การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ II

PROCESSING UNIT AND WARNING SYSTEM OVER MOBILE PHONE II

นายสุรพันธ์ อาดูร รหัส 49362321

นางสาวชมพูนุช วงศ์เกิดนิมิต รหัส 49363991

นายรุ่งโรจน์ สงวนวัฒนา รหัส 49364233

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2552

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ II

ผู้ดำเนินโครงงาน นายสุรพันธ์ อาดูร รหัส 49362321

นางสาวชมพูนุช วงศ์เกิดนิมิต รหัส 49363991

นายรุ่งโรจน์ สงวนวัฒนา รหัส 49364233

ที่ปรึกษาโครงงาน ผู้ช่วยศาสตราจารย์ ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2552

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน

(ผศ. ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง)

.....กรรมการ

(ดร. มุฑิตา สงฆ์จันทร์)

.....กรรมการ

(ดร. นิพัทธ์ จันทรมินทร์)

ชื่อหัวข้อโครงงาน	การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ II
ผู้ดำเนินโครงงาน	นายสุรพันธ์ อาดูร รหัส 49362321
	นางสาวชมพูนุช วงศ์เกิดนิมิต รหัส 49363991
	นายรุ่งโรจน์ สงวนวัฒนา รหัส 49364233
ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2552

.....

บทคัดย่อ

โครงงานชิ้นนี้เกิดขึ้น เนื่องจากความต้องการที่จะพัฒนาระบบที่สามารถแจ้งเตือนภัยล่วงหน้าผ่านสัญญาณโทรศัพท์มือถือ ก่อนเกิดน้ำท่วมแบบฉับพลันได้อย่างมีประสิทธิภาพ เพื่อลดการสูญเสียทั้งชีวิตและทรัพย์สินที่มีสาเหตุมาจากฝนตกและน้ำในแม่น้ำมีการเปลี่ยนแปลงเพิ่มขึ้นอย่างฉับพลัน เพื่อทำการเก็บข้อมูลและวิเคราะห์ข้อมูล ส่งสัญญาณแจ้งเตือนภัยให้กับประชาชนที่อาศัยในพื้นที่เสี่ยงภัยจากภัยพิบัติ โดยนำ ET-USB FIASH DRIVE มาใช้ในการเก็บข้อมูลที่ได้จากอินพุตที่ต้องการ คือ ปริมาณน้ำฝนและปริมาณน้ำท่าในแม่น้ำ เพื่อเก็บข้อมูลและนำมาวิเคราะห์ข้อมูลในการแจ้งเตือนภัยธรรมชาติ โดยอาศัยไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและดำเนินการแจ้งเตือนภัยผ่านสัญญาณโทรศัพท์มือถือ

Project title	Processing Unit and Warning System over Mobile Phone II		
Name	Mr. Surapan	Ardoon	ID. 49362321
	Miss. Chompoonut	Wongkerdnimit	ID. 49363991
	Mr. Rungroj	Sanguangwattana	ID. 49364233
Project advisor	Mr. Yongyut Chonbodeechalermroong, Ph.D.		
Major	Electrical Engineering		
Department	Electrical and Computer Engineering		
Academic year	2009		

Abstract

This project was created according to the need of an effective system development that is capable of sudden flood warning through a mobile phone signal. This system also helps to reduce the loss of both life and property caused by rain and increasing water level in the river. The system will then collect and analyze the data, warning people who live in any risky area by using the ET-USB FLASH DRIVE to collect data from the input including rainfall and water level. The device, as a result, will gather and analyze information for natural disaster warning with a microcontroller as a processor that executes the signal via a mobile phone.

กิตติกรรมประกาศ

โครงงานนี้จะสำเร็จลุล่วงไปได้ด้วยความกรุณาเป็นอย่างยิ่งจาก ผู้ช่วยศาสตราจารย์ ดร.ยง ยุทธ ชนบดีเฉลิมรุ่ง ที่สละเวลาอันมีค่ารับเป็นอาจารย์ที่ปรึกษาโครงงานและความช่วยเหลือตลอดจนให้คำแนะนำต่างๆในการทำโครงงานชิ้นนี้

คณะผู้จัดทำขอขอบพระคุณ ดร. มุฑิตา สงฆ์จันทร์และ ดร. นิพัทธ์ จันทรมินทร์ที่ได้สละเวลารับเป็นกรรมการสอบปริญญานิพนธ์ และให้คำแนะนำเพิ่มเติมอันเป็นประโยชน์ในการแก้ไขปรับปรุงโครงงานนี้

ผู้จัดทำโครงงาน

นายสุรพันธ์ อาดูร

นางสาวชมพูนุช วงศ์เกิดนิมิต

นายรุ่งโรจน์ สงวนวัฒนา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ภัยธรรมชาติที่เกิดขึ้นในปัจจุบันนี้ เกิดขึ้นโดยฉับพลันและบ่อยครั้งมากขึ้น เช่น น้ำหลาก สึนามิ ดินสไลด์ เป็นต้น ทำให้มีผู้ได้รับความเสียหายจากภัยธรรมชาตินี้มากมายนัก เนื่องจากไม่มีระบบเตือนภัยที่มีประสิทธิผลพอที่จะสามารถแจ้งเตือนภัยล่วงหน้าได้ และโทรศัพท์มือถือเป็นอุปกรณ์สื่อสารที่มีประสิทธิภาพในการรับและส่งสัญญาณแบบไร้สาย

ดังนั้น จึงได้นำความสามารถในการรับและส่งสัญญาณแบบไร้สายมาประยุกต์ใช้ในการแจ้งเตือนภัย โดยเครื่องส่งจะรับข้อมูลจากหลายๆด้านเก็บไว้และส่งไปยังไมโครคอนโทรลเลอร์เพื่อวิเคราะห์ข้อมูลและสรุปผล หากผลที่ได้เสี่ยงต่อการเกิดอุทกภัยเครื่องส่งจะทำการส่งสัญญาณไปยังเครื่องเตือนภัยผ่านสัญญาณโทรศัพท์มือถือแจ้งเตือนภัยล่วงหน้าให้กับผู้คนที่อาศัยอยู่ในพื้นที่เสี่ยงอุทกภัย เพื่อช่วยลดความเสียหายที่จะมาพร้อมกับภัยธรรมชาติ โครงงานนี้จึงน่าจะช่วยลดความเสียหายที่จะเกิดขึ้นพร้อมกับภัยธรรมชาติได้

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการรับส่งสัญญาณข้อมูลแบบอนาล็อกและแบบดิจิตอล
2. เพื่อศึกษาการพัฒนาวงจรการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ
3. เพื่อสร้างอุปกรณ์ที่สามารถส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

1. ศึกษาการรับส่งสัญญาณข้อมูลในรูปแบบอนาล็อกและแบบดิจิตอล
2. สร้างอุปกรณ์ที่สามารถวัดระดับน้ำฝน น้ำท่า และน้ำในดินได้
3. พัฒนาอุปกรณ์การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ

1.4 ขั้นตอนการดำเนินงาน

กิจกรรม	ปี2552						
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.
1.ค้นคว้าหาข้อมูลการทำงานของอุปกรณ์ต่างๆ	←	→					
2.จัดซื้ออุปกรณ์ต่างๆ		←	→				
3.รวบรวมเนื้อหาการทำงาน			←	→			
4.จัดทำอุปกรณ์						←	→

กิจกรรม	ปี2553									
	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
5.ทำการทดลอง	←		→							
6.แก้ไขการทำงานของอุปกรณ์				←					→	
7.สรุปผลการทดลองและจัดทำรูปเล่ม									←	→

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

1. เข้าใจถึงหลักการทำงานของไมโครคอนโทรลเลอร์
2. เข้าใจถึงหลักการรับส่งสัญญาณในรูปแบบอนาล็อกและดิจิตอล
3. เข้าใจถึงหลักการส่งสัญญาณแบบไร้สาย
4. สามารถนำอุปกรณ์ของโครงงานนี้ไปใช้ในพื้นที่เสี่ยงอุทกภัยได้

1.6 งบประมาณในการทำโครงงาน

1. ถ่ายเอกสารและค่าเข้าเล่มโครงงานฉบับสมบูรณ์ 500 บาท
2. ค่าอุปกรณ์ในการทำโครงงาน 5,000 บาท
3. ค่าพิมพ์เอกสาร 200 บาท

รวมเป็นเงิน 5,700 บาท(ห้าพันเจ็ดร้อยบาทถ้วน)

บทที่ 2
หลักการและทฤษฎี

จากแนวคิดที่จะสร้างระบบสัญญาณเตือนภัยผ่านทางเครือข่ายโทรศัพท์มือถือ ซึ่งประกอบไปด้วย **ET-USB FLASH DRIVE** บอร์ดโทรศัพท์ซึ่งจะนำไปเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ เพื่อที่จะนำมาเขียนโปรแกรมควบคุมระบบการทำงานของอุปกรณ์ต่าง ๆ โดยมีรายละเอียดดังนี้

2.1 การเชื่อมต่อกับ ET-USB FLASH DRIVE

2.1.1 คุณสมบัติของ ET-USB FLASH DRIVE

1. สามารถติดต่อกับตัวเก็บข้อมูล Flash Drive ที่มีโครงสร้างไฟล์แบบ FAT 12 , FAT 16 หรือ FAT 32 ได้
2. รองรับชื่อไฟล์ในรูปแบบ 8.3 คือ ชื่อไฟล์ไม่เกิน 8 ตัวอักษร นามสกุล 3 ตัวอักษร เช่น A1234567.txt
3. ในระบบ FAT 32 จะไม่รองรับชื่อไฟล์แบบยาว ถ้าชื่อไฟล์ยาวเกิน 8.3 จะแสดงชื่อไฟล์ให้เห็นเพียง 8.3
4. ควบคุมการอ่านเขียน Flash Drive โดยใช้การส่ง Command ผ่านทาง RS232
5. สามารถเลือก Baud Rate ในการติดต่อสื่อสารทาง RS232 ได้
6. สามารถส่ง Command โดยใช้ PC หรือ MCU ได้
7. สามารถสร้างและลบไฟล์ หรือ Directory ใน Flash Drive ได้
8. สามารถกำหนดจำนวน Byte ของข้อมูลที่จะทำการอ่านหรือเขียนจากไฟล์ที่อยู่ใน Flash Drive ได้
9. สามารถกำหนดตำแหน่งที่จะอ่านข้อมูลจากไฟล์ หรือเขียนข้อมูลลงไฟล์ที่อยู่ใน Flash Drive ได้
10. สามารถอ่านข้อมูลออกมาทีเดียวทั้งไฟล์ จากไฟล์ที่อยู่ใน Flash Drive ได้
11. หลังจากปิดไฟล์แล้ว สามารถเปิดไฟล์เดิมขึ้นมาทำการเขียนข้อมูลต่อจากของเดิมได้ โดยข้อมูลเก่ายังอยู่
12. สามารถเปลี่ยนชื่อไฟล์หรือชื่อ Directory ใหม่ได้
13. สามารถเข้าไปอ่านเขียน สร้างหรือลบไฟล์ที่อยู่ใน Directory ย่อยได้
14. สามารถเลือกรูปแบบการส่งคำสั่งได้ 2 แบบ คือ ส่งในรูปแบบอักขระASCII (Extended Mode) หรือส่งในรูปแบบ Hex เลขฐาน16 (Short Mode)

2.1.2 การต่อใช้งาน ET-USB FLASH DRIVE

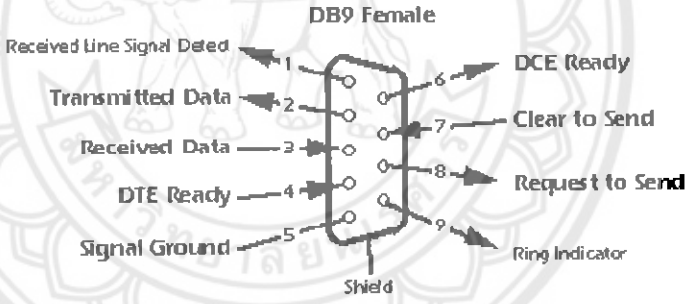
การใช้งาน ET-USB FLASH DRIVE จะใช้การ Interface ผ่านทาง Serial Port (RS232 หรือ Uart) โดยจะต้องกำหนดคุณสมบัติในการติดต่อสื่อสารทาง Serial Port ดังนี้

- Baud Rate จะต้องกำหนดเริ่มต้น default ไว้ที่ 9600 bit/s สามารถส่งคำสั่งเปลี่ยนแปลงได้ในภายหลัง

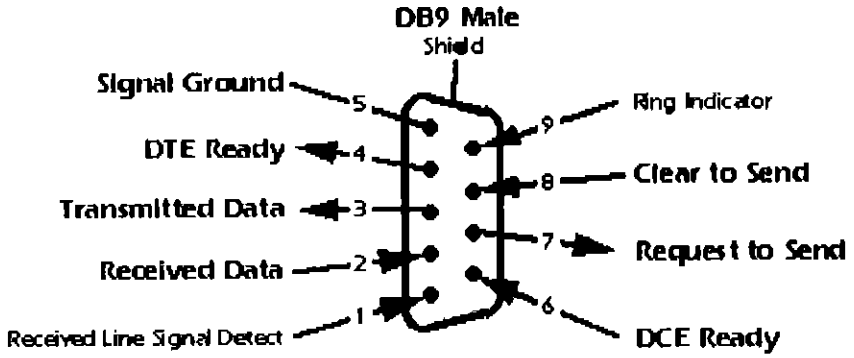
- 8 Data bit , 1 stop bit และ No parity

- Flow Control : ให้กำหนดที่ Hardware ซึ่งก็คือ RTS/CTS จะต้องถูก Enable เพื่อใช้เป็น handshake

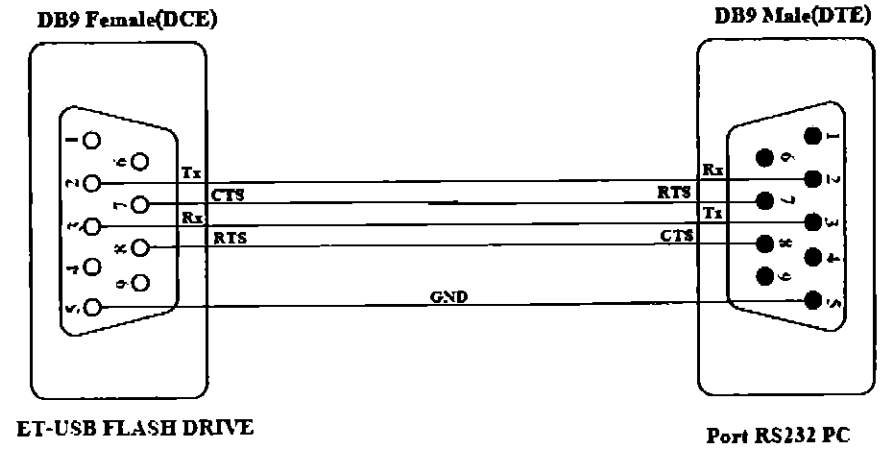
- ในกรณีที่ไม่ต้องการใช้ Handshake ใช้เพียงขา Rx(ขา3) และ Tx(ขา2) และ กราวด์(ขา5) ในการติดต่อสื่อสารเท่านั้น ก็จะต้องทำการ Jump ขา RTS(ขา 7) และ CTS (ขา 8) ที่ Port DB9 ของ ET-USB FLASH DRIVE เข้าด้วยกัน จากนั้นก็ต่อขา Rx และ Tx ของ ET-USB FLASH DRIVE ไปยังขา Rx และ Tx ของอุปกรณ์ที่นำมาควบคุม โดยจะต้องต่อแบบไขว้ขากัน คือ ต่อขา Rx เข้ากับขา Tx และต่อขา Tx เข้ากับ Rx ของอีกฝั่งหนึ่งส่วนกราวด์ให้ต่อเข้าด้วยกัน ดังแสดงในรูปที่ 2.4



