ระบบเซนเซอร์ไร้สาย

Wireless sensor

โดย

นาย อธิษฐ์ กลกะสิงห์

นายคมสันต์ ทองน้อย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ระบบเซ็นเซอร์ไร้สาย

WIRELESS SENSOR

โดย

นายอธิษฐ์ กลกะสิงห์ รหัส 46010908

นายคมสันต์ ทองน้อย รหัส 46010084

อาจารย์ที่ปรึกษา

ดร.ศิริเดช บุญแสง

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

โครงงานเรื่อง ระบบเซ็นเซอร์ไร้สาย

Wireless Sensor

จัดทำโดย นายอธิษฐ์ กลกะสิงห์ รหัส 46010908 ชั้นปีที่ 4

นายคมสันต์ ทองน้อย รหัส 46010084 ชั้นปีที่ 4

อาจารย์ที่ปรึกษา ดร.ศิริเดช บุญแสง

ปริญญานิพนธ์ฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

.....อาจารย์ที่ปรึกษา

(ดร.ศิริเดช บุญแสง)

วันที่ ...../...../.....

ระบบเซ็นเซอร์ไร้สาย

นายอธิษฐ์ กลกะสิงห์ รหัส 46010908
นายคมสันต์ ทองน้อย รหัส 46010084
ดร. ศิริเดช บุญแสง อาจารย์ที่ปรึกษา
ภาคเรียนที่2 ปีการศึกษา 2549

บทคัดย่อ

โครงงานนี้ได้กล่าวถึงการสร้างเครื่องวัดอุณหภูมิแล้วส่งคลื่นวิทยุไปที่ตัวรับสัญญาณ แล้วแสดงผลออกมาทาง LCD เป็นจำนวนเต็ม 2 หลัก และทศนิยม 1 หลัก โดยใช้ตัวมอดดูเลตเป็นโมดูลสำเร็จ (RLP434A และ TLP 434A) ใช้ AT89C51 เป็นตัวเรียกคำสั่งไปที่DS1820 แล้วทำการรับค่าแล้วแปลงส่งออกพอร์ตอนุกรมไปที่TLP434A ขณะที่ตัวรับจะทำการรับค่าจากRLP434A แล้วทำการแปลงข้อมูล ส่งไปที่LCD อีกทั้งยังเป็นตัวควบคุม LCDด้วยและใช้ DS1820 เป็นเซ็นเซอร์ตรวจจับอุณหภูมิแบบดิจิตอล โดยสามารถตรวจจับอุณหภูมิได้ตั้งแต่ -55 ถึง +125 องศาเซลเซียส ความแม่นยำ บวกลบ 0.5 องศาเซลเซียส โดยเครื่องส่งสามาถส่งได้เป็นระยะทางอย่างน้อย 20 เมตร จำนวน 5 ตัว

Wireless Sensor

Mr. Athit Kolkasing ID 46010908

Mr. Khomson TongnoiID46010084

Dr. Siridej Boonsaeng Advisor

Semester 2Year of Education 2006

Abstract

This project is about to create five temperature detectors and transfer the data as radio frequency to only one receiver and display it as 2 digits of integer and 1 digit of decimal on LCD. The instant module (RLP434A and TLP434A) is used as the modulator and RF transmitter. AT89C51 is used to read temperature from the digital temperature sensor (DS18S20). DS18S20 can detect the temperature between -55°C to +125°C. Accuracy ±0.5°C.The transmitter can transfer the data for 20 meters at least of distance.

II

กิตติกรรมประกาศ

โครงงานระบบเซ็นเซอร์ไร้สาย ซึ่งประกอบด้วยชิ้นงานและเอกสารประกอบโครงงานนี้ที่สำเร็จลุล่วงมาด้วยดีมิได้หากขาดอาจารย์ศิริเดช บุญแสง ผู้ให้คำแนะนำ ดูแลในเรื่องของวงจรอย่างใกล้ชิดมาโดยตลอด พร้อมทั้งการให้ความช่วยเหลือจากรุ่นพี่และเพื่อนที่ห้องโปรเจค สุดท้ายบุคคลที่จะลืมมิได้เลยคือ ผู้ปกครองซึ่งคอยนับสนุนเงินทุนและคอยให้กำลังใจเสมอมา

นายอชิษฐ์ กลกะสิงห์
นายคมสันต์ ทองน้อย
ผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่1
บทนำ

ความเป็นมาของโครงงาน

เซนเซอร์ไร้สาย เป็นโครงงาน ที่ใช้ในการวัดอุณหภูมิ แล้วส่งข้อมูลออกมาแบบไร้สาย สามารถนำไปประยุกต์ใช้ได้หลากหลาย เช่นวัดอุณหภูมิของต้นไม้ ในสวน แล้วส่งเข้ามาแสดงผลในบ้าน ช่วยให้เห็นถึงความเปลี่ยนแปลงโดยที่เราไม่จำเป็นต้องไปตรวจสอบ ซึ่งสามารถพัฒนาให้ส่งได้หลายๆต้นพร้อมกันต่อไปได้ ในการใช้งานจริง

วัตถุประสงค์

เพื่อศึกษาการใช้งานไมโครคอนโทรลเลอร์ การสื่อสารแบบไร้สายโดยใช้คลื่นวิทยุ การใช้งาน sensor วัดอุณหภูมิ

ขอบเขตของโครงงาน

เทอม1 โครงงาน wireless sensor นี้ ได้ออกแบบและสร้างแบ่งเป็นส่วนรับและส่วนส่ง ส่วนส่งสามารถรับอุณหภูมิ ได้ตั้งแต่ 0 ถึง 85 องศาเซลเซียส และส่งด้วยคลื่นวิทยุออกไปที่ความถี่ 433.92 MHz และตัวรับสามารถแสดงผลออกมาทาง หน้าจอ LCD ได้

เทอม2 สามารถรับข้อมูลจากตัวส่ง 4 ตัว แล้วแสดงผลสลับกันทางจอ LCD

โครงสร้างของรายงาน

รายงานฉบับนี้ได้อธิบายขั้นตอนและวิธีการออกแบบรวมทั้งวงจร และผลการทดลองทดสอบคุณสมบัติต่างๆ โดยมีเนื้อหาแบ่งเป็นบทต่างๆ ดังนี้

บทที่ 2 ทฤษฎี จะกล่าวถึงทฤษฎี และหลักการพื้นฐานต่างๆ ที่เกี่ยวข้องกับการออกแบบ

บทที่ 3 การออกแบบ จะกล่าวถึงขั้นตอนในการออกแบบส่วนต่างๆ ตั้งแต่ส่วนอุปกรณ์ภาคส่งและภาครับ

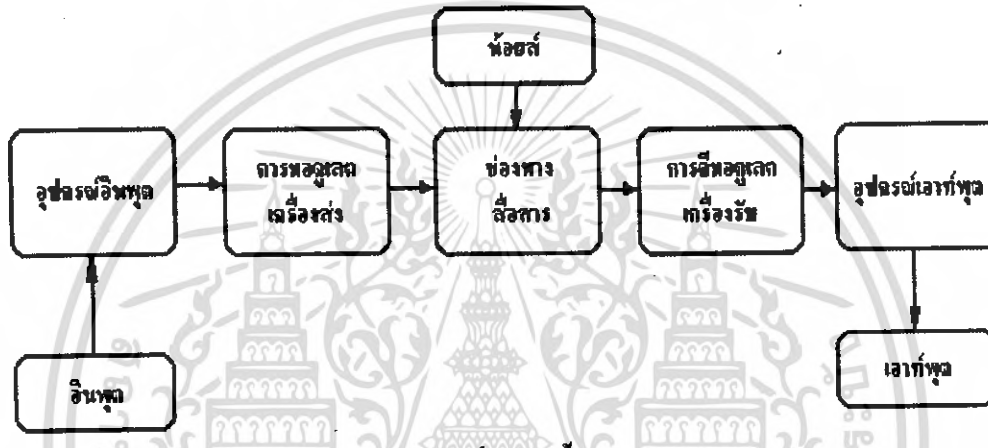
บทที่ 4 การทดลอง และผลการทดลอง จะกล่าวถึงการทดลอง และผลการทดลอง เมื่อทำการทดสอบคุณสมบัติต่างๆของภาครับและภาคส่ง

บทที่ 5 สรุป และวิจารณ์

บทที่ 2
ทฤษฎี

ระบบสื่อสาร

ในระบบสื่อสารไม่ว่าระบบใดก็ตามแผนผังพื้นฐานมักเป็นเหมือนรูปด้านล่างซึ่งระบบสื่อสารบนพื้นฐานประกอบด้วยอุปกรณ์อินพุท (Input device) เครื่องส่ง ช่องทางสื่อสาร (Communication channel) หรือแชนแนล ซึ่งมักมีนอยส์มารบกวนเครื่องรับและอุปกรณ์เอาท์พุท (Output device)



รูปที่ 2.1 ระบบสื่อสารพื้นฐาน

อุปกรณ์อินพุทและเอาท์พุท ความจริงอุปกรณ์อินพุทก็คือ อุปกรณ์ที่แปลงข่าวสารเป็นสัญญาณไฟฟ้า ส่วนอุปกรณ์เอาท์พุทก็คืออุปกรณ์ที่แปลงสัญญาณไฟฟ้ากลับมาเป็นข่าวสารนั่นเอง มีชื่อเรียกแตกต่างกันออกไปแล้วแต่การใช้งานเช่น ในระบบวิทยุกระจายเสียง อุปกรณ์อินพุทอาจเป็นไมโครโฟน และอุปกรณ์เอาท์พุทจะเป็นลำโพง สำหรับไมโครโฟนทำหน้าที่แปลงคลื่นเสียงเป็นสัญญาณไฟฟ้า และส่วนลำโพงทำหน้าที่แปลงสัญญาณไฟฟ้ากลับเป็นคลื่นเสียง

ในทำนองเดียวกัน ในระบบแพร่ภาพทางโทรทัศน์ อุปกรณ์อินพุทก็คือกล้องถ่ายโทรทัศน์ ซึ่งเปลี่ยนพลังงานแสง (จากภาพ) ไปเป็นสัญญาณไฟฟ้า และอุปกรณ์เอาท์พุทก็คือหลอดภาพโทรทัศน์ซึ่งเปลี่ยนสัญญาณไฟฟ้ากลับคืนเป็นพลังงานแสง

อุปกรณ์อินพุทและเอาท์พุทของระบบสื่อสารยังมีอีกมากมาย เช่น คันเคาะโทรเลข เครื่องโทรพิมพ์ เครื่องโทรสาร เครื่องโทรมาตร(telemetry) ฯลฯ อุปกรณ์อินพุทและเอาท์พุทจะต่อเข้ากับเครื่องส่งและเครื่องรับเสมอ

ข่าวสารที่รับและส่งระหว่างกันแบ่งออกเป็น 3 พวกใหญ่ คือ

1. เสียงหรือออดิโอ (audio) ได้แก่ เสียงพูดในระบบโทรศัพท์ เสียงพูด เสียงเพลง หรือเสียงดนตรี ซึ่งต้องการคุณภาพเสียงดีในระบบวิทยุกระจายเสียง

2. ภาพ (picture) ได้แก่ ภาพนิ่งในระบบโทรสาร (facsimile) และระบบส่งภาพระยะไกล (telephoto) ภาพยนตร์ในระบบโทรทัศน์

3. ข้อมูล (data) ส่วนใหญ่ส่งมาเป็นรหัสให้แก่เครื่องยนต์ เครื่องจักร เครื่องคอมพิวเตอร์ ฯลฯ ได้แก่ ข้อมูลและคำสั่งในระบบโทรมาตร ตัวอักษรในระบบโทรพิมพ์หรือโทรเลข ข้อมูลคอมพิวเตอร์ในระบบคอมพิวเตอร์

เครื่องส่ง เครื่องส่งทำหน้าที่รับสัญญาณไฟฟ้าจากอุปกรณ์อินพุท แล้วทำการมอดูเลตกับคลื่นพาห์ความถี่สูง เครื่องส่งประกอบด้วยแหล่งกำเนิดสัญญาณความถี่สูง (เรียกว่า Oscillator) กับมอดูเลต เครื่องส่งส่วนใหญ่มักมีภาคขยายอีกเพื่อให้สัญญาณที่ส่งออกมีกำลังแรง ทำให้สื่อสารได้ไกลขึ้น

ช่องทางสื่อสาร ช่องทางสื่อสารในที่นี้ ได้แก่ บรรยากาศอวกาศว่าง (free space) หรือสายฯลฯ แต่ในที่นี้เราจะกล่าวถึงเฉพาะระบบวิทยุเท่านั้น ช่องทางสื่อสารของระบบวิทยุอาศัยการแผ่คลื่นวิทยุออกไป โดยผ่านบรรยากาศที่เป็นตัวกลาง (medium) ซึ่งคลื่นเดินทางจากเครื่องส่งผ่านไปยังเครื่องรับ

ความถี่และความยาวคลื่น เรานิยมแบ่งคลื่นวิทยุออกเป็นย่านความถี่ต่างๆ โดยมีหน่วยเป็นเฮิรตซ์ (Hertz) ในประวัติศาสตร์การวิทยุ เราแบ่งคลื่นวิทยุตามความยาวคลื่น (Wave length) ความสัมพันธ์ระหว่างความยาวคลื่นกับความถี่เป็นดังนี้

$v = lf$

เมื่อ v คือ ความเร็วของคลื่นวิทยุในอากาศ เท่ากับความเร็วแสง $3x10^8$ เมตร/วินาที

l คือ ความยาวคลื่น มีหน่วยเป็นเมตร

f คือ ความถี่ มีหน่วยเป็นเฮิรตซ์

ตารางที่ 1 แสดงย่านความถี่ ความถี่ และความยาวคลื่น

ย่านความถี่	ความถี่	ความยาวคลื่น
Very Low Frequency (VLF)	ต่ำกว่า 30 kHz	ยาวกว่า 10 km
Low Frequency (LF)	30 – 300 kHz	10 – 1 km
Medium Frequency (MF)	300 – 3000 kHz	1000 – 100 m
High Frequency (HF)	3 – 30 MHz	100 – 10 m
Very High Frequency (VHF)	30 – 300 MHz	10 – 1 m
Ultra High Frequency (UHF)	300 – 3000 MHz	100 – 10 cm
Super High Frequency (SHF)	3 – 30 GHz	10 – 1 cm
Extremely High Frequency (EHF)	30 – 300GHz	10 – 1 mm

นอยส์ เป็นสัญญาณที่เข้ามาแทรกแซงหรือรบกวน (Interfere) นอยส์ที่รับมาได้แบ่งออกเป็น 4 ประเภท คือ

1) นอยส์บรรยากาศ (Atmospheric noise) เกิดขึ้นจากความแปรปรวนของบรรยากาศที่ห่อหุ้มโลก เช่น ฟ้าแลบ ฟ้าผ่า ก่อให้เกิดคลื่นวิทยุแผ่ออกไปรอบโลก นอยส์บรรยากาศเกิดขึ้นอยู่ตลอดเวลาแม้จะไม่มีพายุฝนฟ้าคะนองก็ตาม

2) นอยส์จากอวกาศ (Space noise) เกิดจากดวงอาทิตย์และดวงดาวนับล้านๆดวงในจักรวาล ดวงอาทิตย์เป็นวัตถุที่มีขนาดมหึมาและมีความร้อนสูงถึง 6000 องศาเซลเซียสที่ผิวดวงอาทิตย์ ฉะนั้นดวงอาทิตย์จะแผ่พลังงานออกมามีสเปกตรัมความถี่กว้างมาก พลังงานนี้ปรากฏออกมาเป็นนอยส์คงที่ อย่างไรก็ตามที่ผิวดวงอาทิตย์ยังมีความแปรปรวนอื่นๆอีก เช่น จุดบนดวงอาทิตย์ (Sun spot) การลุกโชติช่วง (Solar flare) ซึ่งก่อให้เกิดนอยส์เพิ่มขึ้นอีก นอกจากนี้ดวงอาทิตย์บางดวงที่ไกลออกไปจากระบบสุริยะจักรวาลก็มีคุณสมบัติเหมือนดวงอาทิตย์ คือ มีความร้อนสูงและสามารถกำเนิดนอยส์มายังโลกได้

3) นอยส์ที่เกิดจากสิ่งประดิษฐ์ที่มนุษย์สร้างขึ้น (Man-made noise) ได้แก่นอยส์จากมอเตอร์ไฟฟ้า เช่นพัดลมที่เป่าผม เครื่องดูดฝุ่น นอกจากนี้ก็ยังมีนอยส์จากระบบจุดระเบิดของรถยนต์ การรั่วของสายไฟแรงสูง หลอดไฟฟลูออเรสเซนต์ ฯลฯ

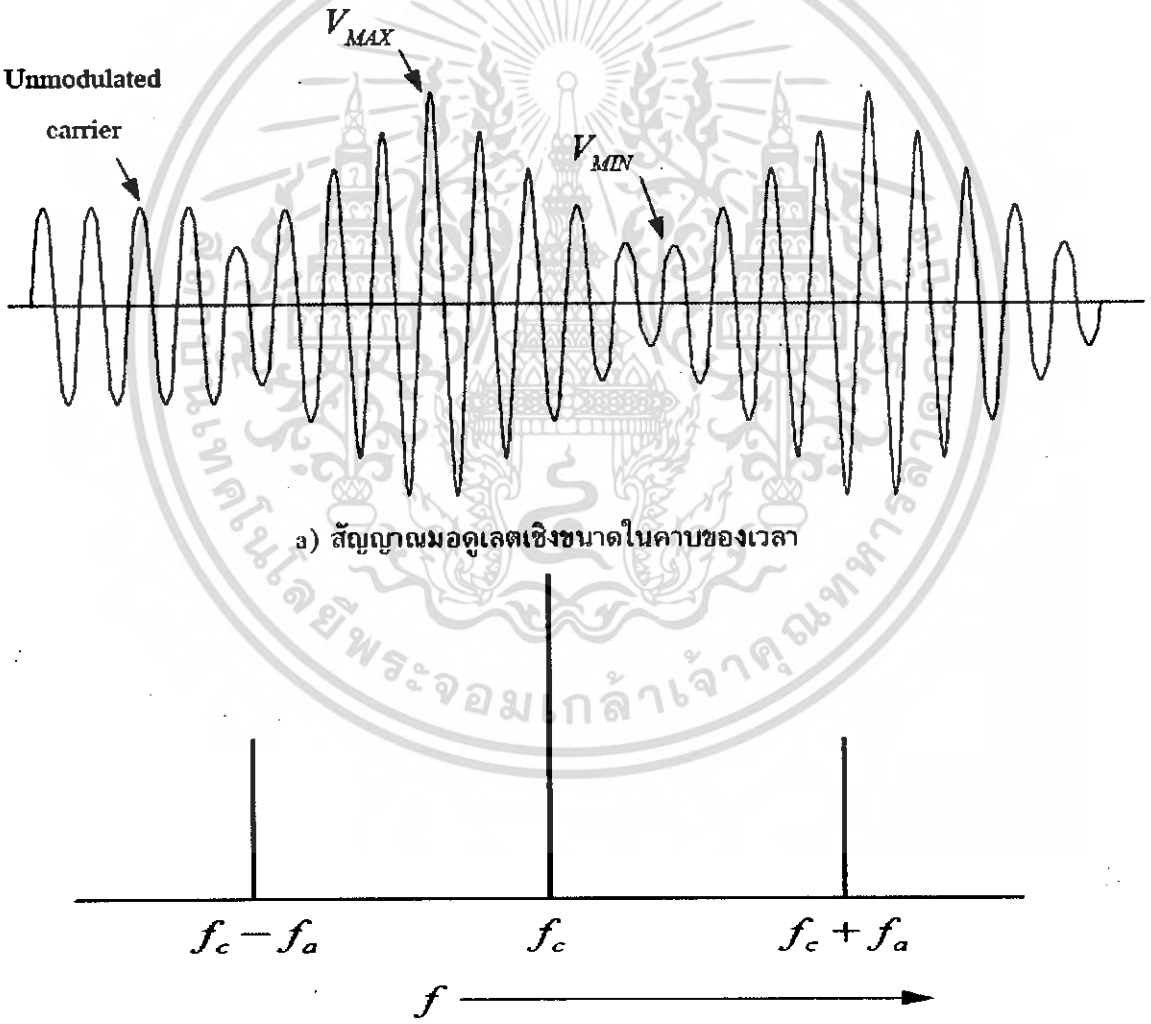
4) นอยส์กายในอุปกรณ์ในเครื่องรับ (Internal noise) แยกเป็น 2 ประเภท คือ นอยส์อุณหภูมิ (Thermal noise) และช็อตนอยส์ (Short noise) นอยส์จากอุณหภูมิเกิดจากการเคลื่อนที่ของอิเล็กตรอนในตัวอุปกรณ์ บางครั้งเรียกว่า จอห์นสัน (Johnson noise) ส่วนช็อตนอยส์เกิดขึ้นในอุปกรณ์แอกทีฟ (Active noise) ทุกชนิด เนื่องจากการรวมตัวของอิเล็กตรอนกับโฮล เช่น ในทรานซิสเตอร์ซึ่งไม่ขึ้นกับอุณหภูมิ

เครื่องรับ เมื่อรับสัญญาณจากเครื่องรับ สัญญาณจะมีกำลังอ่อนลงและยังมีนอยส์เข้ามาแทรกแซงสัญญาณที่ต้องการจะรับอีกด้วย ดังนั้นการรับสัญญาณอ่อน ๆ เช่นนี้ เครื่องรับจึงต้องมีความสามารถพิเศษในการเลือกรับและขยายเอาเฉพาะสัญญาณความถี่ที่ต้องการพร้อมทั้งต้องมีกรรมวิธีในการกำจัดนอยส์หรือต่อสู้เอาชนะนอยส์รบกวน สัญญาณที่รับได้จะผ่านการดีมอดเพื่อแปลงสัญญาณข่าวสารที่เข้ามอดูเลตกลับมา กรรมวิธีนี้ค่อนข้างสลับซับซ้อนพอสมควร

การมอดูเลต และการดีมอดูเลตแบบอนาลอก ความถี่ของข้อมูลหรือสัญญาณโดยทั่วไป มักจะมีความถี่ต่ำ การแพร่กระจายของคลื่นแม่เหล็กไฟฟ้าความถี่ต่ำจะกระทำได้ไม่ดี เพราะสัญญาณความถี่ต่ำจะมีความยาวคลื่นมาก เราสามารถที่จะเลื่อนความถี่ของสัญญาณให้มีค่าสูงขึ้นได้โดยการมอดูเลตสัญญาณที่ต้องการจะส่งกับคลื่นพาห์(carrier) ความถี่สูง หรือกล่าวอีกนัยหนึ่งได้ว่าการมอดูเลตคือกระบวนการที่สัญญาณที่จะส่ง(Modulating Signal) ทำให้คุณสมบัติของคลื่นพาห์ (ขนาด

ความถี่ และเฟส) เปลี่ยนแปลงไปตามสภาวะของสัญญาณ สัญญาณที่ได้จากการมอดูเลตเรียกว่า Modulated Signal (wave)

1) การมอดูเลตเชิงขนาด (AM) คือ ผสมสัญญาณข่าวสารกับสัญญาณพาห์โดยสัญญาณเอาท์พุทที่ได้จากการมอดูเลตจะเป็นสัญญาณพาห์ที่มีการเปลี่ยนแปลงขนาดซึ่งมีการเปลี่ยนแปลงตามขนาดของสัญญาณข่าวสารที่นำมามอดูเลตดังแสดงในรูปที่ 2.2 การเปลี่ยนแปลงของสัญญาณพาห์ในรูป 2.2a เรียกว่า กรอบของสัญญาณการมอดูเลต ซึ่งมีการเปลี่ยนแปลงรูปร่างของสัญญาณเปลี่ยนแปลงตามสัญญาณที่เข้ามามอดูเลต(เป็นบวกหรือลบ) สัญญาณที่แสดงได้มาจากการมอดูเลตกับสัญญาณที่เป็นซายน์ในขณะไม่มีการมอดูเลตสัญญาณพาห์มีขนาดของสัญญาณ อยู่ระหว่าง Vmax และ Vmin หรือมีค่าเท่ากับ (Vmax + V min)/ 2 สิ่งสำคัญที่มีผลต่อขนาดของสัญญาณก็คือสัญญาณที่นำมาเข้ามามอดูเลตซึ่งทำให้ขนาดของ สัญญาณพาห์มีการเปลี่ยนแปลง



a) สัญญาณมอดูเลตเชิงขนาดในคาบของเวลา

b) สัญญาณในคาบของความถี่

รูปที่ 2.2 สัญญาณมอดูเลตเชิงขนาด

จากรูปเกิดจากการมอดูเลตที่มีขนาดของสัญญาณข่าวสารเป็นตัวกำหนดจุดเปลี่ยนแปลงของสัญญาณพาห์ ถ้าขนาดของสัญญาณข่าวสารสูงสุดเท่ากับขนาดของสัญญาณพาห์จะสามารถหาค่าดัชนีการมอดูเลต (m_a) ของการมอดูเลตเชิงขนาดได้จาก

$$m_a = \frac{v_a}{v_c}$$

เมื่อ V_a คือแอมปลิจูดของสัญญาณที่เข้ามามอดูเลต

V_c คือแอมปลิจูดของสัญญาณพาห์

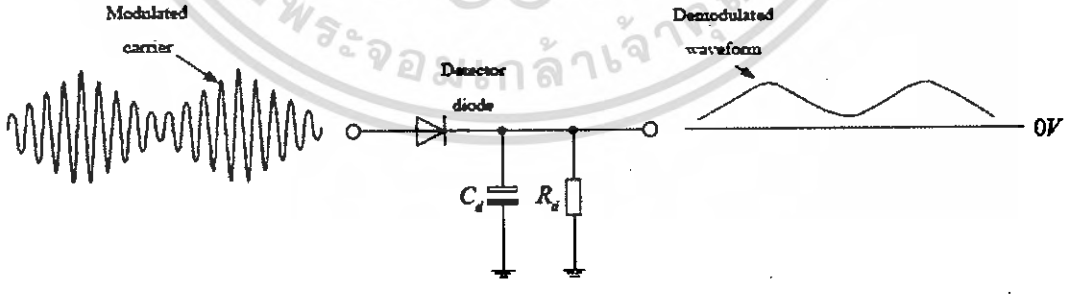
ค่าสูงสุดของดัชนีการมอดูเลตเท่ากับหนึ่ง หากคิดเป็นเปอร์เซ็นต์จะไม่เกิน 100 %

การมอดูเลตเชิงขนาดหากพิจารณาในเรื่องของไซด์แบนด์ที่แสดงใน รูปที่ 2.2b) นั้นจะพบว่าเกิดเป็นสเปกตรัมของไซด์แบนด์ด้านต่ำที่ตำแหน่ง f_c-f_a และสเปกตรัมของไซด์แบนด์ด้านสูงจะปรากฏที่ตำแหน่ง (f_c-f_a) จากไซด์แบนด์ทั้งสองสามารถหาแบนวิดท์ของการมอดูเลตเชิงขนาดซึ่งมีค่าเท่ากับ

$$f_u - f_L = 2f_a$$

ในการส่งสัญญาณวิทยุระบบAMโดยกำหนดให้มีความถี่ในการมอดูเลตสูงสุด5 kHz และมีแบนวิดท์10kHz ข้อดีของการมอดูเลตเชิงขนาดก็คือความง่ายในการผสมสัญญาณ ข้อเสียก็คือมีการรบกวนของสัญญาณอื่นๆได้ง่าย และใช้กำลังส่งสัญญาณมาก

