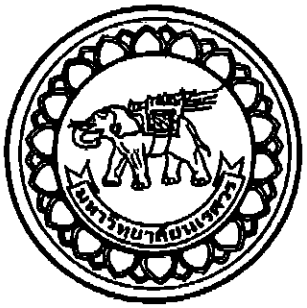


เครื่องวัดปริมาณน้ำฝนแบบเวลาจริง

REAL TIME RAINFALL METER

นาย รักษพล ศรีนุเสน รหัส 44362366

นาย ปัญญา สุธงษา รหัส 44362317

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2547

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน เครื่องวัดปริมาณน้ำฝนแบบเวลาจริง

ผู้ดำเนินโครงงาน นายรักษพล ศรีนุเสน รหัส 44362366

นายปัญญา สุธงษา รหัส 44362317

อาจารย์ที่ปรึกษา ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2547

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง)

..... กรรมการ

(ดร.อัครพันธ์ วงศ์กังแห)

..... กรรมการ

(อาจารย์พนัส นัถฤทธิ์)

หัวข้อโครงงาน	**เครื่องวัดปริมาณน้ำฝนแบบเวลาจริง**
ผู้ดำเนินโครงงาน	**นายรักษพล ศรีนุเสน รหัส 44362366**
	นายปัญญา สุธงษา รหัส 44362317
อาจารย์ที่ปรึกษา	**ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง**
สาขาวิชา	**วิศวกรรมไฟฟ้า**
ภาควิชา	**วิศวกรรมไฟฟ้าและคอมพิวเตอร์**
ปีการศึกษา	**2547**

บทคัดย่อ

ในปัจจุบันมีภัยพิบัติทางธรรมชาติเกิดขึ้นมากมาย ซึ่งอุกทกภัยก็เป็นที่**เกิดขึ้นบ่อยครั้ง** หาก**เรามีเครื่องมือที่สามารถเตือนภัยในส่วนนี้ได้** ก็จะสามารถบรรเทาความ**รุนแรงของเหตุการณ์ที่จะขึ้นได้**

โครงงานวิศวกรรมนี้ศึกษาและออกแบบเครื่องวัดปริมาณน้ำฝนควบคุม**โดยไมโครคอน-โทรลเลอร์** และสามารถเก็บข้อมูลที่ได้ เป็นสถิติเพื่ออ้างอิงแนวโน้มปริมาณน้ำ**ฝนในอนาคตได้**

Project title	Real time rainfall meter		
Name	Mr. Raksapon	Srinusan	ID. 44362366
	Mr. Panya	Suthongsa	ID. 44362317
Project advisor	Adsistance Professor Dr. Yongyut Chonbodeechalermroong		
Major	Electrical Engineering		
Department	Electrical and Computer Engineering		
Academic year	2004		

ABSTRACT

In the present day, many natural disasters have happened, which often are floods. If we have an instrument which can warn of the flood disaster, we can reduce the lost of that event.

This project is to study and manipulate a rain measurement instrument and a data logger, controlled and processed by a microcontroller MCS-51. The controller can display volume per minute of rain and can apply the stored data to be a reference or to analyze raining statistics.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากหลายๆ ท่าน ผู้จัดทำจึงถือโอกาสนี้ขอกราบขอบพระคุณ

ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง ซึ่งเป็นอาจารย์ที่ปรึกษา ซึ่งได้ให้คำปรึกษาชี้แนะแนวทางและข้อคิดเห็นต่างๆ ในการแก้ปัญหาที่เป็นประโยชน์อย่างสูงในการทำโครงงานนี้ให้สำเร็จลุล่วงด้วยดี

ขอบคุณอาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ทุกท่าน ที่ได้ประสิทธิ์ประสาทความรู้จนสามารถนำมาประยุกต์ใช้งานได้

ขอบคุณพี่ๆ เพื่อนๆ และน้องๆ นิสิตภาควิชาวิศวกรรมไฟฟ้าทุกคนที่ได้ให้ความช่วยเหลือในหลายๆ ด้าน ทั้งเรื่องส่วนตัวและเรื่องเรียนด้วยดีเสมอมา

ท้ายนี้ผู้จัดทำโครงงานขอบกราบขอบพระคุณบิดา มารดา และญาติพี่น้องของข้าพเจ้าที่เลี้ยงดูและคอยสนับสนุนด้านการเงิน รวมทั้งเป็นกำลังใจให้ผู้จัดทำเสมอมาจนสำเร็จการศึกษา

คณะผู้จัดทำโครงงาน
นายรักษพล ศรีนุเสน
นายปัญญา สุธงษา

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากประเทศไทยยังถือว่าเป็นประเทศเกษตรกรรม ซึ่งยังต้องพึ่งน้ำฝนในการทำเกษตรกรรม ดังนั้นหากเราสามารถทราบแนวโน้มของปริมาณน้ำฝนในแต่ละปีก็จะมีส่วนช่วยในการทำเกษตรกรรมเป็นไปได้อย่างมีความเหมาะสมและเกิดประโยชน์สูงสุด และในปัจจุบันมีภัยธรรมชาติหลากหลายประการ ซึ่งหนึ่งในนั้นก็คืออุทกภัยซึ่งทำให้เกิดความเสียหายทั้งชีวิตและทรัพย์สิน เป็นผลทำให้การดำรงชีวิตและการพัฒนาชุมชนเป็นไปอย่างยากลำบากดังนั้นเราจึงควรที่จะทราบแนวโน้มของปริมาณน้ำฝนในแต่ละช่วงว่ามีการเปลี่ยนแปลงอย่างไร เป็นการเตรียมความพร้อมในการที่จะรับมือกับภัยพิบัติที่อาจจะเกิดขึ้นอย่างไม่คาดคิด

ซึ่งหากว่าเราสามารถเก็บข้อมูลน้ำฝนในเวลาต่างๆ ได้นั้นก็จะเป็นการดีที่จะนำไปใช้ในการวิเคราะห์รวมกับปัจจัยอย่างอื่นในการพัฒนาชุมชนต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาหลักการทำงานของไมโครคอนโทรลเลอร์และอุปกรณ์อิเล็กทรอนิกส์
2. เพื่อศึกษาและพัฒนาการเก็บข้อมูลโดยใช้ไมโครคอนโทรลเลอร์
3. เพื่อศึกษาและพัฒนาโปรแกรมภาษาซีในการควบคุมไมโครคอนโทรลเลอร์

1.3 ขอบข่ายของงาน

1. พัฒนาซอฟต์แวร์เพื่อการเก็บข้อมูลปริมาณน้ำฝนโดยใช้ไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์อิเล็กทรอนิกส์เพื่อเก็บข้อมูลและส่งข้อมูลนั้นมายังคอมพิวเตอร์

2. ในการวิจัยครั้งนี้ถือเอาความสามารถในการเก็บข้อมูลน้ำฝนแบบตามเวลาจริง (real time)เป็นหลักไม่รวมถึงการนำข้อมูลไปวิเคราะห์และทำข้อมูลแบบสถิติ

1.4 กิจกรรมและขั้นตอนการดำเนินการ

ตารางที่ 1.1 ขั้นตอนการดำเนินการสร้างเครื่องวัดปริมาณน้ำฝน

กิจกรรม	ปี2547		ปี 2548									
	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1. ศึกษาและรวบรวมข้อมูลที่ เกี่ยวข้องกับการวัดปริมาณน้ำฝน												
2. ออกแบบสร้างเครื่องวัดปริมาณน้ำฝนโดยใช้ไมโครคอนโทรลเลอร์												
3. ค้นคว้าและศึกษาเกี่ยวกับการทำงานและโปรแกรมที่ใช้ควบคุมไมโครคอนโทรลเลอร์												
4. ออกแบบวงจรและต่อวงจร												
5.เขียนโปรแกรมควบคุมการทำงาน												
6. ทดสอบการทำงานและบันทึกผล												
7. จัดทำรายงาน												

1.5 ผลที่คาดว่าจะได้รับ

1. สามารถทราบและเก็บข้อมูลของปริมาณน้ำฝนแบบเวลาจริง(real time)ได้
2. ความรู้ด้านการออกแบบและการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์
3. เป็นเครื่องต้นแบบเพื่อทำการสร้างเครื่องวัดปริมาณน้ำฝนที่มีราคาถูกได้

1.6 งบประมาณ

1. อุปกรณ์ไมโครคอนโทรลเลอร์	1,000	บาท
2. ส่วนรับน้ำฝนแบบอลูมิเนียม	1,000	บาท
รวมเป็นเงินทั้งสิ้น	2,000	บาท

บทที่ 2
หลักการและทฤษฎีเบื้องต้น

2.1 หลักการคำนวณและวัดปริมาณน้ำฝน[5]

ในการวัดปริมาณน้ำฝนนั้นจะใช้วิธีวัดเป็นความสูงของระดับน้ำที่ตกลงมาแล้วไม่ซึมลงไปในดินมีหลักการคำนวณดังนี้

$$\text{ปริมาณน้ำฝน} = \frac{\text{ปริมาตรของน้ำฝน}}{\text{พื้นที่หน้าตัดของถังรับน้ำฝน}}$$

ตัวอย่างเช่นเก็บน้ำฝนได้ปริมาตร 100 ลูกบาศก์เซนติเมตรถังน้ำฝนมีพื้นที่หน้าตัด 200 ตารางเซนติเมตร

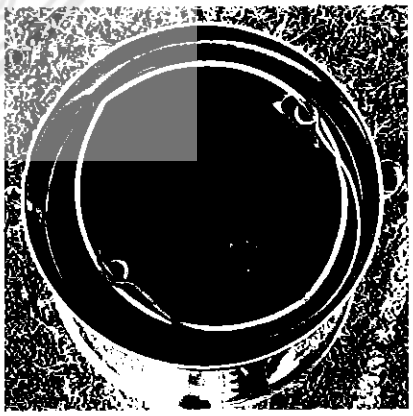
$$\text{จะมีปริมาณน้ำฝน} = \frac{100cm^3}{200cm^2} = 0.5 \text{ cm} = 5 \text{ mm}$$

2.2 เครื่องวัดปริมาณน้ำฝนที่มีในปัจจุบัน

2.2.1 เครื่องวัดปริมาณน้ำฝนแบบชั่งตวง[5]

จะใช้วิธีการเก็บน้ำไว้ในถังเก็บที่อยู่ด้านในแล้วนำน้ำที่ได้ในแต่ละวันมาทำการชั่งตวงในบิกเกอร์ที่ทำการคำนวณแล้วเป็นตัววัด

ก.



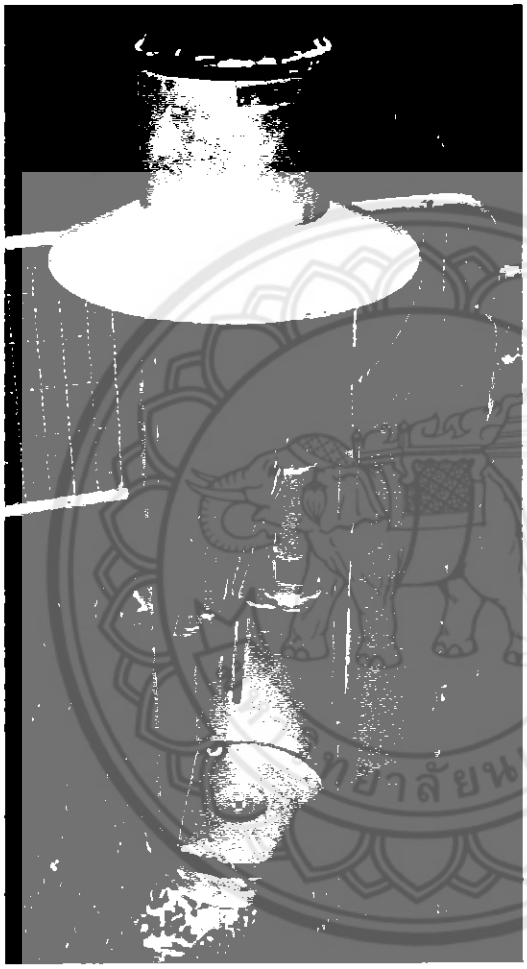
ข.

รูปที่ 2.1 ก.แสดงส่วนรับน้ำฝนและอุปกรณ์ชั่งตวง

รูปที่ 2.1 ข. แสดงอุปกรณ์เก็บน้ำฝน

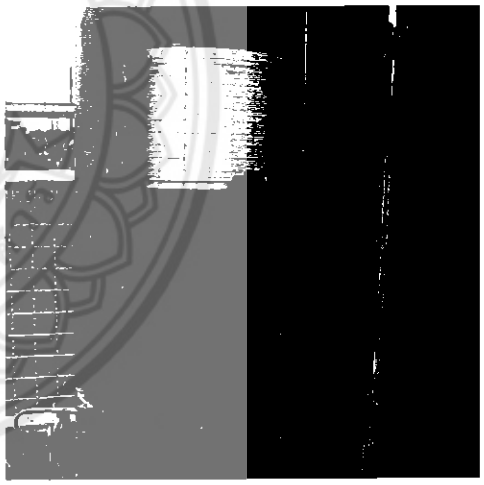
2.2.2 เครื่องวัดปริมาณน้ำฝนแบบกราฟ[5]

จะใช้วิธีการรับน้ำจากที่รองรับแล้วส่งลงไปเก็บในหลอดที่มีลูกลอยยึดติดอยู่กับปากกา การทำงานของเครื่องกราฟจะติดอยู่ที่แกนหมุน และจะหมุนไปเรื่อยๆจนครบรอบจะใช้เวลา 24 ชั่วโมงซึ่งในตอนที่กราฟหมุนอยู่นี้หากมีฝนตกลูกลอยก็จะสูงขึ้น ปากกาที่ทำการเขียนกราฟก็จะมีการยกขึ้นตามไปด้วยทำให้ทราบปริมาณน้ำฝนต่อเวลาได้



ก.

ข.

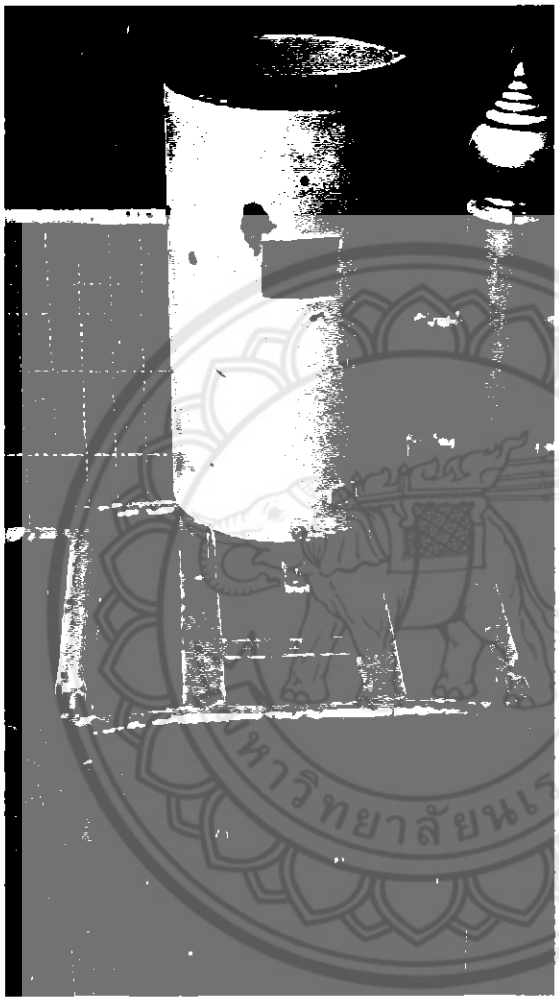


ค.

รูปที่ 2.2 ก.แสดงเครื่องวัดปริมาณน้ำฝนแบบกราฟโดยรวม
รูปที่ 2.2 ข.แสดงส่วนรับน้ำฝน
รูปที่ 2.2 ค.แสดงส่วนกราฟที่ใช้ในการบันทึก

2.2.3 เครื่องวัดปริมาณน้ำฝนแบบคานกระดกเก็บข้อมูลโดย DATA LOGGER[5]

จะใช้วิธีการรับน้ำฝนจากถังรับน้ำฝนแล้วส่งผ่านกรวยรับน้ำลงบนคานกระดก เมื่อมีการกระดกก็จะมีการส่งสัญญาณไปเก็บข้อมูลที่ DATA LOGGER ทำให้สามารถทราบปริมาณน้ำฝนต่อเวลาได้



ก.

ข.

ค.

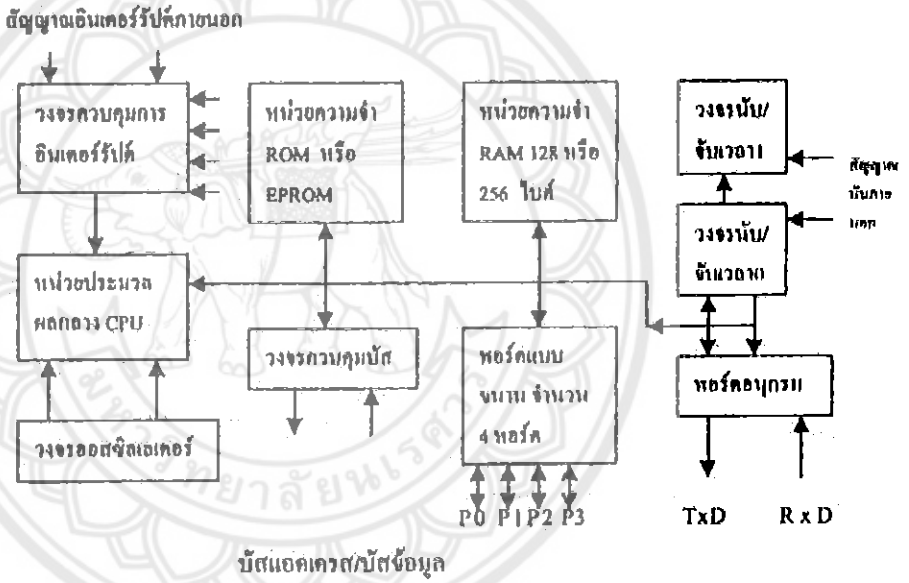
รูปที่ 2.3 ก. แสดงด้านนอกเครื่องวัดปริมาณน้ำฝนแบบคานกระดก

รูปที่ 2.3 ข. แสดงส่วนรับน้ำฝน

รูปที่ 2.3 ค. แสดงส่วนของคานกระดกและสวิทช์รับข้อมูล

2.3 ไมโครคอนโทรลเลอร์ MCS-51[4]

ไมโครคอมพิวเตอร์ มีส่วนประกอบหลักคือ ไมโครโพรเซสเซอร์ หน่วยความจำ และอินพุทเอาท์พุทไมโครโพรเซสเซอร์เรียกย่อๆได้ว่าเป็นหน่วยประมวลผลกลางหรือCPUไมโครคอนโทรลเลอร์หรืออดีตนิยมเรียกว่า ไมโครคอมพิวเตอร์ชิพเดี่ยว (Single Chip Microcomputer) ได้รวมเอาหน่วยความจำอินพุทเอาท์พุทพอร์ท และวงจรพิเศษ เช่นวงจรตั้งเวลาเข้าไว้บนชิพแผ่นเดียว ทำให้เราไม่ต้องต่ออุปกรณ์ภายนอก แม้กระทั่งวงจรกำเนิดสัญญาณนาฬิกา เพียงแค่ต่อไฟเลี้ยง +5V วงจรรีเซ็ท และต่อคริสตอลใช้กำเนิดความถี่ในการทำงานไมโครคอนโทรลเลอร์ก็สามารถทำงานได้แล้ว รูปที่ 2.4 เป็นไดอะแกรมวงจรภายในชิพ MCS-51 ภายในจะเห็นว่ามีCPU หรือไมโครโพรเซสเซอร์ต่อร่วมกับหน่วยความจำบนชิพ อินพุทเอาท์พุทพอร์ท พอร์ทอนุกรม และตัวตั้งเวลา เป็นต้น



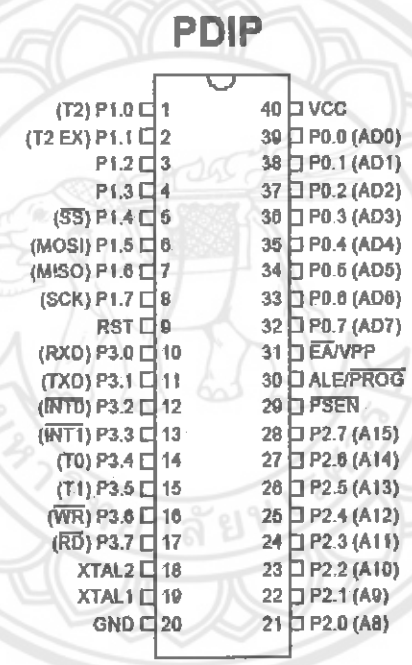
รูปที่ 2.4 แผนภาพบล็อกแสดงหน่วยทำงานพื้นฐานภายในชิพ MCS-51

2.3.1 คุณลักษณะพื้นฐานของ ชิพไมโครคอนโทรลเลอร์ AT 89C52

- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์
- มีหน่วยความจำโปรแกรมชนิดแฟลชขนาด 8 กิโลไบท์สามารถลบได้ด้วยสัญญาณไฟฟ้า
- หน่วยความจำข้อมูล 256 ไบท์
- พอร์ตอินพุตเอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถทำงานได้อย่างอิสระ

- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์ (Full Duplex)