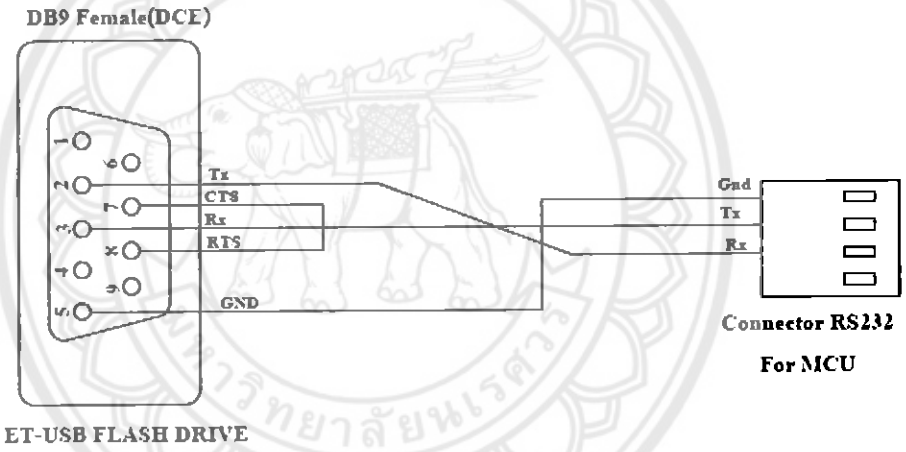
รูปที่ 2.1 ตำแหน่งขา DB9 Female (DCE) ของ ET-USB FLASH DRIVER



รูปที่ 2.2 ตำแหน่งขา DB9 male (DTE) ของ PC



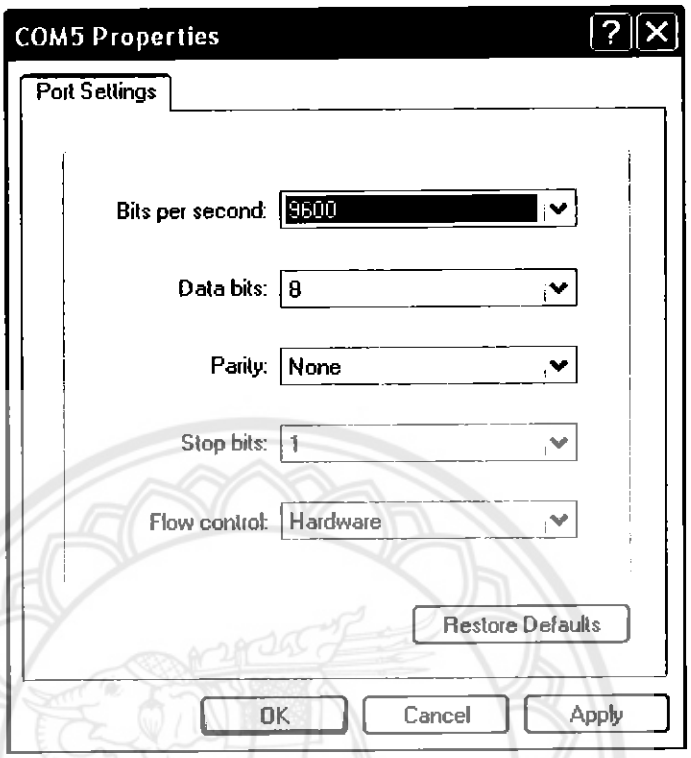
รูปที่ 2.3 การต่อสายสัญญาณ RS232 ระหว่าง ET-USB FLASH DRIVER กับ PC แบบใช้ Handshake(CTS,RTS)



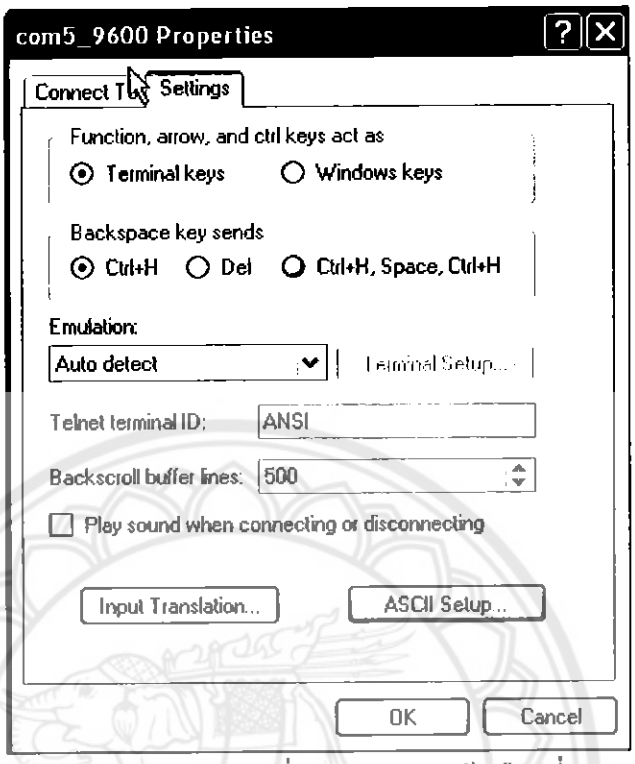
รูปที่ 2.4 การต่อสายสัญญาณ RS232 ระหว่าง ET-USB FLASH DRIVER กับ MCU แบบไม่ใช้ Handshake(Jump ขา 7 CTS และขา 8 RTS เข้าด้วยกัน)

2.1.3 ขั้นตอนการทดสอบใช้งานกับ PC

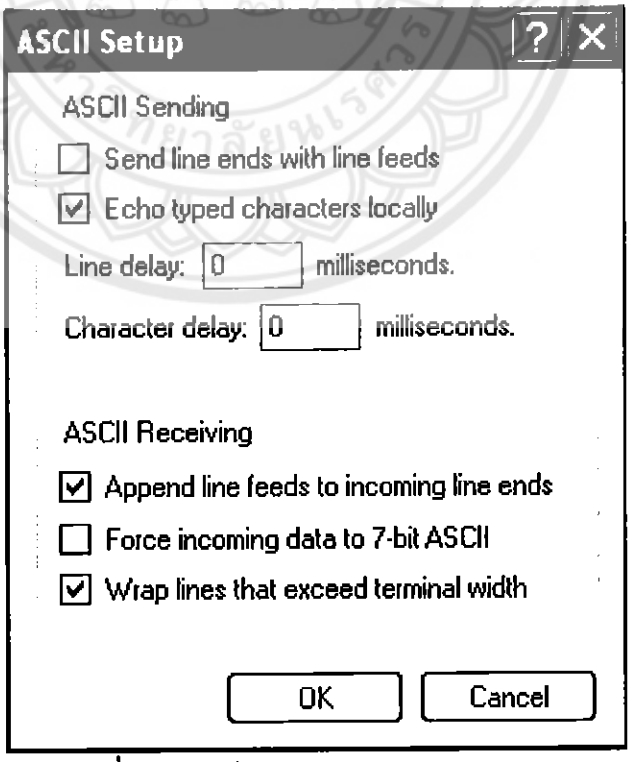
- ต่อ Flash Drive เข้าที่ขั้วต่อ USB ของ ชุด ET-USB FLASH DRIV
- ต่อสาย RS232 จาก PC หรือ MCU เข้าที่ขั้วต่อ RS232 ของชุด ET-USB FLASH DRIVE
- ถ้าส่งคำสั่งผ่านทาง PC ให้เปิดโปรแกรม Hyper Terminal หรือโปรแกรมที่ใช้รับส่งข้อมูลผ่านทาง RS232 ขึ้นมารอไว้โดยให้ Set Up คุณสมบัติดังรูปที่ 2.5 จากนั้นให้กด Ok ก็จะได้หน้าจอ Hyper terminal ขึ้นมา จากนั้นให้คลิกที่ Icon Properties ( ) จะได้หน้าต่างออกมาดังในรูปที่ 2.6 ให้เลือกที่ TAB Setting และคลิกเลือกที่ปุ่ม ASCII Setup จากนั้นก็จะได้หน้าต่างขึ้นมาให้ทำการกำหนดค่าตามในรูปที่ 2.7 จากนั้นให้กด Ok ในแต่ละหน้าต่างเป็นอันเรียบร้อยในการ Set การใช้งาน Hyper terminal



รูปที่ 2.5 การ Set up โปรแกรมHyper Terminal ผ่านทางพอร์ท RS232



รูปที่ 2.6 การ Settings โดยเลือกที่ Tab settings แล้วเลือกที่ ASCII Setup



รูปที่ 2.7 การเซ็ตค่าในหน้าต่าง ASCII Setup

- จ่ายไฟ 7-12 VDC ให้กับชุด ET-USB FLASH DRIVE ให้สังเกตที่ LED Status สีเขียวจะติดค้างนั่นแสดงว่าตัวET-USB FLASH DRIVE ถูกต่อเข้ากับ FLASH DRIVE และ อุปกรณ์ที่ใช้สื่อสารทางด้าน RS232 เรียบร้อยพร้อมใช้งานแล้ว แต่ถ้า LED Status กระพริบสลับระหว่างสีเขียวและแดง แสดงว่า การเชื่อมต่อยังไม่สมบูรณ์ในขณะที่มีการอ่านเขียนข้อมูล LED สีเขียวจะกระพริบ

- หลังจากจ่ายไฟเรียบร้อยแล้วให้รอจนกว่าจะมีข้อความ :

```
Ver 03.55VDAPF On-Line:
Device Detected P2
No Upgrade
D:\>
```

รูปที่ 2.8 ข้อความหลังจากทำการจ่ายไฟให้กับ ET-USB FLASH DRIVE

ปรากฏขึ้นที่หน้าต่าง HyperTerminal และแสดง D:\> พร้อมที่จะรับคำสั่งต่างๆจากผู้ใช้เพื่อติดต่อใช้งาน USB FLASH DRIVE เมื่อพิมพ์คำสั่งเสร็จหรือพิมพ์คำสั่งผิดให้กด Enter เพื่อเริ่มต้นคำสั่งใหม่จะสังเกต D:\> จะขึ้นเสมอแสดงความพร้อมในการรับคำสั่ง โดยค่า default ของ ET-USB Flash Drive จะถูกกำหนดไว้ดังนี้คือ Baud Rate ในการสื่อสาร 9600 bit/s , รับคำสั่งในโหมด Extended Mode และ กำหนดให้มีการรับค่าหรือส่งผ่านค่าที่เป็นตัวเลขในแบบ Binary Mode (IPH) จากนั้นให้ลองทำการทดสอบการเขียนและอ่านไฟล์ ตามตัวอย่างในหัวข้อด้านล่าง

2.1.4 ตัวอย่างการทดสอบ ET-USB FLASH DRIVE

ทดสอบกับโปรแกรม Hyper Terminal

1. ทดสอบการเขียนข้อมูลให้กับไฟล์ชื่อ test01.txt จำนวน 10 Byte

- ส่งคำสั่ง IPA เพื่อกำหนดรูปแบบการผ่านค่าจำนวน Byte ข้อมูลที่จะเขียนให้กับ Monitor ใน ASCII Mode

- ส่งคำสั่ง OPW test01.txt เพื่อเปิดไฟล์สำหรับเขียน

- ส่งคำสั่ง WRF 10 เพื่อเขียนไฟล์โดยระบุจำนวน Byte ที่จะเขียน = 10 Byte แล้ว Enter

- ทำการเขียนไฟล์ 'abcdefghij' เมื่อครบ 10 Byte จะมี Response D:\> ส่งออกมาแสดงว่าเขียนข้อมูลครบแล้ว

- ส่งคำสั่ง CLF test01.txt เพื่อทำการปิดไฟล์ที่เขียน

D:\>	[Prompt ใน Extended mode]
IPA ↵	[กำหนดรูปแบบการส่งจำนวน byte ที่จะเขียนในแบบ ASCII Mode]

D:\>	[Response Prompt]
OPW test01.txt ↵	[ทำการ Open file ชื่อ test01.txt]
D:\>	[Response Prompt]
WRF 10 ↵	[ส่งคำสั่งเขียน file โดยระบุจำนวน byte ที่จะเขียน 10 Byte]
abcdefghij	[เขียน data 10 Byte]
D:\>	[Response Prompt จะแสดงอัตโนมัติเมื่อเขียนข้อมูลตัวที่ 10 เรียบร้อย]
CLF test01.txt ↵	[ส่งคำสั่งปิด test01.txt ที่ได้เปิดเขียนไว้]
D:\>	[Response Prompt สิ้นสุดการเขียน file]

2. ทดสอบการอ่านข้อมูลจากไฟล์ชื่อ test01.txt ออกมาจำนวน 5 Byte ซึ่งค่าที่จะต้องอ่านได้ คือ abcde

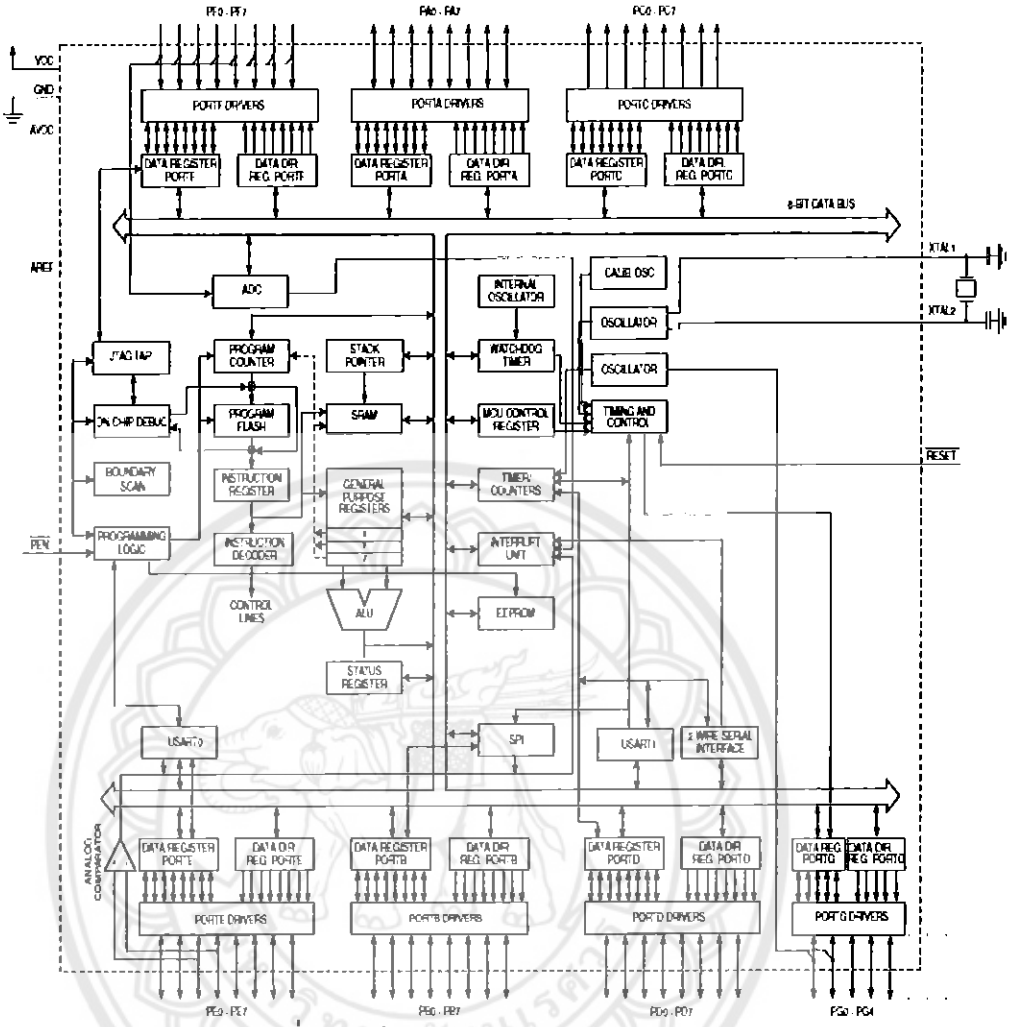
- ส่งคำสั่ง IPA เพื่อกำหนดรูปแบบการผ่านค่า จำนวน Byte ข้อมูลที่จะอ่านให้กับ Monitor ใน ASCII Mode
- ส่งคำสั่ง OPR test01.txt เพื่อเปิดไฟล์สำหรับอ่าน
- ส่ง คำสั่ง RDF 5 เพื่ออ่านไฟล์โดยระบุจำนวน Byte ที่จะอ่าน = 5 Byte แล้ว Enter
- ข้อมูลจะถูกอ่านออกมา 5 byte คือ abcde โดยข้อมูลที่อ่านได้นี้จะถูกนำด้วยค่า 0x0D ส่งออกมาก่อนแล้วถึงตามด้วยข้อมูล 5 byte และปิดด้วยท้ายข้อมูลด้วย D:\>