2) <u>การดีมอดูเลตเชิงขนาด</u> คือ การคืนรูปสัญญาณข่าวสารจากระบบการส่งสัญญาณข่าวสารแบบAM จะทำการแยกสัญญาณข่าวสารออกจากสัญญาณพาห์ที่ส่งมาด้วย โดยจะทำการกรองเอาแต่เฉพาะสัญญาณที่เป็นข่าวสารออกมา ส่วนสัญญาณพาห์ที่เป็นสัญญาณความถี่สูงนั้นจะไม่สามารถผ่านออกมาได้ ซึ่งคุณสมบัติที่กล่าวมานี้เป็นคุณสมบัติของวงจรกรองความถี่ต่ำผ่าน ดังตัวอย่างของวงจรดีมอดูเลตตามรูปที่ 2.3 แสดงวงจรพื้นฐานของวงจรดีมอดูเลตเชิงขนาดไว้ดังนี้



รูปที่2.3 การดีมอดูเลตเชิงขนาด

ไดโอดจะกรองสัญญาณซีกลบทิ้ง ส่วนตัวเก็บประจุและตัวต้านทานจะประกอบกันเป็นวงจรกรองความถี่ต่ำผ่านจะกำจัดสัญญาณพาห์ส่วนที่เหลือออกจากสัญญาณข่าวสารจะเหลือแต่ส่วนที่เป็นสัญญาณข่าวสารเท่านั้นที่ตกคร่อม R อยู่

ในการเลือกใช้ค่า C_d และ R_d นั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณข่าวสารที่ตอบสนอง ค่า C_d และR_d มากสามารถตอบสนองด้วยค่าความถี่สูงๆได้ดี ค่า C_d หรือ R_d น้อยสัญญาณตอบสนองที่เอาท์พุทจะต่ำในการเลือกใช้จะต้องพิจารณาตามความเหมาะสมในการใช้งานด้วย

3) การมอดูเลตเชิงความถี่ (FM) เป็นกระบวนการมอดูเลตที่มีความซับซ้อนและมีแบนด์วิดท์ กว้างกว่าการมอดูเลตเชิงขนาดแต่การมอดูเลตเชิงความถี่ได้มีการปรับปรุงจากข้อเสียต่างๆของการส่งสัญญาณแบบAM ให้มีประสิทธิภาพในการส่งสัญญาณได้ดียิ่งขึ้น โดยที่การส่งสัญญาณแบบ FM สามารถลดสัญญาณรบกวนที่เกิดกับระบบการสื่อสารได้

การมอดูเลตเชิงความถี่ คือ กระบวนการมอดูเลตที่ทำให้ความถี่ของสัญญาณพาห์จะมีการเปลี่ยนแปลงตามความถี่ที่เข้ามามอดูเลต โดยอัตราการเปลี่ยนแปลงความถี่ Δf_c ของสัญญาณพาห์จะเป็นสัดส่วนกับแอมปลิจูด(v_a) ของสัญญาณที่เข้ามามอดูเลตดังในสมการ

$$\Delta f_c = KV_a$$

K คือ ความไวในการเบี่ยงเบนความถี่ (ความถี่/V)

จากรูปที่ 2.4 แสดงสัญญาณพาห์ที่มีการเปลี่ยนแปลงความถี่ตามแอมปลิจูดของสัญญาณข่าวสารสังเกตในรูป เมื่อสัญญาณข่าวสารมีศักย์เป็นค่าบวกส่งผลให้ความถี่ของสัญญาณพาห์มีค่าสูงและความถี่ของสัญญาณจะต่ำเมื่อสัญญาณข่าวสารมีศักย์เป็นลบ ทิศทางของการเปลี่ยนแปลงความถี่จะเริ่มเมื่อสัญญาณข่าวสารเป็นบวก ความถี่ของสัญญาณพาห์จะสูงขึ้นตามกับระดับแอมปลิจูดของสัญญาณข่าวสารจนความถี่ของสัญญาณพาห์สูงสุดที่ระดับแอมปลิจูดของสัญญาณข่าวสารมีแอมปลิจูดสูงสุด และเมื่อแอมปลิจูดของสัญญาณข่าวสารเริ่มลดลงความถี่ของสัญญาณพาห์ ก็จะลดลงจนเมื่อแอมปลิจูดของสัญญาณข่าวสารถึงระดับที่ต่ำสุด ความถี่ของสัญญาณพาห์ก็จะมีค่าต่ำสุดด้วยเช่นกัน

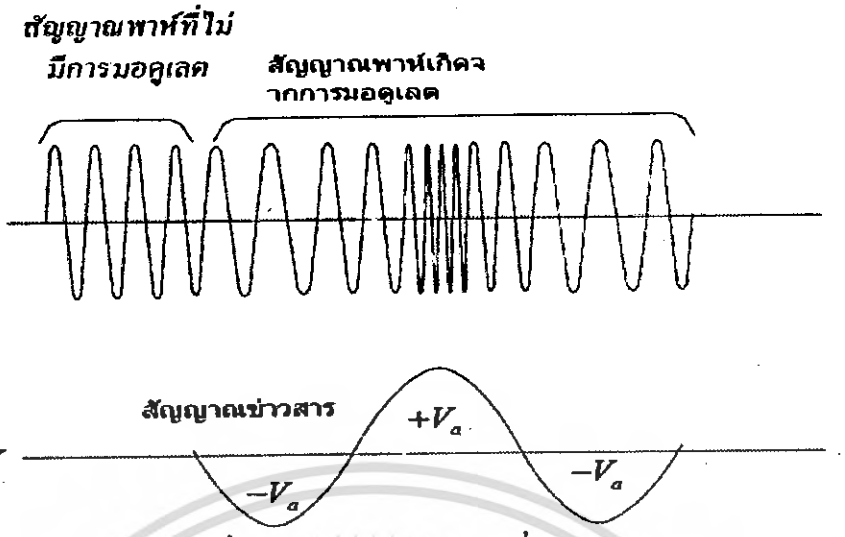
ดัชนีการมอดูเลต(m_f)ของการมอดูเลตเชิงความถี่ (FM) สามารถหาได้จากการเปลี่ยนแปลงความถี่ของสัญญาณพาห์ เกี่ยวกับ ความถี่ของสัญญาณที่เข้ามามอดูเลต ดังสมการต่อไปนี้

$$m_f = \frac{\Delta f_c}{f_a}$$

และจาก $\Delta f_c = KV_a$

$$m_f = \frac{KVa}{f_a}$$

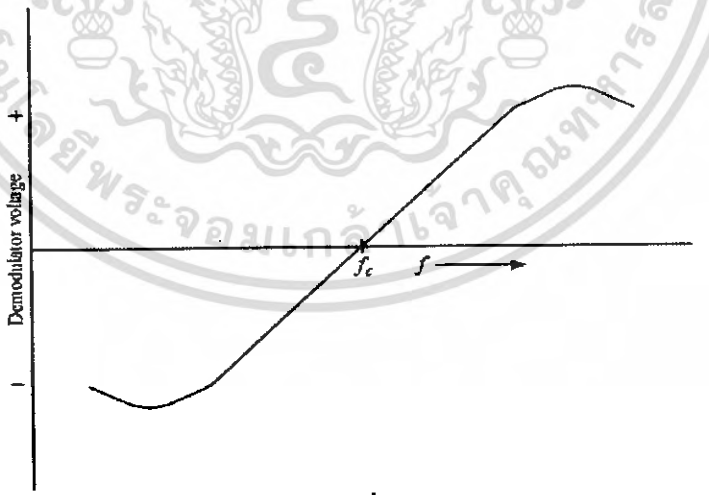
ที่สำคัญดัชนีการมอดูเลตเชิงขนาดและดัชนีการมอดูเลตเชิงความถี่ ต้องน้อยกว่าหรือมีค่าเท่ากับ 1



รูปที่2.4 การมอดูเลตเชิงความถี่

พิจารณาความถี่ไซด์แบนด์ที่เกิดจากการมอดูเลตเชิงความถี่จะมีความแตกต่างกันกับการมอดูเลตเชิงขนาด การมอดูเลตเชิงความถี่เป็นการมอดูเลตที่ใช้สำหรับสัญญาณพาห์เพียงสัญญาณเดียว สามารถทำให้เกิดความถี่ไซแบนด์ต่างๆ ขึ้นมากมาย ซึ่งความถี่ไซด์แบนด์ที่เกิดขึ้นนี้จะมีจำนวนมากหรือน้อยขึ้นอยู่กับดัชนีของการมอดูเลต (f_m)ด้วย

4) การดีมอดูเลตเชิงความถี่ การดีมอดูเลตเชิงความถี่มีอยู่ด้วยกันหลายวิธีการมาก แต่วิธีที่ง่ายที่สุดวิธีหนึ่งก็คือ แนวคิดการจับเอากรอบของสัญญาณข่าวสารออกมาจากความถี่พาหะตามแบบคลื่นรูปสัญญาณแบบ S-curve ที่แสดงในรูปที่ 2.5



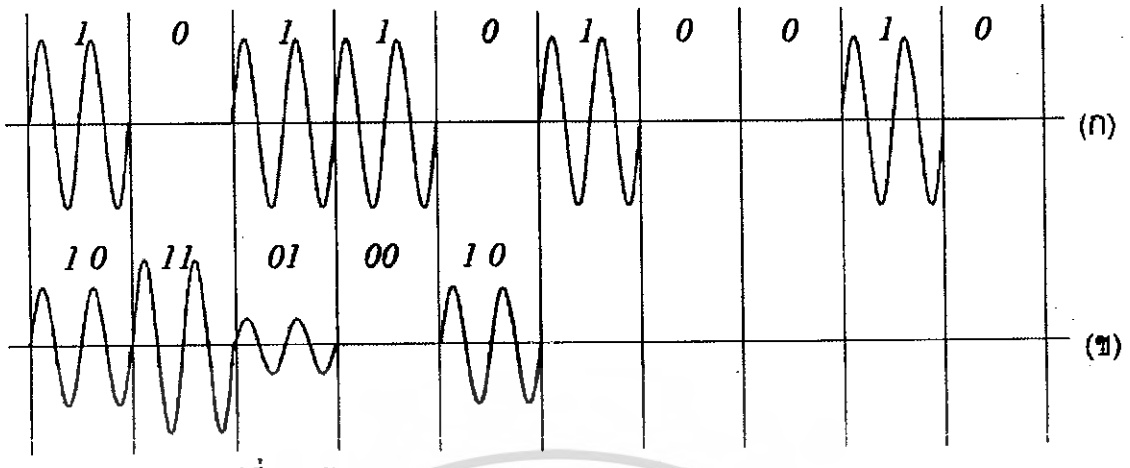
รูปที่2.5 กราฟการตอบสนองของความถี่ในวงจรกาครับรูป S-curve

รูปสัญญาณที่แสดงไว้ไม่ใช่สัญญาณข่าวสารที่ได้จากการดีมอดูเลตเชิงความถี่ แต่เป็นลักษณะการตอบสนองต่อสัญญาณที่เรียกกันว่า "S-curve" เมื่อสัญญาณ FM เข้ามาที่วงจรดีมอดูเลตเป็นช่วงที่มีความถี่ของสัญญาณสูงสัญญาณที่ได้จากการดีมอดูเลตจะได้สัญญาณที่เป็นบวกช่วงความถี่ของสัญญาณพาห์ที่มีความถี่ต่ำเมื่อดีมอดูเลตแล้วจะได้เอาต์พุตที่มีแรงดันเป็นลบ

องค์ประกอบที่สำคัญของการดีมอดูเลตนั้นก็คือ จะต้องสร้างสัญญาณพาห์ให้ตรงกับสัญญาณพาห์ที่ส่งมาจากเครื่องส่งเพื่อให้ได้สัญญาณข่าวสารที่ส่งมาอุปกรณ์ที่ทำหน้าที่คืนรูปสัญญาณข่าวสารจะรับสัญญาณที่ได้จากการมอดูเลตเชิงความถี่เข้ามาแล้วทำการเบี่ยงเบนความถี่ของสัญญาณพาห์นั้น การมอดูเลตเชิงความถี่นั้นก็คือการคืนรูปของสัญญาณข่าวสารโดยจะคืนรูปของสัญญาณข่าวสารในรูปของความชันซึ่งเป็นอัตราส่วนและความแตกต่างของสัญญาณออกมาซึ่งจะมีลักษณะเป็น S-curve ในการดีมอดูเลตเชิงความถี่ด้วยเฟสล็อกลูป จึงเป็นอีกทางเลือกที่สามารถตอบสนองต่อ S-curve ได้ดี แต่จะต้องมีการเพิ่มส่วนของวงจรจูนความถี่ให้ตรงกับความถี่ของสัญญาณพาห์ที่ส่งมาเสียก่อน ส่วนประกอบต่างๆ ภายใน วงจรเฟสล็อกลูป สามารถนำไปใช้เป็นดีมอดูเลตเชิงความถี่และสร้างเป็นวงจรสังเคราะห์ความถี่ (frequency Synthesis) ได้

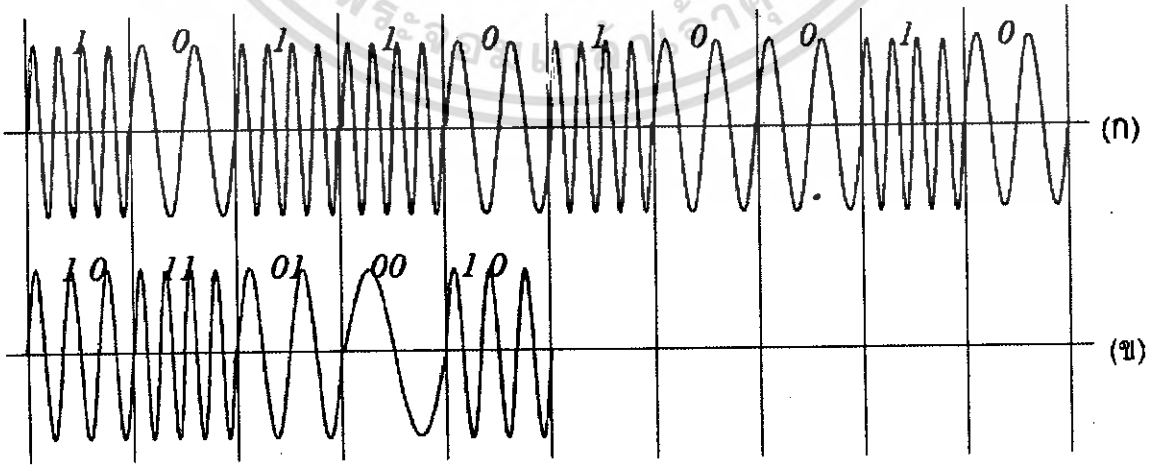
การมอดูเลตแบบดิจิตอล

1) การมอดูเลตทางขนาด (Amplitude-shift Keying:ASK) เป็นการเปลี่ยนขนาดของสัญญาณพาห์ ซึ่งการเปลี่ยนเป็นระดับที่แน่นอน เช่น การเปลี่ยนตามบิต '0' '1' จะได้สัญญาณที่แน่นอนสองระดับ เรียกว่าเป็นการมอดูเลตแบบ binary ASK ในกรณีที่ใช้ระดับสูงสุดแทนข้อมูล '1' และระดับสัญญาณพาห์เป็นศูนย์เมื่อข้อมูลเป็น '0' จะเรียกว่าเป็นการมอดูเลตแบบ on-off keying:OOK ดังแสดงในรูป 2.6a) ส่วนในรูป 2.6b) เป็นการเปลี่ยนระดับของสัญญาณพาห์เป็น 4 ระดับ ซึ่งจะทำให้สามารถแทนบิตข้อมูลได้สองบิตในแต่ละระดับของสัญญาณพาห์เรียกว่าเป็นการมอดูเลตแบบ M-array ASK โดยที่ M = 4 และจะพบว่าในรูปที่ 2.6a) กรณี binary ASK การเปลี่ยนแปลงสัญญาณพาห์แต่ละครั้ง หมายถึงมีบิตข้อมูลหนึ่งบิตถูกส่งออกไปหรือรับเข้ามา แต่ในกรณีรูป 2.6b) การเปลี่ยนแปลงสัญญาณพาห์แต่ละครั้งแสดงว่ามีบิตข้อมูลสองบิตถูกส่งออกไปหรือรับเข้ามาแสดงให้เห็นว่าจำนวนบิตข้อมูลที่เท่ากัน กรณี M-array ASK จะใช้จำนวนการเปลี่ยนแปลงสัญญาณพาห์น้อยครั้งกว่า หมายถึงการส่งได้เร็วขึ้น ซึ่งอัตราการเปลี่ยนแปลงของสัญญาณพาห์นี้จะเรียกว่า บอด (baud)นิด เช่น กรณี binary ASK แต่ถ้าหนึ่งนี้บอดเป็นอัตราการยามเป็นอัตราเร็วของสัญลักษณ์ มีความสัมพันธ์กับอัตราเร็วบิต คือ ถ้าหนึ่งสัญลักษณ์คือหนึ่งบิต อัตราเร็วบิตเท่ากับบอเปลี่ยนแปลงของสัญญาณพาห์มีค่าเท่ากันทั้งกรณี binary ASK และ 4-array ASK ถ้าส่งสัญญาณนี้ไปในช่องสัญญาณหนึ่งจะพบว่าแบนด์วิดท์ของสัญญาณทั้งสองแทบจะไม่ต่างกัน แต่ได้อัตราบิตที่ต่างกัน ด้วยวิธีนี้จึงทำให้สามารถส่งข้อมูลที่มีอัตราบิตสูงไปในช่องสัญญาณที่มีแบด์วิดท์แคบได้



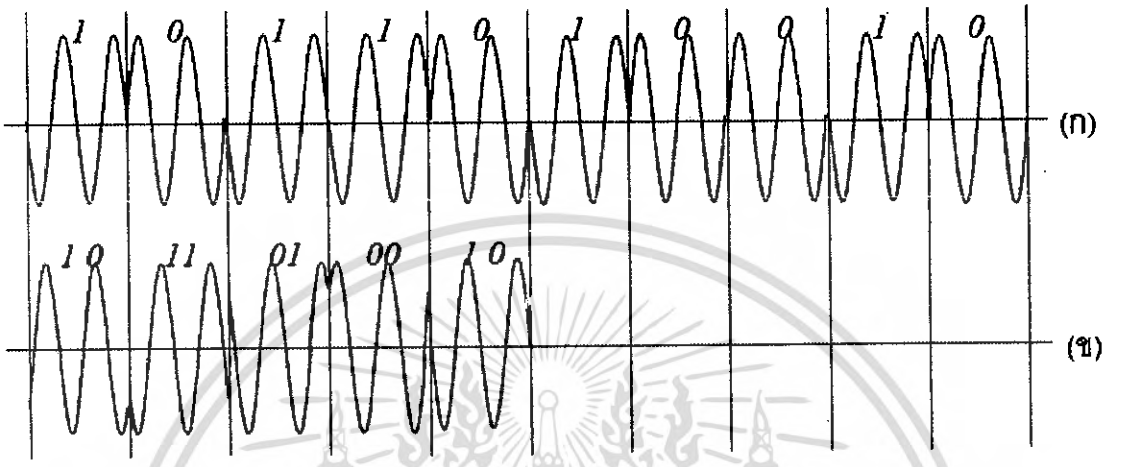
รูปที่ 2.6 สัญญาณ ASK (a) binary ASK (b) 4-Array ASK

2) การมอดูเลตทางความถี่ (Frequency-shift Keying:FSK) เป็นการเปลี่ยนความถี่ของพาห์ ทำนองเดียวกับ ASK การเปลี่ยนแปลงความถี่เนื่องจากการมอดูเลตด้วยสัญญาณดิจิตอลจะทำให้พาห์ของสัญญาณเปลี่ยนไปเป็นความถี่ที่แน่นอนในลักษณะของหารเลื่อนความถี่ ซึ่งถ้าเป็นการมอดูเลตแบบ binary FSK คือมีคอมบิเนชันของบิตข้อมูล 0 กับ 1 ก็จะทำให้สัญญาณพาห์เปลี่ยนไปเป็นความถี่สองความถี่ ดังแสดงในรูปที่ 2.7a) และเมื่อใช้หลายความถี่เพื่อแทนคอมบิเนวันของบิตข้อมูลที่มากขึ้น จะทำให้ความถี่ของพาหะเปลี่ยนไปหลายความถี่ ได้เป็นการมอดูเลตแบบ M-array FSK ดังรูปที่ 2.7b) ซึ่งเป็นกรณี 4-array FSK จะทำให้ได้อัตราบิตที่สูงขึ้นทำนองเดียวกับ M-array ASK แต่การใช้หลายความถี่ในการแทนคอมบิเนชันของบิตข้อมูลเช่นนี้ จะทำให้แบนด์วิดท์ของสัญญาณที่มอดูเลตแล้วกว้างขึ้นอย่างนี้มีนัยสำคัญ ต่างกับกรณีของ M-array ASK ที่ยังคงเป็นความถี่เดียวอยู่ ทำให้แบนด์วิดท์ของสัญญาณที่มอดูเลตแล้วไม่กว้างขึ้นอย่างมีนัยสำคัญ การมอดูเลตแบบ FSK นี้จึงมักนิยมใช้กันในการส่งสัญญาณที่มีอัตราบิตไม่สูงมากนัก เช่น 800-1200 บิต/วินาที



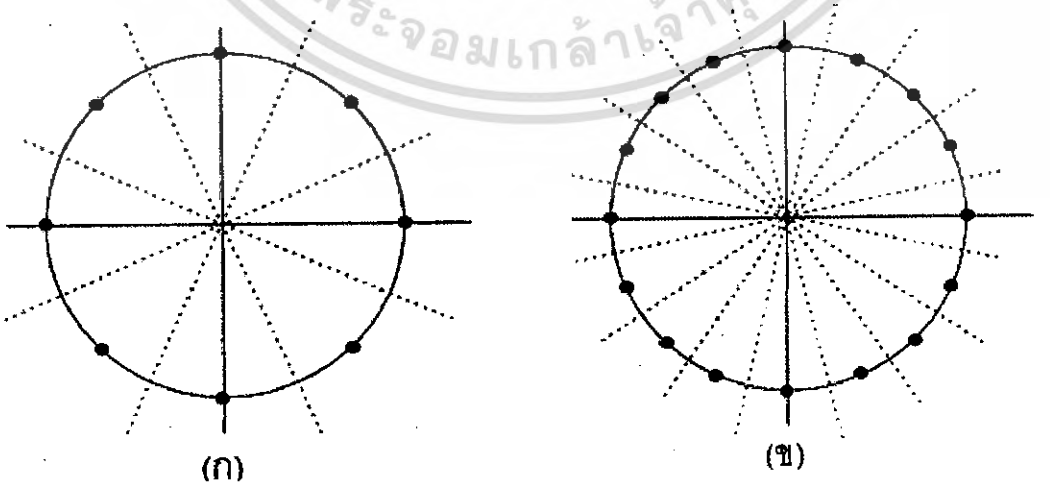
รูปที่ 2.7 สัญญาณ FSK (a) binary FSK (b) 4-Array FSK

3) การมอดูเลตทางเฟส (Phase-shift Keying:PSK) เป็นการเปลี่ยนเฟสของสัญญาณพาห์ตามการเปลี่ยนบิตข้อมูล ในกรณีของ binary PSKเป็นการเปลี่ยนเฟสของพาห์ระหว่างสองเฟสโดยจะเป็นการกลับเฟสคือให้มีความแตกต่างของเฟสมากที่สุดดังแสดงในรูปที่ 2.8a) และเมื่อมอดูเลตแบบ M-array PSK จะได้สัญญาณในรูปที่ 2.8b) ซึ่งเป็น4-array PSK



รูปที่ 2.8 สัญญาณ PSK (a) binary PSK (b) 4-Array PSK

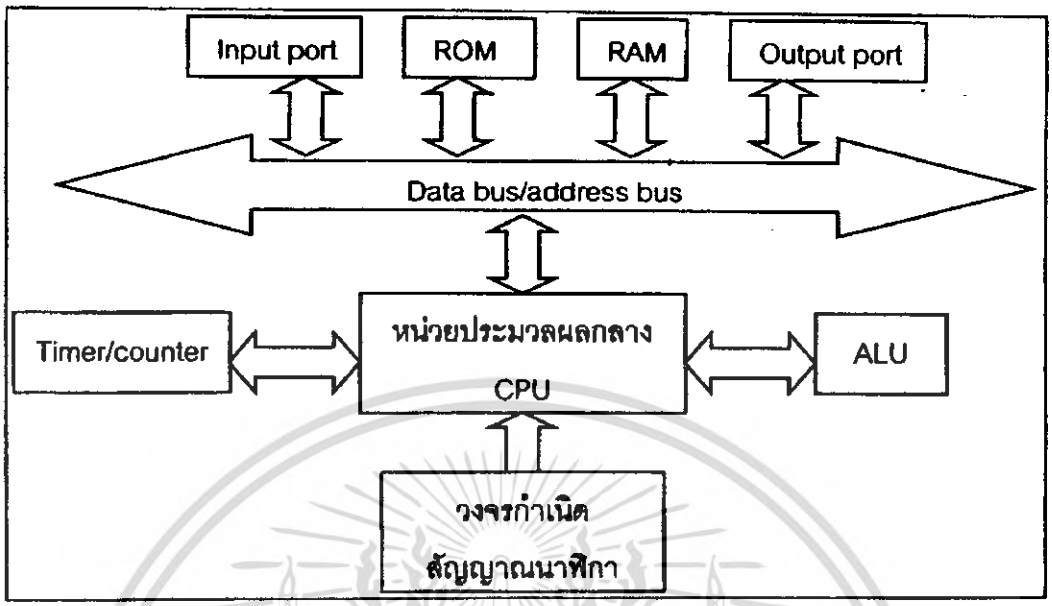
การมอดูเลตแบบ M-array PSK จะไม่ทำให้แบนด์วิดท์ของสัญญาณที่มอดูเลตแล้วกว้างมากขึ้นเมื่อ M สูงขึ้นซึ่งดีกว่าการมอดูเลตแบบ M-array FSK ทำนองเดียวกับ M-array ASK ที่เมื่อเพิ่มจำนวน M จะทำให้ความแตกต่างของแต่ละสัญญาณที่แทนคอมบิเนชันของบิตข้อมูลลดลง โอกาสที่จะเกิดการรับและตีความที่ผิดพลาดจะสูงขึ้น พิจารณารูปที่ 2.9 แสดง signal constallation ของสัญญาณ 8-array PSK และ 16-array PSK จะเห็นว่าความต่างเฟสของแต่ละสัญญาณในกรณีของ 16-array PSK จะน้อยกว่า ทำให้บริเวณที่ยอมให้มีการผิดพลาดของสัญญาณแต่ยังสามารถรับได้อย่างถูกต้องจะแคบกว่ากรณี ของ 4-array PSKมาก (ดูจากบริเวณที่แบ่งโดยเส้นประของแต่ละจุด)



Signal constallation ของสัญญาณ PSK (a) 8-Array PSK (b) 16-Array PSK

นอกจากนี้ยังมีเทคนิคการมอดูเลตสัญญาณดิจิตอลอีกหลายรูปแบบ ในแต่ละแบบก็จะมีข้อดีที่แตกต่างกันไป การมอดูเลตที่นิยมใช้กันนอกจากที่ได้กล่าวไปแล้ว เช่น Quadature Phase-shift Keying:QPSK, Minimum-shift Keying:MSK เป็นต้นการมอดูเลตและการดีมอดูเลตแบบอนาลอก ความถี่ของข้อมูลหรือสัญญาณโดยทั่วไป มักจะมีความถี่ต่ำ การแพร่กระจายของคลื่นแม่เหล็กไฟฟ้าความถี่ต่ำจะกระทำได้ไม่ดี เพราะสัญญาณความถี่ต่ำจะมีความยาวคลื่นมาก เราสามารถที่จะเลื่อนความถี่ของสัญญาณให้มีค่าสูงขึ้นได้โดยการมอดูเลตสัญญาณที่ต้องการจะส่งกับคลื่นพาห์(carrier) ความถี่สูง หรือกล่าวอีกนัยหนึ่งได้ว่าการมอดูเลตคือกระบวนการที่สัญญาณที่จะส่ง(Modulating Signal) ทำให้คุณสมบัติของคลื่นพาห์ (ขนาด ความถี่ และเฟส) เปลี่ยนแปลงไปตามสถาวะของสัญญาณ สัญญาณที่ได้จากการมอดูเลตเรียกว่า Modulated Signal (wave)

