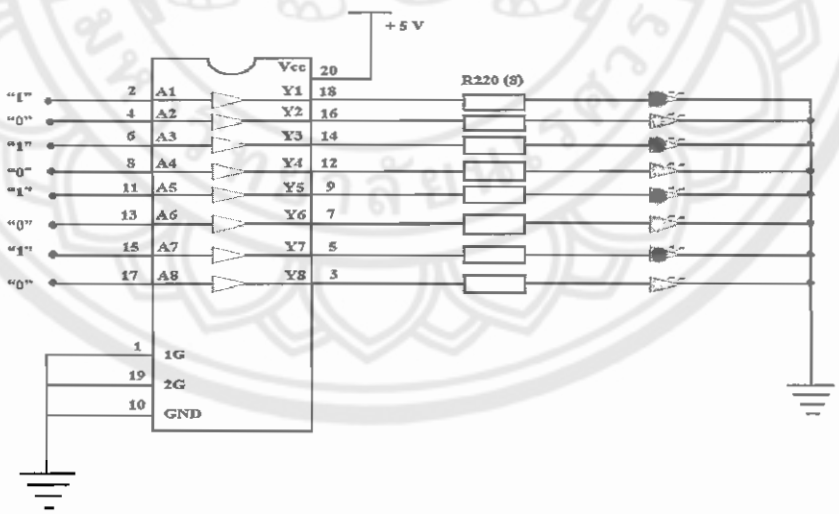
การเชื่อมต่อ DS1307 กับไมโครคอนโทรลเลอร์ MCS 51



รูปที่ 2.27 วงจรการเชื่อมต่อกับ MCS-51 อย่างง่าย

การเชื่อมต่ออย่างง่ายระหว่าง DS1307 กับไมโครคอนโทรลเลอร์แสดงไว้ดังรูปที่ 2.27 โดยใช้P1.6 เป็นขารับส่งข้อมูลแบบอนุกรม ส่วน P1.5 ใช้ขาสัญญาณนาฬิกาควบคุม ตัว DS1307 สามารถรับส่งข้อมูลแบบสองทิศทางได้ และสามารถต่อกับอุปกรณ์อื่นๆ หลายตัวได้โดยมีอุปกรณ์ที่ทำหน้าที่ควบคุมการสื่อสารข้อมูล เรียกว่า มาสเตอร์ สำหรับตัวที่ถูกควบคุมเรียกว่า สเลฟ ในการทดลองนี้จะให้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการอ่านเขียนข้อมูล โดยบิต P1.5 จะเป็นตัวกำหนดสัญญาณนาฬิกาเพื่อควบคุมข้อมูลที่จะอ่านเขียนทาง P1.6 ที่เป็นขาข้อมูล

วงจรขยายสัญญาณ



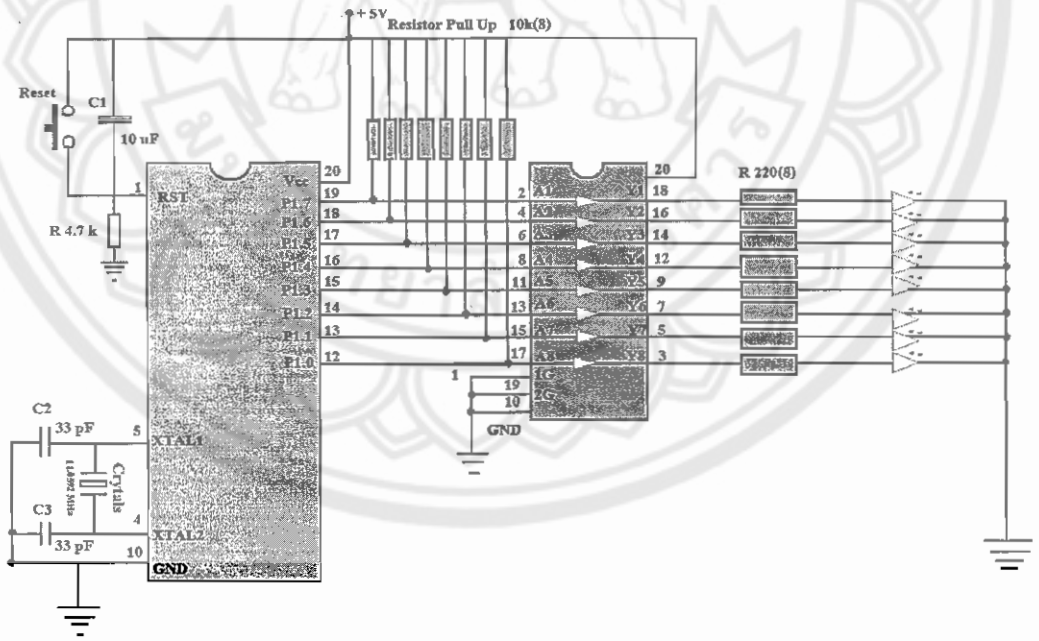
รูปที่ 2.28 วงจรขยายกระแสโดยใช้ไอซีบัฟเฟอร์

ในการเชื่อมต่อไมโครคอนโทรลเลอร์กับอุปกรณ์แสดงผลหลายๆ ตัวนั้นโดยตรงอาจจะทำให้ไมโครคอนโทรลเลอร์ไม่สามารถจ่ายกระแสได้เพียงพอ ดังนั้น ในการเชื่อมต่อกับอุปกรณ์แสดงผลจึงต้องผ่านวงจรขยายกระแสก่อน บัฟเฟอร์ (Buffer) เป็นวงจรขยายกระแสเพื่อจ่ายให้กับอุปกรณ์แสดงผลและป้องกันระบบไมโครคอนโทรลเลอร์ไม่ให้เสียหายหากอุปกรณ์เอาต์พุตเกิดการลัดวงจร บัฟเฟอร์อยู่ในรูปของวงจรรวมหรือไอซี จากรูปที่ 2.28 จะใช้ไอซีบัฟเฟอร์เบอร์ 74244 เป็นตัวขยายกระแส

จากรูปที่ 2.28 เมื่อป้อนสัญญาณไฟลอจิก 1 ที่อินพุต A1 A3 A5 และ Y7 จะทำให้เอาต์พุต Y1 Y3 Y5 และ Y7 มีสถานะเป็นลอจิก 1 หรือแรงดันไฟ 5 โวลต์ เหมือนกับอินพุตจึงทำให้หลอกแสดงผล LED ติด และเมื่อป้อนลอจิก 0 ที่อินพุต A2 A4 A6 และ A8 จะทำให้เอาต์พุต Y2 Y4 Y6 และ Y8 มีสถานะลอจิก 0 เหมือนกับอินพุตจึงทำให้หลอดแสดงผล LED ดับ