D:\>	[Prompt ใน Extended mode]
IPA ↵	[กำหนดรูปแบบการส่งจำนวน byte ที่จะอ่านในแบบ ASCII Mode]
D:\>	[Response Prompt]
OPR test01.tx ↵	[ส่งคำสั่ง Open file ชื่อ test01.txt]
D:\>	[Response Prompt]
RDF 5 ↵	[ส่งคำสั่งอ่าน file ที่เปิดอยู่ โดยระบุจำนวน byte ที่จะอ่าน 5 Byte]
↵ abcdeD:\>	[data ถูกอ่านออกมา 5 Byte โดยข้อมูลจะถูกนำด้วย 0x0D และปิดด้วย <prompt>]
↵	[ส่งคำสั่ง enter เพื่อรอรับคำสั่งต่อไป]
D:\>	[Response <prompt>]

2.2 การเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์

2.2.1 ไมโครคอนโทรลเลอร์ AVR รุ่น "ATmega64"

ไมโครคอนโทรลเลอร์AVRเป็นหนึ่งในไมโครคอนโทรลเลอร์ ผลิตโดยบริษัท ATMEL (ผู้นำทางด้านไมโครคอนโทรลเลอร์ตระกูล MCS-51) AVR จัดเป็นไมโครคอนโทรลเลอร์ตระกูลใหม่จาก ATMEL มีสถาปัตยกรรมแบบ RISC (Advanced RISC architecture)คือหนึ่งคำสั่งทำงานใช้สัญญาณนาฬิกาเพียงหนึ่งลูก (instructions in a single clock cycle) เป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพและความสามารถสูง แบ่งออกเป็นหลายอนุกรม ในแต่อนุกรมยังแบ่งออกเป็นหลายเบอร์ เพื่อรองรับความต้องการที่แตกต่างของผู้ใช้งาน ในขณะที่ยังคงความประสิทธิภาพที่เท่ากัน รายละเอียดและคุณสมบัติภายในไมโครคอนโทรลเลอร์ATmega64 แสดงดังไดอะแกรมรูปที่ 2.9



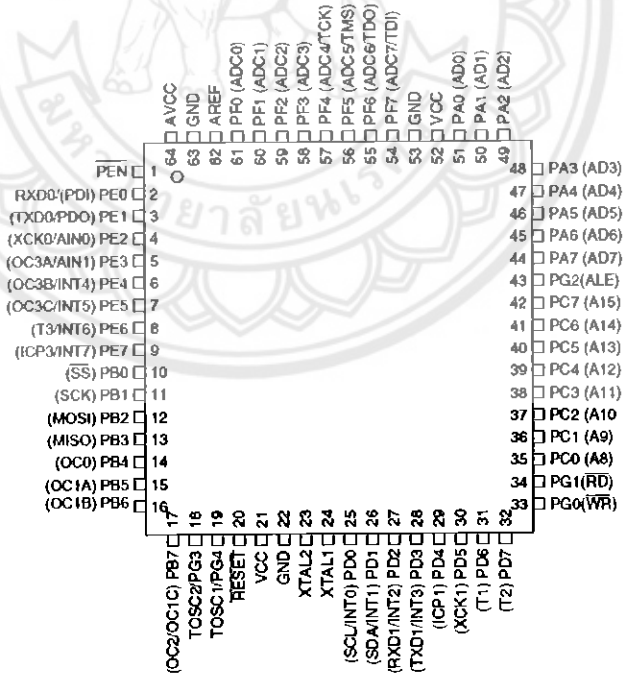
รูปที่ 2.9 บล็อกไดอะแกรม AVR ATmega 64

คุณสมบัติที่สำคัญ

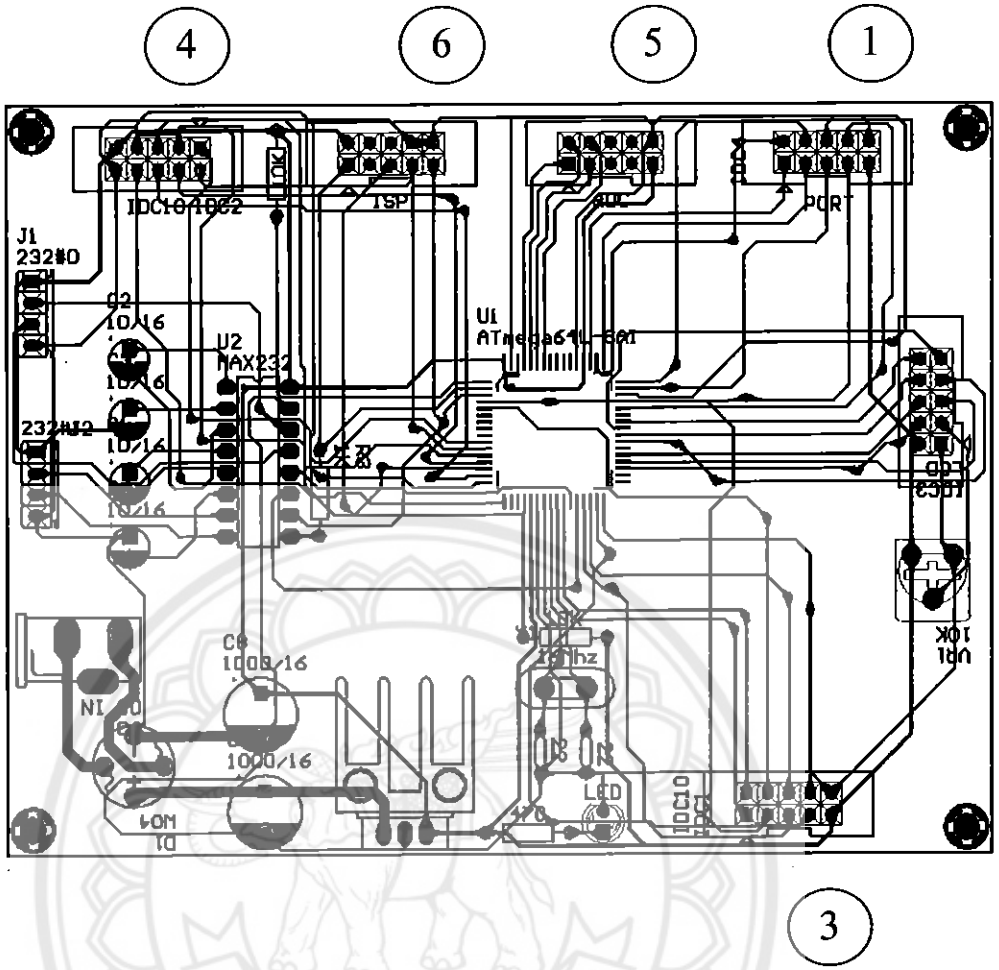
- สถาปัตยกรรมภายในเป็นแบบ Advanced RISC (Reduce Instruction Set Computer)
- มีคำสั่งควบคุมการทำงานมากกว่า100คำสั่งโดยมีความเร็วในการประมวลผล 1 คำสั่งต่อ1สัญญาณนาฬิกา (1 MIP/1 MHz)
- มีรีสเตอร์ใช้งานทั่วไปขนาด8บิตจำนวน 32 ตัว (ทำให้สะดวกต่อการพัฒนาโปรแกรมด้วยภาษาซีเป็นอย่างมาก)
- หน่วยความจำ ROM แบบ Flash (มีโหมดป้องกันหน่วยความจำ) ขนาด 64 กิโลไบต์ (เขียน/ลบได้10,000ครั้ง)
- หน่วยความจำข้อมูลแบบ EEPROM (มีโหมดป้องกันหน่วยความจำ) ขนาด 2 กิโลไบต์ (เขียน/ลบได้100,000ครั้ง)
- หน่วยความจำข้อมูลแบบ SRAM 4 กิโลไบต์

- 2 ไทเมอร์/เคาร์เตอร์ทั้งแบบ 8 บิตและ 16 บิต พร้อมปรีสเกลเลอร์
- มีระบบตรวจสอบความผิดพลาดในการทำงานของซอฟต์แวร์ (Watchdog Timer with On-chip Oscillator)
- โมดูลสร้างสัญญาณ PWM (Pulse Width Modulator) มีจำนวน 6 ช่อง
- มีโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) ขนาด 10 บิตมากถึง 8 ช่อง
- โมดูลเปรียบเทียบแรงดันอะนาลอก (Analog Comparator)
- การสื่อสารข้อมูลอนุกรมมีทั้งแบบ UART (Universal Asynchronus Receiver Transmitters) หรือแบบ RS232,SPI (Serial Peripheral Interface) และแบบ I^2c เป็นต้น
- พอร์ตอินพุตเอาต์พุตขึ้นอยู่กับเบอร์ AVR ที่เลือกใช้งานมีตั้งแต่8ขาจนมากกว่า 100 ขา พอร์ต (ATmega64 มีขาพอร์ตอินพุตเอาต์พุต 53ขา)
- แรงดันไฟเลี้ยงและความเร็วในการทำงานขึ้นอยู่กับเบอร์AVR ที่เลือกใช้งาน

หมายเหตุ คุณสมบัติต่างๆจะเกี่ยวข้องโดยตรงกับเบอร์ของAVRที่เลือกใช้งาน ซึ่งจะมีคุณสมบัติแตกต่างกันไปในแต่ละเบอร์

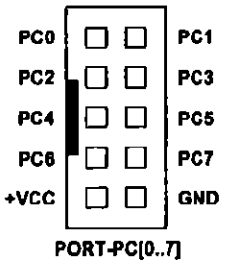


รูปที่ 2.10 ขาพอร์ต AVR ATmega64

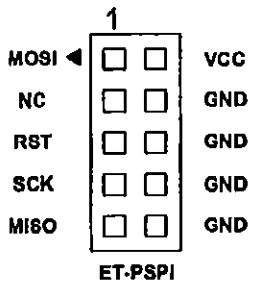


รูปที่ 2.11 วงจรบอร์ดไมโครคอนโทรลเลอร์

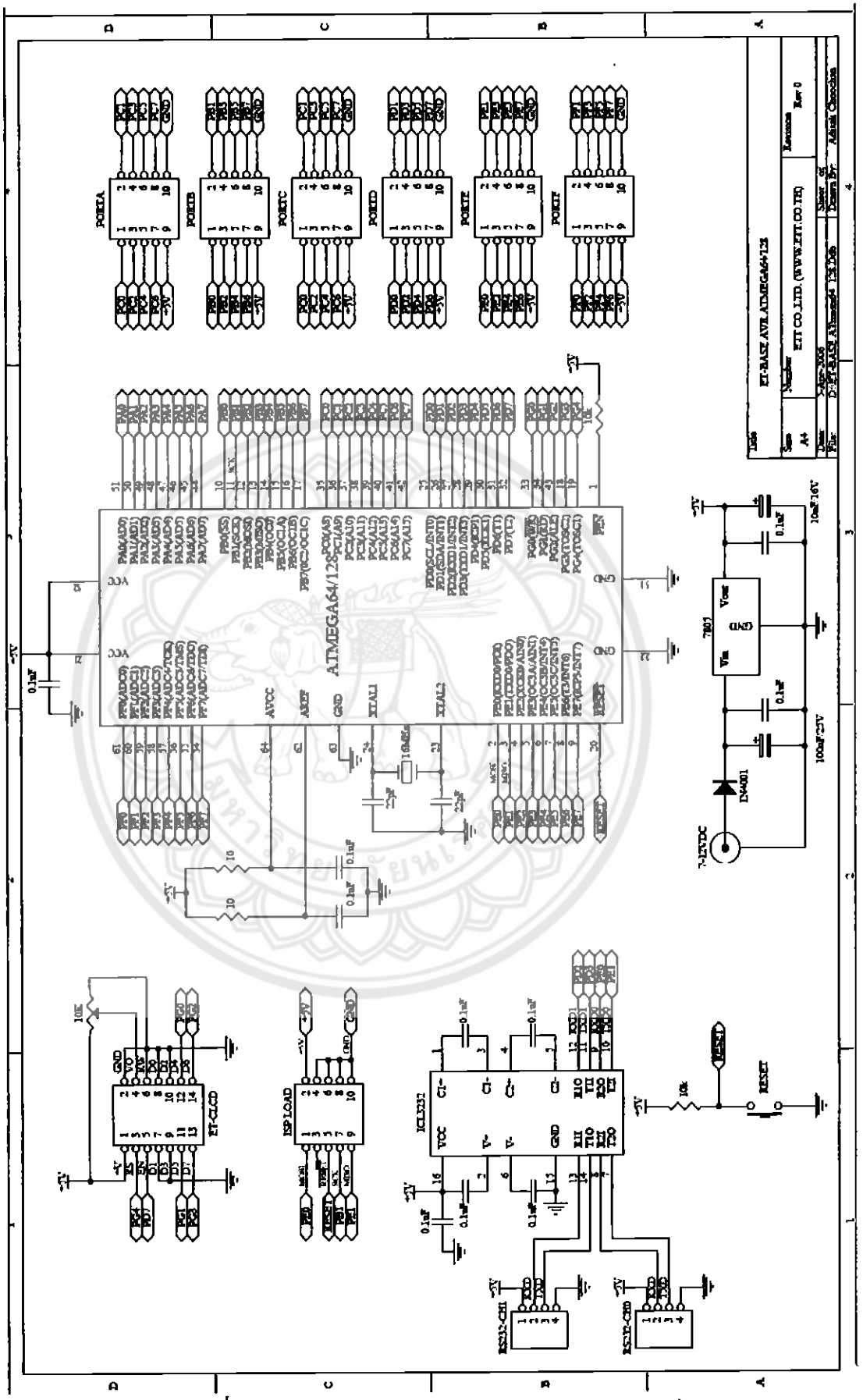
- หมายเลข 1 คือ พอร์ต A ซึ่งประกอบไปด้วย PA0 – PA7
- หมายเลข 2 คือ พอร์ต C ซึ่งประกอบไปด้วย PC0 – PC7 โดยพอร์ตนี้จะถูกเชื่อมต่อออกมารอไว้ยัง Connector ขนาด 10 PIN แบบ IDE โดยการจัดเรียงขาเป็นดังรูป