ไมโครคอนโทรลเลอร์



รูปที่ 2.9โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ทั่วไป

หน่วยประมวลผลกลาง (CPU: Central Processing Unit)

CPU เปรียบได้กับสมองของคนเรานั่นเอง เพราะการคำนวณต่างๆเกิดขึ้นที่นี่ CPU ประกอบด้วยวงจรต่างๆ หลายวงจร เช่น วงจรควบคุมเวลาและระบบการทำงาน (Timing and control Unit) ซึ่งจะทำหน้าที่จัดการทั้งหมดของวงจรทั้งประมวลผลและควบคุมตามคำสั่งที่ได้รับ การคำนวณทางคณิตศาสตร์และลอจิก (ALU : Arith me tic and Logic Unit) โดยจะทำหน้าที่คำนวณและประมวลผลทางคณิตศาสตร์และระบบลอจิก วงจรถอดรหัสคำสั่ง จทำหน้าที่แปลงคำสั่งทั้งหมดให้เป็นภาษาเครื่อง วงจรควบคุมการทำงานของ Counter (Program Counter) วงจรควบคุมสัญญาณนาฬิกา (Oscillator) ตลอดจนหน่วยความจำภายใน Register, Adder , Subtraction , Buffer และอื่นๆที่ใช้ในการเก็บและประมวลผลของ CPU เป็นต้น

หน่วยความจำ (Memory Unit)

ในการเขียนโปรแกรมด้วยภาษา C ให้กับไมโครคอนโทรลเลอร์นั้นต้องคำนึงถึงชนิดของหน่วยความจำและวิธีการเข้าถึงด้วย ซึ่งต่างจากการเขียนบน PC ที่สนใจเพียงชนิดของตัวแปรว่าจะใช้เก็บข้อมูลประเภทใด สำหรับหน่วยความจำในระบบไมโครคอนโทรลเลอร์ PIC นั้น จะมีหน่วยความจำในการใช้งาน 3 ประเภทดังนี้

หน่วยความจำโปรแกรมแบบแฟลช (FLASH Program Memory)

หน่วยความจำแบบแฟลช (Flash ROM) ในปัจจุบันนี้หน่วยความจำชนิดนี้ได้ถูกนำมาใช้กับ Microcontroller หลายบริษัท หลายรุ่น โดยมีคุรสมบัติในการเขียนโปรแกรมและลบโปรแกรมได้มากกว่า 100,000 ครั้ง ซึ่งการทำงานจะมีความเร็วสูงมากเหมาะกับการพัฒนางานที่มีขนาดใหญ่

หน่วยความจำโปรแกรม (Data Memory RAM)

หน่วยความจำส่วนนี้มีไว้ใช้เก็บข้อมูลขณะประมวลผลโปรแกรม สามารถอ่านและเขียนข้อมูลได้ขณะมีไฟเลี้ยง แต่เมื่อไม่จ่ายไฟเลี้ยงข้อมูลต่างๆจะสลายไป หากหน่วยความจำส่วนนี้ไม่พอใช้งาน จะต้องต่อหน่วยความจำแรมจากภายนอกเพิ่ม ปัจจุบันเทคโนโลยีก้าวหน้าขึ้นมากชิปบางตัวจะมีการบรรจุหน่วยความจำประเภท Data Memory เข้าไปในชิปเลย

หน่วยความจำแบบอีอีพรอม (EEPROM Data Memory)

หน่วยความจำแบบ EEPROM เป็นหน่วยความจำที่สามารถขียนและลบโปรแกรมด้วยกระแสไฟฟ้าในหน่วยความจำถาวรของ PROM (Programable Read Only Memory) โดยภายในมีการพัฒนาให้ RAM (Random Access Memory) ที่มีหน่วยความจำชั่วคราวให้เก็บข้อมูลได้ถาวรแบบหน่วยความจำ ROM (Read Only Memory) โดยสามารถเขียนโปรแกรมและลบจำนวนหลายๆครั้งได้

พอร์ตอินพุท/เอาต์พุท (I/O port)

ไมโคคอนโทรลเลอร์จะมีพอร์ตสำหรับติดต่อสื่อสารกับอุปกรณ์ภายนอกแล้วแต่วัตถุประสงค์ในการใช้งานและคุณสมบัติของพอร์ต โดยสามารถติดต่อสื่อสารกับอุปกรณ์ภายนอกทำหน้าที่เป็นอินพุทและเอาต์พุทได้ เช่น Pushbutton , Keypad ,LCD , Timer/Counter ตลอดจนการแปลงสัญญาณ Analog to Digital Converter เป็นต้น

ไมโครคอนโทรลเลอร์MCS51 รุ่น AT89C51

โครงสร้างและสถาปัตยกรรมของไมโครโทรลเลอร์ MCS-51 แบบแฟลช

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ใช้เรียนรู้ในโครงงานนี้จะอ้างอิงถึงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (flash memory) ของ Atmel Corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ในการเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1. หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2. ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์และเครื่องโปรแกรมอีพรอม
3. บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง
4. ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี
5. ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่ หรือเรียกว่า การโปรแกรมในวงจร หรือ ในระบบ (In-system programming) ทำให้การพัฒนาหรือการซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัปเกรดข้อมูลในหน่วยความจำโปรแกรมได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก
6. ชุดคำสั่งและสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ของผู้ผลิตอื่น ไม่ว่าจะเป็นอินเทล , ซีเมนส์ หรือดัลลัส

คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาท์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ในรูปที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็น

ไมโครคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว

สำหรับในรูปที่ 2-2 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบหรือเรียกว่า การใช้โปรแกรมในวงจร ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตที่เพิ่มเติมเข้ามาอีกหนึ่งตัวเป็นไทเมอร์ 2 และวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู

ในตารางที่ 2-1 แสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แต่ละเบอร์ที่ Atmel ผลิตขึ้น และมีใช้งานอยู่ในปัจจุบัน

การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูปที่ 2-3 และ 2-4 โดยมีรายละเอียดขั้นต้น ดังนี้

ขา V_{cc} ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 (P0.0-P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นที่สถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต 1(P1.0-P1.7) มี 8 ขา แต่ละขาสามมารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต 2 (P2.0-P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึง

มีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ต 3 (P3.0-P3.7) มี 8ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากกากรสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0

P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา T1

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยใช้ในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE/PROG (Address Latch Enable / Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของงขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆออกมา

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

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ต คือ พอร์ต 0 ถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุต

ที่พอร์ต 0 และ พอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ต และพอร์ต1 บางขานอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์MCS-51 แบบแฟลชเบอร์ใด

การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วนการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช่ในการขับสัญญาณเอาต์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของ

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

ตารางที่2 หน้าที่ของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของAtmel

พอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น "1" สามารถรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนั้น อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอกิจ "0" จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก "0"แล้ว)

การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่านดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0"ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไป ก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุต แต่ละขา(หรือแต่ละบิต) 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 ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาต์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

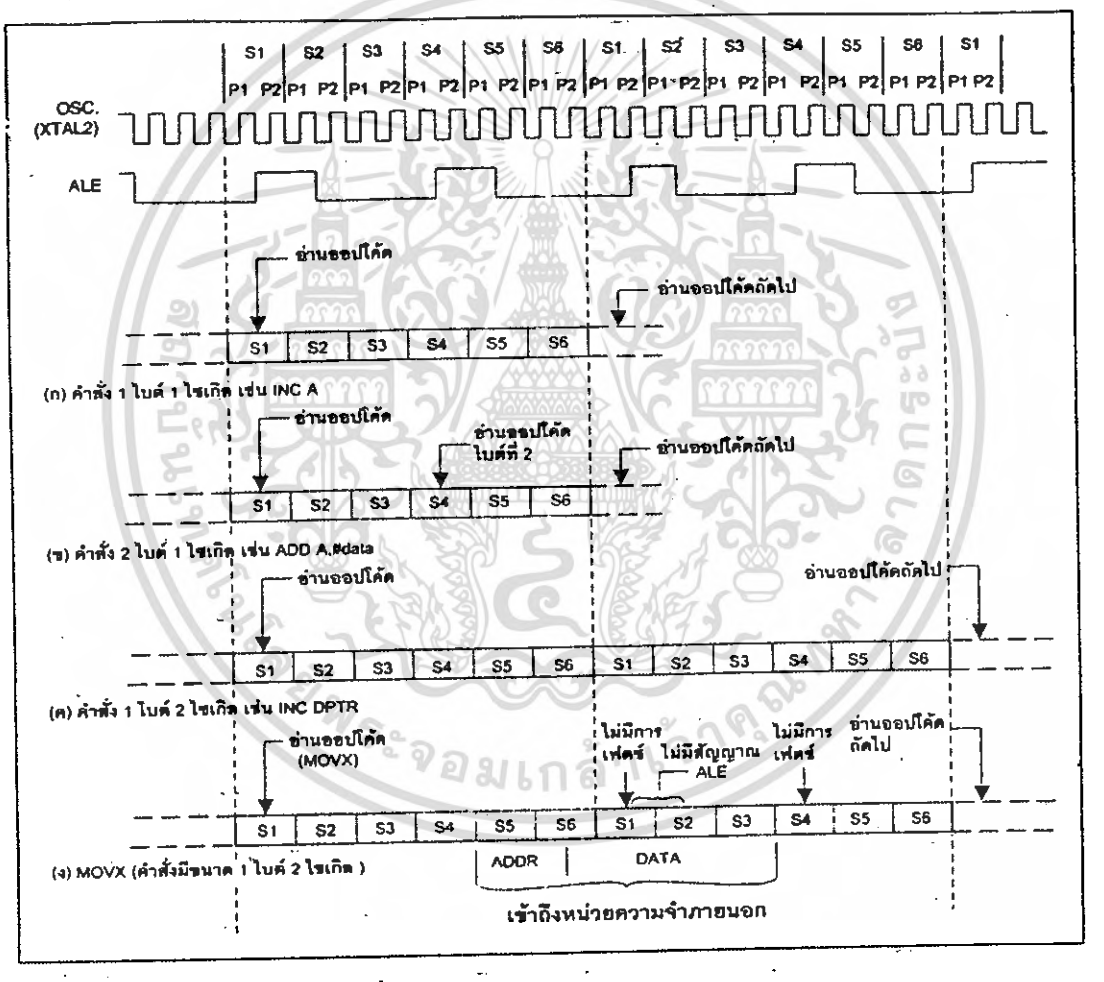
การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงานสถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเสมือนว่าขาพอร์ตนั้นถูกส่งต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะการทำงานของซีพียูและลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของซีพียูจะมีขั้นตอนหลักๆ 2 ขั้นตอนคือ กระบวนการเฟตช์ (fetch) เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผล ขั้นตอนต่อมาคือกระบวนการเอ็กซิคิวต์ (execute) เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นมาโดยกระบวนการก่อนหน้านี้ เมื่อทำการอ็กซิคิวต์คำสั่งเรียบร้อยแล้ว ก็จะไปเริ่มกระบวนการเฟตช์คำสั่งใหม่ต่อไป

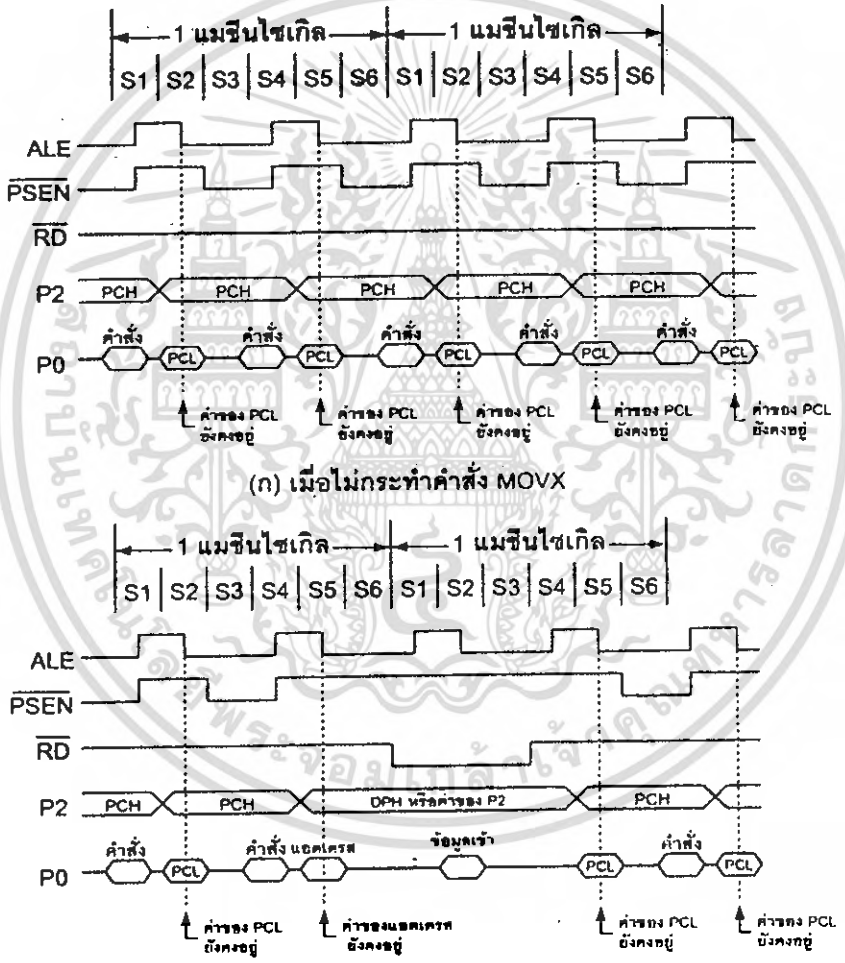


รูปที่ 2.10 ไซเกิลการทำงานของ AT89C51

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์จะเกิดการรีเซตในลักษณะที่เรียกว่า เพาเวอร์ออนรีเซต (power on reset) ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม จังหวะการทำงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากรอบการทำงานหรือแมชีนไซเกิล (machin cycle) ในรูปที่ 2.10 เป็นไดอะแกรมเวลาแสดงจังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51 โดยใน 1 รอบการทำงานหรือแมชีนไซเกิลจะแบ่งย่อยออกเป็น 6

สเตต (state) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตตมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms คาบเวลาทั้งสองภายในหนึ่งสเตตจะเรียกว่า เฟส 1 (phase) และเฟส 2 (phase 2)

ในรูปที่ 2.10 จะเป็นการเอ็กซิคิวต์คำสั่งที่ใช้เวลา 1 ไซเกิล เริ่มต้นที่สเตต 1 จะเป็นการอ่านค่าออปโค้ด อันเป็นกระบวนการแลตช์ของออปโค้ดส่งไปให้รีจิสเตอร์คำสั่ง (Instruction Register : IR) การเฟตช์ครั้งที่สองจะเกิดขึ้นที่สเตต 4 ภายในแมชีนไซเกิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตช์ครั้งที่ 2 ภายในแมชีนไซเกิลเดียวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีเวลา 1 ไซเกิล จะสิ้นสุดการทำงานลงในสเตต 6 ของแมชีนไซเกิลเดียวกัน



รูปที่ 2.11 ไดอะแกรมเวลาแสดงการติดต่อและการเข้าถึงหน่วยความจำโปรแกรม

ในกรณีที่คำสั่งใช้เวลา 2 ไซเกิล การทำงานของคำสั่งนั้นจะสิ้นสุดลงในสเตต 6 ของแมชีนไซเกิลที่สอง ดังในไดอะแกรมรูปที่ 2.10 สำหรับในการกระทำคำสั่ง MOVX นี้ ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไซเกิล จะไม่มีการเฟตช์เกิดขึ้นในไซเกิลที่สองของคำสั่ง MOVX นี้ เนื่องจาก

ซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอก ดังแสดงในไดอะแกรมรูปที่ 2-7 (ง) จะเห็นได้ว่า เวลาในการเอ็กซิคิวต์จะไม่ได้ขึ้นอยู่กับการติดต่อหน่วยความจำโปรแกรมภายในหรือภายนอก

ในรูปที่ 2.11 แสดงสัญญาณและไดอะแกรมเวลาของการเข้าถึงหน่วยความจำโปรแกรมภายนอกโดยในรูปที่ 2-8 (ก) เป็นไดอะแกรมเวลาในขณะที่ยังไม่มีการกระทำคำสั่ง MOVX สัญญาณที่ขา ALE และ PSEN จะเกิดการแอกตีฟ 2 ครั้งภายในหนึ่งแมชีนไซเกิล ในทุกครั้งที่ ALE เกิดการแอกตีฟที่พอร์ต 0 (P0) จะมีค่าของรีจิสเตอร์ PC ในไบต์ต่ำออกมา ในขณะที่พอร์ต 2 (P2) ก็จะมีค่าของ PC ในไบต์สูงเพื่อชี้ไปยังแอดเดรสต่อไปที่ต้องไปดำเนินการ สำหรับขา PSEN ก็จะเกิดการแอกตีฟเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ในกรณีที่กระทำคำสั่ง MOVX เพื่อเข้าถึงหน่วยความจำข้อมูลภายนอก ที่ขา PSEN จะไม่เกิดการแอคตีฟ 2 ครั้งภายใน 1 แมชีนไซเกิล เนื่องจากบัสแอดเดรสและบัสข้อมูลจะถูกใช้ในการติดต่อหน่วยความจำข้อมูลภายนอกแทน แต่สำหรับสัญญาณ ALE ยังคงแอกตีฟตามจังหวะการทำงานเหมือนเดิม

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

ความเร็วในการทำงานภายในไมโครคอนโทรลเลอร์เท่ากับ
ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2)/12
เวลา 1 แมชีนไซเกิล = 1/ความเร็วในการทำงานภายในไมโครคอนโทรลเลอร์

การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำภายในหลักๆอยู่ 2 ส่วนคือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์ ในที่นี้จะกล่าวถึงรายละเอียดของการจัดสรรหน่วยความจำภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช การเชื่อมต่อกับหน่วยความจำภายนอก และข้อมูลเบื้องต้นของรีจิสเตอร์ฟังก์ชั่นพิเศษที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

หน่วยความจำโปรแกรม (Program memory)

ในรูปที่ 2.12 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในเบอร์ต่างๆที่นิยมใช้งาน อันประกอบด้วยเบอร์ AT89C51 และ AT89C52 จะเห็นได้ว่า ทั้งสอง

เบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกหรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังในรูปที่ 3-1 (ก) โดยภายใน AT89C51 จะมีหน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่ AT89C52 จะมีขนาด 8 กิโลไบต์

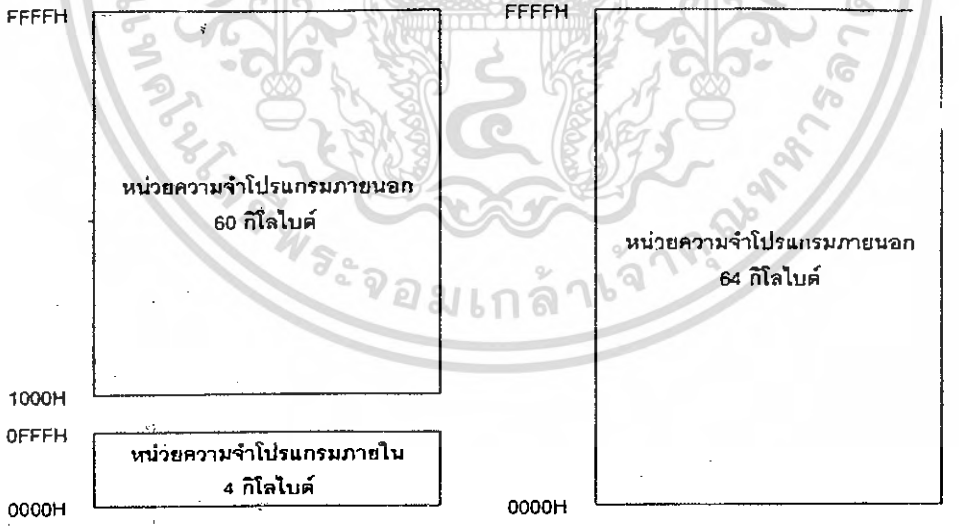
ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน หากใช้ AT89C51 ก็จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อกับหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปรแกรมใช้เก็บข้อมูลของโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ หรือที่เรียกว่า โปรแกรมมอนิเตอร์ (monitor program) หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม (EPROM : Erasable Programmable Read-Only Memory) ซึ่งสามารถทำการอ่านได้เพียงอย่างเดียว

หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซตให้เริ่มต้นการทำงานจะต้องมาเริ่มต้นที่แอดเดรส 0000H นี้เสมอ อย่างไรก็ตาม ในพื้นที่ของหน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายในหรือภายนอกก็ตาม ต้องมีการสงวนพื้นที่บางตำแหน่งเอาไว้สำหรับการบริการอินเตอร์รัปต์ 6 ประเภท ประเภทละ 8 ไบต์ ประกอบด้วย

พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้แอดเดรส 0003H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH



การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์เบอร์ AT89C51

รูปที่2.12 การจัดสรรหน่วยความจำโปรแกรม

พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH

พื้นที่สำหรับบริการอินเตอร์รัปต์ของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H

พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH

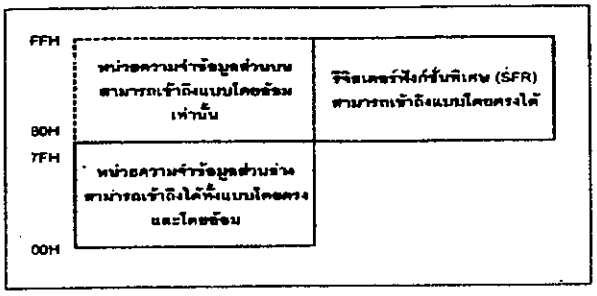
กรณีที่ใช้ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่มีหน่วยความจำภายใน แต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์ ยกตัวอย่าง ไมโครคอนโทรลเลอร์ AT89C51 มีหน่วยความจำโ)รแกรมขนาด 4 กิโลไบต์ มีแอดเดรสอยู่ระหว่าง 0000H-0FFFH เมื่อต่อหน่วยความจำโปรแรกมภายนอกต้องกำหนดให้แอดเดรสอยู่ในช่วง 1000H-FFFFH

การต่อหน่วยความจำภายนอกแสดงดังในรูปที่ 3-2 ขาพอร์ต P.0-P0.7 ใช้เป็นขาข้อมูล D0-D7 และขาแอดเดรสไบต์ต่ำ โดยผ่านวงจรแลตช์ ซึ่งปกติใช้ไอซีเบอร์ 74HC573 และใช้สัญญาณ ALE และ PSEN ในการเลือกใช้งานขา P0.0-P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเดรส ในขณะที่ขา P2.0-P2.7 ใช้ในการเชื่อมต่อกับขาแอดเดรสไบต์สูง A8-A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอกไมโครคอนโทรลเลอร์จะเหลือขาพอร์ตเพียง 16 บิต คือ ที่ขาพอร์ต P1.0-P1.7 และ P3.0-P3.7

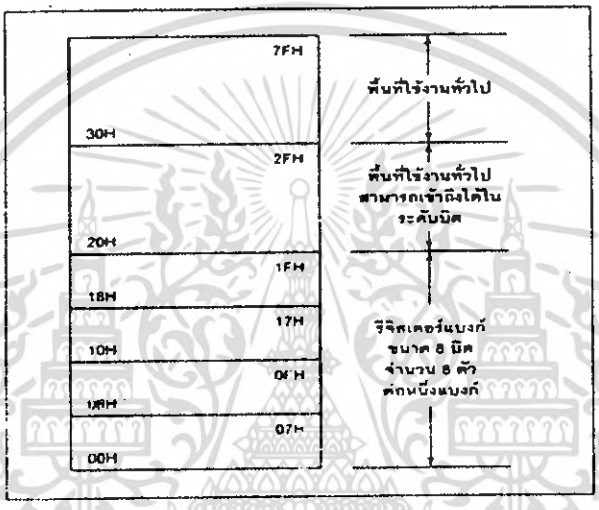
หน่วยความจำข้อมูล

มีด้วยกัน 2 แบบคือ หน่วยความจำภายนอกและภายใน โดยสามารถติดต่อหน่วยความจำภายนอกได้ถึง 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX

หน่วยความจำภายในจะเป็นแบบแรม มีขนาด 128 ไบต์ โดยจะแบ่งเป็นหน่วยความจำส่วนล่าง,ส่วนบนและรีจิสเตอร์ฟังก์ชันพิเศษ ดังแสดงดังรูป



การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช



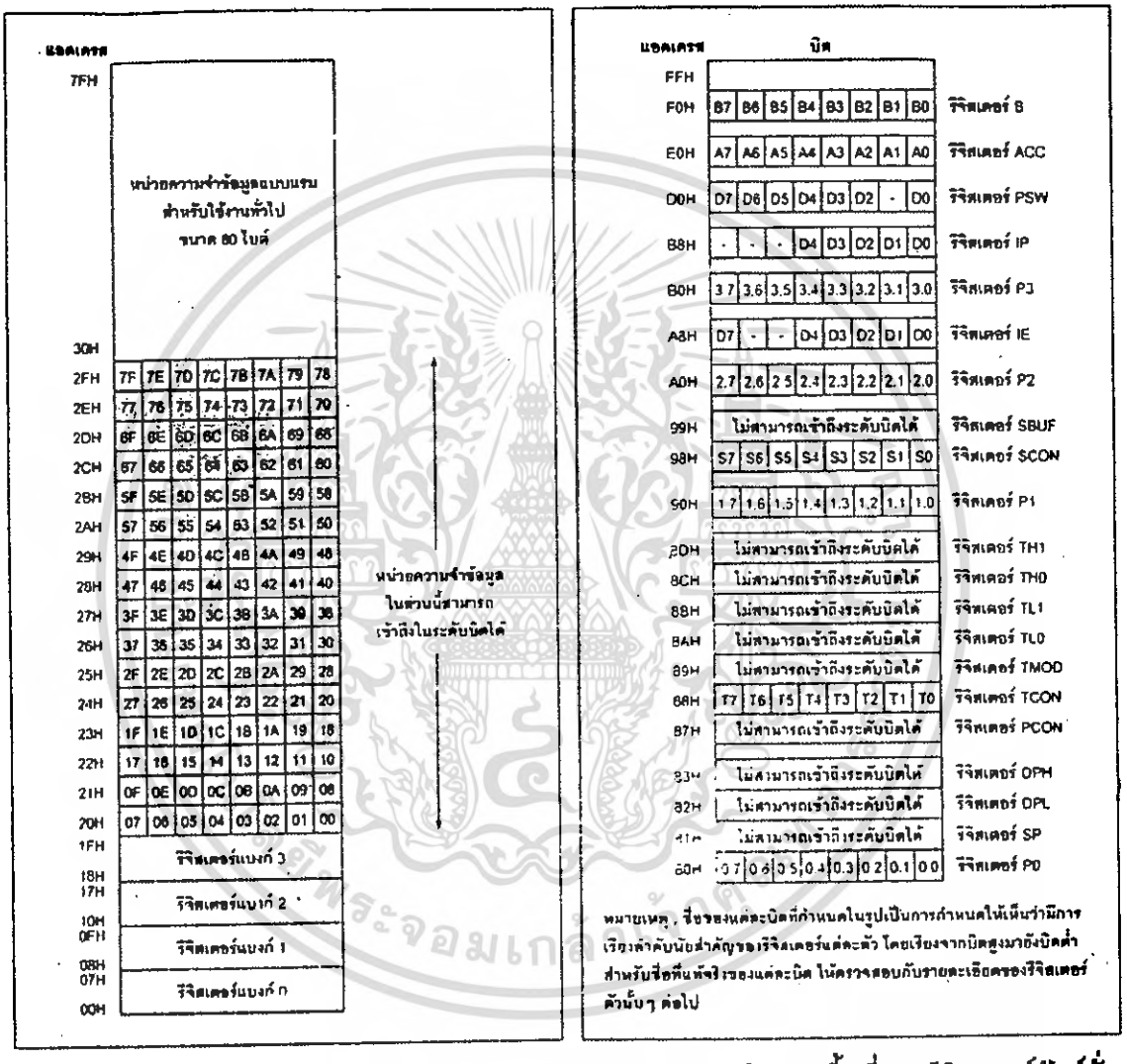
การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รูปที่ 2.13 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลส่วนล่าง