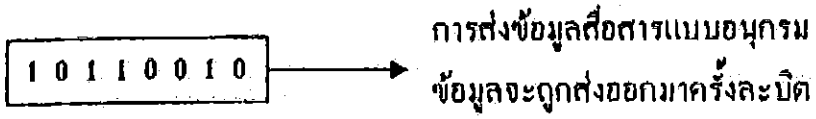
โดยมากแล้วไมโครคอนโทรลเลอร์ตระกูลนี้ มักจะมีรูปร่างของไอซีเป็นแบบ DIP ขนาด 40 ขาดังแผนภาพในรูปที่ 2.5 ซึ่งแต่ละขาสัญญาณจะมีหน้าที่ที่ระบุชัดเจนตามสัญลักษณ์ชื่อย่อที่กำกับในแต่ละขา อย่างไรก็ตามจะมีบางขาสัญญาณที่อาจจะมีหน้าที่ได้มากกว่าหนึ่งอย่างซึ่งจะไม่สามารถใช้งานในเวลาเดียวกันได้ ตัวอย่างเช่น ขาสัญญาณบิต 0 ของพอร์ต 3 (ใช้ตัวย่อ P3.0) อาจจะใช้เป็นขาสัญญาณเอาต์พุต/อินพุตตามปกติ หรืออาจทำหน้าที่เป็นขาสัญญาณอินพุตของข้อมูลสื่อสารแบบอนุกรม (RXD) ให้กับวงจรสื่อสารแบบอนุกรมของไมโครคอนโทรลเลอร์ได้ ซึ่งการกำหนดว่าจะทำงานในลักษณะใดก็ขึ้นอยู่กับการเชื่อมต่อวงจรเข้ากับขาสัญญาณ และโปรแกรมควบคุมของระบบนั้น



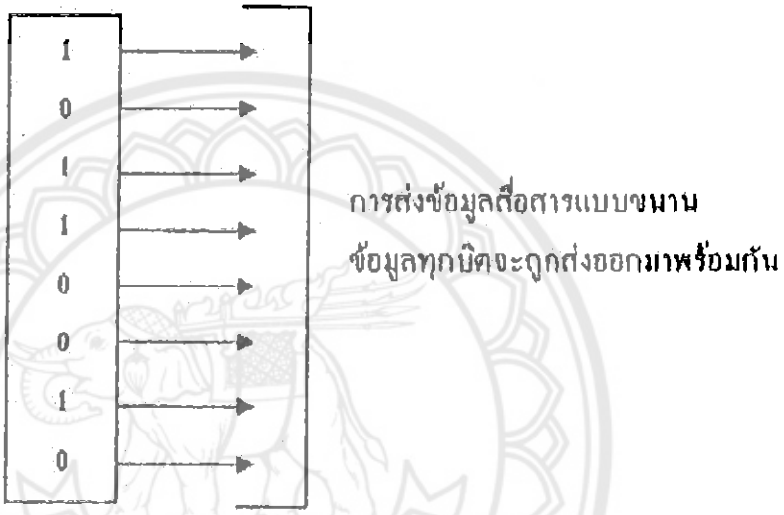
รูปที่ 2.5 แสดงขาต่างๆของไมโครคอนโทรลเลอร์ MCS-51

2.3.2 การสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมเป็นการรับส่งข้อมูลในลักษณะของบิตหรือกลุ่มของบิต คราวละหนึ่งบิตเป็นลำดับเรื่อยไปจนสิ้นสุด การสื่อสารแบบนี้จะมีข้อแตกต่างจากการสื่อสารแบบขนานเป็นอย่างมาก เนื่องจากข้อมูลมีการโอนย้ายมาพร้อมกันจึงมีความจำเป็นต้องใช้จำนวนเส้นสัญญาณมากขึ้นตามจำนวนบิตของข้อมูลด้วยในขณะที่การสื่อสารแบบอนุกรมนั้น ต้องการเส้นสัญญาณเพียงสองหรือสามเส้นเท่านั้น ดังนั้นการสื่อสารแบบขนานจึงไม่เหมาะสมในการสื่อสารกับอุปกรณ์ภายนอกเป็นระยะทางไกลๆเพราะจะทำให้สิ้นเปลืองค่าใช้จ่ายมาก ลองพิจารณาเปรียบเทียบการสื่อสารทั้งสองประเภทได้จากรูป 2.6 และ 2.7



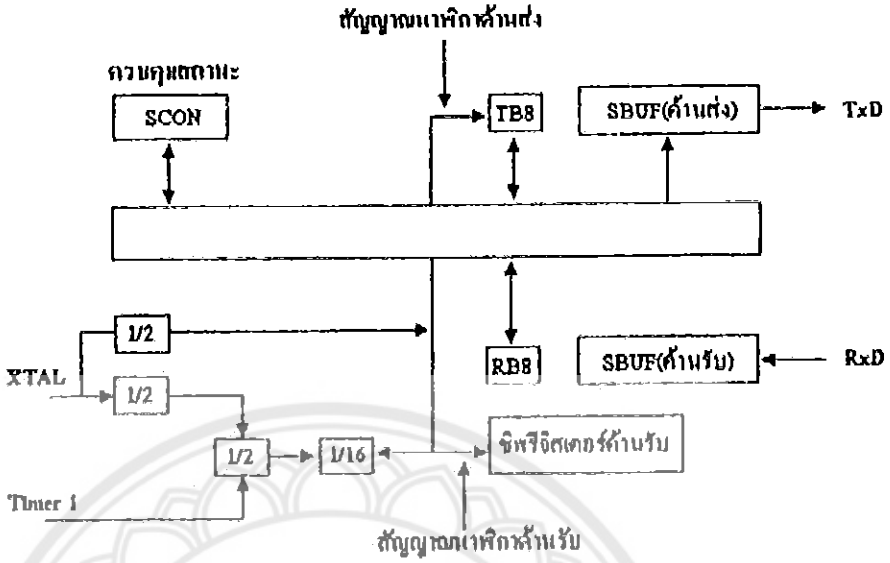
รูปที่ 2.6 แสดงข้อมูลการส่งข้อมูลแบบอนุกรม



รูปที่ 2.7 แสดงข้อมูลสื่อสารแบบขนาน

2.3.3 การจัดการข้อมูลอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

พอร์ตอนุกรมของ 8051 มีโครงสร้างการทำงานในแบบที่เรียกว่าฟูลดูเพล็กซ์ (Full Duplex) ซึ่งหมายถึงความสามารถในการรับและส่งข้อมูลอนุกรมในเวลาเดียวกัน จากรูปที่2.8 แสดงให้เห็นถึงแผนภาพการทำงานอย่างง่ายของวงจรส่วนจัดการข้อมูลอนุกรมของ MCS-51โดยทางด้านวงจรตัวส่ง ประกอบด้วยรีจีสเตอร์ SBUF ทำหน้าที่เก็บข้อมูลที่จะส่งออก การใช้คำสั่งเขียนหรือโอนย้ายข้อมูลมายังรีจีสเตอร์นี้ จะเป็นการส่งข้อมูลนั้นออกไปยังพอร์ตอนุกรมทางขาสัญญาณ TxD โดยอัตโนมัติ ส่วนวงจรด้านตัวรับ ประกอบด้วยรีจีสเตอร์ SBUF เช่นเดียวแต่ทำหน้าที่เก็บข้อมูลที่นำมาจากส่วนของวงจรเลื่อนบิต หรือชิพรีจีสเตอร์ของวงจรจัดการข้อมูลอนุกรมภายในสัญญาณข้อมูลอนุกรมที่รับเข้าจะผ่านมาทางขาสัญญาณ RxD



รูปที่2.8 แผนภาพแสดงการทำงานของวงจรส่วนการรับและส่งข้อมูลอนุกรมของ 8051

พอร์ตอนุกรมของ MCS-51 สามารถโปรแกรมให้ทำหน้าที่ในรูปแบบต่างๆกัน 4 แบบ โดยการกำหนดค่าบิต SM0 และ SM1 ซึ่งอยู่ภายในรีจิสเตอร์ควบคุมและบอกสถานะ SCON ดังตารางที่ 2.2 โหมดการทำงานทั้ง 4 แบบของพอร์ตอนุกรมมีดังนี้

ตารางที่ 2.1 โหมดการทำงานของพอร์ตอนุกรม

โหมดการทำงาน	คำอธิบาย
โหมด0	เป็นการขยายพอร์ตอินพุตเอาต์พุต โดยร่วมกับไอซีชิพรีจีสเตอร์ภายนอก ประเภททีทีแอลหรือซีมอส
โหมด1	ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART (Universal asychronous reciver / transmitter) โดยการใช้กลุ่มข้อมูลแบบ 10 บิต และสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้
โหมด2	ใช้สำหรับเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิตและกำหนดอัตราเร็วในการส่งข้อมูลคงที่
โหมด3	ใช้สำหรับเชื่อต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิตและสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้

นอกจากนี้โหมด 2 และ 3 ยังมีการดำเนินการแบบพิเศษออกไป โดยสามารถนำมาใช้ประโยชน์ในการสื่อสารข้อมูลแบบที่มีไมโครโปรเซสเซอร์หลายตัวทำงานร่วมกันได้ ซึ่งจะได้อธิบายรายละเอียดเป็นลำดับไป

ตารางที่ 2.2 รีจีสเตอร์ควบคุมการทำงานและบอกสถานะการสื่อสารข้อมูลอนุกรม SCON

ชื่อบิต	ตำแหน่ง	ความหมาย
SM0	SCON.7	บิตเลือกโหมดการทำงาน
SM1	SCON.6	บิตเลือกโหมดการทำงาน
SM2	SCON.5	แฟล็กกำหนดการทำงานแบบมัลติโปรเซสเซอร์
REN	SCON.4	แฟล็กยอมให้มีการรับข้อมูล
TB8	SCON.3	ค่าของบิตที่9สำหรับการส่งข้อมูลออก
RB8	SCON.2	ค่าของบิตที่9ของข้อมูลที่รับเข้า
TI	SCON.1	แฟล็กแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูล
RI	SCON.0	แฟล็กแสดงการอินเตอร์รัปต์เมื่อมีข้อมูลรับเข้า

จากแผนภาพตารางที่ 2.2 ชิพรีจีสเตอร์ภายในตัวส่งจะทำหน้าที่เลื่อนบิตข้อมูลออกไปภายนอกโดยไม่มีการบัพเฟอร์ และเมื่อใดที่มีการเขียนข้อมูลให้กับรีจีสเตอร์ SBUF แสดงว่ามีความต้องการที่จะส่งข้อมูลนี้ออกไปแบบอนุกรม สำหรับชิพรีจีสเตอร์ทางด้านรับจะทำการเลื่อนบิตข้อมูลที่รับเข้ามาเก็บไว้ เมื่อบิตของข้อมูลที่รับเข้ามาครบถ้วนตามจำนวนที่กำหนดไว้ตามลักษณะโหมดการทำงานต่างๆแล้ว จะถูกย้ายไปเก็บยังรีจิสเตอร์ SBUF ต่อไป อย่างไรก็ตามการย้ายข้อมูลนี้จะเกิดขึ้นก็ต่อเมื่อรีจีสเตอร์ SBUF นั้นไม่มีข้อมูลที่จะทำการส่งหรือได้ส่งข้อมูลออกไปเสร็จสิ้นแล้ว

2.3.4 การอินเตอร์รัพต์ของการสื่อสารอนุกรม[2]

เนื่องจากการส่งหรือรับข้อมูลอนุกรมในการส่งข้อมูลไบต์หนึ่งๆ ค่อนข้างจะใช้เวลานานหลายมิลลิวินาที ดังนั้นเพื่อการจัดการเกี่ยวกับการสื่อสารแบบนี้เป็นไปอย่างมีประสิทธิภาพ MCS-51 จึงได้กำหนดให้บิตหรือแฟล็กสถานะที่เกี่ยวข้องทั้งหมด จัดรวมอยู่ภายในรีจีสเตอร์ SCON เท่านั้นเช่นแฟล็ก TI ซึ่งมีค่าเป็นหนึ่ง เพื่อแจ้งให้ทราบว่าได้รับข้อมูลผ่านเข้าทางพอร์ตอนุกรม เมื่อแฟล็กตัวใดตัวหนึ่งนี้มีค่าเป็นหนึ่ง จะมีผลทำให้เกิดการอินเตอร์รัพต์ขึ้น ดังนั้นภายในโปรแกรมจะต้องทำการตรวจสอบจากสถาวะของแฟล็กเหล่านี้เอง ว่ามีการอินเตอร์รัพต์ขึ้นด้วยสาเหตุใด จากนั้นจึงค่อยทำการกำหนดค่าศูนย์ให้กับแฟล็กนั้น ลักษณะดังกล่าวนี้จะมีความแตกต่าง

ไปจากการอินเตอร์รัพต์จากสัญญาณอื่นๆเช่น วงจรนับ วงจรจับเวลา เป็นต้น ซึ่งจะมีการกำหนดค่าศูนย์ ให้กับแฟล็กสถานะที่เกี่ยวข้องโดยอัตโนมัติ ภายหลังจากที่ได้เข้าไปทำงานยังส่วนของโปรแกรมย่อยบริการอินเตอร์รัปต์ ดังนั้นจึงขอให้สังเกตความแตกต่างในส่วนนี้ด้วย

2.3.5 กระบวนการรับและส่งข้อมูลอนุกรมของ MCS-51[2]

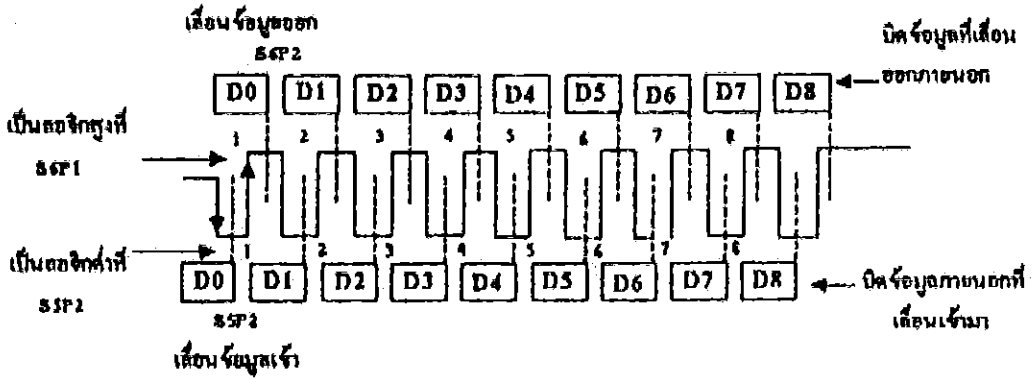
การส่งข้อมูลออกทางพอร์ตอนุกรมของ MCS-51 จะเริ่มต้นขึ้น ภายหลังเมื่อมีการเขียนข้อมูลลงในรีจิสเตอร์ SBUF ข้อมูลนี้จะถูกจัดการด้วยวิธีการทางด้านฮาร์ดแวร์ในการเลื่อนบิตและส่งสัญญาณออกไปภายนอกโดยอัตโนมัติ เมื่อข้อมูลเหล่านี้ได้ส่งออกครบถ้วนแล้วจึงทำการกำหนดค่าของแฟล็ก TI ให้เป็นหนึ่งเพื่อแจ้งให้ทราบว่า ขณะนี้รีจิสเตอร์ SBUF ว่าง และพร้อมที่จะส่งข้อมูลไบต์ต่อไปแล้ว ในกรณีที่ผู้ใช้เขียนข้อมูลใหม่ลงในรีจิสเตอร์ SBUF โดยไม่ต้องรอให้แฟล็ก TI มีค่าเป็นหนึ่งก่อนจะมีผลทำให้ข้อมูลที่ส่งออกไปผิดพลาดได้

สำหรับการรับข้อมูลจากพอร์ตอนุกรมจะต้องเริ่มต้นโดยการกำหนดค่าบิREN(Reciever Enable) ให้มีค่าเป็นหนึ่งก่อน หลังจากนั้นเมื่อมีบิตของข้อมูลถูกส่งเข้ามาจากภายนอกระบบฮาร์ดแวร์ของ MCS-51 จึงจะทำการเลื่อนบิตเหล่านี้เข้ามาโดยอัตโนมัติ และเมื่อบิตสุดท้ายถูกเลื่อนเข้าเรียบร้อยแล้ว ข้อมูลเหล่านั้นจะถูกย้ายมาเก็บยังรีจิสเตอร์ SBUF และทำการกำหนดให้แฟล็ก RI ให้มีค่าเป็นหนึ่ง ซึ่งมีผลทำให้เกิดการอินเตอร์รัปต์โปรแกรมขึ้น

2.3.6 พอร์ตอนุกรมโหมด 0 [4]

การทำงานของพอร์ตอนุกรมในโหมดศูนย์ เป็นการขยายพอร์ตอินพุตหรือพอร์ตเอาต์พุตให้มีจำนวนมากขึ้น โดยจะทำการสร้างสัญญาณนาฬิกาขึ้นเพื่อให้จังหวะของการทำงานที่พร้อมกันสำหรับการเลื่อนบิตเข้าหรือออกจากไอซีรีจิสเตอร์ภายนอก เมื่อมีการโอนย้ายข้อมูลมายังรีจิสเตอร์ในแต่ละครั้งจะมีผลทำให้เกิดการส่งบิตข้อมูลทั้ง 8 บิตออกมา แม้ว่าแฟล็ก สถานะ TI ยังคงมีค่าเป็นหนึ่งอยู่ก็ตาม นอกจากนี้แล้วเมื่อใดก็ตามค่าของแฟล็กสถานะ RI เป็นค่าหนึ่งก็ควรจะย้ายข้อมูลที่รับเข้ามานั้นออกไปจากรีจิสเตอร์ SBUF เสียก่อนที่จะได้มีการกำหนดค่าแฟล็ก RI ให้เป็นศูนย์เพื่อรับข้อมูลต่อไป

การทำงานของพอร์ตอนุกรมในโหมดศูนย์ เป็นการรับและส่งข้อมูลอนุกรมจำนวน 8 บิต โดยเพียงขาสัญญาณ RxD เท่านั้น ส่วนขาสัญญาณ TxD จะนำไปใช้เพื่อเป็นขาสัญญาณนาฬิกาในการให้จังหวะการเลื่อนข้อมูลกับวงจรเลื่อนบิตภายนอก สำหรับอัตราการเลื่อนบิต จะถูกกำหนดไว้ที่ค่าคงที่ที่ค่า1/2 ของความถี่ออสซิลเลเตอร์ จากรูปที่ 2.9 แสดงให้เห็นถึงแผนภาพเวลาสัญญาณต่างๆ ในโหมดศูนย์ เมื่อมีการรับและส่งข้อมูล 1 ไบต์ โดยสัญญาณนาฬิกา โดยสัญญาณนาฬิกาในการเลื่อนบิตนี้จะเกิดขึ้นในตัวของ MCS-51 เอง และมีจุดประสงค์เพื่อนำไปใช้สำหรับวงจรชิพรีจีสเตอร์ภายนอกเท่านั้น



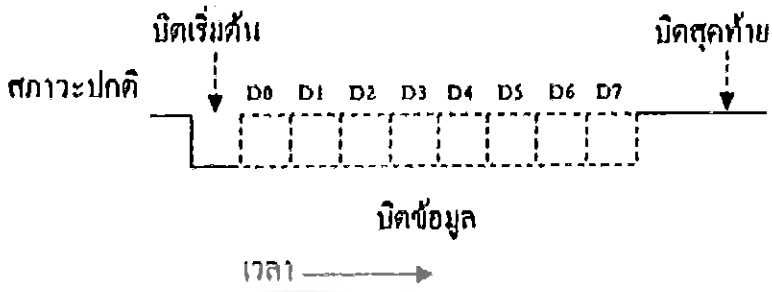
รูปที่2.9 แผนภาพเวลาของสัญญาณอนุกรมโหมดศูนย์

สัญญาณนาฬิกาที่สร้างขึ้นทางขาสัญญาณ TxD นี้จะสลับค่าไปมาจากลอจิกสูงไปต่ำในราวช่วงใกล้เคียงกับเวลาขอบขาลงของสัญญาณ ALE ซึ่งอยู่ในคาบเวลาออสซิลเลเตอร์ที่ 15 ภายหลังจากที่ได้ทำการโอนสั่งย้ายข้อมูลยังรีจิสเตอร์ SBUF หรือคำสั่งที่ทำให้ค่าแฟล็กสถานะ RI เป็นศูนย์ หลังจากนั้นสัญญาณนาฬิกาก็จะเปลี่ยนแปลงอีกครั้งราวช่วงใกล้เคียงกับเวลาขอบขาลงของสัญญาณ ALE ในคาบเวลาออสซิลเลเตอร์หลังจากนั้นอีก 6 คาบ แล้วจะดำเนินไปในลักษณะเช่นนี้จนกระทั่งข้อมูลทั้ง 8 บิต ได้ส่งออกไปเรียบร้อยแล้ว เมื่อสัญญาณขอบขาขึ้นของสัญญาณนาฬิกานี้เกิดขึ้นครบจำนวน 8 ครั้งแล้ว จึงจะมีผลทำให้แฟล็กสถานะ TI หรือ RI มีค่าเป็น 1 ขึ้น และสถานะของสัญญาณ TXD ก็จะเป็นระดับลอจิกสูงไปตลอด

ข้อมูลที่จะถูกส่งออกไปภายนอกจะถูกเลื่อนบิตนัยสำคัญต่ำออกไปก่อนเป็นลำดับแรก โดยจะเริ่มขึ้นในเวลาเริ่มต้นของคาบเวลาออกซิลเลเตอร์ ภายหลังจากที่ได้ทำคำสั่งการโอนย้ายข้อมูลมายังรีจิสเตอร์ SBUF สำหรับบิตแรกของข้อมูลที่รับมานั้นจะถูกแลตช์ไว้ด้วยขอบขาขึ้นของสัญญาณนาฬิกาในคาบเวลาออสซิลเลเตอร์ที่ 24 ภายหลังจากที่ได้มีการกำหนดให้แฟล็กสถานะ RI เป็นค่าศูนย์ หลังจากนั้นในคาบเวลาออสซิลเลเตอร์อีก 12 คาบ ถัดมาก็จะได้รับบิตต่อไป ซึ่งจะดำเนินการในลักษณะเช่นนี้จนกระทั่ง ได้จำนวนบิตข้อมูลครบทั้ง 8 บิต

2.3.7 พอร์ตอนุกรมโหมด 1

การทำงานในโหมด 1 เป็นการสื่อสารข้อมูลอนุกรมจำนวน 10 บิต ประกอบด้วยบิตเริ่มต้นจำนวน 1 บิต บิตข้อมูลจำนวน 8 บิต และบิตสุดท้ายอีก 1 บิต ดังแสดงในรูป 2.10