- หมายเลข 3 คือ พอร์ต D ซึ่งประกอบไปด้วย PD0 – PD7
- หมายเลข 4 คือ พอร์ต B ซึ่งประกอบไปด้วย PE0 – PE7
- หมายเลข 5 คือ พอร์ต F ซึ่งประกอบไปด้วย PF0 – PF7



- หมายเลข 6 คือ พอร์ต ET-PSPI สำหรับเชื่อมต่อกับวงจร ISP PROGRAMMER เพื่อโปรแกรม Hex File ให้กับ AVR



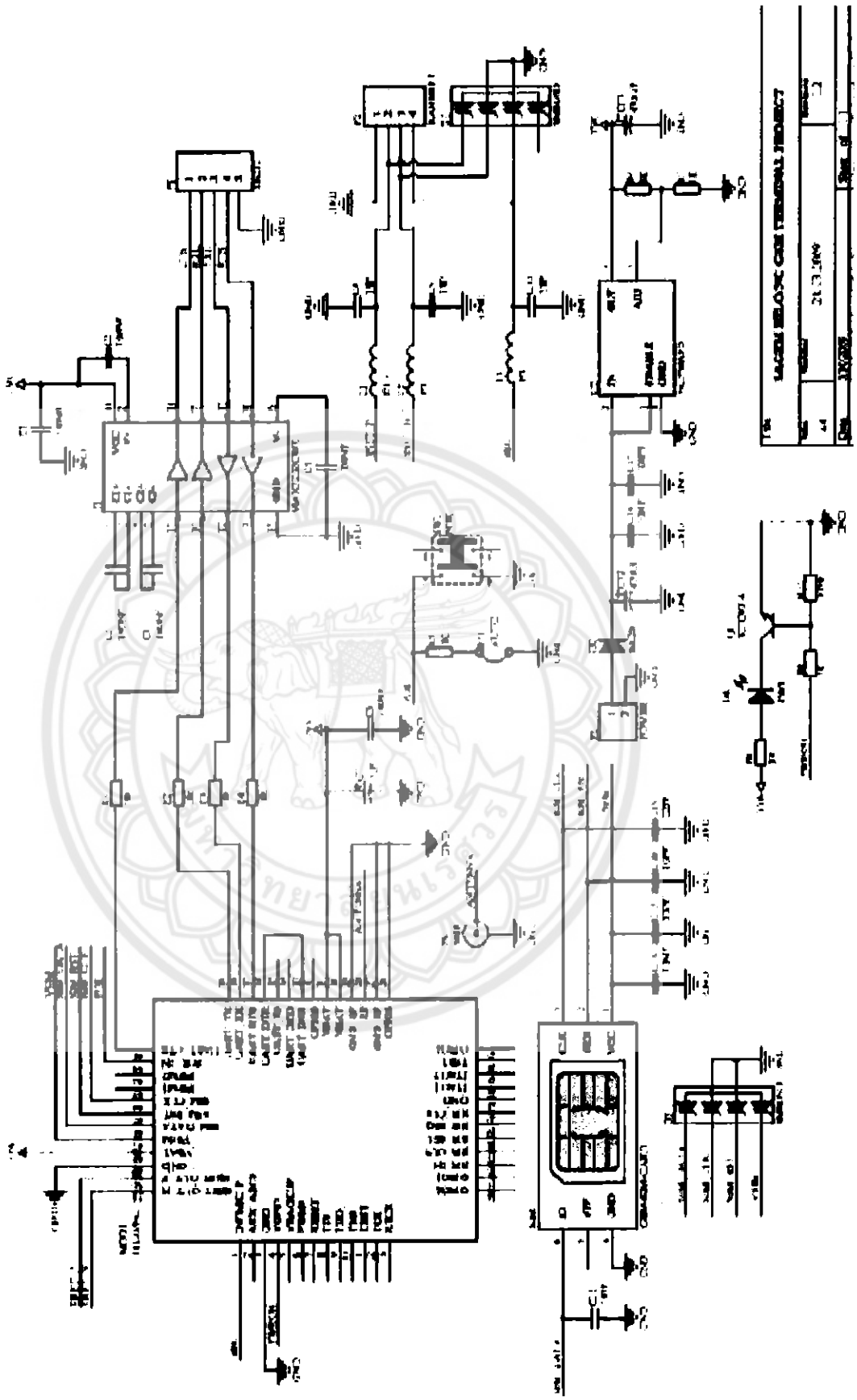
รูปที่ 2.12 บล็อกไดอะแกรมบอร์ดไมโครคอนโทรลเลอร์

2.3 GSM Module (Sagem: Hilo/NC)

คุณสมบัติของ GSM Module (Sagem: Hilo/NC)

- รองรับระบบ GSM ทุกแบนด์ e (850, 900, 1800, 1900Mhz)
- กำลังสูง 2 วัตต์สำหรับย่าน 850 MHz และ 900 MHz, 1 วัตต์สำหรับ 1800 MHz และ 1900 MHz
- สามารถใช้ SIM card ชนิด 3 โวลต์ หรือ 1.8 โวลต์
- สั่งงานผ่าน UART ความเร็วตั้งแต่ 600 ~ 115200 bps มีโหมด auto baud rate
- ใช้คำสั่ง AT Command ตามมาตรฐาน Hays และ Propritary command
- สามารถโทรเข้า -โทรออกได้
- รองรับ Data mode ทั้ง CSD(Circuit Switch Data) , FAX และ GPRS
- GPRS Class 10 สามารถส่งข้อมูลด้วยอัตราเร็วสูงสุด 85.6 kbps downlink และ42.8 kpbs uplink
- รองรับ SMS และ Cell Broadcast ในแบบ text mode
- รองรับ USSD และ Sim Application Tools Kit
- มี Phone book ในตัวโมดูล สามารถบันทึกเลขหมายโทรศัพท์ได้
- สนับสนุนบริการเสริม Caller Line Identification, Call Waiting, Call Hold, Forwarding, Multiparty, Call Barring, Advice of Charge.

รูปที่ 2.13 วงจร Sagem: Hilo/NC



รูปที่ 2.14 บล็อกไดอะแกรม Sagem: Hilo/NC

2.3.1 หลักการรับส่ง SMS ของโทรศัพท์มือถือ

SMS ย่อมาจาก Short Massage Service เป็นบริการส่งข้อความสั้นๆจากโทรศัพท์มือถือต้นทางผ่านชุมสายไปยังโทรศัพท์มือถือปลายทาง โดยสามารถส่งได้สูงสุด 160 ตัวอักษรต่อครั้งตามข้อกำหนดมาตรฐานขององค์การ ETSI (European Telecommunications Standards Institute)

2.3.1.1 โหมดของการรับส่งข้อมูล SMS

แบ่งออกเป็น 2 โหมดคือ Text Mode และ PDU Mode (Protocol Description Unit Mode) การส่งข้อความใน Text Mode นั้นจะเป็นการนำข้อความที่ต้องการส่งมาเข้ารหัสก่อน(โดยตัวเครื่องเอง) แล้วจึงส่งข้อมูลในรูป PDU Mode อีกครั้งหนึ่ง แต่ในบางเครื่องก็ไม่สนับสนุนการส่งแบบ Text Mode ผ่านทาง AT Command แต่หากเป็น PDU Mode จะสามารถส่งได้ เนื่องจากเครื่องจะไม่ต้องทำอาศัยการแปลงข้อมูลอีกชั้น

2.3.1.2 รูปแบบในการส่งข้อมูลในรูป SMS ผ่าน AT Command

มี 2 รูปแบบ คือ Text Mode และ PDU Mode

ก.Text Mode เป็นการส่งข้อมูลในรูปของตัวอักษรได้โดยตรง ซึ่งตัวเครื่องส่วนใหญ่ไม่รองรับการส่งข้อมูลรูปแบบนี้ผ่านทาง AT Command จึงไม่สามารถใช้งานได้สมบูรณ์ เนื่องจากการส่งข้อความใน Text Mode นั้นจะเป็นการนำข้อความที่ต้องการส่งมาเข้ารหัสก่อน (โดยตัวเครื่องเอง) แล้วจึงส่งข้อมูลในรูป PDU Mode อีกครั้งหนึ่ง แต่ในโทรศัพท์บางเครื่องก็ไม่สนับสนุนการส่งข้อความแบบ Text Mode ผ่านทาง AT Command แต่หากส่งข้อความเป็น PDU Mode จะสามารถส่งได้ เนื่องจากโทรศัพท์จะไม่ต้องมีการแปลงข้อมูลอีกชั้นหนึ่ง

ข. PDU Mode PDU (Protocol Data Unit) คือโหมดการทำงานประเภทหนึ่ง ซึ่งจะทำการแปลงรหัสแอสกี (ASCII) ของตัวอักขระแต่ละตัวให้เป็นรหัส PDU ซึ่งรหัส PDU นั้น สามารถนำมาใช้งานได้กับชุดคำสั่ง AT Command ในการส่ง SMS สามารถใช้ได้กับโทรศัพท์มือถือทุกเครื่องที่รับคำสั่ง AT Command ได้ โดยที่การเข้ารหัส PDU มีขั้นตอนดังนี้

- จะต้องทราบรหัสแอสกีแบบเลขฐาน 16 (Hexadecimal) ของแต่ละอักขระ
- แปลงจากรหัสแอสกีแบบเลขฐาน 16 เป็นรหัสแอสกีแบบเลขฐาน 2 (Binary)
- รหัสแอสกีแบบเลขฐาน 2 มาตัดบิตซ้ายสุดทิ้ง
- แปลงเป็นรหัส PDU โดยนำบิตสุดท้ายของตัวอักขระตัวที่ 2 มาวางหน้า 7 บิตของอักขระตัวที่ 1 ซึ่งจะได้รหัส PDU ของอักขระตัวที่ 1 จากนั้นนำ 2 บิตสุดท้ายของอักขระตัวที่ 3 มาวางหน้า 6 บิตที่เหลืออยู่ของอักขระตัวที่ 2 ซึ่งจะได้รหัส PDU ของอักขระตัวที่ 2 จากนั้นนำ 3 บิตสุดท้ายของอักขระตัวที่ 4 มาวางหน้า 5 บิตที่เหลือของอักขระตัวที่ 3 ซึ่งจะได้รหัส PDU ของอักขระตัวที่ 3 จากนั้นทำตามขั้นตอนเดิมไปเรื่อยๆ จนได้รหัส PDU 8 บิต ของทุกอักขระ

- แปลงรหัสPDU 8 บิตที่ได้ให้เป็นรหัส PDU แบบเลขฐาน 16 การเข้ารหัส PDUของคำว่า ALERT จะเห็นว่ารหัส PDU ของคำว่า ALERT คือ 4166514A05 ดังรูปที่ 2.15

Format	A	L	E	R	T
ASCII Hex	41	4C	45	52	54
ASCII Bin	0100 0001	0100 1100	0100 0101	0101 0010	0101 0100
บิตที่จะเข้ารหัส	100 0001	100 1100	100 0101	101 0010	101 0100

PDU	0100 0001	0110 0110	0101 0001	0100 1010	0000 0101
PDU Hex	41	66	51	4A	05

รูปที่ 2.15 ตัวอย่างการเข้ารหัส PDU ของคำว่า ALERT

2.3.1.3 กลุ่มคำสั่งในการส่ง SMS

ก. AT+CMGF เป็นคำสั่งที่ใช้เลือกรูปแบบของการส่งข้อความ ซึ่งมี 2 โหมด คือ SMS PDU Mode กับ SMS Text Mode

<mode> 0 คือ เลือกใช้ PDU mode

1 คือ เลือกใช้ Text mode

ข. AT+CMGS เป็นคำสั่งที่ใช้ส่งSMS โดยมีรูปแบบของคำสั่งดังนี้

<da> คือ หมายเลขโทรศัพท์ปลายทางที่จะส่ง SMS ไป

<text is entered> คือ ข้อความที่ต้องการจะส่ง

<Length> คือ ความยาวข้อมูลของข้อความใน PDU mode

<PDU is given> คือ ข้อมูลในรูปแบบรหัส PDU แบบเลขฐาน 16

<CR> คือ ปุ่ม Enter บนคีย์บอร์ด

<CTRL-Z> คือ ปุ่ม Ctrl และ ปุ่ม Z บนแป้นพิมพ์

2.3.1.4 กลุ่มคำสั่งในการควบคุมและดูสถานะของโทรศัพท์มือถือ

- AT+CPBS เป็นคำสั่งที่ใช้ในการเลือกที่เก็บหน่วยความจำของสมุดโทรศัพท์ (Select Phone Book Memory Storage)

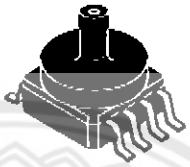
- AT+CPBR เป็นคำสั่งที่ใช้อ่านข้อมูลจากสมุดโทรศัพท์ (Read Phone Book Entries)

2.3.1.5 กลุ่มคำสั่งบริการเครือข่าย

- AT+CLIP เป็นคำสั่งที่ใช้ในการแสดงผลหมายเลขโทรศัพท์ที่โทรเข้า (Calling Line Identification Presentation)

2.4 เซ็นเซอร์ชุดโมโตโรล่า MPXAZ6115A transducer

ความถูกต้องที่อุณหภูมิสูงและความน่าเชื่อถือสูง ใช้วัดความดันสัมบูรณ์



รูปที่ 2.16 MPXAZ6115A Transducer

คุณสมบัติที่สำคัญ

- ทนต่อความชื้นสูง
- ความถูกต้องที่อุณหภูมิสูง
- ข้อผิดพลาด 1.5% สูงสุดกว่า 0 ° ถึง 85 ° C
- เหมาะสำหรับไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์

