สถานีหยั่งสมุดกลาง พระจอมเกล้าลาดกระบัง

อุปกรณ์รวบรวมข้อมูลเคลื่อนที่
PORTABLE DATA LOGGER

โดย
นายนพพณ เลิศชูวงศา
นายนิวัต พุฒิโชติ
นายปกรณ์ อรัณยกานนท์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2541

อุปกรณ์รวบรวมข้อมูลเคลื่อนที่

PORTABLE DATA LOGGER

โดย

นายนพพณ เลิศชูวงศา 38014222

นายนิวัต พุฒิโชติ 38014249

นายปกรณ์ อรัณยกานนท์ 38014266

อาจารย์ที่ปรึกษา

ดร. สุทธิชัย นพนาคีพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **อุปกรณ์รวบรวมข้อมูลเคลื่อนที่**

PORTABLE DATA LOGGER

ผู้จัดทำ **1.นายนพพณ เลิศชูวงศา**

2.นายนิวัต พุฒิโชติ

3.นายปกรณ์ อรัณยกานนท์

..... อาจารย์ที่ปรึกษา

(ดร.สุทธิชัย นพนาคีพงษ์)

อุปกรณ์รวบรวมข้อมูลเคลื่อนที่
PORTABLE DATA LOGGER

โดย นายนพพณ เลิศชูวงศา 38014222
นายนิวัต พุฒิโชติ 38014249
นายปกรณ์ อรัณยกานนท์ 38014266

อาจารย์ที่ปรึกษา ดร.สุทธิชัย นพนาคีพงษ์

บทคัดย่อ

อุปกรณ์รวบรวมข้อมูลเคลื่อนที่ ทำหน้าที่บันทึกความเปลี่ยนแปลงของสภาวะแวดล้อมเช่น อุณหภูมิ เป็นต้น โดยบันทึกสัญญาณจากตัวตรวจจับ (sensor) เพื่อนำข้อมูลความเปลี่ยนแปลงเหล่านี้ถ่ายโอนให้กับคอมพิวเตอร์ส่วนบุคคล ซึ่งมีหลักการทำงานคือ นำสัญญาณอนาล็อกจากตัวตรวจจับมาแปลงเป็นข้อมูลดิจิตอล ณ.ช่วงเวลาที่กำหนดไว้แล้วเก็บข้อมูลนี้ไว้ในหน่วยความจำ (memory) หลังจากทำการบันทึกข้อมูลเสร็จแล้วก็จะถูกนำไปเชื่อมต่อกับเครื่องคอมพิวเตอร์ส่วนบุคคลเพื่อถ่ายโอนข้อมูลให้กับเครื่องคอมพิวเตอร์เพื่อนำข้อมูลนี้ไปวิเคราะห์และใช้งานต่อไป

ABSTRACT

The portable data logger's function is used to record the alteration of environment such as temperature by recording the output signal of a sensor for transfering the data of these alteration to a personal computer. It operates by converting the analog signal from a sensor to digital data at fixed interval and store the data into memory. After finished the record it will be connected to a personal computer to allow and the collected data will be transfered and analysed and used.

สารบัญ

สารบัญรูปภาพ

หน้า

หน้า

สารบัญตาราง

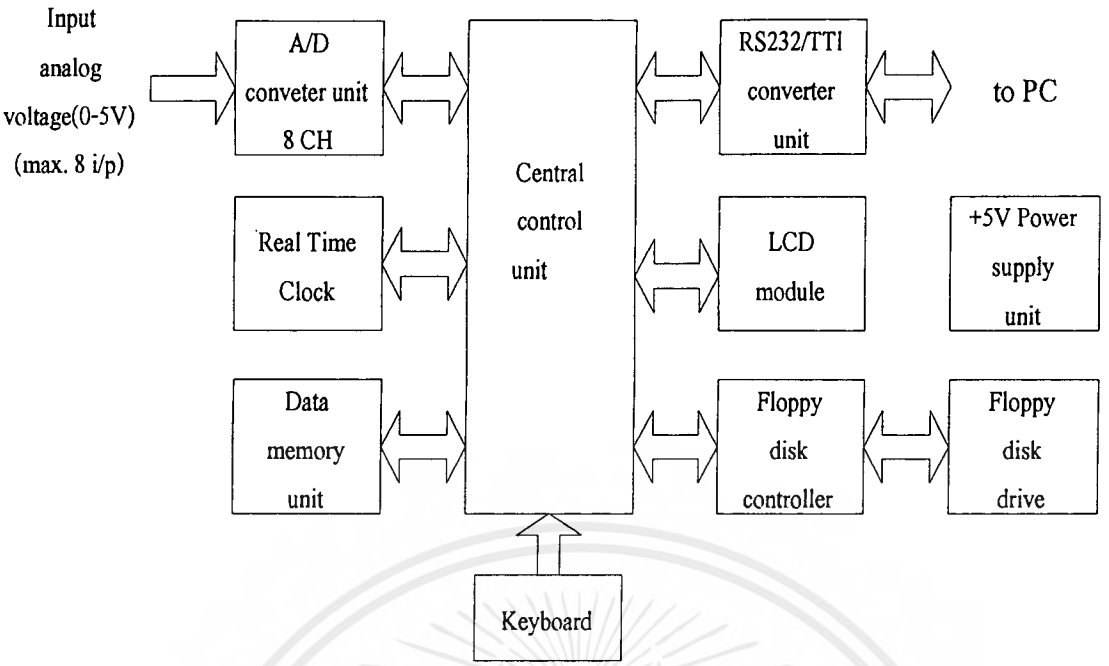
หน้า

บทที่ 1
บทนำ

ในกระบวนการทำงานต่าง ๆ ทางวิศวกรรมจะพบว่าสภาวะแวดล้อมมีผลต่อกระบวนการทำงานดังกล่าว ตัวอย่างเช่น อุณหภูมิ ปริมาณน้ำฝน ความชื้นในบรรยากาศเป็นต้น ดังนั้นจึงมีการวัดลักษณะของสภาวะแวดล้อมและจัดเก็บข้อมูลเพื่อศึกษาผลกระทบจากสภาวะแวดล้อมดังกล่าว ซึ่งจากการใช้เครื่องมือหรืออุปกรณ์วัดแบบเดิมผู้วัดต้องเฝ้ารอวัดตามช่วงเวลาที่กำหนดวัด ซึ่งในการวัดที่ต้องการผลแบบสุ่มวัดต่อเนื่องเป็นช่วงเวลายาวนานจะเป็นการยุ่งยากและเสียเวลา ดังนั้นจึงได้เกิดแนวความคิดขึ้นในการจัดเก็บข้อมูลจากการวัดที่ผู้วัดสามารถกำหนดช่วงเวลาการวัดและบันทึกได้ ยังผลให้เกิดความสะดวกในการเก็บข้อมูลและสะดวกในการวัดไม่ต้องเฝ้าทำการวัด โดยใช้คอมพิวเตอร์ส่วนบุคคล (personal computer) เป็นอุปกรณ์ช่วยในการจัดเก็บข้อมูล แต่ก็ยังเกิดข้อจำกัดขึ้น เช่น ไม่สามารถใช้งานภายนอกสถานที่ที่ไม่มีไฟฟ้า ความไม่สะดวกในเรื่องขนาดของเครื่องคอมพิวเตอร์ เครื่องคอมพิวเตอร์แบบพกพา (notebook) มีราคาแพง เป็นต้น

เพราะฉะนั้น โครงงานนี้จึงสร้างขึ้นเพื่อแก้ไขข้อจำกัดดังกล่าวโดยทำหน้าที่รับข้อมูลที่เป็นสัญญาณอนาล็อกในรูปโวลท์เตจจากเซ็นเซอร์ (sensor) มาแปลงให้เป็นสัญญาณดิจิตอลแล้วบันทึกข้อมูลนี้ไว้ในหน่วยความจำ แล้วถ่ายโอนข้อมูลดังกล่าวให้กับคอมพิวเตอร์ส่วนบุคคลหลังจากทำการวัดเสร็จแล้ว เพื่อเก็บข้อมูลและนำผลข้อมูลการวัดต่าง ๆ ไปใช้งานตามความต้องการต่อไป

จากที่กล่าวมาข้างต้น โครงงานนี้ประกอบด้วยส่วนต่าง ๆ ดังแสดงในบล็อกไดอะแกรมต่อไปนี้



บล็อกไดอะแกรมอุปกรณ์รวบรวมข้อมูลเคลื่อนที่

1. A-to-D converter unit ทำหน้าที่แปลงค่าสัญญาณแรงดันไฟฟ้าที่ได้จากเซ็นเซอร์ไปเป็นสัญญาณดิจิตอล เพื่อนำไปบันทึกและประมวลผลต่อไป

2. Data memory unit ทำหน้าที่เก็บค่าข้อมูลที่ได้จาก A-to-D converter unit เพื่อนำไปถ่ายโอนให้กับคอมพิวเตอร์ส่วนบุคคลต่อไป

3. RS232/TTL converter unit ทำหน้าที่รับ-ส่งข้อมูลกับคอมพิวเตอร์ส่วนบุคคลตามระบบการสื่อสารแบบอนุกรมมาตรฐาน RS232 โดยแปลงสัญญาณแบบ TTL ให้เป็นสัญญาณมาตรฐาน RS232 และแปลงสัญญาณจากมาตรฐาน RS232 ให้เป็นสัญญาณแบบ TTL

4. Real Time Clock ทำหน้าที่เป็นฐานเวลาให้แก่ระบบโดยจะเก็บข้อมูลที่เป็นฐานเวลาทั้งหมดได้แก่ เวลา วัน เดือน ปี

5. Central Control unit ทำหน้าที่ควบคุมการทำงานของ A-to-D converter unit ควบคุมการจัดเก็บข้อมูลใน Data memory unit และควบคุมการรับ-ส่งข้อมูลกับคอมพิวเตอร์ ส่วนบุคคล

6. +5V Power supply unit ทำหน้าที่จ่ายไฟให้แก่ระบบโดยแปลงไฟ 220V หรือจากแบตเตอรี่ 9 - 15V ให้เป็นไฟ +5V เพื่อการใช้งานได้ทุกสถานที่

7. Floppy Disk Controller ทำหน้าที่ควบคุมการติดต่อระหว่าง Central Control unit กับ Floppy Disk Drive

8. FloppyDisk Drive ทำหน้าที่จัดเก็บข้อมูลลงในแผ่นดิสเกตต์

9. LCD Display ทำหน้าที่แสดงผลการทำงานและโต้ตอบกับผู้ใช้

10. Key Board ทำหน้าที่รับคำสั่งและค่าตัวแปรต่าง ๆ จากผู้ใช้

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 Analog to Digital Converter

วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (analog to digital converter) จะทำหน้าที่แปลงสัญญาณอนาล็อกที่เป็นแรงดันไฟฟ้าให้เป็นสัญญาณดิจิตอล เราเรียกวงจรดังกล่าวย่อ ๆ ว่า เอดีซี (ADC)

เอดีซี มีหลายชนิดด้วยกัน เช่น

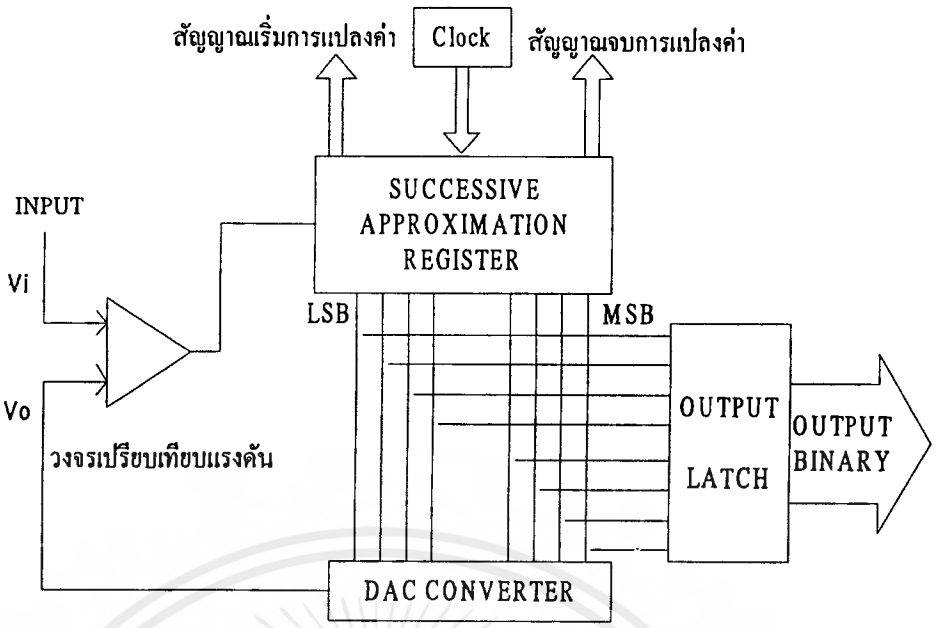
- Dual Slope Type
- Successive Approximation Register Type (SAR)
- Flash Type
- Tracking Type

สำหรับวิธีการแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลนั้นมีมากมายหลายแบบ แต่หาก แบ่งตามความเร็วที่ใช้ในการแปลงสัญญาณมี 3 วิธี ดังแสดงคุณสมบัติของแต่ละวิธีในตารางที่ 2.1

ตารางที่ 2.1 แสดงการเปรียบเทียบวิธีการแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

วิธี	ความเร็ว	ช่วงเวลาการแปลงสัญญาณใน 1รอบ
รวบรวมค่า (integrating)	ช้า	มิลลิวินาที
ประมาณค่าต่อเนื่อง (successive approximation)	เร็ว	ไมโครวินาที
แฟลช (flash)	เร็วมาก	นาโนวินาที

วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลในโครงงานนี้ใช้วิธีประมาณค่าแบบต่อเนื่อง (successive approximation) ซึ่งประกอบด้วย วงจรดีเอซี (DAC: Digital to Analog Converter) วงจรเปรียบเทียบแรงดัน และวงจรรีจิสเตอร์เก็บค่าที่ได้หลังจากการประมาณค่าสัญญาณอินพุทที่รับเข้ามา (SAR: Successive Approximation Register) ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 แสดงส่วนประกอบของวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล

<u>การทำงานของวงจร</u>

การแปลงค่าในแต่ละรอบจะเริ่มขึ้นเมื่อวงจรได้รับสัญญาณเริ่มต้น วงจรรีจิสเตอร์ส่งค่าดิจิตอลที่ได้ประมาณค่าแล้วออกไปยังวงจรดีเอซี เพื่อทำการแปลงค่าเป็นสัญญาณอนาล็อก V_o ส่งกลับมาเปรียบเทียบกับค่า V_i จาก อินพุทว่าค่าใดมากกว่ากัน เพื่อนำไปปรับค่าสัญญาณดิจิตอลแต่ละบิทให้กับรีจิสเตอร์ให้ถูกต้องกับค่าที่ป้อนเข้ามาทางอินพุท การเปรียบเทียบเริ่มจากบิทสูงมายังบิทต่ำเสมอ เมื่อครบทุกบิทวงจร SAR จะส่งสัญญาณสิ้นสุดกระบวนการออกไปและได้สัญญาณดิจิตอลเอาท์พุทที่สัมพันธ์กับค่า V_i ทางด้านอินพุท

หลักการประมาณค่าจะทำโดยวิธีการชั่งน้ำหนัก สมมติว่าเอดีซีเป็นขนาด 3 บิท เทียบกับค่าตัวถ่วงน้ำหนักเป็น 4 กิโลกรัม 2 กิโลกรัม และ 1 กิโลกรัม ตามลำดับ สำหรับลอจิกจาก 111 ถึง 000 สมมติว่า V_i เป็น 6.5 โวลท์ มีน้ำหนัก 6.5 กิโลกรัม นำของหนัก 6.5 กิโลกรัมนี้ขึ้นวางบนตาชั่ง เอาตัวถ่วงน้ำหนักขนาด 4 กิโลกรัมมาถ่วงตาชั่ง พบว่าน้ำหนัก 6.5 กิโลกรัมมีค่ามากกว่า 4 กิโลกรัม จะต้องเซตค่าบิทสูงสุดของ SAR คือ D_2 เป็นลอจิก "1"

จากนั้นหากเพิ่มตัวถ่วงน้ำหนักที่มีขนาด 2 กิโลกรัมเข้าไปจะได้น้ำหนักถ่วงรวมเป็น 6 กิโลกรัม (4+2) เปรียบเทียบกับ 6.5 กิโลกรัม ปรากฏว่าของที่นำมาวางบนตาชั่งนี้มีน้ำหนักมากกว่าอีก วงจร SAR จะเซตค่าบิท รองลงมาคือ D_1 เป็นลอจิก "1" ครั้งสุดท้ายนั้นทดลองเพิ่มน้ำหนักตัวถ่วงอีก 1 กิโลกรัม ได้น้ำหนักตัวถ่วงรวมเป็น 7 กิโลกรัม (4+2+1) เมื่อเทียบกับ 6.5 กิโลกรัมปรากฏว่าของที่นำมาวางบนตาชั่งมีน้ำหนักน้อยกว่า 7 กิโลกรัม วงจร SAR จะกำหนดค่าบิทต่ำสุดคือ D_0 เป็นลอจิก "0" ได้ลอจิกเอาท์พุทเป็นลอจิก "110" ซึ่งเป็นค่าที่สัมพันธ์กับค่าสัญญาณอนาล็อกทางด้านอินพุท

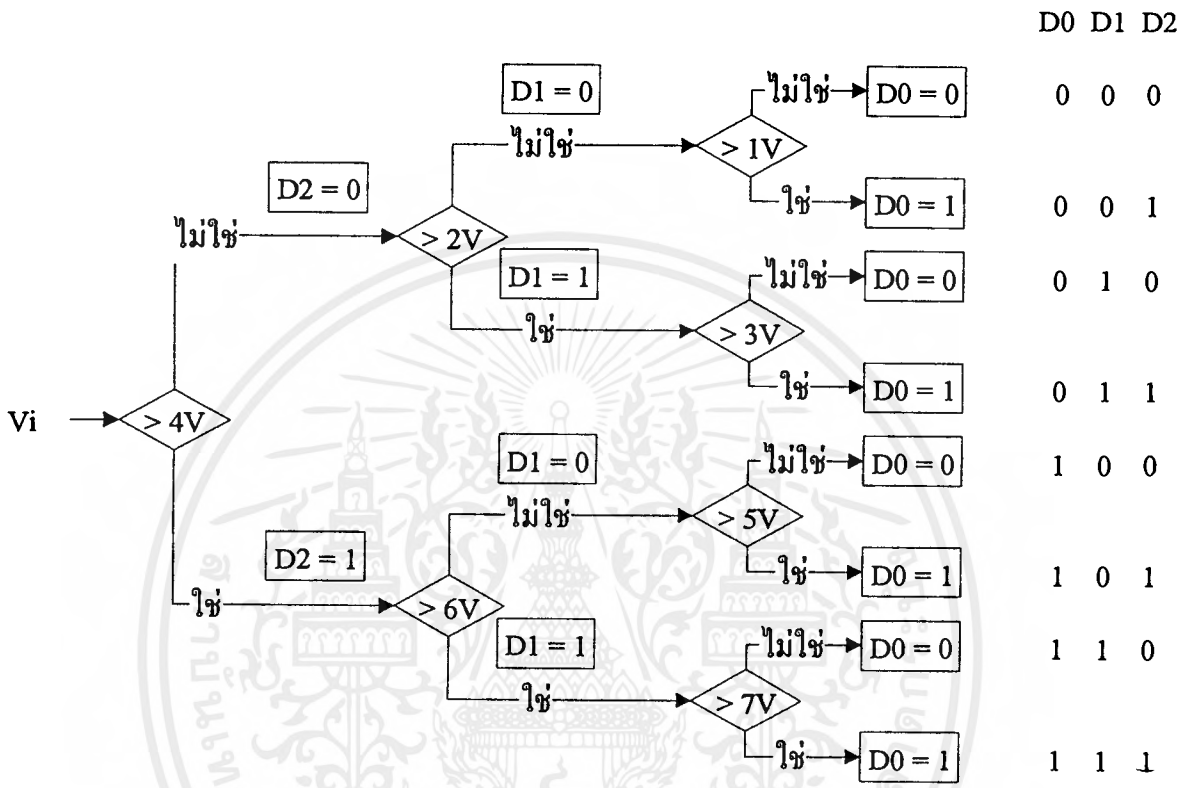
สำหรับเวลาที่ใช้ในการแปลงค่า จะแปรผันตามจำนวนบิทและความถี่ของสัญญาณคล็อก ซึ่งหาได้จากสูตร

$T_c = T(n+1)$

เมื่อ T_c คือ ค่าช่วงเวลาที่ใช้ในการแปลงค่า 1 รอบ

T คือ ค่าช่วงเวลา 1 คาบของสัญญาณคล็อก

n คือ จำนวนบิท



รูปที่ 2.2 แสดงแผนผังการประมาณค่าต่อเนื่อง

2.2 การนำอุปกรณ์รวบรวมข้อมูลเคลื่อนที่ไปประยุกต์ใช้งาน

อุปกรณ์รวบรวมข้อมูลเคลื่อนที่สามารถนำไปประยุกต์ใช้งานโดยนำไปใช้จัดเก็บข้อมูลจากอุปกรณ์วัด (sensor) ต่าง ๆ ซึ่งสามารถนำไปใช้งานภายนอกสถานที่ที่ไม่มีไฟฟ้าเข้าถึงได้

2.2.1. การวัดค่าของอุณหภูมิ

2.2.1.1ข้อกำหนดทั่วไป

การวัดค่าของอุณหภูมิเพื่อความมุ่งหมายทางอุตุนิยมวิทยามีอยู่ 3 อย่างด้วยกันคือ

-วัดอุณหภูมิอากาศ (Air)

-วัดอุณหภูมิของพื้นดิน (Soil)

-วัดอุณหภูมิของน้ำทะเล (Sea)

หน่วยในการวัดมีอยู่ 2 มาตราคือ องศาเซลเซียส กับ ฟาเรนไฮต์ การอ่านค่าของอุณหภูมิต้องให้ได้ใกล้เคียง 1/10 องศามากที่สุด

การวัดค่าของอุณภูมิผิวพื้นของอากาศ (free air) กำหนดให้วัดสูงจากพื้นดิน 1.25 ถึง 2.00 เมตร สำหรับการวัดอุณหภูมิของพื้นดินกำหนดความลึกเป็นมาตรฐานเดียวกันคือ 5,10,20,50 และ 100 เซนติเมตรใต้ผิวดิน สถานีตรวจอากาศเกษตรควรมีเครื่องบันทึกรายงานอุณหภูมิใต้ดินต่อเนื่องกันด้วย นอกจากนี้ยังต้องมีเครื่องบันทึกอุณหภูมิอากาศระดับต่าง ๆ ใกล้ผิวดินสูงขึ้นไปจนถึงประมาณ 10 เมตร ด้วย

สำหรับการวัดอุณหภูมิของน้ำทะเล ปกติจะวัดที่ระดับผิวหน้าน้ำทะเล (sea surface temperature)

2.2.1.2 การติดตั้งเครื่องมือ (Thermometer exposure)

การวัดอุณหภูมิของอากาศ ตัวเทอร์โมมิเตอร์ที่ใช้ต้องไม่ได้กับความกระทบกระเทือนจากแสงแดด จากท้องฟ้า จากโลก และสิ่งอื่น ๆ โดยรอบ แต่ในขณะเดียวกันต้องมีการถ่ายเทของอากาศที่ดีพอ วิธีป้องกันที่ดีที่สุดมีอยู่ 2 วิธีคือ วิธีแรกใช้เรือนเทอร์โมมิเตอร์แบบเป็นบานเกล็ด 2 ชั้น วิธีที่สองใช้โลหะขัดมันดังที่ใช้กับไซโครมิเตอร์แบบแอสมันน์ (Assmann) การติดตั้งเครื่องมือทั้ง 2 วิธีต้องแน่ใจว่าการวัดค่าของอุณหภูมินั้นได้ค่าของอุณหภูมิพื้นผิวของอากาศจริง ๆ มิได้มีอิทธิพลของสิ่งอื่นมารบกวน เป็นต้นว่า ตึกอาคารใหญ่ ๆ หรือพื้นลานคอนกรีตกว้าง ๆ อีกประการหนึ่งพื้นดินที่วัดต้องเป็นหญ้าที่ตัดสั้น ๆ หรือเป็นพื้นดินโดยธรรมชาติของมันเอง

2.2.1.3 อัตราไวของความรู้สึกของเทอร์โมมิเตอร์ (Speed of response of thermometer)

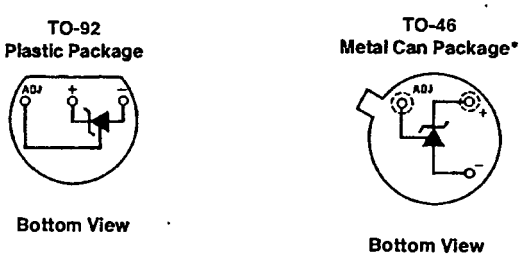
ในการตรวจอุตุนิยมวิทยาประจำวันขณะนี้ ยังไม่มีเทอร์โมมิเตอร์ที่มีความรู้สึกไวมาก ๆ โดยปกติอุณหภูมิของอากาศจะเปลี่ยนแปลงต่อเนื่องกันไปถึง 1 หรือ 2 องศาภายในระยะเวลา 2-3 วินาที เพื่อที่จะให้ได้รับอุณหภูมิที่แท้จริงจำเป็นต้องหาค่าเฉลี่ยจากการอ่านหลาย ๆ ครั้ง ซึ่งวิธีนี้เทอร์โมมิเตอร์ที่มีความรู้สึกเฉื่อยชามาก ๆ จะทำให้ค่าเปลี่ยนแปลงอย่างรวดเร็วนั้นสม่ำเสมอดียิ่งขึ้น

2.2.1.4 การติดตั้งเทอร์โมมิเตอร์

การติดตั้งเทอร์โมมิเตอร์เป็นสิ่งสำคัญ เพราะแก้วเป็นวัตถุทึบย่อมดูดและคายความร้อนได้ดีกว่าอากาศซึ่งเป็นแก๊ส ถ้าแก้วถูกแสงแดดจะร้อนขึ้นกว่าอากาศมาก ครั้นกลางคืนหลอดแก้วคลายความร้อนได้เร็วกว่าอากาศ ค่าของอุณหภูมิที่อ่านได้ก็จะคลาดเคลื่อนไป ฉะนั้นจึงต้องหาวิธีป้องกันแสงแดดและการแผ่ความร้อนนี้ออกเสีย แต่ต้องให้ตัวเทอร์โมมิเตอร์ถูกกับอากาศอย่างแท้จริง

2.2.2 ลักษณะของตัวตรวจวัดอุณหภูมิ

การวัดอุณหภูมิในปัจจุบันมีหลายประเภท เช่น การใช้เทอร์โมมิเตอร์แบบปรอท แต่ก็มีข้อเสียก็คือ อ่านอุณหภูมิได้ยาก และใช้วัดอุณหภูมิที่จุดห่างออกไปไม่ได้ หรือเทอร์โมมิเตอร์ที่ใช้หลักการขยายตัวของโลหะต่างชนิดกันโดยขดอยู่ในลักษณะของก้นหอย ซึ่งสามารถใช้ได้สะดวกแต่มีข้อเสียคือขาดความเที่ยงตรง ส่วนเทอร์โมมิเตอร์แบบอิเล็กทรอนิกส์นั้น จะแสดงค่าอุณหภูมิออกมาในรูปของตัวเลขซึ่งในโครงงานนี้ได้ใช้ ไอซี LM335

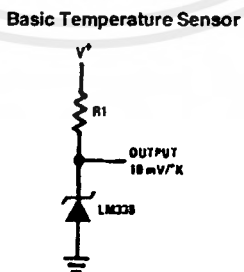


รูปที่ 2.3 ลักษณะรูปร่างและการต่อขาของ LM335

LM335 เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ที่ออกแบบมาสำหรับการตรวจจับอุณหภูมิซึ่งใช้อยู่ในย่านอุณหภูมิตั้งแต่ -40 ถึง 100 องศาเซลเซียส โดยออกแบบมาอยู่ในตัวถังพลาสติกสีดำ ซึ่งมีลักษณะการต่อขาดังรูปที่ 2.3

โดยพื้นฐานแล้ว LM335 มีหลักการทำงานที่คล้ายกับซีเนอร์ไดโอด (zener diode) ดังแสดงในรูปที่ 2.4 โดยแรงดันพังทลาย ซึ่งหมายถึงแรงดันขาออก (voltage output) จากวงจรจะแปรค่าโดยตรงตามค่าอุณหภูมิสัมบูรณ์ โดยมีค่าเท่ากับ 10 มิลลิโวลท์ต่อองศาเคลวิน (mV/K) ในย่านอุณหภูมิที่ออกแบบมาให้ใช้งาน

ค่าของตัวต้านทาน 1 ในรูปที่ 2.4 จะทำหน้าที่ เป็นตัวกำหนดกระแสที่ไหลผ่านอุปกรณ์ตัวนี้ แต่เนื่องจากค่าไดนามิกอิมพิแดนซ์ที่กระแส 1 มิลลิแอมป์ จะมีค่าประมาณ 0.6 โอห์ม อุปกรณ์ตัวนี้จึงสามารถทำงานได้ในย่านกระแสตั้งแต่ 400 ไมโครแอมป์ ถึง 5 มิลลิแอมป์ มีข้อน่าสังเกตคือค่ากระแสฟอร์เวอร์ด (forward current) หรือกระแสรีเวอร์ส (reverse current) ซึ่งไหลผ่านอุปกรณ์ตัวนี้อย่างปลอดภัยมีค่าสูงสุดชั่วขณะหนึ่งไม่ควรเกิน 10 มิลลิแอมป์ ถ้าหากกระแสที่ไหลผ่านมากกว่านี้จะทำให้ตัวไอซีเสียหายได้



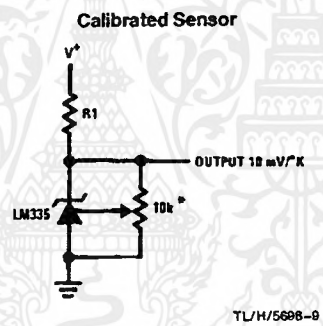
รูปที่ 2.4 วงจรพื้นฐานในการใช้งานของ LM335

ที่อุณหภูมิที่ 25 องศาเซลเซียส และ ที่กระแสรีเวอร์ส 1 มิลลิแอมป์ แรงดันเอาท์พุทจากวงจรในรูปที่ 2.4 จะมีค่าแรงดันเท่ากับ 2.98 โวลท์ ซึ่งได้จาก (273+25) ×0.001 โดยมีข้อจำกัดทางแรงดันต่ำสุดและสูงสุด มีค่าเท่ากับ 2.98 โวลท์ และ 3.04 โวลท์ ตามลำดับ ค่าของ R1 ที่ใช้สามารถคำนวณได้จากกระแสที่ไหลผ่านไอซีตัวนี้เท่ากับ 1 มิลลิแอมป์ โดยใช้สมการ

$$R1 = (V\text{-}Vout)/0.001$$

โดยที่ V คือแรงดันไฟเลี้ยงวงจรและ Vout คือแรงดันเอาท์พุทของไอซีนี้ ซึ่ง R1 นี้จะมีค่าประมาณ (V-3)K ประโยชน์ที่สำคัญของไอซี LM335 นี้ก็คือ จะให้เอาท์พุทที่เป็นเชิงเส้นซึ่งไม่เหมือนกับเอาท์พุทที่ได้จากการตรวจจับอุณหภูมิส่วนใหญ่ซึ่งเอาท์พุทจะไม่เป็นเชิงเส้นกับอุณหภูมิ ซึ่งที่จริงแล้วถ้าเรานำเอาแรงดันเอาท์พุทมาเขียนกราฟระหว่างค่าแรงดัน กับอุณหภูมิตลอดย่านการทำงานของไอซีตัวนี้ เมื่อลากเส้นกราฟไปตัดแกนอุณหภูมิที่ 0 องศาสัมบูรณ์ ค่าแรงดันที่อ่านได้จากกราฟจะเป็น 0 โวลท์