รูปที่2.10 รูปแบบของสัญญาณอนุกรมในโหมด 1 ใช้ข้อมูล 8 บิต

โดยข้อมูลจะถูกส่งออกไปทางขาสัญญาณ TXD และรับเข้ามาทางขาสัญญาณ RXD ในส่วนของข้อมูล 8 บิต ที่ได้รับหรือทำการส่งออกจะเป็นบิตนัยสำคัญต่ำเป็นลำดับแรก และบิตสุดท้ายของข้อมูลที่รับเข้ามาจัดเก็บไว้ในบิต RB8 ภายในรีจิสเตอร์ SCON สำหรับอัตราความเร็วในการส่งข้อมูลของโหมด 1 นั้น สามารถกำหนดเลือกได้

โหมดการทำงานนี้สามารถใช้ในการติดต่อกับพอร์ตสื่อสารอนุกรมแบบ RS-232C ของเครื่องคอมพิวเตอร์ทั่วไปได้ ดังนั้นในกรณีที่ใช้เป็นข้อมูล 7 บิตและไม่ใช้บิตที่ 8 เป็นบิตพาริตี้ก็อาจจะกำหนดค่าบิตนี้ให้เป็น 1 ซึ่งจะทำให้ด้านรับมองบิตนี้เป็นบิตสุดท้ายไป สำหรับกรณีที่ MCS-51 เป็นฝ่ายรับข้อมูลของระบบนี้ซึ่งมีเพียง 7 บิต ก็จะมองค่าของบิตสุดท้ายของข้อมูลที่รับมานี้เป็นค่าของข้อมูลบิตที่ 8 แทนและยังคงรอรับบิตสุดท้ายต่อไป อย่างไรก็ตามเนื่องว่าระดับสัญญาณ ของบิตสุดท้ายนี้จะเป็นระดับลอจิกสูงเช่นเดียวกับสภาวะของสายสื่อสาร เมื่อไม่มีการส่งข้อมูล ดังนั้นระบบก็จะอ่านค่านี้เข้า ซึ่งก็ยังคงถือว่าถูกต้องตามหลักการโดยปริยาย

การส่งข้อมูลจะเกิดขึ้นภายหลังจากที่ได้มีการเขียน หรือโอนย้ายข้อมูลเข้าไปยังรีจิสเตอร์ SBUF โดยผู้เขียนโปรแกรมจะต้องทำการตรวจสอบค่าของแฟล็กสถานะ TI ภายในรีจิสเตอร์ SCON ซึ่งจะมีค่าเป็น 1 ภายหลังที่ข้อมูลได้เลื่อนบิตออกไปภายนอกแล้ว สำหรับการรับข้อมูลจะเริ่มต้นขึ้นเมื่อได้มีการกำหนดค่า 1 ให้กับบิต REN และมีการเปลี่ยนแปลงระดับสัญญาณที่ขาสัญญาณ RxD เกิดขึ้นการสุ่มอ่านค่าบิตข้อมูลเข้ามาจะใช้อัตราเดียวกับอัตราบอดที่ได้กำหนดไว้ในราวช่วงกลางคาบเวลาของบิตหลังจากที่ได้รับข้อมูลครบจำนวน 10 บิตแล้ว และหากมีสภาวะดังในตารางต่อไปนี้เกิดขึ้นก็จะมีผลให้เกิดการย้ายข้อมูลไปเก็บยังรีจิสเตอร์ SBUF

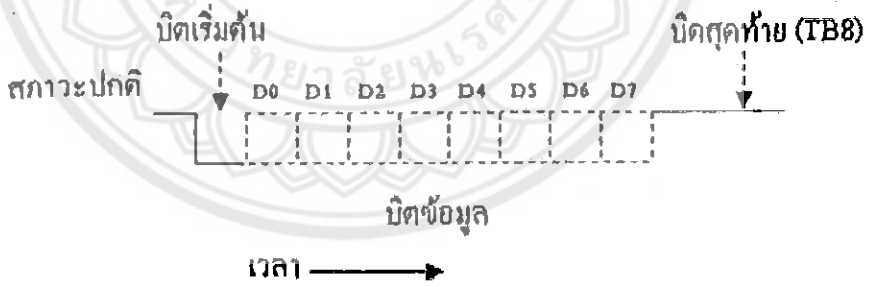
ตารางที่ 2.3 แฟล็กสถานะของรีจีสเตอร์ในการรับข้อมูล

1	แฟล็ก RI มีค่าเป็น 0 (แสดงว่าได้มีการอ่านไบต์ของข้อมูลเข้ามาแล้ว และพร้อมที่จะรับข้อมูลถัดไป)
2	ค่าของบิตสุดท้ายเป็น 1 (แสดงว่าข้อมูลที่ได้รับข้อมูลนั้นถูกต้อง จึงได้โอนย้ายไปเก็บยังรีจีสเตอร์ SBUF โดยไม่สนใจค่าของบิต SM2)

ดังนั้นสรุปได้ว่าข้อมูลที่รับเข้ามาจำนวน 10 บิตนั้น ส่วนของบิตเริ่มต้นไม่มีการนำไปใช้งานอีกต่อไป บิตข้อมูลจำนวน 8 บิตนั้นจะถูกย้ายไปเก็บยังรีจีสเตอร์ SBUF และส่วนของบิตสุดท้ายจะถูกนำไปเก็บในตำแหน่งของบิต RB8 ภายในรีจีสเตอร์ SCON นอกจากนี้ยังมีแฟล็กสถานะ RI ซึ่งจะเป็นค่าหนึ่ง เพื่อบอกสภาวะว่าได้มีการรับข้อมูลใหม่เข้ามาแล้ว ในกรณีที่โปรแกรมได้อ่านข้อมูลจากรีจีสเตอร์ SBUF แล้วแต่ไม่ได้กำหนดบิต RI ให้เป็นค่าศูนย์อีกครั้ง ข้อมูลที่รับเข้ามาใหม่หลังจากนั้นสูญหายไป

2.3.8 พอร์ตอนุกรมโหมด 2 และ 3

การทำงานโหมด 2 หรือโหมด3 ของพอร์ตอนุกรมจะทำการรับส่งข้อมูล 11 บิต เช่นเดียวกัน ซึ่งประกอบด้วย บิตเริ่มต้น บิตข้อมูลจำนวน 8 บิต บิตข้อมูลบิตที่9 และบิตสุดท้าย ดังแสดงในรูปที่ 2.11 แต่สำหรับโหมด 3 จะสามารถเปลี่ยนแปลงอัตราการส่งข้อมูลได้ไม่ได้ถูกกำหนดไว้คงที่เช่นในโหมด 2



รูปที่ 2.11 รูปแบบของสัญญาณข้อมูลอนุกรมในโหมด 2 ส่ง ข้อมูลจำนวน 9 บิต

การส่งข้อมูลอนุกรมในโหมด 2 และ 3 จะต้องนำค่าข้อมูลนั้นไปเก็บยังรีจิสเตอร์ SBUF สำหรับค่าของบิตที่9 ที่เพิ่มขึ้นนั้นนำมาจากค่าของบิต TB8 ภายในรีจีสเตอร์ SCON ซึ่งจะต้องได้รับการกำหนดค่าจากผู้ใช้งาน เมื่อข้อมูลถูกเลื่อนบิตออกไปภายนอกเรียบร้อยแล้ว แฟล็กสถานะ TI จะมีค่าเป็น 1 เช่นเดียวกับโหมดอื่น ๆที่ผ่านมา และผู้ใช้จะต้องทำการเปลี่ยนกลับให้เป็นค่า 0 ตามเดิม

สำหรับการรับข้อมูลจะถูกนำมาเก็บไว้ภายในรีจีสเตอร์ SBUF เช่นเดียวกันโดยค่าของบิตที่ 9 จะนำไปเก็บไว้ยังบิต RB8 ภายในรีจีสเตอร์ SCON

2.3.9 อัตราบ๊อด[4]

อัตราบ๊อดในโหมด 0

อัตราบ๊อดในโหมด 0 ของการใช้พอร์ตอนุกรมจะคงที่ที่ความถี่ออสซิลเลเตอร์คือ

$$\text{อัตราบ๊อดในโหมด 0} = \frac{\text{Oscillator Frequency}}{12}$$

อัตราบ๊อดในโหมด 2

อัตราบ๊อดในโหมด 2 จะขึ้นอยู่กับการปรับค่าบิตใน SMOD ของ SFR ในรีจิสเตอร์ PCON ถ้า SMOD = 0 ซึ่งจะเป็นค่าที่ถูกรีเซตแต่แรก หลังการรีเซต อัตราบ๊อดจะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ ถ้า SMOD เป็น 1 อัตราบ๊อดจะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ มีสูตรเป็น

$$\text{อัตราบ๊อดในโหมด 2} = \frac{2^{SMOD}}{64} \text{ x } OscillatorFrequency$$

อัตราบ๊อดในโหมด 1, 3

เมื่อใช้ตัวจับเวลา 1 เป็นตัวสร้างอัตราบ๊อด อัตราบ๊อดในโหมด 1และ3 จะถูกคำนวณด้วยอัตรา Overflow ที่เกิดขึ้นในตัวจับเวลา 1 และค่าบิตใน SMOD ซึ่งมีสูตรการคำนวณดังนี้

$$\text{อัตราบ๊อดในโหมด 1,3} = \frac{2^{SMOD}}{32} \text{ x Timer1 Overflow Rate}$$

การอินเตอร์รัพต์ในตัวจับเวลา1 ควรที่จะดิสเอเบิ้ลในการใช้งานแบบนี้ตัวจับเวลาในตัวมันเองสามารถที่จะถูกกำหนดให้ใช้เป็นตัวจับเวลาหรือตัวนับในการทำงานในโหมด 3 ในการใช้งานในลักษณะนี้มันจะถูกกำหนดให้ทำงานเป็นตัวจับเวลาในโหมดแบบบรรจุอัตโนมัติ(โดยตั้งให้ HIGH NIBBLE ของ TMOD = 0001B) ในกรณีนี้ อัตราบ๊อดคำนวณได้ดังสูตร

$$\text{อัตราบ๊อดในโหมด 1,3} = \frac{2^{SMOD}}{32} \text{ x } \frac{OscillatorFrequency}{12x(256 - TH1)}$$

ตารางที่ 2.4 ตารางแสดงอัตราบอดมาตรฐานที่นิยมใช้กันในการสื่อสารอนุกรม โดยการกำหนดค่าตัวแปรต่างๆสำหรับการเกิดอัตราบอดโดยใช้ Timer 1

อัตราบอด (โหมด1 และ 3)	ความถี่ ออสซิลเลเตอร์(MHz)	บิต SMOD	C/T	MODE	Reload Value
65.5K	12	1	0	2	FDh
19.2K	11.059	1	0	2	FDh
9600	11.059	0	0	2	FDh
4800	11.059	0	0	2	Fah
2400	11.059	0	0	2	F4h
1200	11.059	0	0	2	E8h
1375	11.059	0	0	2	1Dh
110	6	0	0	2	72h
110	12	0	0	1	FEEBh

2.4 ระบบบัส I^2C และอุปกรณ์ที่เกี่ยวข้อง[1]

I^2C ย่อมาจาก Inter-IC Communication หมายถึงการติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นจากบริษัทฟิลิปส์(Philips)ด้วยจุดมุ่งหมายหลักคือต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายสายสัญญาณนาฬิกาและสายข้อมูลของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางว่าสายข้อมูลอนุกรมหรือ SDA(Serial Data line)ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock line)

2.4.1 คุณสมบัติโดยทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณนาฬิกา 2 ทิศทาง(bi-directional line) ต้องมารต่อตัวต้านทานพูลอัปกับสัญญาณ +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่กับบัส I^2C ต้องมีลักษณะวงจรเดรนเปิด (open drain) หรือคอลเลกเตอร์เปิด(open collector)

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติและสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลในการเข้าถึง 2 ค่าคือ 7 บิต (7- bit addressing)หรือ(10-bit addressing)

2.4.2 หลักการของบัส I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ เส้น คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณีใดติดต่อกันอยู่และอุปกรณ์ใดเป็นตัวรับตัวส่ง

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า **ตัวส่ง (transmitter)**

อุปกรณ์ที่เป็นผู้รับข้อมูลหรือส่งข้อมูล เรียกว่า ตัวรับ (receiver) อุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่าง จะไม่อุปกรณ์ใดบนบัส I^2C ที่ทำหน้าเป็นตัวส่งเพียงเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ **(master)**

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า **สเลฟ (slave)**

ข้อกำหนด 2 ประการที่สำคัญของการติดต่อบนบัส I^2C คือ

1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น
2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามเมื่อสาย SCL **มีสถานะลอจิกสูง** สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้มีการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายว่าเป็นสัญญาณควบคุมแทน

2.4.3 สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สภาวะดังนี้

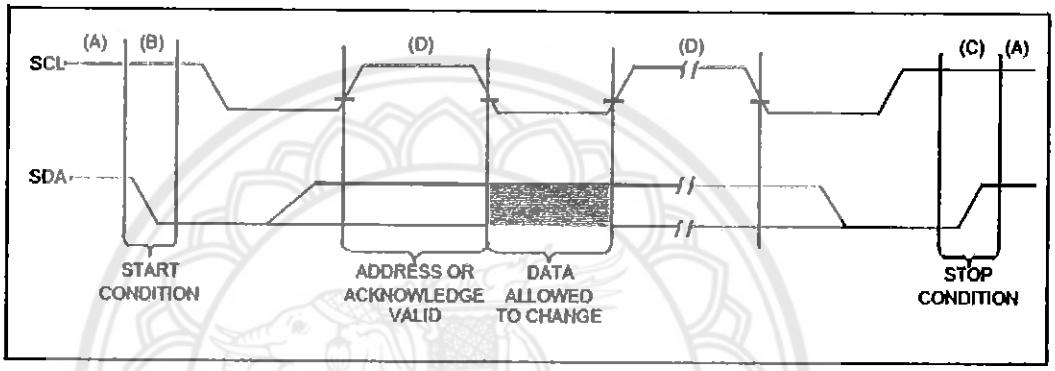
1. บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และSCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่าการถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2. เริ่มต้นการถ่ายทอดข้อมูล (start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงต่ำในขณะที่สาย SCL มีสถานะลอจิกสูง

3. ข้อมูลที่ดำรงอยู่บนบัส (data valid) สภาวะนี้เกิดขึ้นต่อจากสภาวะเริ่มต้นโดยสถานะลอจิกที่เกดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่ SDA ต้องมีค่าคงที่เพื่อให้อุปกรณ์รับรู้ข้อมูลในขณะนั้นว่าเป็น "0" หรือ "1" ข้อมูลอาจมีการเปลี่ยนแปลงได้เมื่อสาย SCL เป็นลอจิกต่ำแต่เมื่อใดก็ตามที่ต้องการให้มีการถ่ายทอดข้อมูลอย่างสมบูรณ์สถานะลอจิกที่ SDA ต้องคงที่ตลอดช่วงเวลาที่ที่สาย SCL เป็นลอจิกสูงหากเกิดการเปลี่ยนแปลงขณะที่ SCL เป็นลอจิกสูงอยู่นั้นอุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือเริ่มต้นก็ได้ทำให้ข้อมูลนั้นมีการผิดพลาดได้

4. รับรู้ข้อมูล (acknowledge) เกิดขึ้นหลังจากการถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์โดยตัวส่งจะส่งข้อมูลมา 1 บิต เรียกว่าบิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้ที่สถานะลอจิกต่ำเพื่อตอบสนองให้ทราบว่าได้รับข้อมูลเรียบร้อยแล้ว

5.หยุดการถ่ายทอดข้อมูล(stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)



รูปที่2.12 ไดอะแกรมเวลาแสดงสถานะต่างๆบนระบบบัส I^2C

2.4.4 การทำงานบนบัส I^2C

เริ่มต้นด้วยการเข้าถึงอุปกรณ์เสียก่อน โดยการเข้าถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การเข้าถึงแบบ 7 หรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การเข้าถึงแบบ 7 บิตก็เพียงพอ แต่ในบางอุปกรณ์ต้องใช้การเข้าถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

2.4.5 การเข้าถึงแบบ 7 บิต(7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ โดยมีรูปแบบแสดงในรูปที่ A3-3 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็นบิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่เขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือการเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น "O" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม(control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง

2.4.6 การเข้าถึงแบบ 10 บิต

จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น11110 ส่วนอีก 2 บิตถัดมาเป็นบิดแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องกี่ติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกับการเข้าถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

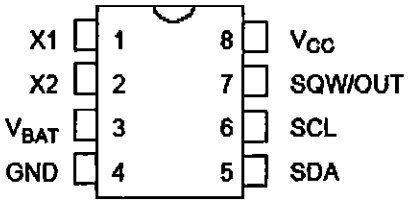
2.5 DS1307 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC)

- เป็นไอซีไทม์คล็อกทำงานบนบัส I^2C ให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรทินด้วย สามารถให้ข้อมูลเวลาๆได้อย่างเที่ยงตรงถึงปีคริสตศักราช 2100

- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายใน สามารถใช้เก็บข้อมูลทั่วไปได้

- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาไว้แม้ไม่มีไฟเลี้ยงไอซี

2.5.1 รายละเอียดขาต่อใช้งานของ DS1307



รูปที่2.13 การจัดขา DS1307

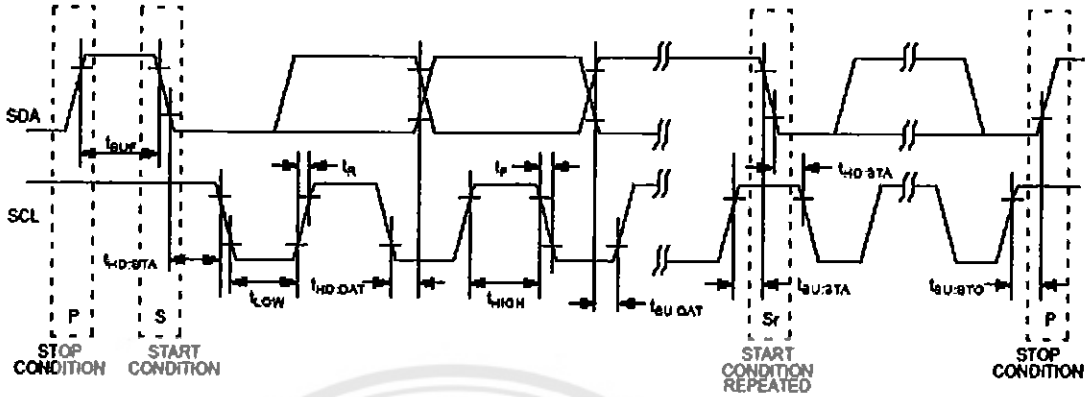
V_{CC} (ขา 8) ต่อกับไฟเลี้ยง + 5V

GND(ขา 4) ต่อกับกราวด์หรือไฟลบ

V_{BAT} (ขา 3)ใช้ต่อกับแบตเตอรี่ 3V

SDA,SCL (ขา 5และ6) เป็นขาสำหรับเชื่อมต่อระบบบัส I^2C เลือกความถี่ได้ 1Hz,4.096 kHz, 8.192 kHz และ 32 kHz ใน X1,X2 (ขา 1 และ 2) ต่อกับคริสตอลความถี่ 32.768 kHz

TIMING DIAGRAM



รูปที่ 2.14 ไดอะแกรมเวลาของ DS1307 บนระบบบัส I^2C

ไอซี DS1307 จัดการเชื่อมต่อในแบบบัส I^2C โดยทำงานเป็นอุปกรณ์สเลฟเสมอ ส่วนประกอบหลักที่สำคัญคือ วงจรออสซิลเลเตอร์ เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง มีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทล์แรม 64 ไบต์ ซึ่งจัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเก็บข้อมูลทั่วไป 56 ไบต์ วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากต่ำกว่า 1.25 x V_{BAT} ก็จะควบคุมให้ DS1307 หยุดทำงาน ทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งานต้องระมัดระวังอย่าให้ไฟเลี้ยงต่ำกว่า 1.25 x V_{BAT} หรือประมาณ 3.75 V ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป

2.5.2 การจัดสรรหน่วยความจำใน DS1307

แอดเดรส	ข้อมูลที่เก็บ
0x00	วินาที
0x01	นาที
0x02	ชั่วโมง
0x03	วัน
0x04	วันที่
0x05	เดือน
0x06	ปี
0x07	รีจิสเตอร์ควบคุม
0x08-0x3F	หน่วยความจำแรม 56 ไบต์

2.5.3 รีจิสเตอร์ควบคุม

มีแอดเดรสอยู่ที่0x07 มีรายละเอียดของแต่ละบิตดังนี้

OUT (Output control) : ใช้ควบคุมระดับลอจิกที่ขา SQW OUT ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น "1" ที่ขา SQW OUT ก็จะเป็น "1" ถ้าบิตนี้เป็น "0" ที่ขา SQW OUT ก็จะเป็น "0"

SQW (Square Wave Enable) : ใช้เอ็นเอเบิลวงจรกำเนิดสัญญาณสี่เหลี่ยมที่ขา SQW OUT ถ้าต้องการให้สัญญาณสี่เหลี่ยมออกให้กำหนดบิตนี้เป็น "1"

(ก) การจัดสรรหน่วยความจำแรมภายใน DS1307

(ข) รายละเอียดของรีจิสเตอร์เก็บค่าเวลาและรีจิสเตอร์ควบคุม DS1307

RS1,RSO (Rate Select) : ใช้เลือกความถี่ของสัญญาณสี่เหลี่ยมที่ออกจากขา SQW OUT

RS1, RS0 (Rate Select) : ใช้เลือกความถี่ของสัญญาณสี่เหลี่ยมที่ออกจากขาSQW/OUT

RST	RS0	ค่าความถี่ของสัญญาณสี่เหลี่ยม
0	0	1Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768kHz

2.5.4 โหมดการทำงานของ DS1307

มี 2 โหมดคือ โหมดเขียนข้อมูล และ โหมดอ่านข้อมูล ในการใช้งานปกติจะใช้เฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1037 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1037 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูลต่อไป

2.5.5 โหมดการเขียนข้อมูล

เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (START :S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1037 ขั้นตอนต่อมาคือ ส่งข้อมูลเพื่อเลือกแอดเดรส ที่ต้องการเขียน จากนั้นรอการตอบรับ DS1037 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรับจาก DS1037 ทุดครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (STOP : P) เป็นการสิ้นสุดกระบวนการเขียนข้อมูล