โดยจะเห็นว่าหน่วยความจำส่วนบนจะซ้อนทับกับรีจิสเตอร์ฟังก์ชันพิเศษ แต่ใช้วิธีการเข้าถึงที่แตกต่างกัน ส่วนในการจัดสรรหน่วยความจำส่วนล่าง 32ไบต์ต่ำสุดที่แอดเดรส 00H-1FH แบ่งเป็น 4 กลุ่มเรียกว่า 4 แบงก์ แต่ละแบงก์มีรีจิสเตอร์ 8ตัวคือ R0-R7 การติดต่อกับหน่วยความจำที่แบงก์ใดให้กำหนดที่รีจิสเตอร์PSW หน่วยความจำ 16 ไบต์ถัดมาที่แอดเดรส 20H – 2FH เป็นพื้นที่สำหรับใช้งานทั่วไป สามารถเข้าถึงได้ในระดับบิต และหน่วยความจำที่เหลือ 80ไบต์เก็บไว้ใช้งานสแต็ก สามรถเข้าถึงได้ในระดับไบต์

รีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Register : SFR)

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของ AT89C51 มีด้วยกัน 22 ตัว มีแอดเดรสอยู่ระหว่าง 80H – FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงได้โดยตรง ในรูป 2.14 แสดงการจัดสรรพื้นที่ของรีจิสเตอร์ SFR แต่ละตัวในหน่วยความจำส่วนบนสำหรับรายละเอียดเบื้องต้นของ



โครงสร้างของหน่วยความจำข้อมูลภายในส่วนบนของไมโครคอนโทรลเลอร์ MCS-51

การจัดสรรพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR)

รูปที่ 2.14 คือรงสร้างของหน่วยความจำข้อมูลและการจัดสรรพื้นที่ SFR

รีจิสเตอร์ SFR มีดังนี้

รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word : PSW)

เป็นรีจิสเตอร์ขนาด 8 บิตสามารถเข้าถึงได้ในระดับบิตทำหนาที่เก็บสถานะการทำงานของตปรแกรมในขณะนั้นจะเรียกสถานะต่างๆของโปรแกรมว่า flag

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
CY	AC	F0	RS1	RS0	OV	-	P

CY : แฟลกทด (Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิก แล้วค่าของแอคคิวมูเลเตอร์เกิน 255 (ฐาน10) หรือ FFH

AC : แฟลกทดเสริม (Auxillary Carry flag) เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายังบิต 4

F0 : แฟลกใช้งานทั่วไป เมื่อผู้เขียนดปรแกรมกำหนดค่าที่บิตนี้แล้วไม่ว่าจะกระทำคำสั่งใดๆที่บิตนี้จะไม่มีการเปลี่ยนแปลง

RS1 : บิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับ RS0 เพื่อเลือกรีจิสเตอร์แบงก์ของ R0-R7

RS0 : บิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับ RS1เพื่อเลือกรีจิสเตอร์แบงก์ของ R0-R7

OV : บิตเกิน (Overflow) เป็น "1" เมื่อมีกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ทำให้เกิดการทดข้ามจากบิต6 มายังบิต 7 ของแอกคิวมูเลเตอร์ หรือแอกคิวมูเลเตอร์มีค่าเกิน 127 (ฐาน 10) นอกจากนั้นยังใช้เป็นการแสดงค่าลบด้วย

- : บิตนี้ผู้ใช้สามารถกำหนดใช้งานได้อย่างอิสระ

P : บิตพาริตี้ใช้ในการตรวจสอบจำนวนค่า "1" ภายในแอกคิวมูเลเตอร์ ถ้าหากในแอกคิวมูเลเตอร์มีจำนวนบิตที่เป็น "1" รวมกันเป็นเลขคู่ บิตนี้จะเป็น "0" ถ้ารวมกันเป็นเลขคี่บิตนี้จะเป็น "1"

แอกคิวมูเลเตอร์ (Accumulator : ACC)

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์ โดยเฉพาะอย่างยิ่งในการคำนวณทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้ให้แก่ซีพียูเพื่อทำการประมวลผลต่อไป อาจเรียกสั้นๆว่า รีจิสเตอร์ A หรือ ACC รีจิสเตอร์นี้สามารถเข้าถึงระดับบิตได้

รีจิสเตอร์ B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ F0H มีหน้าที่พิเศษคือ หากต้องการคูณหรือหารทางคณิตศาสตร์ ต้องนำข้อมูลที่ทำการคูณหรือหารมาเก็บไว้ที่รีจิสเตอร์ B แล้วจึงกระทำคำสั่งการคูณหรือหารในรีจิสเตอร์ B ต่อไป

ในกรณีที่ไม่ได้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์ B นี้ในการเก็บข้อมูลทั่วไปได้เหมือนกับรีจิสเตอร์ปกติ และสามารถเข้าถึงได้ในระดับบิตได้เหมือนกับรีจิสเตอร์ A

สแต็กพอยน์เตอร์ (Stack Pointer : SP)

หรือรีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81H ใช้ในการเก็บค่าตัวชี้สแต็กซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดดไปทำงานที่โปรแกรมย่อย หรือกระดดโปรแกรม

ข่อยกลับมายังโปรแกรมหลัก เมื่อมีการรีเซตเกิดขึ้น ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H ดังนั้น แอดเดรสของพื้นที่ที่สำรองไว้ทำนห้าที่เป็นสแต็กจะเท่ากับ 08H

รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data Poiter : DPTR)

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไบต์สูง (DPH) และรีจิสเตอร์ชี้ข้อมูลไบต์ต่ำ (DPL) แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H สำหรับ DPL และ 83H สำหรับDPH รีจิสเตอร์ DPTR นี้ใช้สำหรับเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

รีจิสเตอร์พอร์ต (Port register)

เป็นรีจิสเตอร์ขนาด 8 บิตที่ใช้เก็บข้อมูลของแต่ละพอร์ตของ AT89C51 มี 4 ตัว คือพอร์ต 0 – พอร์ต 4 มีแอดเดรสคือ 80H ,90H,A0H และ B0H ตามลำดับ รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิตเมื่อต้องการขยนข้อมูลไปยังพอร์ตต่างๆของไมโครคอนโทรลเลอร์ต้อกระทำผ่านพอร์ตนี้ทุกครั้ง

รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่อยู่ใน AT89C51 โดยจะแบ่งออกเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลและรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล

รีจิสเตอร์ไทเมอร์ (Timer register)

เป็นรีจิสเตอร์ขนาด 16 บิต แบ่งเป็นข้อมูลไบต์สูงและไบต์ต่ำเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ เพื่อใช้ในการสร้างฐานเวลา,จับเวลาหรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน บางที่เรียกรีจิสเตอร์ตัวนี้ว่า รีจิสเตอร์ไทเมอร์เคาน์เตอร์

ใน AT89C51 จะมีรีจิสเตอร์ไทเทอร์เคาน์เตอร์ 2 ตัวแบ่งเป็น T0 หรือ Timer0 และ T1 หรือ Timer1 ในรีจิสเตอร์ยังแบ่งเป็นรีจิสเตอร ไทเมอร์ไบต์ต่ำ (TL) และรีจิสเตอร์ไทเมอร์ไบต์สูง (TH) เหมือนกัน โดยรีจิสเตอร์ TL0 มีแอดเดรสอยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8BH ในขณะที่ TL1 และ TH1 มีแอดเดรสอยู่ที่ 8CH และ 8DH

รีจิสเตอร์ควบคุม (Control register)

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกียวข้องกับการกำหนดอัตราการรับส่งข่อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงาน

รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรม

รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์

รีจิสเตอร์ IE และ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองอินเตอร์รัปต์ โดยIE เป็นรีจิสเตอร์สำหรับเอ็นเอเบิลหรือใช้ในการกำหนดลักษณะตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP

เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะให้ซีพียู ตอบสนองการเกิดอินเตอร์รัปต์ในลักษณะใดก่อนหรือหลัง

พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS51

AT89C51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพลกซ์ 1 ชุด โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RXD และขา P3.1 เป็นขาส่งออกข้อมูลหรือ TXD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของAT89C51 เป็นแบบอะซิงโครนัสปกติแล้วพอร์ตอนุกรมของ AT89C51 จะใช้มาตรฐาน RS-232

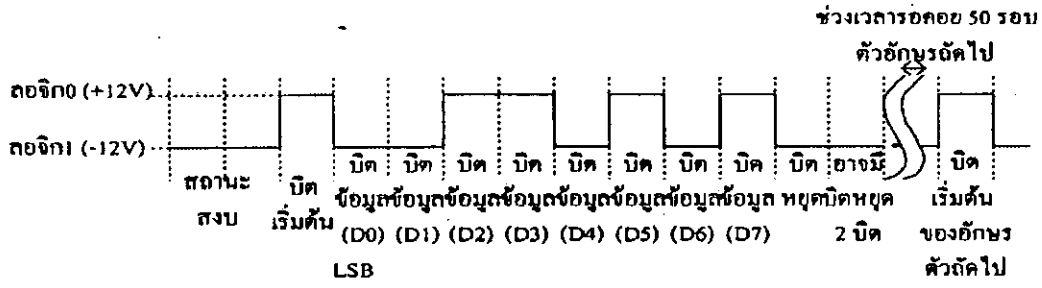
การสื่อสารข้อมูลแบ่บอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น มีขนาด 1 บิต
2. บิตข้อมูลอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด มีขนาด 1 บิต

รูปที่ 2.15 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วย บิตพาริตี้ (parity bit) เพื่อใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ บิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะเป็นลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต,1.5บิตหรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว



รูปที่ 2.15 รูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

อัตราเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่าตั้งแต่ 110 ถึง 19,200 บิตต่อวินาที

การตรวจสอบบิตพาริตี้สามารถกำหนดให้เป็นแบบคี่, แบบคู่ หรือไม่มีการตรวจสอบบิตพาริตี้ก็ได้พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งออกไป 1 ไบต์รวมพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่างเช่นข้อมูลที่จะส่งมีขนาด 8 บิต มีค่าเท่ากับ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" เป็นจำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART(Univesal Asynchronous Reciver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับต้องกำหนดการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือคู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่า เป็นคู่หรือคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อความออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวท่านั้น ถ้าข้อมูลที่ส่งมีความผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการติดตั้งพาริตี้เป็น NONE ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบพาริตี้

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน AT89C51

ในการทำงานของพอร์ตอนุกรมใน AT89C51 มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัวดังนี้

รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ มีขนาด 8 บิตแบ่งเป็น 2 ส่วน รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล และรับข้อมูล เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมายังขา RxD หรือขา P3.0 ของ AT89C51

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H สามารถเข้าถึงได้ในระดับบิตมีรายละเอียดการทำงานดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
SM0	SM0	SM2	REN	TB8	RB8	TI	RI

SM0 – SM1 (Serial port mode bit0-1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายใน AT89C5I

SM2 : ใช้ในการเอ็นเอเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมใน AT89C51 ถ้าบิตนี้เป็น "1" บิต RI จะไม่แอคตีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" ในการทำงานโหมด "1" ถ้าบิตนี้เซตบิต RI จะไม่แอคตีฟถ้ายังไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable serial reception) : ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยการะบวนการทางซอฟแวร์ ถ้าต้องการให้มีการรับข้อมูลต้องเซตบิตนี้ให้เป็น "1"

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมใน AT89C51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น "0" ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด สำหรับในการทำงานในโหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟท์แวร์

TI (Transmit Interupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของ AT89C51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟท์แวร์เท่านั้น

RI (Receive Interupt flag) : ใช้แสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรม สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟท์แวร์เท่านั้น

โหมดการทำงานของพอร์ตอนุกรมใน AT89C51

พอร์ตอนุกรมใน AT89C51 สามารถเลือกการทำงานได้ 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกบอดได้
3. โหมด 2 สามารถกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของ AT89C51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่ง คือ อัตราการถ่ายทอดข้อมูล หรือ อัตราบอด ซึ่งการกำหนดอัตาบอดนั้นจะขึ้นอยู่กับความถี่ของสัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานที่สามารถเลือกอัตราบอดได้อย่างอิสระนั้นคือ โหมด 1 และโหมด 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลวไทม์ของไทเมอร์ 1 ถ้าหากไทเมอร์ 1 มีการเกิดโอเวอร์โฟลวในอัตราที่สูงมากเท่าใด อัตราบอดก็จะมีค่าสูงมากขึ้นตาม นั่นหมายความว่า อัตราในการถ่ายทอดข้อมูลจะสูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราบอดในโหมด 1 และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือ โหมด 8 บิตแบบตั้งค่าการนับอัตโนมัติ และการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH1 จึงเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอด

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น "0" ค่าของการรีโหลดให้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด)

แต่ถ้าบิต SMOD เกิดการเซตจะเป็นการเอ็นเอเบิลการทวีคูณของอัตราบอด ดังนั้นการกำหมดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

โมดูลLCD

รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะประกอบด้วย 3 ส่วนหลักๆดังนี้

ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิปที่นิยมใช้คือเบอร์ HD44780 และ HD61830 โดยHD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุมแบบกราฟิก

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

โดยโมดูล LCD แบบอักขระจะประกอบด้วย

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-table) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

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

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บตัวอักษรที่มีการสร้างเพิ่มใหม่ ในกาณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

RS	R/W	E	การทำงาน
0	0	พัลส์ขาลง	เขียนคำสั่ง
0	1	พัลส์	อ่านสถานะของโมดูลLCD
1	0	พัลส์ขาลง	เขียนข้อมูล
1	1	พัลส์	อ่านข้อมูล

ตารางที่3 ตารางแสดงความสัมพันธ์ของขา RS,R/W และ E

โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด

โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด รายละเอียดการทำงานแต่ละขามีดังต่อไปนี้

Vss (ขา1) : ต่อกราวด์

Vdd (ขา2) : ต่อไฟเลี้ยง 5 โวลต์

Vo (ขา3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น "0" ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) : เป็นขาที่ใช้ในการเลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0–D7 (ขา 7 -14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก

คำสั่งควบคุมโมดูล LCD

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

1.คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้ในเขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D ให้เป็น "1"

2.คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอเซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลของจอแสดงผลไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03 ก็ได้

3.คำสั่งเลือกโหมดการป้อนข้อมูล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	0	1	I/D	S

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

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้นข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรส DDRAM เพิ่มขึ้น

4.ควบคุมการแสดงผล

มีรายละเอียดคำสั่งข้อมูลดังนี้

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 ใช้ควบคุมการแสดงเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้ทีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์หือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H – 0FH ที่ใช้บ่อยคือ 0CH เป็นคำสั่งให้เปิดจอแสดงผลแต่ไปแสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์ และให้กระพริบ

5.คำสั่งการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

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	1CH – 1FH

6.คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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 “1” จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด ก็กำหนดให้บิต N นี้ให้เป็น “1” จุดที่น่าสังเกต คือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น “1” เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น “1” จะเป็นการแสดงผลแบบ 5x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCDทำงานในแบบ 8 บิต แสดงผล 2 บรรทัดและเลือกความละเอียดเป็น 5x7 จุด

7.คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต7 เป็น "0" บิต6 เป็น "1" ส่วนอีก6บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H-3FH.

8.คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต7 ต้องเป็น "1" และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH – 0FFH ทั้งนี้จำนวนแอดเดรยังขึ้นกับการกำหนดสถานะที่บิต N ด้วยหากบิต N เป็น "0" แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H – 0CFH และถ้าบิต N เป็น "1" แอดเดรสของ DDRAM จะมี2ช่วง คือ 8CH – 87H และ 0C0H – 0C7H

9.คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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 พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วยโดยบิต 0-บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

ระบบสื่อสารข้อมูลอนุกรมแบบหนึ่งสาย(1-wire serial bus)

ระบบสื่อสารแบบนี้จะใช้สายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่นๆ เนื่องจากสายข้อมูลนั้นจะทำหน้าที่เสมือนหนึ่งเป็นสายสัญญาณนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาหรือต่อไปนี้จะขอเรียกว่า ไทม์สลอต โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆ ในการสื่อสารข้อมูลในแต่ละไทม์สลอต มีการกำหนดขอบเขตไว้อย่างชัดเจนการถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละไทม์สลอตนั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิต ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ ระบบสื่อสารแบบนี้เหมาะที่จะนำมาใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน หรือสร้างเป็นโครงข่ายสื่อสารแบบทวิสต์แพร์ก็ได้

คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย

สายสัญญาณบนระบบบัสแบบหนึ่งสายนี้จะเป็นสายสัญญาณแบบสองทิศทาง แต่ข้อมูลจะสามารถเดินทางได้ในทิศทางเดียวภายในช่วงเวลาหนึ่งๆนั่นคือ มีลักษณะคล้ายกับการสื่อสารแบบ Half duplex ตัวอย่างที่เห็นได้ชัดคือ การใช้งานวิทยุสื่อสารหรือวิทยุสมัครเล่น อุปกรณ์บนระบบบัสต้องมีการระบุอย่างชัดเจนว่าตัวใดเป็นอุปกรณ์มาสเตอร์ ตัวใดเป็นอุปกรณ์สเลฟ โดยส่วนใหญ่อุปกรณ์มาสเตอร์เป็นไมโครคอนโทรลเลอร์ ส่วนอุปกรณ์สเลฟได้แก่ ไอซีตรวจวัดอุณหภูมิ ไอซีหน่วยความจำแรม เป็นต้น อุปกรณ์มาสเตอร์จะเป็นตัวจัดเตรียมความพร้อมของสายสัญญาณและควบคุมการถ่ายทอดข้อมูลบนสายสัญญาณนั้น ข้อมูลทั้งหมดไม่ว่าจะเป็นข้อมูลควบคุมหรือข้อมูลใช้งานจะถูกส่งลงบนสายสัญญาณนี้ทั้งหมด ในระหว่างการทำงานอุปกรณ์มาสเตอร์และสเลฟสามารถเป็นได้ทั้งตัวส่งและตัวรับ ขึ้นอยู่กับเงื่อนไขของการทำงานในขณะนั้น

สายสัญญาณของระบบบัสนี้ต้องกำหนดสภาวะปกติไว้ที่ลอจิกสูง สามารถทำได้โดยการต่อตัวต้านทานค่าประมาณ 4.7 กิโลโอห์ม พูลอัพกับไฟเลี้ยง +5 V ดังนั้นอุปกรณ์ที่นำเข้ามาต่อบนระบบบัสนี้ต้องออกแบบให้ภาคเอาต์พุตที่ต้องต่อกับสายสัญญาณมีลักษณะเป็นคอลเล็กเตอร์เปิดหรือเดรนเปิด

คุณสมบัติของไทม์สล็อต

อุปกรณ์มาสเตอร์จะเป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสหนึ่งสายนี้ที่สามารถทำการอินนิเชียลสายสัญญาณได้ โดยอุปกรณ์มาสเตอร์จะกำเนิดจุดเริ่มต้นของไทม์สล็อตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลาหนึ่ง จากนั้นก็จะทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์ อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะของสายสัญญาณต่อไป จนเสร็จสิ้นกระบวนการ แต่ถ้าหากอุปกรณ์มาสเตอร์ต้องการส่งข้อมูลก็จะสามารถดำเนินการต่อไปได้เลย

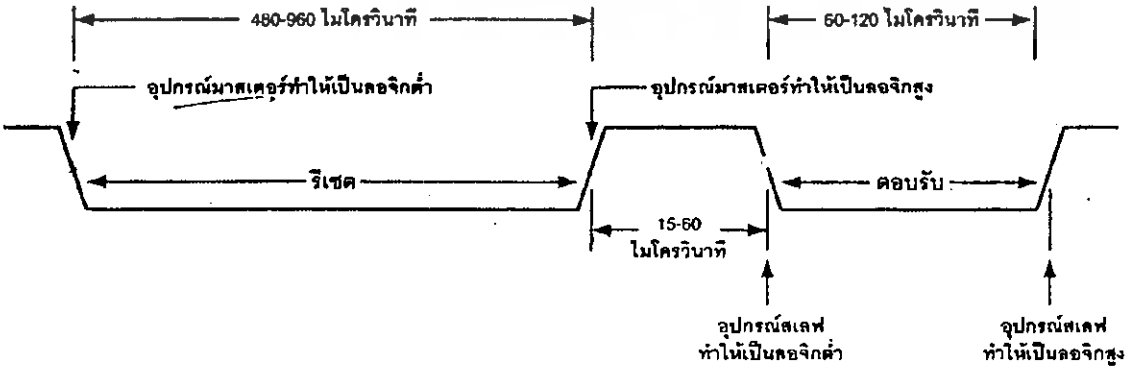
ฟังก์ชันของไทม์สล็อตที่กำหนดโดยอุปกรณ์มาสเตอร์มีด้วยกัน 4 ฟังก์ชัน คือ ไทม์สล็อตของการรีเซต (RESET) ,การอ่านข้อมูล(READ DATA), การเขียนข้อมูล "1" (WRITE ONE) และการเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตรีเซตใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟ ในขณะที่ไทม์สล็อตการอ่านจะสำหรับอ่านข้อมูลที่ส่งออกมาจากอุปกรณ์สเลฟ ส่วนไทม์สล็อตการเขียนข้อมูล "1"และ "0" ใช้สำหรับเขียนข้อมูลไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ

ทางด้านอุปกรณ์สเลฟมีฟังก์ชันของไทม์สล็อตอยู่ทั้งสิ้น 3 ฟังก์ชันคือ ไทม์สล็อตของการตอบสนอง (PRESENCE), การเขียนข้อมูล "1" (WRITE ONE) และการเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตของการตอบสนองใช้สำหรับตอบสนองการติดต่อจากอุปกรณ์มาสเตอร์ โดยอุปกรณ์สเลฟตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณเพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า ขณะนี้สามารถติดต่อกันได้แล้ว ส่วนไทม์สล็อตการเขียนข้อมูล "1" และ "0" ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์มาสเตอร์ผ่านทางสายสัญญาณของระบบ ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การแยกแยะฟังก์ชันของแต่ละไทม์สล็อตจะใช้ความยาวของคาบเวลาและลักษณะของรูปสัญญาณเป็นตัวกำหนด และทุกครั้งที่มีการเปลี่ยนแปลงฟังก์ชันต้องทำให้สายสัญญาณอยู่ในสภาวะว่างเสมอ ซึ่งก็คือการทำให้สายสัญญาณเป็นลอจิกสูงอย่างน้อยเป็นเวลา 1 ไมโครวินาที

ไทม์สล็อตของการรีเซตและตอบสนอง

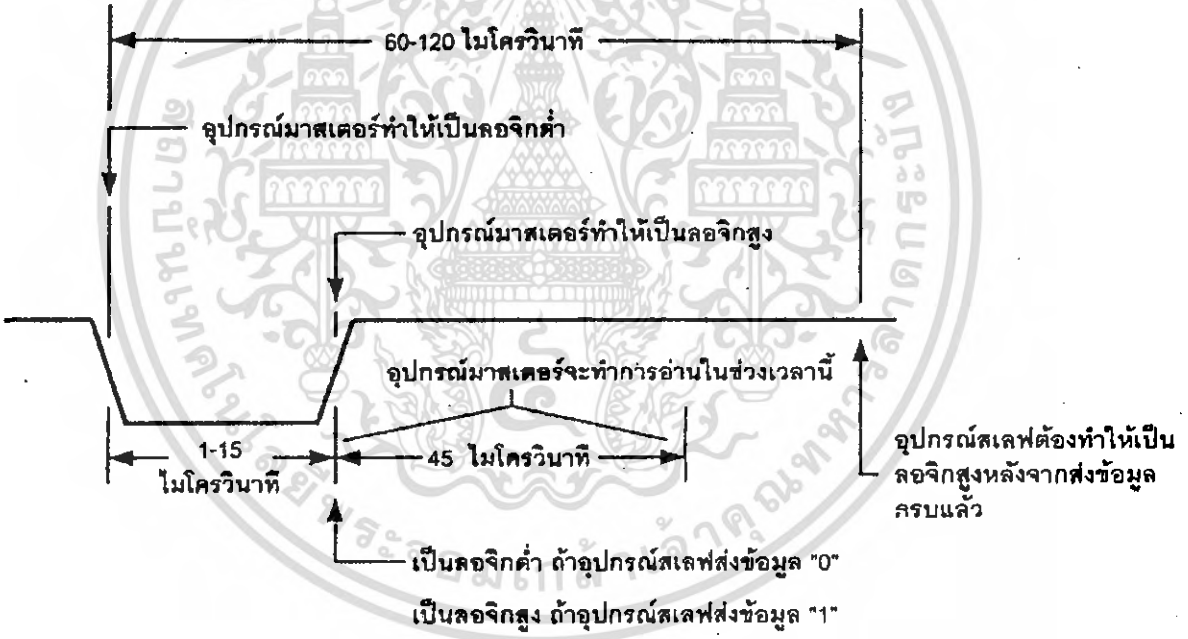
อุปกรณ์มาสเตอร์ทำให้เกิดการรีเซตบนสายสัญญาณเพื่อแจ้งแก่อุปกรณ์สเลฟ โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องอย่างน้อยเป็นเวลา 480 ไมโครวินาที และจะต้องทำให้สายสัญญาณกลับมาเป็นลอจิกสูงภายใน 480 ไมโครวินาทีหลังจากนั้น ถ้าหากมีอุปกรณ์สเลฟต่ออยู่บนสายสัญญาณ จะมีการตอบสนองสัญญาณรีเซตนั้นด้วยสายสัญญาณตอบสนอง (PRESENCE) โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องนานประมาณ 60-240 ไมโครวินาที หลังจากสัญญาณรีเซตปรากฏประมาณ 15 -60 ไมโครวินาที ในรูปที่2.16แสดงไทม์สล็อตของการรีเซตและการตอบสนอง