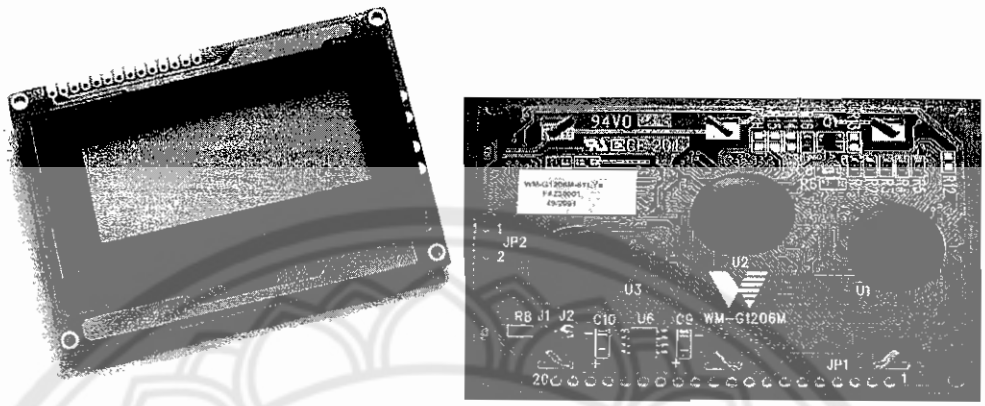
การเชื่อมต่อไมโครคอนโทรลเลอร์กับหลอดแสดงผล LED

การเชื่อมต่อไมโครคอนโทรลเลอร์กับหลอดแสดงผล LED จะใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C2051 ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาดเล็กมี 20 ขาต่อออกพอร์ต P1 ที่ขา 11 ถึงขา 19 ผ่านไอซีบัฟเฟอร์เบอร์ 7422 เพื่อขยายกระแสไปขับหลอดแสดงผล LED ขนาด 8 บิต และจำมีตัวต้านทาน 10k โอห์มเป็นตัวต้านทานพูลอัพต่อระหว่างพอร์ต P1 กับไอซีบัฟเฟอร์โดยการทำงานของหลอดแสดงผลจะขึ้นอยู่กับการเขียนโปรแกรมของไมโครคอนโทรลเลอร์



รูปที่ 2.29 การเชื่อมต่อไมโครคอนโทรลเลอร์ AT89C2051 กับหลอดแสดงผล LED

2.10 โครงสร้าง แอลซีดี โมดูล (LCD Module)



รูปที่ 2.30 แอลซีดีโมดูล (LCD Module)

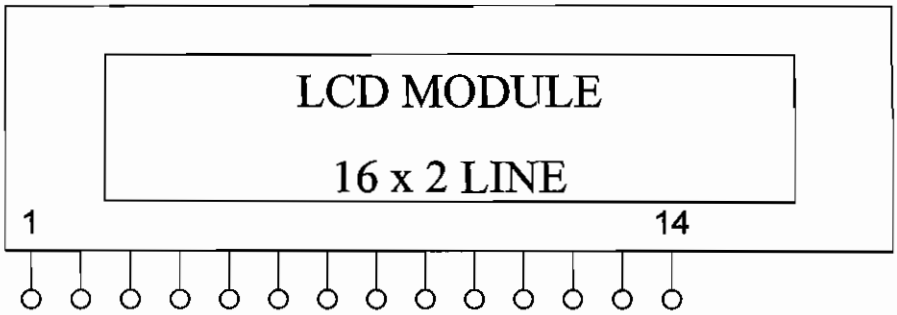
2.10.1 ส่วนประกอบหลักของแอลซีดี โมดูล

ใน แอลซีดี โมดูล จะมีส่วนประกอบหลักๆ 3 ส่วน ดังนี้

ตัวแสดงผล (Display) ภายในผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอกดังนั้นจึงมีมุมมองข้อมูลที่แสดงผลบนจอ

ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของแอลซีดีโมดูล เช่น ลบจอภาพ แสดงตัวอักษรหรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุมแอลซีดี แบบอักขระ ส่วน HD44780 ใช้ควบคุมแอลซีดี แบบกราฟฟิก

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตาม ที่กำหนด ชิปที่ใช้ทำหน้าที่นี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น แอลซีดี โมดูล มีอยู่หลายรุ่น และคุณสมบัติแตกต่างกันไป ซึ่งแบ่งได้เป็น 2 แบบ คือ แบบDot matrix และ Graphic โดยแบบ Dot matrix จะแสดงผลเป็นแบบ 5x8 Dot หรือ 5x10 Dot มีตั้งแต่ 1 Line, 2 Line และ 4 Line ซึ่งการใช้งานแต่ละแบบจะใกล้เคียงกัน ลักษณะขาสัญญาณของ แอลซีดี โมดูล แบบ 1 Line ดังรูปที่ 2.31



รูปที่ 2.31 โครงสร้างแอลซีดีโมดูล

ตารางที่ 2.3 หน้าที่ของขาใช้งาน แอลซีดี โมดูล

ลำดับขา	หน้าที่
ขา 1 (GND)	เป็นกราวด์ ใช้ต่อกับระบบ กราวด์ ของไมโครคอนโทรลเลอร์
ขา 2 (VCC)	เป็นไฟเลี้ยงวงจรของ แอลซีดี มีขนาด +5 VDC
ขา 3 (Vee)	เป็นขาสำหรับปรับความเข้มของจอ แอลซีดี โดยที่เมื่อต่อกับ VCC จะมีความเข้มต่ำสุด และเมื่อต่อกับ กราวด์ จะมีความเข้มมากสุด โดยปกติจะต่อกับ กราวด์ เสมอเพื่อความสะดวกในการต่อ
ขา14 (RS)	Register Select ใช้สำหรับบอก แอลซีดี ให้ทราบว่าข้อมูลที่ส่งให้นั้นเป็น Instructionหรือ Data โดยเมื่อขานี้ เป็น “0” หมายถึง Instruction เป็น “1” หมายถึง Data
ขา 5 (R/$\overline{W}$)	ใช้สำหรับกำหนดว่าเป็นการอ่านหรือเขียนข้อมูลให้กับ แอลซีดี โดยเมื่อขานี้ เป็น “0” หมายถึงเป็นการเขียนข้อมูล เป็น “1” หมายถึงเป็นการอ่านข้อมูล
ขา 6 (E)	เป็นขา Enable เมื่อขานี้ เป็น “1” ใช้สำหรับบอก แอลซีดี ว่าอุปกรณ์ภายนอกต้องการติดต่อด้วย เป็น “0” ตัว แอลซีดี จะไม่สนใจในสัญญาณ RS, R/$\overline{W}$ และ ($DB_7 - DB_0$)
ขา 7 – 14 ($DB_7 - DB_0$)	เป็นขา Data Bus สำหรับอ่านหรือเขียนข้อมูลให้กับตัว แอลซีดี

2.10.2 การเชื่อมต่อ แอลซีดี โมดูล เข้ากับไมโครคอนโทรลเลอร์ สามารถต่อได้โดยตรงกับตัว MCS-51 หรือต่อผ่าน 8255 ก็ได้ ในที่นี้จะต่อโดยตรงกับตัว MCS-51 ดังนี้

- ขาสัญญาณข้อมูล D0-D7 (ขา 7-14) ต่อเข้ากับ MCS-51 พอร์ท 2
- ขา RS (ขา 14) ต่อเข้ากับ MCS-51 พอร์ท 0 บิต 0
- ขา R/$\overline{W}$ (ขา 5) ต่อเข้ากับ MCS-51 พอร์ท 0 บิต 1
- ขา E (ขา 6) ต่อเข้ากับ MCS-51 พอร์ท 0 บิต 2

2.10.3 ชุดคำสั่งของ แอลซีดี โมดูล

1. คำสั่งเคลียร์ตัวแสดงผล (CLEAR DISPLAY)