2.5.6 โหมดการอ่านข้อมูล

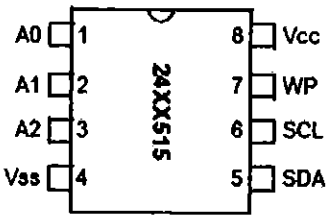
เริ่มต้นการทำงานเหมือนกับการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1037 เมื่อตอบรับเรียบร้อยแล้ว DS1037 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรสหรือ 1 ไบต์โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดล่วงหน้าด้วยโหมดการเขียนข้อมูล วิธีการง่ายๆคือ เข้าสู่โหมดการเขียนข้อมูลก่อนเมื่อถึงจังหวะที่ต้องเขียนข้อมูลให้ทำการสร้างสภาวะเริ่มต้น และส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้งตามด้วยเลือกโหมดการอ่านข้อมูลข้อมูลที่ออกมาจาก DS1037 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้

2.6 หน่วยความจำ อีอีพรอมอนุกรม 24LC515

หน่วยความจำอีอีพรอมอนุกรมเบอร์ 24LC515 เป็นหน่วยความจำแบบนอน-วาโลไทล์ (non-valotile) หมายความว่า สามารถเก็บรักษาข้อมูลอยู่ได้โดยไม่ต้องจ่ายไฟเลี้ยง สามารถ เขียน-อ่าน-ลบได้ด้วยสัญญาณไฟฟ้าใช้การเชื่อมต่อในลักษณะอนุกรมแบบระบบบัส I^2C มีขนาดหน่วยความจำเท่ากับ 512 กิโลบิต หรือ 64 กิโลไบต์ สามารถต่อพ่วงได้ทั้งหมด 8 ตัว โดยการกำหนดแอดเดรสทางฮาร์ดแวร์ที่ขา A0-A2 มีคุณสมบัติทางเทคนิคโดยสรุปดังนี้

- ขนาดของหน่วยความจำ 512 กิโลบิต หรือ 64 กิโลไบต์
- แรงดันใช้งาน 2.5-5.5 V
- กระแสขณะแสตนด์บายสูงสุด 100 mA
- กระแสขณะอ่านข้อมูลสูงสุด 1 mA
- กระแสขณะเขียนข้อมูลสูงสุด 3 mA
- ความถี่ของสัญญาณนาฬิกาต่ำสุด สูงสุด 100-400 kHz
- รอบในการเขียนข้อมูล 1 ล้านครั้ง
- ระยะเวลาการเก็บข้อมูล มากกว่า 200 ปี
- ระยะเวลาในการเขียนสูงสุด 5 มิลลิวินาที
- มีฟังก์ชั่นป้องกันการเขียนข้อมูลทับ

2.6.1 รายละเอียดขาต่อใช้งานของ 24LC515



รูปที่ 2.15 การจัดขาของ 24LC515

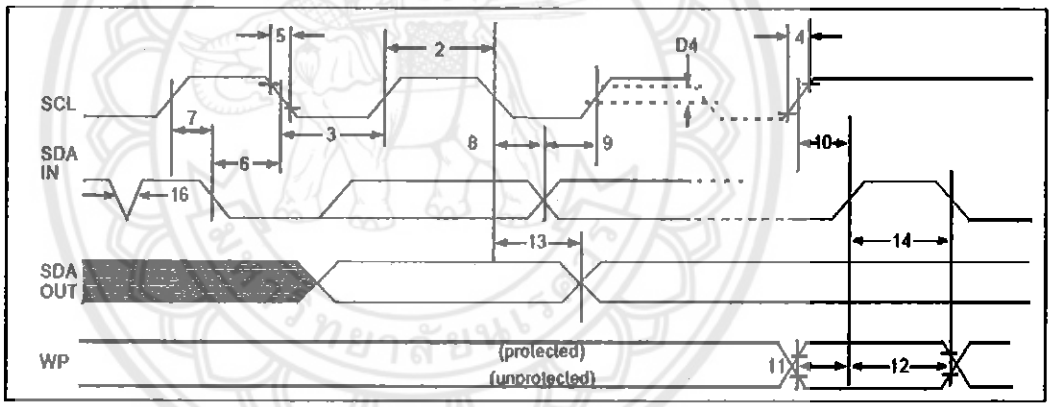
V_{CC} (ขา 8) ต่อกับไฟเลี้ยง + 5V

WP (ขา 7) ถ้าขานี้เป็นลอจิก 1 จะสามารถอ่านได้อย่างเดียว

SDA,SCL (ขา 5และ6) เป็นขาสำหรับเชื่อมต่อระบบบัส I^2C

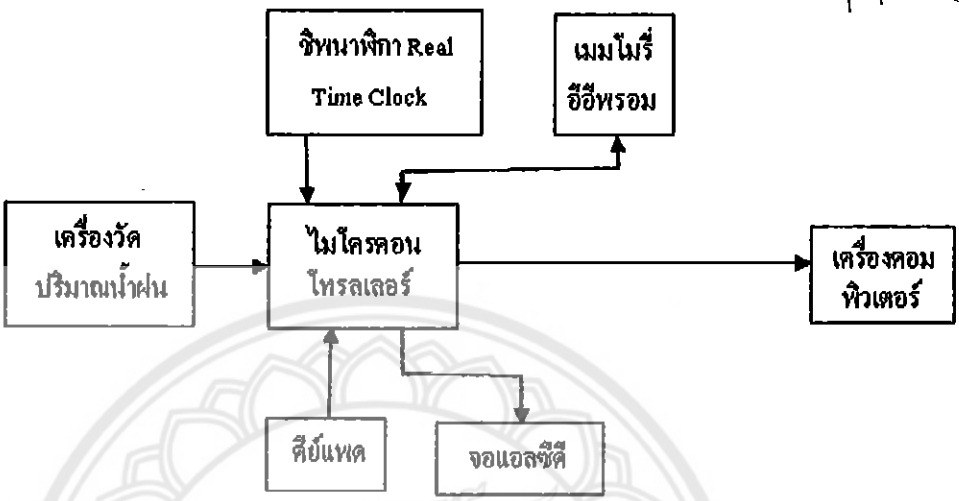
V_{SS} (ขา 4) ต่อกับกราวด์หรือไฟลบ

A0-A2 (ขา 1-3) เป็นขากำหนดแอดเดรสให้ต่อลงกราวด์ทั้งหมดหรือปล่อยลอยไว้



รูปที่2.16 ไดอะแกรมเวลาของ 24LC515

บทที่ 3
การออกแบบเครื่องวัดปริมาณน้ำฝน

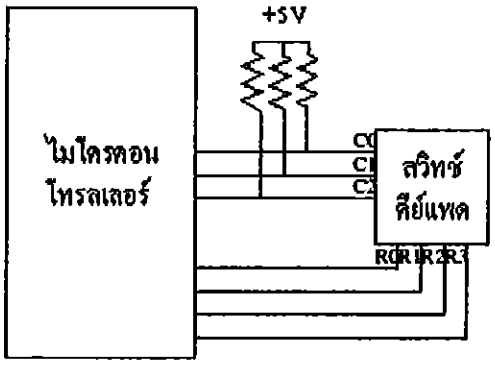


รูปที่ 3.1 แสดงบล็อกไดอะแกรมของระบบเครื่องวัดปริมาณน้ำฝน

วงจรของเครื่องวัดปริมาณน้ำฝนแสดงดังรูปที่ 3.1 โดยเมื่อเริ่มต้นทำงานไมโครคอน-โทรลเลอร์จะรับค่าเวลาจากชิพนาฬิกา Real Time Clock แล้วทำการแสดงค่าของวันที่ เดือน ปี เวลาทางจอแอลซีดี อีกทั้งยังสามารถตั้งค่าของวัน เวลาได้ทางสวิทช์คีย์แพด เมื่อมีการรับค่าน้ำฝนจากทางเครื่องวัดปริมาณน้ำฝนก็จะมีการเก็บข้อมูลวันเวลา พร้อมทั้งค่าปริมาณน้ำฝนไปเก็บที่เมมโมรี่อีอีพรอมพร้อมทั้งแสดงค่าปริมาณน้ำฝนทางจอแอลซีดีด้วย ซึ่งข้อมูลที่เก็บอยู่ที่อีอีพรอมนี้ยังสามารถส่งค่าเข้าไปเก็บยังคอมพิวเตอร์ได้อีกด้วย

3.1 การเชื่อมต่อสวิทช์คีย์แพดกับไมโครคอนโทรลเลอร์

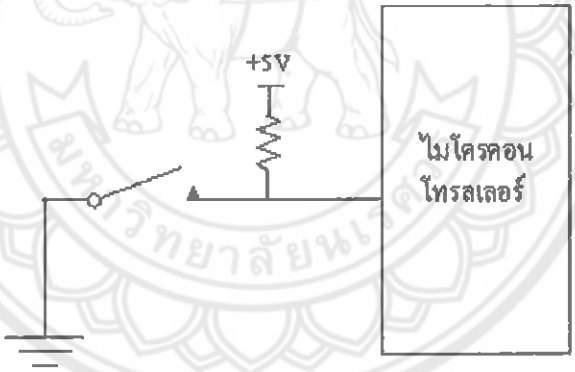
ในการเชื่อมต่อนี้ทำการเชื่อมต่อสวิทช์คีย์แพด(keypad) 4x3 ที่มีวงจรสวิทช์แบบเมตริก (matrix switch) เชื่อมต่อสายสัญญาณเพียง 7 เส้นแต่สามารถอ้างตำแหน่งได้ถึง 12 ตำแหน่ง



รูปที่3.2 วงจรแสดงการเชื่อมต่อสวิทช์คีย์แพดกับไมโครคอนโทรลเลอร์

3.2 การเชื่อมต่อเครื่องรับวัดปริมาณน้ำฝนกับไมโครคอนโทรลเลอร์

ในการเชื่อมต่อเครื่องวัดปริมาณน้ำฝน กับบอร์ดไมโครคอนโทรลเลอร์นั้นเครื่องวัดปริมาณน้ำฝนแบบคานกระดกนั้น จะมีตัวรับสัญญาณเป็นเหมือนกับสวิทช์ 1 ตัว โดยเมื่อมีการกระดกลงมากดสวิทช์ก็จะทำให้สภาวะที่พอร์ตนั้นเปลี่ยนจากลอจิก "1" เป็นลอจิก "0" เป็นผลให้ไมโครคอนโทรลเลอร์รับทราบว่ามีการรับน้ำฝนเข้ามาได้

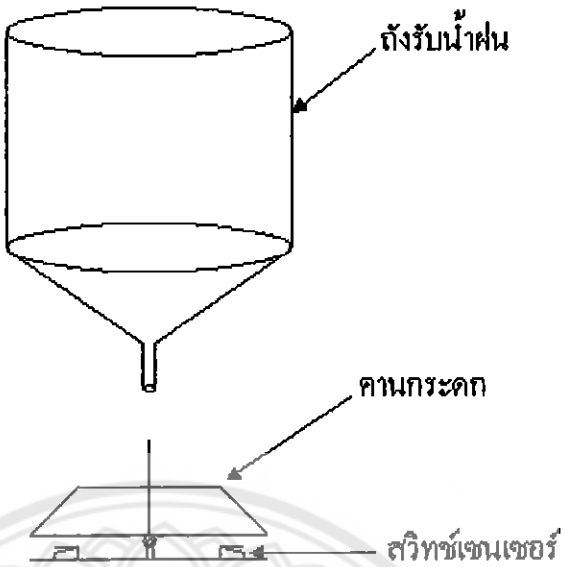


รูปที่3.3 วงจรแสดงการเชื่อมต่อเครื่องวัดปริมาณน้ำฝนกับไมโครคอนโทรลเลอร์

3.3 โครงสร้างเครื่องวัดปริมาณน้ำฝนแบบคานกระดก

ประกอบด้วยส่วนประกอบดังนี้

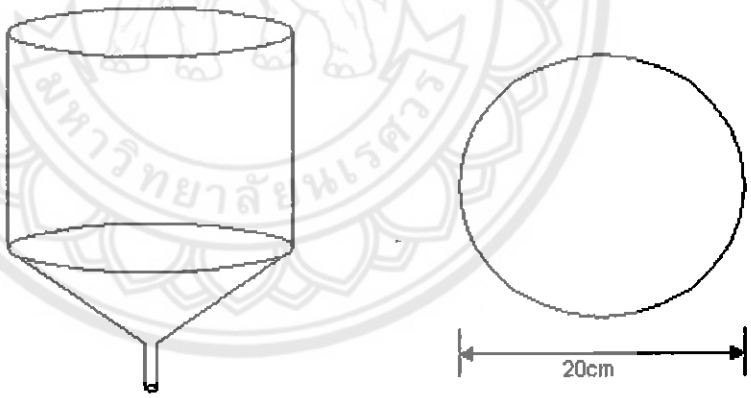
1. ส่วนถังรับน้ำฝน
2. ส่วนคานกระดก
3. ส่วนสวิทช์เซนเซอร์



รูปที่3.4 แสดงเครื่องวัดปริมาณน้ำฝนแบบคานกระดก

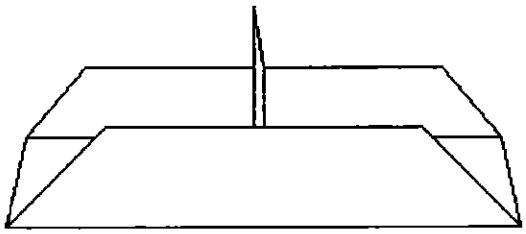
3.3.1 ถังรับน้ำฝน

ถังรับมีลักษณะทรงกระบอกมีเส้นผ่านศูนย์กลาง 20 เซนติเมตร

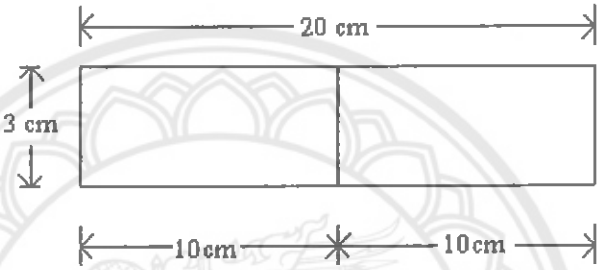


รูปที่3.5 แสดงรายละเอียดถังรับน้ำฝน

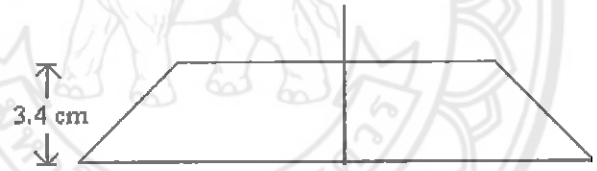
3.3.2 คานกระดก



รูปที่3.6 แสดงส่วนรับน้ำของคานกระดกในลักษณะเอียงข้าง



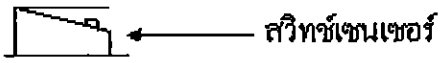
รูปที่3.7 ภาพด้านบนแสดงความกว้างและความยาวของคานกระดก



รูปที่3.8 ภาพด้านหน้าแสดงความสูงของคานกระดก

3.3.3 สวิทช์เซนเซอร์

สวิทช์เซนเซอร์มีลักษณะเป็นสวิทช์กดติดปล่อยดับ



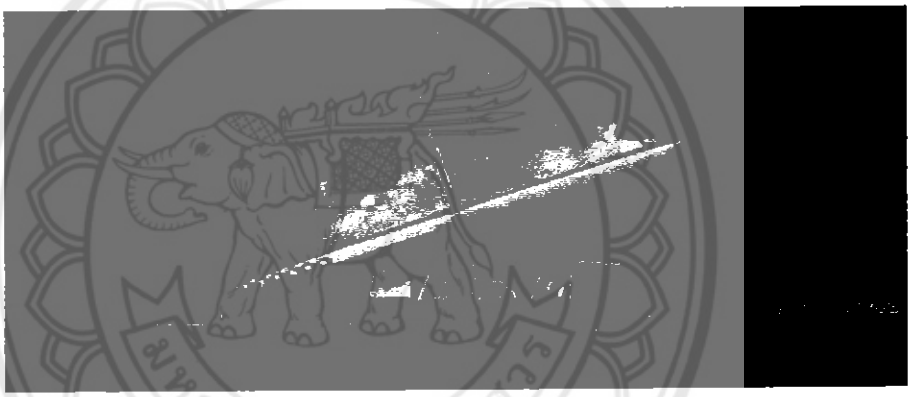
รูปที่3.9 แสดงลักษณะเซนเซอร์

3.3.4 แสดงรูปอุปกรณ์เมื่อสร้างสำเร็จแล้ว

รูปที่ 3.10 แสดงถังรับน้ำด้านหน้า

รูปที่ 3.11 แสดงถังรับน้ำด้านบน

รูปที่ 3.12 แสดงถังรับน้ำด้านข้าง



ก.

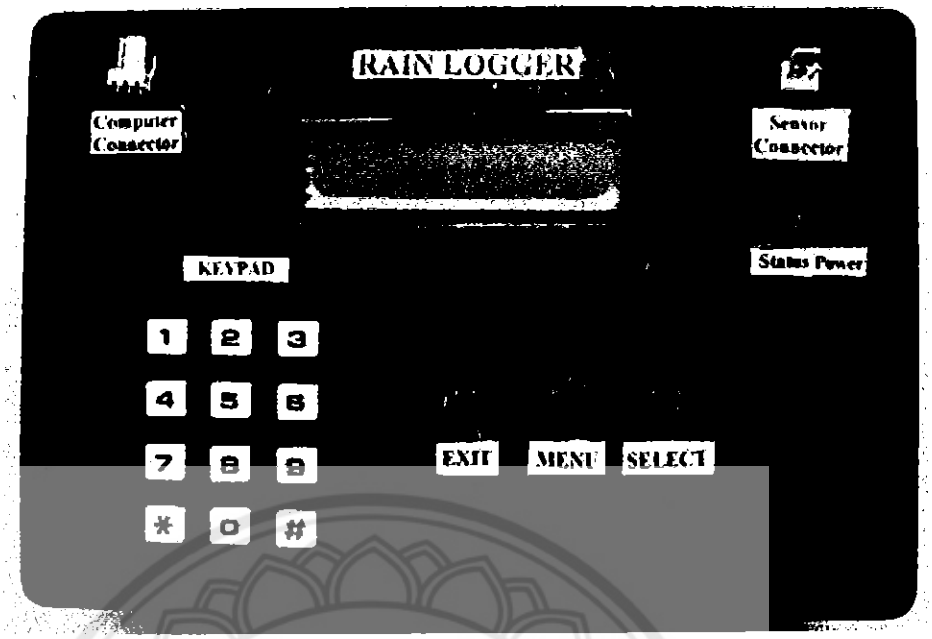
ข.

รูปที่ 3.13 ก. แสดงคานกระดกด้านข้าง

รูปที่ 3.13 ข. แสดงคานกระดกด้านบน

รูปที่ 3.14 แสดงส่วนสวิทช์เซ็นเซอร์รับค่าการเคาะ

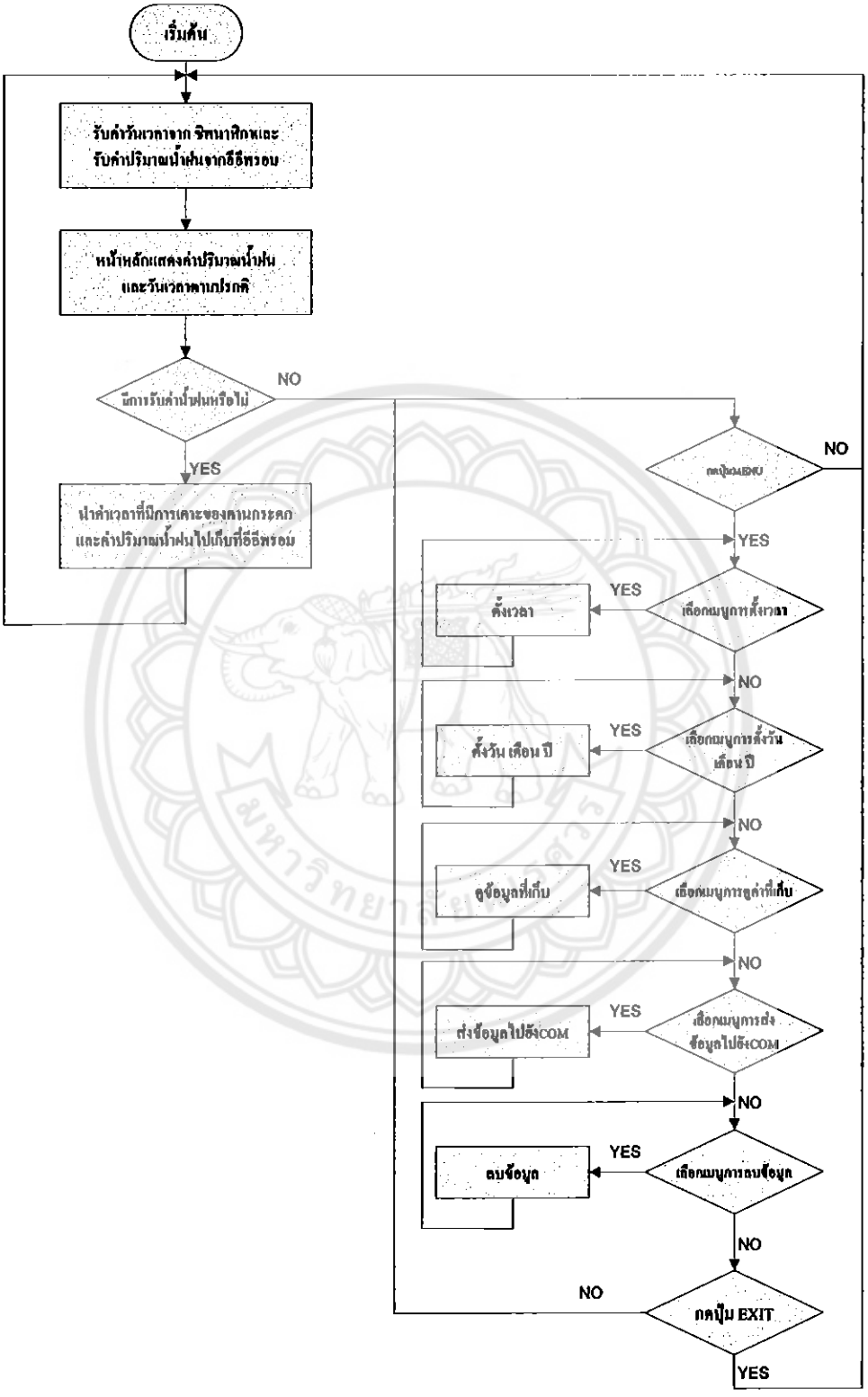
รูปที่ 3.15 การเชื่อมต่ออุปกรณ์ต่างๆเข้าด้วยกัน