รูปที่2.16ไทม์สล็อตของการรีเซตและตอบสนอง

ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

เมื่อต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1- 15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายสัญญาณกลับมาเป็นลอจิกสูง อุปกรณ์สเลฟจะส่งข้อมูลมาให้อุปกรณ์มาสเตอร์โดยถ้าข้อมูลเป็น "0" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกต่ำนานประมาณ 45 ไมโครวินาทีแล้วทำให้สายสัญญาณกลับมาสู่สถาวะลอจิกสูงอีกครั้ง แต่ถ้าข้อมูลเป็น "1" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที นั่นคือในไทม์สล็อตนี้จะต้องใช้เวลารวมไม่เกิน 120 ไมโครวินาที ในขณะที่อุปกรณ์มาสเตอร์จะใช้เวลาในการอ่านข้อมูลอยู่ระหว่าง 15 และ 60 ไมโครวินาทีหลังจากเริ่มต้นไทม์สล็อตนี้ ในรูปที่ 2.17 แสดงรูปสัญญาณของไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการเขียนข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์อ่าน อุปกรณ์สเลฟก็ต้องทำการเขียน

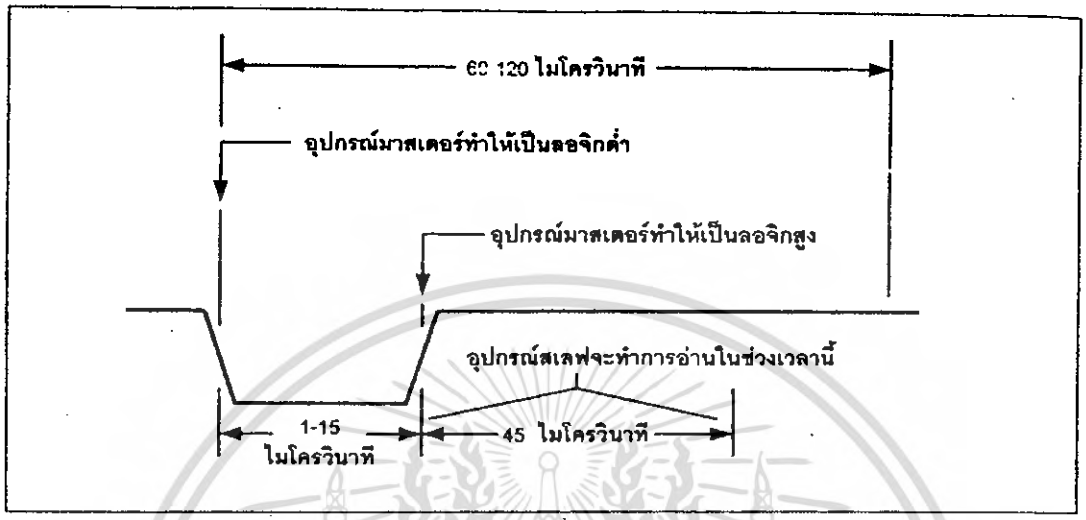


รูปที่ 2.17 ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

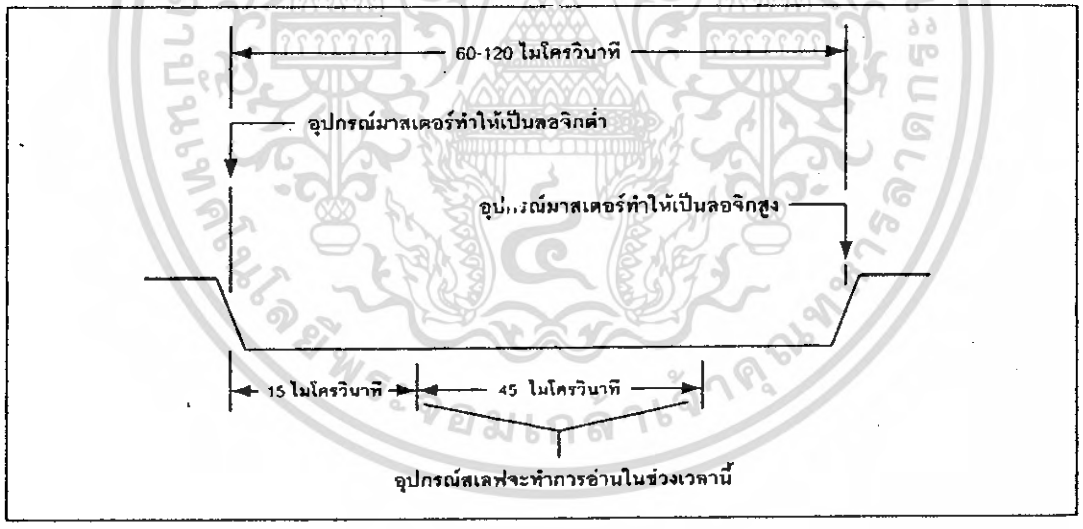
ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์

เมื่ออุปกรณ์มาสเตอร์ต้องการเขียนข้อมูล อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1 – 15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง แล้วดำเนินการเขียนได้ในทันที ถ้าข้อมูลที่ต้องการเขียนไปยังอุปกรณ์สเลฟเป็น "0" อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำนานประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สถาวะลอจิกสูงอีกครั้ง แต่ถ้าต้องการเขียนข้อมูล "1" อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีกประมาณ 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60 -

120 ไมโครวินาที ในรูปที่ แสดงรูปสัญญาณของไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการอ่านข้อมูลของอุปกรณ์สเลฟ ลไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์เขียน อุปกรณ์สเลฟก็ต้องทำการอ่านข้อมูล



ไทม์สล็อตการเขียนข้อมูล "1" ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์สเลฟ



ไทม์สล็อตการเขียนข้อมูล "0" ของอุปกรณ์มาสเตอร์

รูปที่ 2.18 ไทม์สล็อตการเขียนข้อมูล1 และ 0 ของอุปกรณ์มาสเตอร์

รูปแบบการสื่อสารข้อมูลแบบหนึ่งสาย (1-WIRE™ Communication Protocol)

ในการติดต่อสื่อสารข้อมูลในแบบบัสหนึ่งสาย อุปกรณ์มาสเตอร์ก็จะสามารถติดต่อกับอุปกรณ์สเลฟได้ครั้งละหนึ่งตัวเท่านั้น ดังนั้นอุปกรณ์สเลฟแต่ละตัวต้องมีข้อมูลกำหนดแอดเดรสเฉพาะตัวโดยเก็บไว้ในหน่วยความจำรอมในอุปกรณ์สเลฟตัวนั้น โดยปกติอุปกรณ์สเลฟในระบบ

บัสหนึ่งสายของ DALLAS นี้จะมีหน่วยความจำขนาด 64 บิต หรือ 8 ไบต์ สำหรับเก็บข้อมูลต่างๆ ที่สำคัญของอุปกรณ์แต่ละตัว ซึ่งประกอบด้วย

1. รหัสของตระกูลจำนวน 8 บิต
2. เลขหมายประจำตัว (Serial number) จำนวน 48 บิต
3. รหัสตรวจสอบความผิดพลาด (CRC : Cyclical Reduncy Cheek) จำนวน 8 บิต
4. ผู้ใช้สามารถอ่านข้อมูลประจำตัวของอุปกรณ์สเลฟได้ ด้วยการใช้คำสั่งอ่านหน่วยความจำรอม (READ ROM) ในกรณีที่สายสัญญาณมีอุปกรณ์สเลฟเพียงตัวเดียว ไม่จำเป็นต้องอ้างแอดเดรสในการติดต่อ

รูปแบบการติดต่อบนสายระบบบัสหนึ่งสายจะเริ่มต้นขึ้นเมื่ออุปกรณ์มาสเตอร์ทำการรีเซต และกำหนดแอดเดรสของอุปกรณ์ที่ทำการติดต่อกับหน่วยความจำรอมหรือสคริปรอม (Skip ROM) จากนั้นรอการตอบรับจากอุปกรณ์สเลฟเมื่ออุปกรณ์ตอบรับสมบูรณ์ก็จะสามารถเริ่มต้น ขั้นตอนการอ่านหรือเขียนข้อมูลต่อไป

การกำหนดค่าเวลาสำหรับติดต่อกับอุปกรณ์บนระบบบัส 1 สาย

ในการทำงานบนบัสหนึ่งสายนี้ของไมโครคอนโทรลเลอร์ MCS-51 จะต้องเขียนโปรแกรมเพื่อสร้างสัญญาณให้มีรูปแบบตรงกับข้อกำหนดการสื่อสารชนิดนี้ที่เรียกว่าไทม์สล็อต ซึ่งจะต้องมีการหน่วงเวลาเป็นช่วงสั้นๆ เพื่อให้ตรงกับเงื่อนไขในการสื่อสาร เนื่องจากในการเขียนโปรแกรมภาษา C นั้น คำสั่งแต่ละคำสั่งจะใช้เวลามาน้อยแตกต่างกันซึ่งไม่อาจจะคาดเดาได้ นอกจากจะดูจากไฟล์ลิสติ้ง (Listing file) ที่ C คอมไพเลอร์สร้างขึ้นจากการแปลงคำสั่งภาษา C เป็นภาษาแอสเซมบลี ซึ่งแต่ละคำสั่งของภาษาแอสเซมบลีจะใช้เวลาที่แน่นอน ซึ่งเป็นจำนวนหน่วยของแมชชีนไซเคิล โดยที่

เวลา 1 แมชชีนไซเกิล = $12/f_{XTAL}$ สำหรับไมโครคอนโทรลเลอร์ MCD-51 มาตรฐานทั่วไป

เวลา 1 แมชชีนไซเกิล = $6/f_{XTAL}$ สำหรับไมโครคอนโทรลเลอร์ MCD-51 เบอร์ P89C51RD2HBP และ P89C51RD2BN เมื่อเลือกโหมด 6 สัญญาณนาฬิกาต่อไซเคิล

สำหรับการวิจัยในปริญญานิพนธ์นี้ จะใช้ไมโครคอนโทรลเลอร์ MCD-51 เบอร์ P89C51RD2HBP ที่เลือกโหมด 6 สัญญาณนาฬิกาต่อไซเคิล ส่วนสัญญาณนาฬิกาหลักใช้คริสตอลความถี่ 11.0529 MHz

ดังนั้นเวลา 1 แมชชีนไซเกิล = $6/f_{XTAL} = 6/11.0529\times10^{6} = 0.54$ ไมโครวินาที

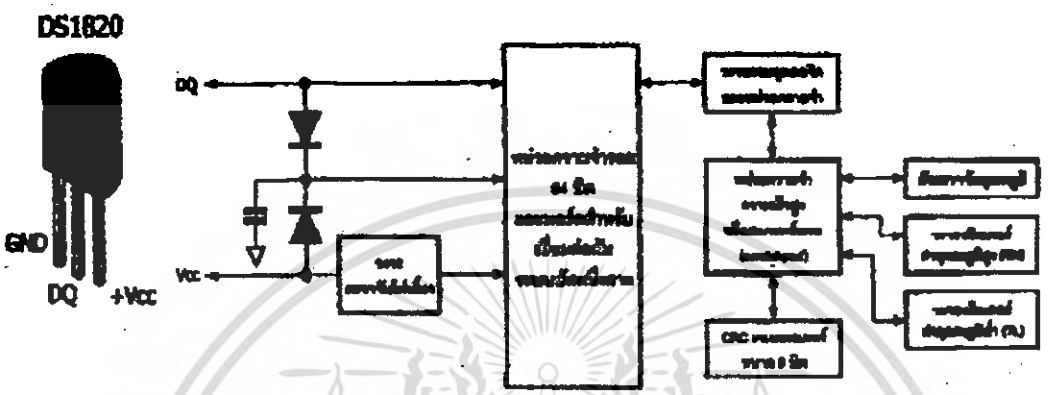
แต่ถ้าใช้ไมโครคอนโทรลเลอร์ MCD-51 มาตรฐานกับคริสตอลความถี่ 11.0529 MHz

ดังนั้นเวลา 1 แมชชีนไซเกิล = $12/f_{XTAL} = 12/11.0529\times10^{6} = 11.08$ ไมโครวินาที

ในการเขียนคำสั่งซึ่งเรียกจากไฟล์ไลบารีร่วมกับคำสั่ง ซึ่งใช้หน่วงเวลาให้แต่ละไทม์สล็อตมีค่าตรงตามข้อกำหนดของการสื่อสารข้อมูลบนบัสหนึ่งสาย

การติดต่อกับอุปกรณ์ในระบบบัส 1 สายของไมโครคอนโทรลเลอร์ MCS-51

ลักษณะของไอซีตรวจวัดอุณหภูมิ DS1820 เป็นไอซีตรวจวัดอุณหภูมิที่ใช้การติดต่อแบบระบบบัสหนึ่งสายมีขาต่อใช้งานเพียงสามขาคือ DQ ซึ่งเป็นขาเชื่อมต่อกับระบบบัส ขาต่อไฟเลี้ยงภายนอกและขากราวด์ และมีโครงสร้างภายในดังภาพที่ 3.5



รูปที่ 2.20 โครงสร้างการทำงานในไอซีตรวจจับอุณหภูมิ

หัวใจสำคัญของไอซี DS1820 อยู่ที่ตรวจวัดอุณหภูมิและหน่วยความจำเร็ว ที่เรียกว่า สแครตช์แพด (Scratchpad) มีการจัดสรรหน่วยความจำส่วนนี้แสดงในตารางที่ 3.1 เมื่อตรวจวัดอุณหภูมิก็จะนำค่าที่วัดได้นี้มาเก็บไว้ใน สแครตช์แพด ที่ไบต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนเมื่อค่าอุณหภูมิที่สูงขึ้น หรือต่ำลงถึงค่าที่กำหนดโดยค่าอุณหภูมิที่กำหนดนี้จะเก็บไว้ในที่ สแครตช์แพด ในไบต์ที่ 2 และ 3

	ไบต์
ข้อมูลอุณหภูมิไบต์ต่ำ (TL)	0
ข้อมูลอุณหภูมิไบต์สูง	1
ข้อมูลอุณหภูมิค่าสูง	2
ข้อมูลอุณหภูมิค่าต่ำ (TL)	3
สำรองไว้	4
สำรองไว้	5
รีจิสเตอร์เก็บค่าการนับ	6
รีจิสเตอร์เก็บค่าการนับต่อ $^{\circ}$C	7
CRC	8

ตารางที่4 การจัดสรรพื้นที่ของสแครตช์แพดใน DS1820

คำสั่งเพื่อควบคุมการทำงานของ DS1820

ในการติดต่อกับไอซี DS1820 จะต้องมีคำสั่งที่ต้องส่งแก่ DS1820 เนื่องจากการใช้งานรูปแบบการทำงาน คำสั่งที่ใช้มากที่สุดมีด้วยกัน 3 คำสั่งดังนี้

1) คำสั่งไม่ติดต่อกับหน่วยความจำหรือสคริปรอม (Skip ROM) เนื่องจากในการใช้งาน DS1820 โดยปกติแล้ว DS1820 อยู่บนสายสัญญาณเพียงตัวเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้นจึงไม่ต้องติดต่อกับหน่วยความจำรอมเพื่ออ่านข้อมูลของคำสั่งสคริปรอมที่ต้องทำให้ DS1820 คือ 0CCH

2) คำสั่งแปลงอุณหภูมิ (Convert Temperature) มีค่าเท่ากับ 44H เมื่อส่งคำสั่งนี้ให้ DS1820 จะต้องทำการวนลูปรออย่างน้อย 200 มิลลิวินาที เพื่อที่จะให้ DS1820 ได้ใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลมาเก็บไว้ในสแครตช์แพด

3) คำสั่งอ่านข้อมูลจากสแครตช์แพด (Read Scratchpad) มีค่าเท่ากับ 0BEH เมื่อส่งคำสั่งนี้ให้ DS1820 จะทยอยส่งข้อมูลค่าอุณหภูมิออกมาทั้งหมด

บทที่3

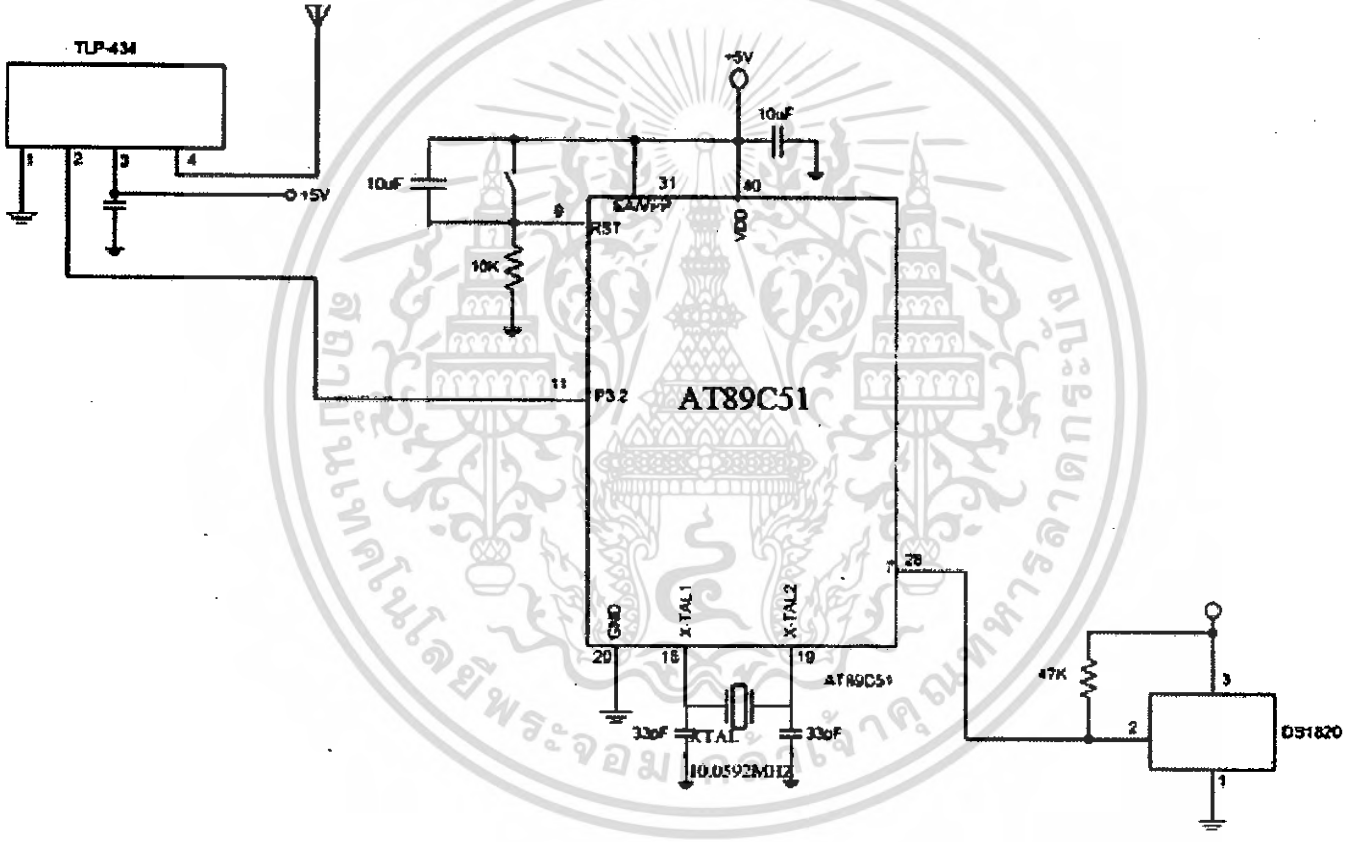
การออกแบบ

การออกแบบวงจรจะแบ่งเป็น 2 ส่วนคือ

1. ส่วนของภาคส่งสัญญาณ
2. ส่วนของภาครับสัญญาณ

3.1 ภาคส่งสัญญาณ

ทำหน้าที่ รับสัญญาณ จากเซ็นเซอร์วัดอุณหภูมิ แล้วทำการ ส่งสัญญาณไปกับคลื่นวิทยุ วงจรภาคส่งทั้งหมดสามรถแสดงได้ดังรูป



รูปที่ 3.1 วงจรภาคส่งสัญญาณ

หลักการทำงานของวงจร

วงจรนี้เป็นวงจรภาคส่งโดยใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C51 ซึ่งจะต้องต่อขาต่างๆเข้ากับวงจร คือ ขา 18 และ ขา 19 ต่อเข้ากับXTAL ทำงานร่วมกับ C33pF 2ตัวเป็นวงจรสร้างสัญญาณนาฬิกาให้กับตัวคอนโทรลเลอร์ 11.0592MHz ส่วนขา (RST)9 เมื่อขานี้ได้รับลอจิก 1 เป็นเวลามากกว่า 2 แมชชีนไซเคิล จะเป็นการรีเซ็ตการทำงานทั้งหมดของ AT89C51 โดยจะมีการต่อสวิตช์ไว้ดังรูป

ส่วนขา(P2.7) 27 ต่อเข้ากับขา DQ ของ DS18S20 โดยมีความต้านทานต่อพูลอัพไว้ สำหรับข้อมูลอุณหภูมิที่อ่านได้จะถูก AT89C51ทำการประมวลผลและปรับรูปแบบของข้อมูลให้เหมาะสม (กระบวนการทั้งหมดจะถูกควบคุมโดยโปรแกรมที่บรรจุอยู่ภายใน)แล้วทำการส่งข้อมูลด้วยพอร์ตอนุกรมภายในของAT89C51ออกทางขา(TXD) 11 ซึ่งต่อเข้ากับ ขา 2 ของ TLP 434

โดยมี source code ของโปรแกรมAT89C51กาคส่งสัญญาณ คือ

```
************************************************
; Program              : Temperature DS1820 01
; Description    : Temperature DS1820 1-Wire Reading (0-127C), External power
;*******************************************************************************
;-------------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------------
ONEWIRE         BIT     P1.5    ; 1-Wire Interface (DS1820 Temp. Sensor)
LCD_EN          BIT     P3.6    ; LCD Module Enable (Active High : Level)
LCD_RS          BIT     P3.7    ; LCD Module Register Select
;-------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------
FLAG            EQU     02FH        ; User FLAG
BUSY            BIT     FLAG.0      ; Define BUSY as bit

LCD_ADDR        EQU     030H        ; For keep LCD Address
LCD_DATA        EQU     031H        ; For keep LCD Data
ONEWIRE_DATA    EQU     032H        ; For keep ONEWIRE Data
TEMP            EQU     033H        ; For keep Temp. Data (Temp L only)
BUFFER          EQU     034H
;-------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------

                ORG     0000H           ; Reset Vector
                MOV     P0,#00000000B   ; Clear Databus
```

```
                SETB    ONEWIRE                 ; Clear Onewire bus
                MOV     TMOD,#021H      ; T1 8Bit Auto, T0 16Bit
                MOV     TH1,#0E8H       ; 1200 bps Timer1 Default
                MOV     TL1,#0E8H       ;
                SETB    TR1             ; Start Timer1
                MOV     SCON,#040H      ; Mode1 RX Disable

MAIN:           CLR     P1.5            ; Off Transmitter

LOOP:           ACALL   DS1820_RST      ; DS1820 Reset
                ACALL   DS1820_PRES     ; DS1820 Presence
                MOV     ONEWIRE_DATA,#0CCH      ; Write Skip ROM
                ACALL   DS1820_WR       ;
                MOV     ONEWIRE_DATA,#044H
                ACALL   DS1820_WR       ;
                SETB    BUSY            ; Set bit BUSY
PRES_CHK_LOOP:  ACALL   DS1820_RST      ; DS1820 Reset
                ACALL   DS1820_PRES     ; DS1820 Presence
                JB      BUSY,PRES_CHK_LOOP      ; Wait for Busy
                NOP                     ; Delay
                NOP                     ;
                NOP                     ;
                NOP                     ;
                ACALL   DS1820_RST      ; DS1820 Reset
                ACALL   DS1820_PRES     ; DS1820 Presence
                MOV     ONEWIRE_DATA,#0CCH      ; Write Skip ROM
                ACALL   DS1820_WR
                MOV     ONEWIRE_DATA,#0BEH
                ACALL   DS1820_WR
                ACALL   DS1820_RD       ; Read DS1820
                MOV     TEMP,ONEWIRE_DATA
```

```
                MOV     A,TEMP
                RR      A
                CLR     ACC.6
                CLR     ACC.7
                SETB    P1.5
                MOV     BUFFER,A
                MOV     A,#11001100B
                MOV     SBUF,A              ; Send Data to SBUF
                JNB     TI,$                ; Wait until TX already (TI=1)
                CLR     TI                  ; Clear TI
                MOV     A,BUFFER
                MOV     SBUF,A              ; Send Data to SBUF
                JNB     TI,$                ; Wait until TX already (TI=1)
                CLR     TI                  ; Clear TI

                ACALL   DS1820_RST          ; DS1820 Reset
                ACALL   DS1820_PRES         ; DS1820 Presence

                AJMP    LOOP
;-------------------------------------------------------------
; DS1820 Data Read
;-------------------------------------------------------------
DS1820_RD:      MOV     R4,#8               ; Set loop 8 times
                CLR     A                   ; Clear ACC.
DS1820_RD_LOOP: CLR     ONEWIRE             ; Clear ONEWIRE
                NOP                         ; Delay
                NOP                         ;
                SETB    ONEWIRE             ; Set ONEWIRE
                NOP                         ; Delay
                NOP                         ;
                NOP                         ;
```

```
                NOP                             ;
                MOV     C,ONEWIRE   ; Get ONEWIRE to Carry Flag
                ACALL   ONEWIRE_DELAY           ; Delay 75 us
                RRC     A                       ;
                DJNZ    R4,DS1820_RD_LOOP ; Do until 8 times
                MOV     ONEWIRE_DATA,A
                RET                             ; Return
;---------------------------------------------------------------
; DS1820 Data Write
;---------------------------------------------------------------
DS1820_WR:      MOV     R4,#8                   ; Set loop 8 times
                MOV     A,ONEWIRE_DATA  ;GetONEWIRE_DATA
DS1820_WR_LOOP: RRC     A           ; Rotate ACC. with Carry Flag
                JNC     DS1820_WR_L             ; Carry Flag was set ?
                CLR     ONEWIRE                 ; Set => TX high
                NOP                             ; Delay
                NOP                             ;
                NOP                             ;
                NOP                             ;
                SETB    ONEWIRE                 ; Set ONEWIRE
                ACALL   ONEWIRE_DELAY           ; Delay 75 us
                AJMP    DS1820_WR_NX            ; Jump to next write
DS1820_WR_L:    CLR     ONEWIRE                 ; Clear => TX Low
                ACALL   ONEWIRE_DELAY           ; Delay 75 us
                SETB    ONEWIRE                 ; Set ONEWIRE
                NOP                             ; Delay
                NOP                             ;
                NOP                             ;
                NOP                             ;
DS1820_WR_NX:   DJNZ    R4,DS1820_WR_LOOP ; Do until 8 times
                RET                             ; Return
```

```
;-------------------------------------------------------------
; DS1820 Reset
;-------------------------------------------------------------
DS1820_RST:     CLR     ONEWIRE             ; Clear ONEWIRE
                ACALL   DELAY_1ms           ; Delay
                SETB    ONEWIRE             ; Set ONEWIRE
                MOV     R4,#8               ; Delay
                DJNZ    R4,$                ;
                RET                         ; Return
;-------------------------------------------------------------
; DS1820 Receive Presence Pulse
;-------------------------------------------------------------
DS1820_PRES:    MOV     R4,#8               ; Set Loop wait 1
DS1820_PRES_1:  MOV     R3,#0               ; Set Loop wait 2
DS1820_PRES_2:  JNB     ONEWIRE,DS1820_PRES_3
                DJNZ    R3,DS1820_PRES_2    ; Wait loop check 2
                DJNZ    R4,DS1820_PRES_1    ; Wait loop check 1
                RET                         ; Return
DS1820_PRES_3:  JNB     ONEWIRE,$           ; Wait until ONEWIRE set
                MOV     R4,#8               ; Delay
                DJNZ    R4,$                ;
                CLR     BUSY                ; Clear BUSY Flag
                RET                         ; Return

;-------------------------------------------------------------
; Dummy Delay time ONEWIRE_DELAY, LCD_DELAY, 50u, 100u, 1m, 10m, 100m, 1s
;-------------------------------------------------------------
ONEWIRE_DELAY:  MOV     R6,#012H            ; Each loop = 75 us
ONEWIRE_DELAY_1: NOP
                NOP
```