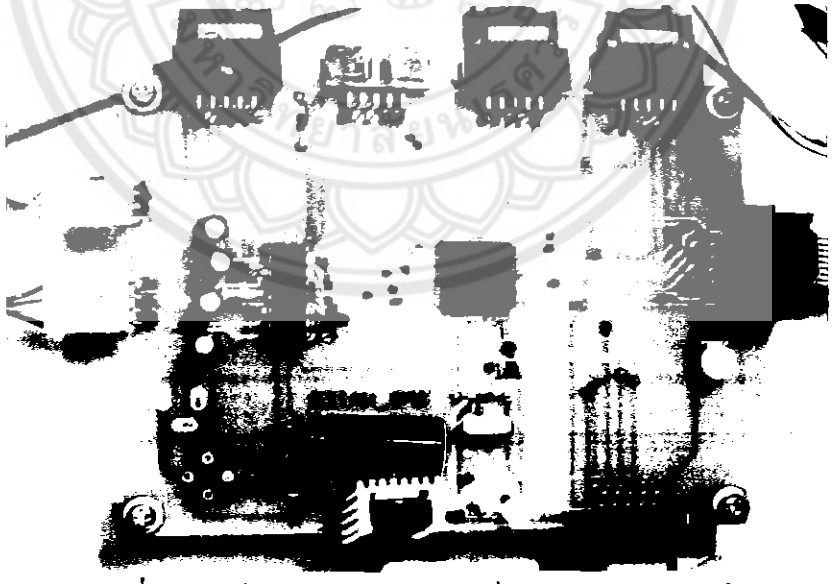
บทที่ 3
วิธีการออกแบบ

ในบทที่ผ่านมาเป็นการศึกษาในเรื่องของที่มาและความสำคัญ รวมถึงหลักการต่าง ๆ ของการสร้างระบบสัญญาณเตือนภัยผ่านเครือข่ายโทรศัพท์ที่มีการเชื่อมต่อกับ ไมโครคอนโทรลเลอร์ และบอร์ดโทรศัพท์ สำหรับบทนี้จะเป็นการศึกษาถึงการออกแบบและการจัดวางของอุปกรณ์ต่าง ๆ โดยแบ่งออกเป็นส่วน ๆ ได้ดังนี้ คือ ส่วนของภาคประมวลผลและเก็บข้อมูล และส่วนของภาครับข้อมูล

3.1 ส่วนควบคุม

3.1.1 ส่วนของเครื่องควบคุมภาคส่ง

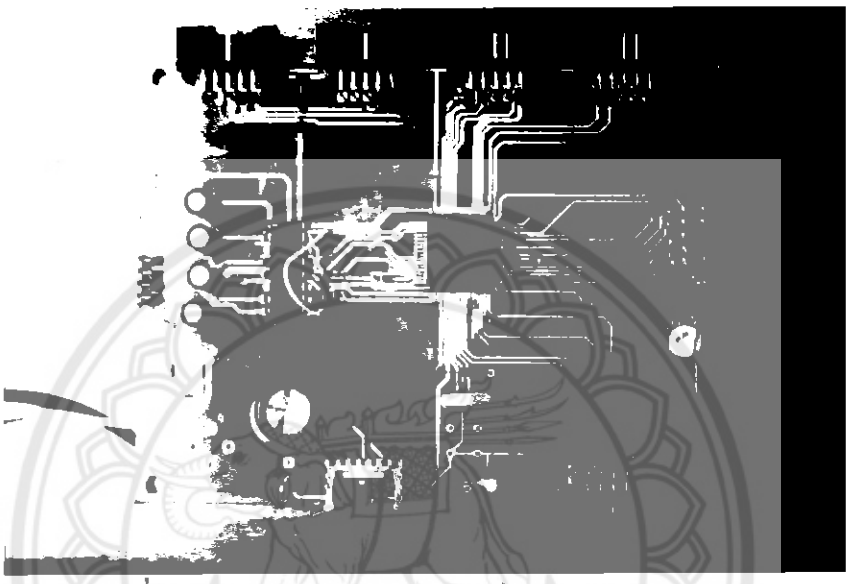
ในส่วนของภาคตัวส่งข้อมูลเราใช้ ไมโครคอนโทลเลอร์ของ "ATMEGA64" มีหลักการการทำงานคือเมื่อได้รับ สัญญาณอินพุต ไมโครคอนโทลเลอร์จะทำการประมวลผลลง "ET-USB FLASH DRIVE" โดยผ่านทาง Port RS232#1 และถ้าหากผลที่ได้ตรงตามเงื่อนไขความเสี่ยงภัย ไมโครคอนโทรลเลอร์จะส่ง AT COMMAND ออกไปยัง "GSM Module SAGEM HILO" ให้ส่ง SMS เตือนภัยไปยังภาครับและแจ้งไปยังผู้ดูแลระบบ โดยผ่านทาง Port RS232#2



รูปที่ 3.1 บอร์ดไมโครคอนโทลเลอร์ ATMEGA 64 ภาคส่ง

3.1.2 ส่วนของเครื่องควบคุมภาครับ

ในส่วนของเครื่องควบคุมภาครับจะใช้ ไมโครคอนโทรลเลอร์รุ่น "ATMEGA64" มีหลักการทำงานดังนี้ คือ เมื่อ GSM Module SAGEM HILO ได้รับ SMS จากภาคส่งข้อมูลโดยผ่านทาง Port RS232 ที่ใช้ในการเชื่อมต่อกับไมโครคอนโทรลเลอร์จะทำการตรวจสอบว่าเป็น SMS จากภาคส่งแล้วจะส่งสัญญาณเอาท์พุทออกไปยังรีเลย์ให้จ่ายกระแสไฟฟ้าเข้าวงจรไซเรน เพื่อแจ้งเตือนภัยและเก็บค่าลง EEPROM



รูปที่ 3.2 บอร์ดไมโครคอนโทลเลอร์ ATMEGA 64 ภาครับ

3.2 ส่วนของเครื่องเก็บข้อมูล

ในส่วนของเครื่องเก็บข้อมูลลง USB จะใช้ ET-USB FLASH DRIVE มีหลักการทำงานดังนี้ คือ รับข้อมูลจากไมโครคอนโทลเลอร์ "ATMEGA64"ที่รับมาจากอินพุตหลังจากนั้นจะทำการบันทึกข้อมูลเป็น TEXT FILE ลง USB เพื่อเก็บไว้เป็นสถิติทุกๆหนึ่งชั่วโมง

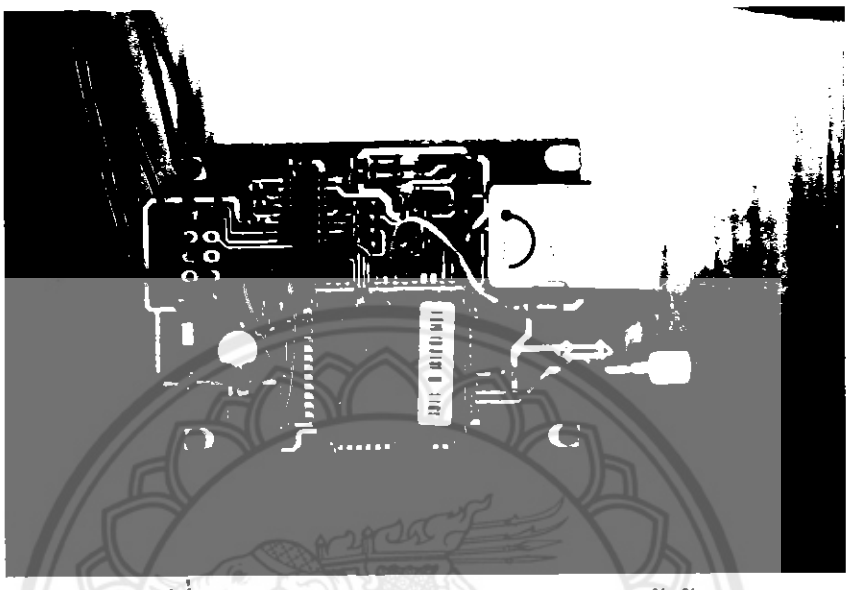
รูปที่ 3.3 ET-USB FLASH DRIVE

3.3 ส่วนของภาคส่ง –รับสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือ

สำหรับในส่วนของภาคส่งสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือนี้ จะใช้ไมโครคอนโทลเลอร์ รุ่น "ATMEGA64" ใช้งานร่วมกับ GSM Module SAGEM HILO ควบคุมระบบโดยผ่านการส่งข้อความ ซึ่งมีการตั้งค่าข้อความตามที่เราต้องการให้ส่งเพื่อเตือนภัยประชาชนในพื้นที่และผู้ดูแลระบบ

รูปที่ 3.4 วงจร GSM Module SAGEM HILO ตัวส่ง

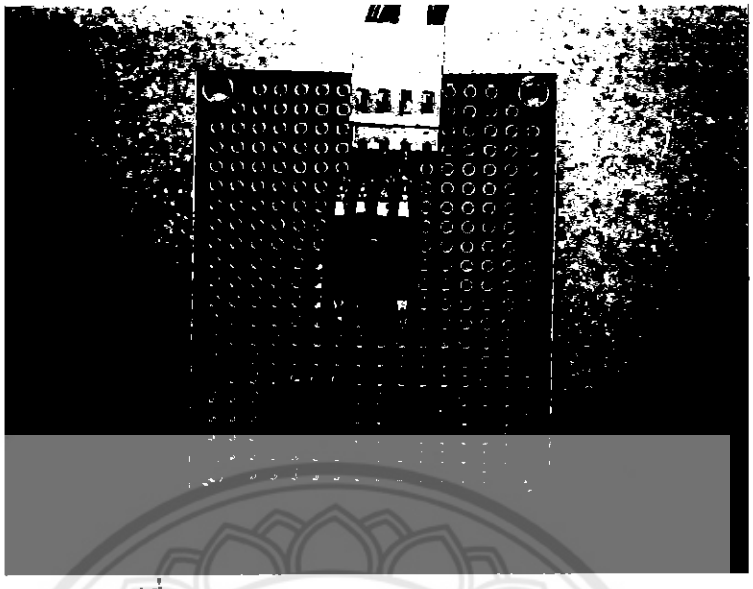
ในส่วนของภาครับสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือนี้ จะใช้ GSM Module SAGEM HILO ใช้งานร่วมกับไมโครคอนโทลเลอร์ รุ่น "ATMEGA64" ควบคุมระบบโดยผ่านการรับข้อความ ซึ่งเมื่อรับข้อความจากภาครับแล้วไมโครคอนโทรลเลอร์จะตรวจสอบว่าเป็นเบอร์ภาคส่งแล้วจึงจะแจ้งเตือนภัย



รูปที่ 3.5 วงจร GSM Module SAGEM HILO ตัวรับ

3.4 ส่วนของการรับข้อมูล

สำหรับในการวัดน้ำท่าและปริมาณน้ำในดินเราจะใช้ Sensor Motorola รุ่น MPXAZ6115A เพื่อวัดแรงดันอากาศในสายยางต่อ 1 หน่วยพื้นที่ โดยจะวัดหน่วยออกมาเป็น "เซนติเมตร" โดย Sensor จะต่อกับสายยางเพื่อใช้วัดแรงดันอากาศในท่อสายยางจะแปรผันกับระดับน้ำยิ่งระดับน้ำสูงแรงดันอากาศในสายยางจะเพิ่มขึ้นตาม

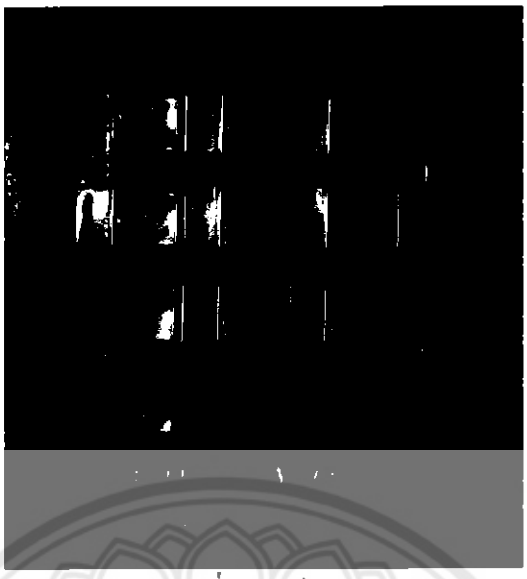


รูปที่ 3.6 วงจร Sensor Motorola MPXAZ4115C

สำหรับในการวัดน้ำฝนเราจะใช้ สวิทช์หน้าสัมผัส เป็นตัวเก็บค่าซึ่งจะให้สัญญาณออกมาเป็นพลัสส์ ซึ่งจะได้หน่วยของน้ำฝนออกมาเป็น 1 ครั้งต่อ 1 มิลลิเมตร

รูปที่ 3.7 สวิทช์หน้าสัมผัส

ดีย์แพด จะทำหน้าที่ใช้เป็นตัวเรียกดูข้อมูลจาก Flash drive จากทางด้านตัวส่ง และยังมีฟังก์ชัน manual ที่จะกดส่งทดสอบ SMS ได้เลย ในส่วนของตัวรับ จะเป็นการแสดงผลของ SMS ที่เตือนภัยมาแสดงให้ดู

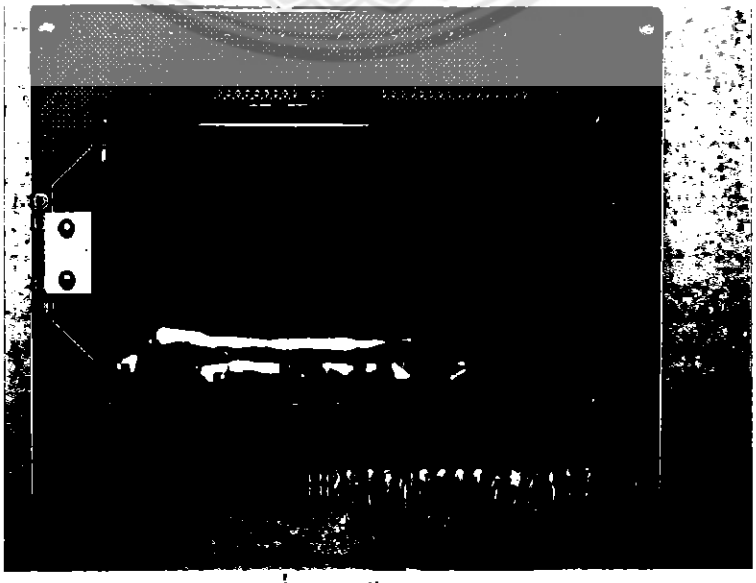


รูปที่ 3.8 คีย์แพด

3.5 ส่วนของภาคตัวแสดงผล

3.5.1 หน้าจอแสดงผล

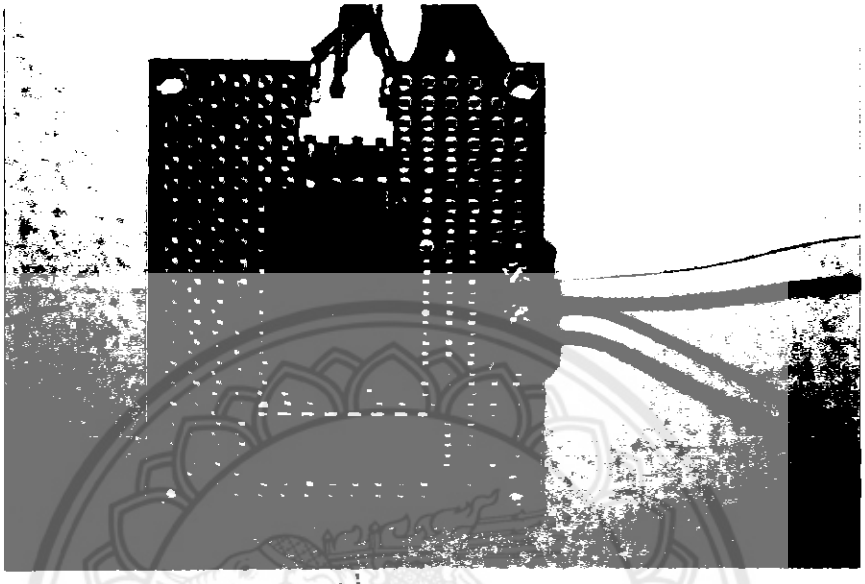
จะเป็นตัวแสดงผล โดยเราใช้หน้าจอแสดงผล ขนาด 4x16 บิต ทั้งเพื่อให้ทราบถึงค่าของน้ำท่า ปริมาณของน้ำฝนและปริมาณน้ำในดิน โดยหน้าจอแสดงผลจะต่ออยู่กับ บอร์ด ATMEGA 64 ทั้งตัวรับและตัวส่ง โดยในส่วนของภาคส่งหน้าจอแสดงผลจะแสดงค่า เบอร์โทรศัพท์ วันที่ เดือน ปี เวลาและปริมาณน้ำท่า น้ำฝน และปริมาณน้ำในดิน ในส่วนของภาครับจะแสดงผลออกมาเพื่อให้อ่านค่า SMS ที่เตือนภัย และค่า EEPROM ล่าสุด 5 ค่า