แม้ว่าเอาท์พุทของ LM335 จากวงจรในรูปที่ 2.4 จะอยู่ในข้อจำกัดที่กำหนดมา เราก็ควรจะมีการต่อขาสำหรับปรับแต่งความถูกต้องโดยการต่อตัวต้านทานปรับค่าเพียงตัวเดียวคร่อม LM335 ดังแสดงในรูปที่ 2.5 ในการใช้งานให้ปรับค่า VR1 ให้เอาท์พุทเท่ากับ 2.982 โวลท์ เมื่ออุณหภูมิอยู่ที่ 25 องศาเซลเซียส เพื่อที่จะทำให้เกิดความเที่ยงตรงเพิ่มขึ้น ตลอดย่านอุณหภูมิที่ใช้งาน



รูปที่ 2.5 การปรับแต่งความถูกต้องโดยใช้ตัวต้านทานปรับค่าได้

การปรับความถูกต้องของอุณหภูมิเพียงครั้งเดียว จะทำให้ถูกต้องตลอดย่านอุณหภูมิที่ใช้งานนั้น เนื่องจากเอาท์พุทจะแปรผันเป็นสัดส่วนโดยตรงกับอุณหภูมิสัมบูรณ์ โดยเอาท์พุทจะ ลดลงเป็น 0 โวลท์ที่อุณหภูมิ 0 องศาสัมบูรณ์ ดังนั้นการปรับค่าความลาดชันที่อุณหภูมิค่าหนึ่งให้ถูกต้องจะทำให้เกิดความถูกต้องตลอดย่านอุณหภูมิ ซึ่งการปรับค่าความถูกต้องจะทำได้ง่ายกว่าพวกอุปกรณ์ที่ไม่เป็นเชิงเส้น เช่น เทอร์โมคัปเปิล (Thermocouple) เป็นต้น

2.2.2.1 ความร้อนที่เกิดขึ้นในตัว

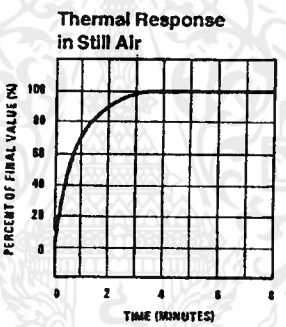
ไม่ว่าระบบตรวจจับอุณหภูมิใด ๆ ก็ตาม ความร้อนใด ๆ ที่เกิดขึ้นจากกระแสที่ไหลผ่านอุปกรณ์ที่ตรวจจับจะมีผลต่อค่าอุณหภูมิของตัวอุปกรณ์ ตลอดจนแรงดันเอาท์พุทที่เกิดขึ้น สำหรับ LM335 นั้นควรจะให้ทำงานที่กระแสต่ำสุดซึ่งเพียงพอที่จะขับให้วงจรภายในไอซีทำงานได้

ถ้าใช้อุปกรณ์ตรวจจับในสถานที่ซึ่งค่าความต้านทานทางอุณหภูมิต่อสิ่งแวดล้อมนั้นมีค่าคงที่ ค่าความผิดพลาดจากความร้อนที่เกิดขึ้นในตัวอุปกรณ์เองสามารถที่จะปรับให้ถูกต้องได้ซึ่งจะทำให้อุปกรณ์นั้นทำงานด้วยกระแสคงที่โดยไม่ขึ้นอยู่กับอุณหภูมิความร้อนที่เกิดขึ้นกับอุปกรณ์ แต่จะแปรผันโดยตรงกับแรงดันซีเนอร์และอุณหภูมิสัมบูรณ์ ดังนั้นค่าความผิดพลาดที่เกิดขึ้นจากความร้อนในตัวเองจะแปรผันโดยตรงกับค่าอุณหภูมิสัมบูรณ์และความเป็นเชิงเส้นของสเกลอุณหภูมิ

2.2.2.2 คุณสมบัติเฉพาะตัว

ในวงจรทั่วไปของ LM335 ซึ่งไม่ได้ปรับค่าความถูกต้องไว้ ให้ทำงานที่กระแส 1 มิลลิแอมป์ ค่าความผิดพลาดเนื่องจากอุณหภูมิจะเท่ากับ 2 องศา (สูงสุด 6 องศา) ที่อุณหภูมิ 25 องศาเซลเซียส หรือ 4 องศา (สูงสุด 9 องศา) ตลอดย่านการทำงาน เมื่อได้รับการปรับค่าความถูกต้องไว้ ค่าความผิดพลาดที่อุณหภูมิจำกัดไว้เป็น 2 องศา ความไม่เป็นเชิงเส้นที่ค่ากระแส 1 มิลลิแอมป์จะเท่ากับ 0.3 องศา ตลอดย่านที่ใช้งาน

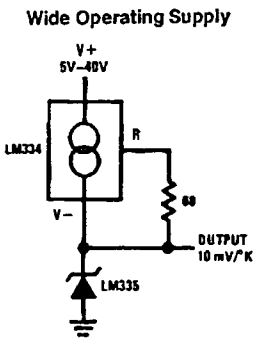
ในอากาศอุปกรณ์ตัวนี้ต้องใช้เวลาประมาณ 3 นาทีที่จะมีค่าอุณหภูมิสุดท้ายหลังจากที่อุณหภูมิได้เปลี่ยนไป (ในรูปที่ 2.6) ค่าคงที่ของเวลาจะมีค่าเท่ากับ 80 วินาที



รูปที่ 2.6 เวลาการตอบสนองของ LM335 ต่อการเปลี่ยนแปลงของอุณหภูมิในอากาศ

ในน้ำมันที่ไหลวนให้เคลื่อนที่ไปมา ค่าอุณหภูมิสุดท้ายจะถึงภายใน 3 นาที ค่าคงที่ของเวลาจะเท่ากับ 1 วินาที อุปกรณ์จะคงที่อยู่ภายในช่วง 0.2 องศาตลอด 1,000 ชั่วโมงถึงแม้ว่าที่อุณหภูมิ 125 องศาเซลเซียส

วงจรในรูปที่ 2.4 และ 2.5 เหมาะสำหรับใช้เมื่อแรงดันไฟเลี้ยงวงจรมีค่าค่อนข้างคงที่ ถ้าคาดว่าแรงดันไฟเลี้ยงวงจรจะเปลี่ยนแปลงในย่านกว้าง ควรจะใช้ LM334 ซึ่งเป็นตัวจ่ายกระแสคงที่ร่วมกับตัวต้านทานภายนอก ดังในรูปที่ 2.7 เพื่อกำหนดค่ากระแสของ LM335 ให้มีค่าประมาณ 1 มิลลิแอมป์สำหรับทุก ๆ ค่าไฟเลี้ยงของวงจร



รูปที่ 2.7 วงจรที่สามารถทำงานกับแหล่งจ่ายไฟได้ในย่านกว้าง

2.3 Serial Port Communication

จุดประสงค์หลักของการใช้ซีเรียลพอร์ตคือ การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์โดยเฉพาะการสื่อสารระยะไกลด้วยโมเด็มและโครงข่ายโทรศัพท์สาธารณะ แต่ก็สามารถนำมาใช้ในการเชื่อมต่ออุปกรณ์อื่นๆกับเครื่องคอมพิวเตอร์ เช่น เมาส์

การเชื่อมต่อแบบพาราเรลพอร์ตจะมีความเร็วและการอินเตอร์เฟสที่ง่ายแต่ว่าจำนวนขาของพอร์ตนั้นมีถึง 25 ขา (อาจจะใช้ไม่หมดทุกขา) และความยาวสายเคเบิลสูงสุดได้เพียง 5 เมตร ถึงอย่างไรก็ตามมันก็ให้ความเร็วสูงถึงประมาณ 100kbit/sec จึงเหมาะการส่งข้อมูลสำหรับเครื่องพรินเตอร์ส่วนพอร์ตแบบ PS/2 นั้นก็สามารถส่งข้อมูลได้เพียงทิศทางเดียว

ในทางตรงกันข้ามเรามาส่งข้อมูลแบบ 2 ทิศทางทางซีเรียลพอร์ตโดยใช้จำนวนขาแค่ 9 ขา (อาจจะใช้ไม่หมดทุกขา) และเพราะว่าสัญญาณไฟฟ้าที่ใช้สำหรับซีเรียลพอร์ตได้ถูกออกแบบมาสำหรับการส่งข้อมูลระยะทางไกลกว่าแบบพาราเรลพอร์ตที่ใช้แบบ TTL โวทล์เตจและก็การใช้โมเด็มทำให้ซีเรียลพอร์ตสามารถใช้กับการติดต่อระยะทางไกลมากขึ้น

การส่งข้อมูลแบบอนุกรม (serial) จะส่งข้อมูลแบบเป็นกลุ่มของบิทโดยจะส่งทีละบิท บนหนึ่งเส้นสายสัญญาณโดยข้อมูลบนสายสัญญาณจะเปลี่ยนไปตามเวลาในการส่งสัญญาณแต่ละค่าของสัญญาณไม่เหมือนกับแบบขนาน (parallel) ซึ่งจะส่งพร้อมกันทั้งกลุ่มของบิทโดยส่งไปพร้อมกันทั้งหมดในกลุ่มของสายสัญญาณหลายเส้นในเวลาเดียวกัน การส่งข้อมูลเราต้องพิจารณาถึงความหมายของคำต่อไปนี้

Transmitter : ตัวส่ง

Receiver : ตัวรับ

Transmission medium : ซึ่งจะเป็นเส้นทางในการส่งข้อมูลจาก Transmitter ไปยัง Receiver ซึ่งสิ่งสำคัญที่จะมีผลต่อการพิจารณาตัวกลางการส่งข้อมูลดังนี้

1. ลักษณะทางกายภาพของตัวกลาง ซึ่งโดยทั่วไปจะมีที่ใช้บ่อยอยู่ดังนี้

- Directline เส้นสัญญาณประเภทนี้ก็คือการต่อเชื่อมระหว่างเครื่องคอมพิวเตอร์กันโดยตรงหรือไม่ก็เชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์อื่นโดยตรง

- Dial-up line เป็นลักษณะทางกายภาพของการเชื่อมต่อผ่านโครงข่ายสาธารณะโดยผ่านโมเด็ม สำหรับแบบ directline จะเป็นการเชื่อมต่อสัญญาณทางไฟฟ้ากันโดยตรงแต่ในแบบ dial-up line จะต้องมีการเปลี่ยนลักษณะสัญญาณก่อนผ่านไปในตัวกลาง

2. การสื่อสารแบบ ซิงโครนัส (synchronous mode) กับ แบบอะซิงโครนัส (asynchronous mode) ในการสื่อสารอนุกรมแบบซิงโครนัส (synchronous) การสื่อสารทั้งหมด (ทั้งตัวส่งและตัวรับ) จะถูกควบคุมโดยใช้นาฬิกาส่วนกลาง (master clock) และข้อมูลที่ส่งต้องสัมพันธ์กับนาฬิกานี้ หรือการให้อักขระในการซิงค์ข้อมูลจะถูกส่งในรูปบล็อกของข้อมูลที่มีอัตราเร็วที่คงที่ จุดเริ่มต้นและจุดสุดท้ายของบล็อกจะเป็นรูปแแบบบิทที่กำหนดขึ้นมาเฉพาะ สำหรับในการสื่อสารแบบอนุกรมอะซิงโครนัส จะมีการกำหนดรูปแบบในการส่งข้อมูลเป็นชุดสั้นๆที่มีบิทเริ่มต้น บิทลงท้ายและบิตของข้อมูลแต่ละชุดข้อมูลต้องคงที่ อัตราการส่งการส่งทั้งทางด้านส่งและรับต้องกำหนดให้เท่ากัน

3. ทิศทางการส่งข้อมูล แบ่งเป็น ซิมเพล็กซ์ (simplex) , ฮาร์ฟดูเพล็กซ์ (half-duplex) , ฟูลดูเพล็กซ์ (full-duplex)

ข้อเสียในการใช้สัญญาณ TTL

1. ระดับสัญญาณ TTL มักถูกเหนี่ยวนำจากสัญญาณรบกวนภายนอกได้ง่าย

2. การสูญเสีย (loss) ไปในสายทำให้ระดับแรงดันของสัญญาณที่ส่งออกไปลดลง ซึ่งมีผลกระทบที่สำคัญต่อระดับแรงดัน 0-5 โวลท์ เพราะว่าการสูญเสียระดับแรงดันไปเพียง 2- 3 โวลท์ มีผลต่อการเปลี่ยนลอจิกต่างๆที่ได้รับ

การสื่อสารโดยใช้ระดับสัญญาณ TTL มักใช้ในการสื่อสารแบบขนานซึ่งจะทำได้ยากในการส่งระยะไกล

เหตุผลเพราะว่า 1. ความเชื่อถือได้ คือ

-สัญญาณการส่งแบบขนานจะต้องไม่ถูกหน่วงให้ช้าลง

-เฟสสัญญาณจะต้องไม่เปลี่ยน

2. ราคาของสายเคเบิลแบบขนานและคอนเนคเตอร์มีราคาแพง เพราะฉะนั้นการส่งระยะไกลจึงใช้การส่งแบบอนุกรม (serial)

2.3.1 RS 232 C Physical Layer Communication protocols

คือ ข้อกำหนดที่เกี่ยวข้องกับการเชื่อมต่อทางกล (mechanical connections) คุณสมบัติทางสัญญาณไฟฟ้าและคุณสมบัติที่เกี่ยวข้องกับหน้าที่ของสัญญาณเหล่านี้ ซึ่งข้อกำหนดที่ทำขึ้นนี้ก็เพื่อให้อุปกรณ์ที่ผลิตจากบริษัทแต่ละแห่งสามารถใช้งานร่วมกันได้

ในการสื่อสารข้อมูล ข้อมูลจะถูกส่งไปมาระหว่างอุปกรณ์อิเล็คทรอนิกส์ผ่านทางการอินเตอร์เฟส ซึ่งประกอบไปด้วยสัญญาณไฟฟ้า สายเคเบิลที่เป็นตัวนำสัญญาณไฟฟ้าและคอนเนคเตอร์ ข้อมูลจะถูกส่งด้วยการเปลี่ยนแปลงของกระแสและแรงดัน การสื่อสารข้อมูลจะเป็นไปได้ก็ต่อเมื่อ อุปกรณ์ต่างๆเป็นไปตามข้อกำหนดของ physical layer protocol ซึ่งมีดังต่อไปนี้

-คุณสมบัติทางกล :ได้มีการกำหนดรายระเอียดของขนาดคอนเนคเตอร์,จำนวนขา, ฟังก์ชั่นการทำงานของขาต่าง ๆ เส้นผ่านศูนย์กลางของขาต่างๆ ตำแหน่งของคอนเนคเตอร์และคุณสมบัติของสายเคเบิล

-คุณสมบัติทางสัญญาณไฟฟ้า : มีการกำหนดคุณสมบัติทางไฟฟ้าของสัญญาณที่ควบคุมการแลกเปลี่ยนข้อมูลและวงจรไฟฟ้าต่างๆที่เกี่ยวข้อง รวมทั้งการกำหนดอัตราเร็วสูงสุดในการส่งข้อมูล ระดับกระแสและระดับแรงดันในการแทนลอจิกต่างๆ และคุณสมบัติของวงจรรับส่ง

-รายละเอียดตามหน้าที่ของสัญญาณ : สัญญาณต่างๆที่ใช้ในการอินเตอร์เฟสมักถูกกำหนดชื่อจากลักษณะการทำงาน โดยดูทิศทางการส่งของข้อมูลว่าส่งจากตัวส่งหรือตัวรับ และความสัมพันธ์ของสัญญาณ

2.3.2 คุณสมบัติของสัญญาณไฟฟ้า

สัญญาณที่ขาทุกขาของคอนเนคเตอร์ RS 232 จะเป็นสภาวะ ในสภาวะใดสภาวะหนึ่งต่อไปนี้

mark / space

on / off

logic 0 or logic 1

ความสัมพันธ์ระหว่างสถานะของสัญญาณคู่ต่างๆ กับแรงดันที่แสดงไว้ในตารางที่ 2.2

ตารางที่ 2.2 สถานะของสัญญาณกับแรงดันตามมาตรฐาน RS232

VOLTAGE RANGE	LOGIC	MEANING	TTY	SYNONYMS
+3V TO +25V	0	ON	SPACE	ACTIVE, "ASSERTED"
-25V TO -3V	1	OFF	MARK	INACTIVE, "DISASSERTED"

ในการแทนลอจิกหนึ่งหรือสถานะ mark ตัวขับสัญญาณไดเวอร์ต้องจ่ายแรงดันระหว่าง -5 ถึง -15 โวลท์ ส่วนในลอจิกศูนย์หรือ space ตัวขับสัญญาณต้องจ่ายแรงดัน +5 ถึง +15 โวลท์

ตัวเก็บประจุ ที่ต่อขนานกับอุปกรณ์รับปลายทางมีค่าไม่เกิน 2500 พิโคฟารัด (ความยาวน้อยกว่า 50ฟุต)

แรงดันขณะเปิดวงจรหรือไม่มีโหลด จะต้องไม่เกิน 25 โวลท์

วงจรรับสัญญาณต้องสามารถทนต่อการลัดวงจรที่เกินขึ้นได้ (จากการไม่ตั้งใจ) โดยไม่ทำความเสียหายให้อุปกรณ์ตัวมันเองหรือที่เกี่ยวข้อง

2.3.3 ลักษณะการทำงานของเซอร์กิตต่างๆ

เซอร์กิตต่างๆได้แสดงไว้ในตาราง ที่ 2.3 สามารถแยกออกเป็นประเภทต่างๆได้ 5 ประเภทดังต่อไปนี้

กราวนด์ , เซอร์กิตข้อมูล , เซอร์กิตควบคุม , เซอร์กิตสัญญาณเวลา , เซอร์กิตของแชลแนล (channel) ที่สอง

ตารางที่ 2.3 ลักษณะของเซอร์กิตต่าง ๆ

PIN Number for 9 PINS	PIN Number for 25 PINS	Common Name	RS-232C Name	DESCRIPTION	SIGNAL DIRECTION ON DCE
	1		AA	PROTECTIVE GROUND	-
3	2	TXD	BA	TRANSMITTED DATA	IN
2	3	RXD	BB	RECEIVED DATA	OUT
7	4	RTS	CA	REQUEST TO SEND	IN
8	5	CTS	CB	CLEAR TO SEND	OUT
6	6	DSR	CC	DATA SET READY	OUT
5	7	GND	AB	SIGNAL GROUND	-
1	8	CD	CF	RECEIVE LINE SIGNAL DETECTOR	OUT
	9		-	(REVERSE FOR DATA SET TESTING)	-
	10		-	(REVERSE FOR DATA SET TESTING)	-
	11			UNASSIGNDED	-
	12	S-CD	SCF	SECONDARY RECEIVED LINE SIGNAL DETECTOR	OUT
	13	S-CTS	SCB	SECONARY CLEAR TO SEND	OUT
	14	S-TXD	SBA	SECONDARY TRANSMITTED DATA	IN
	15		DB	TRANSMIT SIGNAL ELEMET TIMING (DCE SOURCE)	OUT
	16	S-RXD	SBB	SECONARY RECEIVE DATA	OUT
	17		DD	RECEIVER SIGNAL ELEMENT TIMING (DCE SOURCE)	OUT
	18			UNASSIGNDED	-
	19	S-RTS	SCA	SECONDARY REQUEST TO SEND	IN
4	20	DTR	CD	DATA TERMINAL READY	IN
	21		CG	SIGNAL QUANLITY DETECTOR	OUT

Number for 9 PIN	Number for 25 PIN	Common Name	RS-232C Name	DESCRITION	SIGNAL DIRECTION ON DCE
9	22		CE	RING INDICATOR	OUT
	23		CH/CI	DATA SIGNAL RATE SELECTOR (DTE/DCE SOURCE)	IN/OUT
	24		DA	TRANSMIT SIGNAL ELEMET TIMING (DCE SOURCE)	IN
	25			UNASSIGNDED	-

CIRCUIT AA : Protective Ground

ลวดตัวนำของเซอร์กิตนี้จะถูกต่อเข้ากับตัวถังอุปกรณ์เพื่อใช้เป็นสายดิน

CIRCUIT AB : Signal Ground

เซอร์กิตนี้ถูกใช้เป็นจุดอ้างอิงของสัญญาณของเซอร์กิตต่างๆ

CIRCUIT BA : Transmitted Data

สัญญาณของเซอร์กิตนี้จะถูกส่งจาก DTE ไปยัง DCE , DTE จะทำให้เซอร์กิต BA (Transmitted data) มีสถานะลอจิกเป็น 1 ตลอดเวลาที่ไม่มีการส่งข้อมูล

ในระบบทุกระบบที่ใช้มาตรฐาน RS-232C DTE จะไม่ทำการส่งข้อมูลนอกจากเซอร์กิตต่อไปนี้เป็น 0 (on)

เซอร์กิต CA (request to send)

เซอร์กิต CB (clear to send)

เซอร์กิต CC (data set ready)

เซอร์กิต CD (data terminal ready) ในบางกรณีเราอาจต้องใช้ null modem ร่วมกับเซอร์กิตนี้ เช่น ในระบบคอมพิวเตอร์เรา ต้องการเชื่อมต่อคอมพิวเตอร์ 2 เครื่องเข้าด้วยกันเพื่อให้เครื่องคอมพิวเตอร์ 2 เครื่องส่งและรับข้อมูลกันได้

CIRCUIT BB: Received Data

สัญญาณของเซอร์กิตนี้จะถูกส่งผ่านจาก DCE ไปยัง DTE เซอร์กิตนี้จะอยู่ในสถานะลอจิก 1 ตลอดเวลาที่ไม่มีการส่งข้อมูล

ในการส่งข้อมูลแบบ half-duplex เมื่อ Request to Send (เซอร์กิต CA) อยู่ในสภาวะลอจิก 0 Receive Data (เซอร์กิต BB) จะมีสภาวะลอจิก 1 นอกจากนี้ Receive Data จะอยู่ในสภาวะอีกช่วงในระยะเวลาสั้นๆ ระยะเวลาหนึ่งหลังจากที่ Request to Send เปลี่ยนจากสภาวะ on (ลอจิก 0) เป็นสภาวะoff (ลอจิกหนึ่ง)เมื่อมีการส่งข้อมูลเกิดขึ้นเรียบร้อยแล้ว

CIRCUIT CA: Request to Send

สัญญาณ Request to send จะถูกส่งจาก DTE ไปยัง DCE ลักษณะการทำงานร่วมกันของสัญญาณ Request to send (RTS) และสัญญาณ Clear to Send (CLS) ซึ่งเกิดขึ้นระหว่างการส่งข้อมูลระหว่าง DTE และ DCE

ในการส่งข้อมูลเมื่อ request to send อยู่ในสถานะของลอจิกเป็น 0 (on) จะทำให้ DCE รับข้อมูลจาก DTE บางครั้งจะต่อ Request to Send เข้ากับ Clear to Send โดยตรงทำให้เมื่อ DTE ส่ง RTS จะได้รับสัญญาณ CTS ทันที อย่างไรก็ตามที่ Request to Send เปลี่ยนสถานะจากลอจิก 0 (on) เป็นลอจิก 1 (off) แล้ว Request to Send จะ on ใหม่ได้อีกครั้งหนึ่งก็ต่อเมื่อ DCE สั่งให้ CTS เปลี่ยนสถานะจาก on เป็น off แล้ว การทำเช่นนี้เพื่อป้องกันไม่ให้เกิด overrun error (DTE ส่งข้อมูลชุดใหม่มาอีก ขณะที่ DCE ยังส่งข้อมูลชุดเก่าไม่เรียบร้อย (โมเด็มส่งข้อมูลผ่านเครือข่าย) การทำ handshake โดยใช้ Request to Send กับ Clear to Send ที่อธิบายไปนั้นใช้ได้กับการส่งข้อมูลทีละคาร์แรกเตอร์หรือทีละบล็อก เช่นหนึ่งคาแรกเตอร์ประกอบด้วยบิทต่างๆ 10 บิท เมื่อทำการส่งข้อมูลจะต้องทำให้ handshake ระหว่างข้อมูลแต่ละคาร์แรกเตอร์ เมื่อส่งข้อมูลครบ 10 บิท จะป้อนสัญญาณลงสาย Request to send และคอยรับสัญญาณจาก DCE ทางสาย Clear to Send สำหรับการทำ handshake ในการส่งข้อมูลทีละบล็อกนั้น DTE จะส่งคาร์แรกเตอร์พิเศษที่บอกจุดสิ้นสุดของบล็อกเพิ่มเข้าไปด้วย และเมื่อถึงจุดสิ้นสุดของบล็อกข้อมูล DTE จะ off สัญญาณ Request to Send ในการตอบสัญญาณต่อไปนี้ DCE จะ off สัญญาณ Clear to Send เมื่อ DCE ส่งข้อมูลออกไปยังเครือข่ายเรียบร้อยแล้ว

จากที่อธิบายไว้ในเซอร์กิต BA นั้น DTE จะส่งข้อมูลได้ก็ต่อเมื่อสัญญาณ Request to Send, Clear to Send, Data Terminal Ready และ Data set ready เหล่านี้ on (ลอจิก 0) เรียบร้อยแล้ว เมื่อ Data Terminal Ready และ Data Set Ready on เรียบร้อยแล้วสัญญาณ Request to Send จะ on ตามมา และ DCE จะตอบสนองสัญญาณนี้โดยการส่ง Clear to Send on กลับมายัง DTE , DTE จึงจะส่งข้อมูลออกมาทาง Transmitted Data ได้ (เราสามารถ on สัญญาณ RTS เมื่อไหร่ก็ได้ตราบใดที่สัญญาณ CTS off โดยไม่ต้องสนใจสัญญาณของเซอร์กิตอื่นๆ)

CIRCUIT CB: Clear to Send

สัญญาณนี้เป็นสัญญาณควบคุมที่ส่งจาก DCE ไปยัง DTE เพื่อบอกให้ DTE ทราบว่า DCE พร้อมที่จะรับข้อมูลที่ส่งมาจาก DTE บนสาย Transmitted Data แล้วเมื่อสัญญาณ Clear to Send อยู่ในสภาวะ on รวมทั้งสัญญาณ Request to Send ,Data set Ready หรือ Data Terminal Ready มีสภาวะเป็น on ด้วย การ on ของสัญญาณเหล่านี้ จะบอกให้ DTE ทราบว่าข้อมูลที่ส่งไปยัง DCE จะถูก DCE รับไว้เพื่อทำการส่งต่อไปยัง Communication Channel ต่อไป เมื่อสัญญาณ Clear to send อยู่ในสภาวะ off จะบอกให้ DTE ทราบว่า DCE ยังไม่พร้อมที่จะรับข้อมูล ดังนั้น DTE จะยังไม่ส่งข้อมูลออกมา (ต้องป้อนสัญญาณ RTS ใหม่)

Clear to Send จะอยู่ในสภาวะ on ก็ต่อเมื่อสัญญาณ Request to Send และ Data Set Ready จะอยู่ในสภาวะ on ทั้งคู่ ถ้าไม่ใช้ขา Request to Send ให้ถือว่าสัญญาณ Request to Send เป็น on ตลอด

เวลา ดังนั้นสภาวะของ Clear to Send เป็นอย่างไรจะขึ้นอยู่กับสถานะของสัญญาณ Data Set Ready ว่าเป็น on หรือ off

ในการอินเตอร์เฟสตามมาตรฐาน RS-232-C ซึ่งทำการอินเตอร์เฟสระหว่าง DTE และ DCE ในกรณีที่มีเครือข่ายสวิชชิ่งโทรศัพท์เข้ามาเกี่ยวพันด้วยเราจะต้องใช้เซอร์กิดต่อไปนี้เข้ามาเกี่ยวพันด้วย

เซอร์กิต CC : Data Set Ready

เซอร์กิต CD : Data Terminal Ready

เซอร์กิต CE : Ring Indicator

เซอร์กิต CF : Receive Line Signal Detector

เนื่องจากการทำงานของเซอร์กิตต่อไปนี้เกี่ยวพันกับเครือข่ายโทรศัพท์ดังนั้นถ้าเราทำการติดต่อระยะสั้นๆโดยไม่ผ่านข่ายสายโทรศัพท์เราสามารถตัดเซอร์กิตต่อไปนี้ออกไปได้ การประยุกต์การใช้งานเซอร์กิตดังกล่าวเป็นการใช้งานร่วมกับอุปกรณ์ต่อโทรศัพท์อัตโนมัติ

หน้าที่สำคัญของ Data Set Ready และ Data Terminal Ready คือมันแสดงให้เราทราบว่าอุปกรณ์ของเราพร้อมที่จะทำการสื่อสารข้อมูลหรือไม่

CIRCUIT CC : Data Set Ready

สัญญาณนี้เป็นสัญญาณควบคุมที่ส่งมาจาก DCE ไปยัง DTE ในกรณีที่ Data Set Ready อยู่ในสภาวะ on แสดงว่า DCE ได้ต่อกับ Communication Channel เรียบร้อยแล้ว การที่ Data Set Ready เป็น on หมายความว่า DCE ของเรา(local)ได้ต่อกับDCE ของเครื่องคอมพิวเตอร์ที่เราต้องการติดต่อด้วย(Remote) ได้ตอบรับการเรียกทำให้เกิดการเชื่อมต่อกันของCommunication Channel ขึ้นระหว่าง DCE ทั้งสองด้าน เมื่อ Communication Channel ถูกเชื่อมต่อแล้วระบบก็จะเข้าสู่โหมดการส่งข้อมูล

CIRCUIT CD : Data Terminal Ready

สัญญาณควบคุมจะถูกส่งจาก DTE ไปยัง DCE สัญญาณ Data Terminal Ready ต้องอยู่ในสภาวะ on ก่อนที่ DCE จะ on สัญญาณ Data Set Ready หลังจากที่DCE เชื่อมต่อช่องสัญญาณสื่อสารเรียบร้อยแล้วถ้าสัญญาณ Data Terminal Ready เปลี่ยนจาก on เป็น off DCE จะตัดการเชื่อมต่อใน Communication Channel ทิ้งทันที

CIRCUIT CE : Ring Indicator

สัญญาณนี้เป็นสัญญาณนี้เป็นสัญญาณควบคุมที่ส่งจากDCE ไปยัง DTE เมื่อสัญญาณนี้มีสภาวะเป็น on แสดงว่า DCE ได้รับสัญญาณกริ่งเข้ามา เพราะฉะนั้นเราใช้สัญญาณควบคุมตัวนี้สำหรับโมเด็มที่มีความสามารถในการตอบรับการเรียกอัตโนมัติ

CIRCUIT CF : Received Line Signal Detector

สัญญาณนี้ส่งจาก DCE ไปยัง DTE เมื่อ DCE ได้รับสัญญาณ carrier (ซึ่งต้องเป็นไปตามข้อกำหนดของบริษัทผู้ผลิต DCE ด้วย)ที่ส่งมาจาก DCE อีกด้านหนึ่ง (remote side) สัญญาณ Received line Signal เป็น on แสดงว่า DCE จับสัญญาณใน Communication Channel ซึ่งจะนำไป Demodulate ได้แล้ว ในโมเด็มแบบต่างๆสายเส้นนี้จะถูกต่อกับ LED เพื่อแสดงให้รู้ว่าขณะนี้มีสัญญาณ Carrier เข้ามา

สัญญาณนี้อาจจะถูกเรียกว่า Data Carrier Detector DCD

เทอร์มินัลที่ใช้งานส่วนมากเราต้องให้ขานี้มีสภาวะเป็น HIGH ตลอดเวลาทั้งการรับและส่งข้อมูลในกรณีต่อสัญญาณขานี้เข้ากับขาสัญญาณตัวอื่นบนคอนเน็คเตอร์ เช่น Data Terminal Ready