ตารางที่ 2.4 คำสั่งเคลียร์ตัวแสดงผล

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

คำสั่ง CLEAR DISPLAY เป็นคำสั่งที่ใช้เขียนข้อมูลหรือตัวอักษรว่าง (Space) ลงใน DDRAM ทั้งหมด และทำการกำหนดค่า DDRAM ADDRESS เป็น 0 และเคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายสุดของจอแสดงผล

2. คำสั่ง CURSOR AT HOME

ตารางที่ 2.5 คำสั่ง CURSOR AT HOME

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	*

คำสั่ง CURSOR AT HOME หรือ RETURN HOME เป็นคำสั่งที่ใช้ในการเลื่อนตำแหน่งของเคอร์เซอร์ไปอยู่ที่ตำแหน่งซ้ายสุดของจอแสดงผล โดยข้อมูลที่อยู่ใน DDRAM หรือที่หน้าจอแสดงผลจะไม่เปลี่ยนแปลง

3. คำสั่งโหมดในการป้อนข้อมูล (ENTRY MODE SET)

ตารางที่ 2.6 คำสั่งโหมดในการป้อนข้อมูล

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

คำสั่งในการป้อนข้อมูล (ENTRY MODE SET) ใช้สำหรับกำหนดการเลื่อนของเคอร์เซอร์และตำแหน่งแอดเดรสของ DDRAM ดังนี้

I/D เป็นบิตที่ใช้ในการกำหนดการเลื่อนเคอร์เซอร์และตำแหน่งแอดเดรสของ DDRAM ว่าจะให้เพิ่มหรือลดลงเมื่อเขียนหรืออ่านข้อมูล

บิต I/D = 0 แอดเดรสของ DDRAM จะลดลง

บิต I/D = 1 แอดเดรสของ DDRAM จะเพิ่มขึ้นส่วนเคอร์เซอร์จะเลื่อนตามตำแหน่ง แอดเดรสของ DDRAM

S เป็นบิตที่ใช้กำหนดลักษณะของการแสดงผลเมื่อมีการเขียนข้อมูล

บิต S = 1 เมื่อเขียนข้อมูลใหม่ลงไปแล้วตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูล เดิมจะถูกผลักไปทางซ้าย

บิต S = 0 เมื่อเขียนข้อมูลใหม่ลงไปแล้วตัวเคอร์เซอร์จะเลื่อนไปทางขวา

4. คำสั่งควบคุมการแสดงผล (DISPLAY ON/OFF)

ตารางที่ 2.7 คำสั่งควบคุมการแสดงผล

RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

คำสั่งควบคุมการแสดงผล เป็นคำสั่งที่ใช้ในการเปิดปิดจอแสดงผลและเคอร์เซอร์มีลักษณะดังนี้

D = 0 กำหนดให้ปิดจอแสดงผล (Display OFF)

D = 1 กำหนดให้เปิดจอแสดงผล (Display ON)

C = 0 กำหนดให้ปิดเคอร์เซอร์ (Cursor OFF)

C = 1 กำหนดให้เปิดเคอร์เซอร์ (Cursor ON)

B = 0 กำหนดให้ไม่มีการกระพริบที่เคอร์เซอร์

B = 1 กำหนดให้มีการกระพริบที่เคอร์เซอร์ (กระพริบเป็นรูปสี่เหลี่ยมทึบ)

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และตัวอักษร (DISPLAY SHIFT)

ตารางที่ 2.8 คำสั่งควบการเลื่อนเคอร์เซอร์และตัวอักษร

RS	$R/\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	*	*

คำสั่งควบคุมการเลื่อนเคอร์เซอร์และตัวอักษร เป็นการควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล โดยขึ้นอยู่กับการกำหนดบิต S/C และ R/L โดยมีลักษณะดังนี้

ตารางที่ 2.9 การกำหนดบิต S/C และ R/L

S/C	R/L	ลักษณะการเลื่อน
0	0	เลื่อนเคอร์เซอร์ไปทางซ้าย
0	1	เลื่อนเคอร์เซอร์ไปทางขวา
1	0	เลื่อนตัวอักษรตัวใหม่ไปทางซ้าย
1	1	เลื่อนตัวอักษรตัวใหม่ไปทางขวา

6. คำสั่งการกำหนดฟังก์ชั่นการทำงาน (FUNCTION SET)

ตารางที่ 2.10 คำสั่งการกำหนดฟังก์ชั่นการทำงาน

RS	$R/\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	*	*

DL = 0 กำหนดให้ติดต่อกับ แอลซีดี โมดูลเป็นแบบ 4 บิต

DL = 1 กำหนดติดต่อกับ แอลซีดี โมดูลเป็นแบบ 8 บิต

N = 0 กำหนดการแสดงผลแบบ 1 บรรทัด

N = 1 กำหนดการแสดงผลตั้งแต่ 2 บรรทัดขึ้นไป

F = 0 กำหนดความละเอียดของการแสดงผลเป็น 5 x 7 Dot

F = 1 กำหนดความละเอียดของการแสดงผลเป็น 5 x 10 Dot

2.10.4 ฟังก์ชั่นการเขียนโปรแกรมภาษา C เพื่อติดต่อกับ แอลซีดี โมดูล

lcd_init

เป็นฟังก์ชั่นอินิเชียลแอลซีดี โมดูลกำหนดการติดต่อแอลซีดี โมดูลแบบ 4 บิต แสดงผล 2 บรรทัดที่ความละเอียด 5 x 7 จุด ควรเรียกใช้ฟังก์ชั่นนี้ในตอนต้นของโปรแกรมหลักก่อนใช้งานฟังก์ชั่นใดๆ

รูปแบบ Void lcd_init ( )

การคืนค่า ไม่มีการคืนค่า

lcd_delay และ delay_ms

เป็นฟังก์ชั่นหน่วงเวลาสำหรับสร้างสัญญาณพัลส์เอ็นเอเบิลป้อนให้กับขา E ของแอลซีดี โมดูล

รูปแบบ Void lcd_delay (unsigned int tict)

พารามิเตอร์ Tick ใช้กำหนดค่าหน่วงเวลาในหน่วยมิลลิวินาที กำหนดค่าได้ตั้งแต่ 0 ถึง 65,535

การคืนค่า ไม่มีการคืนค่า

lcd_command

เป็นฟังก์ชั่นส่งคำสั่งควบคุมการแสดงผลที่แอลซีดีโมดูล

รูปแบบ Void lcd_command (unsingned char com)

พารามิเตอร์ Com สำหรับกำหนดรหัสคำสั่งควบคุมการแสดงผลที่แอลซีดี โมดูล

การคืนค่า ไม่มีการคืนค่า

lcd_text

เป็นฟังก์ชั่นส่งข้อมูลอักขระแสดงผลที่แอลซีดีโมดูล

รูปแบบ Void lcd_text (unsigned char text)

พารามิเตอร์ Text สำหรับกำหนดข้อมูลอักขระที่ต้องการแสดงผลที่แอลซีดี

การคืนค่า ไม่มีการคืนค่า

lcd_puts

เป็นฟังก์ชั่นทำหน้าที่ส่งสายอักขระแสดงผลที่แอลซีดี โมดูล

รูปแบบ Void lcd_puts (unsigned char line , char *p)