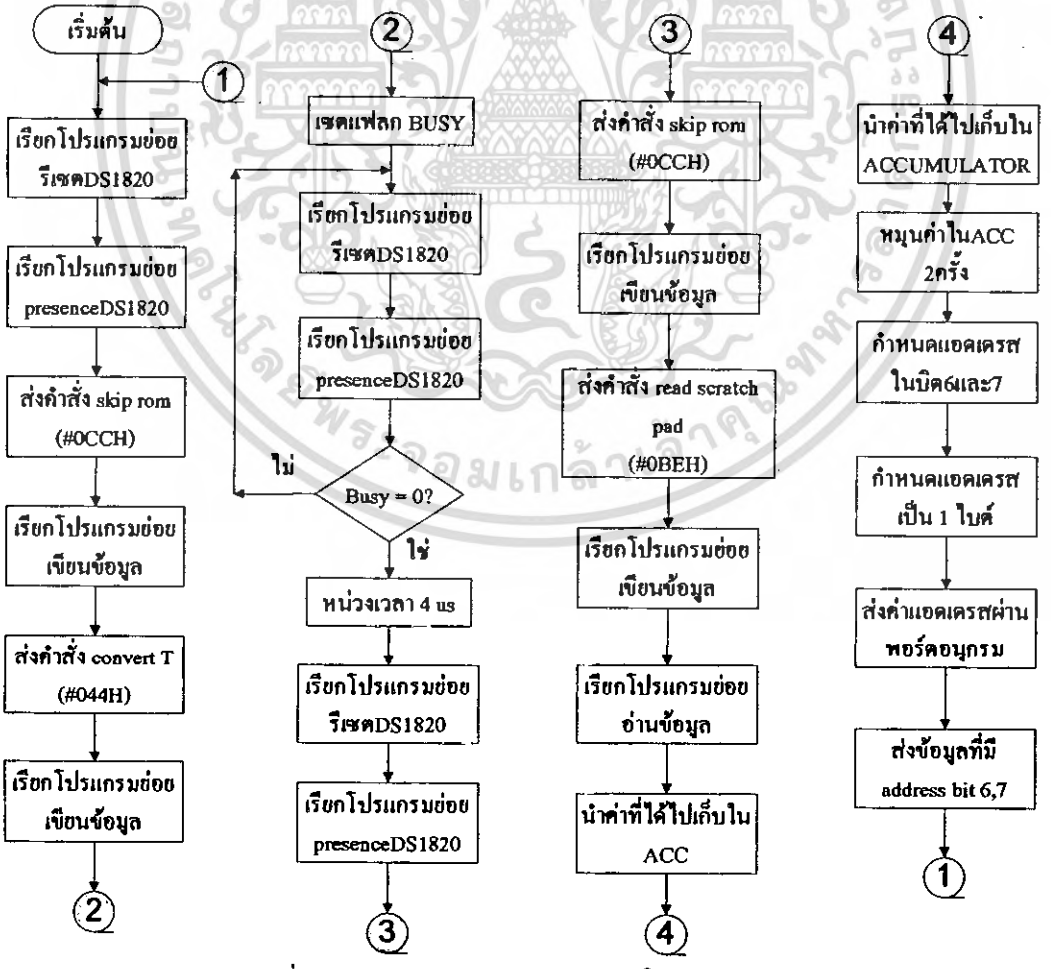
```
                DJNZ    R6,ONEWIRE_DELAY_1
                RET

LCD_DELAY:      MOV     R7,#002             ; Do 2 times
LCD_DELAY_1:    MOV     R6,#0E6H            ; Each loop = 1 ms
LCD_DELAY_2:    NOP
                NOP
                DJNZ    R6,LCD_DELAY_2
                DJNZ    R7,LCD_DELAY_1
                RET

DELAY_50us:     MOV     R6,#00CH            ; Each loop = 50 us
DELAY_50us_1:   NOP
                NOP
                DJNZ    R6,DELAY_50us_1
                RET
DELAY_100us:    MOV     R6,#017H            ; Each loop = 100 us
DELAY_100us_1:  NOP
                NOP
                DJNZ    R6,DELAY_100us_1
                RET
DELAY_1ms:      MOV     R6,#0E6H            ; Each loop = 1 ms
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6,DELAY_1ms_1
                RET
DELAY_10ms:     MOV     R7,#010             ; Do 10 times
DELAY_10ms_1:   MOV     R6,#0E6H            ; Each loop = 1 ms
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
```

```
                RET
DELAY_100ms:    MOV     R7,#100             ; Do 100 times
DELAY_100ms_1:  MOV     R6,#0E6H            ; Each loop = 1 ms
DELAY_100ms_2:  NOP
                NOP
                DJNZ    R6,DELAY_100ms_2
                DJNZ    R7,DELAY_100ms_1
                RET
DELAY_1s:       MOV     R5,#100             ; Do 100 times
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5,DELAY_1s_1
                RET
                END
```

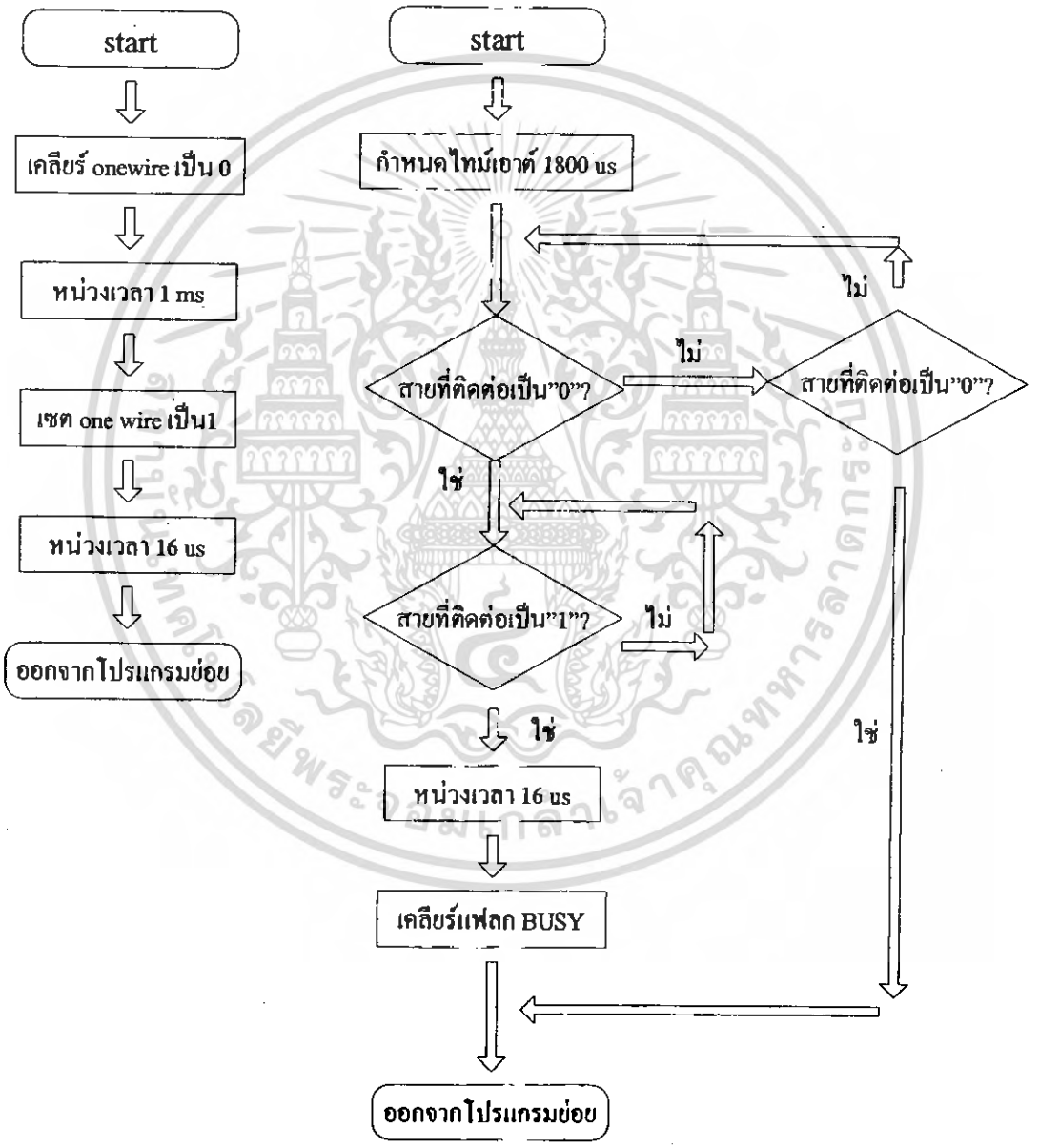
Flow chart การทำงานภายใน AT89C51 (ภาคส่ง)



รูปที่ 3.2 Flow chart การทำงานภายใน AT89C51 (ภาคส่ง)

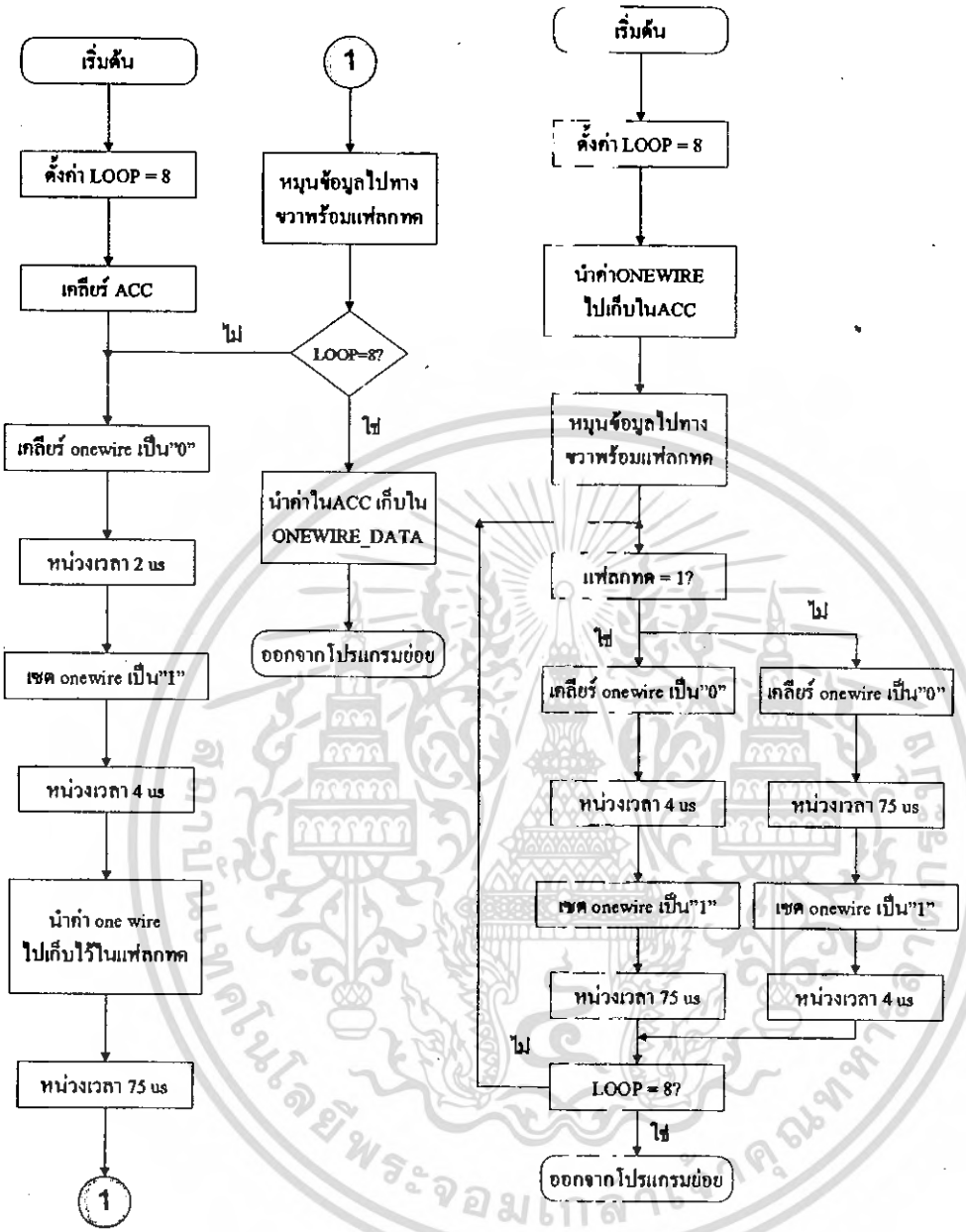
ขั้นแรกทำการรีเซตบัส DS1820 โดยเรียกโปรแกรย่อย รีเซต DS1820 จากนั้นรอการตอบรับเพื่อให้ DS1820 พร้อมสำหรับการทำงาน เมื่อ DS1820 รับค่าอุณหภูมิมาแล้ว ก็ต้องนำข้อมูลมาแปลงให้เป็นข้อมูลแบบดิจิตอล โดยการเขียนคำสั่ง 44 H ให้ DS1820 เพื่อให้ทำการแปลงข้อมูลและนำไปเก็บไว้ใน scratch pad นำมาเก็บไว้ในตัวแปร จากนั้นจะทำการหมุนข้อมูลตัวแปรไปทางขวาเพื่อกำหนดแอดเดรส แล้วทำการกำหนดแอดเดรสที่บิต 6 และบิต 7 จากนั้นจึงส่งค่าอุณหภูมิออกทางพอร์ตอนุกรม

Flow chart โปรแกรมย่อย DS1820



โปรแกรม รีเซต DS1820 โปรแกรมรอการตอบรับจากDS1820

รูปที่ 3.3 Flow chart โปรแกรมย่อย DS1820



โปรแกรมย่อยการอ่านข้อมูลจาก DS1820 โปรแกรมย่อยการเขียนข้อมูลไปDS1820

รูปที่ 3.4 โปรแกรมย่อยอ่านและเขียน ข้อมูลไปDS1820

คำอธิบายโปรแกรมย่อยที่ใช้

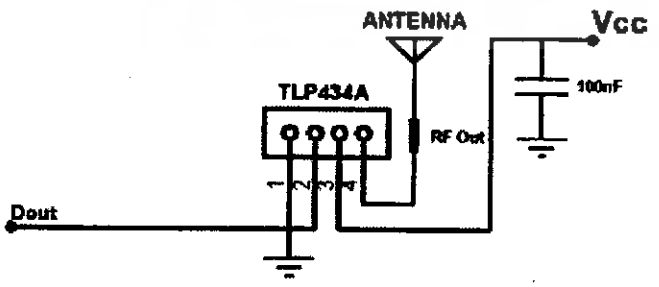
DS1820_RD เป็นโปรแกรมย่อยที่ใช้ในการอ่านค่าจาก DS1820 โดยในขั้นแรกจะทำการเคลียร์สัญญาณ ONEWIRE ให้เป็น "0" ก่อน จากนั้นให้หน่วงเวลาไปประมาณ 2 us แล้วเซตสัญญาณ ONEWIRE ให้เป็น "1" แล้วหน่วงเวลาไปประมาณ 4 us (ไม่เกินค่า Read Data Valid 15us) จึงทำการอ่านค่าสัญญาณ ONEWIRE เก็บไว้ที่แฟลกทด แล้วหน่วงเวลาไป 75us (ค่าไทม์สล็อตอยู่ระหว่าง 60-120us) แล้วหมุนข้อมูลไปทางขวาโดยใช้แฟลกร่วมด้วย ทำเช่นนี้ 8 ครั้งจะได้ค่ามาเก็บไว้ในรีจิสเตอร์ ONEWIRE_DATA

DS1820_WR เป็นโปรแกรมย่อยที่ใช้ในการเขียนค่าจากรีจิสเตอร์ ONEWIRE_DATA ไปยัง DS1820 โดยขั้นแรกจะทำการหมุนข้อมูลไปทางขวาโดยใช้แฟลกทดร่วมด้วยจากนั้นให้ทำการตรวจสอบแฟลกทดว่าถูกเซตหรือไม่ ถ้าถูกเซต ก็ให้ทำการเขียนไทม์สล็อตของลอจิก "1" คือทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 4us (อยู่ในช่วง Write 1 Low Time)แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 75us (ให้มีค่าไทม์สล็อตอยู่ระหว่าง 60-120us) แต่ถ้าแฟลกทดไม่ถูกเซต ก็ให้ทำการเขียนไทม์สล็อตของลอจิก "0" คือทำการเขียนสัญญาณ ONEWIRE เป็นเวลา 75us (อยู่ในช่วง Write 0 Low Time) แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 4us แทน แล้วกลับไปหมุนข้อมูลใหม่ รวมเป็น 8 ครั้ง ก็จะเสร็จสิ้นการเขียนข้อมูล

DS1820_RST เป็นโปรแกรมย่อยที่ใช้สร้างสถานะรีเซต โดยจะทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 1ms (อยู่ในช่วง Reset Time Low 480-4800us) จากนั้นก็จะทำการเซตสัญญาณ ONEWIRE เป็นเวลา 16us (พิจารณาจากค่า Presence Detect High 15-60us) เพื่อหน่วงเวลารอ Presence Pulse จาก DS1820 ต่อไป

DS1820_PRES เป็นโปรแกรมย่อยที่ทำหน้าที่รอรับการตอบรับจาก DS1820(Presence Pulse) และใช้ทำหน้าที่รอการสิ้นสุดกระบวนการแปลงค่าอุณหภูมิด้วย โดยจะทำการวนตรวจสอบว่ามีการเคลียร์สัญญาณ ONEWIRE เป็น "0" หรือไม่ ในช่วงเวลาประมาณ 8.85 ms (การตอบรับปรกติจะสามารถตรวจสอบได้ในกายในช่วง Presence Detect High 15-60us ในกรณีที่รอการแปลงค่าอุณหภูมิ จะต้องใช้เวลาช่วง Temperature Conversion Time 200us – 500us) โดยถ้าเกินจากช่วงเวลานี้จะคืนค่าแฟลก BUSY เป็น "1" อีกครั้ง แล้วจะทำการหน่วงเวลาไปประมาณ 16us ก่อนจะทำการเคลียร์แฟลก BUSY เป็น "0" เพื่อแสดงว่ามีการตอบรับกลับมา จึงจะกลับเข้าสู่การทำงานหลักได้

RF Module TLP434A ทำการส่งข้อมูลที่ได้จาก AT89C51 ซึ่งเป็นข้อมูลแบบอนุกรมออกไปด้วยคลื่นวิทยุความถี่ 433.92 MHz โดยเป็นการมอดดูเลทสัญญาณแบบ Amplitude Shift Keying (ASK)

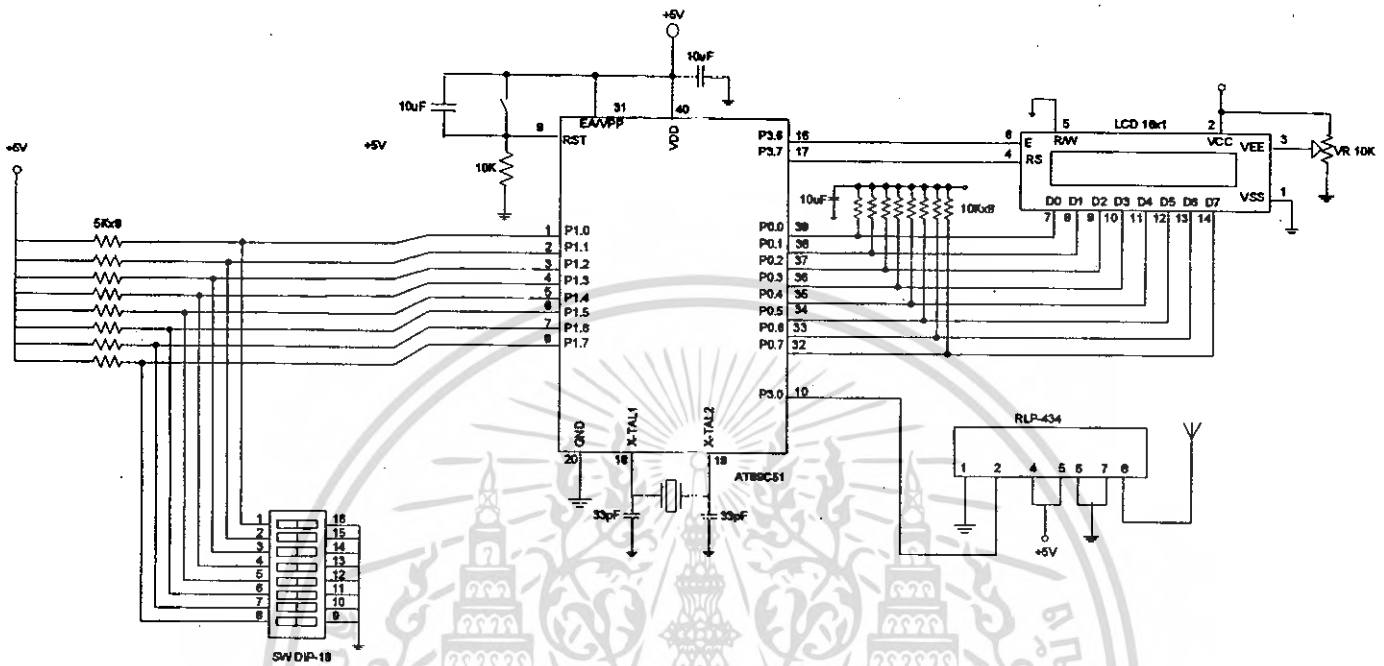


รูปที่ 3.5 วงจร RF Module TLP434A

วงจรภาครับสัญญาณ

ทำหน้าที่รับสัญญาณจากภาคส่งแล้วนำไปแสดงผลทาง หน้าจอ LCD

วงจรภาครับทั้งหมดสามรถแสดงได้ดังรูป



รูปที่3.6 วงจรภาครับสัญญาณ

ส่วนรับข้อมูล

ใช้ไมโครคอนโทรลเลอร์AT89C51 ซึ่งเป็นเบอร์เดียวกับภาคส่งโดยเซ็ตค่า OSCILLATOR เท่ากับภาคส่งต่อขา และดาต้าที่ได้รับ จาก RLP–434 เข้าทางขา RX (ขา10) โดยรับข้อมูลด้วยพอร์ตอนุกรม แล้วส่งออกมาเป็นแบบขนาน 8 บิตไปที่หน้าจอแอลซีดี รวมทั้งควบคุมการทำงานของแอลซีดีด้วย