2.3.4 PC Serial Communication

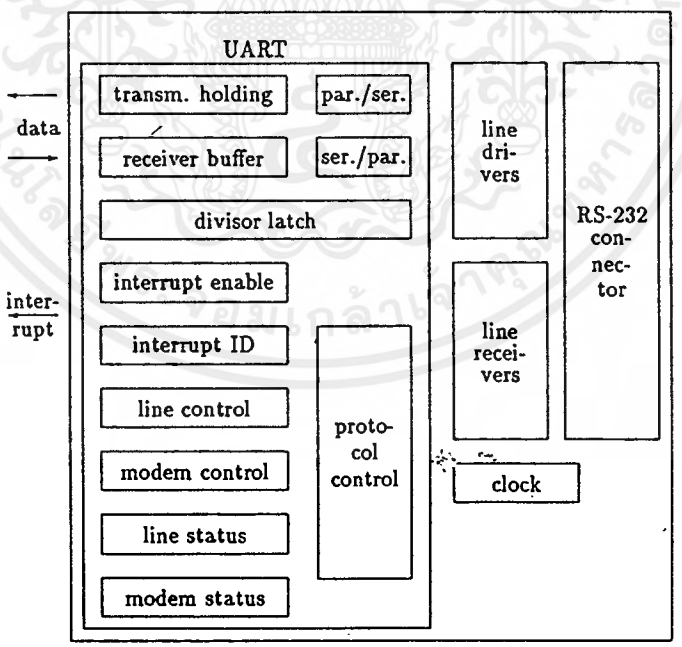
การอินเตอร์เฟสของเครื่องคอมพิวเตอร์กับการส่งสัญญาณแบบอนุกรมนั้น ข้อมูลจะต้องถูกเปลี่ยนจากขนานเป็นแบบอนุกรมด้วยอุปกรณ์ parallel in serial out register และ serial in parallel out register และส่วนที่ใช้ส่งสัญญาณ handshake สำหรับป้องกันไม่ให้อุปกรณ์ส่ง ส่งข้อมูลเร็วกว่าอุปกรณ์รับ

การจัดการข้อมูลนี้จะถูกจัดด้วยอุปกรณ์ส่งข้อมูลอนุกรมแบบโปรแกรมได้ เช่น อุปกรณ์ตระกูล 8250 ที่สามารถเรียกว่า a Universal Asynchronous Receiver Transmitter (UART) หรือไม่ก็อุปกรณ์ 8251 ที่สามารถจัดการได้ทั้ง synchronousและ asynchronous จะเรียกว่า a Universal Synchronous Asynchronous Receiver Transmitter (USART)

ลักษณะของซีเรียลพอร์ตจะเป็นคล้ายๆกับซีพียูที่ประกอบด้วยรีจีสเตอร์ที่มากมาย บางกลุ่มก็ใช้สำหรับใช้ส่งและรับข้อมูล บางกลุ่มจะใช้ในการเซ็ตพารามิเตอร์และ บางกลุ่มใช้ในการตรวจดูสถานะของการทำงาน อุปกรณ์ซีเรียลพอร์ตที่ใช้ใน PC จะสามารถพบได้บน Asynchronous Communication Adapter (ACA) หรือ PC/AT จะเป็นส่วนหนึ่งของ serial and parallel adapter ตำแหน่งของซีเรียลพอร์ตรีจีสเตอร์ จะพบได้ในช่วง I/O แอดเดรสที่กำหนด แอดเดรสไว้ในช่วงที่แน่นอน ส่วนสำหรับอุปกรณ์ PS/2 พอร์ตจะถูกพบได้ใน POS การควบคุมซีเรียลพอร์ตสามารถกระทำได้โดยตรงจากไอโอแอดเดรสสำหรับควบคุมจากรีจีสเตอร์แต่ละตัวหรือไม่ก็ควบคุมจาก hardware-interrupt ที่อ้างอิงพอร์ต COM1 และ COM2 จาก IRQ4 หรือ IRQ 3 สำหรับรีจีสเตอร์ในประเภท 8250 UART chip จะเหมือนกับใน 16450และ16550

ตารางที่ 2.4 Serial-port register address

REGISTER	ADDRESS COM1:	ADDRESS COM2:	DLA BIT
TRANSMITTER-HOLDING REGISTER	3F8H	2F8H	0
RECEIVER-BUFFER REGISTER	3F8H	2F8H	0
DIVISOR-LACTH REGISTER	3F8H	2F8H	1
DIVISOR-LACTH REGISTER	3F9H	2F9H	1
INTERRUPT-ENABLE REGISTER	3F9H	2F9H	0
INTERRUPT-IDENTIFICATION REGITER	3FAH	2FAH	
LINE-CONTROL REGISTER	3FBH	2FBH	
MODEM-CONTROL REGISTER	3FCH	2FCH	
LINE-STATUS REGISTER	3FDH	2FDH	
MODEM-STATUS REGISTER	3FEH	2FEH	

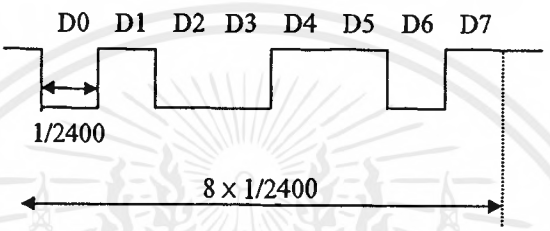


รูปที่ 2.8 The serial port

2.3.4.1 Bit Rate

สิ่งสำคัญมากสิ่งหนึ่งในการรับส่งข้อมูลแบบอนุกรมคือ ความถี่ที่ใช้ในการส่งข้อมูลซึ่งจะต้องสัมพันธ์กันระหว่างอุปกรณ์ที่ทำการรับและส่งข้อมูล ความถี่ที่ใช้นี้มีชื่อเรียกว่า อัตราความเร็ว (bit rate) ซึ่งมีความหมายถึง "อัตราการรับส่งข้อมูลเป็นจำนวนบิทใน 1 วินาที" ถ้าหากว่าเครื่องส่งใช้อัตราความเร็วที่ไม่สัมพันธ์กันกับเครื่องรับแล้ว ก็จะทำให้การรับส่งข้อมูลเกิดผิดพลาดขึ้นได้

โดยทั่วไปค่าของอัตราความเร็วนั้นจะใช้ค่าต่าง ๆ ดังต่อไปนี้คือ 110 , 150 , 300 , 1200 , 2400 , 4800 และ 9600 บิทต่อวินาที สำหรับในที่นี้จะสมมติว่าต้องการที่จะส่งข้อมูลแบบอนุกรมด้วยอัตรา 2400 บิทต่อวินาที และข้อมูลที่ต้องการจะส่งก็คือ 0B2H หรือ 10110010B ซึ่งเราสามารถที่จะแสดงได้ในรูปของสัญญาณดังรูป



รูปที่ 2.9 แสดงสัญญาณของข้อมูลที่ถูกส่งไปตามสายส่งสัญญาณ

จากรูปจะเห็นได้ว่าความกว้างของสัญญาณของแต่ละบิทจะมีค่าเท่ากับ 1/bit rate วินาที ซึ่งจากอัตราเร็วที่ต้องการใช้คือ 2400 บิทต่อวินาทีนั้นจะทำให้ความกว้างของแต่ละบิทที่จะส่งไปตามสายส่งนี้มีค่า 1/2400 วินาที ทำให้เราสามารถคำนวณเวลาที่จะต้องใช้ในการรับส่งข้อมูลแต่ละไบต์ (8 บิท) ได้เท่ากับ 8x416 ไมโครวินาที หรือ 3328 ไมโครวินาที อย่างไรก็ตามเพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้ จึงมีการเพิ่มบิทต่าง ๆ ลงไปในแต่ละไบต์ของข้อมูลเพื่อช่วยในการตรวจสอบความถูกต้องของข้อมูลที่เครื่องรับได้รับมา (แต่ไม่ได้หมายความว่าเมื่อเพิ่มบิทต่าง ๆ เหล่านี้เข้าไปแล้วจะทำให้การส่งผ่านข้อมูลมีความถูกต้อง 100%) สำหรับบิทต่าง ๆ ที่เพิ่มเข้ามานี้ก็คือ บิทเริ่มต้น (start bit), บิทหยุด (stop bit) และพาริตี้บิท (parity bit) ซึ่งจะทำให้ข้อมูลในแต่ละไบต์ที่ส่งออกไปนี้มีมากกว่า 8 บิท และเวลาที่ใช้ในการรับส่งข้อมูลก็จะมากขึ้นตามไปด้วย

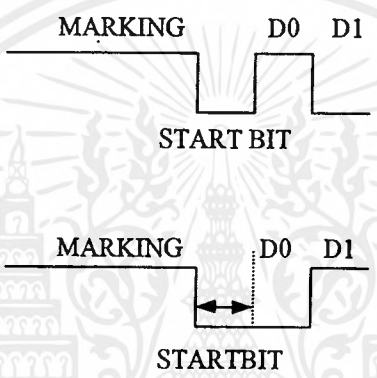
2.3.4.2 บิทเริ่มต้น (Start Bit)

ในการส่งผ่านข้อมูลแบบอนุกรม เราจำเป็นที่จะต้องทำให้อุปกรณ์ที่จะรับข้อมูลทราบว่าข้อมูลที่ส่งมานั้นเริ่มต้นที่จุดใด ดังนั้นเราจึงจำเป็นต้องเพิ่มข้อมูล 1 บิทลงไปก่อนหน้าข้อมูลจริง (actual data) ที่จะทำการส่ง (การส่งข้อมูลแบบอนุกรมจะส่งบิท D0 เป็นบิทแรก และบิท D7 เป็นบิทสุดท้าย) นั่นก็คือทำการเพิ่มบิทนี้ลงไปข้างหน้าบิท D0 นั่นเอง และเรียกบิทนี้ว่า "บิทเริ่มต้น"

หน้าที่ของบิทเริ่มต้นนี้นอกจากจะใช้ในการบอกว่าข้อมูลนั้นเริ่มต้นที่ใดแล้ว ยังทำงานร่วมกับบิทหยุด (stop bit) เพื่อช่วยในการแยกข้อมูลแต่ละชุดออกจากกันและความกว้างของบิทนี้จะเท่ากับความกว้างของบิทอื่น ๆ ในข้อมูลที่จะส่ง (D0-D7)

เมื่ออุปกรณ์ที่จะส่งข้อมูลยังไม่ได้ทำการส่งข้อมูลใด ๆ ออกมานั้น สายส่งจะอยู่ในสภาวะที่เรียกว่า "มาร์คกิง (marking) " ซึ่งเป็นสภาวะที่ไม่มีการรับส่งข้อมูลใด ๆ เกิดขึ้น ในที่นี้เราจะสมมติให้มาร์คกิงของสายส่งเป็นลอจิก "1" บิทเริ่มต้นที่จะเพิ่มเข้าไปนี้จะมีลอจิกที่ตรงข้ามกับลอจิกของมาร์คกิง ดังนั้นในกรณีนี้บิทเริ่มต้นจะมีลอจิกเป็น "0"

สำหรับบิทเริ่มต้นนี้จะมีความกว้างเท่ากับ 1 บิทของข้อมูล เช่น ใน 1 บิทของข้อมูลมีความยาวเท่ากับ 413 ไมโครวินาที บิทเริ่มต้นก็จะมีความกว้างของรูปสัญญาณเท่ากับ 413 ไมโครวินาทีด้วย ในรูปที่ 2.10 จะแสดงให้เห็นถึงบิทเริ่มต้นที่เพิ่มเข้าไปก่อนหน้าข้อมูล



รูปที่ 2.10 แสดงการเพิ่มบิทเริ่มต้นเข้าไปก่อนหน้าบิท D0 ในกรณีที่บิท D0 เป็น "1" และ "0" ตามลำดับ

2.3.4.3 พาริตี้บิท (Parity Bit)

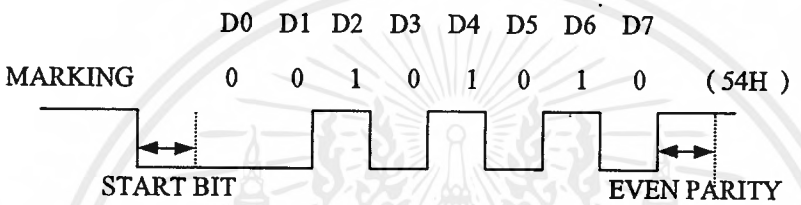
สำหรับบิทนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่า ข้อมูลที่ได้รับเข้ามานั้นมีความถูกต้องเหมือนกับข้อมูลที่ถูกส่งออกมาหรือไม่ (ถึงแม้ว่าการตรวจสอบบิทนี้จะไม่พบความผิดพลาดแต่ก็ไม่ได้หมายความว่าข้อมูลที่รับเข้ามานี้จะถูกต้อง 100%) โดยที่บิทนี้จะทำหน้าที่ในการบอกให้ส่วนรับข้อมูลทราบว่าข้อมูลที่ส่งออกมาในแต่ละไบต์นั้นมีจำนวนบิทที่เป็น "1" อยู่เป็นจำนวนคี่ หรือจำนวนคู่ เช่น ข้อมูล 54H หรือ 01010100B จะมีจำนวนบิทที่เป็น "1" อยู่เป็นจำนวนคี่ เป็นต้น สำหรับบิทที่ใช้ในการตรวจสอบนี้เรียกว่า "พาริตี้บิท"

พาริตี้บิทนี้จะถูกส่งออกมาโดยอุปกรณ์ส่งข้อมูล ซึ่งบิทจะเป็น "1" หรือ "0" นั้นขึ้นอยู่กับข้อมูลที่ส่งออกมา (D0-D7) ว่ามีจำนวนบิทที่เป็น "1" เป็นจำนวนคี่หรือจำนวนคู่ และยังขึ้นอยู่กับอุปกรณ์รับส่งข้อมูลด้วยว่าถูกออกแบบ (โปรแกรม) ไว้ให้รับส่งพาริตี้ในลักษณะของพาริตี้คู่หรือพาริตี้คี่ อีกด้วย

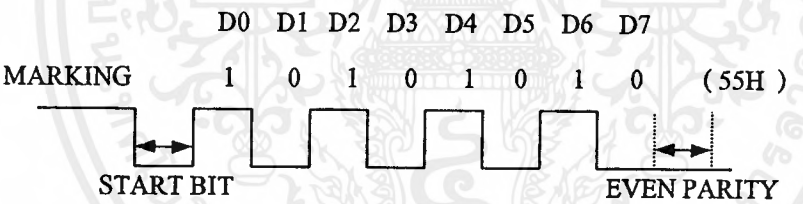
ในกรณีที่อุปกรณ์รับส่งข้อมูลออกแบบไว้ให้เป็นพาริตี้คู่ อุปกรณ์ส่งข้อมูลจะทำการส่งพาริตี้บิท เป็นลอจิก "1" ออกไปเมื่อจำนวนบิทที่เป็น "1" ของข้อมูล (D0-D7) เป็นจำนวนคี่ และจะทำการส่งพาริตี้บิทเป็นลอจิก "0" เมื่อจำนวนบิทที่เป็น "1" ของข้อมูลเป็นจำนวนคู่ (คือจะทำให้จำนวนบิทที่เป็น "1" ของข้อมูลรวมกับพาริตี้บิทแล้วเป็นจำนวนคู่นั่นเอง) สำหรับพาริตี้คี่ก็เช่นกัน คือพาริตี้บิทจะเป็น

"1" ในกรณีที่จำนวนบิทที่เป็น "1" ของข้อมูลเป็นจำนวนคู่ และจะเป็น "0" ในกรณีที่เป็นจำนวนคี่ ในที่นี้จะสมมติว่าอุปกรณ์ถูกออกแบบไว้สำหรับพาริตี้คู่และเราต้องการที่จะส่งข้อมูลออกไปให้กับส่วนรับข้อมูลเป็นจำนวนสองไบต์คือ 54H และ 55H

เมื่อเราส่งข้อมูล 54H ออกไป ซึ่งมีจำนวนบิทที่เป็น "1" เป็นจำนวนคี่ดังนั้นในกรณีนี้อุปกรณ์ส่งข้อมูลก็จะทำการส่งพาริตี้บิทเป็นลอจิก "1" ตามออกมาด้วย เพื่อจะให้บิทที่เป็น "1" ของข้อมูล 54H รวมกับพาริตี้บิทแล้วได้เป็นจำนวนคู่ ส่วนข้อมูล 55H นั้นจำนวนบิทที่เป็น "1" นั้นเป็นจำนวนคู่อยู่แล้วดังนั้นอุปกรณ์ส่งข้อมูลก็จะส่งพาริตี้บิทเป็น "0" ให้กับส่วนรับข้อมูล ดังในรูปที่ 2.11 และรูปที่ 2.12 สำหรับส่วนรับข้อมูลนั้น เมื่อทำการรับข้อมูลเข้ามาแล้วก็จะตรวจสอบสัญญาณว่าจำนวนบิทที่เป็น "1" ของข้อมูล รวมกับพาริตี้บิทนั้นเป็นจำนวนคู่หรือไม่ ถ้าหากว่าเป็นจำนวนคี่ก็แสดงว่าข้อมูลที่ได้รับเข้ามามีความผิดพลาดเกิดขึ้น (แต่ไม่ได้หมายความว่าถ้าเป็นจำนวนคู่แล้ว ข้อมูลที่รับเข้ามาจะถูกต้องเสมอไป)



รูปที่ 2.11 แสดงการเพิ่มพาริตี้ที่เป็น "1" ลงไปในข้อมูลแต่ละไบต์



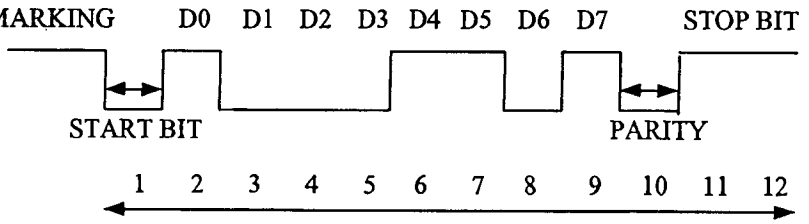
รูปที่ 2.12 แสดงการเพิ่มพาริตี้ที่เป็น "0" ลงไปในข้อมูลแต่ละไบต์

สิ่งสำคัญอีกสิ่งหนึ่งก็คือ ถ้าอุปกรณ์ส่งข้อมูลทำการส่งในลักษณะพาริตี้คู่หรือคี่ก็ตาม ส่วนรับข้อมูลก็จะต้องทำการรับในลักษณะเดียวกับอุปกรณ์ส่งข้อมูลด้วย เช่นในกรณีที่อุปกรณ์ส่งข้อมูลทำการส่งข้อมูลในลักษณะพาริตี้คู่ อุปกรณ์รับข้อมูลก็จะต้องทำการรับข้อมูลในลักษณะพาริตี้คู่ด้วย เป็นต้น

2.3.4.4 บิทหยุด (Stop Bit)

เป็นบิทสุดท้ายที่เพิ่มเข้าไป ใช้ในการตรวจสอบจุดสิ้นสุดของข้อมูลบิทนี้จะถูกเพิ่มเข้าไปหลังพาริตี้บิท ถ้าอุปกรณ์รับข้อมูลตรวจไม่พบบิทนี้ก็แสดงว่า ข้อมูลที่ได้รับเข้ามานั้นมีความผิดพลาดเกิดขึ้น สำหรับบิทหยุดนี้อาจมีจำนวนบิทเป็น 1 , 1.5 , 2 บิทก็ได้ รูปที่ 2.13 จะแสดงข้อมูลทั้ง 8 บิทที่ส่งออกมารวมทั้งบิทเริ่มต้น , บิทหยุด และพาริตี้บิทด้วย ซึ่งจะเห็นว่า สิ่งที่ส่งออกมาในแต่ละไบต์นั้นไม่ได้มีเพียง

ข้อมูล 8 บิทเท่านั้น แต่อาจจะมีได้ถึง 12 บิท (กรณีที่ส่งบิทหยุดออกมา 2 บิท) ดังนั้น ถ้าเราทำการส่งข้อมูลด้วยอัตรา 2400 bps เราจะต้องใช้เวลาในการส่งทั้งหมดเท่ากับ 12x416 ไมโครวินาที หรือ 4.99 มิลลิวินาที ไม่ใช่ 3328 ไมโครวินาทีดังที่คำนวณไว้ในตอนต้น



รูปที่ 2.13 แสดงการเพิ่มบิทต่าง ๆ ที่ใช้สำหรับป้องกันความผิดพลาดที่จะเกิดขึ้นในการรับส่งข้อมูลแบบอนุกรม

2.3.5 Serial Port Register

รีจีสเตอร์ต่างๆจะถูกอ่านและเขียนโดยซีพียูจากไอโอพอร์ตในช่วงแอดเดรส 3F8H ถึง 3FEH สำหรับพอร์ต COM1 และ 2F8H ถึง 2FEH สำหรับพอร์ต COM2 การเข้าถึงรีจีสเตอร์จะใช้บิท DLAB ใน line control register เพื่อควบคุมการเข้าถึงรีจีสเตอร์ 2 ตัวในแอดเดรสเดียวกัน

Transmitter-holding register (THR) จะเป็นรีจีสเตอร์ 8 บิทสำหรับการเขียนข้อมูลลงใน รีจีสเตอร์และส่งออกไปแบบซีเรียล (บิท LSB จะถูกส่งก่อน)

Receiver-buffer register (RBR) จะเป็นรีจีสเตอร์ขนาด 8 บิทใช้ในการรับข้อมูล รีจีสเตอร์ 2 ตัวนี้จะใช้แอดเดรสตำแหน่งเดียวกัน (THR จะเข้าถึงโดยการเขียน ส่วน RBR จะเข้าถึงโดยการอ่าน)

DLA (divisor latch) จะเป็นรีจีสเตอร์ขนาด 16 บิทจะใช้ในการหารสัญญาณนาฬิกา (1.8432 MHz) ซึ่งจะเก็บค่าของตัวหาร (divisor) สำหรับอัตราบอด (baud rate) ต่าง ๆ ดังตารางที่ 2.5

ตารางที่ 2.5 Divisor for standard baud rate

BAUD RATE	DIVISOR	CODE
110	1047	0
150	768	1
300	384	2
600	192	3
1200	96	4
2400	48	5
4800	24	6
9600	12	7
19200	6	8

Interrupt enable register (IER)ใช้ควบคุมการส่งสัญญาณอินเตอร์รัปท์ไปสู่ซีพียู

ตารางที่ 2.6 Interupt-enable register และ interupt-identification register

Bit	Interupt-enable register	Interupt-identification register
0	ENABLE RECEIVE DATA AVAILABLE INTERRUPT	=0,INTERRUPT PENDING
1	ENABLE THR REGISTER	INTERRUPT ID,BIT 0
2	ENABLE RECEIVER LINE STATUS INTERRUPT	INTERRUPT ID,BIT 1
3	ENABLE MODEM STATUS INTERRUPT	NOT USED(=0)
4,..,7	NOT USED(=0)	NOT USED(=0)

Interrupt identification register ใช้เก็บข้อมูลสำหรับคอยการอินเตอร์รัปท์ (ใช้ 3 บิทสุดท้าย) ลำดับของการอินเตอร์รัปท์ 3 (สูงสุด) , 0 (ต่ำสุด)

receiver line status interrupt (priority 3)จะเกิดขึ้นเมื่อเกิดโอเวอร์รันเออเรอร์ (RBR ยังเต็มเมื่อมีข้อมูลใหม่ส่งมา) หรือพาริตี้เออเรอร์ที่เกิดขึ้นจากข้อมูลที่รับเข้ามามีรูปแบบการใส่พาริตี้ที่ผิดหรือเฟรมมิ่งเออเรอร์ที่เกิดจากการจัดเพ็คเก็ตที่ผิด

received data available interrupt (priority 2) เมื่อข้อมูลในRBR พร้อมที่จะถูกอ่าน

THR empty interrupt (priority 1) เมื่อข้อมูลถูกส่งไปเรียบร้อย

The modem-status interrupt (priority 0) เมื่อมีการเปลี่ยนสถานะของโมเด็ม

Line-control register (LCR) เป็น 8 บิทรีจิสเตอร์ถูกใช้ในการโปรแกรมพารามิเตอร์สำหรับการสื่อสารข้อมูล

ตารางที่ 2.7 Line-control register และ modem-control register

Bit	Line-control register	Modem-control register
0	word-length select, bit 0	DATA TERMINAL READY
1	word-length select, bit 1	REQUEST TO SEND
2	number of stop bits	OUT1
3	parity enable	OUT2
4	even parity select	LOOP
5	"stick" parity	NOT USED(=0)
6	set break (=0, enable serial output)	NOT USED(=0)
7	Divisor-latch access bit (DLAB)	NOT USED(=0)

Modem-control register (MCR) สำหรับการควบคุมสัญญาณ Data terminal ready (DTR) และ Request to send และ Out1, Out2 ,Loop จะถูกใช้สำหรับการทดสอบพอร์ตอนุกรม(self test)

Line-status register จะถูกใช้สำหรับบอกข้อมูลแสดงสถานะสำหรับการส่งข้อมูล

Modem-status register จะประกอบด้วยข้อมูลแสดงสถานะจากสายภายนอก ข้อมูล 4 บิทบนจะคงที่ส่วนข้อมูล 4 บิทล่างจะมีการเปลี่ยนค่า

ตารางที่ 2.8 Line-status register และ modem-status register

BIT	LINE-STATUS REGISTER	MODEM-STATUS REGISTER
0	DATA READY	DELTA CLEAR TO SEND
1	OVERRUN ERROR	CELTA DATA SET READY
2	PARITY ERROR	TRAILING EDGE RING INDICATOR
3	FRAMING ERROR	DELTA DATA CARRIER DETECT
4	BREAK INTERRUPT	CLEAR TO SEND
5	THR EMPTY	DATA SET READY
6	TRANSMITTER SHIFT REGISTER EMPTY	RING INDICATOR
7	NOT USED (=0)	DATA CARRIER DETECTOR

2.3.6 Interrupt 14H

รอมไบออสจะเตรียมพร้อมไว้สำหรับการสื่อสารแบบอนุกรมในรูปของกลุ่มของฟังก์ชั่นบริการใน INT 14H ซึ่งสรุปไว้ในตารางที่ 2.10

ตารางที่ 2.9 INT 14H service review

SERVICE	FUNCTION
AH=0	INITAILIZE SERIAL PORT
AH=1	SEND CHARACTER
AH=2	RECEIVE CHARACTER
AH=3	READ STATUS
AH=4	"EXTENDED"INITAILZE(PS/2 ONLY)
AH=5	"EXTENDED"SERIAL PORT CONTROL(PS/2 ONLY)

ซึ่งจะแบ่งเป็นรูปแบบย่อยๆสำหรับการใช้รีจีสเตอร์ AH สำหรับบริการย่อย AH=04H และ AH=05H สำหรับเครื่องที่รับรองการใช้ PS /2

สำหรับโปรแกรมบริการ 14H จะรองรับได้ถึง 4 พอร์ตอนุกรม สำหรับการระบุพอร์ตจะใช้รีจีสเตอร์ DX (0,1,2,3)

ตารางที่ 2.10 SERVICE AH=00H - Initailize Serial Port

BIT	AL = MODEM STATUS	AH = LINE STATUS
0	DELTA CLEAR TO SEND	DATA READY
1	DELTA SET READY	OVERRUN ERROR
2	TRAILING-EDGE RING INDICATOR	PARITY ERROR
3	DELTA RECEIVED-LINE SIGNAL DETECT	FRAMING ERROR
4	CLEAR TO SEND	BREAK DETECTOR
5	DATA SET READY	TRANSMITTER-HOLDING REGISTER EMPTY
6	RING INDICATOR	TRANSMITTER-SHIFT REGISTER EMPTY
7	RECEIVED-LINE SIGNAL DETECT	TIME-OUT

บิทที่ 5 6 และ 7 ของ รีจีสเตอร์ AL ดังในตารางของ DLA

บิทที่ 3 และ 4 ใช้สำหรับการเซ็ตพาริตี้ โดย 00 และ 10 สำหรับไม่มีพาริตี้, 01 สำหรับพาริตี้คี่, 11 สำหรับพาริตี้คู่

บิทที่ 2 แสดงบิทหยุด (stopbit) โดย 1 สต๊อปบิทใช้ 0 และ 2 สต๊อปบิทใช้ 1

บิทที่ 1 และ 0 สำหรับความยาวข้อมูล โดย ความยาว 7 บิทใช้ 10 และ ความยาว 8 บิทใช้ 11

ตารางที่ 2.11 Service AH=01H / 02H -Send /Receive Character

INPUT	FUNCTION
AH = 01H AL = CHARACTER TO SEND DX = SERIAL PORT NO.	CHARACTER IS TRANSMITTED AH = LINE STATUS
AH = 02H DX = SERIAL PORT NO.	CHARACTER IS RECEIVED AL = RECEIVED CHARACTER AH = LINE STATUS

ตารางที่ 2.12 Service AH=03H - Read Status

INPUT	FUNCTION
AH = 03H DX = SERIAL PORT NO.	STATUS IS RETURNED AL = MODEM STATUS AH = LINESTATUS

ตารางที่ 2.13 Service AH = 04H -Extended Initialize (PS /2)

INPUT	FUNCTION
AH = 04H DX = SERIAL PORT NO. AL = 0 -NO BREAK, =1 - BREAK BH = PARITY CONTROL BL = STOP BITS(0 AND 1 VALID) CH="WORDLENGTH"(0 TO 3 VALID) CL=BAUD RATE (0 TO 8 VALID)	COMMUNICATION PARAMETERS ARE SET AL = MODEM SATTUS AH = LINE SATATUS

ตารางที่ 2.14 Service AH= 05H -Extended Port Control (PS/2)

INPUT	FUNCTION
AH= 05H, AL= 00H DX = SERIAL PORT NO.	MODEM-CONTROL REGISTER IS READ BL = MODEM-CONTROL REGISTER
AH= 05H, AL= 00H DX = SERIAL PORT NO. BL = MODEM-CONTROL REGISTER	MODEM-CONTROL REGISTER IS WRITTEN AND STATUS RETURNED AL = MODEM STATUS AH = LINE STATUS

2.3.7 การรับส่งข้อมูลแบบอนุกรมผ่าน 8051

สำหรับไมโครคอนโทรลเลอร์ 8051 สามารถรับและส่งข้อมูลโดยผ่านพอร์ตอนุกรมซึ่งมีโครงสร้างการทำงานในแบบที่เรียกว่า ฟูลดูเพล็กซ์ (Full Duplex) ในการรับและส่งข้อมูลแบบอนุกรมได้ในเวลาเดียวกันโดยทางด้านวงจรของตัวส่ง (Transmitter) จะส่งข้อมูลออกไปยังพอร์ตอนุกรมทางขาสัญญาณ TXD (พอร์ต 3.1) ส่วนวงจรด้านตัวรับ (Receiver) จะรับสัญญาณข้อมูลอนุกรมเข้ามาทางขาสัญญาณ RXD (พอร์ต 3.0)

พอร์ตอนุกรมของ 8051 สามารถโปรแกรมการทำงานได้ 4 โหมดโดยเลือกที่บิท SM0 และ SM1 ซึ่งอยู่ในรีจิสเตอร์ควบคุม การทำงานทั้ง 4 โหมดของพอร์ตอนุกรมมีดังนี้

โหมด 0 : ใช้รับส่งข้อมูล 8 บิท โดยการส่งจะเลื่อนทีละบิท โดยส่งบิท 0 ออกไปก่อนทางขา RXD และไม่มีการส่งบิทเริ่มต้น แต่จะส่ง shift clock ทางขา TXD ความเร็ว $\frac{1}{12}$ เท่าของ CPU clock

โหมด 1 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART(Universal Asynchronous Receiver/ Transmitter) โดยส่งแบบ 10 บิท ซึ่งประกอบด้วย ข้อมูล 8 บิท บิทเริ่มต้น 1 บิท บิทหยุด (stop bit) 1 บิท และสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้ โดยขึ้นกับบิท SMOD ในรีจิสเตอร์ PCON และอัตราโอเวอร์โฟลว์ของไทม์เมอร์ 1

โหมด 2 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิท และกำหนดอัตราความเร็วในการส่งข้อมูลเท่ากับ 1/32 และ 1/64 ของ CPU clock โดยควบคุมที่บิท SMOD ในรีจิสเตอร์ PCON