รูปที่3.16 ลงอุปกรณ์ในกล่องเรียบร้อย

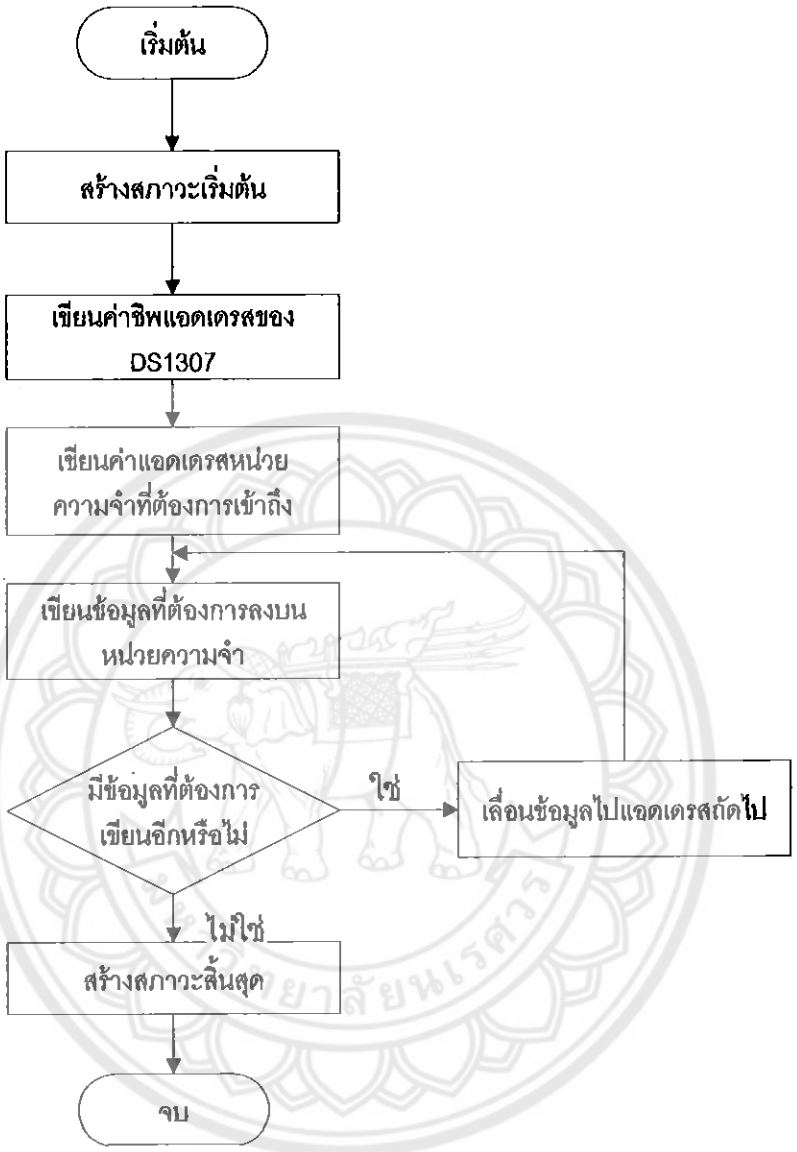
รูปที่ 3.17 การเชื่อมส่วนรับน้ำฝน คานกระดกและตัวเก็บข้อมูลไว้ด้วยกัน

3.4 โฟลว์ชาร์ตโปรแกรมหลัก



รูปที่ 3.18 แสดงโฟลว์ชาร์ตโปรแกรมหลัก

3.4.1 โฟลว์ชาร์ตของโปรแกรมการเขียนค่าเวลาและข้อมูลลงบนชิพนาฬิกา DS1307

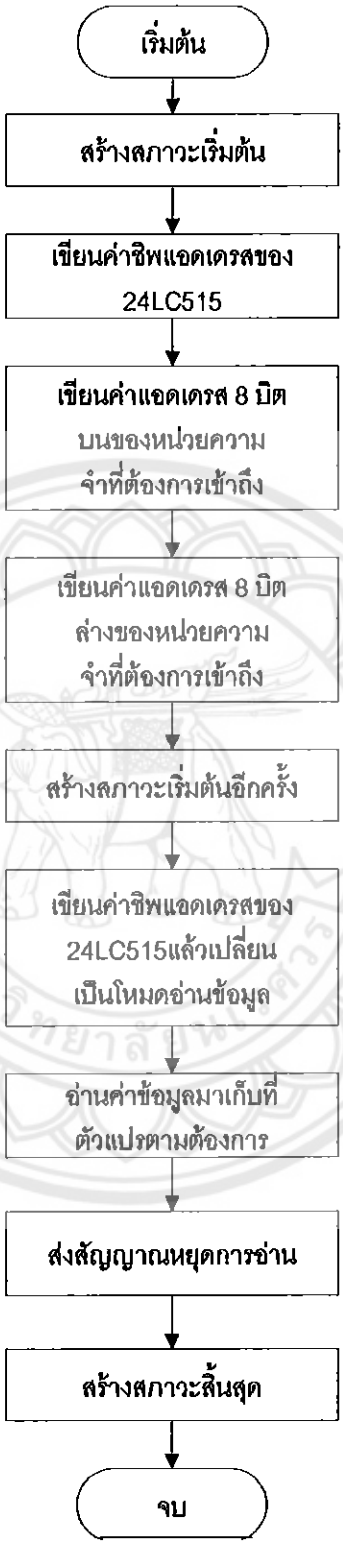


รูปที่ 3.19 แสดงโฟลว์ชาร์ตการเขียนค่าเวลาและข้อมูลลงบนชิพนาฬิกา DS1307

3.4.1 โฟลว์ชาร์ตของโปรแกรมการอ่านค่าเวลาและข้อมูลจากชิพนาฬิกา DS1307

รูปที่ 3.20 แสดงโฟลว์ชาร์ตการเขียนค่าเวลาและข้อมูลลงบนชิพนาฬิกา DS1307

3.4.2 โฟลว์ชาร์ตของโปรแกรมการอ่านค่าข้อมูลจากชิพอีอีพรอม 24LC515



รูปที่ 3.21 โฟลว์ชาร์ตของโปรแกรมการอ่านค่าข้อมูลจากชิพอีอีพรอม 24LC515

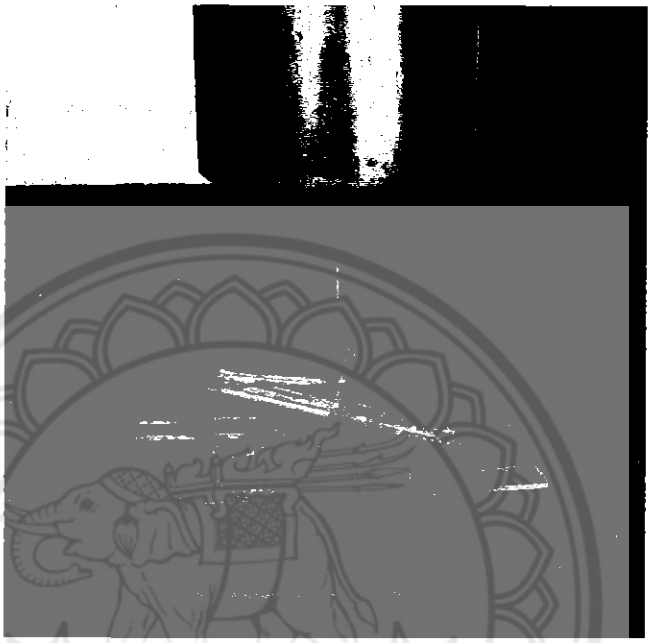
3.4.3 โฟลว์ชาร์ตของโปรแกรมการเขียนข้อมูลลงบนชิพอีอีพรอม 24LC515

รูปที่ 3.22 โฟลว์ชาร์ตของโปรแกรมการเขียนข้อมูลลงบนชิพอีอีพรอม 24LC515

บทที่ 4

การทดลอง

4.1 การทดลองรับน้ำและการทดลองคานกระดก

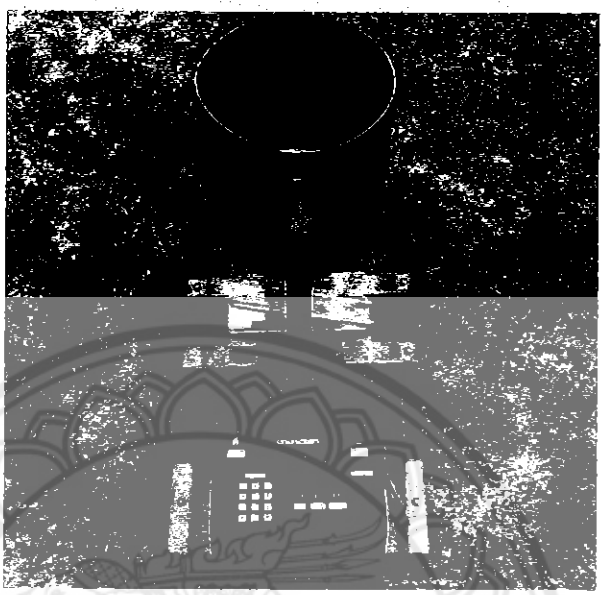


รูปที่ 4.1 แสดงการรับน้ำจากถังรับน้ำแล้วมีการไหลลงมาเก็บที่คานกระดก

รูปที่ 4.2 เมื่อมีการรับน้ำฝนได้ในปริมาณที่กำหนดคานกระดกก็จะทำการเคาะเซนเซอร์

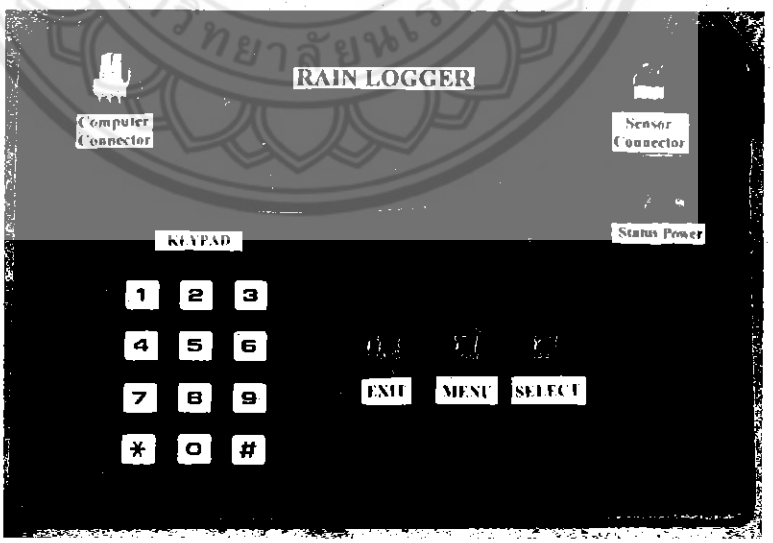
4.2 การทดลองต่อส่วนรับน้ำเข้ากับเครื่องรับข้อมูลและการแสดงค่าปริมาณน้ำฝน

ทำการเชื่อมต่อสายสัญญาณสวิทช์เซนเซอร์เข้ากับคอนเน็คเตอร์ทางด้านขวาของเครื่องรับข้อมูล



รูปที่4.3 แสดงการต่อส่วนรับน้ำเข้ากับเครื่องรับข้อมูล

ในหน้าการแสดงผลหลักจะแสดงค่าเวลาและค่าปริมาณน้ำฝนซึ่งเมื่อมีการรับสัญญาณการเคาะไฟสถานะสีเขียวจะติดและจะมีการเก็บข้อมูลในอีอีพรอมต่อมาไมโครคอนโทรลเลอร์จะทำการคำนวณค่าปริมาณน้ำฝนแล้วแสดงค่าในหน่วยไมโครเมตรต่อนาทีทางจอแอลซีดี



รูปที่4.4 การแสดงผลหน้าหลักแสดงค่าปริมาณน้ำฝนและเวลาทางจอแอลซีดี

4.3 การทดลองตั้งเวลา

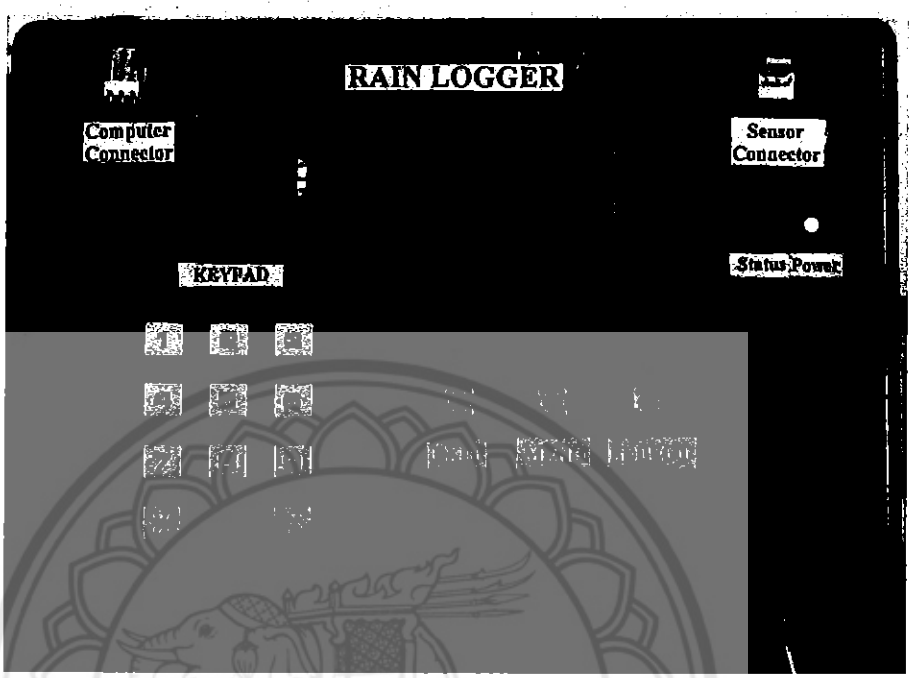
จากหน้าแสดงผลหลักทำการกดปุ่ม MENU เพื่อเลื่อนลูกศรไปที่เมนู SETTING TIME

รูปที่ 4.5 แสดงการเลือกเมนูตั้งเวลา

เมื่อลูกศรอยู่ที่เมนู SETTING TIME ทำการกดปุ่ม SELECT เพื่อเข้าไปทำการตั้งเวลา ซึ่งจอแอลซีดีจะมีเคอร์เซอร์กระพริบในตำแหน่งที่ต้องการรับค่า

รูปที่ 4.6 แสดงหน้าจอเตรียมพร้อมรับค่าเวลา

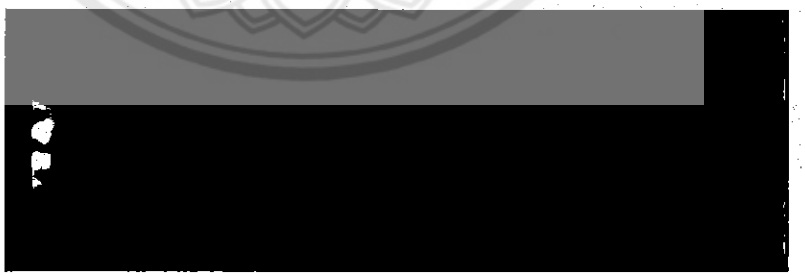
ทำการป้อนค่าเวลาทางสวิทช์คีย์แพด (SWITCH KEYPAD) เมื่อทำการป้อนค่าเวลาในแต่ละตำแหน่งเคอร์เซอร์กระพริบก็จะเลื่อนไปทางขวาเพื่อทำการรับค่าใหม่ไปเรื่อยๆจนครบ แล้วจึงทำการแสดงหน้าเมนู SETTING TIME อีกครั้งหนึ่ง



รูปที่ 4.7 แสดงการป้อนค่าเวลาทางสวิทช์คีย์แพด(SWITCH KEYPAD)

4.4 การทดลองตั้งวัน เดือน ปี

จากหน้าแสดงผลหลักทำการกดปุ่ม MENU เพื่อเลื่อนลูกศรไปที่เมนู SETTING DATE

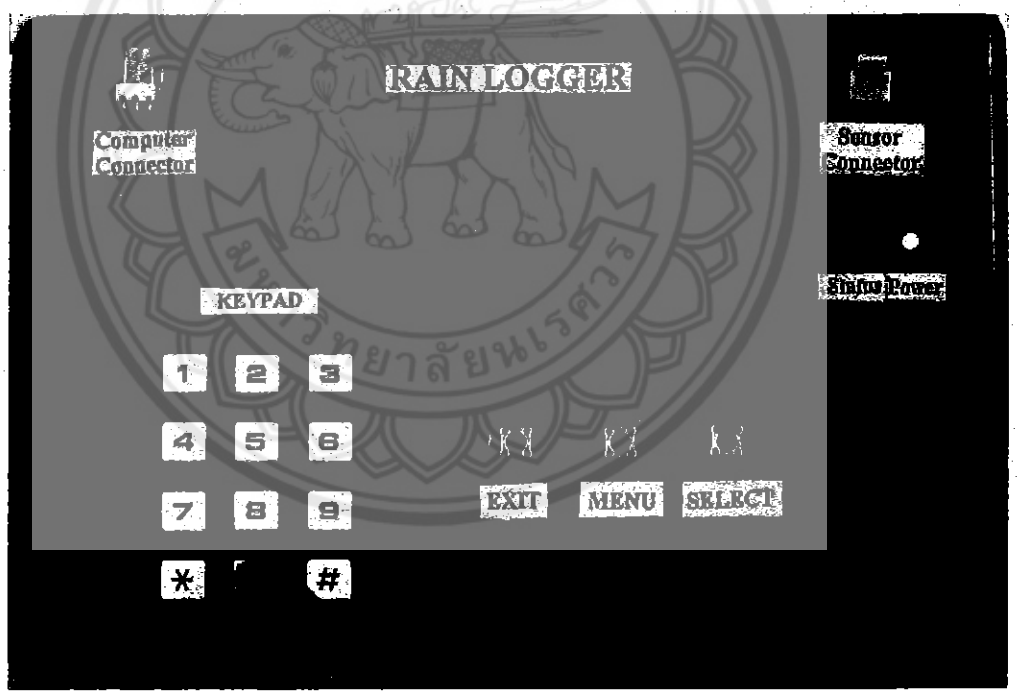


รูปที่ 4.8 แสดงการเลือกเมนูตั้งวัน เดือน ปี

เมื่อลูกศรอยู่ที่เมนู SETTING DATE ทำการกดปุ่ม SELECT เพื่อเข้าไปทำการตั้งวัน เดือน ปี ซึ่งจอแอลซีดีก็มีเคอร์เซอร์กระพริบในตำแหน่งที่ต้องการรับค่า

รูปที่ 4.9 แสดงหน้าจอเตรียมพร้อมรับค่าวัน เดือน ปี

ทำการป้อนค่าเวลาทางสวิทช์คีย์แพด (SWITCH KEYPAD) เมื่อทำการป้อนค่าเวลาในแต่ละตำแหน่งเคอร์เซอร์กระพริบก็จะเลื่อนไปทางขวาเพื่อทำการรับค่าใหม่ไปเรื่อยๆจนครบ แล้วจึงทำการแสดงหน้าเมนู SETTING TIME อีกครั้งหนึ่ง



รูปที่ 4.10 แสดงการป้อนค่าวัน เดือน ปี ทางสวิทช์คีย์แพด(SWITCH KEYPAD)

4.5 การทดลองดูข้อมูลปริมาณน้ำฝนที่เก็บในอีอีพรอม

จากหน้าแสดงผลหลักทำการกดปุ่ม MENU เพื่อเลื่อนลูกศรไปที่เมนู VIEW DATA

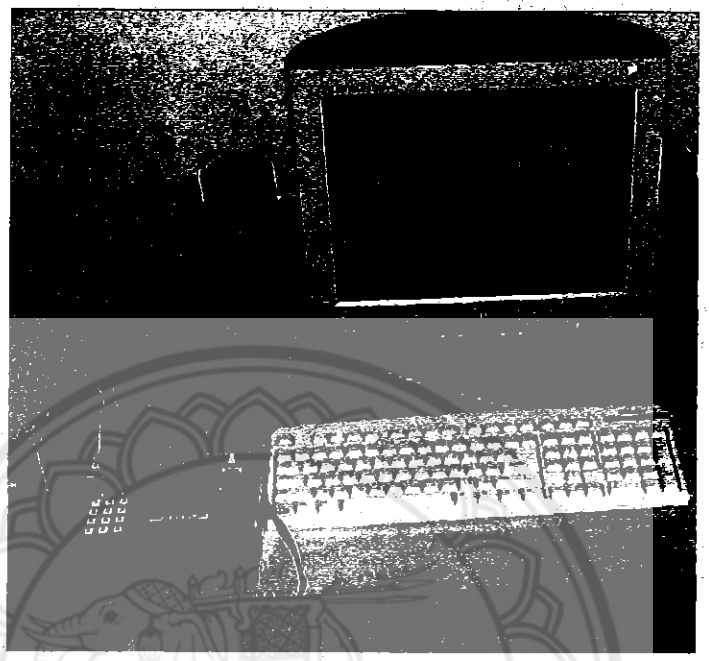
รูปที่ 4.11 แสดงเมนูการเลือกดูข้อมูลปริมาณน้ำฝน