พารามิเตอร์ Line ใช้กำหนดแอดเดรสแรกของสายอักขระที่จะเริ่มการแสดงผลที่แอลซีดี

P ใช้กำหนดการเข้าถึงแอดเดรสของสายอักขระที่จะส่งไปแสดงยังแอลซีดี

การคืนค่า ไม่มีการคืนค่า

inttolcd

เป็นฟังก์ชั่นทำหน้าที่แสดงค่าข้อมูลตัวเลขจำนวนเต็มที่แอลซีดี โมดูล

รูปแบบ Void inttolcd (unsigned char addr , unsigned int value , usigned char base)

พารามิเตอร์

addr สำหรับกำหนดแอดเดรสแรกของสายอักขระที่จะเริ่มการแสดงผลที่ แอลซีดี

value ใช้กำหนดจำนวนเต็มที่ต้องการนำไปแสดงที่แอลซีดี โมดูล โดยค่าที่แสดงทำได้เฉพาะค่าบวก ตั้งแต่ 0 ถึง +65,535

base ใช้กำหนดเลขฐานที่ต้องการแสดงที่แอลซีดี โมดูล ควรกำหนดเป็น 10 (สำหรับแสดงในแบบเลขฐานสิบ) หรือ 16 (สำหรับแสดงในแบบเลขฐานสิบหก)

การคืนค่า ไม่มีการคืนค่า

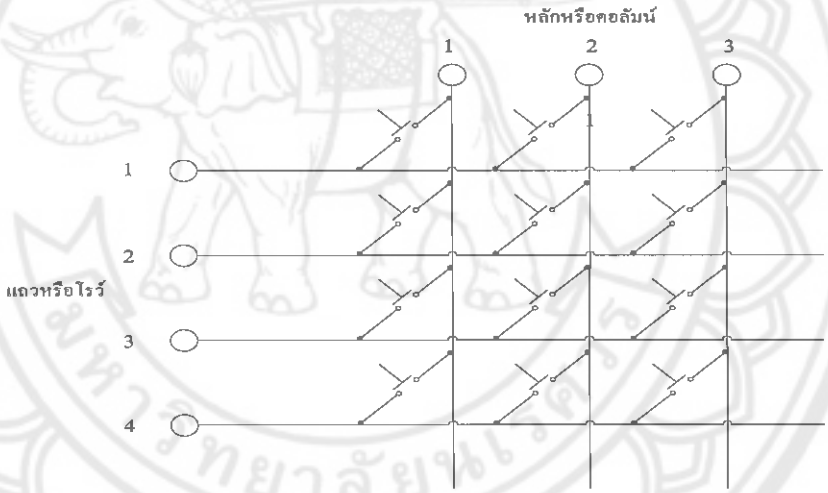
2.11 การใช้งานคีย์แพด (Keypad)

วิธีการอ่านค่าหรือรับค่าจากสวิตช์นั้นมีด้วยกัน 2 ลักษณะใหญ่ๆคือ แบบต่อเข้ากับไฟเลี้ยงหรือกราวด์โดยตรง และต่อแบบวงจรแมทริกซ์ (Matrix Switch) ในรูปที่ 2.32 จะเห็นว่าสวิตช์จะถูกต่อในแนวแกนตั้งและแนวแกนนอน จะเรียกแนวแกนตั้งว่าหลักหรือคอลัมน์ (Column) และเรียกแนวแกนนอนว่าแถวหรือโรว์ (Row) ดังนั้นค่าของสวิตช์ในแต่ละตำแหน่งจะต้องประกอบด้วยหลักและแถว

รูปที่ 2.32 Keypad ที่ใช้ภายในโครงงาน

กระบวนการในการอ่านค่าของสวิตช์แบบนี้นั้นจะใช้วิธีการเขียนโปรแกรมในไมโครคอนโทรลเลอร์ โดยจะต้องใช้สายทั้งหมด 7 เส้น ซึ่งเป็นสายของหลัก 3 เส้นและเป็นสายของแถว 4 เส้น สำหรับสวิตช์แบบ 4 x 3 ต่อเข้ากับไมโครคอนโทรเลอร์จะทำการส่งข้อมูล “0” ไปยังพอร์ทที่ต่อกับด้านแถวทีละเส้นตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายด้านแถวของคีย์แพดไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่ด้านหลักเข้ามาด้วย หากไม่มีการกดค่าด้านหลักก็จะเป็น “1” ทั้งหมด ถ้าหากมีการกดคีย์ค่าของด้านหลักก็จะไม่เป็น “111” อีกต่อไป เป็นการแจ้งให้ทราบว่ามีการกดคีย์แพดเกิดขึ้นแล้ว จากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งต่อไป โดยการค้นหาตำแหน่งนั้นสิ่งที่ได้มาอย่างแรกคือค่าตำแหน่งของคีย์นั้น จากนั้นก็จะนำตำแหน่งนั้นไปเปิดตารางข้อมูล เพื่อที่จะได้ค่าที่ต้องการนำไปแสดงผลที่แท้จริง

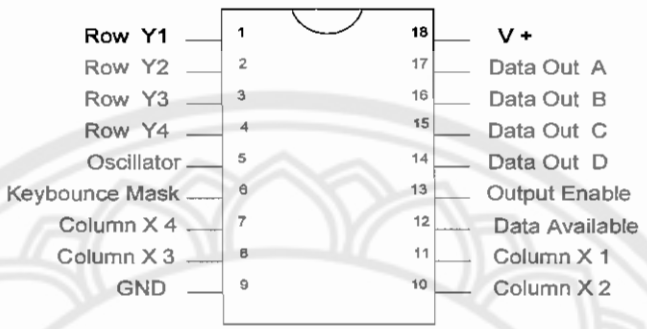
ข้อดีของสวิตช์แบบนี้ก็คือสามารถรองรับการเพิ่มของสวิตช์ได้อย่างสะดวก เพียงแค่แก้ไขซอฟต์แวร์ เพียงเล็กน้อยเท่านั้น ทำให้วงจรสวิตช์แบบแมทริกซ์นี้เป็นที่นิยมกันมากในระบบควบคุมอัตโนมัติหรือกึ่งอัตโนมัติ โดยในการใช้งานทั่วไปจะเรียกสวิตช์แบบแมทริกซ์นี้ว่า “คีย์แพด”ดังแสดงในรูปที่ 2.33



รูปที่ 2.33 วงจรสวิตช์แบบแมทริกซ์หรือคีย์แพด

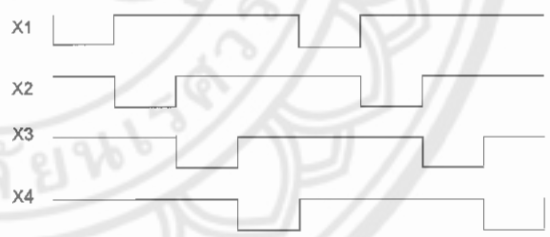
2.11.1 การใช้ไอซีสแกนคีย์สวิตช์

ในปัจจุบันได้มีการออกแบบไอซีสแกนคีย์สวิตช์แบบแมทริกซ์ โดยเอาต์พุตที่ออกจากไอซีจะเป็นเลขไบนารี จากนั้นนำไปต่อกับพอร์ทของระบบไมโครคอนโทรลเลอร์ทำให้ระบบไมโครคอนโทรลเลอร์ไม่ต้องเสียเวลาในการทำโปรแกรมสแกนคีย์สวิตช์ ในที่นี้ใช้ไอซีเบอร์ 74C922