โหมด 3 : ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิทและสามารถเปลี่ยนแปลงอัตราความเร็วในการส่งข้อมูลได้โดยควบคุมที่บิท SMOD และอัตราโอเวอร์โฟลว์ของ Timer 1

ตารางที่ 2.15 แสดงการกำหนดค่าบิท SM0 , SM1 เพื่อเลือกโหมดการรับส่งข้อมูลอนุกรม

SM0	SM1	โหมด	การทำงาน
0	0	0	ทำงานเป็น shift register อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์
0	1	1	8 บิท UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้
1	0	2	9 บิท UART อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ ขึ้นกับบิท SMOD ในรีจิสเตอร์ PCON
1	1	3	9 บิท UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดเองได้

สำหรับในโครงงานนี้ได้เลือกใช้การรับและส่งข้อมูลในโหมด 1 โดยข้อมูลที่รับส่ง 1 ชุด มีขนาด 10 บิท คือ บิทเริ่มต้น 1 บิท (ลอจิก 0) , บิทข้อมูล 8 บิท และบิทหยุด (stop bit) 1 บิท (ลอจิก 1)

และอัตราความเร็วในการส่งข้อมูลมีค่าเป็น 9600 บิทต่อวินาที โดยอัตราความเร็วในการรับส่งข้อมูลคำนวณได้จากสมการ

$$Baudrate = \frac{2^{SMOD}}{32} \times \frac{OscillatorFrequency}{12 \times [256 - (TH1)]}$$

ตารางข้างล่างเป็นค่าอัตราความเร็วค่าต่าง ๆ ที่ใช้กันมากและบอกว่าสามารถใช้ได้จากไทม์เมอร์ 1 อย่างไร

ตารางที่ 2.16 ค่าที่ต้องนำไปไว้ในรีจิสเตอร์ของไทม์เมอร์ 1 เมื่อใช้อัตราความเร็วมาตรฐานต่าง ๆ

baud rate	ความถี่ของคริสตอล	บิต SMOD	ไทม์เมอร์ 1		
			C/T	โหมด	ค่าที่ใช้โหลด
Mode 0 Max: 1 MHz	12 MHz	X	X	X	X
Mode 2 Max: 375K	12MHz	1	X	X	X
Mode 1,3: 62.5K	12 MHz	1	0	2	FFH
19.2K	11.059 MHz	1	0	2	FDH
9.6K	11.059 MHz	0	0	2	FDH
4.8K	11.059 MHz	0	0	2	FAH
2.4K	11.059 MHz	0	0	2	F4H
1.2K	11.059 MHz	0	0	2	E8H
137.5K	11.986 MHz	0	0	2	1DH
110K	6 MHz	0	0	2	72H
110K	12 MHz	0	0	1	FEEBH

จากตารางจะเห็นว่าในการทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 0 จะมีความเร็วในการส่งข้อมูลมากที่สุดเมื่อเปรียบเทียบกับโหมดอื่นที่ความถี่คริสตอลค่าเดียวกัน และจะเห็นว่าหากเลือกใช้คริสตอลความถี่ 11.059 เมกกะเฮิรตซ์ จะสามารถจะสามารถตั้งค่าอัตราความเร็วในโหมด 1 และโหมด 3 เป็นค่ามาตรฐานที่ใช้กันทั่วไปได้ เช่น 1200 , 2400 , 4800 , 9600 , 19200 จึงเป็นเหตุผลสำคัญที่ในระบบควบคุมส่วนใหญ่เลือกใช้คริสตอลความถี่ 11.059 เมกกะเฮิรตซ์ มากกว่า 12 เมกกะเฮิรตซ์

จากตาราง นอกจากจะแสดงค่าอัตราความเร็วค่าต่าง ๆ เปรียบเทียบให้เห็นแล้ว ตารางนี้ยังแสดงค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ (SFR) TH1 ที่ค่าอัตราความเร็วมาตรฐานต่าง ๆ ให้ทราบอีกด้วยและผู้เขียนโปรแกรมสามารถนำค่านี้ไปใช้ได้เลย

โครงงานนี้ได้กำหนดใช้ไทม์เมอร์ 1 ทำงานในโหมด 2 และความถี่ของคริสตอลที่ใช้เท่ากับ 11.059 เมกกะเฮิรตซ์ บิท SMOD = 0 และรีจิสเตอร์ TH1 = FDH (= 253 D)

$$Baudrate = \frac{2^{0}}{32} \times \frac{11.059 \times 10^{6}}{12 \times [256 - 253]}$$

จะได้ Baud rate = 9600 บิตต่อวินาที

การทำงานในการรับส่งข้อมูลจะต้องใช้งานรีจิสเตอร์ใช้งานเฉพาะ (SFR) ต่าง ๆ ดังนี้

SCON (Serial Port Control Register) เป็นรีจิสเตอร์ที่ควบคุมการรับและส่งข้อมูลแบบพอร์ตอนุกรม ในที่นี้ได้กำหนดให้การรับส่งข้อมูลทำงานในโหมด 1โดยข้อมูล 1 ชุดมีขนาด10 บิท ซึ่งประกอบด้วยบิทเริ่มต้น 1 บิท (ลอจิก 0) , บิทข้อมูล 8 บิท และบิทหยุด 1 บิต (ลอจิก 1) และเป็นรีจิส-เตอร์เพื่อให้มีการรับข้อมูลเข้ามาทาง RXD

TMOD (Timer/counter Mode Register) เป็นรีจิสเตอร์ที่ทำหน้าที่ควบคุมการทำงานของตัวจับเวลาหรือตัวนับ ในการใช้งานนี้ได้กำหนดใช้งานไทม์เมอร์ 1 และเลือกการทำงานของไทม์เมอร์ 1 ในโหมด 2 โดยทำงานแบบ " รีโหลด (Reload) " และเป็นรีจิสเตอร์ที่กำหนดให้วงจรในการรับส่งข้อมูลทำงาน

TCON (Timer Control Register) เป็นรีจิสเตอร์เพื่อใช้ตรวจสอบสภาวะการทำงานของตัวจับเวลา (timer) ในที่นี้ได้ใช้รีจิสเตอร์นี้ในการตรวจสอบว่ามีข้อมูลเข้ามาทาง RXD หรือไม่และใช้ตรวจสอบในการส่งข้อมูล

IE (Interrupt Enable Register) เป็นรีสิสเตอร์ที่ใช้ควบคุมในการขัดจังหวะ (interrupt)

PCON (Power Control Register) เป็นรีจิสเตอร์ที่ควบคุมการใช้งานในรูปของการลดกำลังงาน และควบคุมอัตราเร็วในการส่งข้อมูล

รีจิสเตอร์ทั้งหมดที่กล่าวมานี้จะต้องทำงานที่สอดคล้องกันหมดจึงจะทำให้การรับส่งข้อมูลแบบ UART ทำงานในอัตราความเร็วที่กำหนด

กระบวนการรับและส่งข้อมูลอนุกรมของ 8051 (โหมด1)

- การส่งข้อมูล

1. เลือกโหมดในการส่งโดยใส่ค่าใน SCON ตามตารางที่ 2.15
2. เลือก baud rate ตามตารางที่ 2.16 โดยนำค่าที่ใช้โหลดใส่ใน TH1 , ทำบิท SMOD ใน PCON ให้เป็น 0 หรือ 1 ตามค่าที่กำหนดในตาราง และโปรแกรมไทม์เมอร์ 1 ให้ทำงานในโหมด 2 (ใช้ TMOD)
3. โปรแกรมให้ไทม์เมอร์ 1 ทำงาน
4. เริ่มส่งข้อมูลโดยนำข้อมูลที่จะส่งใส่ใน SBUF
5. ทุกครั้งที่ส่งข้อมูลครบ 1 ไบต์จะต้องเคลียร์ TI ด้วยคำสั่ง CLR TI

- การรับข้อมูล

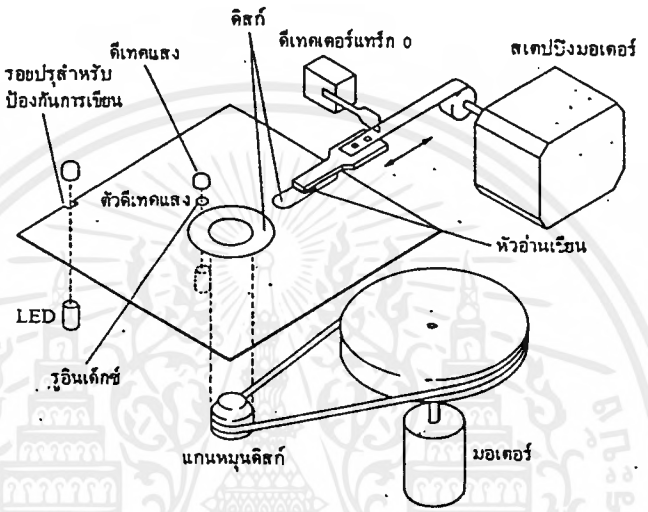
การรับข้อมูลมีลำดับขั้นตอนลักษณะเดียวกันกับการส่งข้อมูลแต่ต่างกันตรงที่ต้องเซทบิท REN ใน SCON และทุกครั้งที่รับข้อมูลครบ 1 ไบต์จะต้องเคลียร์ RI ด้วยคำสั่ง CLR RI

2.3.8 Floppy disk

2.3.8.1 โครงสร้างของดิสก์ไดรฟ์

มอเตอร์ขับดิสก์ เป็นมอเตอร์ที่ทำหน้าที่ในการหมุนดิสเกตต์ซึ่งสปินเดิลมอเตอร์จะต้องมีความเร็วที่เที่ยงตรงมาก

ส่วนประกอบที่สำคัญของดิสก์ไดรฟ์แสดงดังรูปที่ 2.14



รูปที่ 2.14 โครงสร้างพื้นฐานของดิสไดรฟ์

สเตปปิงมอเตอร์ เป็นมอเตอร์ที่ใช้ในการเลื่อนหัวอ่าน/เขียน ไปยังแทร็กที่ต้องการซึ่งวิธีการเลื่อนหัวอ่าน/เขียนที่นิยมใช้ได้แก่ การใช้แผ่นแถบโลหะที่บาง เคลื่อนที่ไปมาได้ ดังรูปที่ 2.14

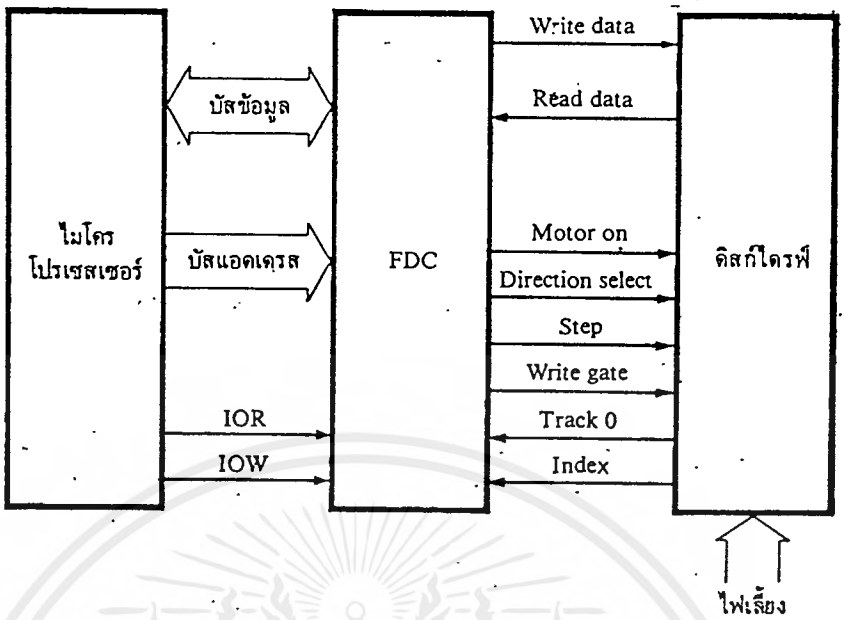
ตัวตรวจสอบแทร็ก 0 ทำหน้าที่ตรวจสอบว่าหัวอ่าน/เขียน อยู่ในตำแหน่งแทร็ก 0 หรือไม่

ตัวตรวจจับรูอินเด็กซ์ มีหน้าที่ตรวจสอบว่า ดิสเกตต์หมุนไปถึงตำแหน่งเริ่มต้นของ เซกเตอร์แรกหรือยัง

ตัวตรวจจับการป้องกันการเขียน ทำหน้าที่ตรวจสอบว่าดิสเกตต์มีการป้องกันการเขียนข้อมูลลงไปหรือไม่

หัวอ่าน/เขียน คือหัวอ่าน/เขียนดิสเกตต์

การติดต่อระหว่างดิสก์ไดรฟ์กับวงจรควบคุมดิสไดรฟ์มีสัญญาณที่ใช้ในการติดต่อดังรูปที่ 2.15



รูปที่ 2.15 สัญญาณในการติดต่อระหว่างดิสก์ไดรฟ์และวงจรควบคุมดิสก์ไดรฟ์

การอีนาเบิลมอเตอร์ เป็นสัญญาณควบคุมสปินเดิลมอเตอร์เพื่อให้ดิสเกตต์หมุน

การเลือกไดรฟ์ เป็นสัญญาณบอกว่ามีการติดต่อระหว่างวงจรควบคุมดิสก์ไดรฟ์กับดิสก์ไดรฟ์

การเขียนข้อมูล คือ ข้อมูลที่ส่งมาจากวงจรควบคุมดิสก์ไดรฟ์เพื่อเขียนลงบนดิสเกตต์เป็นสัญญาณในการติดต่อระหว่างดิสก์ไดรฟ์และวงจรควบคุมดิสก์ไดรฟ์ โดยข้อมูลที่ส่งมาจะเป็นซีเควน-เชียล

การอ่านข้อมูล คือข้อมูลส่งจากดิสก์ไดรฟ์ไปยังวงจรควบคุมดิสก์ไดรฟ์ เป็นข้อมูลที่อ่านจากดิสเกตต์ที่ต้องการ โดยเป็นข้อมูลซีเควนเชียล

อีนาเบิลการเขียน เป็นสัญญาณที่อนุญาตให้มีการเขียนข้อมูลลงบนดิสเกตต์

เลือกหัวอ่าน เป็นสัญญาณกำหนดหัวอ่าน/เขียนว่าต้องการอ่าน/เขียนข้อมูลจากหัวอ่านด้านใดในการอ่าน/เขียนดิสเกตต์ 2 ด้าน

ทิศทาง เป็นสัญญาณควบคุมการเลื่อนตำแหน่งของหัวอ่าน/เขียนว่าจะให้เลื่อนออกหรือเลื่อนเข้าจากตำแหน่งแทร็กเดิม เพื่อค้นหาแทร็กใหม่

สเตป เป็นสัญญาณกำหนดให้มีการเลื่อนหัวอ่าน/เขียนไป 1 แทร็ก ในทิศทางที่กำหนดโดยสัญญาณทิศทาง

แทร็ก 0 เป็นสัญญาณที่ส่งจากดิสก์ไดรฟ์ไปยังวงจรควบคุมดิสก์ไดรฟ์ เพื่อบอกว่าตำแหน่งหัวอ่าน/เขียนอยู่ในตำแหน่ง แทร็ก 0

อินเด็กซ์ เป็นสัญญาณที่ส่งจากดิสก์ไดรฟ์ไปยังวงจรควบคุมดิสก์ไดรฟ์ เมื่อรูอินเด็กซ์ถูกตรวจจับได้

ป้องกันการเขียน เป็นสัญญาณที่ส่งจากดิสก์ไดรฟ์ไปยังวงจรควบคุมดิสก์ไดรฟ์ เพื่อบอกว่าดิสเกตต์มีการป้องกันการเขียนข้อมูล

ข้อกำหนดรายละเอียดที่สำคัญของดิสก์ไดรฟ์

track to track time คือเวลาที่ใช้ในการเลื่อนหัวอ่าน/เขียนไป 1 แทร็ก

seek time คือเวลาที่ใช้ในการเลื่อนหัวอ่าน/เขียนจากแทร็กใดแทร็กหนึ่งไปยังแทร็กที่ต้องการ โดยค่าเวลานี้มีค่าเท่ากับจำนวนแทร็กที่เลื่อนคูณกับ track to track time

setting time คือช่วงเวลาที่ต้องรอให้หัวอ่าน/เขียนหยุดนิ่งหลังจากการที่ seek เพื่อทำการอ่าน/เขียนได้

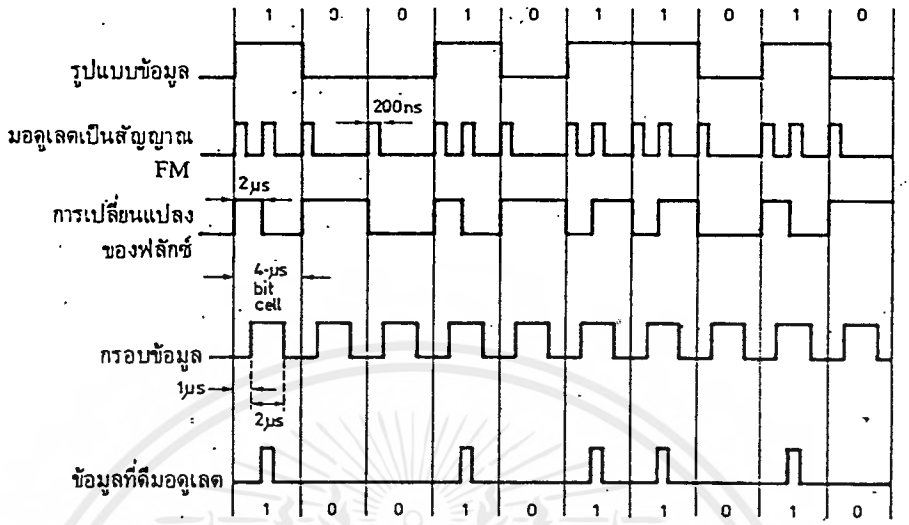
head load time คือเวลาที่ใช้ในการให้ดิสเกตต์หมุนได้ความเร็วคงที่หลังจากมอเตอร์เริ่มหมุน

2.3.8.2 การมอดูเลชันของข้อมูล

การเก็บข้อมูลมีความจำเป็นที่จะต้องทำการมอดูเลชัน เพื่อลดความผิดพลาดในการอ่าน/เขียนข้อมูล เนื่องจากความเร็วของดิสเกตต์ไม่คงที่ และเพื่อที่จะเพิ่มความจุในการเก็บข้อมูลของ ดิสเกตต์ ซึ่งหลักการของการมอดูเลชันที่ใช้ในฟลอปปี้ดิสก์ ทำโดยการผสมสัญญาณนาฬิกากับข้อมูล

วิธีการมอดูเลชันที่ใช้ในฟลอปปี้ดิสก์ มีหลายวิธีดังนี้

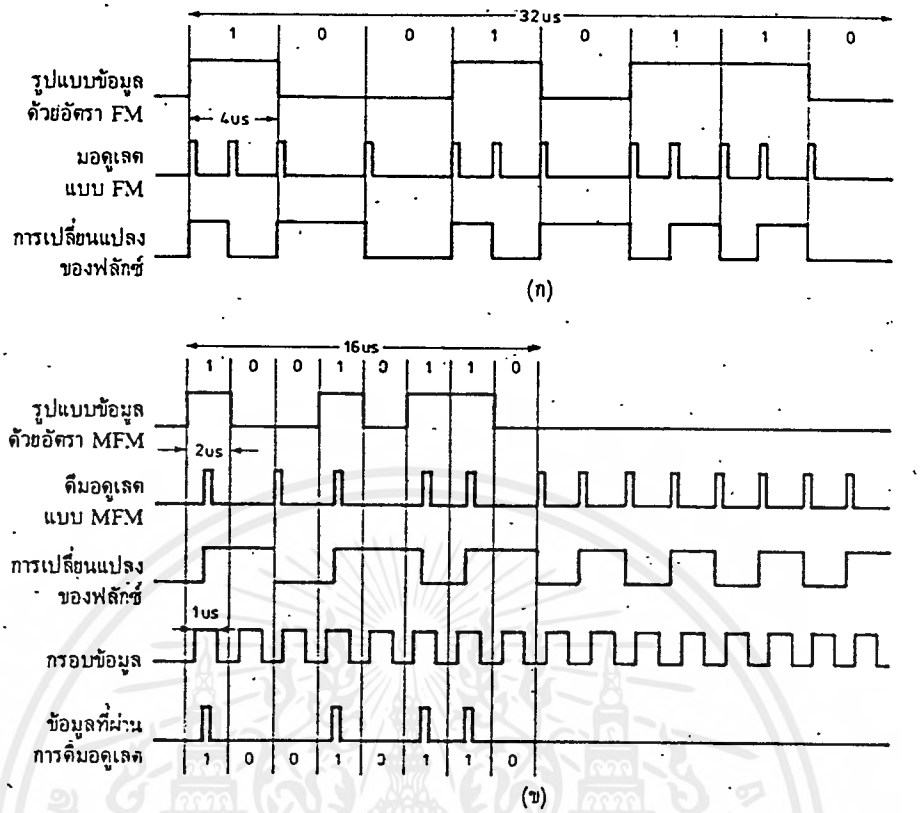
FM (frequency modulation) วิธีการนี้ใช้การบันทึกข้อมูลลงดิสเกตต์แบบความหนาแน่นเท่าเดียว (single density) หลักการมอดูเลชันทำโดยการใส่ข้อมูลระหว่างสัญญาณนาฬิกา ดังนั้นในการดีมอดูเลชันก็กระทำโดยอ่านข้อมูลในช่วงสัญญาณนาฬิกา ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 การมอดูเลชันโดยวิธี FM

เราจะเห็นว่าการมอดูเลชันแบบ FM จะมีอัตราการบันทึกข้อมูลต่ำ เพราะอัตราการบันทึกข้อมูลขึ้นกับจำนวนครั้งของการเปลี่ยนแปลงของฟลักซ์ต่อวินาที และในขณะที่ทำการส่งลอจิก "1" ข้อมูลที่ถูกมอดูเลชัน จะมีอัตราการเปลี่ยนแปลงของฟลักซ์สูงกว่าขณะทำการส่งลอจิก "0" ดังนั้นสัญญาณนาฬิกาหรืออัตราการส่งข้อมูลจะต้องมีค่าเพียงครึ่งหนึ่งของอัตราการบันทึกข้อมูลของดิสเกตต์

MFM (modified frequency modulation) วิธีการนี้ใช้ในการบันทึกข้อมูลลงบน ดิสเกตต์แบบความหนาแน่น 2 เท่า (double density) ซึ่งพัฒนาหลักการจากวิธี FM วิธี MFM จะพยายามกำจัดสัญญาณนาฬิกาที่ไม่จำเป็นออกไป โดยมีหลักการของการมอดูเลชันแสดงในรูปที่ 2.17



รูปที่ 2.17 เปรียบเทียบการมอดูเลชันแบบ FM (ก) กับการมอดูเลชันแบบ MFM (ข)

1. เมื่อบันทึกข้อมูลลอจิก "1" ทำการสร้างพัลส์ในช่วงตรงกลางของบิทเซลเหมือนวิธี FM แต่ตัดสัญญาณนาฬิกาออก

2. เมื่อบันทึกลอจิก "0" ทำการส่งสัญญาณนาฬิกา ยกเว้นเมื่อข้อมูลก่อนหน้าเป็นลอจิก "1"

วิธีการมอดูเลชันแบบ MFM จะได้อัตราการส่งข้อมูลมีค่าเท่ากับอัตราการบันทึกข้อมูลของฟลอปปี้ดิสก์

M^2FM (modified modified frequency modulation) วิธีนี้ดัดแปลงจากวิธี MFM เล็กน้อย คือ ในกรณี M^2FM จะส่งสัญญาณนาฬิกาเมื่อข้อมูลที่ส่งเป็นลอจิก "0" ติดต่อกัน 2 บิทขึ้นไป

2.3.8.3 ส่วนประกอบของแผ่นดิสก์

แผ่นดิสก์จะถูกแบ่งออกเป็น แทร็ก (track) แผ่นดิสก์ที่เก็บข้อมูลแบบ 2 ด้าน (double side density) ปกติจะมี 40 แทร็ก แทร็กทั้ง 40 แทร็กของฟลอปปี้ดิสก์ (แทร็กที่ 0 จะอยู่รอบนอก ส่วนแทร็กที่ 39 จะอยู่ใกล้กับบริเวณที่เป็นจุดศูนย์กลาง) จะถูกแบ่งย่อยออกเป็นเซกเตอร์ (sector) เซกเตอร์หนึ่งจะจุข้อมูลได้ 512 ไบต์

เซกเตอร์บนแผ่นดิสก์ ฟลอปปี้ดิสก์แบบ 2 ด้านจะมีลักษณะดังต่อไปนี้ ส่วนบูตเรคอร์ดจะอยู่บนลอจิคัลเซกเตอร์ที่ 0 ซึ่งเป็นเซกเตอร์ลำดับแรกบนแผ่นดิสก์ เซกเตอร์ 1 กับ 2 และเซกเตอร์ 3 กับ 4 แต่ละคู่จะเก็บ FAT ซึ่งมีขนาด 2 เซกเตอร์โดยจะมี 2 ชุดเพื่อไว้ตรวจสอบซึ่งกันและกัน ส่วนไดเรกทอรีจะใช้เนื้อที่ 7 เซกเตอร์โดยจะใช้ลอจิคัลเซกเตอร์หมายเลข 5 ถึงหมายเลข 11

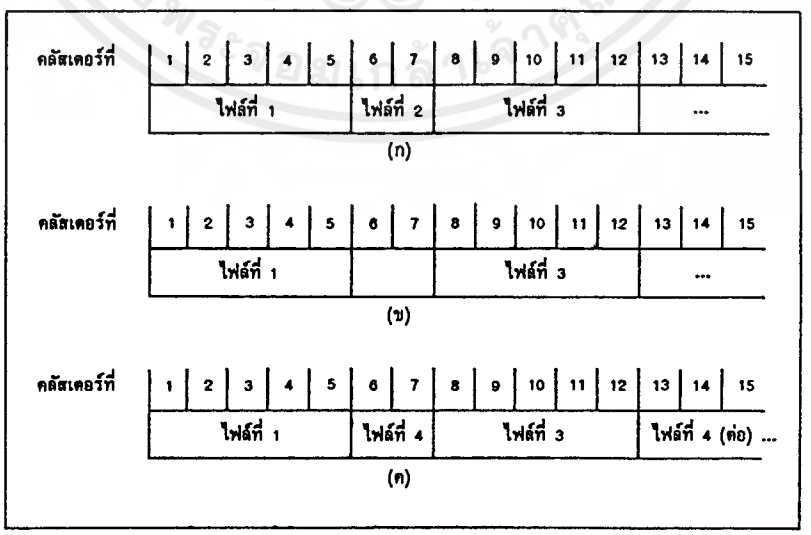
บูตเรคอร์ด เซกเตอร์แรกบนแผ่นดิสก์จะเป็นที่เก็บบูตเรคอร์ด ถ้าบนแผ่นดิสก์ไม่มีไฟล์ของระบบ (system file) แผ่นดิสก์นั้นจะไม่สามารถบูตได้ถึงแม้บนแผ่นดิสก์นั้นจะมีส่วนของบูตเรคอร์ดอยู่

เมื่อผู้ใช้พยายามบูตเครื่อง PC โปรแกรมในรอมไบออส (ROM BIOS) จะพยายามอ่านข้อมูลจากบูตเรคอร์ดในไดรฟ์เอ (ถ้าไม่มีแผ่นดิสก์ใส่อยู่ที่ไดรฟ์เอเครื่องจะตรวจสอบว่ามีฮาร์ดดิสก์อยู่หรือไม่) ถ้ามีแผ่นดิสก์อยู่ที่ไดรฟ์เอ ส่วนของบูตเรคอร์ดในแผ่นดิสก์จะถูกอ่านเข้ามา ต่อจากนั้น การควบคุมต่าง ๆ จะถูกส่งให้บูตเรคอร์ดเป็นผู้ทำงาน

File Allocation Table (FAT) และคลัสเตอร์ ส่วนที่อยู่ถัดจากบูตเรคอร์ดจะเป็น FAT เซกเตอร์บนแผ่นดิสก์จะถูกรวมเข้าเป็นคลัสเตอร์ซึ่งแต่ละคลัสเตอร์จะเป็นอิสระต่อกัน จำนวนเซกเตอร์บนแผ่นดิสก์จะสามารถเปลี่ยนแปลงได้ตามรูปแบบของแผ่นดิสก์ เช่น แผ่นดิสก์แบบ 2 ด้าน (double-side) จะมี 2 เซกเตอร์ใน 1 คลัสเตอร์ แผ่นดิสก์แบบที่มีความจุสูง (high-density) จะมี 1 เซกเตอร์ต่อ 1 คลัสเตอร์

แต่ละคลัสเตอร์จะมีรายการอยู่ หรือหมายความว่ามีตารางอยู่ที่ตำแหน่งใดตำแหน่งหนึ่งบนแผ่นดิสก์ โดยตารางนี้จะเก็บข้อมูลของแต่ละคลัสเตอร์ ซึ่งก็คือ FAT

ไฟล์สามารถแบ่งออกเป็นส่วนย่อยเพื่อสามารถเก็บไว้ในแผ่นดิสก์ได้ ส่วนย่อยที่สุดของไฟล์ก็คือคลัสเตอร์ ลักษณะของคลัสเตอร์ของไฟล์ในแผ่นดิสก์แสดงดังรูปที่ 2.18 (ก) และจากรูปที่ 2.18 ถ้าไฟล์หมายเลข 2 ถูกลบ จะเป็นดังรูปที่ 2.18 (ข) และหลังจากนั้นถ้ามีการเขียนไฟล์หมายเลข 4 ลงบนแผ่นดิสก์ การใช้เนื้อที่ว่างของแผ่นดิสก์ก็ควรจะใช้เนื้อที่ว่างทั้งหมดที่ปรากฏ ถึงแม้ว่าไฟล์หมายเลข 4 จะยาวแต่ก็ควรถูกแยกออกเป็น 2 ส่วนดังรูปที่ 2.18 (ค)



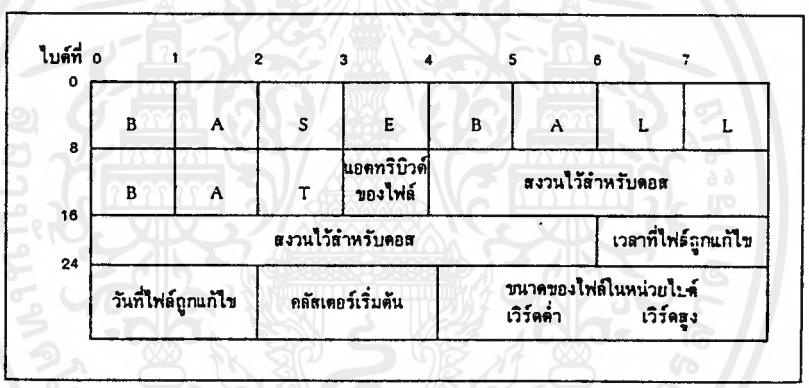
รูปที่ 2.18 ข้อมูลของไฟล์ใน FAT

สำหรับวิธีนี้แต่ละคลัสเตอร์ของไฟล์ที่ข้อมูลถูกนำไปเก็บไว้ อาจอยู่กระจัดกระจายบนแผ่นดิสก์ ซึ่งขึ้นอยู่กับว่าบนแผ่นดิสก์มีที่ว่างที่ตำแหน่งใด ซึ่ง FAT จะเป็นตัวจัดการกับข้อมูลที่แยกกันนี้

ไดเรกทอรี ถัดจาก FAT ทั้ง 2 ชุดในแผ่นดิสก์เข้ามาก็จะเป็นส่วนของไดเรกทอรี สำหรับแผ่นดิสก์แบบ 2 ด้าน 9 เซกเตอร์ ไดเรกทอรีจะอยู่ที่ลอจิคัลเซกเตอร์ที่ 5 ถึง เซกเตอร์ที่ 11