เมื่อลูกศรอยู่ที่เมนู VIEW DATA ทำการกดปุ่ม SELECT เพื่อเข้าไปดูข้อมูลปริมาณน้ำฝนที่เก็บในอีอีพรอมอีกทั้งยังสามารถเลื่อนดูปริมาณน้ำฝนค่าต่างๆ โดยการเลื่อนขึ้น-ลง ด้วยปุ่ม SELECT และ ปุ่ม MENU หากต้องการกลับหน้าหลักให้กดปุ่ม EXIT

รูปที่ 4.12 ข้อมูลปริมาณน้ำฝนที่แสดงทางจอแอลซีดี

4.6 การทดลองโหลดข้อมูลเข้าคอมพิวเตอร์

ทำการเชื่อมสายส่งข้อมูลเข้าที่ COMPUTER CONNECTOR ทางด้านซ้ายของเครื่องวัดปริมาณน้ำฝนเข้ากับ COM1 หรือ SERIAL PORT ของเครื่องคอมพิวเตอร์

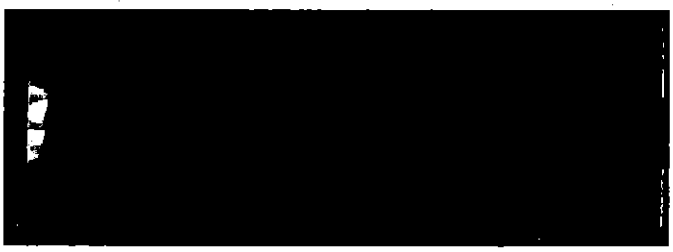


รูปที่ 4.13 การเชื่อมต่อสายสัญญาณจากเครื่องวัดปริมาณน้ำฝนเพื่อส่งข้อมูลไปยังเครื่องคอมพิวเตอร์

จากหน้าแสดงผลหลักทำการกดปุ่ม MENU เพื่อเลื่อนลูกศรไปที่เมนู UPLOAD DATA เพื่อส่งข้อมูลไปยังเครื่องคอมพิวเตอร์

รูปที่ 4.14 การเลือกเมนูเพื่อส่งข้อมูลไปยังคอมพิวเตอร์

เมื่อลูกศรอยู่ที่เมนู UPLOAD DATA ทำการกดปุ่ม SELECT เพื่อเตรียมพร้อมที่จะส่งข้อมูลไปยังคอมพิวเตอร์



รูปที่ 4.15 เตรียมพร้อมส่งข้อมูลไปยังคอมพิวเตอร์

ทำการเปิดโปรแกรมในส่วนของการโหลดข้อมูลมายังคอมพิวเตอร์แล้วกดปุ่ม Download

รูปที่ 4.16 แสดงโปรแกรมที่ใช้โหลดข้อมูลมายังคอมพิวเตอร์

ทำการโหลดข้อมูลปริมาณน้ำฝนมายังคอมพิวเตอร์

รูปที่ 4.17 แสดงข้อมูลที่ทำการส่งมายังคอมพิวเตอร์แล้ว

4.7 การทดลองลบข้อมูลในอีอีพรอม

จากหน้าแสดงผลหลักทำการกดปุ่ม MENU เพื่อเลื่อนลูกศรไปที่เมนู CLEAR MEMORY เพื่อเข้าไปลบข้อมูลในอีอีพรอม

รูปที่ 4.18 การเลือกเมนูลบข้อมูล

เมื่อลูกศรอยู่ที่เมนู CLEAR MEMORY ทำการกดปุ่ม SELECT เพื่อเข้าเมนูยืนยันที่จะลบข้อมูลในอีอีพรอมเมื่อกดปุ่ม SELECT อีกครั้งข้อมูลทั้งหมดจะถูกลบ

รูปที่ 4.19 การเลือกเมนูยืนยันที่จะลบข้อมูล

4.8 การเปรียบเทียบค่าที่สามารถเก็บและแสดงผลได้กับค่าที่ได้การคำนวณในส่วนของคานกระดก

ใช้วิธีการหาปริมาตรคงที่ที่ใช้ในการรับน้ำในแต่ละครั้ง เพื่อใช้ในการคำนวณหาค่าปริมาณน้ำฝนนั้นทำได้โดยการใช้หลอดฉีดยาดูดน้ำแล้วทำการหยดลงในส่วนของคานกระดกครั้งละ 1 cc หรือ 1 cm^3 ปรากฏว่าในส่วนของคานกระดกด้านซ้ายเมื่อมองจากด้านหน้าของตัวรับน้ำฝนสามารถรับน้ำได้ 36 cm^3 ส่วนในด้านขวา สามารถรับน้ำได้ 38 cm^3 ดังนั้นในส่วนของการนำค่าคงที่ไปใช้ในการคำนวณจึงใช้ค่าเฉลี่ยคือ 37 cm^3 ซึ่งค่าความผิดพลาดในการรับน้ำในแต่ละครั้งคือ 2.7%

รูปที่ 4.20 แสดงการหาปริมาตรที่แท้จริงซึ่งใช้ในการเคาะคานกระดกในแต่ละครั้ง

4.8.1 ค่าปริมาณน้ำฝนที่ได้จากการเคาะคานกระดกในแต่ละครั้ง

พื้นที่หน้าตัดที่ใช้ในการรับน้ำ $= \pi r^2 = 3.14(10^2) = 314\ cm^2$

ปริมาตรที่ใช้ในการกระดก 37 cm^3

$\therefore$ ค่าปริมาณน้ำฝนที่ได้จากการเคาะแต่ละครั้ง $= \frac{37\ cm^3}{314\ cm^2} = 0.1178$ cm หรือ 1.178 mm

จากการทดลองรับค่าน้ำฝนจนมีการเคาะคานกระดก 1 ครั้งซึ่งมีเวลาในการเคาะห่างจากครั้งก่อน 78 วินาที

4.8.2 ค่าที่ได้จากการคำนวณ

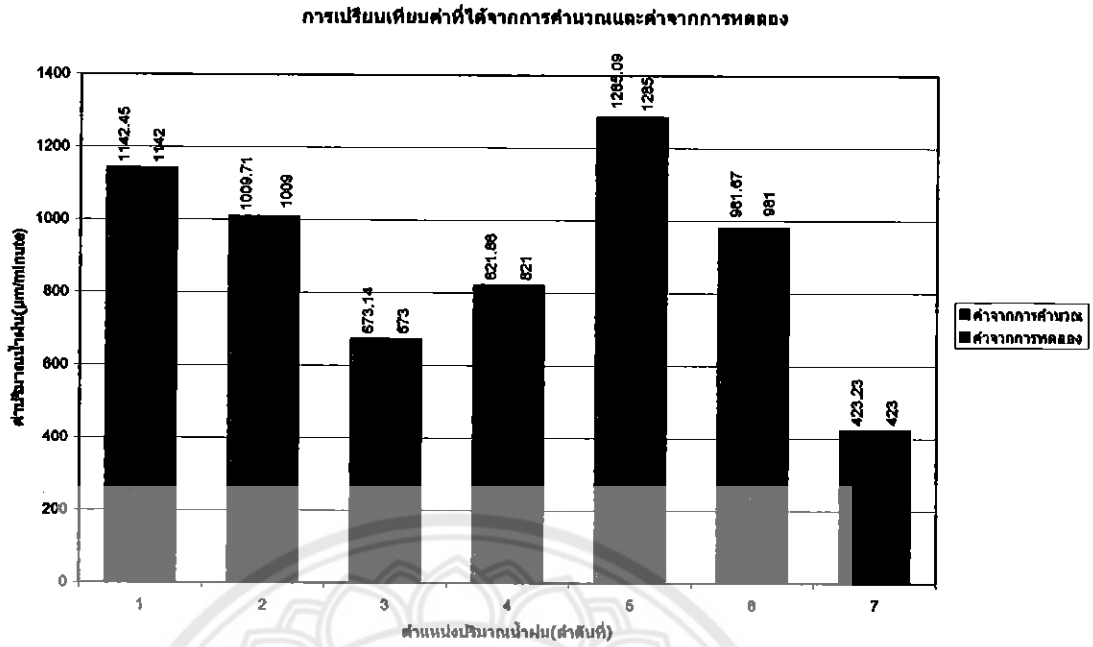
ค่าปริมาณน้ำฝนที่ได้ต่อเวลาเป็นนาทีคือ

$$= \frac{1.178\ mm}{\frac{78\ s}{60\ s}} = 0.9062 \text{ mm/minute หรือ } 906.2 \text{ μm/minute}$$

ค่าที่ได้จากการทดลอง 906 μm/minute ซึ่งมีความผิดพลาดในการเก็บข้อมูลคือ 0.022%

ตารางที่ 4.1 การเปรียบเทียบค่าที่ได้จากการคำนวณและค่าที่ได้จากการทดลอง

ระยะเวลาที่ห่างกันในการเคาะแต่ละครั้ง(วินาที)	ค่าที่ได้จากการคำนวณ (μm/minute)	ค่าที่ได้จากการทดลอง (μm/minute)	ค่าความผิดพลาด %
49	1142.45	1142	0.04
70	1009.71	1009	0.07
105	673.14	673	0.02
86	821.86	821	0.10
55	1285.09	1285	0.01
72	981.67	981	0.06
167	423.23	423	0.05



รูปที่ 4.21 แผนภูมิการเปรียบเทียบค่าที่ได้จากการคำนวณและค่าที่ได้จากการทดลอง

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลองรับน้ำและคานกระดกปรากฏว่า เมื่อมีการรับน้ำในที่รองรับแล้วส่งมาที่คานกระดกจะมีการรับน้ำในปริมาณที่กำหนดแล้วทำการเตาะสวิทช์เซนเซอร์ ส่งผลให้ไมโครคอนโทรลเลอร์นำข้อมูลที่ได้มาวิเคราะห์ผลแล้วแสดงค่าปริมาณน้ำฝนต่อเวลาจริงได้ อีกทั้งยังสามารถตั้งเวลาและวัน เดือน ปี ที่ใช้ในการเก็บข้อมูลปริมาณน้ำฝนได้ ส่วนเมื่อมีการเก็บข้อมูลมายังอีอีพรอมที่ใช้เป็นดาต้าล็อกเกอร์แล้ว ก็สามารถเลือกดูค่าเวลาและปริมาณน้ำฝนที่เก็บอยู่ได้ด้วยจอแอลซีดีที่อยู่บนเครื่องได้โดยไม่ต้องต่อผ่านคอมพิวเตอร์อีกด้วย และเมื่อหน่วยความจำที่ใช้เก็บข้อมูลเต็มหรือต้องการที่จะลบข้อมูลที่เก็บไว้ก็สามารถทำการลบข้อมูลได้

5.2 ข้อเสนอแนะและแนวทางแก้ไข

1.เนื่องจากความไม่สะดวกในการทำแผ่นปริ้นท์วงจร จึงได้ประกอบบอร์ดอินเตอร์เฟสโดยใช้สายไวร์-แรบปิ้ง ซึ่งอาจจะทำให้เกิดสัญญาณรบกวนได้ง่าย

2.เนื่องจากขาดความชำนาญและขาดเครื่องมือ และอุปกรณ์ที่เพียงพอทำให้การสร้างส่วนของคานกระดกนั้น ปริมาตรที่ได้ในการกระดกแต่ละครั้งอาจมีความคลาดเคลื่อนได้

3. ในส่วนของคานกระดกและสวิทย์เซนเซอร์นั้น เป็นส่วนที่มีการเคลื่อนไหวอุปกรณ์ที่ใช้ในการสร้างเป็นอะครีลิกและพลาสติกซึ่งจะมีผลทำให้เกิดการชำรุดได้ง่าย

4. อุปกรณ์ที่ต้องใช้ในวงจรหลายตัวมีการติดต่อผ่านทาง I^2C Bus ซึ่งมีความซับซ้อนและเป็นเรื่องที่ผู้ทำการวิจัยต้องศึกษาใหม่ทำให้มีความยุ่งยากในการใช้อุปกรณ์แต่ละตัวพอสมควร

5. ในโครงงานนี้มีความจำเป็นที่จะต้องใช้เนื้อที่ในไมโครคอนโทรลเลอร์ เป็นจำนวนมาก เพื่อทำการเก็บข้อมูลเพื่อนำไปจัดเก็บใน database ทำให้มีความยุ่งยากในการเขียนโปรแกรมเพื่อให้กะทัดรัดและเปลืองหน่วยความจำน้อยที่สุด

6. เนื่องจากโครงงานที่สร้างขึ้นครั้งนี้เป็นในส่วนของเครื่องมือวัด ผลการทดลองที่ได้นั้นให้ผลได้ตามการคำนวณและทางทฤษฎี ยังไม่สามารถนำไปตากฝนและทำการวัดค่าจริงเพื่อทำการเปรียบเทียบกับเครื่องมือที่ใช้งานจริงได้ซึ่งในโอกาสหน้าควรจะมีการพัฒนาต่อยอดต่อไป

5.3 แนวทางการพัฒนาต่อไป

1.ต้องมีการพัฒนาอุปกรณ์ในการชั่งตวงของคานกระดกให้มีปริมาตรที่ถูกต้องแม่นยำมากขึ้น

2. ต้องมีการซีลซ์อุปกรณ์ให้สามารถกันความชื้นได้ เนื่องจากอุปกรณ์ที่ใช้เป็นอุปกรณ์อิเล็กทรอนิคส์

3. พัฒนาโปรแกรมเพื่อให้สามารถแจ้งเตือนเมื่อฝนมีการตกหนักจนถึงขั้นวิกฤติ

4. มีการปรับตั้งเปรียบเทียบกับเครื่องมือที่ใช้ในปัจจุบันและสามารถนำไปใช้งานได้จริง

เอกสารอ้างอิง

[1] ธีรบูลย์ หล่อวิเชียรรุ่ง และคณะ. ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษาซี. พิมพ์ครั้งที่ 1. กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. 2547.

[2] ผศ.ธีรวัฒน์ ประกอบผล. การประยุกต์ใช้งานไมโครคอลโทรลเลอร์. พิมพ์ครั้งที่ 6. กรุงเทพฯ: 2543.

[3] ผศ.ธีรวัฒน์ ประกอบผล. การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี. กรุงเทพฯ : สำนักพิมพ์ ส.ส.ท. 2546.

[4] ผศ.พิพัฒน์ เลาสงคราม. ไมโครคอนโทรลเลอร์. พิมพ์ครั้งที่1 , กรุงเทพฯ : หจก.วีเจ พริ้นติ้ง.

[5] ศูนย์กรมอุทกวิทยาและบริหารน้ำ ภาคเหนือตอนล่าง สำนักงานอุทกและบริหารน้ำ กรมชลประทาน พิษณุโลก.