โดยมี Source code ของโปรแกรมภายใน AT89C51 ภาครับคือ

```
;*******************************************************************************
; Program                : RECIVE
;*******************************************************************************
;-------------------------------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------------------------------
ONEWIRE                BIT     P2.7    ; I-Wire Interface (DS1820 Temp. Sensor)
```

```
LCD_EN          BIT     P3.6    ; LCD Module Enable (Active High : Level)
LCD_RS          BIT     P3.7            ; LCD Module Register Select

;---------------------------------------------------------------
; Define User Register
;---------------------------------------------------------------
FLAG            EQU     02FH    ; User FLAG
BUSY            BIT     FLAG.0  ; Define BUSY as bit

LCD_ADDR        EQU     030H    ; For keep LCD Address
LCD_DATA        EQU     031H    ; For keep LCD Data
ONEWIRE_DATA    EQU     032H    ; For keep ONEWIRE Data
TEMP            EQU     033H    ; For keep Temp. Data (Temp L only)
ADDRESS         EQU     034H
ADDRESS_2       EQU     023H
REC             EQU     024H
;---------------------------------------------------------------
; Main Program.
;---------------------------------------------------------------
                ORG     0000H           ; Reset Vector
                MOV     P0,#00000000B   ; Clear Databus
                SETB    ONEWIRE                 ; Clear Onewire bus
                MOV     TMOD,#021H      ; T1 8Bit Auto, T0 16Bit
                MOV     TH1,#0E8H       ; 1200 bps Timer1 Default
                MOV     TL1,#0E8H       ;

                SETB    TR1             ; Start Timer1
                MOV     SCON,#050H      ; Mode1 RX Disable

MAIN:           ACALL   INIT_LCD        ; Call LCD Initial subroutine
```

```
                MOV     LCD_ADDR,#000H      ; Set Address 00H
                ACALL   SET_ADDR_LCD        ;
                MOV     DPTR,#TITLE_1
                ACALL   WRLINE_LCD
                ACALL   DELAY_1s            ; Delay
                ACALL   DELAY_1s            ;
                MOV     A,#11111111B
                MOV     P1,A
                MOV     A,P1
DIP_A:          CJNE    A,#00000001B,DIP_B
                MOV     ADDRESS,#00001111B
                MOV     ADDRESS_2,#00000000B
                JMP     LOOP
DIP_B:          CJNE    A,#00000010B,DIP_C
                MOV     ADDRESS,#10101010B
                MOV     ADDRESS_2,#01000000B
                JMP     LOOP
DIP_C:          CJNE    A,#00000100B,DIP_D
                MOV     ADDRESS,#11111111B
                MOV     ADDRESS_2,#10000000B
                JMP     LOOP
DIP_D:          CJNE    A,#00001000B,DIP_A
                MOV     ADDRESS,#11110000B
                MOV     ADDRESS_2,#11000000B
                JMP     LOOP
LOOP_2:
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_2
                ACALL   WRLINE_LCD
```

```
                MOV     LCD_ADDR,#040H
                ACALL   SET_ADDR_LCD
                MOV     R4,#08H
                MOV     A,ADDRESS
LOOP_3:
                RLC     A
                JC      ONE
                MOV     LCD_DATA,#'0'
                ACALL   WRCHAR_LCD
                DJNZ    R4,LOOP_3
                AJMP    LOOP_4
ONE:            MOV     LCD_DATA,#'1'
                ACALL   WRCHAR_LCD
                DJNZ    R4,LOOP_3
LOOP_4:         ACALL   DELAY_1S
                ACALL   DELAY_1S
LOOP:
                JNB     RI,$            ; Wait byte received
                CLR     RI              ;
                MOV     A,SBUF          ; Get byte from SBUF

                CJNE    A,ADDRESS,LOOP
                JNB     RI,$            ; Wait byte received
                CLR     RI              ;
                MOV     A,SBUF          ; Get byte from SBUF
                PUSH    ACC
                RLC     A
                MOV     REC.7,C
                RLC     A
                MOV     REC.6,C
                MOV     A,REC
                CJNE    A,ADDRESS_2,LOOP
```

```
                POP     ACC
                RL      A
                CLR     ACC.0
                CLR     ACC.7

                MOV     TEMP,A
                MOV     LCD_ADDR,#000H          ; Set Address 00H
                ACALL   SET_ADDR_LCD            ;
                MOV     DPTR,#SCR_TEMP
                ACALL   WRLINE_LCD

                MOV     LCD_ADDR,#040H          ; Set Address 40H
                ACALL   SET_ADDR_LCD            ;

                MOV     A,TEMP                  ; Get Temp. Data
                CLR     C                       ; Clear Carry Flag
                RRC     A
                MOV     LCD_DATA,A  ; Write Temp. in 3 Decimal Digits
                ACALL   HEX2LCD                 ;
                MOV     LCD_ADDR,#044H          ; Set Address 44H
                ACALL   SET_ADDR_LCD            ;
                MOV     A,TEMP                  ; Get Temp. Data
                JNB     ACC.0,WRITE_0C          ; Check LSB was set?
                MOV     LCD_DATA,#'5'           ; Set => Load '5'
                AJMP    WRITE_NEXT      ; Jump to write LCD
WRITE_0C:       MOV     LCD_DATA,#'0'   ; Not set => Load '0'
WRITE_NEXT:     ACALL   WRCHAR_LCD              ; Write to LCD
                ACALL   DELAY_1s        ; Delay
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_3
                ACALL   WRLINE_LCD
```

```
                MOV     LCD_ADDR,#040H
                ACALL   SET_ADDR_LCD
                MOV     R4,#08H
                MOV     A,STORE
LOOP_6:         RLC     A
                JC      ONE
                MOV     LCD_DATA,#'0'
                ACALL   WRCHAR_LCD
                DJNZ    R4,LOOP_6
                AJMP    LOOP_5
ONE_2:          MOV     LCD_DATA,#'1'
                ACALL   WRCHAR_LCD
                DJNZ    R4,LOOP_6
LOOP_5:         ACALL   DELAY_1S
                ACALL   DELAY_1S
                AJMP    LOOP                    ; Jump to loop

;-------------------------------------------------------------
; HEX Code to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------
HEX2LCD:        PUSH    ACC                     ; Push ACC.
                MOV     A,LCD_DATA              ; Get Data
                MOV     B,#100                  ;
                DIV     AB                      ; Divide by 100
                ADD     A,#030H                 ; Convert to ASCII
                CJNE    A,#030H,HEX2_LCD_NX     ; Check x100 = 0 ?
                MOV     A,#' '                  ; 0 => Write Space
HEX2_LCD_NX:    MOV     LCD_DATA,A              ;
                ACALL   WRCHAR_LCD              ; Write x100
                MOV     A,B                     ;
                MOV     B,#10                   ;
```

```
                DIV     AB              ; Divide by 10
                ADD     A,#030H         ; Convert to ASCII
                MOV     LCD_DATA,A      ;
                ACALL   WRCHAR_LCD      ; Write Lower HEX Code
                MOV     A,B             ; Get Remainder x1
                ADD     A,#030H         ; Convert to ASCII
                MOV     LCD_DATA,A      ;
                ACALL   WRCHAR_LCD      ; Write Lower HEX Code
                POP     ACC             ; Pop ACC.
                RET                     ; Return
;---------------------------------------------------------------
; LCD Initialize
;---------------------------------------------------------------
INIT_LCD:       ACALL   DELAY_100ms     ; Delay
                CLR     LCD_RS                  ; Clear LCD_RS Pin
                MOV     P0,#00111000B   ; 8bit Mode
                CALL    LCD_CLK                 ; Pulse LCD Clock
                ACALL   DELAY_10ms      ; Delay
                MOV     P0,#00111000B   ; 8bit Mode
                ACALL   LCD_CLK                 ; Pulse LCD Clock
                ACALL   LCD_OFF                 ; Display Off
                ACALL   LCD_CLR                 ; Clear Display
                MOV     P0,#00000110B   ; Entry Mode
                ACALL   LCD_CLK                 ; Pulse LCD Clock

                ACALL   LCD_HOME        ; Return Home Display
;---------------------------------------------------------------
; LCD Clear Display
;---------------------------------------------------------------
LCD_CLR:        CLR     LCD_RS                  ; Clear LCD_RS Pin
                MOV     P0,#00000001B   ; Display Clear
                ACALL   LCD_CLK                 ; Pulse LCD Clock
```

```
                RET
;---------------------------------------------------------------
; LCD Return Home
;---------------------------------------------------------------
LCD_HOME:       CLR     LCD_RS          ; Clear LCD_RS Pin
                MOV     P0,#00000010B   ; Return Home
                ACALL   LCD_CLK         ; Pulse LCD Clock
                RET
;---------------------------------------------------------------
; LCD Display Off
;---------------------------------------------------------------
LCD_OFF:        CLR     LCD_RS          ; Clear LCD_RS Pin
                MOV     P0,#00001000B   ; Display Off
                ACALL   LCD_CLK         ; Pulse LCD Clock
                RET
;---------------------------------------------------------------
; LCD Clk
;---------------------------------------------------------------
LCD_CLK:        SETB    LCD_EN          ; Pulse Clock to LCD_EN
                ACALL   LCD_DELAY
                CLR     LCD_EN
                ACALL   LCD_DELAY
                RET
;---------------------------------------------------------------
; LCD Display On
;---------------------------------------------------------------
LCD_ON:         CLR     LCD_RS          ; Clear LCD_RS Pin
                MOV     P0,#00001100B   ; Display On
                ACALL   LCD_CLK
                RET
;---------------------------------------------------------------
; Set LCD Address
```

```
; I/P:          LCD_ADDR
;---------------------------------------------------------------
SET_ADDR_LCD:   CLR     LCD_RS              ; Clear LCD_RS Pin
                MOV     A,LCD_ADDR          ; Move LCD_ADDR to ACC.
                SETB    ACC.7               ; Set bit ACC.7
                MOV     P0,A                ; Move to DATABUS
                ACALL   LCD_CLK             ; Pulse LCD Clock
                RET
;---------------------------------------------------------------
; Write Character to show LCD
; I/P:          LCD_DATA
;---------------------------------------------------------------
WRCHAR_LCD:     SETB    LCD_RS              ; Set LCD_RS Pin
                MOV     P0,LCD_DATA         ; Move LCD_DATA to DATABUS
                ACALL   LCD_CLK             ; Pulse LCD Clock
                ACALL   LCD_ON              ; Display On
                RET
;---------------------------------------------------------------
; Write Line of 16 Character from ROM
; I/P:          DPTR : Locate ROM Address
;---------------------------------------------------------------
WRLINE_LCD:     MOV     R0,#0               ; Clear loop counter
WRLINE_LCD_1:   SETB    LCD_RS              ; Set LCD_RS Pin
                CLR     A                   ; Clear ACC.
                MOVC    A,@A+DPTR           ; Move data from @DPTR to ACC.
                MOV     P0,A                ; Move ACC. to DATABUS
                ACALL   LCD_CLK             ; Pulse LCD Clock
                INC     DPTR                ; Increase Pointer
                INC     R0                  ; Increase loop counter
                CJNE    R0,#8,WRLINE_LCD_1  ; Do until 8 times
                MOV     LCD_ADDR,#040H      ; Set Later 8 Char. Address
                ACALL   SET_ADDR_LCD
```

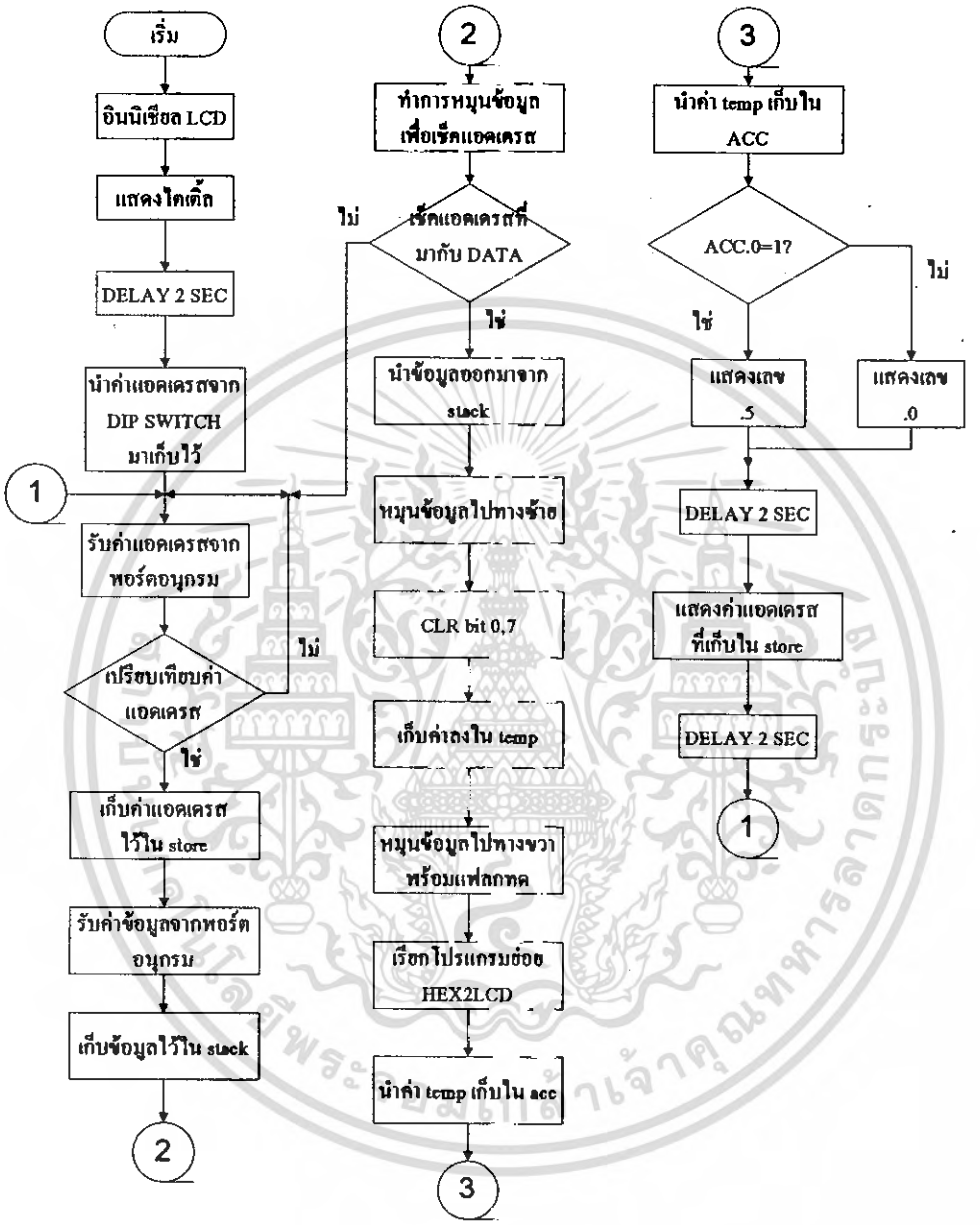
```
WRLINE_LCD_2:   SETB    LCD_RS                  ; Set LCD_RS Pin
                CLR     A                       ; Clear ACC.
                MOVC    A,@A+DPTR
                MOV     P0,A                    ; Move ACC. to DATABUS
                ACALL   LCD_CLK                 ; Pulse LCD Clock
                INC     DPTR                    ; Increase Pointer
                INC     R0                      ; Increase loop counter
                CJNE    R0,#16,WRLINE_LCD_2     ; Do until 8+8 times
                ACALL   LCD_ON                  ; Display On
                RET
;---------------------------------------------------------------
; Dummy Delay time ONEWIRE_DELAY, LCD_DELAY, 50u, 100u, 1m, 10m, 100m, 1s
;---------------------------------------------------------------
ONEWIRE_DELAY:      MOV     R6,#012H            ; Each loop = 75 us
ONEWIRE_DELAY_1:    NOP
                    NOP
                    DJNZ    R6,ONEWIRE_DELAY_1
                    RET
LCD_DELAY:          MOV     R7,#002             ; Do 2 times
LCD_DELAY_1:        MOV     R6,#0E6H            ; Each loop = 1 ms
LCD_DELAY_2:        NOP
                    NOP
                    DJNZ    R6,LCD_DELAY_2
                    DJNZ    R7,LCD_DELAY_1
                    RET

DELAY_50us:     MOV     R6,#00CH                ; Each loop = 50 us
DELAY_50us_1:   NOP
                NOP
                DJNZ    R6,DELAY_50us_1
                RET
```

```
DELAY_100us:    MOV     R6,#017H        ; Each loop = 100 us
DELAY_100us_1:  NOP
                NOP
                DJNZ    R6,DELAY_100us_1
                RET
DELAY_1ms:      MOV     R6,#0E6H        ; Each loop = 1 ms
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6,DELAY_1ms_1
                RET
DELAY_10ms:     MOV     R7,#010         ; Do 10 times
DELAY_10ms_1:   MOV     R6,#0E6H        ; Each loop = 1 ms
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
                RET
DELAY_100ms:    MOV     R7,#100         ; Do 100 times
DELAY_100ms_1:  MOV     R6,#0E6H        ; Each loop = 1 ms
DELAY_100ms_2:  NOP
                NOP
                DJNZ    R6,DELAY_100ms_2
                DJNZ    R7,DELAY_100ms_1
                RET
DELAY_1s:       MOV     R5,#100         ; Do 100 times
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5,DELAY_1s_1
                RET
;-------------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------------
TITLE_1:        DB      ' KMITL SENSOR '
```

```
SCR_TEMP:        DB          ' Temp :   . ',0DFH,'C '
                 END
```

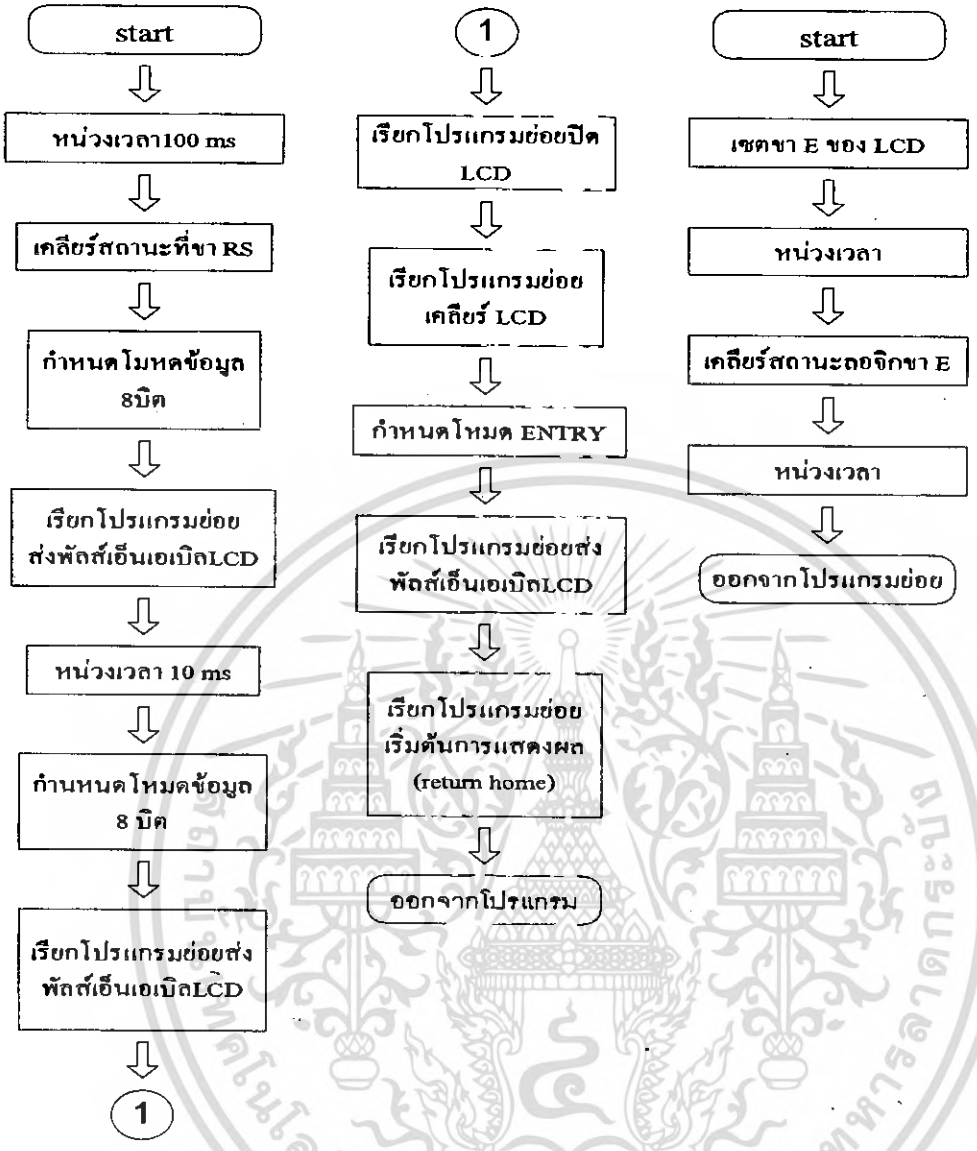
Flow chart Main program



รูปที่ 3.7 Flow chart main program ภายในAT89C51

คำอธิบายโปรแกรมที่ใช้

ขั้นแรกกำหนดการทำงานของ LCD ก่อน เรียกว่าการ Initial LCD ซึ่งมีขั้นตอนดังนี้

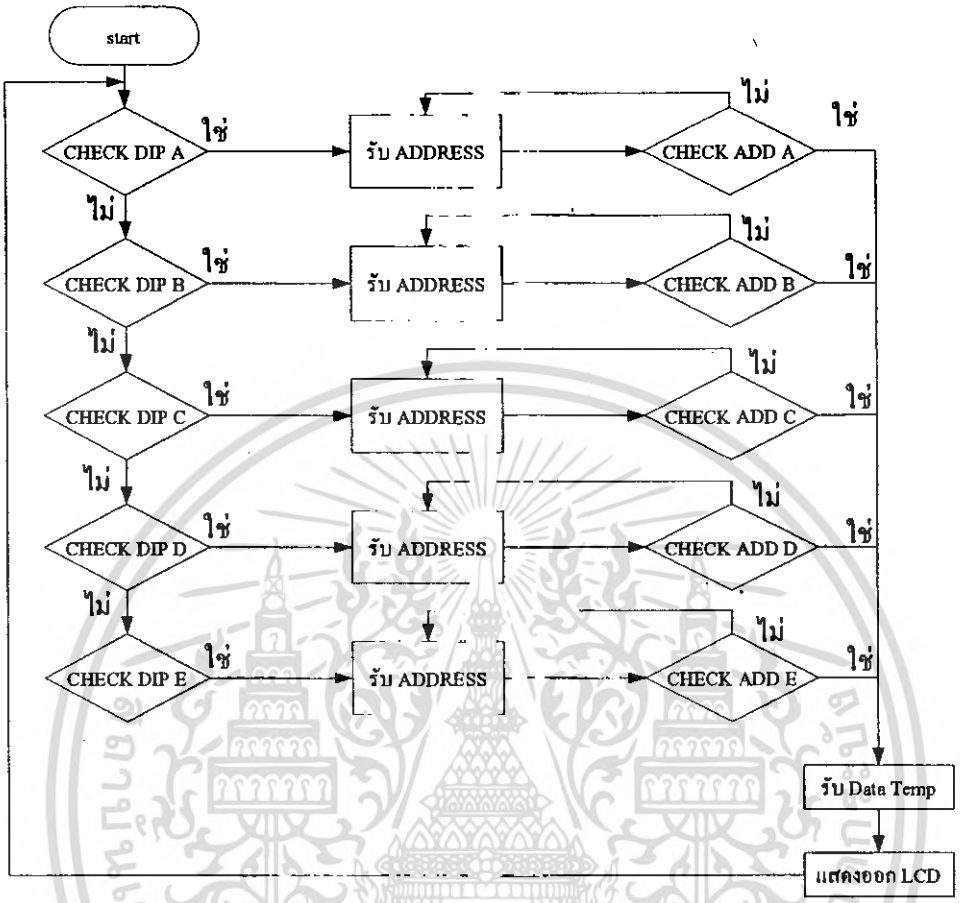


ซ้ายโปรแกรมย่อยอินนิเชียล LCD ขวาโปรแกรมย่อยCLK ที่ขา EN

รูปที่ 3.8 Flow chart โปรแกรมย่อยอินนิเชียล และ CLK LCD

จากนั้นโปรแกรมจะตรวจสอบข้อมูลจาก DIP SWITCH เพื่อดูว่าผู้ใช้เลือกที่จะดูข้อมูลจาก Sensor ตัวไหน แล้วทำการรับค่าจาก Sensor ตัวนั้นโดยตรวจสอบจาก address ที่กาคส่งส่งออกมาพร้อมกับข้อมูล sensor แต่ละตัวจะส่งแอดเดรสออกมาไม่เหมือนกัน โปรแกรมจะตรวจสอบ DIP SWITCH ก่อน เมื่อได้ค่าข้อมูลจาก DIP SWITCH โปรแกรมก็จะเลือกตรวจสอบแอดเดรส ที่สอดคล้องกับค่า DIP SWITCH นั้น ขั้นตอนเหล่านั้นแสดงเป็น Flow chart ได้ดังนี้

Flow chart เลือก DIP SWITCH



รูปที่ 3.9 Flow chart เลือก DIP SWITCH

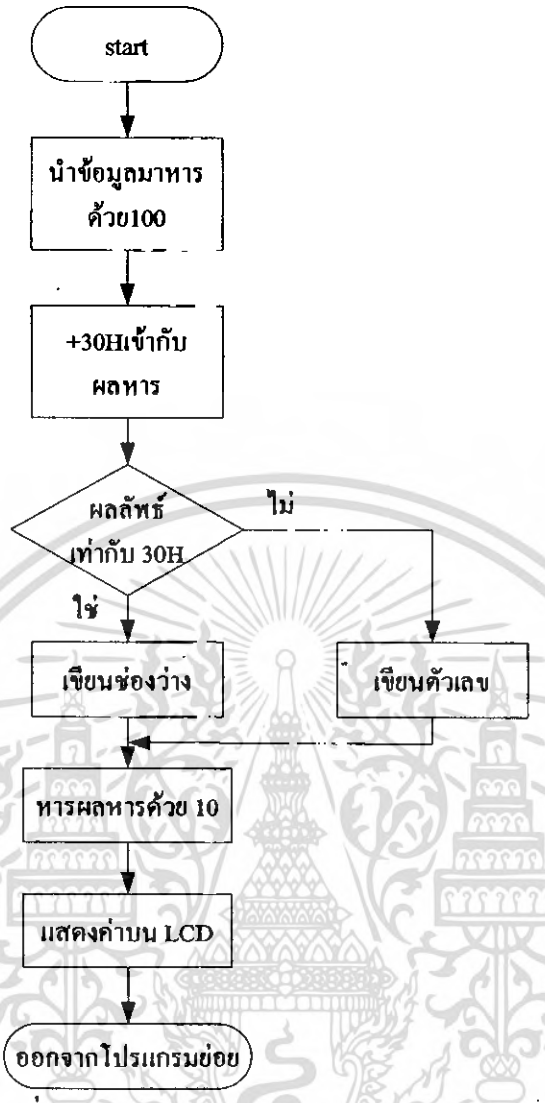
เมื่อรับค่าอุณหภูมิแล้วเราจะต้องเช็คแอดเดรสที่มากับข้อมูลอีกครั้งโดยเราได้กำหนดไว้ในตอนส่งแล้วว่าให้บิต 6 และ บิต 7 ของข้อมูลเป็นแอดเดรส จากนั้นเราจะทำการวนข้อมูลไปทางซ้ายผ่านแฟลกทดแล้วเก็บข้อมูลไว้ในตัวแปร rec.7 จากนั้นจะทำการวนอีกครั้งแล้วทำการเก็บไว้ในตัวแปร rec.6 จากนั้นจะทำการตรวจสอบว่าตรงกับแอดเดรสที่กำนหดไว้หรือไม่ ถ้าตรงก็เข้าสู้ขั้นตอนการแปบงอุณหภูมิ ถ้าไม่ตรงให้กลับไปตรวจสอบแอดเดรสใหม่ตั้งแต่ต้น

เมื่อได้ข้อมูลอุณหภูมิที่ต้องการแล้ว โปรแกรมก็จะทำการประมวลผลทางคณิตศาสตร์ เพื่อแปลงข้อมูลดิจิตอลให้เป็นเลขฐาน 10 โดยมีขั้นตอนดังนี้

1.ข้อมูลอุณหภูมิที่ได้จะมีทั้งหมด 8 บิต โปรแกรมจะทำการตรวจสอบบิตที่ 0 หรือบิตทางด้านขวาสุด หากเป็น 0 จะได้เป็นทศนิยม 0.0 แต่ถ้าหากเป็น 1 จะได้เป็นทศนิยม 0.5

2.เลื่อนบิตข้อมูลไปทางขวา 1 บิต ข้อมูลก็จะเหลือ 7 ยิต จากนั้นนำไปแปลงเป็นเลขฐาน 10

3.นำค่าข้อมูลในข้อ 1 และข้อ 2 มารวมกันจะได้เป็นค่าอุณหภูมิฐาน 10 พร้อมเลขทศนิยม โดยแสดงเป็นFlow chart ได้ดังนี้



รูปที่ 3.10 Flow chart โปรแกรมย่อย HEX 2 LCD

เมื่อได้ข้อมูลเป็นเลขฐาน 10 แล้วก็นำข้อมูลเหล่านั้นแสดงออกทาง LCDต่อไป

คำอธิบายโปรแกรมย่อยที่ใช้

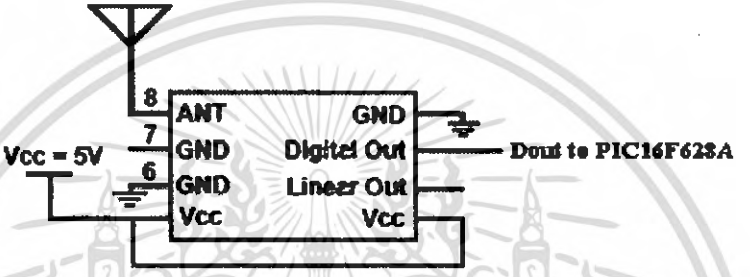
HEX2LCD เป็นโปรแกรมย่อยที่ใช้แปลงค่ารีจิสเตอร์ มาแสดงเป็นเลขฐานสิบ 3 หลักที่โมดูล LCD คดยนำค่าจากรีจิสเตอร์ LCD_DATA มาหารด้วย 100 ก่อน แล้วนำค่าตัวส่วนมาแสดงเป็นเลขหลักแรกแต่ถ้าค่าที่ได้เป็นศูนย์ จะเขียนเป็นช่องว่างแทน ต่อมานำค่าเศษที่เหลือจากการหารครั้งแรก มาหารด้วย 10 แล้วนำค่าตัวส่วนมาแสดงน LCD เป็นเลขหลักต่อมา และนำค่าเศษจากการหารครั้งที่สองมาแสดงเป็นเลขหลักสุดท้าย (การนำค่าเลขมาแสดงบน LCD ต้องบวกค่านั้นด้วย 30H ให้เป็นรหัสแอสกี้ก่อนจึงนำไปใช้ได้)

INIT_LCD เป็นโปรแกรมย่อยสำหรับเตรียม LCD ให้พร้อมในการทำงาน โดยจะเรียกเพียงครั้งแรกครั้งเดียว หลังจากเริ่มจ่ายไปให้กับตัว LCD โดยจะตั้งค่าการเชื่อมต่อเป็นแบบ 8 บิต เคลียร์ข้อความใน DDRAM ตั้งค่าการเพิ่มแอดเดรส DDRAM โดยอัตโนมัติ และเคลียร์ค่าแอดเดรสใน DDRAM ด้วย

SET_ADDR_LCD เป็นโปรแกรมย่อยสำหรับตั้งค่าแอดเดรส DDRAM (ตำแหน่งของเคอร์เซอร์)โดยรับค่าจากตัวแปร LCD_ADDR

WRLINE_LCD เป็นโปรแกรมย่อยสำหรับเขียนข้อความ LCD 16 ตัว จากค่าที่เก็บอยู่ในโปรแกรม โดยจะใช้ร่วมกับรีจิสเตอร์ DPTR ในการอ้างถึงตำแหน่งข้อความ ที่ต้องการนำมาแสดงบน LCD

RF Module RLP434A ทำหน้าที่ดีมอดดูเลทสัญญาณคำสั่งออกจากคลื่นพาห์ และส่งข้อมูลที่ดีมอดดูเลทแล้วต่อไปที่ AT89C51 เพื่อทำการแปลงข้อมูลเป็นแบบขนาน 4 บิตส่งต่อไปที่ LCD