จากรูปที่ 2.19. จะสังเกตเห็นชุดของข้อมูลในไดเรกทอรีของไฟล์ชื่อ BASEBALL.BAT แต่ละชุดของข้อมูลในไดเรกทอรีจะมีขนาด 32 ไบต์ 8 ไบต์แรกจะเก็บชื่อไฟล์ ซึ่งในกรณีนี้ก็คือ BASEBALL 3ไบต์ถัดไปคือไบต์ที่ 8 ถึงไบต์ที่ 10 เป็นที่เก็บส่วนขยายของไฟล์ก็คือ BAT ข้อมูลอีก 11 ไบต์จะเก็บ แอตทริบิวต์ของไฟล์ ส่วนไบต์ที่ 12 ถึงไบต์ที่ 21 จะสงวนไว้สำหรับดอสและเพื่อการขยายในอนาคต

เวลาที่ไฟล์ถูกสร้างและหรือเวลาสุดท้ายที่ไฟล์ถูกเขียนจะถูกเก็บไว้ที่ไบต์ 22 ถึงไบต์ที่ 25 ส่วนไบต์ที่ 26 และไบต์ที่ 27 จะเก็บหมายเลขของคลัสเตอร์เริ่มต้นของไฟล์บนแผ่นดิสก์และอีก 4 ไบต์สุดท้ายคือไบต์ที่ 28 ถึงไบต์ที่ 31 จะเก็บขนาดของไฟล์



รูปที่ 2.19 ข้อมูลของไฟล์ BASEBALL.BAT ในชุดข้อมูลของไดเรกทอรี

ส่วนที่เก็บข้อมูลในแผ่นดิสก์ ถัดจากส่วนของไดเรกทอรี จะเป็นส่วนที่เก็บข้อมูลจริง ซึ่งทุกๆ เซกเตอร์ที่มีข้อมูลจะมีหมายเลขอยู่ใน FAT ส่วนชื่อไฟล์และความยาวของไฟล์จะถูกเก็บไว้ในไดเรกทอรี อย่างไรก็ตามในที่นี้จะกล่าวถึงสิ่งที่เกี่ยวข้องกับการทำงานของไฟล์และการทำงานของแผ่นดิสก์โดยการใช้ภาษาแอสเซมบลีจัดการกับเซกเตอร์และคลัสเตอร์บนแผ่นดิสก์ ซึ่งไบออสและดอสจะมีการจัดการที่แตกต่างกันในการกำหนดเซกเตอร์ โดยฟังก์ชันของดอสจะจัดการกับเซกเตอร์แรกที่ใช้งานได้บนแผ่นดิสก์ ส่วนฟังก์ชันของไบออสจะจัดการเหมือนกับเป็นเซกเตอร์แรกทางกายภาพบนแผ่นดิสก์จริงๆ

ตารางของดิสเกตต์ (diskette table) โดยทั่วไปเมื่อกล่าวถึงเรื่องโครงสร้างของแผ่นดิสก์ ก็ต้องกล่าวถึงเรื่องตารางของดิสเกตต์ด้วย ตารางของดิสเกตต์หรือ diskette base table จะเป็นตารางที่มี

ความยาว 11 ไบต์ ตารางนี้จะเก็บค่าของพารามิเตอร์ที่ใช้ในการปฏิบัติงานของดิสก์ไดรฟ์ ตารางนี้จะถูกเก็บไว้ใน ROM ที่ตำแหน่งของหน่วยความจำตอนบน ซึ่งไม่สามารถจะเปลี่ยนแปลงได้

ค่า 2 ไบต์แรกของตารางจะเป็นค่าที่เกี่ยวกับ step rate time และ head load/unload time ซึ่งเป็นค่าคงที่ที่หัวอ่านใช้เมื่อติดต่อกับแทร็กหนึ่ง ๆ

ค่าไบต์ที่ 3 เป็นค่าเกี่ยวกับเวลาที่ใช้ของฮาร์ดแวร์ ซึ่งสามารถเปลี่ยนแปลงได้โดยไม่เกิดข้อผิดพลาดใด ๆ ในการทำงาน ค่านี้เป็นค่าของเวลาที่มอเตอร์หมุนแผ่นดิสเกตต์จะหยุดหมุนหลังจากการทำงานใด ๆ

ไบต์ที่ 4 จะบอกถึงจำนวนไบต์ต่อเซกเตอร์

ไบต์ที่ 5 เป็นไบต์ที่บอกให้คอสรู้ว่ามีกี่เซกเตอร์ในแต่ละแทร็ก

3 ไบต์ถัดไปจะเกี่ยวข้องกับรูปแบบของเซกเตอร์บนแผ่นดิสก์รวมทั้งบางเรื่อง เช่น ความยาวของ gap และขนาดของข้อมูล

ไบต์ที่ 9 เป็นไบต์ที่เก็บค่าซึ่งจะเอาไว้ใส่ในเซกเตอร์บนแผ่นดิสก์ที่ถูกฟอร์แมตใหม่บนเครื่อง PC

ไบต์ที่ 10 เป็นค่าของเวลาที่หัวอ่านใช้เพื่อเลื่อนหัวอ่านให้อยู่เหนือแทร็กที่ต้องการ

ไบต์สุดท้ายคือ ไบต์ที่ 11 เป็นค่าของเวลาที่มอเตอร์หมุนแผ่นดิสก์เริ่ม warm up และเริ่มต้นทำงาน

บทที่ 3
การคำนวณและการสร้าง

โครงงานนี้ใช้ ไมโครคอนโทรลเลอร์ ทำการควบคุมคือไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ของ INTEL ซึ่งในความสะดวกในการสร้างและพัฒนาโครงงานจึงได้ใช้ MCS-51 เป็นแบบที่มีหน่วยความจำโปรแกรมรอมภายใน (INTERNAL PROGRAM ROM MEMORY) เป็นหน่วยความจำชนิดแฟล็ชเมมโมรี่(FLASH MEMORY) ซึ่งมีคุณสมบัติในการ เขียน-ลบ ได้ถึง 1000 ครั้ง ของบริษัท INTEL เบอร์ 89C52 ไมโครคอนโทรลเลอร์เบอร์ 89C52 นี้มีคุณสมบัติ ที่เหมาะกับการใช้งานในโครงงานนี้ คือเป็นไอซีชนิดซีมอส(CMOS)และมีหน่วยความจำโปรแกรมรอมภายในขนาด 8 กิโลไบท์ (4K BYTE) สำหรับการพัฒนาโครงงานนี้ต่อไปที่มีความซับซ้อนมากขึ้น ก็สามารถเปลี่ยนไปใช้ไอซีในกลุ่มเบอร์เดียวกันที่มีขนาดหน่วยความจำโปรแกรมรอมภายในที่ใหญ่ขึ้นโดยที่มีโครงสร้างภายใน และการวางขาที่ตรงกัน โดยเบอร์ 89C55 มีหน่วยความจำโปรแกรมรอมภายใน ขนาด 20K BYTEและทำให้เรา สามารถถอดเปลี่ยนไอซีเพียงอย่างเดียวโดยที่ไม่ต้องทำการเปลี่ยนวงจรหรือเปลี่ยนการเดินสายภายในโครงงานเลยเพื่อทำการเพิ่มขนาดหน่วยความจำโปรแกรม

ส่วนคุณสมบัติอื่นๆของชิปก็เหมือน MCS-51ทั่วไป และส่วนสำคัญ คือ สามารถอ้างหน่วยความจำโปรแกรมรอมภายนอก(ถ้าใช้เพิ่ม)ได้สูงสุด 64K BYTE และสามารถอ้างหน่วยความจำเก็บข้อมูลภายนอก(EXTERNAL RAM MEMORY)ได้สูงสุด 64K BYTE มีอินพุท-เอ้าท์พุทพอร์ทขนาด 32BIT มีพอร์ทอนุกรมแบบ FULLDUPLEX ASYNCHRONOUS. จำนวน 1พอร์ท มีเคาท์เตอร์ไทมเมอร์ขนาด16BIT จำนวน 2 ตัว สามารถทำ IDEL MODE และ POWER DOWN MODE ซึ่งเป็นการประหยัดพลังงานได้

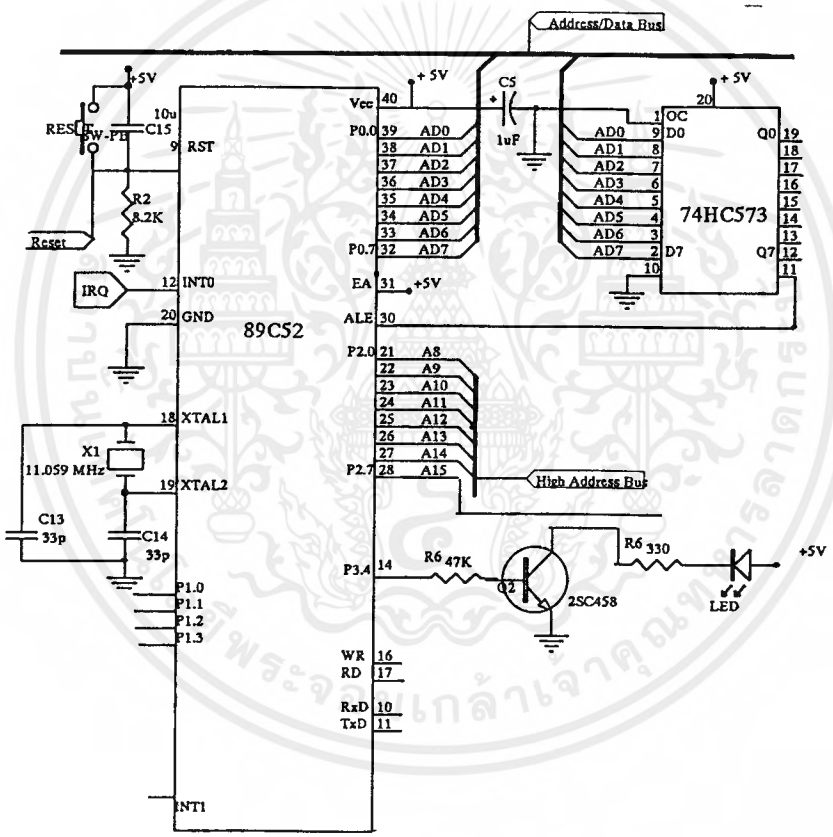
ในส่วนของ A/D converter สามารถรับสัญญาณอินพุทอนาล็อกได้ 8 ช่อง ขนาดแรงดันอินพุท 0-5 โวลท์ แปลงเป็นสัญญาณดิจิตอลขนาด 8 บิท และยังมีส่วนของการจัดเก็บข้อมูลบนจานแม่เหล็ก (diskette) คือฟลอปปี้ดิสก์ไดรฟ์ (floppy disk drive) ขนาด 3 ½ นิ้ว (1.44 MB) กับ ฟลอปปี้ดิสก์คอนโทรลเลอร์ (floppy disk controller) ซึ่งเหมือนกับที่ใช้ในเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป ซึ่งในฟลอปปี้ดิสก์ไดรฟ์จะมีวงจรควบคุมการถ่ายโอนข้อมูลและควบคุมระบบกลไกทางกลของฟลอปปี้ดิสก์ เราเพียงแต่สั่งงาน (โดยคำสั่ง) จากฟลอปปี้ดิสก์คอนโทรลเลอร์ให้เพียงเขียน/อ่านตามแทร็กที่ต้องการเท่านั้น และเพื่อการใช้งานที่สะดวกโครงงานนี้จึงสามารถสั่งงาน , รับค่าพารามิเตอร์ต่าง ๆ ได้โดยตรงจึงได้เพิ่มส่วนจอแสดงผลซึ่งเป็น LCD module แบบตัวอักขระ ขนาด 16 ตัว 1 แถว และส่วนคีย์บอร์ดขนาด 4 คีย์

ลักษณะของโครงงานนี้คือสามารถนำไปใช้เก็บข้อมูล(LOG)ภายนอกโดยสามารถเคลื่อนย้ายได้สะดวก จึงไม่ต้องมีแหล่งจ่ายไฟหรือสายต่อภายนอก(ขณะ LOG DATA) ทำให้ต้องใช้แบตเตอรี่เป็นแหล่งจ่ายพลังงาน ซึ่งต้องออกแบบวงจรให้ประหยัดพลังงานที่สุด โดยเลือกใช้ ไมโครคอนโทรเลอร์ชนิดซีมอสที่กินไฟน้อย และไอซีที่ใช้ร่วมในวงจรเป็นไอซีชนิดซีมอสที่ทำงานในระดับแรงดันของ ไอซีชนิดทีทีแอล คือไอซีแบบ HIGH SPEED CMOS (74HC SERIES)

3.1 ส่วน CENTRAL CONTROL UNIT

3.1.1 วงจรหน่วยประมวลผล MICROCONTROLLER และ LATCH

ไมโครคอนโทรลเลอร์ที่ใช้คือ 89C51 และในโครงงานจะมีหน่วยความจำเก็บข้อมูลภายนอก (EXTERNALRAMMEMORY) ขนาด32KBYTE จึงต้องมีวงจร LATCH ADDRESS เพราะการติดต่อหน่วยความจำภายนอก 89C51 จะMULTIPLEX แอดเดรสไบท์ต่ำกับ บัสข้อมูลที่พอร์ท 0 เราจึงใช้ไอซี LATCH มา DEMULTIPLEX ให้ DATA/ADD BUS เป็น DATA BUS กับ LOW ADD BUS เราเลือกใช้ไอซี เบอร์ 74HC573 ที่มีโครงสร้างวงจรภายใน OCTAL TRANSPARENT LATCH เหมือนเบอร์ 373 แต่การจัดขาของ 537 จะสะดวกในการเชื่อมต่อมากกว่า เพราะขาทางด้านอินพุทอยู่ทางด้านเดียวกัน และเอาท์พุทอยู่อีกทางด้านหนึ่ง ขาที่ควบคุมการ LATCH ใช้สัญญาณ ALE จาก 89C51 ได้เป็นพอร์ต ADDRESS BUS 16 bit , DATA BUS 8 bit



รูป 3.1 วงจรไมโครคอนโทรลเลอร์และแลทช์ (LATCH)

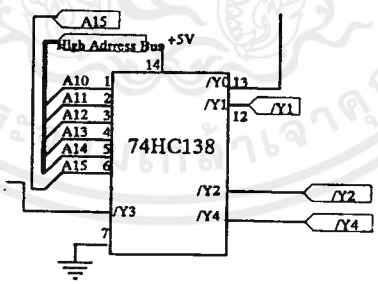
3.1.2 ส่วน DECODER

จาก ADDRESS BUS 16 bit เราใช้อ้างในการต่ออุปกรณ์ภายนอก คือ RAM , A/D 1 channel, RTC , A/D 8 channel , ฟลอปปี้ดิสก์คอนโทรลเลอร์ , LCD DISPLAY เราใช้ไอซีทีทีแอล เบอร์ 74HC138 ซึ่งเป็น 3 LINES TO 8 LINES DECODER/DEMULTIPLEX ซึ่งให้เอาท์พุทที่ DECODE

ได้ และใช้ INPUT 3 เส้น กับ GATE CONTROL 3 เส้น โดยเราจะแบ่งให้บัฟเฟอร์แรมอยู่ในส่วนแรก (32 KB แรก) จึงเหลือพื้นที่อีก 32 KB หลัง ในการแบ่งส่วนให้ DECODER เราจึงใช้ขาแอดเดรส A15 (MSB-ADDRESS) ซึ่งจะเป็น 0 เมื่ออ้างหน่วยความจำใน ADDRESS ครึ่งแรก จะเป็น 1 เมื่ออ้าง ADDRESS ครึ่งหลัง เราจึงต้องต่อ A15 กับขา GATE CONTROL ของ 138 (พร้อมทั้งต่อ A15 กับขา CS ของแรม) เราเหลือขา GATE CONTROL อีก 2 เส้น นำไปต่อกับ A14 , A13 ตามลำดับ ทำให้ DECODER ทำงานเมื่อ A15 , A14 , A13 เป็น"100" เท่านั้น (ADDRESS 8000H-A000H ช่วง 8 KB) และต่อขา IP DECODER คือ A,B,C เท่ากับ A12,A11,A10 ได้ 8 ส่วน O/P DECODER ส่วนละ 1 KB ตาม MEMORY MAP ดังนี้

MEMORY MAP

0000H - 7FFFH	=	32K	RAM
8000H - 83FFH	=	4K	RTC
8400H - 87FFH	=	4K	ADC1
8800H - 8BFFH	=	4K	ADC2
8C00H - 8FFFH	=	4K	LCD
9000H - 93FFH	=	4K	FDC
9400H - 97FFH	=	4K	NO USED (AUX1)
9800H - 9BFFH	=	4K	NO USED (AUX2)
9C00H - 9FFFH	=	4K	NO USED (AUX3)
A000H - FFFFH	=	24K	NO DECODE (NO USED)

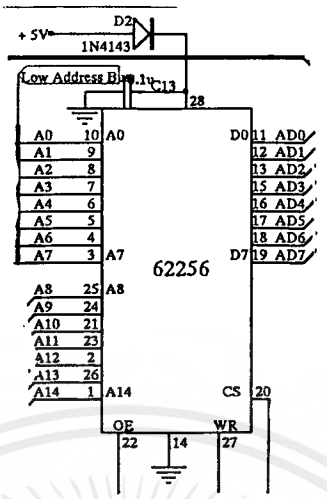


รูปที่ 3.2 วงจรส่วน DECODER

3.2 ส่วน DATA MEMORY UNIT BUFFER

ส่วนนี้ประกอบด้วยเมมโมรี (RAM) ขนาด 32K BYTE ใช้เป็นบัฟเฟอร์ของระบบและเป็นพื้นที่ทำงานของโปรแกรมภายใน ซึ่งเราจะอ่าน/เขียนข้อมูลกับแผ่นดิสก์โดยนำข้อมูลมาพักไว้ที่แรมบัฟเฟอร์นี้ก่อน เช่นการเขียนข้อมูลที่บันทึกได้ หรือการอ่าน/เขียนในส่วนของระบบจัดเก็บไฟล์ของแผ่นดิสก์

เพราะการติดต่อกับแผ่นดิสก์จะสามารถอ่าน/เขียนข้อมูลได้อย่างน้อยสุดทีละ 512 ไบต์ เท่ากับขนาดของ 1 เซกเตอร์ของแผ่นดิสก์ขนาด 3 ½ นิ้ว

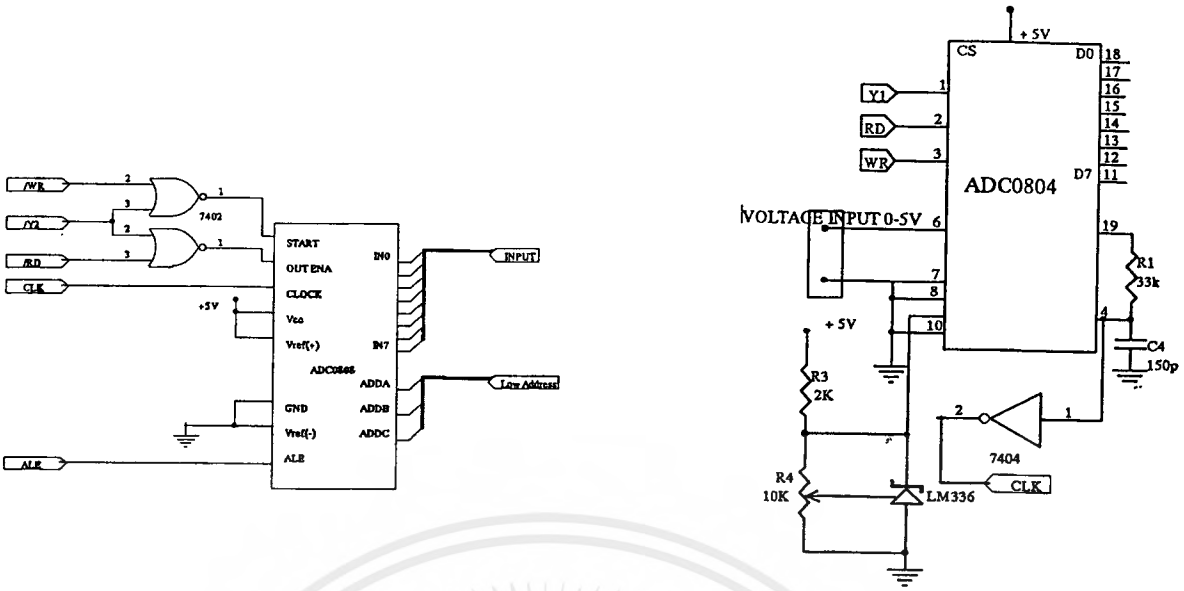


รูปที่ 3.3 วงจรส่วน DATA MEMORY UNIT

3.3 ส่วน A/D CONVERTER UNIT

เราใช้ A/D ทั่วไปคือ เบอร์ ADC 0804 ที่ให้ผลการแปลงเป็น 256 ระดับ (8 bit) ในที่นี้เราต่อ VIN+ เป็นสัญญาณเข้าและต่อ VIN- กับ GROUND ช่วงแรงดันที่แปลงได้คือ 0-5 โวลต์ จากแรงดัน VREF /2 = 2.500 โวลท์ ส่วนแรงดัน VREF มาจาก VOLTTAGE REFERENCE DIODE ซึ่งเป็นซีเนอร์ไดโอดที่มีสัมประสิทธ์ทางอุณหภูมิน้อยมาก คือ แรงดันจะเปลี่ยนแปลงไปน้อยเมื่อเทียบกับอุณหภูมิ A/D มีสัญญาณนาฬิกาภายในที่ใช้ในการควบคุมขั้นตอนการแปลงของวงจรแปลงภายใน A/D เชื่อมต่อ A/D กับ DATA BUS(8 bit)

A/D ตัวที่สองเป็น เบอร์ ADC 0808 ต่ออินพุทได้ 8 ช่อง แต่ละช่องมีแรงดัน 0-5 โวลท์มีกราวนด์ร่วมกับระบบ , V+ref ต่อกับแหล่งจ่ายไฟ 5 โวลท์ , V- ref ต่อกับกราวนด์ และสัญญาณนาฬิกาต่อกับสัญญาณนาฬิกา (clock) ของ ADC 0804



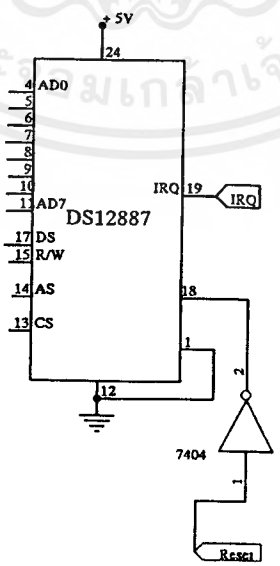
รูปที่ 3.4 วงจรส่วน A/D CONVERTER UNIT

3.4 ส่วน REAL TIME CLOCK

REAL TIME CLOCK (RTC) ใช้เบอร์ DS12887A ซึ่งมี LITIUM BATTERRY, CRYSTAL OSCILLATOR อยู่ภายในCHIP MODULE โดยมีอายุการใช้งานของแบตเตอรี่ภายใน เป็นเวลา 10ปี ความเที่ยงตรงของนาฬิกาภายในRTCคือ คลาดเคลื่อนไม่เกิน1นาทีต่อเดือนในขณะที่ไม่มีแหล่งจ่ายไฟให้เลย

เราสามารถตั้งค่าให้ RTC ทำการINTERRUPTได้ทุกๆคาบเวลา (PERIODIC) หรือเมื่อถึงเวลาที่ตั้งไว้ (ALARM) และRTCยังมีหน่วยความจำแรมภายในจำนวน114 BYTE ที่สามารถใช้งานได้ทั่วไป ค่าในแรมจะเหมือนค่าใน RTCที่จะยังคงค่าอยู่ได้แม้ว่าไฟจะดับแล้วก็ตาม

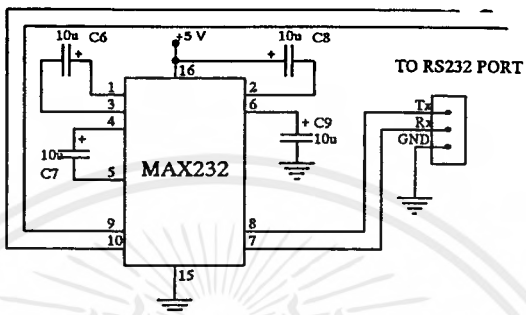
เราต่อขาอินเตอร์รัพท์เข้า 89C51ที่ขา INT0



รูปที่ 3.5 วงจรส่วน REAL TIME CLOCK

3.5 ส่วน RS-232/TTL CONVERTER UNIT

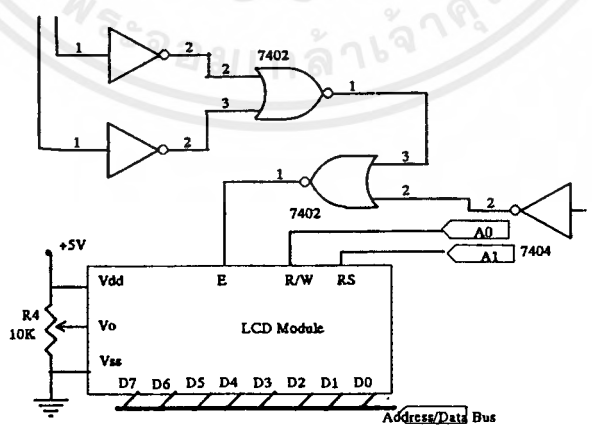
เราใช้พอร์ทอนุกรม ที่ขา TXD และRXD เป็นระดับสัญญาณของทีทีแอลอยู่ในช่วง 0 ถึง +5V เราใช้ไอซีเป็นตัวขับ(เปลี่ยน)ระดับไปเป็นระดับแรงดันตามมาตรฐานของ RS-232 คือ +12 ถึง -12V การส่งสัญญาณ ใช้สายสัญญาณเพียง 3 เส้นคือ TXDATA ,RXDATA ,GROUND



รูปที่ 3.6 วงจรส่วน RS-232/TTL CONVERTER UNIT

3.6 ส่วนจอแสดงผล (LCD MODULE DISPLAY)

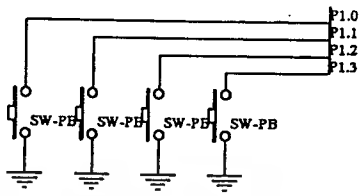
ส่วนนี้สามารถแสดงผลตัวอักษรภาษาอังกฤษและตัวเลขได้ 16 ตัวอักขระ 1 แถว ซึ่งใช้งานได้ง่ายเพราะระบบไม่ต้องสแกนจอแสดงผลเพียงแต่ส่งคำสั่งควบคุมและค่ารหัสแอสกี (ASCII) ของตัวอักษรเรียงกันมาก็สามารถแสดงผลได้ และสามารถปรับแต่งความเข้มของจอได้จากตัวต้านทานปรับค่าได้ 10 กิโลโอห์ม



รูปที่ 3.7 วงจรส่วนจอแสดงผล

3.7 ส่วนคีย์บอร์ด (KEYBOARD SWITCH)

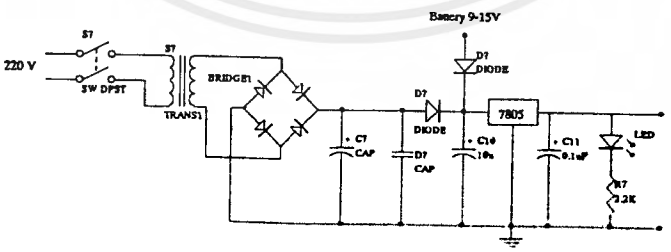
สำหรับคีย์บอร์ดใช้ 4 คีย์สวิทช์ต่อกับพอร์ท 1 ของไมโครคอนโทรลเลอร์ โดยในพอร์ท 1 ของไมโครคอนโทรลเลอร์จะมีโครงสร้างที่มีการ pull up ภายในแล้วเราเพียงแต่ต่อสวิทช์กับกราวนด์เท่านั้นก็ใช้งานได้โดยเมื่อมีการกดสวิทช์จะทำระดับแรงดันที่ขาพอร์ท 1 ที่เราเซทไว้เป็นอินพุทมีระดับ 0



รูปที่ 3.8 วงจรส่วนคีย์บอร์ด

3.8 ส่วนวงจรจ่ายแรงดัน (+5 V POWER SUPPLY UNIT)

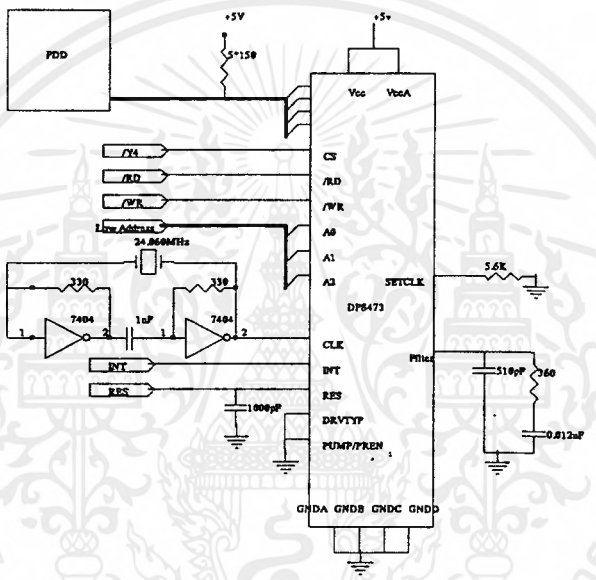
โครงงานนี้สามารถใช้แหล่งจ่ายไฟจากไฟ 220 โวลท์ตามบ้านทั่วไปหรือใช้แบตเตอรี่ 12 โวลท์ในกรณีที่ใช้งานภายนอกสถานที่ซึ่งไม่สะดวกในการต่อสายไฟ 220 โวลท์ โดยจะมีส่วนจ่ายแรงดันประกอบด้วย หม้อแปลงใช้ในการลดแรงดัน วงจรเรกติไฟร์และวงจรกรองกระแส ซึ่งที่จุดนี้สามารถต่อพ่วงแบตเตอรีเข้ามาได้สำหรับเป็นแหล่งจ่ายไฟสำรองในกรณีที่ไม่ใช้ไฟ 220 โวลท์ หลังจากชุดนี้จะเป็นส่วนแปลงไฟ (REGLATOR) ใช้ไอซี 7805 เป็น REGULATOR แปลงแรงดันไฟฟ้าจาก +9V ถึง +15V เป็น +5V



รูปที่ 3.9 วงจรส่วน +5V POWER SUPPLY UNIT

3.9 ส่วนวงจร FLOPPY DISK CONTROLLER

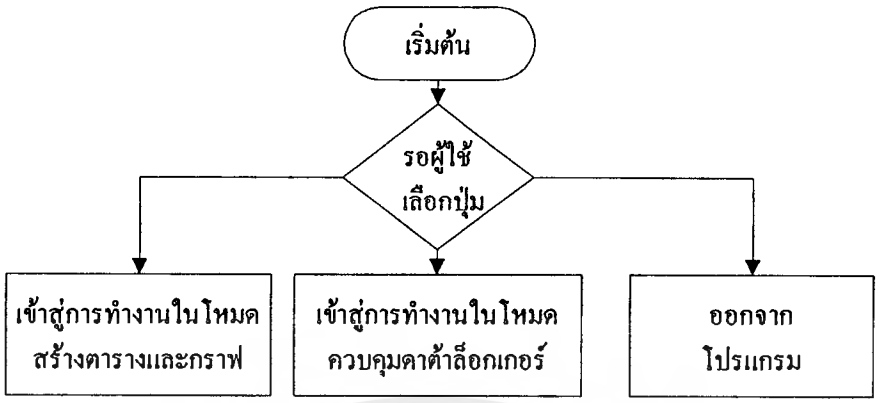
เราใช้ไอซีเบอร์ DP 8473 ที่สามารถใช้งานได้ง่าย ต่อพ่วงกับอุปกรณ์ภายนอกน้อยและเป็นไอซีที่พัฒนามาจากตระกูลของ INTEL ทำให้สามารถเชื่อมต่อกับไมโครคอนโทรลเลอร์เบอร์ MCS51 ของ INTEL ด้วยกันได้ง่าย และมีคอนเนคเตอร์ที่สามารถต่อกับฟลอปปี้ดิสก์ไดรฟ์ได้โดยตรงถึง 2 ตัวโดยไม่ต้องใช้บัฟเฟอร์เพิ่มเติมทำให้วงจรไม่ใหญ่มาก เราใช้ดิสก์ไดรฟ์ขนาด 3 ½ นิ้ว 1.44 MB และต่อขาคอนเนคเตอร์เผื่อดิสก์ไดรฟ์ไว้ 2 ตัว (ไดรฟ์ A และ B) แต่เราใช้เพียง 1 ตัวจึงต้องใช้สายแพแบบไขว้สายมาต่อกับไดรฟ์เป็นไดรฟ์ A