ภาคผนวก ก

```
#include<stdio.h>
#include <ctype.h>
#include<string.h>
#include<intrins.h>
#include<reg52.h>                    // Header include register of P89C51RD2
#include<i2c.h>                      // Module function i2c bus
#include<lcd.h>                      // Module function LCD
#include<scankey.h>                  // Module function scankey 4x3
#define DS1307_ID 0xD0               // Define constant DS1307_ID
#define AT24C16_ID 0xA0

#define reg_addr      0x00

#define DEF_addr 0x20

#define dat_cnt_addhi 0x00
#define dat_cnt_addlo 0x01
#define last_rec_addhi 0x02
#define last_rec_addlo 0x03



sbit sw_MENU = P1^7;
sbit sw_ESC = P1^6;
sbit sw_ENT = P2^7;
sbit sensor = P3^2;
sbit sensor2 = P3^3;

sbit LED = P3^7;

bit err = 0;
data unsigned char eep_dat[2];       // For keep data in EEPROM for read/write
unsigned char eep_addr = 0,eep_value = 0,eep_buff = 0,eep_buff2 = 0;
```

```
//---------------------------
char    last_addr_hi=reg_addr;
char    last_addr_lo=DEF_addr;

char    addr_hi=reg_addr;
char    addr_lo=DEF_addr;

//---------------------------

//unsigned char result1;
//unsigned char secLast,secCur;
//bit countYet=0;
//char RainDG1,RainDG2;
//---------------------------

unsigned char dig1,dig2,dig3,dig4,dig5;
//unsigned char digx1,digx2,digx3;

bit scr_nomal = 1;                          // Bit check screen display Nomal page
unsigned char sec,min,hour,day,date,month,year,control;    // For keep Date and Time
code unsigned char *_Date = "          ";      // Pointer for screen text "Date"
code unsigned char *Time = "uM/min     ";                  // Pointer for screen text
"Time"
code unsigned char *setdate = "Setting Date";              // Pointer for screen text
"Setting Date"
code unsigned char *dmy = "dd/mm/yy";                                  // Pointer
for screen text "dd/mm/yy"
code unsigned char *settime = "Setting Time";              // Pointer for screen text
"Setting Time"
code unsigned char *hms = "hh:mm:ss";                                  // Pointer
for screen text "hh/mm/ss"
unsigned char date_buf[6];              // For keep Date value from push key
unsigned char time_buf[6];                                    // For keep Time
value from push key
unsigned char page = 1;                                                    //
For keep page display default = page1

//unsigned int rec;
unsigned char secOld;
unsigned long secCnt;

unsigned int Result;
```

```
unsigned char subMenu = 1;

        unsigned char x_key = 0;                // Keep for check push key
        unsigned char push_key = 0;     // Keep for value push key
/*************************************************************************
Text for menu
*************************************************************************/
code unsigned char *msg_MenuSetup =      "     Setup     ";
code unsigned char *msg_MenuLoading =   "  Loading...    ";

code unsigned char *msg_MenuClrMem =   "  CLEAR MEMORY  ";
code unsigned char *msg_MenuSetTime =   "  SETTING TIME  ";
code unsigned char *msg_MenuSetDate =   "  SETTING DATE  ";
code unsigned char *msg_ViewData =              "  VIEW DATA     ";
code unsigned char *msg_Upload =          "  UPLOAD DATA   ";

code unsigned char *msg_Connect =                "Wait for connect";
code unsigned char *msg_Upload_dat =      " Connecting...  ";
code unsigned char *msg_PC_connect =      "   Link to PC   ";

code unsigned char *msg_ClearLog =        "Clear  memory!!!";
code unsigned char *msg_YesNo =   " Please confirm ";
code unsigned char *msg_MemClear =        "  Memory empty  ";
code unsigned char *msg_Done =            "    All done    ";

code unsigned char *msg_Wait =            "Moment please...";
code unsigned char *msg_Review =          "REVIEW         ";

//code unsigned char *msg_RainDisp =      "      Cm/minute";

unsigned int i,h;
void delay_10ms(unsigned char d10ms)
        {
             while(d10ms--)
                for(h=0;h<=700;h++);  //time delay 10ms
        }

void delay_50ms(unsigned char d50ms)
        {
             while(d50ms--)
                for(i=0;i<=3550;i++);  //time delay 50ms
        }
void delay_100ms(unsigned char d100ms)
        {
                while(d100ms--)
```

```
            delay_50ms(2);
        }

void delay_1s(unsigned char d1s)
        {
            while(d1s--)
            delay_100ms(10);
        }

/*****************************************************************************/


/*****************************************************************************/
/********************** Function convert to ascii and send to LCD
*****************/
/*****************************************************************************/
void send_to_lcd(unsigned char value)
{
        unsigned char buf = 0;                  // For keep value convert

        buf = value & 0xF0;          // Filter for high byte
        buf = (buf>>4)|(0x30);       // Shift to 4 time(swap) OR 0x30 for convert to ascii
code
        lcd_text(buf);               // Send to show LCD
        buf = value & 0x0F;                              // Filter for low byte
        buf = buf | 0x30;            // OR 0x30 for convert to ascii code
        lcd_text(buf);                                   // Send to show LCD
}

void DisplayRain(void)
{

        lcd_text(dig5|0x30);
        lcd_text(dig4|0x30);
        lcd_text(dig3|0x30);
        lcd_text(dig2|0x30);
        lcd_text(dig1|0x30);
}


void send_int_to_lcd(unsigned int value)
{
```

```
        unsigned char buf = 0;                  // For keep value convert

//      unsigned char dig1,dig2,dig3,dig4,dig5;

        dig1=value%10;

        value=value/10;
        dig2=value%10;

        value=value/10;
        dig3=value%10;

        value=value/10;
        dig4=value%10;

        value=value/10;
        dig5=value%10;

        DisplayRain();

}

void RainCaculate(void)
{
        if(secCnt!=0)
        {
        Result=1592/secCnt;
        Result=Result*60;
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x80);                  // Set LCD address 06H
        send_int_to_lcd(Result);
        }
}

/*****************************************************************************/
/********************** Function read data from DS1307 ***********************/
/*****************************************************************************/
unsigned char DS1307_rd(unsigned char addr)
{
        unsigned char ret;                      // For keep return value
        i2c_start();            // i2c start condition
```

```
        i2c_wrdata(DS1307_ID);      // Write DS1307 ID for connect
        i2c_wrdata(addr);                                   // Write RAM address on
DS1307 for connect

        i2c_start();                                        // i2c start condition
        i2c_wrdata(DS1307_ID+1);            // Write DS1307 ID for Read Mode
connect

        ret = i2c_rddata();         // Read data and keep to ret
  i2c_stop();                                               // i2c stop condition
  return(ret);                                          // return value
}
/*****************************************************************************/
/****************** Function write hour/min/sec on chip DS1307 ******************/
/*****************************************************************************/
void DS1307_wrtime()
{
        hour = (time_buf[1]<<4)|(time_buf[2]);                  // Convet input
hour value from key to Hex
        min = (time_buf[3]<<4)|(time_buf[4]);           // Convet input min value
from key to Hex
        sec = (time_buf[5]<<4)|(time_buf[6]);           // Convet input sec value
from key to Hex
        if(hour>0x23)                                       // Check hour over range
        hour = 0x23;            // IF hour over range reload maximum value
        if(min>0x59)                                        // Check min over range
        min = 0x59;                                             // IF min over
range reload maximum value
        if(sec>0x59)                                        // Check sec over range
        sec = 0x59;                                             // IF sec over
range reload maximum value
        i2c_start();            // i2c start condition
        i2c_wrdata(DS1307_ID);                      // Write DS1307 ID for connect
        i2c_wrdata(0x00);                                   // Write control byte to
access RAM address 00H
        i2c_wrdata(sec);                                    // Write sec on RAM
address 00H
        i2c_wrdata(min);                                    // Write min on RAM
address 01H
```

```
        i2c_wrdata(hour);                                   // Write hour on RAM
address 02H
        i2c_stop();             // i2c stop condition
}
/*****************************************************************************/
/********************** Function write date/month/year on chip DS1307 **************/
/*****************************************************************************/
void DS1307_wrdate()
{
        date = (date_buf[1]<<4)|(date_buf[2]);              // Convet input date value
from key to Hex
        month = (date_buf[3]<<4)|(date_buf[4]);             // Convet input month
value from key to Hex
        year = (date_buf[5]<<4)|(date_buf[6]);              // Convet input year value
from key to Hex
        if(year>0x99)                                       // Check year over range
        year = 0x99;                                        // IF year over range
reload maximum value
        if(month>0x12)                                      // Check month over range
        month = 0x12;                                       // IF month over range
reload maximum value
        if(date>0x31)                                       // Check date over range
        date = 0x31;                                        // IF date over range
reload maximum value
        i2c_start();                                        // i2c start condition
        i2c_wrdata(DS1307_ID);                              // Write DS1307 ID for connect
        i2c_wrdata(0x04);                                   // Write control byte to
access RAM address 04H
        i2c_wrdata(date);                                   // Write date on RAM
address 04H
        i2c_wrdata(month);                                  // Write month on RAM address
05H
        i2c_wrdata(year);                                   // Write year on RAM
address 06H
        i2c_stop();                                                 // i2c stop
condition
}
/*****************************************************************************/
```

```
/********************** Function display screen on
LCD*****************************/
/*****************************************************************************************/
void scr_datetime(void)
{
        unsigned char i;                        // Keep for couter loop
        /*-------------Show text screen on LCD("Date")----------------*/
        lcd_clear();                            // Clear LCD display
        lcd_jumporigin();       // Go to origin address
        lcd_command(0x0C);                      // None curcer and bring
        lcd_command(0x80);      // Set LCD address 00H
        for(i=0;i<12;i++)       // Loop for show screen "Date" by pointer _Date
        lcd_text(*(_Date+i));
        /*-------------------------------------------------------*/
        /*-------------Show text screen on LCD("Time")----------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC0);                      // Set LCD address 40H
        for(i=0;i<12;i++)                       // Loop for show screen "Time"
by pointer Time
        lcd_text(*(Time+i));
        /*-------------------------------------------------------*/
}
/********************** Display Normal(Page1) ********************************/
//--------------------- Date  dd/mm/yy--------------------------------//
//--------------------- Time  hh/mm/ss--------------------------------//
/*****************************************************************************/
void display_datetime(void)
{
                sec = DS1307_rd(0);             // Read sec before show on LCD
                delay_10ms(1);
                min = DS1307_rd(1);             // Read min before show on LCD
                delay_10ms(1);
                hour = DS1307_rd(2);            // Read hour before show on LCD
                delay_10ms(1);
                day = DS1307_rd(3);             // Read day before show on LCD
                delay_10ms(1);
                date = DS1307_rd(4);            // Read date before show on LCD
                delay_10ms(1);
                month = DS1307_rd(5);           // Read month before show on LCD
                delay_10ms(1);
```

```
        year = DS1307_rd(6);            // Read year before show on LCD

        /*-------------------------Show Date-------------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x88);              // Set LCD address 06H
        send_to_lcd(date);              // Write date show on LCD
        lcd_text('/');                          // Write "/" show on LCD
        send_to_lcd(month);             // Write month show on LCD
        lcd_text('/');                          // Write "/" show on LCD
        send_to_lcd(year);              // Write year show on LCD
        /*-----------------------------------------------------------*/

        /*-------------------------Show Time-------------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC8);              // Set LCD address 46H
        send_to_lcd(hour);              // Write hour show on LCD
        lcd_text(':');          // Write ":" show on LCD
        send_to_lcd(min);                       // Write min show on LCD
        lcd_text(':');                          // Write ":" show on LCD
        send_to_lcd(sec);                       // Write sec show on LCD
        /*-----------------------------------------------------------*/
}
/************************ Display Setting Date(Page2) ******************************/
//---------------------- Setting Date ----------------------------------------//
//---------------------- dd/mm/yy --------------------------------------------//
/***********************************************************************************/
void display_setdate(void)
{
        unsigned char i;                        // Keep for couter loop
        /*-----------Show text screen on LCD("Setting Date")----------------*/
        lcd_clear();                    // Clear LCD display
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x82);              // Set LCD address 02H
        for(i=0;i<12;i++)                       // Loop for show screen "Setting
Date" by pointer setdate
        lcd_text(*(setdate+i));
        /*-----------------------------------------------------------*/
        /*-------------Show text screen on LCD("dd/mm/yy")-------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0F);              // Display on and currer bring
        lcd_command(0xC3);              // Set LCD address 43H
        for(i=0;i<8;i++)                        // Loop for show screen
"dd/mm/yy" by pointer dmy
        lcd_text(*(dmy+i));
```

```
        /*-------------------------------------------------------------*/
        /*----------- Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0F);                  // Set display ON and curcer bring
        lcd_command(0xC3);                  // Set LCD address 43H
        /*-------------------------------------------------------------*/
}
/***********************Display Setting Time(Page3)*****************************/
//---------------------- Setting Time ---------------------------------------//
//---------------------- hh/mm/ss -------------------------------------------//
/*******************************************************************************/
void display_settime(void)
{
        unsigned char i;                        // Keep for couter loop
        /*-------------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                            // Clear LCD display
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x82);                  // Set LCD address 02H
        for(i=0;i<12;i++)                       // Loop for show screen "Setting
Time" by pointer settime
        lcd_text(*(settime+i));
        /*-------------------------------------------------------------*/
        /*-------------Show text screen on LCD("hh/mm/ss")--------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC3);                  // Set LCD address 43H
        for(i=0;i<8;i++)                        // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(hms+i));
        /*-------------------------------------------------------------*/
        /*----------- Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0F);                  // Display ON and OFF curcer bring
        lcd_command(0xC3);                  // Set LCD address 43H
        /*-------------------------------------------------------------*/
}


/*******************************************************************************/
/********************** Function  Display Text "VIEW DATA"
***************************/
```

```
/*****************************************************************************/


void displayMenuViewData(void)
{
        unsigned char i;                                // Keep for couter loop
        /*----------Show text screen on LCD("  VIEW DATA    ")----------*/
        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                  // Set LCD address 80H
        for(i=0;i<16;i++)                               // Loop for show screen
        lcd_text(*(msg_ViewData+i));
        /*------------------------------------------------------------*/

        lcd_jumporigin();                               // Go to origin address
        lcd_command(0xC0);                  // Set LCD address 43H
        for(i=0;i<16;i++)                               // Loop for show screen
        lcd_text(*(msg_Upload+i));


        lcd_command(0x80);                  // Set LCD address 43H
        lcd_text('>');
        /*------------------------------------------------------------*/
        /*---------- Return to set curcer bring for insert value ----------*/
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x0C);                  //  OFF curcer
        /*------------------------------------------------------------*/
}


/*****************************************************************************/
/********************** Function Display "SETTING TIME "*****************/
/*****************************************************************************/
void displayMenuSetTime(void)
{
        unsigned char i;                                // Keep for couter loop

        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                  // Set LCD address 02H
        for(i=0;i<16;i++)                               // Loop for show screen
        lcd_text(*(msg_MenuSetTime+i));
```

```
        /*-------------------------------------------------------------*/

        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC0);                  // Set LCD address C0H
        for(i=0;i<16;i++)                       // Loop for show screen
        lcd_text(*(msg_MenuSetDate+i));

        lcd_command(0x80);                  // Set LCD address 80H
        lcd_text('>');
        /*-------------------------------------------------------------*/
        /*----------- Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0C);                  //  OFF curcer
        /*-------------------------------------------------------------*/
}

/*****************************************************************************/
/********************** Function Display "SETTING DATE "*********************/
/*****************************************************************************/

void displayMenuSetDate(void)
{
        unsigned char i;                        // Keep for couter loop

        lcd_clear();                            // Clear LCD display
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x80);                  // Set LCD address 80H
        for(i=0;i<16;i++)                       // Loop for show screen
        lcd_text(*(msg_MenuSetDate+i));
        /*-------------------------------------------------------------*/
        /*-------------Show text screen on LCD("hh/mm/ss")--------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC0);                  // Set LCD address C0H
        for(i=0;i<16;i++)                       // Loop for show screen
        lcd_text(*(msg_ViewData+i));

                lcd_command(0x80);              // Set LCD address 80H
        lcd_text('>');
        /*-------------------------------------------------------------*/
        /*----------- Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                       // Go to origin address
```

```
        lcd_command(0x0C);                  // OFF curcer
        /*-------------------------------------------------------------*/
}

/*****************************************************************************/
/********************** Function Display "CLEAR MEMORY"*******************/
/*****************************************************************************/

void displayMenuClrMem(void)
{
        unsigned char i;                                // Keep for couter loop
        /*-----------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                  // Set LCD address 02H
        for(i=0;i<16;i++)                               // Loop for show screen
        lcd_text(*(msg_MenuClrMem+i));
        /*-------------------------------------------------------------*/
        /*-----------Show text screen on LCD("hh/mm/ss")----------------*/
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0xC0);                  // Set LCD address 43H
        for(i=0;i<16;i++)                               // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(msg_MenuSetTime+i));

        lcd_command(0x80);                  // Set LCD address 43H
        lcd_text('>');
        /*-------------------------------------------------------------*/
        /*---------- Return to set curcer bring for insert value ----------*/
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x0C);                  // OFF curcer
        /*-------------------------------------------------------------*/
}


/*****************************************************************************/
/********************** Function Display "UPLOAD  DATA"*******************/
/*****************************************************************************/

void displayMenuUpload(void)
{
        unsigned char i;                                // Keep for couter loop
```

```
        /*-------------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                      // Set LCD address 02H
        for(i=0;i<16;i++)                               // Loop for show screen "Setting
Time" by pointer settime
        lcd_text(*(msg_Upload+i));
        /*-------------------------------------------------------------------*/
        /*-------------Show text screen on LCD("hh/mm/ss")--------------------*/
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0xC0);                      // Set LCD address 43H
        for(i=0;i<16;i++)                               // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(msg_MenuClrMem+i));

                lcd_command(0x80);                      // Set LCD address 43H
        lcd_text('>');
        /*-------------------------------------------------------------------*/
        /*----------- Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x0C);                      //  OFF curcer
        /*-------------------------------------------------------------------*/
}

/*****************************************************************************************/
/********************** Function Display "SETUP    LOADDING...
"*****************/
/*****************************************************************************************/

void display_EnterSetup(void)
{
        unsigned char i;                                // Keep for couter loop
        /*-------------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                      // Set LCD address 02H
        for(i=0;i<16;i++)                               // Loop for show screen "Setting
Time" by pointer settime
        lcd_text(*(msg_MenuSetup+i));
        /*-------------------------------------------------------------------*/
        /*-------------Show text screen on LCD("hh/mm/ss")--------------------*/
        lcd_jumporigin();                               // Go to origin address
```

```
        lcd_command(0xC0);                  // Set LCD address 43H
        for(i=0;i<16;i++)                   // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(msg_MenuLoading+i));

        lcd_command(0x80);                  // Set LCD address 43H
        lcd_text('>');
        /*-------------------------------------------------------------*/
        /*---------- Return to set curcer bring for insert value ----------*/
        lcd_jumporigin();                   // Go to origin address
        lcd_command(0x0C);                  //  OFF curcer
        /*-------------------------------------------------------------*/
}

/*****************************************************************************/
/********************** Function Set Date  for chip DS1307*******************/
/*****************************************************************************/

void mySetDate(void)
{
                                    display_setdate();          // Call function
display page2
                                    while(x_key < 6)            // Check for end
insert setting Date value 6 time(dd/mm/yy)
                                    {
                                        push_key = scankey();
                        // Scankey and Keep to push_key
                                        if(push_key != 0xFF)        // Check
value key for push?
                                        {
                                            x_key++;                //
increase counter 1 time
                                            date_buf[x_key] = push_key;
        // Keep date setting for write to ds1307

                                            // date_buf[1]date_buf[2]=>dd

                                            // date_buf[3]date_buf[4]=>mm

                                            // date_buf[5]date_buf[6]=>yy
```

```
                                        push_key = push_key | 0x30;
        // convert to ascii code
                                        lcd_text(push_key);          //
send value input key on LCD


                                        if(x_key==2)            // IF push
key 2 time shift curcer 1 time
                                        {
                                                lcd_jumporigin();
                                                lcd_command(0xC6);

                                        }
                                        if(x_key==4)
                        // IF push key 4 time shift curcer 1 time
                                        {
                                                lcd_jumporigin();
                                                lcd_command(0xC9);
                                        }
                                }
                        }
                        DS1307_wrdate();
                // Write Date data to DS1307
                        x_key = 0;
                        // Reload counter
                        scr_nomal = 1;
                        page = 1;
                        // For next time to return page1(show real time)

}


/*****************************************************************************/
/********************** Function Set Time for chip DS1307*****************/
/*****************************************************************************/

void mySettime(void)
{
        display_settime();
```

```
                    while(x_key < 6)
                // Check for end insert setting Date value 6
time(hh/mm/ss)
                    {
                        push_key = scankey();
            // Scankey and Keep to push_key
                        if(push_key != 0xFF)
            // Check value key for push?
                        {
                            x_key++;
                        // increase counter 1 time
                            time_buf[x_key] = push_key;
    // Keep date setting for write to ds1307

                        // date_buf[1]date_buf[2]==>dd

                        // date_buf[3]date_buf[4]==>mm

                            // date_buf[5]date_buf[6]==>yy
                            push_key = push_key | 0x30;
    // convert to ascii code
                            lcd_text(push_key);          //
send to LCD

                            if(x_key==2)
                        // IF push key 2 time shift curcer 1 time
                            {

                                lcd_jumporigin();
                                lcd_command(0xC6);
                            }
                            if(x_key==4)
                        // IF push key 4 time shift curcer 1 time
                            {
                                lcd_jumporigin();
                                lcd_command(0xC9);
                            }
                        }
                    }
                    DS1307_wrtime();
                // Write Time data to DS1307
```

```
                                x_key = 0;
                                // Reload counter
                                page = 1;
                                // For next time to return page1(show real time)
                                scr_nomal = 1;

}

/*****************************************************************************/
/********************** Function Read data from 24LC515*****************/
/*****************************************************************************/


unsigned char AT24C16_rd(unsigned char xhi,unsigned char xlo)
{
        unsigned char dat = 0;          // For keep return value from read in EEPROM
        err = 0;        // Clear status bit flag Error
        /*-----------------------------Convert address from serial port ----------------------------
-----*/

        /*-------------------------------------------------------------------------------------------*/
        i2c_start();                    // i2c start condition
        if(i2c_wrdata(0xA0))            // Write AT24c16 ID with address byte 1 for
connect
        {
                i2c_stop();             // If connect error stop
                err = 1;                // Set bit flag for Alarm by Post text
        }
        i2c_wrdata(xhi);                // Write address Hi byte
        i2c_wrdata(xlo);                // Write address low byte
        i2c_start();                    // i2c start condition
        i2c_wrdata(0xA1);       // Write AT24C16 ID for Read Mode connect
        dat = i2c_rddata();             // Read data and keep to dat
        i2c_NACK();                     // Send signal for stop read
        i2c_stop();                             // i2c stop condition
        delay_10ms(1);
        return(dat);                    // return value
}

/*****************************************************************************/
```

```
/************************ Function write data to 24LC515*****************/
/*************************************************************************************/

void AT24C16_wr(unsigned char xhi,unsigned char xlo,unsigned char dat)

{
//      unsigned char addr_h,addr_l,addr_l2;// For keep between convert address
        err = 0;                    // Clear status bit flag Error
        /*-----------------------------------Convert address from serial port ----------------------
---------*/

        /*-------------------------------------------------------------------------------------------*/
        i2c_start();                        // i2c start condition
        if(i2c_wrdata(0xA0))                // Write AT24c16 ID with address byte 1 for
connect
        {
                i2c_stop();                 // If connect error stop
                err = 1;                    // Set bit flag for Alarm by Post text
        }
        i2c_wrdata(xhi);                    // Write address Hi byte
        i2c_wrdata(xlo);                    // Write address low byte
        i2c_wrdata(dat);                            // Write data in EEPROM
        i2c_stop();                         // i2c stop condition
        delay_10ms(1);
}

/*************************************************************************************/
/************************ Function increase address for index
memory*****************/
/*************************************************************************************/

void inc_lastaddr(void)
{
        if(last_addr_hi < 0xE0)
        {
        last_addr_lo++;
        if(last_addr_lo==0x00)last_addr_hi++;
        }
}
```

```
/*****************************************************************************/
/********************** Function increase address for index
memory*****************/
/*****************************************************************************/

void inc_curaddr(void)
{
        if(addr_hi < 0xE0)
        {
                addr_lo++;
        if(addr_lo==0x00)addr_hi++;
        }
}


void dec_curaddr(unsigned char dec_cnt)
{

while(dec_cnt)
{
        if(addr_lo==0x00 && addr_hi > 0x00)
        {
        addr_lo--;
        addr_hi--;
        }
        else if(addr_lo > 0x00)
        {
        addr_lo--;
        }
        dec_cnt--;
}

}

/*****************************************************************************/
/********************** Function show data in memory*****************/
/*****************************************************************************/

void myViewData(void)
{
                int addr_cnt=0;
                char addr_tmp_hi,addr_tmp_lo;
        //      char dat_tmp;
                int RecCnt;
```

```
//              int RecCntTmp;
                addr_hi=reg_addr;
                addr_lo=DEF_addr;

        RecCnt=1;
        lcd_clear();                                    // Clear LCD display
        while(1)
        {

                unsigned char i;                                // Keep for couter loop


        lcd_jumporigin();
        lcd_command(0x80);
        for(i=0;i<16;i++)
        lcd_text(*(msg_Review+i));
        addr_cnt=0;


                while(AT24C16_rd(addr_hi,addr_lo)!=0xAB)
                {

                        addr_cnt++;
                        if(addr_cnt>5)
                        {
                        addr_hi=addr_tmp_hi;
                        addr_lo=addr_tmp_lo;

                        break;
                        }
                        inc_curaddr();
                }

                addr_tmp_hi=addr_hi;
                addr_tmp_lo=addr_lo;


                inc_curaddr();
                date = AT24C16_rd(addr_hi,addr_lo);

                inc_curaddr();
                month = AT24C16_rd(addr_hi,addr_lo);

                inc_curaddr();
                year = AT24C16_rd(addr_hi,addr_lo);
```

```
inc_curaddr();
hour = AT24C16_rd(addr_hi,addr_lo);

inc_curaddr();
min = AT24C16_rd(addr_hi,addr_lo);

inc_curaddr();
sec = AT24C16_rd(addr_hi,addr_lo);


/*----------Show Date----------*/
lcd_jumporigin();                    // Go to origin address
lcd_command(0x88);                   // Set LCD address 06H
send_to_lcd(date);                   // Write date show on LCD
lcd_text('/');                       // Write "/" show on LCD
send_to_lcd(month);                  // Write month show on LCD
lcd_text('/');                       // Write "/" show on LCD
send_to_lcd(year);                   // Write year show on LCD
/*------------------------------------------------------------*/
/*----------Show Time----------*/
lcd_jumporigin();                    // Go to origin address
lcd_command(0xC8);                   // Set LCD address 46H
send_to_lcd(hour);                   // Write hour show on LCD
lcd_text(':');          // Write ":" show on LCD
send_to_lcd(min);                    // Write min show on LCD
lcd_text(':');                       // Write ":" show on LCD
send_to_lcd(sec);                    // Write sec show on LCD

inc_curaddr();
date = AT24C16_rd(addr_hi,addr_lo); // Dig5
inc_curaddr();

month = AT24C16_rd(addr_hi,addr_lo);// Dig4

inc_curaddr();
year = AT24C16_rd(addr_hi,addr_lo);// Dig3

inc_curaddr();
hour = AT24C16_rd(addr_hi,addr_lo);// Dig2

inc_curaddr();
min = AT24C16_rd(addr_hi,addr_lo);// Dig1

lcd_jumporigin();
```

```
        lcd_command(0xC0);


        lcd_text(date|0x30);
        lcd_text(month|0x30);
        lcd_text(year|0x30);
        lcd_text(hour|0x30);
        lcd_text(min|0x30);
        lcd_text('u');
        lcd_text('M');


delay_50ms(1);
                while(1)
                {
                if(sw_ESC==0)break;
                if(sw_ENT==0)
                        {
                        inc_curaddr();
                        break;
                        }
                if(sw_MENU==0 )
                        {
//                      addr_lo=addr_lo-15;
                        dec_curaddr(25);
                        break;
                        }

                }
                        if(sw_ESC==0)break;
        }

}


/*****************************************************************************/
/********************** Function Clear  memory*****************/
/*****************************************************************************/

void myClrData(void)
{
unsigned xaddr_hi,xaddr_lo;
```

```
unsigned char i,my_Cnt1;                        // Keep for couter loop
        /*-------------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                            // Clear LCD display
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x80);                  // Set LCD address 02H
        for(i=0;i<16;i++)                       // Loop for show screen "Setting
Time" by pointer settime
        lcd_text(*(msg_ClearLog+i));
        /*-------------------------------------------------------------------*/
        /*-------------Show text screen on LCD("hh/mm/ss")-------------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC0);                  // Set LCD address 43H
        for(i=0;i<16;i++)                       // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(msg_YesNo+i));


        /*-------------------------------------------------------------------*/
        /*------------ Return to set curcer bring for insert value -----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0C);                  //  OFF curcer
        /*-------------------------------------------------------------------*/
        delay_100ms(10);
        while(1)
        {

        if(sw_ESC==0)break;
        if(sw_ENT==0)
                {
                        xaddr_hi=0x00;
                        xaddr_lo=DEF_addr;


                        AT24C16_wr(xaddr_hi,xaddr_lo-14,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-13,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-12,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-11,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-10,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-9,0xCC);
                        AT24C16_wr(xaddr_hi,xaddr_lo-8,0xCC);
```

```
                AT24C16_wr(xaddr_hi,xaddr_lo-7,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-6,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-5,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-4,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-3,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-2,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo-1,0xCC);
                AT24C16_wr(xaddr_hi,xaddr_lo,0xCC);

                AT24C16_wr(xaddr_hi,xaddr_lo+1,0xAB);


                for(my_Cnt1=2;my_Cnt1<=27;my_Cnt1++)
                {
                AT24C16_wr(xaddr_hi,xaddr_lo+my_Cnt1,0xFF);

                }

                addr_hi=dat_cnt_addhi;
                addr_lo=dat_cnt_addlo;
                AT24C16_wr(addr_hi,addr_lo,0x00);

                addr_hi=dat_cnt_addhi;
                addr_lo=dat_cnt_addlo+1;
                AT24C16_wr(addr_hi,addr_lo,0x00);
//*************************************************

                AT24C16_wr(0x00,0x02,0x00);
                AT24C16_wr(0x00,0x03,DEF_addr);


        lcd_clear();                    // Clear LCD display
    lcd_jumporigin();                   // Go to origin address
    lcd_command(0x80);                  // Set LCD address 02H
    for(i=0;i<16;i++)                   // Loop for show screen "Setting
Time" by pointer settime
    lcd_text(*(msg_Done+i));

    lcd_jumporigin();                   // Go to origin address
    lcd_command(0xC0);
    for(i=0;i<16;i++)
```

```
        lcd_text(*(msg_MemClear+i));

        /*-------------------------------------------------------------*/
        /*---------- Return to set curcer bring for insert value ----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0C);                  //  OFF curcer
        /*-------------------------------------------------------------*/
        delay_100ms(10);
        break;

                }

        }

}


/*****************************************************************************************/
/********************** Function read the last memory reccord address
*****************/
/*****************************************************************************************/

void get_lastAddr(void)
{
        last_addr_hi = AT24C16_rd(0x00,0x02);
        last_addr_lo = AT24C16_rd(0x00,0x03);
}


/*****************************************************************************************/
/********************** Function reccord last memory address to
EEPROM*****************/
/*****************************************************************************************/

void set_lastAddr(void)
{
        AT24C16_wr(0x00,0x02,last_addr_hi);
        AT24C16_wr(0x00,0x03,last_addr_lo);
}
```

```
void Pulse_LED(void)
{
        LED=0;
        delay_100ms(3);
        LED=1;
}

/*****************************************************************************/
/********************** Function send data to serial port*****************/
/*****************************************************************************/

void tx_serial(unsigned char dat)
{
        unsigned char buff = 0;                 // Buffer for process convert
        /*-----------------------------------Convert high byte-----------------------------------
-----*/
        buff = dat & 0xF0;                      // Filter dat high byte
        buff = buff>>4;                         // Shift 4 time to low byte
        if(buff<=0x09)          // Check value if <= 9 work in block
        {
                buff =  buff | 0x30;    // Convert to ascii code between "0" to "9"
                printf("%c",buff);      // Post text to Terminal
        }
        else                                    // If > 9 con work in block
        {
                buff = (buff-9) | (0x40); // Convert to ascii code between "a" to
"f"(assume)
                printf("%c",buff);      // Post text to Terminal
        }
        /*-----------------------------------------------------------------------------------*/
        /*-----------------------------------Convert low byte-----------------------------------
----*/
        buff = 0;                               // Clear buffer for new convert
        buff = dat & 0x0F;      // Filter dat low byte
        if(buff<=0x09)          // Check value if <= 9 work in block
        {
                buff =  buff | 0x30;    // Convert to ascii code between "0" to "9"
                printf("%cH\n",buff); // Post text to Terminal
        }
        else                                    // If > 9 con work in block
        {
```

```
                buff = (buff-9) | 0x40;  // Convert to ascii code between "a" to
"f"(assume)
                printf("%cH\n",buff); // Post text to Terminal
        }
        /*-------------------------------------------------------------------------------------------------*/
}

/*****************************************************************************************/
/********************** Function reccord data into memory*****************/
/*****************************************************************************************/
void Rec_Dat(void)
{
                char my_Cnt1;

//          TrigCount++;

                Pulse_LED();
                RainCaculate();
                secCnt=0;
                get_lastAddr();

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,0xAB);  // Add start byte

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,date);

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,month);

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,year);

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,hour);

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,min);

                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,sec);


                inc_lastaddr();
```

```
            AT24C16_wr(last_addr_hi,last_addr_lo,dig5);

            inc_lastaddr();
            AT24C16_wr(last_addr_hi,last_addr_lo,dig4);

            inc_lastaddr();
            AT24C16_wr(last_addr_hi,last_addr_lo,dig3);

            inc_lastaddr();
            AT24C16_wr(last_addr_hi,last_addr_lo,dig2);

            inc_lastaddr();
            AT24C16_wr(last_addr_hi,last_addr_lo,dig1);



            set_lastAddr();

            for(my_Cnt1=0;my_Cnt1<=8;my_Cnt1++)
                {
                inc_lastaddr();
                AT24C16_wr(last_addr_hi,last_addr_lo,0xFF);
                }

}

/*****************************************************************************/
/********************** Function display "Wait for connect"*****************/
/*****************************************************************************/

void disp_Wait_connect(void)
{
        unsigned char ii;

                                        // Keep for couter loop
        /*----------Show text screen on LCD("Setting Time")----------------*/
        lcd_clear();                                    // Clear LCD display
        lcd_jumporigin();                               // Go to origin address
        lcd_command(0x80);                  // Set LCD address 02H
```

```
        for(ii=0;ii<16;ii++)                    // Loop for show screen "Setting
Time" by pointer settime
        lcd_text(*(msg_PC_connect+ii));
        /*-------------------------------------------------------------*/
        /*-----------Show text screen on LCD("hh/mm/ss")-----------------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0xC0);                  // Set LCD address 43H
        for(ii=0;ii<16;ii++)                    // Loop for show screen
"hh/mm/ss" by pointer hms
        lcd_text(*(msg_Connect+ii));

        lcd_command(0x80);                  // Set LCD address 43H
        lcd_text('>');
        /*-------------------------------------------------------------*/
        /*---------- Return to set curcer bring for insert value ----------*/
        lcd_jumporigin();                       // Go to origin address
        lcd_command(0x0C);                  // OFF curcer
        /*-------------------------------------------------------------*/


}

/*****************************************************************************/
/********************** Function upload data to PC*****************/
/*****************************************************************************/

void myUpData(void)
{

//      char ss;
        int i;
        char cmdbuf [10];

        int addr_cnt=0;
        char dat_tmp;
        unsigned char ii;

        addr_hi=reg_addr;
        addr_lo=DEF_addr;


disp_Wait_connect();
```

```
while(1)
{
  gets (cmdbuf, sizeof (cmdbuf));    //Read RS232 to char array
                    for (i = 0; i < sizeof (cmdbuf); i++)
                    {
              cmdbuf[i] = toupper(cmdbuf[i]);      /* Upper Chars */
              }
                    if(strcmp(cmdbuf,"START")==0)
                    {

                    addr_hi=reg_addr;
                    addr_lo=DEF_addr;

                    // Keep for couter loop
                    /*----------Show text screen on LCD("Setting Time")----
----------*/
                    lcd_clear();                              // Clear
LCD display
                    lcd_jumporigin();                         // Go to
origin address
                    lcd_command(0x80);                  // Set LCD address
02H
                    for(ii=0;ii<16;ii++)                      // Loop for
show screen "Setting Time" by pointer settime
                    lcd_text(*(msg_PC_connect+ii));
                    /*-------------------------------------------------------*/
                    /*----------Show text screen on LCD("hh/mm/ss")--------
----------*/
                    lcd_jumporigin();                         // Go to
origin address
                    lcd_command(0xC0);                  // Set LCD address
43H
                    for(ii=0;ii<16;ii++)                      // Loop for
show screen "hh/mm/ss" by pointer hms
                    lcd_text(*(msg_Upload_dat+ii));

                    /*-------------------------------------------------------*/

                    /*---------- Return to set curcer bring for insert value -----
------*/
                    lcd_jumporigin();                         // Go to
origin address
                    lcd_command(0x0C);                  //  OFF curcer
```

```
                    /*-------------------------------------------------------------*/

                        printf("READY\n");


                        }

                        if(strcmp(cmdbuf,"R")==0)
                                {
                                LED=0;
                                inc_curaddr();
                                dat_tmp = AT24C16_rd(addr_hi,addr_lo);
                        //       myDebug();
                                tx_serial(dat_tmp);
                                LED=1;
                                }

                                if(strcmp(cmdbuf,"STOP")==0)
                                {
                                break;
                                }

                }
}

/*****************************************************************************/
/********************** Function assign start value ***************************/
/*****************************************************************************/

void system_init(void)
{
 SCON  = 0x50;              /* SCON: mode 1, 8-bit UART, enable rcvr   */
 TMOD |= 0x20;              /* TMOD: timer 1, mode 2, 8-bit reload     */
 TH1   = 0xfd;             /* TH1:  reload value for 2400 baud        */
 TR1   = 1;                /* TR1:  timer 1 run                       */
 TI    = 1;

  sw_MENU = 1;
  sw_ESC = 1;
```

```
  sw_ENT = 1;
        get_lastAddr();

}


/*****************************************************************************************/
/*********************** Main function  *****************/
/*****************************************************************************************/

void main(void)                                         // Main loop
{
        bit senTmp;
        bit senTmp2;
                senTmp=sensor;
                senTmp2=sensor2;

                secCnt=0;

        system_init();
        lcd_init();                 // Initial set LCD display


        while(1)                // Infinite loop
        {

//      rainCaculate();

/*      if unsigned char secOld,secNew;
unsigned int secCnt;
*/
        if(secOld != sec)
        {
        secOld = sec;
        secCnt++;
        }

         if(sw_ENT==0 && sensor==1 && sensor2==1)
        // IF switch Setting Date for push then display page2
          {
                Rec_Dat();

                }
```

```
            if(sensor!=senTmp && sensor2 != senTmp2 )
        // IF switch Setting Date for push then display page2
          {

              Rec_Dat();
              senTmp=sensor;
              senTmp2=sensor2;
              }


          if(sw_MENU==0)                                  // IF switch Setting Date
for push then display page2
          page = 4;              // Load page value = 2(for display page2)

        /*----------Check condition display page1,page2 or page3----------*/
          switch(page)                        // Check display page
          {
              case 0 : break;          // IF page = 0 out(because this time show page2 or
page3 )

              case 1 : if(scr_nomal)
                                  {
                                      scr_datetime();
                                      lcd_jumporigin();
                                      lcd_command(0x80);
                                      DisplayRain();
                                      scr_nomal = 0;
                                  }
                                  display_datetime();
                        // Call function display page1(Real Date and Real Time)
                                  break;
                                  // Out
                                      // Out


              case 4 : display_EnterSetup();

                                  while(sw_MENU==0);
                                  delay_100ms(10);
                                  subMenu=1;
                                  while(1)
                        // Check for end insert setting Date value 6 time(hh/mm/ss)
                                  {
```

```
                if(sw_ESC==0)break;

                if(sw_MENU==0)
            {
                    subMenu+=1;
                    if(subMenu>5)subMenu=1;
                }

                if(subMenu==1)displayMenuSetTime();
                if(subMenu==2)displayMenuSetDate();

if(subMenu==3)displayMenuViewData();
                if(subMenu==4)displayMenuUpload();
                if(subMenu==5)displayMenuClrMem();

                delay_100ms(5);
                while(sw_MENU==1)
                {
                if(sw_ENT==0 && subMenu==2 )
                {
                    mySetDate();
                    displayMenuSetDate();
                    delay_1s(1);
                }

                if(sw_ENT==0 && subMenu==1 )
                {
                    mySettime();
                    displayMenuSetTime();
                    delay_1s(1);
                }

                if(sw_ENT==0 && subMenu==3 )
                {
                    myViewData();
                    displayMenuViewData();
                    delay_1s(1);
                }

                if(sw_ENT==0 && subMenu==5 )
                {
                    myClrData();
                    displayMenuClrMem();
```

```
                delay_1s(1);
            }

                if(sw_ENT==0 && subMenu==4
)
            {
                myUpData();
                displayMenuUpload();
                delay_1s(1);
            }

            if(sw_ESC==0)break;
            };

            }
            page = 1;
            // For next time to return page1(show real time)
            scr_nomal = 1;
            break;
            // Out

        }
        /*-------------------------------------------------------*/
    }
}
```