รูปที่ 3.9 หน้าจอแสดงผล

3.5.2 Relay

จะเป็นอุปกรณ์ที่ต่ออยู่กับตัวรับโดยเมื่อภาคส่งและภาครับได้รับ SMS เตือนภัยแล้วจะสั่งให้ Relay ทำงานโดยจะได้เสียงผ่านทางไซเรนเป็นการแจ้งเตือนภัย



รูปที่ 3.10 วงจร Relay

3.6 เงื่อนไขการแจ้งเตือน

3.6.1 กรณีที่ 1

เมื่อมีข้อมูลน้ำท่าเต็มตลิ่งน้อยกว่า 50 cm เข้ามาให้เตือนปริมาณน้ำฝนตามปกติ เช่น น้ำฝน รับข้อมูลเข้ามาทีละ 1 mm และทำการเก็บข้อมูลไว้โดยที่

ในเวลา 1 ชั่วโมง ถ้าน้ำฝนเกิน	35	mm	แจ้งเตือน Flash (Rain),(L)
	50	mm	แจ้งเตือน Flash (Rain),(M)
	70	mm	แจ้งเตือน Flash (Rain),(H)
ในเวลา 1 วัน ถ้าน้ำฝนเกิน	90	mm	แจ้งเตือน Flash (Rain),(L)
	100	mm	แจ้งเตือน Flash (Rain),(M)
	110	mm	แจ้งเตือน Flash (Rain),(H)
ในเวลา 3 วัน ถ้าน้ำฝนเกิน	185	mm	แจ้งเตือน Flash (Rain),(L)
	200	mm	แจ้งเตือน Flash (Rain),(M)
	215	mm	แจ้งเตือน Flash (Rain),(H)

3.6.2 กรณีที่ 2

เมื่อมีข้อมูลน้ำฝนเข้ามาทำให้น้ำท่าที่เต็มตลิ่งเกิน 50 cm ให้เก็บค่าไว้ แล้วเลื่อนการเตือนของปริมาณน้ำฝนไปหนึ่งขั้น เช่น

น้ำฝน รับข้อมูลเข้ามาทีละ 1 mm และทำการเก็บข้อมูลไว้โดยที่

ในเวลา 1 ชั่วโมง ถ้าน้ำฝนเกิน	35 mm	แจ้งเตือน Flash (Rain),(M)
	50 mm	แจ้งเตือน Flash (Rain),(H)
	70 mm	แจ้งเตือน Flash (Rain),(H)
ในเวลา 1 วัน ถ้าน้ำฝนเกิน	90 mm	แจ้งเตือน Flash (Rain),(M)
	100 mm	แจ้งเตือน Flash (Rain),(H)
	110 mm	แจ้งเตือน Flash (Rain),(H)
ในเวลา 3 วัน ถ้าน้ำฝนเกิน	185 mm	แจ้งเตือน Flash (Rain),(M)
	200 mm	แจ้งเตือน Flash (Rain),(H)
	215 mm	แจ้งเตือน Flash (Rain),(H)

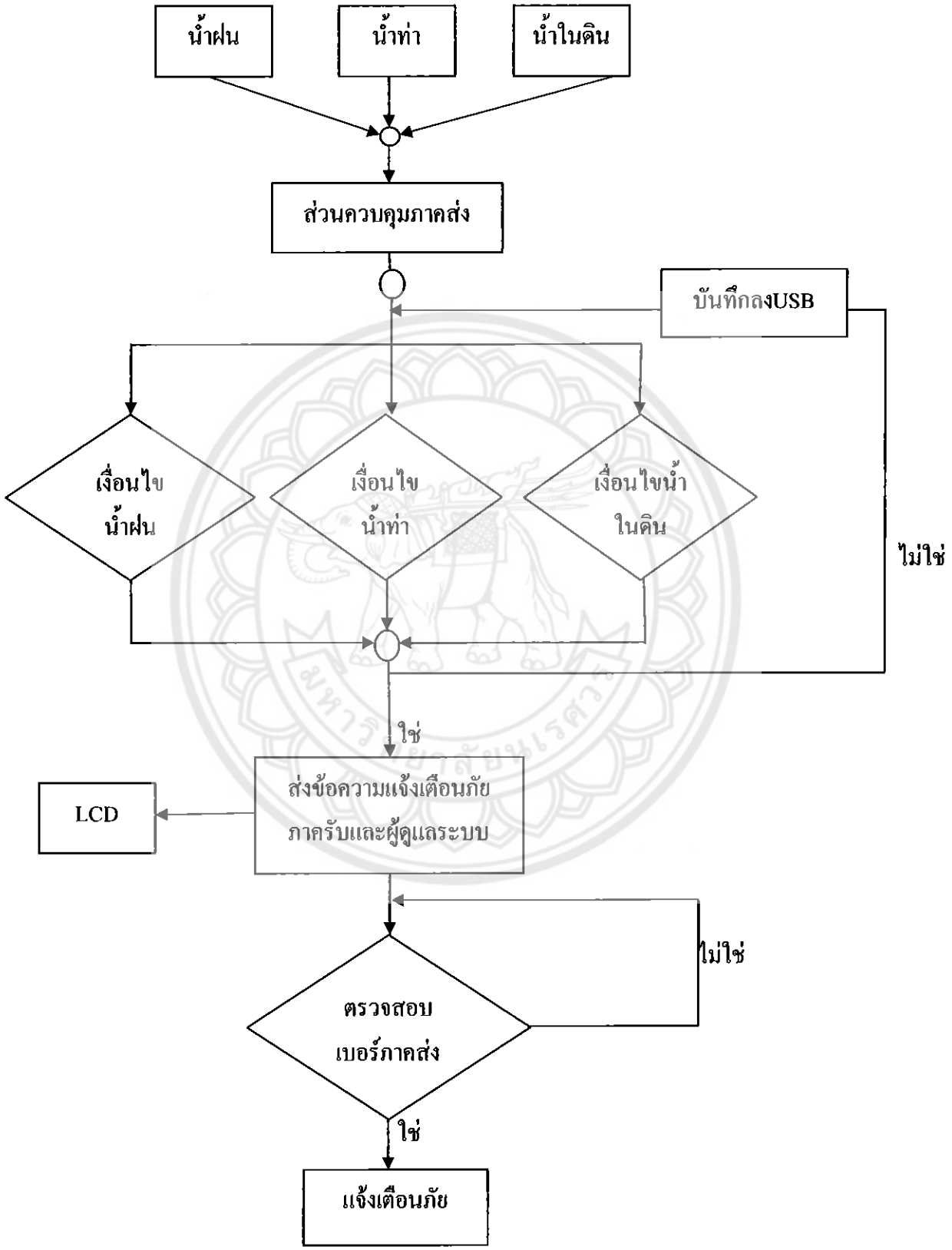
ความชื้นของดิน เมื่อมีข้อมูลเข้ามา 50 cm ขึ้นไปแจ้งเตือนเป็นระดับสูงสุด

3.6.3 กรณีน้ำท่า

ระดับน้ำท่าเกิน	65 mm ขึ้นไป	แจ้งเตือนภัยFlash(Water),(L)
ระดับน้ำท่าเกิน	80 mmขึ้นไป	แจ้งเตือนภัย Flash(Water),(M)
ระดับน้ำท่าเกิน	95 mmขึ้นไป	แจ้งเตือนภัย Flash(Water),(H)

ปริมาณน้ำในดินเมื่อมีข้อมูลเข้ามา 50 cm ขึ้นไปแจ้งเตือนเป็นระดับสูงสุด

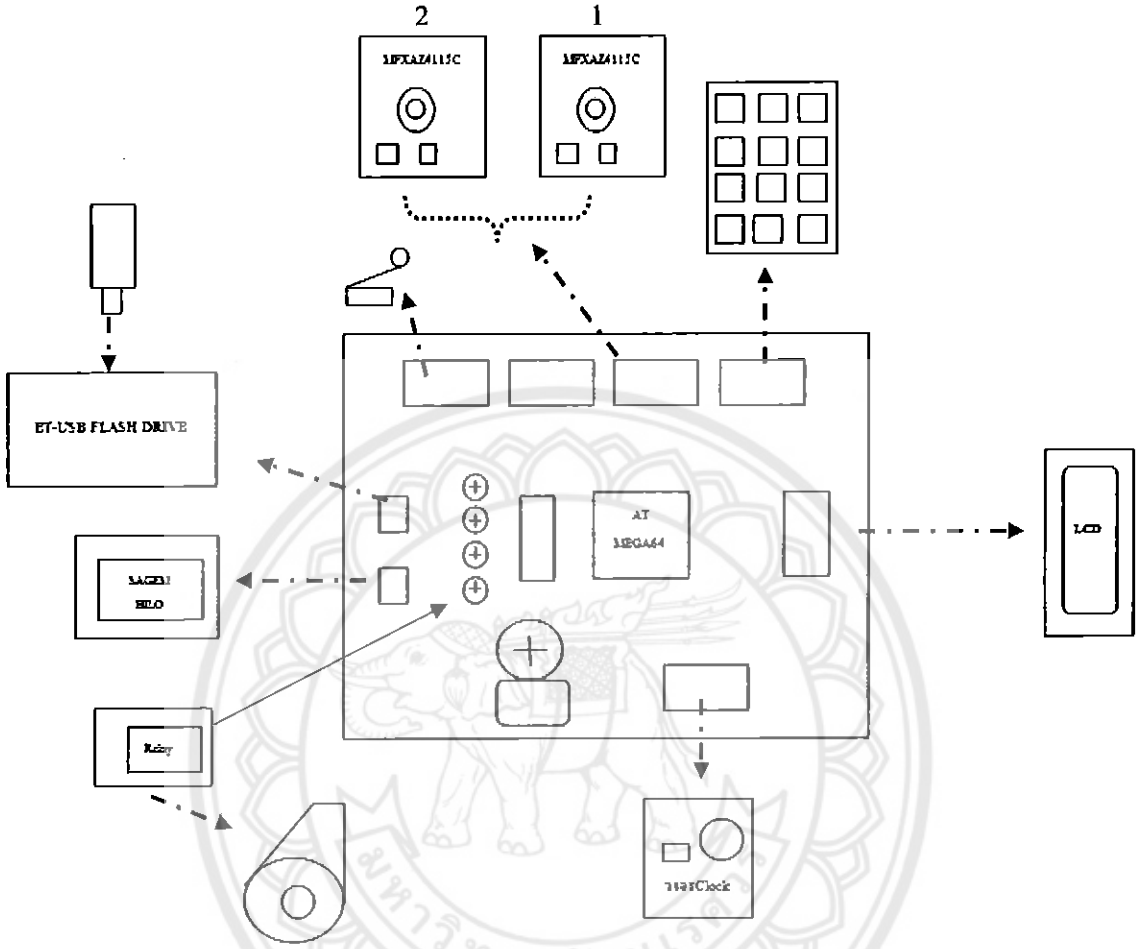
3.7 แผนผังการทำงาน



รูปที่ 3.11 ผังงานแสดงการทำงานโปรแกรมของเครื่องเตือนภัย

3.8 แผนผังการต่ออุปกรณ์

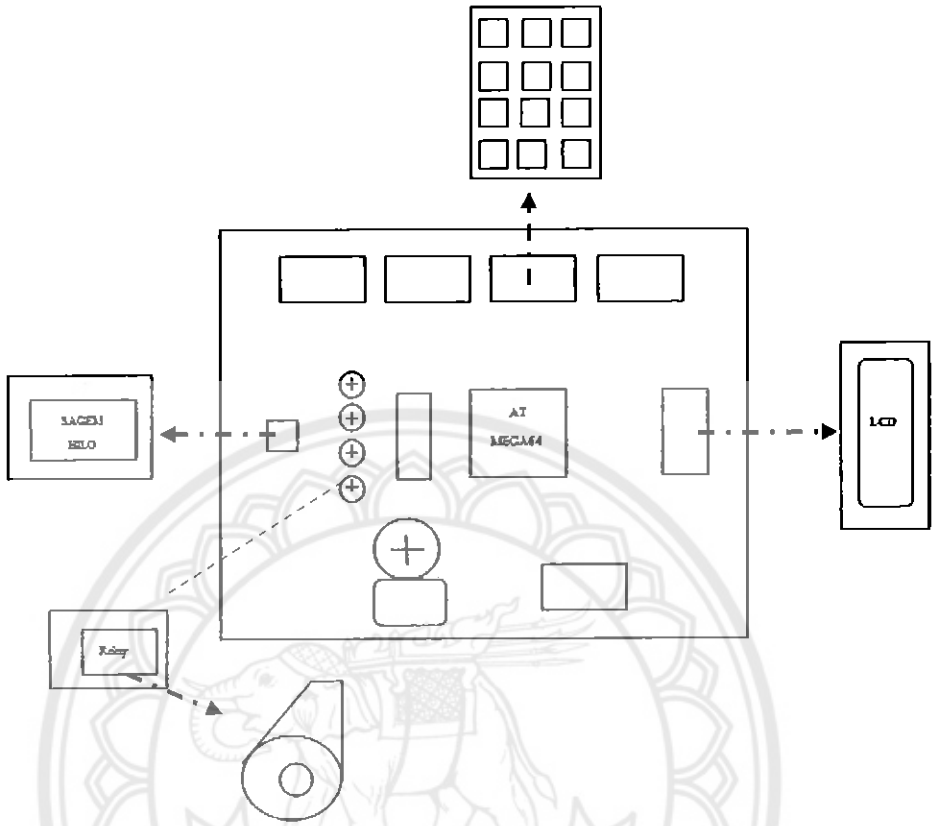
3.8.1 แผนผังการต่ออุปกรณ์ภาคส่ง



รูปที่ 3.12 แผนผังการต่ออุปกรณ์ภาคส่ง

จากภาพเราสามารถอธิบายการต่อวงจรได้อย่างคร่าวๆ คือบอร์ดไมโครคอนโทรลเลอร์เป็นตัวที่จะประมวลผลซึ่งจะอยู่ตรงกลาง จากนั้นก็จะมีอุปกรณ์ต่างที่ทำงานร่วมกันเข้ามาต่อจนเป็นเครื่องส่งภาคเตือนภัย โดยอุปกรณ์ที่จะเข้ามาต่อ ก็คือ จอหน้าจอแสดงผล , วงจรนาฬิกา , โมดูลโทรศัพท์ SAGEM HILO , ET-USB FLASH DRIVE , KEYPAD , เซนเซอร์ Motorola MPXAZ4115C (โดยหมายเลขที่ 1 จะวัดน้ำท่า และ หมายเลข 2 จะวัดปริมาณน้ำในดิน) , Flash drive , สวิทช์หน้าสัมผัส , Relay(จะต่อสายไฟออกมาจากใต้บอร์ดอยู่แล้ว) รวมทั้งไซเรน