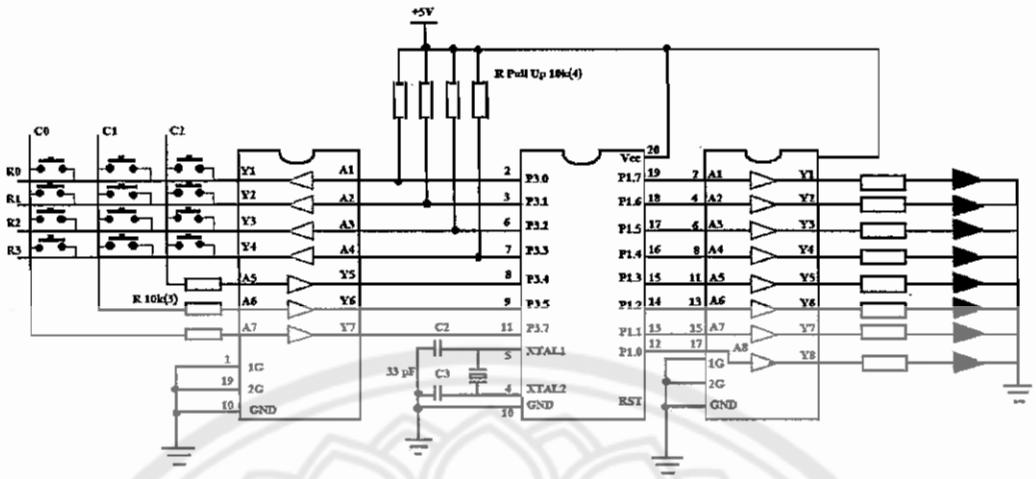
รูปที่ 2.34 แสดงไอซีสแกนคีย์สวิตช์

Switch	Data Output D	C	B	A
Y1, X1	0	0	0	0
Y1, X2	0	0	0	1
Y1, X3	0	0	1	0
Y1, X4	0	0	1	1
Y2, X1	0	1	0	0
Y2, X2	0	1	0	1
Y2, X3	0	1	1	0
Y2, X4	0	1	1	1
Y3, X1	1	0	0	0
Y3, X2	1	0	0	1
Y3, X3	1	0	1	0
Y3, X4	1	0	1	1
Y4, X1	1	1	0	0
Y4, X2	1	1	0	1
Y4, X3	1	1	1	0
Y4, X4	1	1	1	1



รูปที่ 2.35 แสดงไอซีสแกนคีย์สวิตช์ (ต่อ)

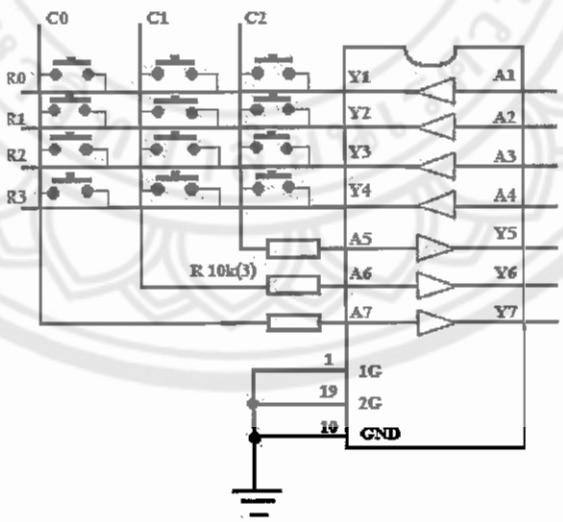
2.11.2 วงจรเมทริกซ์สวิตซ์



รูปที่ 2.36 วงจรเมทริกซ์สวิตซ์

จากรูปที่ 2.36 เป็นวงจรเมทริกซ์สวิตซ์ขนาด 4x3 สามารถเชื่อมต่ออินพุตได้ 12 บิต วงจรเมทริกซ์สวิตซ์จะต่อสวิตซ์แบบตารางโดยจะมีแถวและหลักทำให้สามารถเพิ่มจำนวนของอินพุตได้โดยมีจำนวนของสายสัญญาณเท่าเดิม หลักการตรวจสอบการทำงานของเมทริกซ์สวิตซ์จะวนรอบส่งลอจิก 0 ออกจากแถว R0 แล้วเลื่อนไปแถว R1 R2 และ R3 ตามลำดับ จากนั้นรอรับสัญญาณการกดสวิตซ์ของแต่ละหลักเข้าทางพอร์ตอินพุต แล้วจึงวนรอบกลับไปทำงานใหม่

2.11.3 การทำงานของวงจรเมทริกซ์สวิตซ์



รูปที่ 2.37 การตรวจสอบเมทริกซ์สวิตซ์

การอธิบายหลักการทำงานของเมทริกซ์สวิตซ์โดยภาพรวมอาจทำให้ยากต่อการทำความเข้าใจ ดังนั้น เพื่อให้ง่ายต่อการทำความเข้าใจจึงอธิบายการตรวจสอบเมทริกซ์สวิตซ์ทีละแถวโดยเริ่มจากแถว R0

จากรูปที่ 2.37 จะตรวจสอบการกดสวิตซ์ในแถว R0 โดยการส่งสัญญาณลอจิก 0 ออกทางพอร์ตจากนั้นรอรับข้อมูลจากการกดสวิตซ์ทางอินพุต C0 C1 และ C2 ถ้าสวิตซ์ในหลักใดถูกกดจะได้สัญญาณลอจิก 0 แสดงดังตารางที่ 2.10

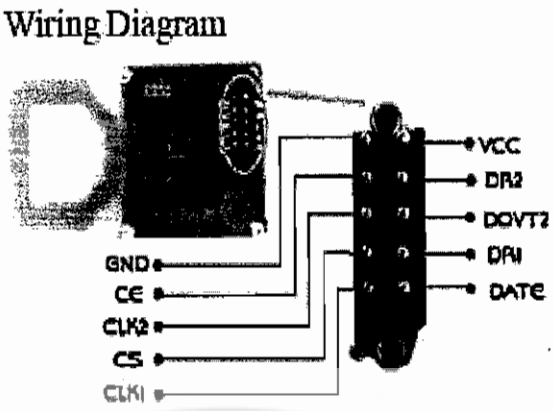
ตารางที่ 2.11 แสดงสถานการณ์กดเมทริกซ์สวิตซ์

สถานะการกดสวิตซ์	P3.4 / C1	P3.5 / C2	P3.7 / C3
ไม่มีการกดสวิตซ์	1	1	1
กดสวิตซ์ในหลัก C0	0	1	1
กดสวิตซ์ในหลัก C1	1	0	1
กดสวิตซ์ในหลัก C2	1	1	0