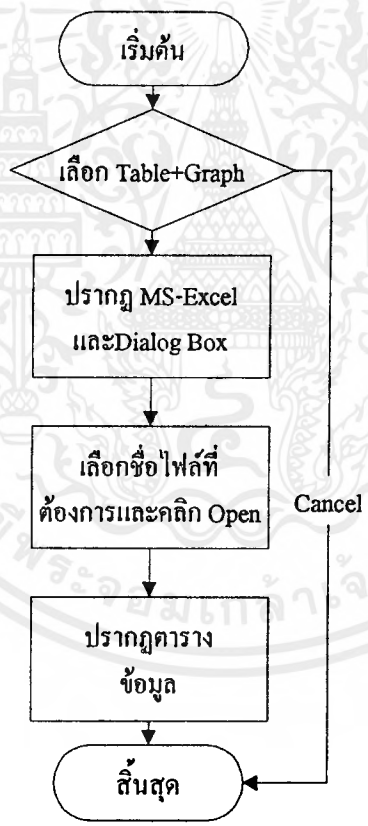
รูปที่ 3.10 วงจร FLOPPY DISK CONTRILLER

3.10 ส่วนของโปรแกรม(SOFT WARE)

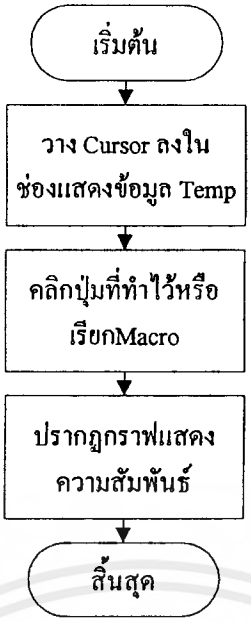
3.10.1 โปรแกรมที่ใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับอุปกรณ์รวบรวมข้อมูลเคลื่อนที่ ใช้โปรแกรมภาษาเคลไฟล์โดยมีแผนผังการทำงานดังนี้



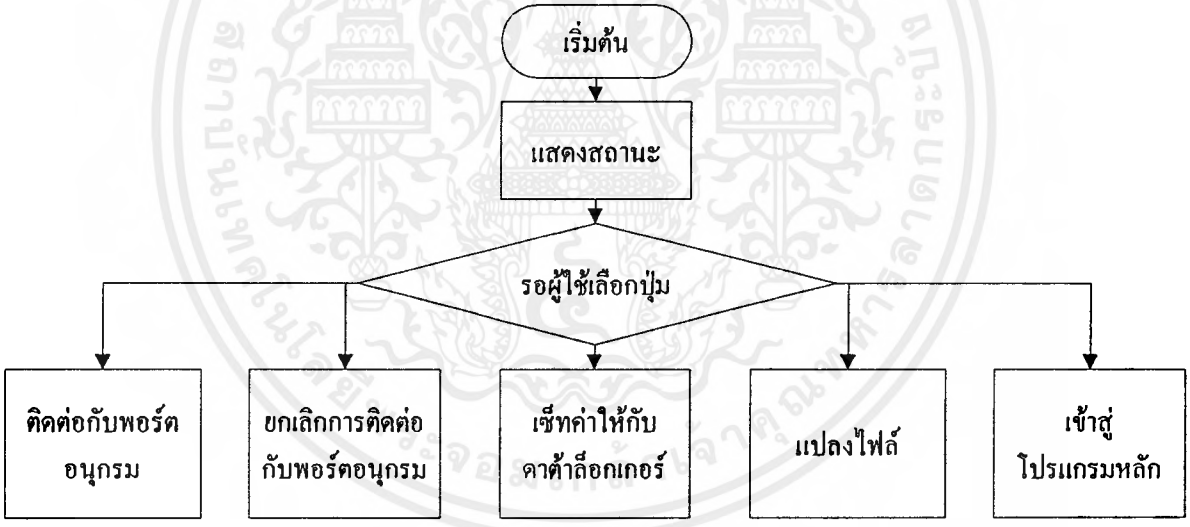
รูปที่ 3.11 แผนผังการทำงานของโปรแกรมโดยรวม



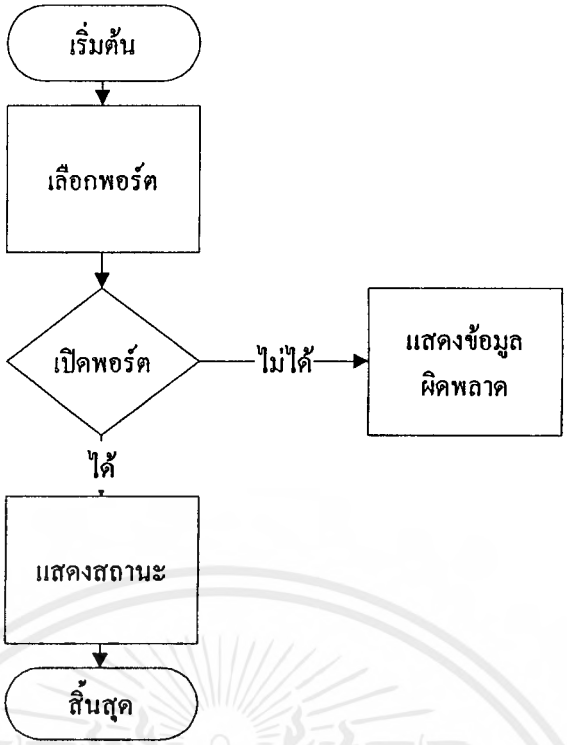
รูปที่ 3.12 แผนผังการทำงานในส่วนโปรแกรมการสร้างตาราง



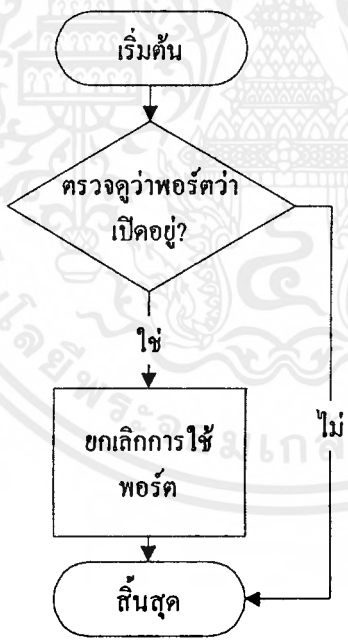
รูปที่ 3. 13แผนผังการทำงานในส่วนโปรแกรมการสร้างกราฟ



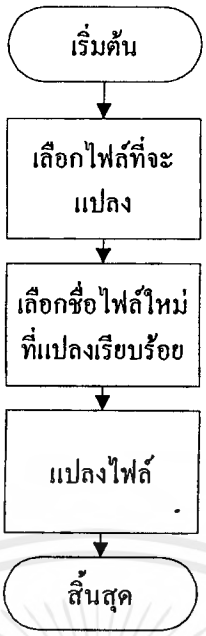
รูปที่ 3.14แผนผังการทำงานในส่วนโปรแกรมการควบคุมอุปกรณ์เก็บข้อมูล



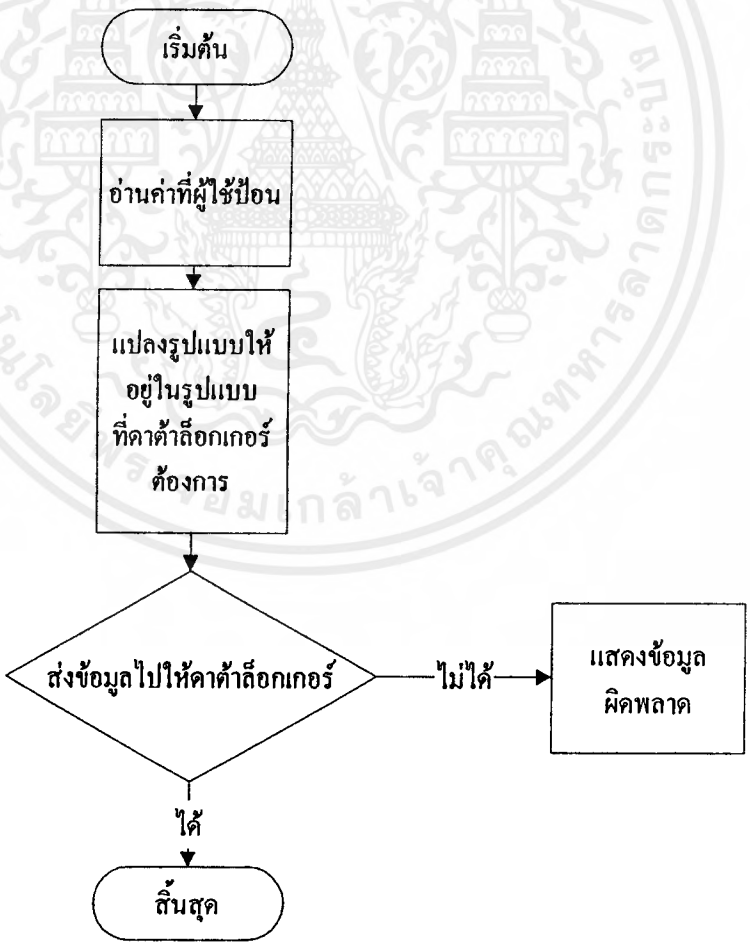
รูปที่ 3.15 แผนผังการทำงานในส่วนโปรแกรมการติดต่อกับพอร์ตอนุกรม



รูปที่ 3. 16แผนผังการทำงานในส่วนของการยกเลิกการติดต่อกับพอร์ต

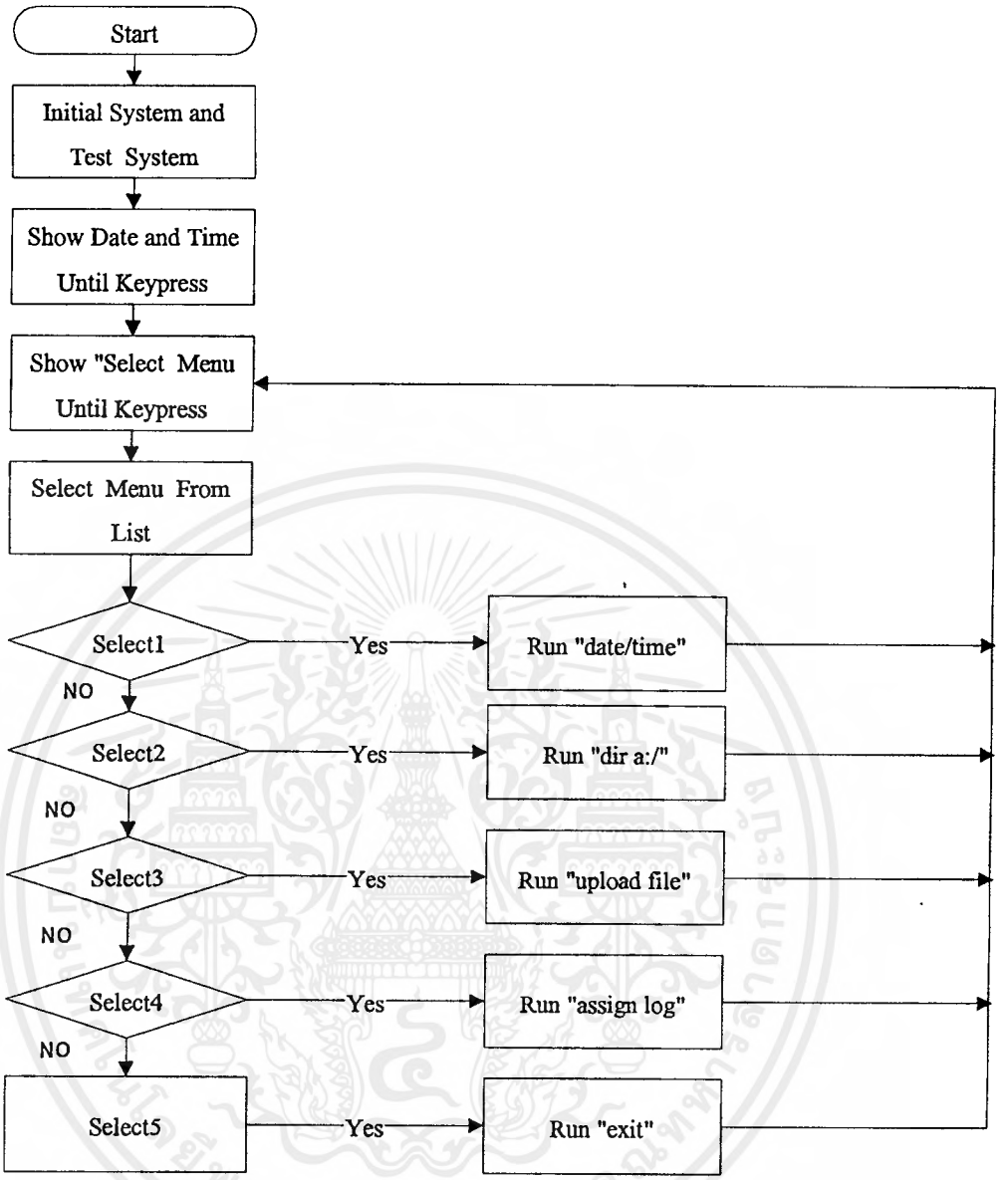


รูปที่ 3.17 แผนผังการทำงานในส่วนของการแปลงไฟล์

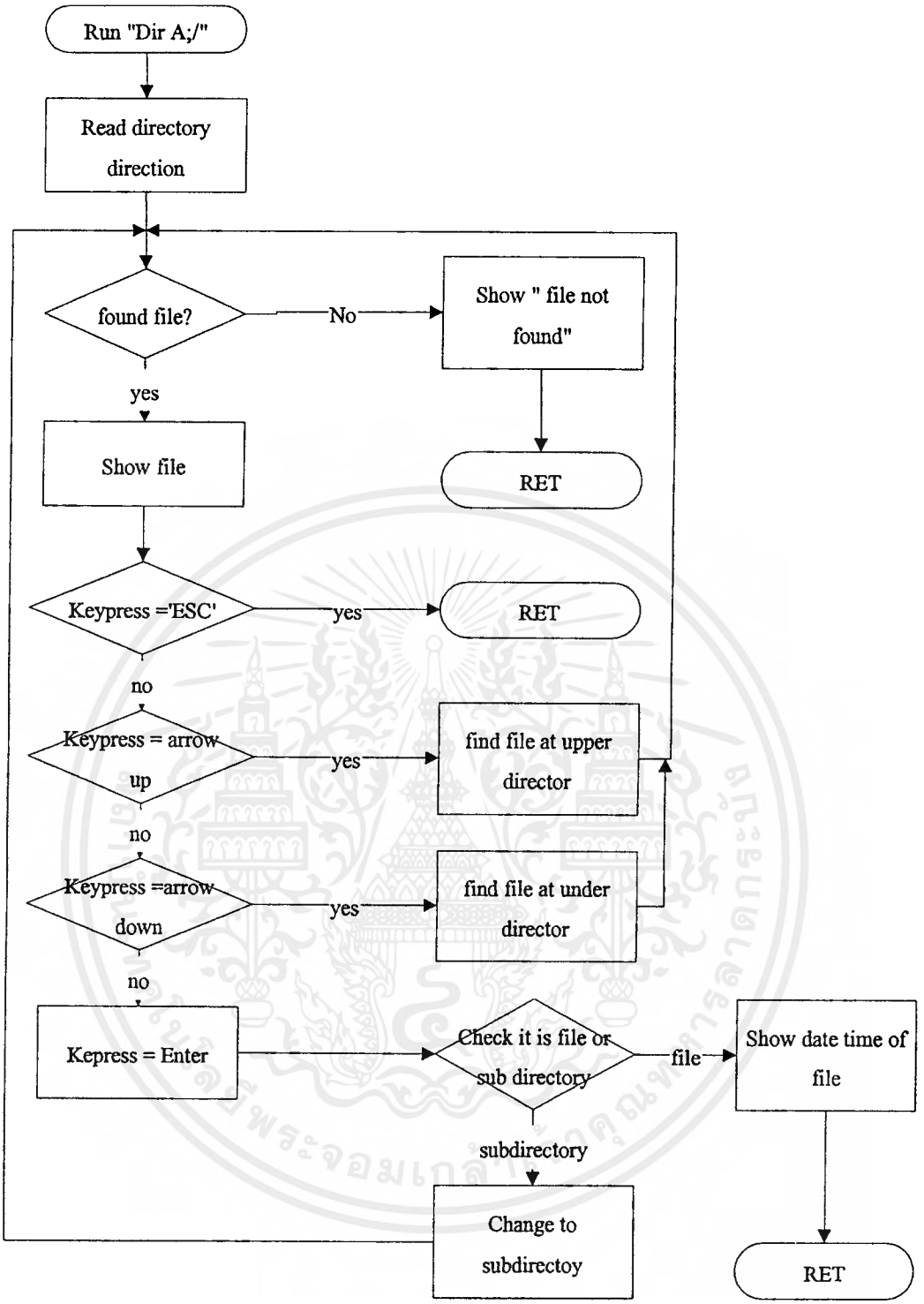


รูปที่ 3.18 แผนผังแสดงการทำงานในส่วนการควบคุมอุปกรณ์เก็บข้อมูล

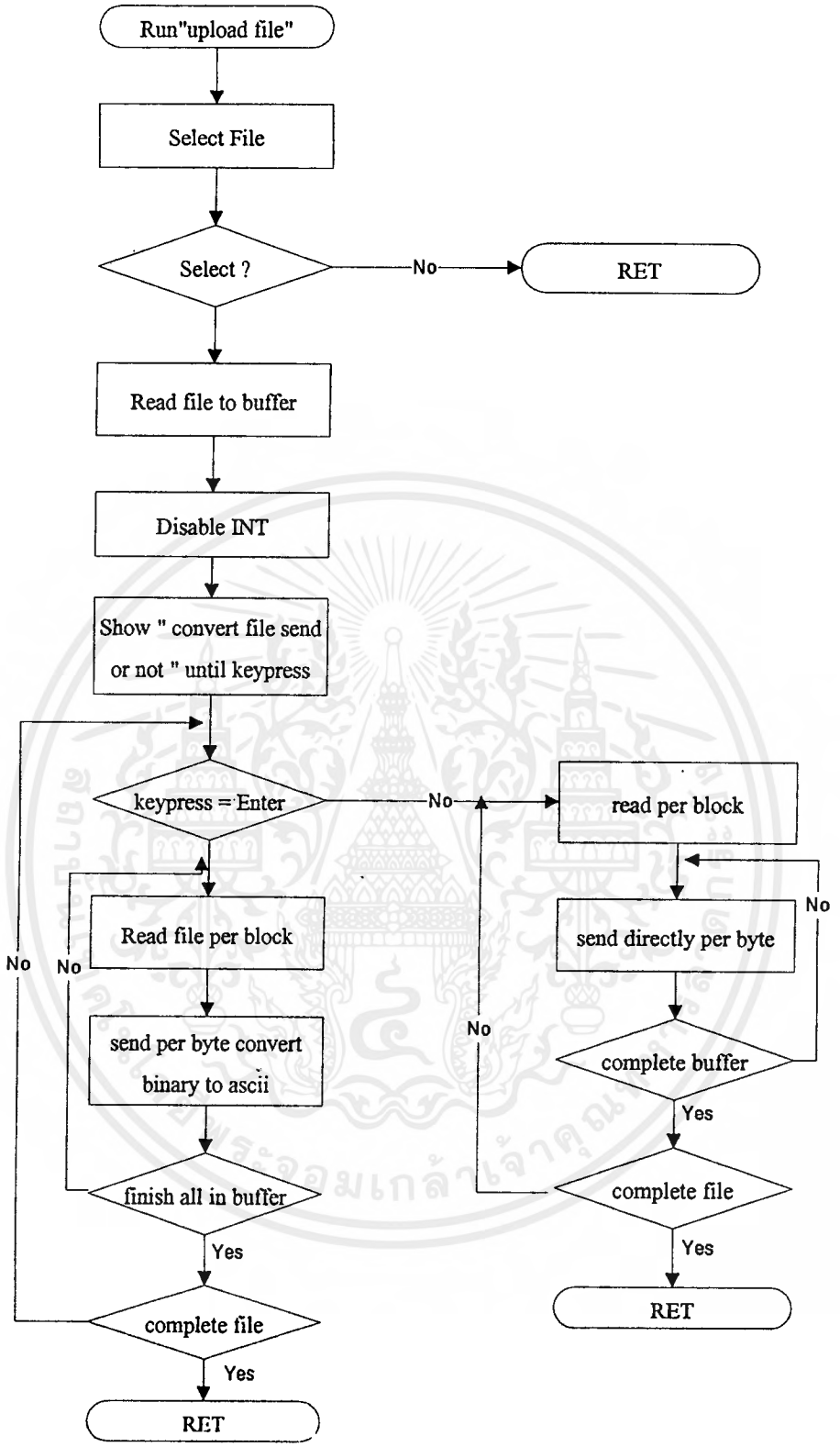
3.10.2 โปรแกรมที่ใช้สั่งงานไมโครคอนโทรลเลอร์



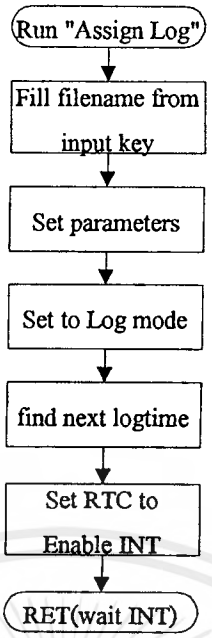
รูปที่ 3.19 แผนผังแสดงการทำงานภาพรวม



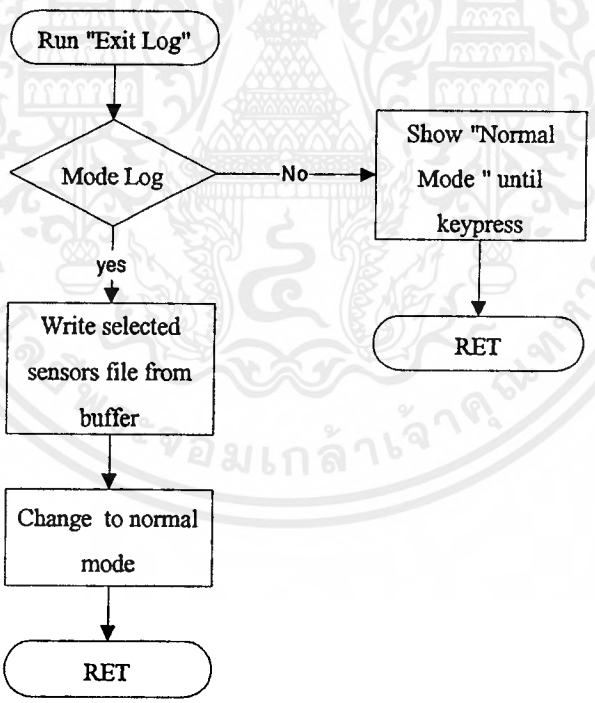
รูปที่ 3.22แผนผังแสดงการทำงานคำสั่ง DIR A:\



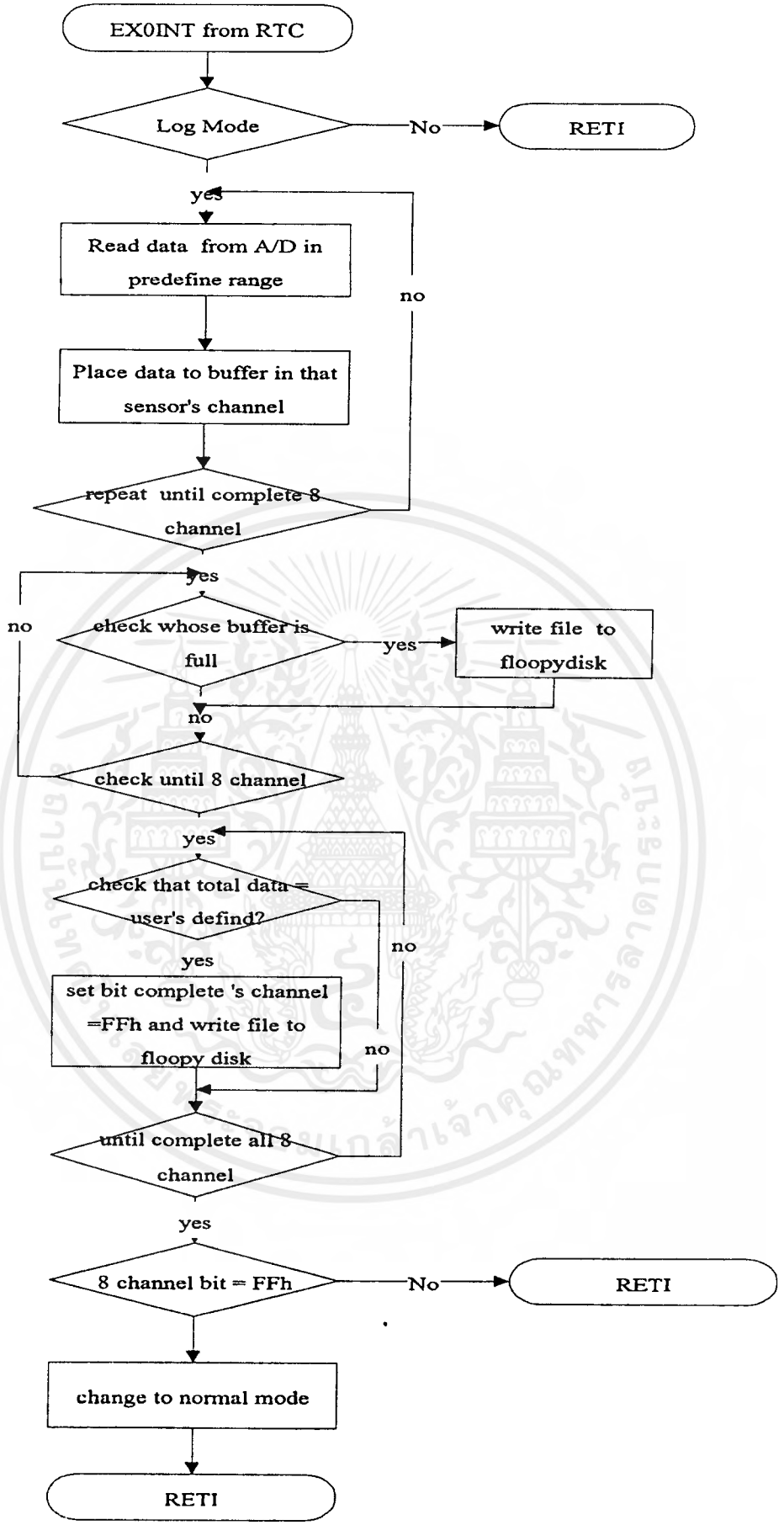
รูปที่ 3.23แผนผังแสดงการทำงาน UP LOAD FILE