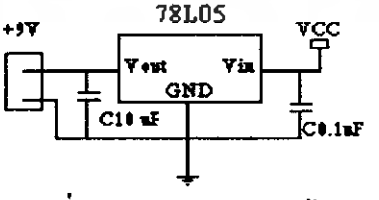
รูปที่3.11 วงจร RF Module RLP434A

ส่วนแสดงผล

ใช้ แอลซีดีขนาด 16ตัวอักษร1บรรทัด ต่อใช้งานในโหมด 8 บิต ในส่วนขาควบคุมจะมีอยู่ 3 ขา คือ ขา EN ทำหน้าที่อีเนเบิล แอลซีดีให้ทำงาน, ขาR/W โดยในการออกแบบจะต่อลงกราวด์ เพราะว่าจะใช้งานในส่วนของเขียนคำสั่งและข้อมูลลงไปเท่านั้นและสุดท้ายของขาควบคุม RS ทำหน้าที่ล็อกชนิดของข้อมูลที่ประมวลผลในขณะนั้นว่าเป็นข้อมูลหรือคำสั่ง และที่ขา 3 จะต่อ VR1 เพื่อทำหน้าที่ปรับความเข้มของตัวอักษรด้วย

วงจร POWER SUPPLY

วงจรนี้ใช้ไอซีเร็กกูเลเตอร์ 78L05 โดยจ่ายไฟเลี้ยงขนาด 5 โวลต์ให้กับวงจร เพื่อให้ได้ไฟที่คงที่ โวลต์ไม่ลดลงเมื่อแบตเตอรี่ ไฟตก จะทำให้ยังคงที่ที่ 5 โวลต์เท่าเดิม



รูปที่ 3.12 วงจรแหล่งจ่ายไฟ

วิเคราะห์และสรุปผลการทดลอง

ในรายงานนี้ได้กล่าวถึงเป็นมาของโครงการ แนวคิด ทฤษฎี และรายละเอียดการสร้างเครื่องวัดอุณหภูมิแบบไร้สาย รวมถึงการทดสอบการทำงานเบื้องต้น โดยใช้ ไอซีวัดอุณหภูมิแบบดิจิตอล ทำการวัดแล้วส่งค่าด้วยคลื่นวิทยุ ไปแสดงผลที่จอLCD โดยสามารถวัดอุณหภูมิได้ตั้งแต่ 63 องศาเซลเซียส ถึง 0 องศาเซลเซียส โดยมีหลักการทำงานคร่าวๆ คือ ใช้ AT89C51 เป็นอุปกรณ์มาสเตอร์ ทำการส่งคำสั่งไปที่ DS1820 และทำการรับข้อมูลอุณหภูมิ แล้วส่งไปที่ TLP 434 จากนั้น TLP จะทำการมอดดูเลตส่งเป็นคลื่นวิทยุ ไปที่RLP จากนั้นจะใช้AT89C51เป็นตัวรับค่าแล้วทำการแปลงสัญญาณ hex เป็น ASCII CODE ไปแสดงผลในLCD รวมทั้งอินนิเชียล LCD ด้วย

ปัญหาและแนวทางแก้ไข

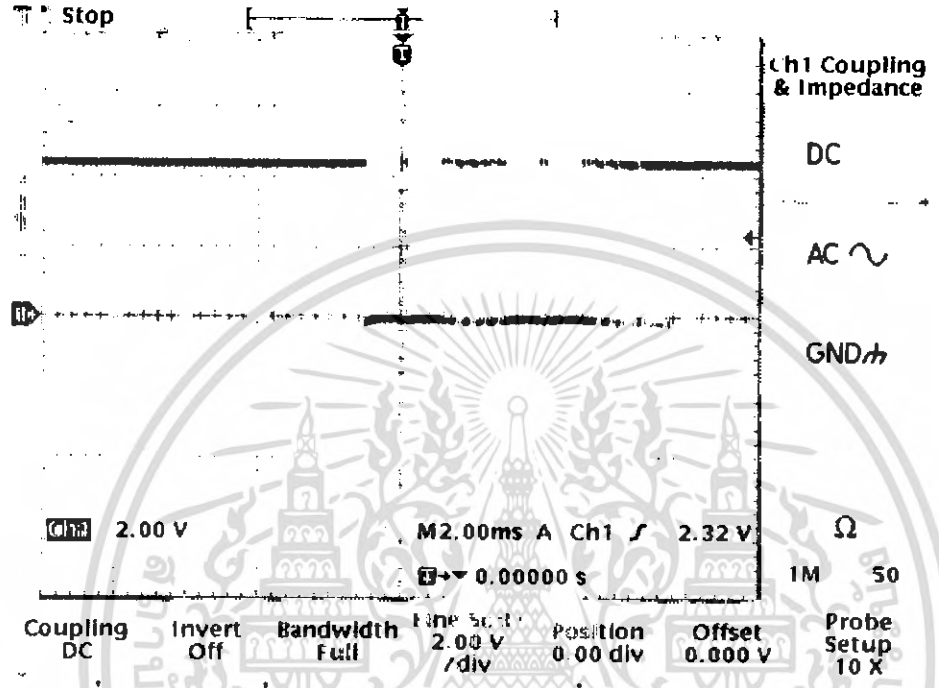
ปัญหาจะเกิดจาก NOISE ที่รบกวนวงจร และปัญหาการแทรกสอดกันระหว่างตัวส่ง ซึ่งได้พยายามใช้วิธีในการแก้ไขหลายวิธีด้วยกันแต่วิธีที่สามารถทำให้ลดการรบกวนกันมากที่สุดคือการตรวจสอบแอดเดรส 2 ครั้งดังได้กล่าวไว้แล้วในตอนต้น แต่วิธีนี้ยังมีการรบกวนอยู่บ้าง

ประโยชน์ที่ได้รับ

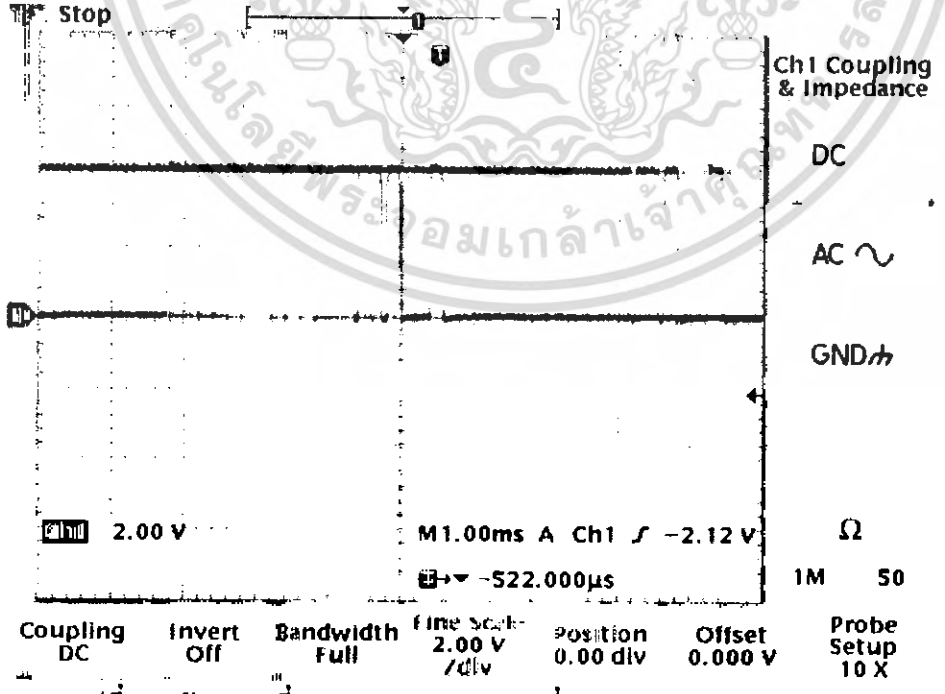
สามารถนำความรู้ในไมโครคอนโทรลเลอร์ MCS51ไปประยุกต์ใช้งานต่างๆ รวมทั้งความรู้เกี่ยวกับเซนเซอร์วัดอุณหภูมิ ระบบบัสหนึ่งสาย การมอดูเลต การออกแบบลายวงจร การแก้ปัญหารวมถึงการใช้โมดูล LCDด้วย

ผลการทดลอง

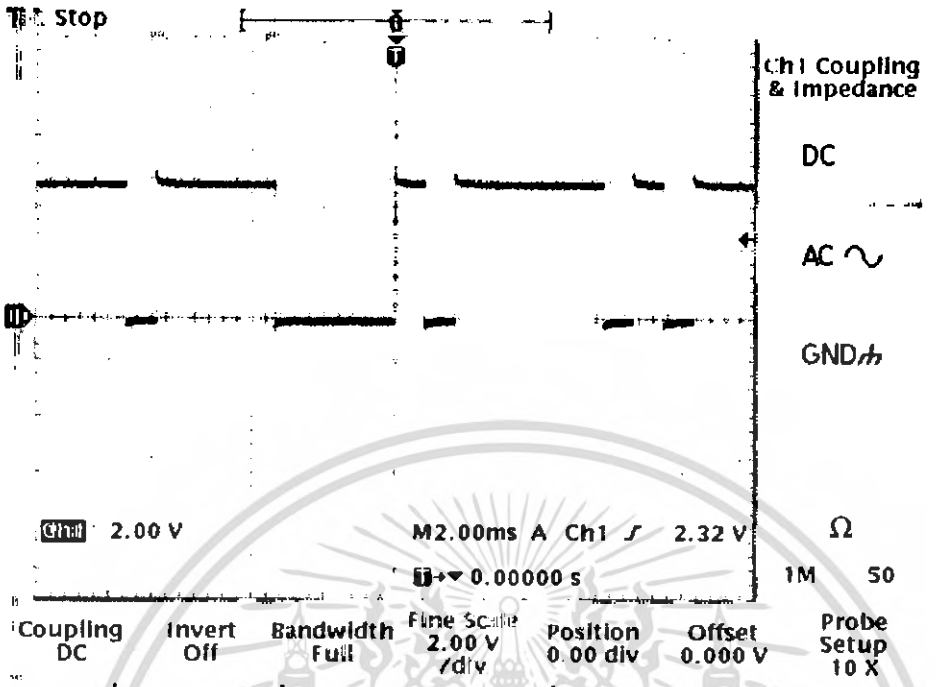
ภาพสัญญาณที่จุดต่างๆ



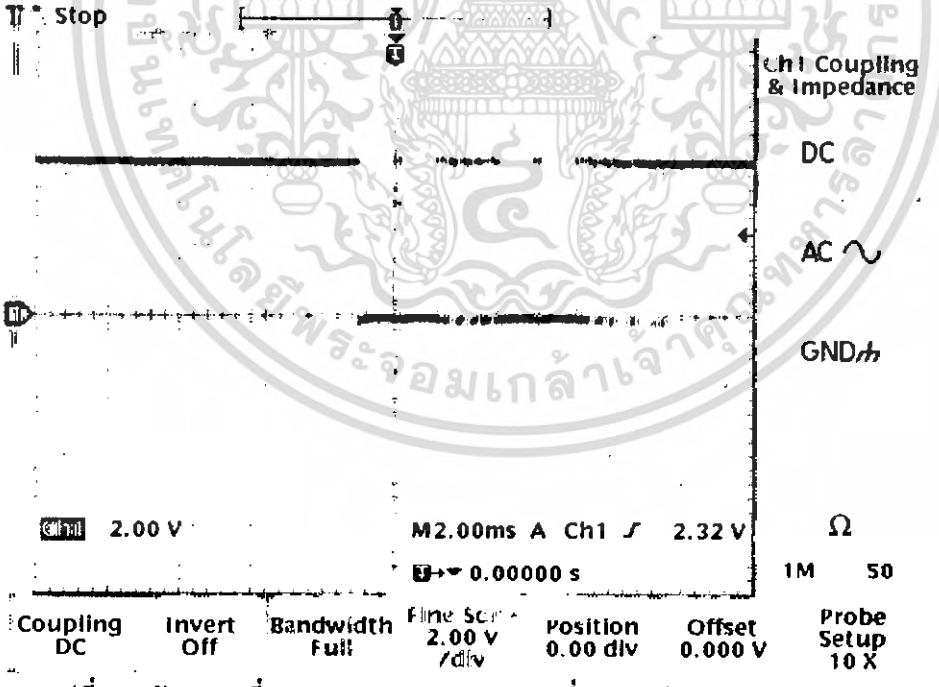
รูปที่ 4.1 สัญญาณที่ขา Data out ของ DS1820 ที่อุณหภูมิ 25 องศาเซลเซียส



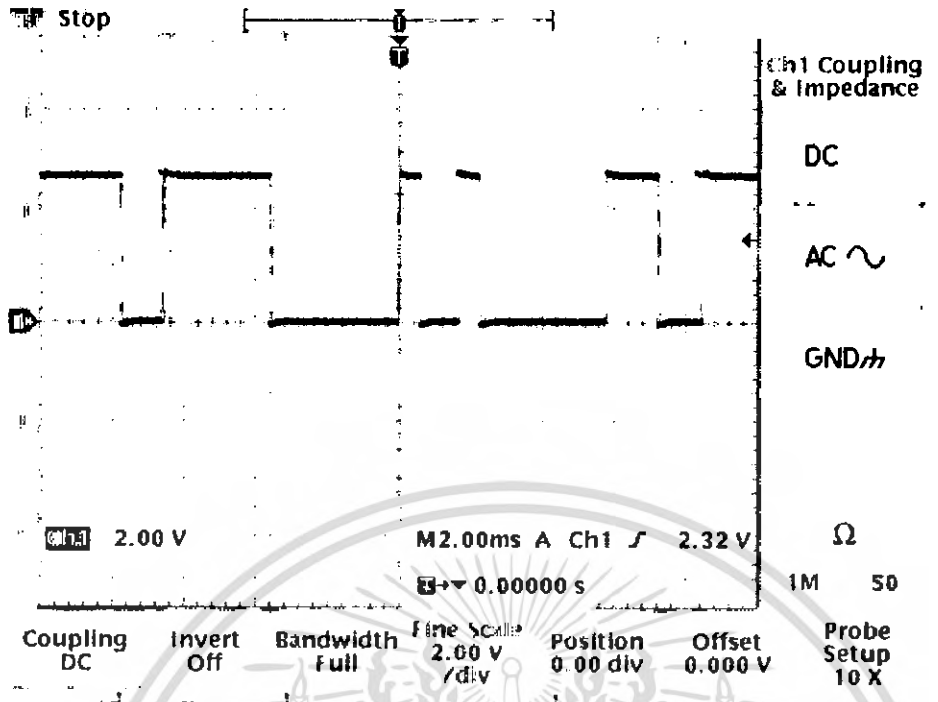
รูปที่ 4.2 สัญญาณที่ขา Tx ของ AT89C51 ที่อุณหภูมิ 25 องศาเซลเซียส



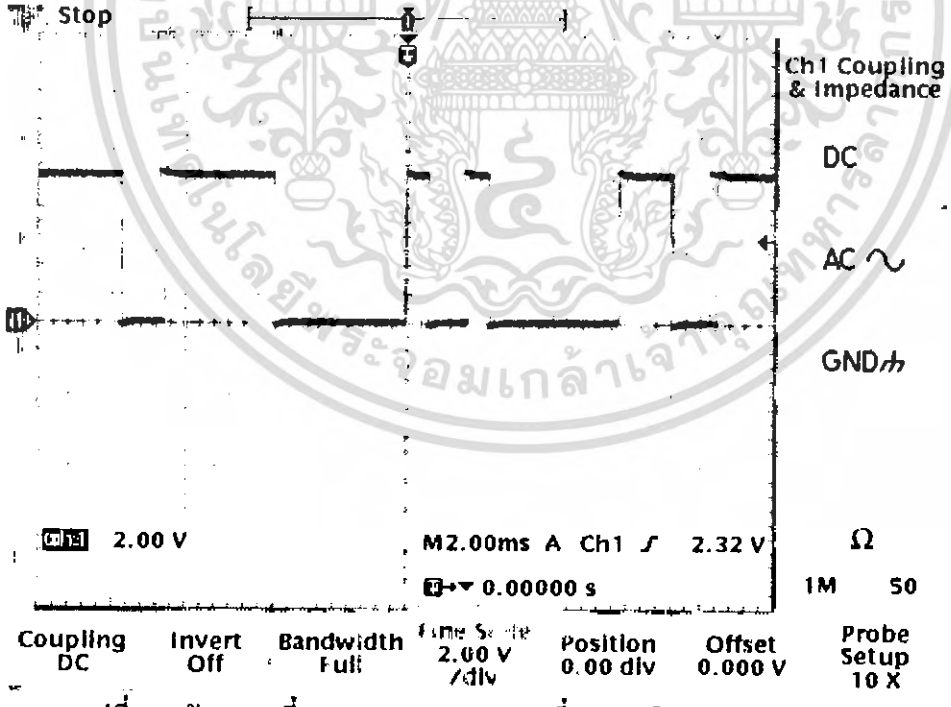
รูปที่ 4.3 สัญญาณที่ขา Rx ของ AT89C51 ที่อุณหภูมิ 25 องศาเซลเซียส



รูปที่ 4.4 สัญญาณที่ขา Data out ของ DS1820 ที่อุณหภูมิ 50 องศาเซลเซียส

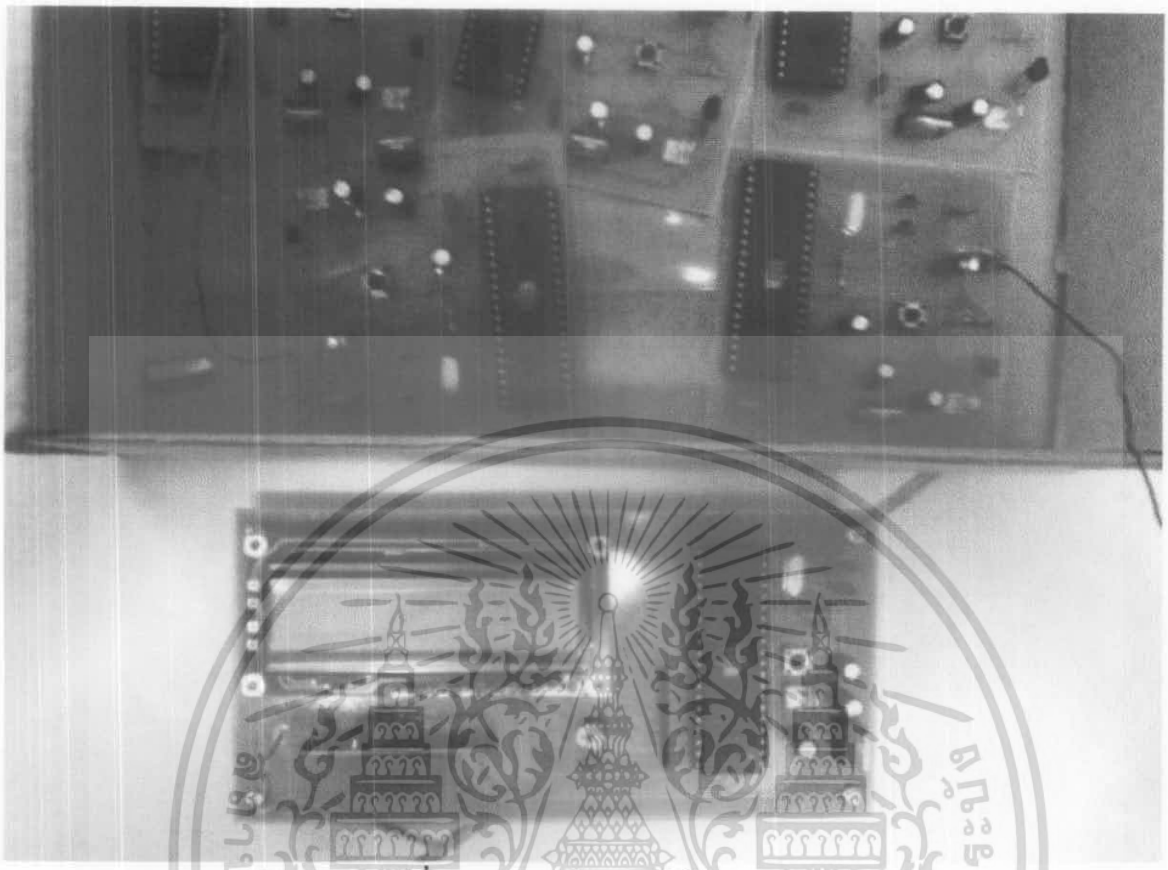


รูปที่ 4.5 สัญญาณที่ขา Tx ของ AT89C51 ที่อุณหภูมิ 50 องศาเซลเซียส



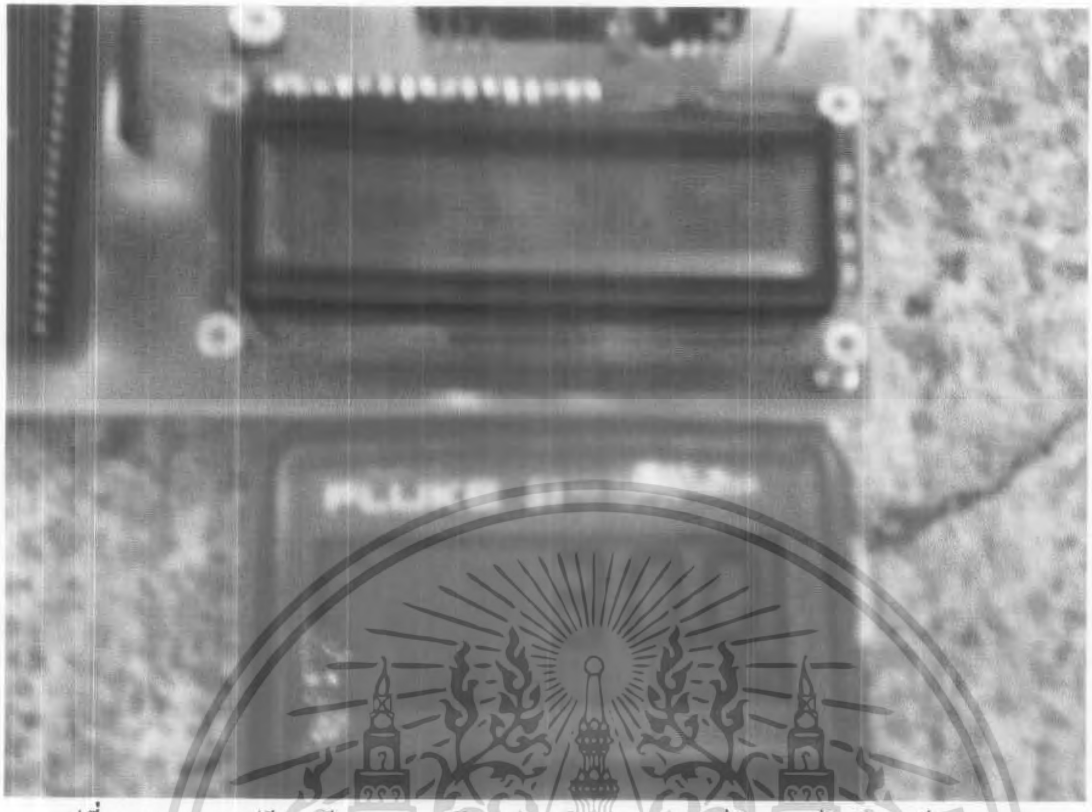
รูปที่ 4.6 สัญญาณที่ขา Rx ของ AT89C51 ที่อุณหภูมิ 50 องศาเซลเซียส

ภาพผลงาน



รูปที่ 4.7 ผลงานเซ็นเซอร์ไร้สาย

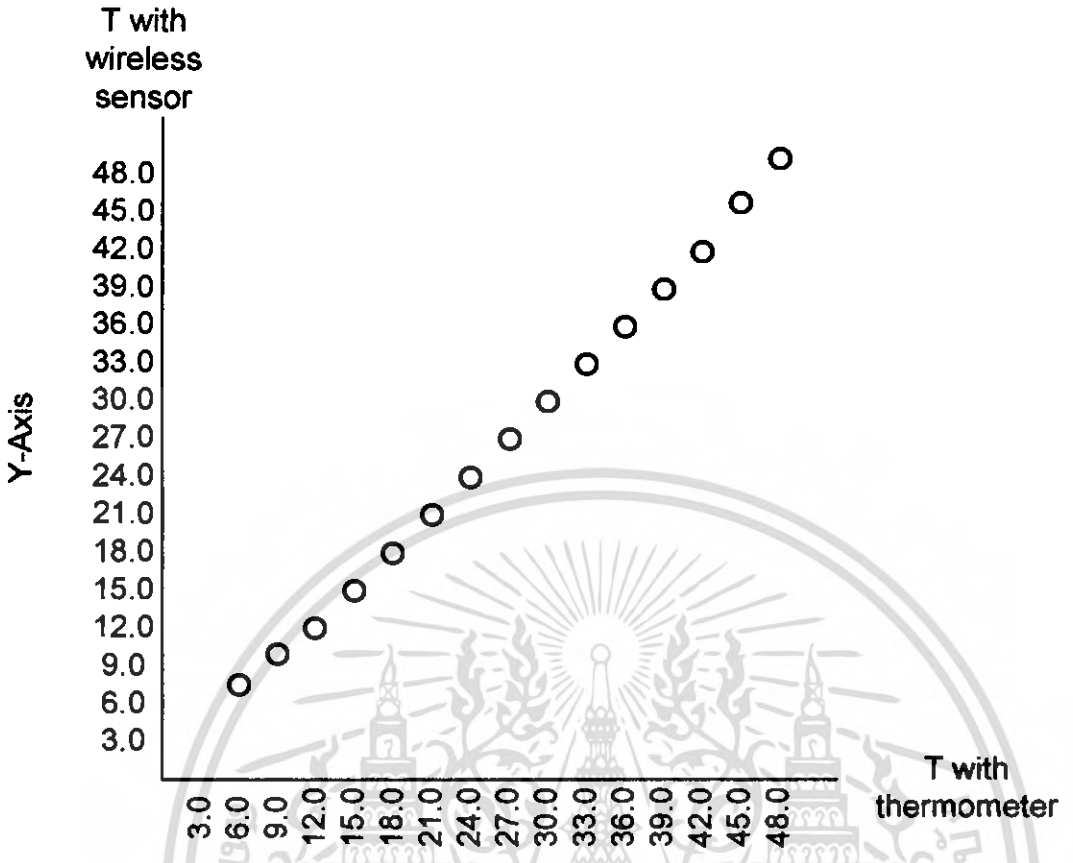
รูปที่ 4.8 เทอร์โมมิเตอร์ที่นำมาใช้เป็นมาตรฐานเปรียบเทียบ



รูปที่ 4. 9 ผลการเปรียบเทียบผลการวัดอุณหภูมิของเซนเซอร์กับเทอร์โมมิเตอร์มาตรฐาน

อุณหภูมิที่วัดได้จากเทอร์โมมิเตอร์	อุณหภูมิที่วัดได้จากผลงาน	อุณหภูมิที่วัดได้จากเทอร์โมมิเตอร์	อุณหภูมิที่วัดได้จากผลงาน
6.0	7.0	30.0	30.0
9.0	10.0	33.0	33.0
12.0	12.0	36.0	36.0
15.0	15.0	39.0	39.0
18.0	18.0	42.0	42.0
21.0	21.0	45.0	46.0
24.0	24.0	48.0	49.0
27.0	27.0		

ตารางที่5 ตารางผลการทดลองวัดอุณหภูมิระหว่างใช้เทอร์โมมิเตอร์กับชิ้นผลงาน



รูปที่4.10 กราฟความสัมพันธ์ระหว่างผลการวัดอุณหภูมิ

ระยะทางไกลสุดที่สามารถรับค่าได้ 20 เมตร