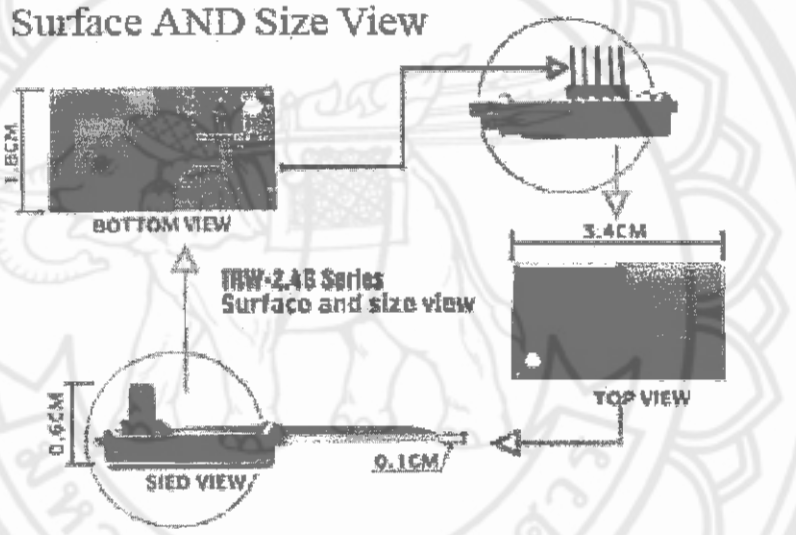
2.12 โมดูลความถี่วิทยุ 2.4 GHz

เป็นโมดูลสำเร็จรูปที่ใช้ รับ – ส่ง ข้อมูล ในแบบอนุกรม ใช้กับความถี่ 2.4 GHz ปรับแต่งสำเร็จรูป พร้อมมีสายอากาศในตัว ซึ่งสามารถใช้งานได้ในระยะไกล 250 m. (ความเร็วข้อมูล 250 Kbps) หรือระยะ 150 m. (ความเร็ว 1 Mbps) ในพื้นที่โล่งแจ้ง

1) ความถี่ในการใช้งาน 2.4 ~ 2.524 GHz
2) มีรูปแบบและความเร็วในการรับ ส่งข้อมูลแบบ (Gaussian Frequency Shift Keying)
3) ทำงานที่ความต่างศักย์ทางไฟฟ้า 3 V
4) กำลังงานเอาต์พุต + 4 dBm
5) ความเร็วในการรับส่งข้อมูลถึง 1 Mbps, 250 Kbps
6) ขนาด 20.0 * 36.7 * 2.4 mm
7) ทำงานที่อุณหภูมิ -40 ~ +85 Centigrade
8) ระยะทางในการรับส่งสัญญาณ 280 m (250 Kbps), 150 m (1 Mbps)
9) มีสายอากาศรับส่งสัญญาณในตัวโมดูลความถี่วิทยุ



รูปที่ 2.38 แสดงลักษณะขาของโมดูลความถี่วิทยุ (TRW – 2.4 GHz)



รูปที่ 2.39 แสดงรายละเอียดทางด้านบน, ด้านข้างและด้านหน้าของตัวโมดูลความถี่วิทยุ

2.12.1 การจัดขาของโมดูลความถี่วิทยุ (TRW - 2.4 GHz)

ตารางที่ 2.12 แสดงลักษณะการทำงานของขาของตัวโมดูลความถี่วิทยุ

ขา	ชื่อ	ลักษณะการทำงานของขา	รายละเอียด
1	GND	Power	Ground (0 V)
2	CE	Input	Chip Enable activates Rx or Tx mode
3	CLK2	I/O	Clock output/Input for Rx data channel 2
4	CS	Input	Chip Select aetivates Configuration mode
5	CLK1	I/O	Clock Input (Tx) & I/O(Rx) for data channel 1 3 - wire interface
6	DATA	I/O	Rx data channel 1/Tx data input /3 – wire interface
7	DR1	Output	Rx data ready at data channel 1 (Shock Burst Only)
8	DOUT2	Output	Rx data channel 2
9	DR2	Output	Rx data ready at data channel 2 (Shock Burst only)
10	VCC	Power	Power Supply (+ 3V DC)

** การเซตค่าของแต่ละขาในโหมดต่างๆสามารถดูได้จากตารางที่ 2.12

ตารางที่ 2.13 การเซตค่าของแต่ละขาในโหมดต่างๆ

โหมด	PWR_UP	CE	CS
Active (RX/TX)	1	1	0
Configuration	1	0	1
Stand by	1	0	0
Power down	0	X	X

2.12.2 โหมดการทำงานของ (TRW - 2.4 GHz)

โหมดในการใช้งานของ (TRW - 2.4 GHz) มีอยู่ 2 โหมด Shock Burst Mode และ Direct Mode โดยในโครงงานนี้ได้กำหนดให้โมดูลความถี่วิทยุมีการทำงานในโหมด Shock Burst ซึ่งมีรายละเอียดดังต่อไปนี้

โหมด Shock Burst

โหมด Shock Burst เป็นการใช้เทคโนโลยีรับ/ส่งข้อมูลบนชิปแบบเข้าก่อน – ออกก่อน (First in – First out) โดยในการส่งข้อมูลมีทั้งระดับอัตราในการส่งบิตข้อมูลมีทั้งความเร็วต่ำและระดับความเร็วสูง เมื่อโมดูลความถี่วิทยุทำงานในโหมด Shock Burst สามารถเพิ่มการเข้าถึงระดับข้อมูลได้สูง (1 Mbps) โดยใช้ย่านความถี่ 2.4 GHz และต้องใช้ไมโครคอนโทรลเลอร์ความเร็วสูงในการประมวลผล โดยการจัดกระบวนการประมวลผลให้เหมาะสมกับโปรโตคอลบนชิปจะทำให้ได้รับประโยชน์จากโมดูลความถี่วิทยุที่ตามมาดังนี้

1. ประหยัดกระแส
2. ระบบมีราคาต่ำ (เนื่องจากไมโครคอนโทรลเลอร์มีราคาถูก)
3. ลดการชนกันของข้อมูลเมื่อใช้เวลาในการส่งระยะสั้นๆ

รายละเอียดทางไฟฟ้า

ตารางที่ 2.14 รายละเอียดทางไฟฟ้า

ชื่อ	ความหมาย	ค่าต่ำสุด	ค่าสูงสุด	หน่วย
	เงื่อนไข			
VDD	แรงดัน	1.9	3.6	V
TEMP	อุณหภูมิ	-40	85	C
	อินพุตดิจิตอล			
V_{IH}	ระดับแรงดันอินพุต HIGH	VDD - 0.3	VDD	V
V_{IL}	ระดับแรงดันอินพุต LOW	Vss	0.3	V
	เอาต์พุตดิจิตอล			
V_{OH}	ระดับแรงดันเอาต์พุตHIGH(I_{OH}=0.5mA)	VDD - 0.3	VDD	V
V_{OL}	ระดับแรงดันเอาต์พุต LOW(I_{OH}=0.5mA)	Vss	0.3	V
	เงื่อนไขต่างๆเกี่ยวกับความถี่			
fop	ช่วงความถี่วิทยุ	2400	2524	MHz
R_{GFSK}	อัตราการส่งข้อมูลในโหมด Shock Burst	> 0	1000	kbps
	กระแสในการใช้งานที่อัตราส่งต่างกันของ 1 ช่องสัญญาณ			
I_{VDD}	250 kbps	18		mA
I_{VDD}	1000 kbps	19		mA