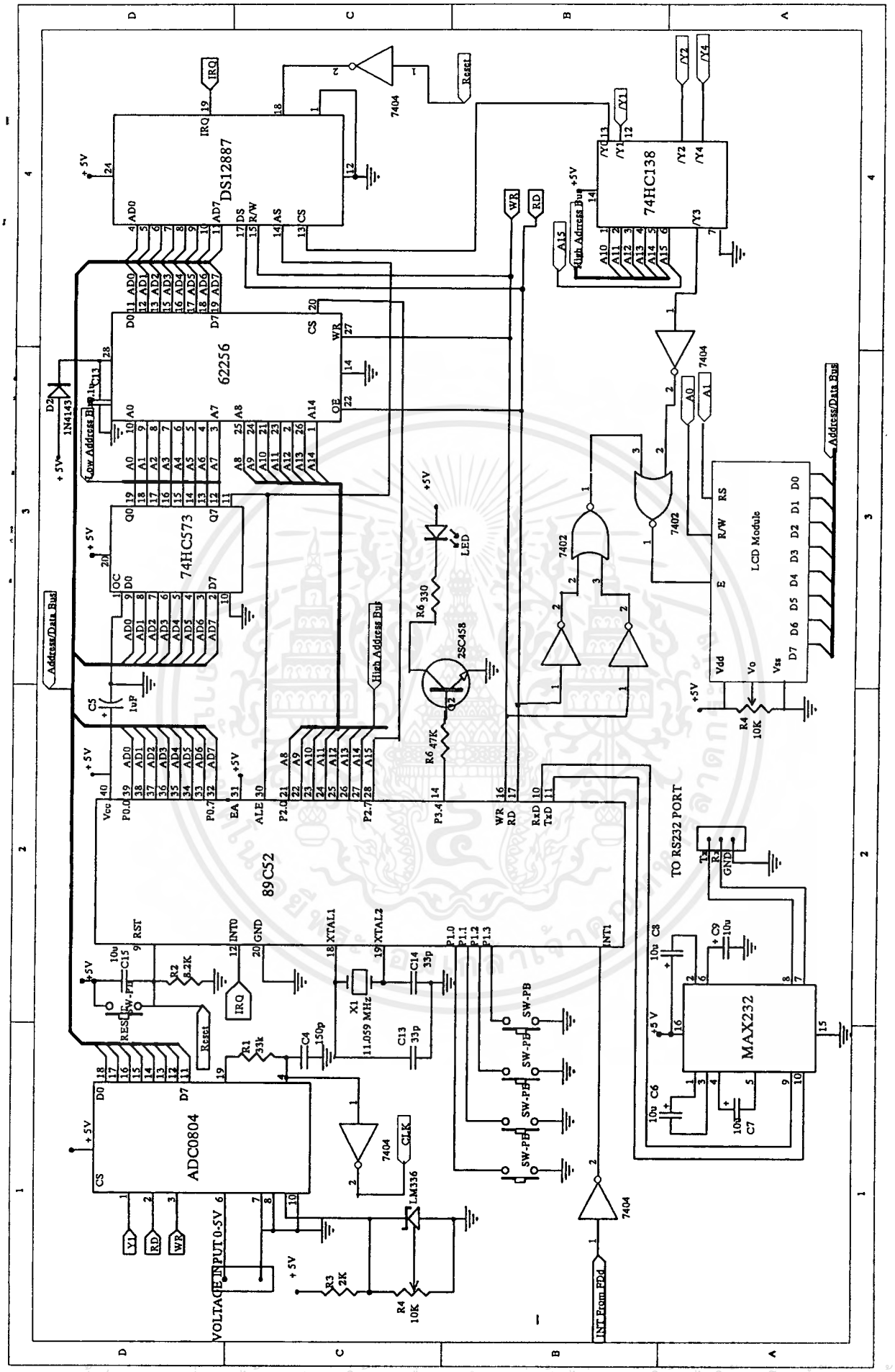
รูปที่ 3.24แผนผังแสดงการ ASSIGN LOG



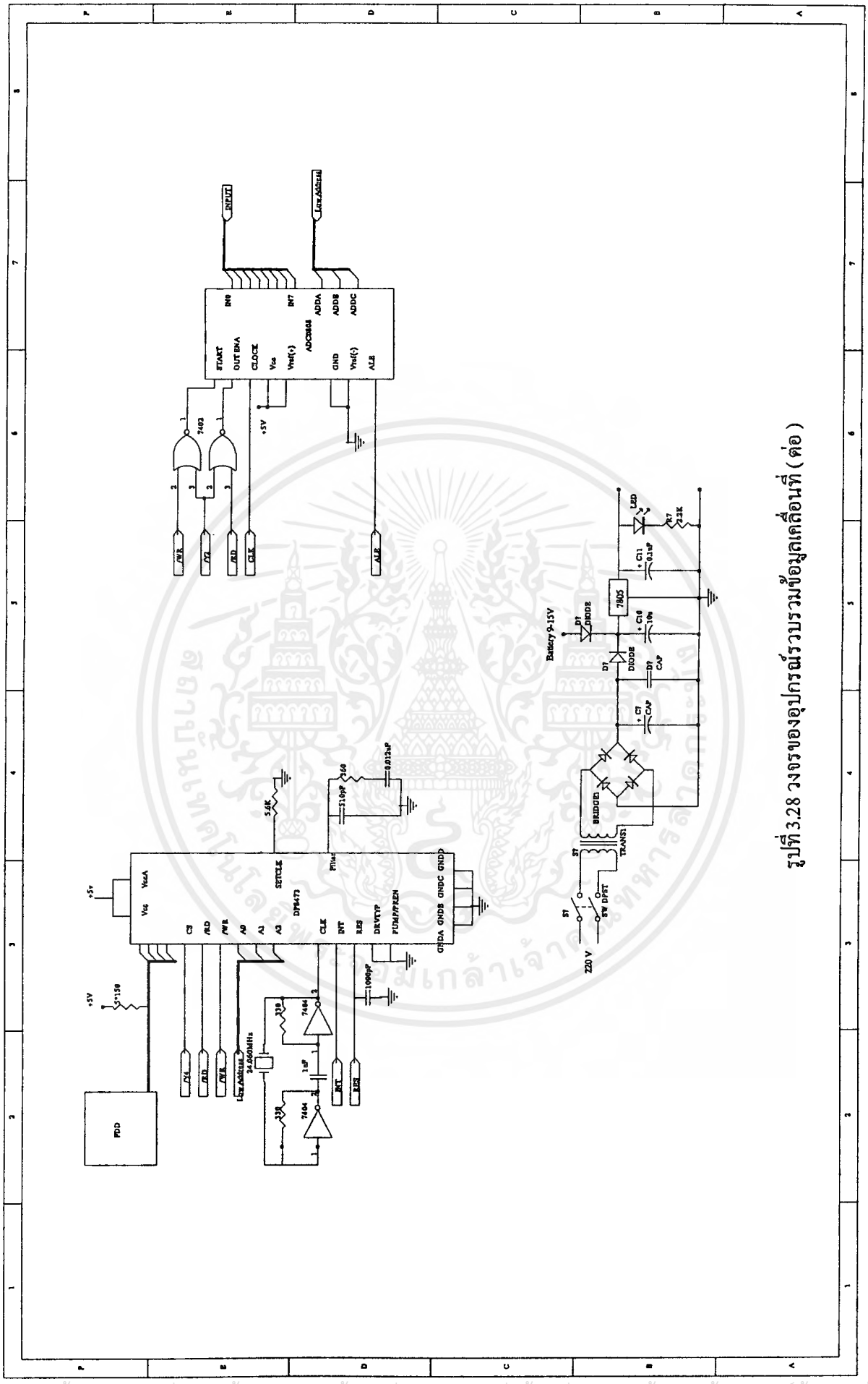
รูปที่ 3.25แผนผังการ EXIT LOG



รูปที่ 3.26 แผนผังแสดงการอินเตอร์รัพท์จากภายนอกโดย RTC



รูปที่ 3.27 วงจรของอุปกรณ์รวบรวมข้อมูลเคลื่อนที่



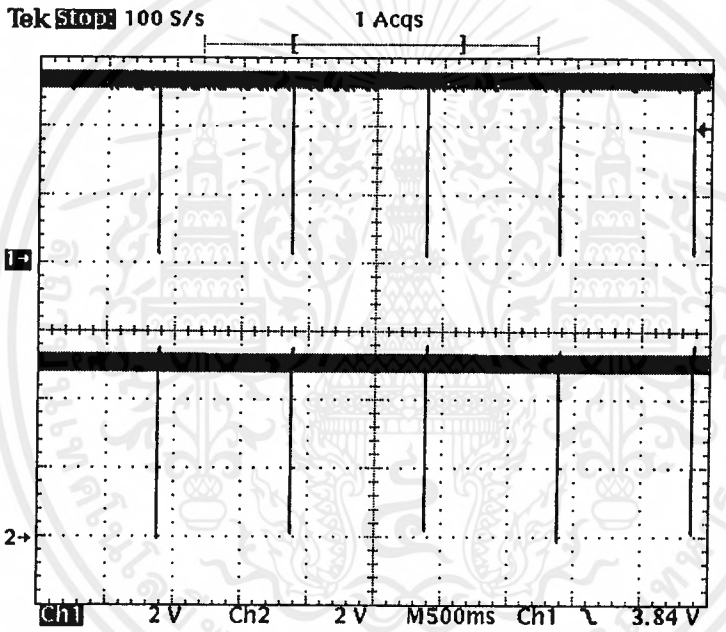
รูปที่ 3.28 วงจรของอุปกรณ์รวบรวมข้อมูลเคลื่อนที่ (ต่อ)

บทที่ 4

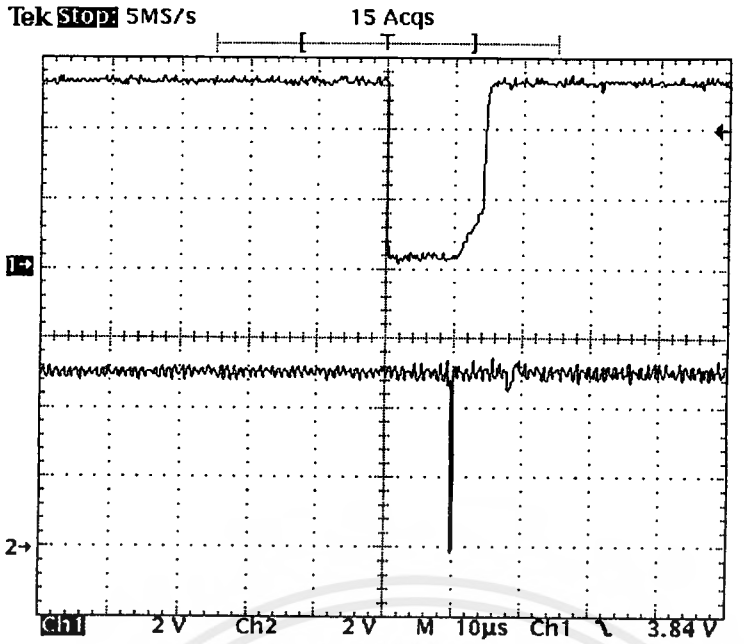
การทดลองและผลการทดลอง

4.1 ทดสอบส่วน REAL TIME CLOCK

ทำการทดลองโดยการใช้คำสั่งให้ไมโครคอนโทรลเลอร์ทำการเก็บข้อมูลทุก ๆ 1 วินาที ซึ่งจะใช้ส่วนของ RTC (real time clock) เป็นตัวนับเวลา แล้วทำการวัดสัญญาณที่ส่งมาจาก RTC ที่ขา INT0 ของไมโครคอนโทรลเลอร์และวัดสัญญาณที่ขา RD ของไมโครคอนโทรลเลอร์ซึ่งเป็นสัญญาณสำหรับเคลียร์อินเตอร์รัพท์ที่ไมโครคอนโทรลเลอร์ส่งไปยัง RTC สัญญาณที่ได้แสดงดังรูปที่ 4.3 โดย CH1 แสดงสัญญาณที่ส่งมาจาก RTC สัญญาณสำหรับเคลียร์อินเตอร์รัพท์ และ CH2 แสดงสัญญาณสำหรับเคลียร์อินเตอร์รัพท์



รูปที่ 4.1 สัญญาณที่ส่งมาจาก RTC และสัญญาณสำหรับเคลียร์อินเตอร์รัพท์



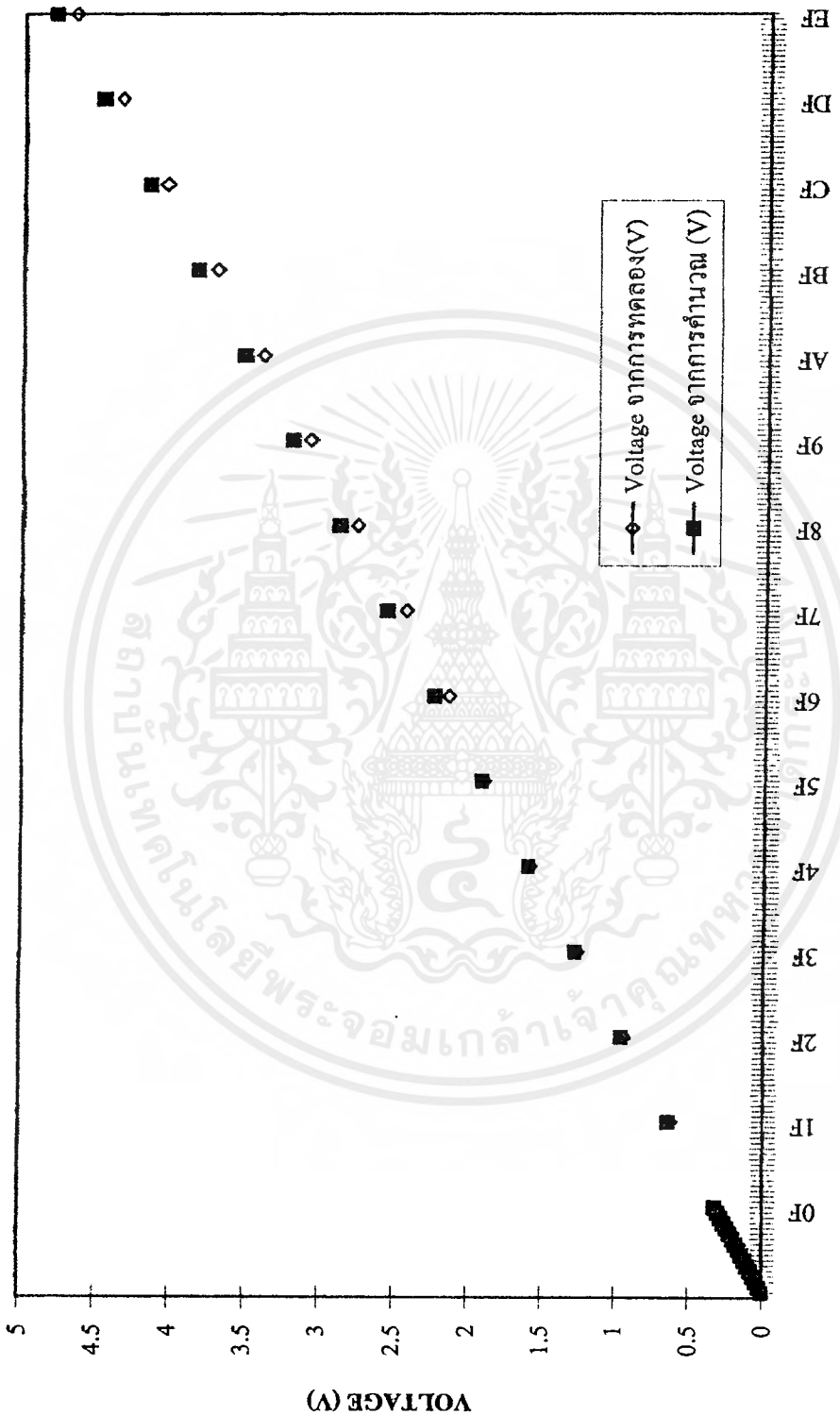
รูปที่ 4.2 ขยายรูปที่ 4.1

4.2 ทดสอบการทำงานของ A/D CONVERTER UNIT

4.2.1 ทำการทดลองโดยป้อนแรงดันที่ขา Vin ของไอซี ADC0804 ซึ่งเป็นแรงดันอินพุทแล้วสังเกตสัญญาณที่เป็นเอาท์พุท 8 บิท โดยเพิ่มระดับแรงดันจาก 0 ถึง 5 โวลท์ แล้วเปรียบเทียบค่าที่ได้จากการทดลองกับค่าที่ได้จากการคำนวณ ค่าที่ได้จากการคำนวณนั้นค่าแรงดันของแต่ละระดับจะต่างกัน $= \frac{5}{255} \approx 0.02V$ ผลที่ได้แสดงดังตารางที่ 4.1 และรูปที่ 4.5

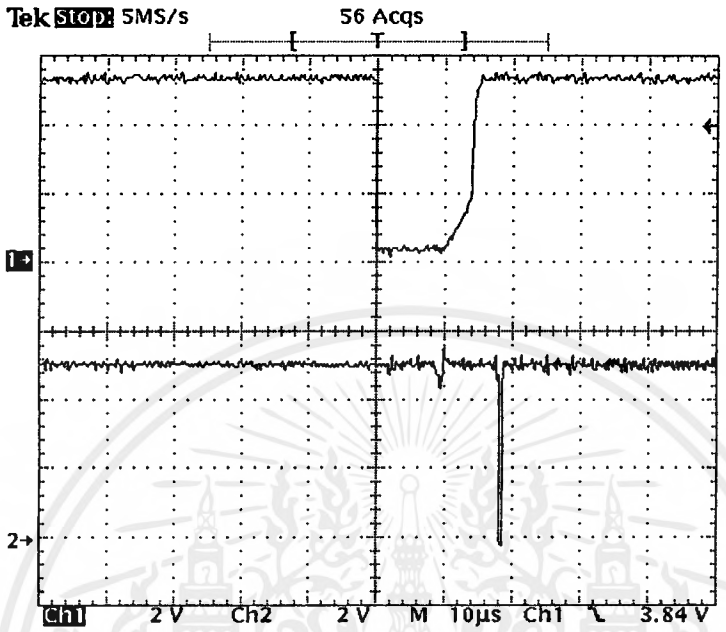
ตารางที่ 4.1 แรงดันและรหัสที่ได้จากการทดลองเทียบกับการคำนวณ

รหัส (HEX)	Voltage จากการทดลอง(V)	Voltage จากการคำนวณ (V)
00	0	0
01	0.0087	0.02
02	0.029	0.04
03	0.0488	0.06
04	0.0695	0.08
05	0.089	0.1
06	0.11	0.12
07	0.1294	0.14
08	0.15	0.16
09	0.169	0.18
0A	0.19	0.2
0B	0.2	0.22
0C	0.222	0.24
0D	0.243	0.26
0E	0.263	0.28
0F	0.284	0.3
10	0.306	0.32
20	0.62	0.64
30	0.938	0.96
40	1.26	1.28
50	1.58	1.6
60	1.898	1.92
70	2.14	2.24
80	2.43	2.56
90	2.76	2.88
A0	3.08	3.2
B0	3.4	3.52
C0	3.71	3.84
D0	4.05	4.16
E0	4.35	4.48
F0	4.66	4.8



รูปที่ 4.3 กราฟแรงดันและรหัสที่ได้จากการทดลองเทียบกับการคำนวณ

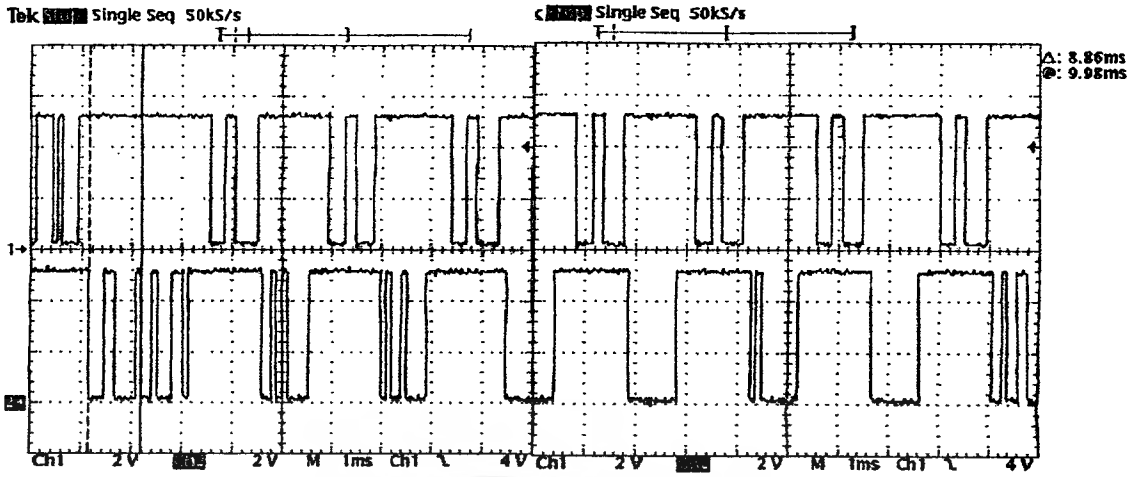
4.2.2 ทำการทดลองโดยให้ไมโครคอนโทรลเลอร์ทำการเก็บข้อมูลทุก ๆ 1 วินาทีในทำนองเดียวกันกับหัวข้อที่ 4.3 วัดสัญญาณที่ขา INT0 เพื่อสังเกตสัญญาณที่ส่งมาจาก RTC (CH1) และวัดสัญญาณที่ขา WR ของ A/D เพื่อสังเกตว่าไมโครคอนโทรลเลอร์ส่งสัญญาณมาให้ A/D ทำงานหรือไม่ (CH2) สัญญาณที่ได้แสดงดังรูปที่ 4.6



รูปที่ 4.4 สัญญาณที่ส่งมาจาก RTC และสัญญาณที่ไมโครคอนโทรลเลอร์ส่งมาให้ A/D ทำงาน

4.3 ทดสอบการส่งและรับสัญญาณระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์และส่วน RS232/TTL CONVERTER UNIT

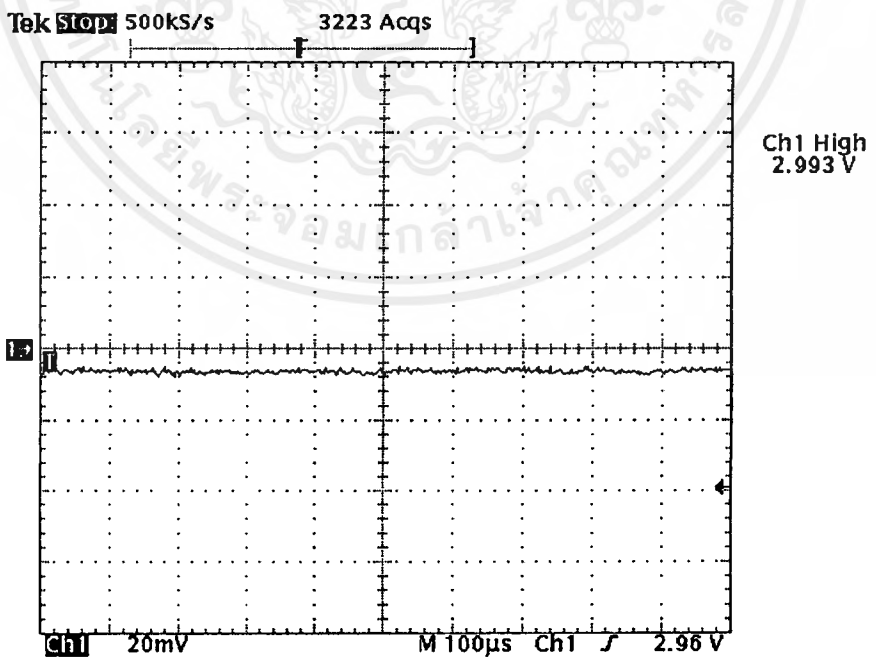
ทำการทดลองโดยใช้คำสั่งให้ไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ทำแฮนด์เช็ค (hand check) ระหว่างกัน แล้ววัดสัญญาณที่ขา TXD และขา RXD ของไมโครคอนโทรลเลอร์เพื่อสังเกตการส่งสัญญาณของไมโครคอนโทรลเลอร์ที่ขา TXD (CH2) และสังเกตการทำงานของส่วน RS232/TTL converter unit ที่ขา RXD (CH1) สัญญาณที่ได้แสดงดังรูปที่ 4.7



รูปที่ 4.5 สัญญาณที่ไมโครคอนโทรลเลอร์ส่งไปยังคอมพิวเตอร์และสัญญาณที่ไมโครคอนโทรลเลอร์รับจากคอมพิวเตอร์

4.4 ทดสอบการทำงานจริงของโครงงาน

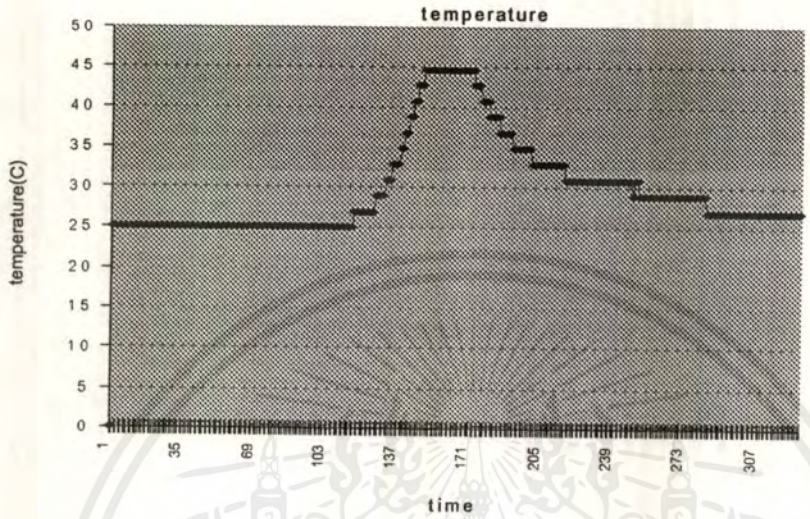
4.3.1 ทดลองโดยการวัดระดับสัญญาณจากเอาท์พุทของเซนเซอร์ที่ 26 องศาเซลเซียสได้ระดับสัญญาณ 2.99 โวลท์ โดยใช้เซนเซอร์อุณหภูมิ LM335 ซึ่งมีระดับการเปลี่ยนแปลง 10 มิลลิโวลท์/เคลวิน โดยทำการคาลิเบรต (calibrate) ค่าแล้ว



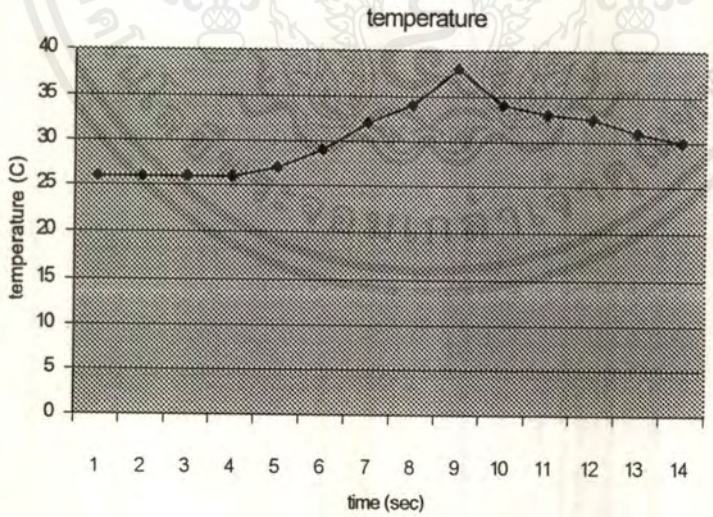
รูปที่ 4.6 สัญญาณจากเอาท์พุทของเซนเซอร์อุณหภูมิ ที่ 26 องศาเซลเซียส

4.3.2 ทดลองเก็บผลข้อมูลการเปลี่ยนแปลงของอุณหภูมิของเซนเซอร์เปรียบเทียบกับการเปลี่ยนแปลงของอุณหภูมิที่วัดจากเทอร์โมมิเตอร์เมื่อให้สภาวะการเปลี่ยนแปลงอุณหภูมิดังนี้