3.8.2 แผนผังการต่ออุปกรณ์ภาครับ



รูปที่ 3.13 แผนผังการต่ออุปกรณ์ภาครับ

จากภาพเราสามารถอธิบายการต่อวงจรได้อย่างคร่าวๆ คือ บอร์ดไมโครคอนโทรลเลอร์เป็นตัวที่จะประมวลผล จากนั้นก็จะมีอุปกรณ์ต่างที่ทำงานร่วมกันเข้ามาต่อจนเป็นเครื่องรับภาคเตือนภัย โดยอุปกรณ์ที่จะเข้ามาต่อ ก็คือ จอหน้าจอแสดงผล, โมดูลโทรศัพท์ SAGEM HILO , KEYPAD , Relay(จะต่อสายไฟออกมาจากใต้บอร์ดอยู่แล้ว) รวมทั้งไซเรน

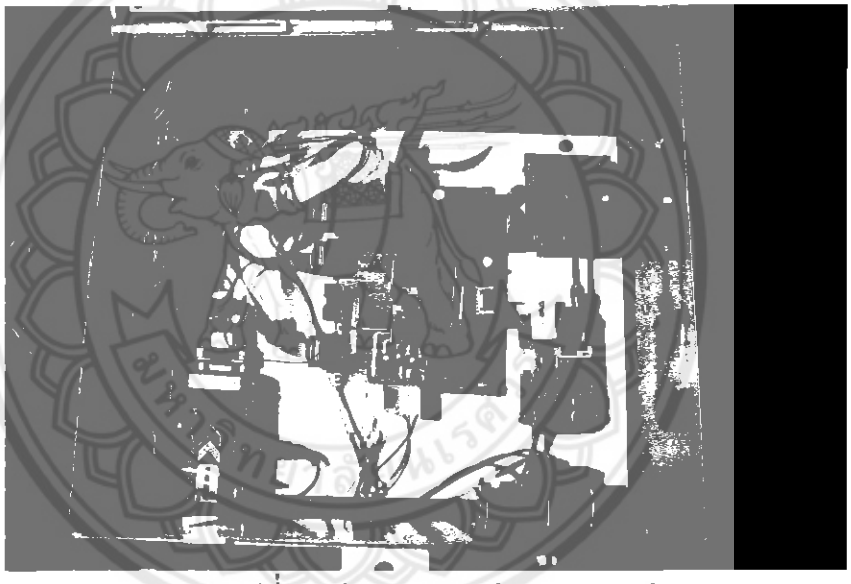
บทที่ 4

ผลการทดลอง

4.1 ระบบการทำงาน

สำหรับระบบการทำงานสามารถแบ่งออกเป็น 2 ส่วนหลักๆ คือ ภาคส่งและภาครับ

4.1.1 ระบบการทำงานภาคส่ง ประกอบไปด้วย ระบบรับสัญญาณข้อมูลอินพุท ระบบควบคุมการประมวลผล และส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือไปยังหมายเลขปลายทาง เมื่อต่ออุปกรณ์เข้าด้วยกันแล้ว จากรูปข้างล่างนั้นจะสามารถประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือได้



รูปที่ 4.1 ส่วนประกอบต่างๆของภาคส่ง

จากรูปที่ 4.1 ประกอบไปด้วยอุปกรณ์ ต่อไปนี้

- Sensor จะทำหน้าที่รับข้อมูลเป็นสัญญาณอนาล็อกเปลี่ยนเป็นดิจิตอลทันทีโดยจะรับค่าไว้ 2 ค่า คือ ค่าน้ำท่าและค่าน้ำในดิน

- ไมโครคอนโทรลเลอร์ ทำหน้าที่ ควบคุมการสั่งงานของโปรแกรม โดยรับอินพุท เข้ามาเก็บไว้ และทำการตัดสินใจ ส่งคำสั่งไปยัง โมดูลโทรศัพท์มือถือ โดยผ่านพอร์ต RS232 ที่เชื่อมต่อกัน

- โมดูลโทรศัพท์ ทำหน้าที่รับคำสั่ง จากไมโครคอนโทรลเลอร์ และทำการส่งข้อความไปยังหมายเลขปลายทาง

- EEPROM ทำหน้าที่โดย EEPROM เป็นหน่วยความจำที่สามารถเขียน - อ่านได้ และสามารถ เขียนซ้ำได้หลายๆ ครั้ง และเมื่อ ไม่มีแหล่งจ่ายไฟจ่ายให้กับหน่วยความจำ ข้อมูลก็จะไม่สูญหาย ประโยชน์ ที่ได้คือ เก็บค่า อินพุทที่รับเข้ามา และ มีขนาดข้อมูลที่เล็ก

- Relay ทำหน้าที่ เหมือนเป็นสวิทช์รับอินพุทจากไมโครคอนโทรลเลอร์เพื่อ เปิด-ปิด ไซเรน

- ET USB FLASH DRIVE ทำหน้าที่เก็บข้อมูลผ่าน พอร์ตRS232 โดยจะเก็บข้อมูลปริมาณน้ำฝน น้ำท่า และ ปริมาณน้ำในดิน

- Real Time Clock ทำหน้าที่เป็นตัวบอกเกี่ยวกับการแสดงเวลาทั้งหมด คือ จะแสดงค่า วัน/เดือน/ปี และ เวลา

รูปที่ 4.2 ส่วนแสดงผลของภาคส่ง

- จอหน้าจอแสดงผล ทำหน้าที่แจ้งสถานะ การรับอินพุท และ การส่งข้อความบนจอ LCD

- KEYPAD คือ อุปกรณ์ที่เกิดจากการนำสวิตซ์ หลายๆตัวมาต่อเข้าด้วยกันแบบเมตริกซ์ เสมือนเป็นคีย์บอร์ดขนาดเล็กอันหนึ่ง โดยใช้วิธีการสแกนทำโดยส่งลอจิก 0 ออกทีละแถว ถ้าไม่มีคีย์ไหนถูกกดคอลัมน์นั้นจะมีค่าเป็นลอจิก 1 แต่ถ้ามีการกดคอลัมน์จะมีค่าเป็น 0 โดยจะกำหนดหมายเลข KEYPAD ไว้ดังนี้

หมายเลข 0 คือ การเคลียร์ EEPROM ตอนเปิดเครื่อง (เมื่อเข้าโหมดการทำงานจะเป็นการเรียกดูค่าน้ำฝนสะสม จาก EEPROM)

หมายเลข 2 คือ โหมดการตั้งเวลา เมื่อเปิดเครื่องหรือเวลาที่เครื่องทำงานแล้ว สามารถตั้งเวลาได้ทันที โดยการตั้งเวลา จะเข้าไปโหมด เลือกวันที่ (วันจันทร์คือวันที่ 1 เรียงไปจนถึงวัน

อาทิตย์ คือ วันที่ 7) จากนั้น ก็จะเป็นการเลือก วัน/เดือน/ปี จากนั้นก็จะเป็นการเลือกเวลา ชั่วโมง/นาที/วินาที แล้วก็กด # เพื่อออกจากการทำงานปกติ

หมายเลข * คือ การเข้าโหมดการทำงาน

หมายเลข 4 คือ การ Test โหมด RF Low

หมายเลข 5 คือ การ Test โหมด RF Mid

หมายเลข 6 คือ การ Test โหมด RF Hi

หมายเลข 7 คือ การเคลียร์ข้อมูลทั้งหมด (โดย หมายเลข 7 นี้จะต้องกดติดต่อกัน 3 ครั้ง)

หมายเลข 8 คือ ยกเลิกการเตือนภัย

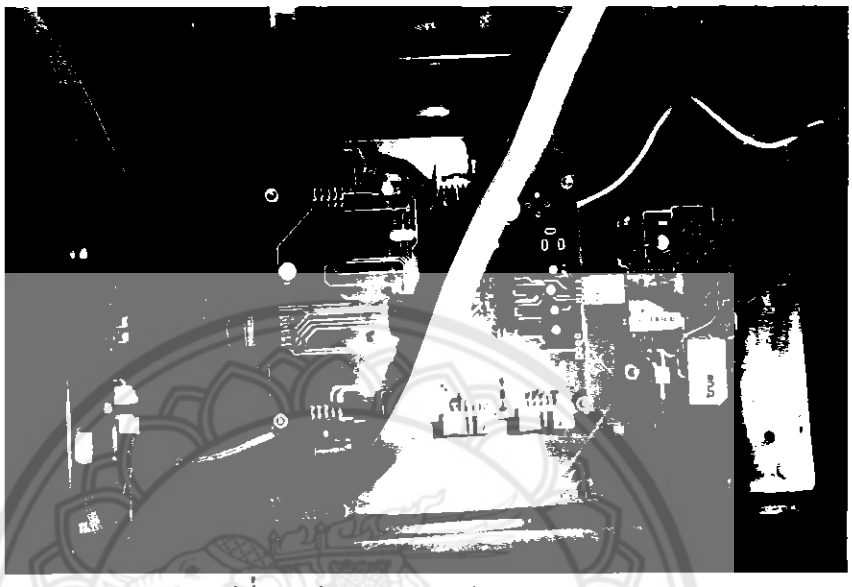
หมายเลข 9 คือ ดูค่าน้ำฝนสะสมทั้งหมด

หมายเลข # คือ การเคลียร์หน้าจอเมื่อเปิดเครื่อง (เมื่อเข้าโหมดการทำงานจะเป็นการเรียกดูค่าจาก USB 5 ค่าล่าสุดที่บันทึกลงไป)

รูปที่ 4.3 สวิทช์หน้าสัมผัส

- สวิทช์หน้าสัมผัส เป็นตัวรับสัญญาณอนาล็อกเป็นดิจิตอล โดยจะรับค่าปริมาณน้ำฝน (1ครั้ง/1มิลลิเมตร)

4.1.2 ระบบการทำงานภาครับ ประกอบไปด้วย ระบบควบคุมการรับสัญญาณเตือนภัยทางโทรศัพท์มือถือ จากยังหมายเลขต้นทาง เมื่อต่ออุปกรณ์เข้าด้วยกันแล้ว จากรูปข้างล่างนั้นจะสามารถอ่านดูข้อความแจ้งเตือนภัยได้

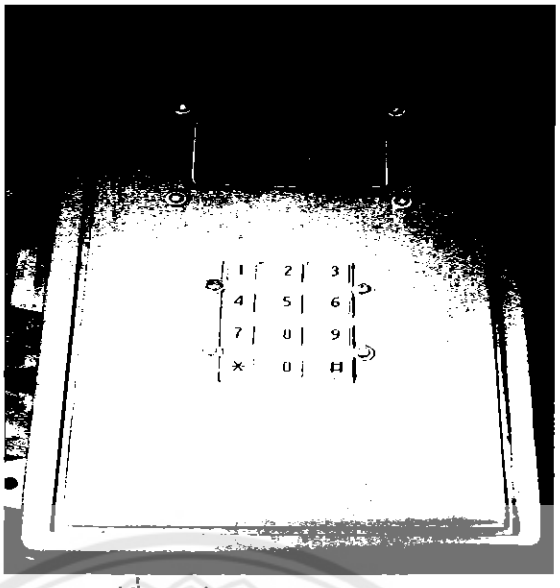


รูปที่ 4.4 ส่วนประกอบต่างๆของภาครับ

- โมดูลโทรศัพท์จะรับข้อความที่เข้ามาจากหมายเลขต้นทาง และส่งข้อมูลไปไมโครคอนโทรลเลอร์ โดยผ่านพอร์ต RS232 ที่เชื่อมต่อกัน

- ไมโครคอนโทรลเลอร์ จะนำข้อมูลที่ได้ สั่งให้โปรแกรมอ่านข้อความ แล้วโชว์ที่ LCD เพื่อดูข้อความ และทำการสั่งเตือนภัยตามเงื่อนไขที่กำหนดไว้

- Relay ทำหน้าที่เหมือนเป็นสวิทช์รับอินพุทจากไมโครคอนโทรลเลอร์เพื่อเปิด-ปิดไซเรน



รูปที่ 4.5 ส่วนแสดงผลของภาครับ

- จอหน้าจอแสดงผล ทำหน้าที่แจ้งสถานะ เพื่ออ่านข้อความที่รับเข้ามาและจะโชว์ที่หน้าจอหน้าจอแสดงผล

- คีย์แพด คือ อุปกรณ์ที่เกิดจากการนำสวิตซ์ หลายๆตัวมาต่อเข้าด้วยกันแบบเมตริกซ์เสมือนเป็นคีย์บอร์ดขนาดเล็กอันหนึ่ง โดยใช้วิธีการสแกนทำโดยส่งลอจิก 0 ออกทีละแถว ถ้าไม่มีคีย์ไหนถูกกดคอลัมน์นั้นจะมีค่าเป็นลอจิก 1 แต่ถ้ามีการกดคอลัมน์จะมีค่าเป็น 0 โดยจะกำหนด

หมายเลข คีย์แพด ไว้ดังนี้

หมายเลข 1 คือ คำสั่ง Back

หมายเลข 2 คือ การ Test ไซเรน

หมายเลข 3 คือ คำสั่ง Next

หมายเลข 8 คือ ยกเลิกการเตือนภัย

หมายเลข 0 คือ แสดงระดับการเตือนภัยที่ส่งเข้ามา

หมายเลข * คือ การเคลียร์ EEPROM

4.2 ผลการทดลอง

4.2.1 ผลการทดลองภาคส่ง

เมื่อรับอินพุทเข้ามา 3 อินพุท คือ น้ำฝน น้ำท่า และปริมาณน้ำในดิน จากนั้นจะทำการเก็บข้อมูลที่ส่งเข้ามาแล้วตัดสินใจ เมื่อข้อมูลที่ได้เกินกว่าค่าที่กำหนด ก็จะมีการส่งข้อความเตือนภัยไปยังหมายเลขปลายทาง ซึ่งจะมีหลักเกณฑ์ในการแจ้งเตือนดังนี้

4.2.2 ผลการทดลองภาครับ

เมื่อรับข้อความเตือนภัยแล้วสั่งให้แสดงข้อความผ่านจอ LCD โดยแสดงข้อความเช็คความพร้อมในการใช้งานทุกวันจันทร์ในเวลา 08.00 น. และแสดงเวลา วัน เดือน ปี ของข้อมูลที่ส่งเข้ามาสามารถตรวจสอบข้อมูลย้อนหลังได้ นอกจากนี้จะแสดงการเตือนภัยด้วยเสียงไซเรน เมื่อมีข้อมูลเข้ามาดังนี้