หลักการทำงานในโหมด Shock Burst

เมื่อทำการกำหนดค่าให้โมดูลทำงานในโหมด Shock Burst แล้วการทำงานของโมดูลในการ รับ/ส่ง ข้อมูลมีหลักการทำงานดังนี้

การส่งข้อมูลในโหมด Shock Burst

โดยการเชื่อมต่อไมโครคอนโทรลเลอร์กับขา CE, CLK1, DATA ของตัวโมดูล เมื่อไมโครคอนโทรลเลอร์ต้องการส่งข้อมูลให้กับโมดูลต้องทำการเซตขา CE ให้อยู่ในสถานะ "high" เพื่อกระตุ้นให้โมดูลทำการประมวลผลข้อมูล ทำการเซตไมโครคอนโทรลเลอร์ให้อยู่ในสถานะ "low" เพื่อกระตุ้นให้โมดูลทำการส่งข้อมูล

การรับข้อมูลในโหมด Shock Burst

โดยไมโครคอนโทรลเลอร์ทำการเชื่อมต่อกับขา CE, CLK1, DR1 และ DATA (กรณีที่ใช้ช่องสัญญาณเพียงช่องเดียว) เมื่อ RF package มีแอดเดรสที่ถูกต้องและขนาดของข้อมูลที่เข้ามา ตัวโมดูลจะทำการเซตค่าให้ขา CE อยู่ในสถานะ "high."

เมื่อข้อมูลที่รับเข้ามาถูกต้อง (แอดเดรสและ CRC ถูกต้อง) โมดูลจะทำการย้าย preamble, address, และ CRC โดยแจ้งไปยังไมโครคอนโทรลเลอร์ให้ทำการเซตค่า DR1 ให้อยู่ในสถานะ "high" และเซตค่า CE ให้อยู่ในสถานะ "low" เพื่อบอกว่าขณะนี้ทำการรับข้อมูลอยู่

ไมโครคอนโทรลเลอร์จะทำการเซตค่าเพื่อให้รับข้อมูลได้เหมาะสมและเมื่อทำการรับข้อมูล เสร็จเรียบร้อยจะทำการเซตค่าให้ DR1 ให้อยู่ในสถานะ "low" เพื่อเตรียมพร้อมที่จะรับข้อมูลที่เข้ามาใหม่

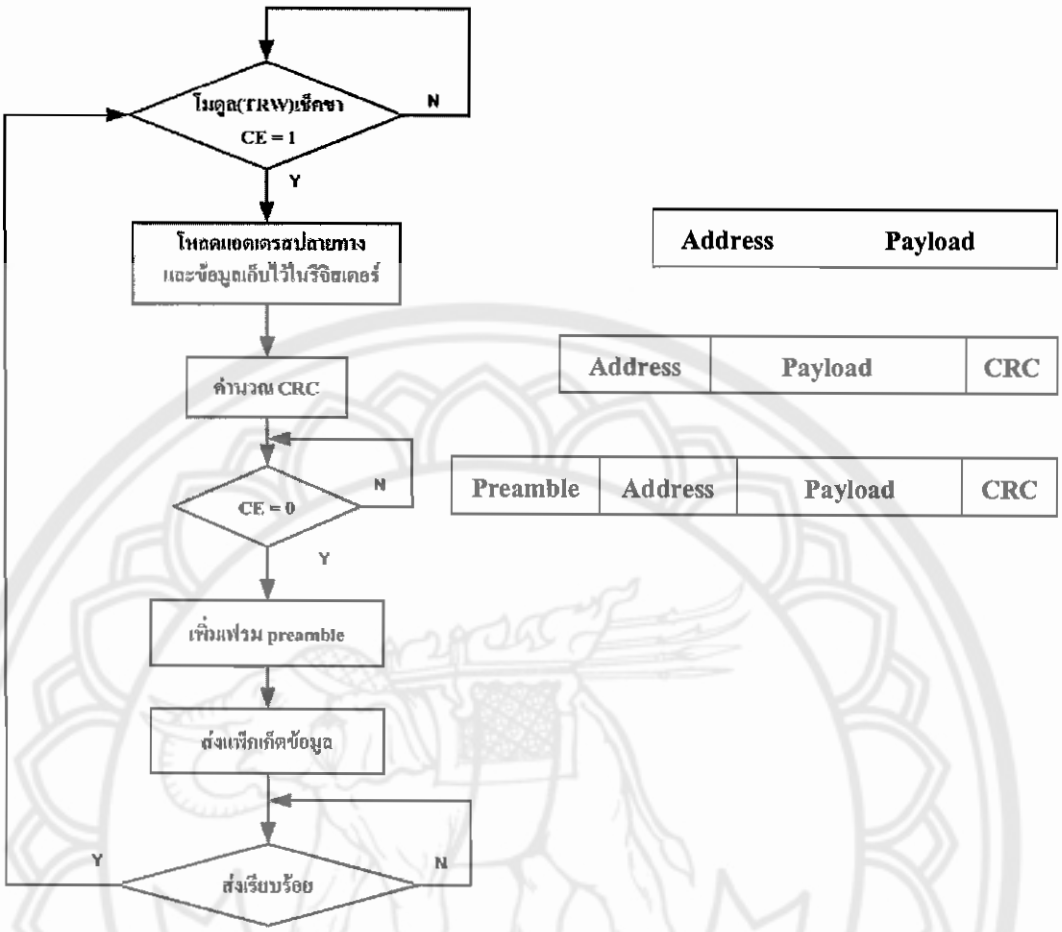
2.12.3 ส่วนประกอบของชุดข้อมูล

Preamble	Address	Payload	CRC

ส่วนประกอบของชุดข้อมูลแบ่งได้เป็น 4 ส่วน คือ

- **Preamble** เป็นส่วนแรกของชุดข้อมูล เพื่อแสดงจุดเริ่มต้นของเฟรมข้อมูล
- **Address** เป็นส่วนที่ระบุแอดเดรสของตัวรับข้อมูล
- **Payload** เป็นส่วนที่เก็บข้อมูล
- **CRC** เป็นส่วนที่ใช้ตรวจสอบความผิดบิตของชุดข้อมูล