ภาคผนวก ข

ส่วนของFile Include I^2C

```
#include<intrins.h>
//sfr  P5   = 0xE8;
//sfr  P4   = 0xC0;
sbit SDA = P1^0;
sbit SCL = P1^1;
void i2c_delay(void)
{
        unsigned char i;
        for(i=0;i<50;i++)
        _nop_();
}
void i2c_clk(void)
{
        i2c_delay();
        SCL = 1;
        i2c_delay();
        SCL = 0;
}
void i2c_start(void)
{
        if(SCL)
        SCL = 0;
        SDA = 1;
        SCL = 1;
        i2c_delay();
        SDA = 0;
        i2c_delay();
        SCL = 0;
}
void i2c_stop(void)
{
        if(SCL)
        SCL = 0;
        SDA = 0;
        i2c_delay();
        SCL = 1;
        i2c_delay();
        SDA = 1;
}
bit i2c_wrdata(unsigned char dat)
{
        bit data_bit;
```

```
        unsigned char i;
        for(i=0;i<8;i++)
        {
                data_bit = dat & 0x80;
                SDA = data_bit;
                i2c_clk();
                dat = dat<<1;
        }
        SDA = 1;
        i2c_delay();
        SCL = 1;
        i2c_delay();
        data_bit = SDA;
        SCL = 0;
        i2c_delay();
        return(data_bit);                       // if send_bit = 0 i2c OK!
}
unsigned char i2c_rddata(void)
{
        bit rd_bit;
        unsigned char i,dat;
        dat = 0x00;
        for(i=0;i<8;i++)
        {
                i2c_delay();
                SCL = 1;
                i2c_delay();
                rd_bit = SDA;
                dat = dat<<1;
                dat = dat | rd_bit;
                SCL = 0;
        }
        SDA = 1;
        i2c_delay();
        i2c_clk();
        SCL = 1;
        return(dat);
}

void i2c_NACK()
{
        SDA = 1;
        i2c_delay();
        i2c_clk();
        SCL = 1;
}
```

ส่วนของFile Include SCANKEY

```
sbit c1 = P2^0;        // Bit Column
sbit c2 = P2^1;              // Bit Column2
sbit c3 = P2^2;              // Bit Column3
sbit r1 = P2^3;         // Bit Row1
sbit r2 = P2^4;         // Bit Row2
sbit r3 = P2^5;         // Bit Row3
sbit r4 = P2^6;         // Bit Row4

void delay_db(int time)
{
        do
        {
         time--;
        }while(time>0);

}
unsigned char scankey(void)    // function scankey 4x3
{
 unsigned char ret = 0xFF;
        c1 = 0;                          // scan colum1
        if(r1==0)                                                    //check push key 1
        {
                delay_db(30000);
                ret = 0x01;
        }               // send index = 1
        if(r2==0)                                                    //check push key 4
        {
                delay_db(30000);
                ret = 0x04;
        }                                                    // send index = 4
        if(r3==0)
        {                                                    //check push key 7
                delay_db(30000);
                ret = 0x07;                                                  // send
index = 7
        }

        c1 = 1;                 // stop check colum1

        c2 = 0;                                                      // scan colum2
        if(r1==0)
        {
                delay_db(30000);
        //check push key 2
                ret = 0x02;
        }                                                    // send index = 2
        if(r2==0)                                                    //check push key 7
```

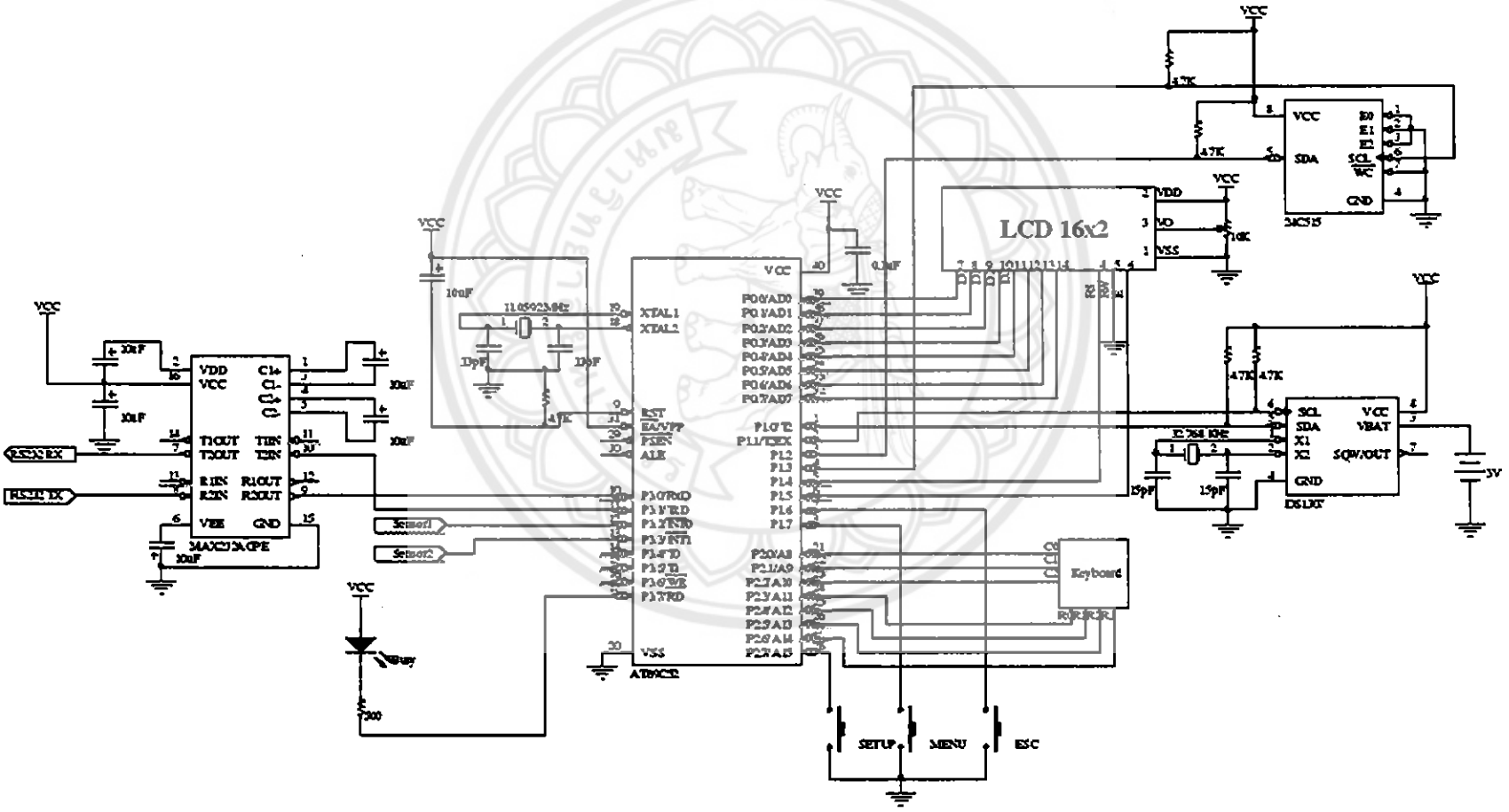
```
        {
                delay_db(30000);
                ret = 0x05;
        }                                               // send index = 5
        if(r3==0)                                               //check push key 8
        {
                delay_db(30000);
                ret = 0x08;
        }                                               // send index = 8
        if(r4==0)                                               // check push key
0
        {
                delay_db(30000);
                ret = 0x00;
        }                                               // send index = 0
        c2 = 1;                                                 // stop check
colum2

        c3 = 0;                                                 // scan colum3
        if(r1==0)
        {
                delay_db(30000);
        //check push key 3
                ret = 0x03;
        }                                               // send index = 3
        if(r2==0)                                               //check push key 6
        {
                delay_db(30000);
                ret = 0x06;
        }                                               // send index = 6
        if(r3==0)                                               //check push key 9
        {
                delay_db(30000);
                ret = 0x09;
        }
                                        // send index = 9
        c3 = 1;                                                 // stop check
colum3
        return(ret);
}
```

ส่วนของFile Include LCD

```
sbit e =P1^5;
sbit rs = P1^4;
void delay(int tick)
{
        unsigned int i,j;
        for(i=0;i<tick;i++)
        for(j=0;j<250;j++);
}
void lcd_command(unsigned char com)       //write command
{
        rs = 0;
        e = 1;
        P0 = com;
        //delay(10);   //Defult
        delay(1);
        e = 0;
        delay(1);
        //delay(10);   //Defult
}
void lcd_text(unsigned char text)      //write command
{
        rs = 1;
        e = 1;
        P0 = text;
        delay(1);
        //delay(10);   //Defult
        e = 0;
        delay(1);
        //delay(10);   //Defult
}
void lcd_clear()
{
        lcd_command(0x01);
}
void lcd_jumporigin()
{
        lcd_command(0x02);
}
void lcd_init()
{
        delay(250);
        delay(250);
        lcd_command(0x38);      //on display ,8 bit display ,5*7 dot
        lcd_command(0x0C);      //none curser
        lcd_command(0x01);          //clear screen
}
```

ภาคผนวก ง

บอร์ดวงจรเครื่องวัดปริมาณน้ำฝนแบบเวลาจริง



ประวัติผู้เขียนโครงงาน

ชื่อ นายรักษพล ศรีนุเสน

เกิดวันที่ 6 ตุลาคม 2525

ภูมิลำเนา 197/2 ถนนปัทมานนท์ ต.ในเมือง อ.เมือง จ.ร้อยเอ็ด

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนอนุบาลร้อยเอ็ด
- จบระดับมัธยมศึกษาจากโรงเรียนร้อยเอ็ดวิทยาลัย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 5
  สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E -Mail : ron_sa_pak@hotmail.com

ชื่อ นายปัญญา สุธงษา

เกิดวันที่ 6 มิถุนายน 2525

ภูมิลำเนา 83/28 ม.8 ต.หัวรอ อ.เมือง จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนสะพานที่3
- จบระดับมัธยมศึกษาจากโรงเรียนพิษณุโลกศึกษา
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 5
  สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-Mail : HADTAMUKAY@hotmail.com