4.3.2.1 เมื่อเพิ่มอุณหภูมิให้สูงขึ้นช่วงหนึ่ง

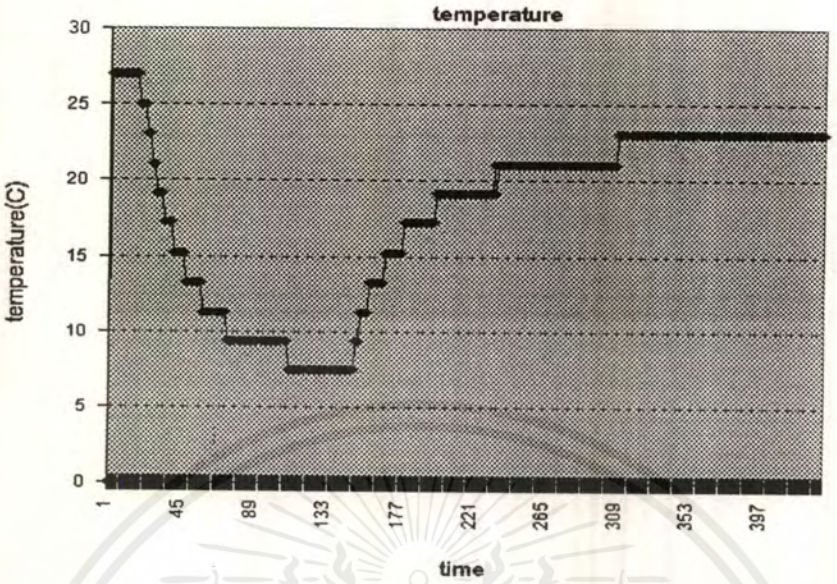


รูปที่ 4.7 การเปลี่ยนแปลงอุณหภูมิของเซนเซอร์เมื่อเพิ่มอุณหภูมิช่วงหนึ่ง

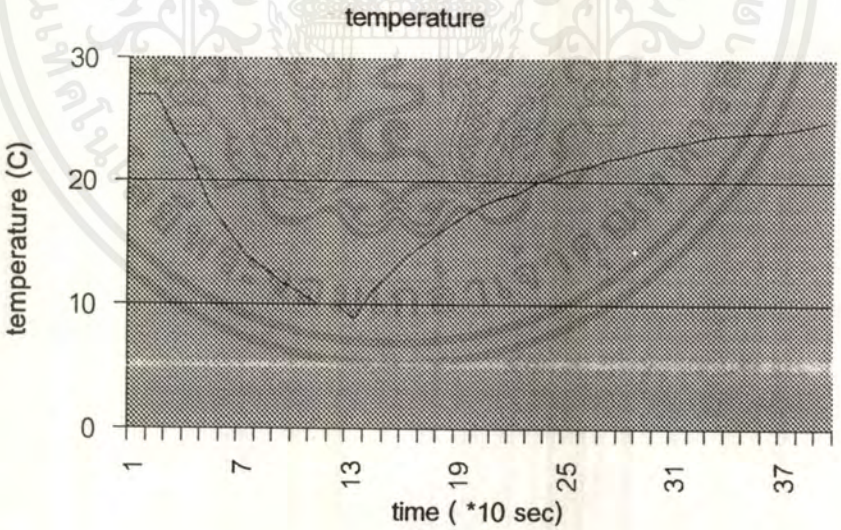


รูปที่ 4.8 การเปลี่ยนแปลงอุณหภูมิของเทอร์โมมิเตอร์เมื่อเพิ่มอุณหภูมิช่วงหนึ่ง

4.3.2เมื่อลดอุณหภูมิลงช่วงหนึ่ง

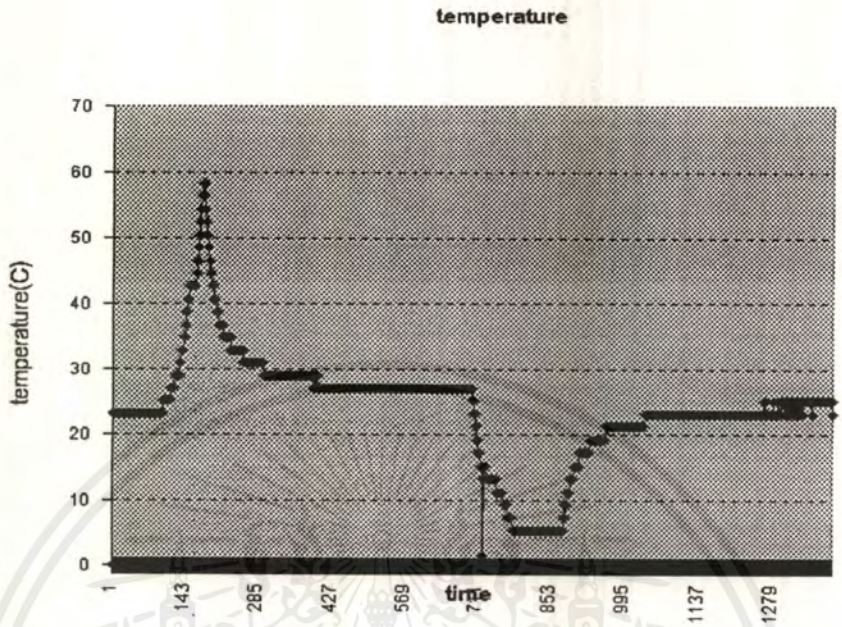


รูปที่ 4.9 การเปลี่ยนแปลงอุณหภูมิของเซนเซอร์เมื่อลดอุณหภูมิช่วงหนึ่ง

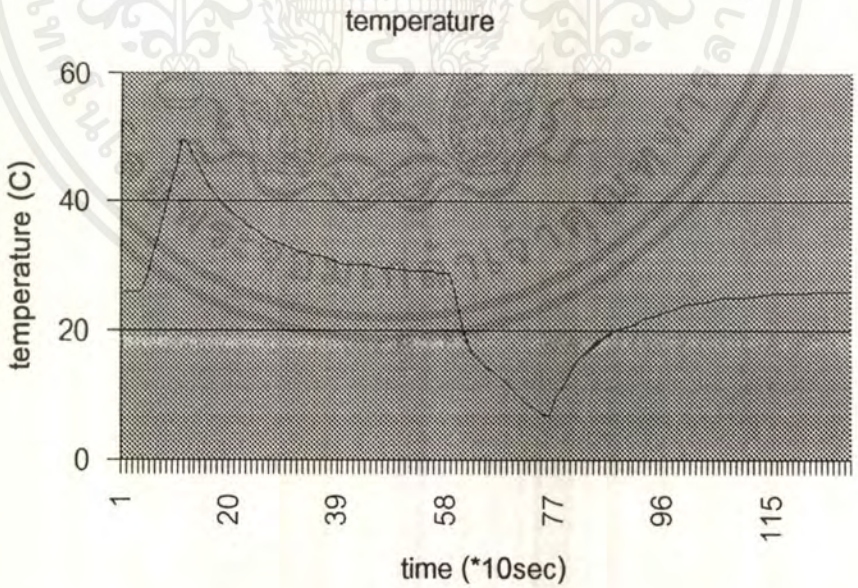


รูปที่ 4.10 การเปลี่ยนแปลงอุณหภูมิของเทอร์โมมิเตอร์เมื่อลดอุณหภูมิช่วงหนึ่ง

4.3.3เมื่อเพิ่มและลดอุณหภูมิเป็นช่วง ๆ

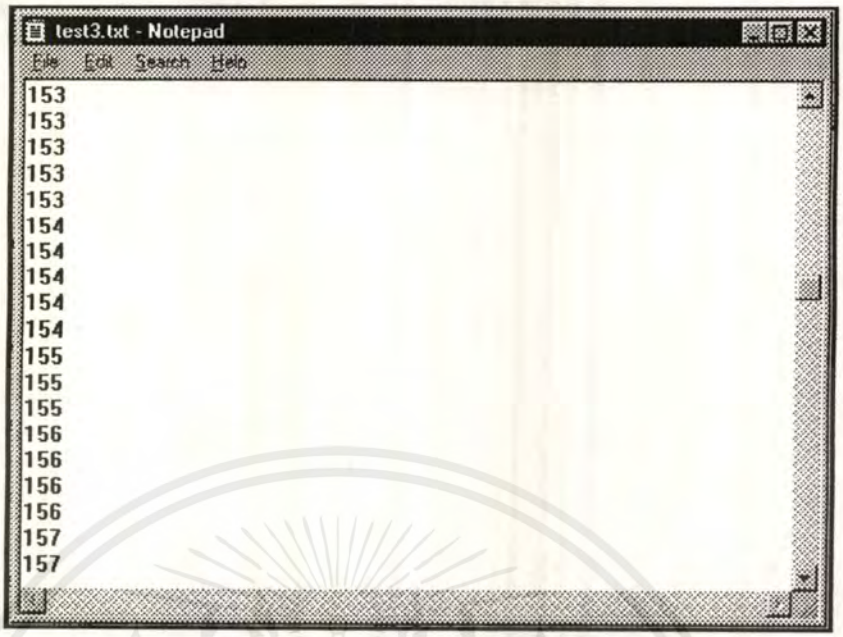


รูปที่ 4.11 การเปลี่ยนแปลงอุณหภูมิของเซนเซอร์เมื่อเพิ่มและลดอุณหภูมิ

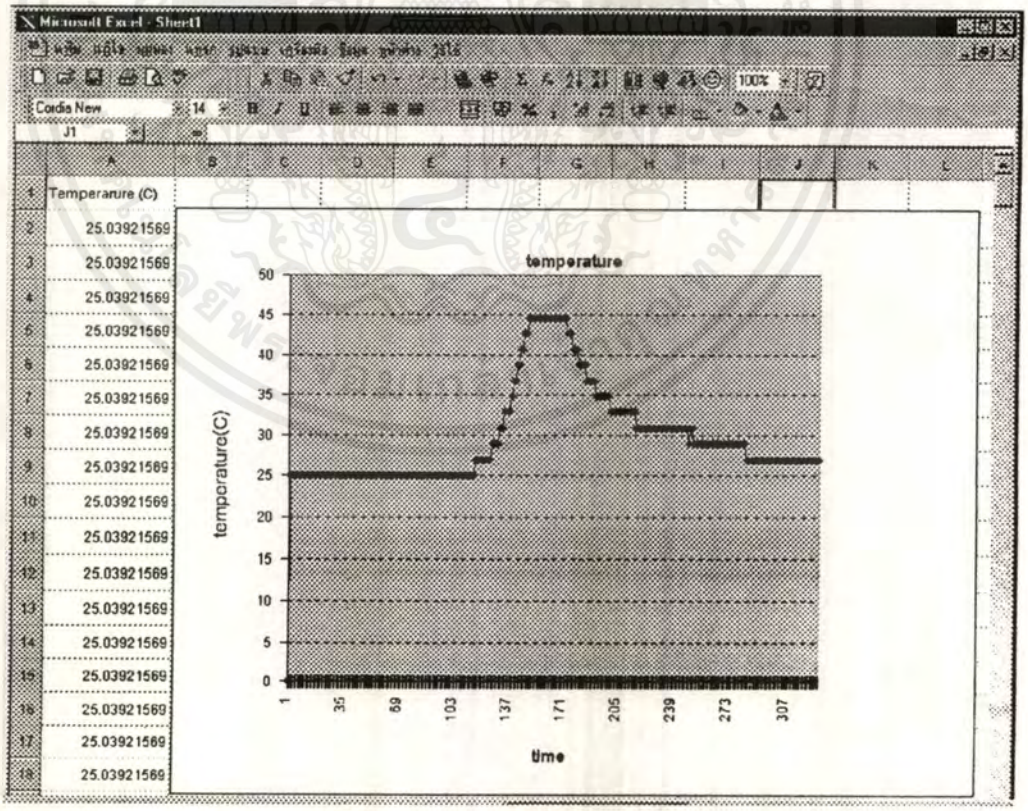


รูปที่ 4.12 การเปลี่ยนแปลงอุณหภูมิของเทอร์โมมิเตอร์เมื่อเพิ่มและลดอุณหภูมิ

4.5 การวิเคราะห์ข้อมูลเพื่อแสดงผลเป็นกราฟ



รูปที่ 4.13 ข้อมูลก่อนวิเคราะห์



รูปที่ 4.14 ข้อมูลที่วิเคราะห์แล้ว

4.6 ทดสอบการติดต่อกับผู้ใช้โดยผ่านทางคีย์บอร์ด

รูปที่ 4.15 แสดงสถานะการทำงานของส่วนติดต่อกับผู้ใช้

บทที่ 5
บทวิจารณ์และบทสรุป

โครงงานนี้มีลักษณะของการสุ่มเก็บข้อมูลเป็นคาบเวลาตามที่ผู้ใช้จะกำหนดซึ่งจะไม่สนใจรายละเอียดสัญญาณภายในช่วงนั้น ทำให้อาจจะพลาดข้อมูลสำคัญบางจุดถ้ากำหนดคาบเวลาไม่เหมาะสม แต่สามารถแก้ไขได้โดยลดคาบเวลาของการเก็บข้อมูลแล้วหาค่าเฉลี่ยของจำนวนข้อมูลสุ่มช่วงหนึ่งมาใช้ในการแสดงผลแทน เช่น ต้องการเก็บข้อมูลทุกชั่วโมง เวลาใช้จริงอาจเก็บทุก 10 นาทีแล้วเฉลี่ยภายใน 1 ชั่วโมง เป็นต้นขอบเขตในการเก็บค่าขึ้นอยู่กับ A/D จึงทำให้โครงงานนี้มีข้อจำกัดที่รายละเอียดของขนาดสัญญาณเป็น 8 บิทในช่วงค่าของแรงดันที่เข้ามาจากวงจรส่วนเซนเซอร์ 0-5 โวลท์ และเนื่องจากการใช้อุปกรณ์ฟลอปปี้ดิสก์ทำให้คาบเวลาเร็วที่สุดในการเก็บข้อมูลไม่สามารถทำได้ต่ำกว่า 1 วินาที เพราะเวลาการบันทึกข้อมูลลงแผ่นดิสก์ใช้เวลาอุปกรณ์เก็บข้อมูลนี้เหมาะสำหรับการใช้งานโดยที่ไม่จำเป็นต้องต่อกับคอมพิวเตอร์ตลอดเวลาทำให้นานพอสมควร

การเก็บข้อมูลนอกห้องทดลองเป็นไปได้ง่าย สามารถควบคุมอุปกรณ์เซนเซอร์ที่ใช้เก็บข้อมูลได้ถึง 8 ตัวพร้อมกันโดยที่แต่ละตัวสามารถควบคุมการเก็บเป็นอิสระต่อกัน ปริมาณข้อมูลสูงสุดที่สามารถเก็บได้ทั้งหมดประมาณ 1.44 ล้านตัวอย่างข้อมูล ทำให้สามารถเก็บข้อมูลจากเซนเซอร์ทั้ง 8 ตัวในทุกๆวินาที ได้นานถึง (1.44*2^20)/(8*60*60) ประมาณ 52 ชั่วโมง ความสามารถในการควบคุมมีทั้งการตั้งเวลาเริ่มต้น คาบเวลาในการเก็บ จำนวนข้อมูลการเก็บ ซึ่งสามารถกำหนดได้ง่ายจากคอมพิวเตอร์และสามารถเก็บข้อมูลที่เก็บแล้วเป็นเท็กซ์ไฟล์เรียกดูในภายหลังได้ และยังได้เขียนโปรแกรมที่อำนวยความสะดวกในการดูกราฟของลักษณะสัญญาณไว้ด้วย

แนวทางในการพัฒนาต่อ การใช้ไดนามิกแรมในการเก็บข้อมูลทำให้สามารถเก็บข้อมูลได้เร็วและมากขึ้น การเปลี่ยนอุปกรณ์การบันทึกข้อมูลเป็นอุปกรณ์ที่มีความจุมาก การพัฒนาความสามารถในการเก็บให้สามารถเก็บข้อมูลได้รายละเอียดที่มากขึ้น และสามารถเลือกเก็บข้อมูลได้ในช่วงที่มีความสำคัญหรือเฉพาะข้อมูลในลักษณะที่ต้องการ โดยพัฒนาการเขียนโปรแกรมและเพิ่มวงจรตรวจจับอัตราการเปลี่ยนแปลง นอกจากนี้ในส่วนของการสื่อสารอาจจะเพิ่มความสามารถเป็นการติดต่อกับอุปกรณ์เก็บข้อมูลในระยะไกล เช่นเพิ่มโมเด็มทำให้สามารถสั่งงานควบคุมอุปกรณ์ผ่านหน้าจอคอมพิวเตอร์ในห้องทดลองได้

หนังสืออ้างอิง

[1] กนก กุสุมาลย์นุกูลและไกรวุฒิ มั่นเสถียรสิน , "**คู่มือการเขียนโปรแกรม Delphi4**".บริษัท ซัคเซส มีเดีย จำกัด , 2541

[2] โกศน ปลื้มใจ , "**ระบบตรวจจับคุณภาพน้ำ**".ปริญญานิพนธ์ สาขาวิศวกรรมการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2538

[3] ฉัตรชัย กางกั้น , "**อินไซท์ Excel 5**".บริษัทไปรวิชั่น จำกัด , 2539

[4] ชูชัย ธนสารตั้งเจริญและคณะ , "**การสื่อสารข้อมูล**".สำนักพิมพ์ฟิสิกส์เซ็นเตอร์ , 2533

[5] นวรัตน์ ปานเทวัญและรุ่งฤดี อนุศาสน์สิริ , "**เครื่องตรวจวัดและรับส่งข้อมูลระยะไกล**".ปริญญานิพนธ์ สาขาวิศวการวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2538

[6] บุญเลิศ เอี่ยมทัศนา , "**เรียนรู้ภาษาปาสคาลด้วยเทอร์โบปาสคาล 4.0-5.0**".บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2536

[7] บุญเลิศ เอี่ยมทัศนา , "**แอดวานซ์เทอร์โบปาสคาล 4.0**". บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2536

[8] ยืน ภู่วรวรรณ , "**เทคโนโลยีฮาร์ดแวร์ IBM PC**".บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2533

[9] ผศ.สมยศ จุณณะปิยะ , "**การใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS51**".คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพมหานคร, 2537

[10] สุวิทย์ เกี่ยวศรีกุลและเอนก ขาวพรหม , "**การเชื่อมโยงเครื่องวัดระดับสัญญาณเสียงเข้ากับคอมพิวเตอร์ส่วนบุคคล**".ปริญญานิพนธ์ สาขาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2537

[11] ศิวัฒน์ ศิวะบวรและคณะ , "**การประยุกต์ใช้งานภาษาซี**". บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2535

[12] Reed Business Information Ltd , "**Electronics World Volume104 number1743 March1998**". BPCC Magazine (Carlisle)Ltd.

[13] William A. Shay , "**Understanding Data Communication and Networks**".PWS Publishing Co. , 1995

ภาคผนวก

```
unit PaChart1;

interface

uses
  Windows, Messages,      SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
   StdCtrls,    ExtCtrls;

const
   xlChart = -4109;
   xlWorksheet = -4167;

type
  TForm1 = class(    TForm)
     Button1:    TButton;
     Button2:    TButton;
     OpenDialog1:     TOpenDialog;
     Button3:    TButton;
     Image1:    TImage;
     Label1:    TLabel;
     procedure Button1Click(Sender:         TObject);
     procedure    FormDestroy(Sender:      TObject);
     procedure Button2Click(Sender:         TObject);
     procedure Button3Click(Sender:         TObject);

  private
     { Private declarations }
  public
     { Public declarations }
      MyApp: Variant;
     F1:textfile;
      s,t:string;
      x,a,b,j:integer;
  end;

var
  Form1: TForm1;

implementation
uses
     comobj,    PaSetup;
{$R *.DFM}


procedure TForm1.Button1Click(Sender:         TObject);
const
     vmax=5;
var    sheet,range:variant;
       i:integer;
       c:real;
begin
      myapp:=    createOleObject('    excel.application');
      myapp.visible:= true;
      myapp.workbooks.add(     xlworksheet);
      myapp.workbooks[1].worksheets[1].         name:= 'Temp1';
      sheet:=    myapp.workbooks[1].worksheets['Temp1'];
      sheet.columns.columns[1].columnwidth:=15;
      sheet.cells[1,1]:= '     Temperarure (C)';
     i:= 2 ;
     if OpenDialog1.Execute then
```

```
        begin
          AssignFile(F1, OpenDialog1.Filename);
        Reset(F1);
        while not Eof(F1) do
             begin
               ReadIn(F1, s);
             a:=     StrToInt(s);
             c:= ((a*vmax/255)/0.01)-273 ;
               sheet.cells[i,1]:= c;
             i:=i+1;
             end;
          CloseFile(F1);
          end;//if
end;//procedure1

procedure TForm1.FormDestroy(Sender:        TObject);
begin
      if not    varisEmpty(  MyApp) then
         MyApp.quit;
end;

procedure TForm1.Button2Click(Sender:        TObject);
begin
     Form1.Close;
end;

procedure TForm1.Button3Click(Sender:        TObject);
begin
     Form1.Hide;
     Form2.Show;
end;

end.


unit PaChart2;

interface

uses
  Windows, Messages,     SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  StdCtrls;

type
  TForm3 = class(    TForm)
    Edit1:    TEdit;
    Button1:    TButton;
    Label1:    TLabel;
    procedure Button1Click(Sender:        TObject);
  private
    { Private declarations }
  public
     inputname :string{ Public declarations }
  end;

var
 Form3: TForm3;

implementation
```

```
{$R *.DFM}

procedure TForm3.Button1Click(Sender:        TObject);
begin
     inputname := edit1.text ;
    Form3.hide ;
end;

end.unit   PaSetup;

interface

uses
  Windows, Messages,      SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
   StdCtrls,    ComCtrls,    CPDrv,   ExtCtrls;

type  Tarraygroup = array [1..8] of       boolean ;
      Tdatfile = record
       start_d :string ;
       start_h :string ;
       start_m :string ;
       start_s :string ;
       delay_h :string ;
       delay_m :string ;
       delay_s :string ;
       totalbyte :string ;
     end;
type
  TForm2 = class(    TForm)
    Label52:    TLabel;
     Setup_Now:    TButton;
    PageControl1:     TPageControl;
    Sensor1:    TTabSheet;
     Start_Hour:    TLabel;
     Start_Minute:     TLabel;
     Start_Sec:    TLabel;
     Delay_Hour:    TLabel;
     Delay_Minute:     TLabel;
     Delay_Sec:    TLabel;
    Label50:    TLabel;
    ComboBox2:     TComboBox;
    ComboBox3:     TComboBox;
    ComboBox5:     TComboBox;
    combobox4:     TComboBox;
    ComboBox6:     TComboBox;
    ComboBox7:     TComboBox;
    ComboBox57:     TComboBox;
    Sensor2:    TTabSheet;
    Label2:    TLabel;
    Label3:    TLabel;
    Label4:    TLabel;
    Label5:    TLabel;
    Label6:    TLabel;
    Label7:    TLabel;
    Label51:    TLabel;
    ComboBox14:     TComboBox;
    ComboBox58:     TComboBox;
    Sensor3:    TTabSheet;
    Label9:    TLabel;
    Label10:    TLabel;
```

```
    Label11:    TLabel;
    Label12:    TLabel;
    Label13:    TLabel;
    Label14:    TLabel;
    Label53:    TLabel;
    ComboBox17:     TComboBox;
    ComboBox19:     TComboBox;
    ComboBox21:     TComboBox;
    ComboBox20:     TComboBox;
    ComboBox18:     TComboBox;
    ComboBox16:     TComboBox;
    ComboBox59:     TComboBox;
    Sensor4:    TTabSheet;
    Label16:    TLabel;
    Label17:    TLabel;
    Label18:    TLabel;
    Label19:    TLabel;
    Label20:    TLabel;
    Label21:    TLabel;
    Label54:    TLabel;
    ComboBox24:     TComboBox;
    ComboBox26:     TComboBox;
    ComboBox28:     TComboBox;
    ComboBox27:     TComboBox;
    ComboBox25:     TComboBox;
    Combobox23:     TComboBox;
    ComboBox60:     TComboBox;
    Sensor5:    TTabSheet;
    Label23:    TLabel;
    Label24:    TLabel;
    Label25:    TLabel;
    Label26:    TLabel;
    Label27:    TLabel;
    Label28:    TLabel;
    Label55:    TLabel;
    ComboBox31:     TComboBox;
    ComboBox33:     TComboBox;
    ComboBox35:     TComboBox;
    ComboBox34:     TComboBox;
    ComboBox32:     TComboBox;
    ComboBox30:     TComboBox;
    ComboBox61:     TComboBox;
    Sensor6:    TTabSheet;
    Label30:    TLabel;
    Label31:    TLabel;
    Label32:    TLabel;
    Label33:    TLabel;
    Label34:    TLabel;
    Label35:    TLabel;
    Label56:    TLabel;
    ComboBox38:     TComboBox;
    ComboBox40:     TComboBox;
    ComboBox42:     TComboBox;
    ComboBox41:     TComboBox;
    ComboBox39:     TComboBox;
    ComboBox37:     TComboBox;
    combobox62:     TComboBox;
    Sensor7:    TTabSheet;
    Label37:    TLabel;
    Label38:    TLabel;
    Label39:    TLabel;
```

```
    Label40:    TLabel;
    Label41:    TLabel;
    Label42:    TLabel;
    Label57:    TLabel;
    ComboBox45:     TComboBox;
    ComboBox47:     TComboBox;
    ComboBox49:     TComboBox;
    ComboBox48:     TComboBox;
    ComboBox46:     TComboBox;
    ComboBox44:     TComboBox;
    combobox63:     TComboBox;
    Sensor8:    TTabSheet;
    Label44:    TLabel;
    Label45:    TLabel;
    Label46:    TLabel;
    Label47:    TLabel;
    Label48:    TLabel;
    Label49:    TLabel;
    Label58:    TLabel;
    ComboBox52:     TComboBox;
    ComboBox54:     TComboBox;
    ComboBox56:     TComboBox;
    ComboBox55:     TComboBox;
    ComboBox53:     TComboBox;
    combobox51:     TComboBox;
    ComboBox64:     TComboBox;
     Select_sensor:     TGroupBox;
    Sensor_1:     TCheckBox;
    Sensor_2:     TCheckBox;
    Sensor_3:     TCheckBox;
    Sensor_4:     TCheckBox;
    Sensor_5:     TCheckBox;
    Sensor_6:     TCheckBox;
    Sensor_7:     TCheckBox;
    Sensor_8:     TCheckBox;
     Exitnow:   TButton;
    CommPortDriver1:     TCommPortDriver;
    Timer1:    TTimer;
     selcommport:    TComboBox;
    connect:    TButton;
    Disconnect:     TButton;
    Panel2:    TPanel;
    Panel3:    TPanel;
    convert:    TButton;
    SaveDialog1:     TSaveDialog;
    OpenDialog1:     TOpenDialog;
    combobox9:    TComboBox;
    combobox11:     TComboBox;
    combobox12:     TComboBox;
    combobox13:     TComboBox;
    combobox10:     TComboBox;
    procedure    FormCreate(Sender:     TObject);
    procedure    Setup_NowClick(Sender:      TObject);
    procedure    DisconnectClick(Sender:       TObject);
    procedure    connectClick(Sender:     TObject);
    procedure Timer1Timer(Sender:        TObject);
    procedure    ExitnowClick(Sender:     TObject);
    procedure    convertClick(Sender:     TObject);

  private
    { Private declarations }
```

```
    checkdata :  Tarraygroup;
    datfile :array [1..8] of      Tdatfile ;
   procedure    readdata(  sel : Tarraygroup);
   procedure check;
   procedure    Updatestat;
   procedure    senddirectoryname(    a:string);
   procedure con2num(c1,c2:char;var        num :byte);
   procedure    hextable(  c:char;var n :byte);
  public
   { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Unit1, PaChart2, PaChart1, report;

{$R *.DFM}
function   Hour:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
  with  SystemTime do
   result:=wHour  ;

end;
function   min:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
  with  SystemTime do
   result:=wMinute  ;

end;
function   sec:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
  with  SystemTime do
   result:=wsecond  ;

end;
function   Day:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
  with  SystemTime do
   result:=wday  ;

end;
function   month:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
```

```
  with  SystemTime do
    result:=wmonth  ;
end;
function   year:integer  ;
var
  SystemTime:    TSystemTime;
begin
  GetLocalTime(   SystemTime);
  with  SystemTime do
    result:=wyear  ;
end;

procedure TForm2.senddirectoryname(       a:string);
var  lengths,i:byte ;
     direcname :string[8];
begin
      lengths:=length(a);
    for   i := 1 to 8-lengths do
     a := a+' ';
     direcname := a ;
    for i:= 1 to 8 do
     commportdriver1.sendchar(      direcname[  i]);
end ;

procedure TForm2.Updatestat;
const _ databits: array[   TDataBits] of string = ('5','6','7','8');
      _parity: array[    TParity] of string = ('N','E','O','M','S');
      _ stopbits: array[   TStopBits] of string = ('1','1.5','2');
      _ hwflow: array[   THwFlowControl] of string = ('      None','None+DTR
on','RTS/CTS');
      _ swflow: array[   TSwFlowControl] of string = ('      None','XON/XOFF');
var s,s1,s2: string;
begin
  // Updates the connection status
  if commportdriver1.Connected then
    s := 'Connected to "' + commportdriver1.PortName + '"'
  else
    s := 'Not connected';

  // Show current frame settings
  s1 :=   IntToStr( commportdriver1.BaudRateValue ) + ',' +
       _ databits[ commportdriver1.DataBits ] + ',' +
       _parity[ commportdriver1.Parity ] + ',' +
       _ stopbits[ commportdriver1.StopBits ];


  // Show current flow control settings
  s2 := '  Hw:' + _ hwflow[ commportdriver1.HwFlow ] + ' -        Sw:' +
  _swflow[ commportdriver1.SwFlow ];
  //Display at panel 2
  panel2.Caption := s+'  '+s1+' '+s2;
end;
procedure TForm2.Check;
begin

  if Sensor_1.checked  then
        begin
             checkdata[1] := true ;
        end ;
   if Sensor_2.checked  then
        begin
```

```
                checkdata[2] := true ;
            end ;
        if Sensor_3.checked  then
            begin
                checkdata[3] := true ;
            end ;
        if Sensor_4.checked  then
            begin
                checkdata[4] := true ;
            end ;
        if Sensor_5.checked  then
            begin
                checkdata[5] := true ;
            end ;
         if Sensor_6.checked  then
            begin
                checkdata[6] := true ;
            end ;
         if Sensor_7.checked  then
            begin
                checkdata[7] := true ;
            end ;
         if Sensor_8.checked  then
            begin
                checkdata[8] := true ;
            end ;

end;

procedure TForm2.readdata (     sel : Tarraygroup) ;
var a: byte  ;
begin
  for a := 1 to 8 do
  if  sel[a] then
     begin
      case a  of
       1:   with    datfile[1] do
       begin
          start_h :=  combobox2.text;
          start_m :=  combobox3.text;
          start_s :=  combobox4.text;
          delay_h :=  combobox5.text;
          delay_m := combobox6.text;
          delay_s := combobox7.text;
          totalbyte := combobox57.text;
       end;
       2:   with    datfile[2] do
       begin
          start_h := combobox9.text;
          start_m := combobox10.text;
          start_s := combobox11.text;
          delay_h := combobox12.text;
          delay_m := combobox13.text;
          delay_s := combobox14.text;
          totalbyte := combobox58.text;
        end;
        3:   with    datfile[3] do
       begin
          start_h := combobox16.text;
          start_m := combobox17.text;
          start_s := combobox18.text;
```

```
          delay_h := combobox19.text;
          delay_m := combobox20.text;
          delay_s := combobox21.text;
          totalbyte := combobox59.text;
        end;
         4:   with     datfile[4] do
       begin
          start_h := combobox23.text;
          start_m := combobox24.text;
          start_s := combobox25.text;
          delay_h := combobox26.text;
          delay_m := combobox27.text;
          delay_s := combobox28.text;
          totalbyte := combobox60.text;
        end;
         5:   with     datfile[5] do
       begin
          start_h := combobox30.text;
          start_m := combobox31.text;
          start_s := combobox32.text;
          delay_h := combobox33.text;
          delay_m := combobox34.text;
          delay_s := combobox35.text;
          totalbyte := combobox61.text;
        end;
         6:   with     datfile[6] do
       begin
           start_h := combobox37.text;
          start_m := combobox38.text;
          start_s := combobox39.text;
          delay_h := combobox40.text;
          delay_m := combobox41.text;
          delay_s := combobox42.text;
          totalbyte := combobox62.text;
        end;
         7:   with     datfile[7] do
       begin
          start_h := combobox44.text;
          start_m := combobox45.text;
          start_s := combobox46.text;
          delay_h := combobox47.text;
          delay_m := combobox48.text;
          delay_s := combobox49.text;
          totalbyte := combobox63.text;
        end;
         8:   with     datfile[8] do
       begin
          start_h := combobox51.text;
          start_m := combobox52.text;
          start_s := combobox53.text;
          delay_h := combobox54.text;
          delay_m := combobox55.text;
          delay_s := combobox56.text;
          totalbyte := combobox64.text;
        end;
        end;{case}
        end;{if}
end;{procedure}

procedure TForm2.FormCreate(Sender:        TObject);
  var  i:integer ;
```

```
begin

    for i:=1 to 8 do
    begin
     checkdata[  i] := false;
    with   datfile[  i] do
    begin
        start_d := '0';      //not use
        start_h := '0';
        start_m := '0';
        start_s := '0';
        delay_h := '0';
        delay_m := '0';
        delay_s := '0';
        totalbyte := '0';
    end ;
    end;
end;

procedure TForm2.Setup_NowClick(Sender:        TObject);
var s : array[1..8,1..6] of byte ;
    t : array [1..8 ] of int64 ;
     tmp :int64;
     i, j, totallo, totalmid, totalhi :byte ;
     checkbit, c :byte;
     finame :string ;
begin
   if not (commportdriver1.connected) then
   begin
     messageDlg('connect com port      first!',mtwarning,[    mbOK],0);
//display error
    exit ;
   end;
   repeat
   Form3.show ;
    finame := Form3.inputname ;
   if length(   finame) > 8 then
      messageDlg('message length <= 8',mtwarning,[         mbOK],0);  //display
error
   until length(   finame) <= 8 ;
    tmp := 0 ;
   check;
    readdata(  checkdata);
    checkbit := 0;  // check bit for assign channel to          datalogger but
now use
   if  checkdata[1] then    checkbit:=checkbit or $01 ;
   if  checkdata[2] then    checkbit:=checkbit or $02 ;
   if  checkdata[3] then    checkbit:=checkbit or $04 ;
   if  checkdata[4] then    checkbit:=checkbit or $08 ;
   if  checkdata[5] then    checkbit:=checkbit or $16 ;
   if  checkdata[6] then    checkbit:=checkbit or $32 ;
   if  checkdata[7] then    checkbit:=checkbit or $64 ;
   if  checkdata[8] then    checkbit:=checkbit or $128;
   for i:= 1 to 8 do
   begin
      with   datfile[  i] do
      begin
        t[  i] := strtoint64(   totalbyte) ;
        s[i,1            int(  delay_s ) ;
        s[i,2           toint(  delay_m ) ;
        s[i,3         trtoint(  delay_h) ;
```

```
        s[i,4] :=    strtoint(  start_s) ;
        s[i,5] :=    strtoint(  start_m) ;
        s[i,6] :=    strtoint(  start_h) ;
       end;
          tmp :=  tmp+t[ i] ;
    end;{for}

    if  tmp >= 1400000 then
     begin
      messageDlg('total     accquistion data not over 1.4Mbit',mtwarning,
[mbOK],0);  //display error
     exit;
     end;


    with commportdriver1 do //send header and 1byte
      begin
       sendbyte($85);
       sendbyte($A7);
       senddirectoryname(    finame);
      end;
    with commportdriver1 do
    begin
    for i:= 1 to 8 do
    begin
         totallo := t[   i] mod 256 ;
         tmp := t[  i] div 256     ;
         totalmid :=    tmp mod 256 ;
         totalhi :=   tmp div 256 ;
        if   totalhi > 255 then // display error ;
         begin
           messageDlg('total data exceed 24 byte ',       mtwarning    ,
[mbOK],0);
          exit;
         end;
         sendbyte(  totallo);
         sendbyte(  totalmid);
         sendbyte(  totalhi);
        for j:= 1 to 6 do
            sendbyte(s[   i,j]);
    end ;{for}
    if  readbyte(c)
     Then
        if c = $15 then
            MessageDlg('Send complete',       mtInformation,
          [  mbOk], 0)
         else
         messageDlg('   Datalogger is    busy',mtwarning,[    mbOK],0)
     Else
         messageDlg('Comport     doesn"t received flow control
bit',mtwarning,[    mbOK],0)
    end{with}
    end;

procedure TForm2.DisconnectClick(Sender:          TObject);
begin
    if not (commportDriver1.Connected) then
      Exit;

      // Disconnect
  commportDriver1.Disconnect;
```

```
  if not commportDriver1.Connected then
  begin
     panel2.caption := 'disconnect';
      connect.enabled := true ;
  end;
end;

procedure TForm2.connectClick(Sender:         TObject);
var  newportname :string ;
begin
   newPortName := Trim(      Selcommport.Text );
   if  newPortName = '' then
   begin
        messageDlg('Please select items in        combobox',mtwarning,
[mbOK],0);
       exit; {exit this procedure}
   end;
   commportDriver1.Portname :=       newportname ;
   if commportDriver1.connected = false then  // check for port have
connect yet ??
   begin
      commportDriver1.connect
   end;
   if commportDriver1.connected = false then  //  program can not
connect check port first
       messageDlg('check your      port',mtwarning,[    mbOK],0);
   end;

procedure TForm2.Timer1Timer(Sender:         TObject);
begin

 panel3.caption := 'Today is  ' +        DateToStr(Date)+
                   ' and time is  ' +         TimeToStr(Time);
 updatestat  ;
end;



procedure TForm2.ExitnowClick(Sender:         TObject);
begin
     Form2.Close;
     Form1.Show;
end;

procedure Tform2.hextable(      c:char;var n :byte);
  begin
    c:=  upcase(c);
    case c of
      '0' : n := 0 ;
      '1' : n := 1 ;
      '2' : n := 2 ;
      '3' : n := 3 ;
      '4' : n := 4 ;
      '5' : n := 5 ;
      '6' : n := 6 ;
      '7' : n := 7 ;
      '8' : n := 8 ;
      '9' : n := 9 ;
      'A' : n := 10 ;
      'B' : n := 11  ;
      'C' : n :=  12 ;
```

```
        'D' : n := 13 ;
        'E' : n := 14 ;
        'F' : n := 15 ;
    end;
    end;

procedure Tform2.con2num(c1,c2:char;var          num :byte);
   var n1,n2 :byte;
   begin
      hextable(c1,n1);
      hextable(c2,n2);
      num := n1*16 + n2;
   end;

procedure TForm2.convertClick(Sender:          TObject);
var  finame,filenam :string ;
     fp2,fp1 :   textfile  ;
      intnum :byte ;
      txt :  string ;
     n1,n2:char;

begin
    if not(opendialog1.execute) then  exit;
     finame := opendialog1.filename;
    if (not    fileExists(  finame) ) then
     begin
        messageDlg('File must receive before        save',mtwarning,[    mbOK],0);
       exit ;
     end
    else
     begin
        if not(SaveDialog1.Execute) then  exit;
         AssignFile(Fp2,  SaveDialog1.Filename);
         AssignFile(Fp1,finame );
       Reset(Fp1);
       filenam:=SaveDialog1.Filename;
       if   FileExists(SaveDialog1.filename) then
       if   Application.MessageBox( 'Do you want to replace this file',
                          'Warning',
                         MB_OKCANCEL or MB_ICONERROR ) <> IDOK
       then
          begin
                  closefile(fp1);
              exit;
          end;
       Rewrite(Fp2);
       n1:='0'; n2:='0';
     end;
       while not(eof(fp1))  do
       begin
                  repeat
                  read(fp1,n1);
                  until (n1 <>' ')and  (n1 <> #13)  and (n1 <> #10)  ;
                  repeat
                  read(fp1,n2);
                  if n2 = ' '   then
                       messageDlg('This File has some        problem',mtwarning,
[mbOK],0) ;
                  until (n2 <> ' ') and  (n2 <> #13)  and (n2 <> #10);
                  con2num(n1,n2,intnum);
                     txt:=inttostr(   intnum);
```

```
                writeln(fp2,txt);

        end;

         closefile(fp2);
         closefile(fp1);
end ;
end.
```