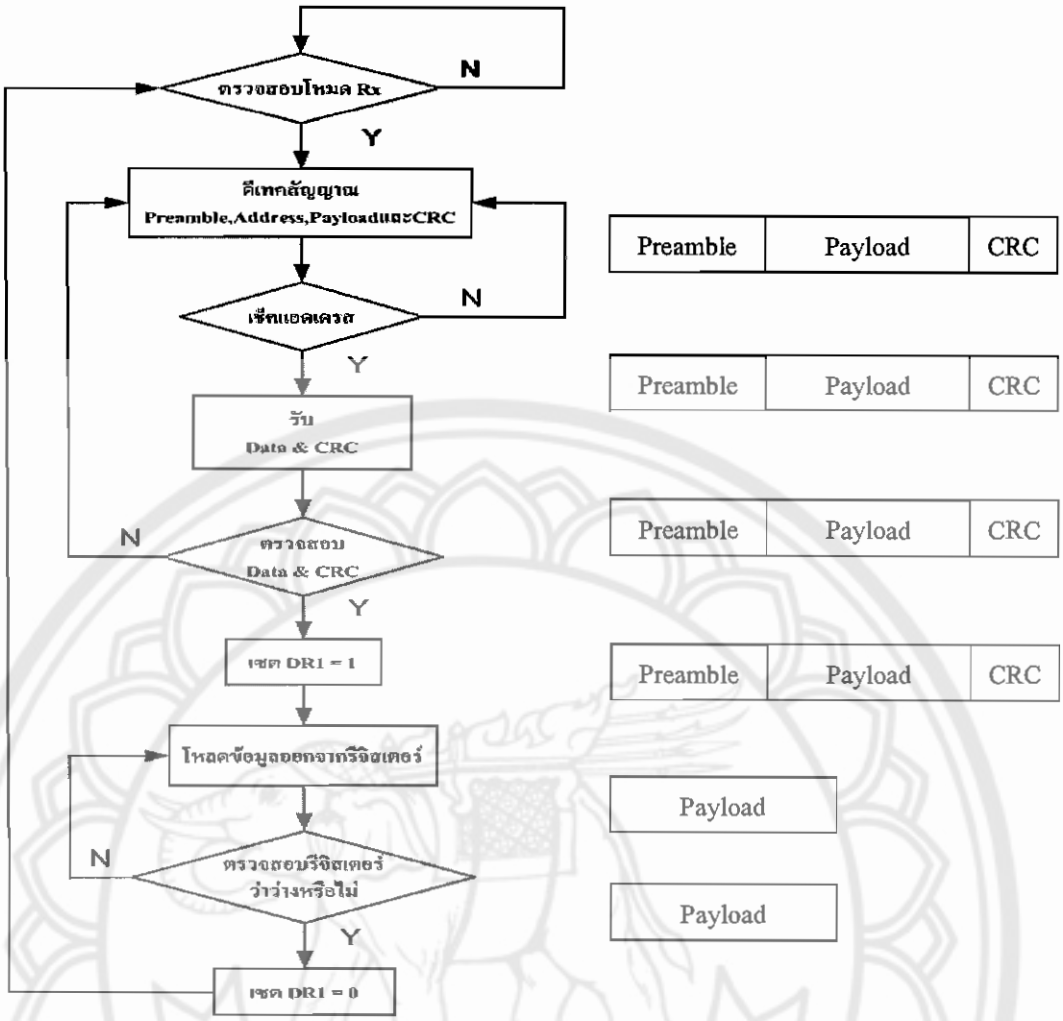
หลักการทำงานของโมดูลความถี่วิทยุ (TRW - 2.4 GHz) ในการส่งและรับข้อมูล



รูปที่ 2.40 โฟลว์ชาร์ต shock Burst Transmit ของโมดูลความถี่วิทยุ (TRW - 2.4 GHz)

อธิบายการทำงานของโฟลว์ชาร์ต ตัวส่ง

หลังจากทำการกำหนดรูปแบบ (Configuration) ให้เป็นโหมด Tx แล้ว จากนั้นหลักการทำงานของโมดูลจะเป็นดังโฟลว์ชาร์ตรูปที่ 2.39 โดยโมดูลจะตรวจสอบว่าขา CE เป็น High หรือไม่ ถ้าใช่โมดูลจะรับแอดเดรสปลายทางและข้อมูล จากนั้นจะทำการคำนวณหา CRC เพื่อเป็นเฟรมที่ใช้ในการตรวจสอบข้อมูล จากนั้นจะตรวจสอบขา CE ถ้าเป็น 0 แสดงว่าเฟรมข้อมูลและ CRC พร้อมที่จะส่งแล้ว จากนั้นโมดูลจะทำการเพิ่มเฟรม preamble เข้าไปแล้วทำการส่งแพ็คเก็ตของข้อมูลทั้งหมดและเมื่อส่งเสร็จเรียบร้อยแล้ว จะทำการวนกลับไปยังจุดเริ่มต้นของโฟลว์ชาร์ตใหม่อีกครั้ง



รูปที่ 2.41 โฟลว์ชาร์ต Shock Burst Receive ของโมดูลความถี่วิทยุ (TRW - 2.4 GHz)

อธิบายการทำงานของโฟลว์ชาร์ต ตัวรับ

จากรูปที่ 2.40 หลังจากทำการกำหนดรูปแบบ ให้เป็นโหมด Rx แล้วโมดูลจะทำการดีเทคสัญญาณจากโมดูลตัวส่ง หลังจากดีเทคสัญญาณพบแล้วจะทำการตรวจสอบแอดเดรสที่ส่งมาตรงกับแอดเดรสประจำตัวของโมดูลตัวรับที่ทำการตั้งไว้หรือไม่ ถ้าใช่จะทำการรับข้อมูลกับ CRC จากนั้นทำการเช็คข้อมูลว่าถูกต้องหรือไม่ ถ้าถูกต้องจะทำการเซต DR = 1 high จากนั้นจะทำการโหลดข้อมูลออกจากรีจิสเตอร์ของโมดูลโดยจะตรวจสอบถ้ารีจิสเตอร์ของโมดูลว่าว่างหรือไม่ จากนั้นจะทำกาเซต DR1= 0 และวนกลับไปยังจุดเริ่มของโฟลว์ชาร์ต

2.12.4 ตำแหน่งบิตข้อมูลของโมดูลความถี่วิทยุ

ในการที่จะกำหนดค่าให้กับตัวโมดูลความถี่วิทยุเพื่อให้โมดูลความถี่วิทยุทำงานในโหมดที่เราต้องการทราบก่อนว่าต้องกำหนดค่าอะไรบ้างลงในตำแหน่งบิตที่เท่าไร ดังนั้นตารางข้างล่างนี้แสดงตำแหน่งของบิตภายในตัวโมดูลความถี่วิทยุที่ใช้ในการกำหนดค่าให้กับตัวโมดูลความถี่วิทยุ

ตารางที่ 2.15 แสดงรายละเอียดของตำแหน่งบิตภายในตัวโมดูลความถี่วิทยุ

ตำแหน่งบิต	จำนวนบิต	ชื่อ	ความหมาย
143 : 120	24	Test	จองไว้สำหรับทดสอบข้อมูล
119 : 112	8	DATA2_W	จำนวนความกว้างของบิตข้อมูล Channel 2
111 : 104	8	DATA1_W	จำนวนความกว้างของบิตข้อมูล Channel 1
103 : 64	40	ADDR2	แอดเดรสของตัวรับ Channel 2
63 :24	40	ADDR1	แอดเดรสของตัวรับ Channel 1
23 :18	6	ADDR_W	จำนวนบิตที่จองไว้สำหรับแอดเดรสตัวรับ
17	1	CRC_L	จำนวนความกว้างของบิต CRC
16	1	CRC_EN	ยอมให้สร้าง CRC สำหรับ Tx และมีการตรวจสอบบิต CRC สำหรับ Rx
15	1	RX2_EN	กำหนดจำนวนสัญญาณ Rx
14	1	CM	เลือกโหมด (Direct หรือ Shock Burst)
13	1	REDR_SB	เลือกอัตราการส่งผ่านข้อมูล(1 Mbps และ 250 Kbps)
12 : 10	3	XO_F	ความถี่คริสตอล
9 : 8	2	RF_PWR	เลือกค่ากำลังของตัวส่ง
7 : 1	7	RF_CH#	เลือกความถี่
0	1	RXEN	เลือกโหมดรับ / ส่ง