แจ้งเตือน Low ไซเรนเสียงดัง 10 วินาที หยุด 50 วินาที

แจ้งเตือน Middle ไซเรนเสียงดัง 20 วินาที หยุด 40 วินาที

แจ้งเตือน High ไซเรนเสียงดัง 30 วินาที หยุด 30 วินาที

4.3 กราฟและตารางแสดงผลการทดลอง

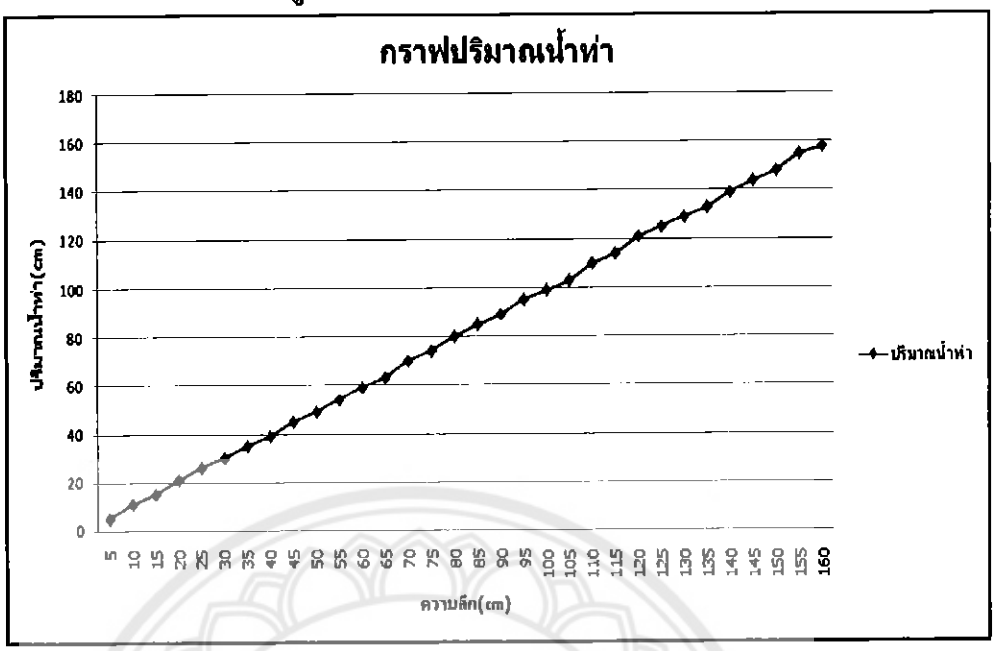
4.3.1 ผลการทดลองแสดงปริมาณน้ำท่า

ตารางที่ 4.6 แสดงผลปริมาณน้ำท่า

ความลึก (cm)	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
ปริมาณน้ำท่า (cm)	5	11	15	21	26	30	35	39	45	49	54	59	63	70	74	80	85	89

ความลึก (cm)	95	100	105	110	115	120	125	130	135	140	145	150	155	160
ปริมาณน้ำท่า (cm)	95	99	103	110	114	121	125	129	133	139	144	148	155	158

รูปที่ 4.7 กราฟแสดงผลปริมาณน้ำท่า



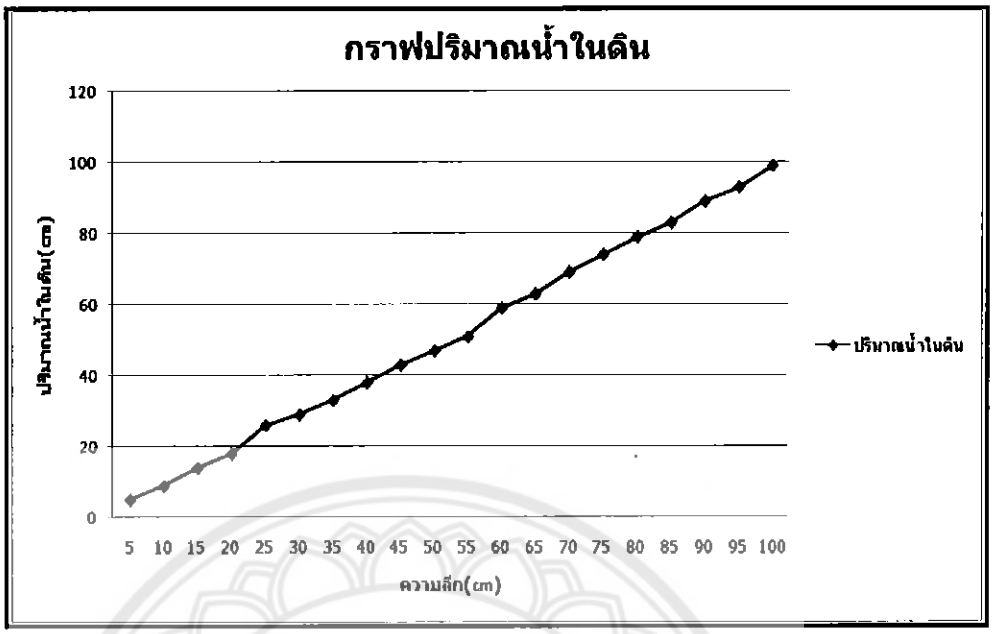
4.3.2 ผลการทดลองปริมาณน้ำในดิน

ตารางที่ 4.8 แสดงปริมาณน้ำในดิน

ความลึก (cm)	5	10	15	20	25	30	35	40	45	50
ปริมาณน้ำในดิน(cm)	5	9	14	18	26	29	33	38	43	47

ความลึก (cm)	55	60	65	70	75	80	85	90	95	100
ปริมาณน้ำในดิน(cm)	51	59	63	69	74	79	83	89	93	99

รูปที่ 4.9 กราฟแสดงปริมาณน้ำในดิน



4.3.3 ผลการทดลองน้ำฝนสะสม

ตารางที่ 4.10 แสดงปริมาณน้ำฝนใน 96 ชั่วโมง

เวลา (hr)	1	2	3	4	5	6	7	8	9	10	11	12
ปริมาณน้ำฝน (mm)	5	5	8	10	15	19	20	20	20	22	23	23

เวลา (hr)	13	14	15	16	17	18	19	20	21	22	23	24
ปริมาณน้ำฝน (mm)	25	25	25	30	35	45	80	85	87	87	88	88

เวลา (hr)	25	26	27	28	29	30	31	32	33	34	35	36
ปริมาณน้ำฝน (mm)	89	90	90	90	90	99	105	122	129	131	145	149

เวลา (hr)	37	38	39	40	41	42	43	44	45	46	47	48
ปริมาณน้ำฝน (mm)	155	155	162	165	170	172	175	179	180	185	189	190

เวลา (hr)	49	50	51	52	53	54	55	56	57	58	59	60
ปริมาณน้ำฝน (mm)	190	190	192	192	192	192	193	193	194	194	194	195

เวลา (hr)	61	62	63	64	65	66	67	68	69	70	71	72
ปริมาณน้ำฝน (mm)	195	195	195	195	196	197	197	199	199	199	201	201

เวลา (hr)	73	74	75	76	77	78	79	80	81	82	83	84
ปริมาณน้ำฝน (mm)	202	202	205	205	205	207	212	212	215	216	216	217

เวลา (hr)	85	86	87	88	89	90	91	92	93	94	95	96
ปริมาณน้ำฝน (mm)	217	220	224	224	225	225	225	226	226	226	226	226

ตัวอย่าง จากตารางผลการทดลองพบว่า

กรณีที่ 1 เงื่อนไขใน 1 ชั่วโมง ตัวอย่างเช่น ชั่วโมงที่ 18 – 19 จะพบว่าในช่วงเวลานี้ปริมาณน้ำฝนต่างกันตามเงื่อนไข คือ 35 มิลลิเมตร ทำให้เครื่องจะทำการส่ง SMS และเตือนภัยผ่านไซเรน โดยเงื่อนไขของปริมาณน้ำฝน 1 ชั่วโมง

คือผลต่างของน้ำฝน = ชั่วโมงปัจจุบัน – ชั่วโมงที่แล้ว

กรณีที่ 2 เงื่อนไขใน 24 ชั่วโมง ตัวอย่างเช่น เราจะดูปริมาณน้ำฝนชั่วโมง 39 โดยถ้าจะให้เป็นเงื่อนไขปริมาณน้ำฝน 24 ชั่วโมง ก็คือต้องย้อนไป หลังจากนั้น 24 ชั่วโมง นั่นคือเราจะต้องไปเช็คที่ชั่วโมงที่ 15 จากนั้น ก็คิดผลต่างน้ำฝนซึ่งถ้าเป็นไปตามเงื่อนไขของ 24 ชั่วโมง ก็จะทำการแจ้งเตือนในที่นี้ ผลต่างที่ได้เท่ากับ 137 มิลลิเมตร ซึ่งจะทำให้เครื่องทำการส่ง SMS และเตือนภัยผ่านไซเรน โดยเงื่อนไขของปริมาณน้ำฝน 24 ชั่วโมง คือ จะหาได้เป็น 2 กรณีคือ

กรณีที่ 2.1 คิดที่จำนวนชั่วโมงน้อยกว่า 24 ชั่วโมง จะได้สมการคือ

{จำนวนชั่วโมง(1 - 23) – 24} + 72 = จำนวนชั่วโมงที่ต้องการหาผลต่าง

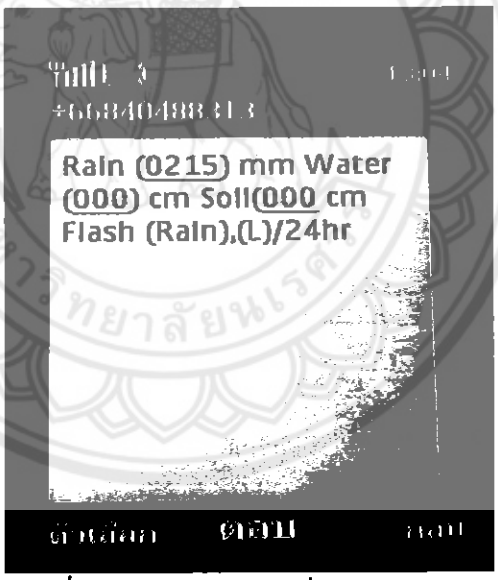
กรณีที่ 2.2 คิดที่จำนวนชั่วโมงที่มากกว่า 24 ชั่วโมง จะได้สมการคือ

จำนวนชั่วโมง – 24 = จำนวนชั่วโมงที่ต้องการหาผลต่าง

กรณีที่ 3 เงื่อนไขใน 72 ชั่วโมง ตัวอย่างเช่น เราดูปริมาณน้ำฝนชั่วโมงที่ 84 โดยถ้าจะให้เป็นเงื่อนไขปริมาณน้ำฝน 72 –ชั่วโมง ก็คือต้องย้อนไปเช็คได้หลังจากนั้น 72 ชั่วโมง นั้นคือเราจะต้องไปเช็คที่ชั่วโมงที่12 จากนั้นก็จะคิดผลต่างน้ำฝนซึ่งถ้าเป็นไปตามเงื่อนไขของ 72 ชั่วโมง ก็จะทำการแจ้งเตือน ในที่นี้ ผลต่างที่ได้เท่ากับ 194 มิลลิเมตร ก็จะทำการส่ง SMS และเตือนภัยผ่านไซเรน โดยเงื่อนไขของปริมาณน้ำฝน 72 ชั่วโมง คือ หาได้จากสมการ คือ

ค่าน้ำฝนปัจจุบัน – ค่าน้ำฝนชั่วขณะ = เงื่อนไขน้ำฝน 72 ชั่วโมง

รูปที่ 4.11 การส่งSMSจากเครื่องส่ง



รูปที่ 4.12 ข้อความจากเครื่องผู้ดูแลระบบ

บทที่ 5
บทสรุป

ในบทนี้จะกล่าวถึง การสรุปผลการทดลองในโครงงานนี้ทั้งหมด พร้อมทั้งวิเคราะห์ปัญหาที่เกิดขึ้นจากการทดลอง ตลอดจนข้อเสนอแนะเพิ่มเติมในการนำโครงงานนี้ไปพัฒนาต่อ

5.1 สรุปผลการดำเนินโครงงาน

จากโครงงานโครงงานการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ ทำให้ได้ข้อสรุปจากผลการดำเนินงานดังนี้

จากผลการทดลองกรณีที่ 1 ให้ระดับน้ำท่าต่ำกว่า 50 cm.

ใน 1 ชั่วโมง ปริมาณน้ำฝนเกิน 35 mm. แจ้งเตือนภัยเป็น Low ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 50 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 70 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ในเวลา 1 วัน ปริมาณน้ำฝนเกิน 90 mm. แจ้งเตือนภัยเป็น Low ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 100 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 110 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ในเวลา 3 วัน ปริมาณน้ำฝนเกิน 185 mm. แจ้งเตือนภัยเป็น Low ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 200 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 215 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้

จากผลการทดลองกรณีที่ 2 ให้ระดับน้ำท่ามากกว่า 50 cm.

ใน 1 ชั่วโมง ปริมาณน้ำฝนเกิน 35 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 50 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 70 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ในเวลา 1 วัน ปริมาณน้ำฝนเกิน 90 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 100 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 110 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ในเวลา 3 วัน ปริมาณน้ำฝนเกิน 185 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 200 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้
ปริมาณน้ำฝนเกิน 215 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้

กรณีน้ำท่า

ระดับน้ำท่าเกิน 65 mm. แจ้งเตือนภัยเป็น Low ได้ตามเงื่อนไขที่กำหนดไว้

ระดับน้ำท่าเกิน 80 mm. แจ้งเตือนภัยเป็น Mid ได้ตามเงื่อนไขที่กำหนดไว้

ระดับน้ำท่าเกิน 95 mm. แจ้งเตือนภัยเป็น Hi ได้ตามเงื่อนไขที่กำหนดไว้

และกรณีความชื้นของดิน เมื่อมีข้อมูลเข้ามามากกว่า 50 เซนติเมตรขึ้นไปแจ้งเตือนภัยเป็น Hi สรุป เครื่องเตือนภัยสามารถแจ้งเตือนภัยได้ทำงานตามเงื่อนไขที่กำหนดไว้ได้ทุกกรณี

5.2 ปัญหาที่เกิดขึ้นและแนวทางการแก้ไข

จากการทำโครงงานการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือพบปัญหาและอุปสรรคต่างๆ ดังนี้

1. การดำเนินงานในด้านโปรแกรม บางครั้งเกิดปัญหาข้อผิดพลาดในด้านการพัฒนาโปรแกรม แก้ไขโดยศึกษาด้วยตนเองและสอบถามผู้ที่เชี่ยวชาญในด้านนี้

2. ระบบมีระยะเวลาการทำงานจำกัด ขึ้นอยู่กับโปรโมชั่นของโทรศัพท์มือถือ แก้ไขโดยการเลือกเครือข่ายบริการที่ให้ระยะเวลาการใช้งานได้นาน

3. ET-USB Flash Driveที่ใช้บันทึกข้อมูล มีปัญหากับ Flash Drive บางรุ่น ทำให้ไม่สามารถเก็บข้อมูลได้ตามต้องการ

5.3 แนวทางในการพัฒนาต่อไป

1.ในอนาคตอาจเพิ่มอุปกรณ์ตรวจรอยเลื่อนของดิน เพื่อนำมาเป็นข้อมูลในการแจ้งเตือนดินถล่ม

2.ในกรณีที่เครือข่ายไม่พร้อมให้บริการจะไม่สามารถส่ง SMSได้ ควรพัฒนาด้านโปรแกรมให้ระบบ สามารถส่ง SMS ได้เมื่อเครือข่ายพร้อมให้บริการ

เอกสารอ้างอิง

[1] ประจิน พลังสันติกุล. (2549.). การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR ด้วยภาษา C กับ WinAVR (C Compiler). กรุงเทพฯ: บริษัท แอพซอฟต์เทค จำกัด

[2] ประจิน พลังสันติกุล. (2551). การประยุกต์ใช้งานภาษา C สำหรับไมโครคอนโทรลเลอร์ AVR เล่ม 2. กรุงเทพฯ: บริษัท แอพซอฟต์เทค จำกัด

[3] คู่มือการใช้งาน manual-ET-USB20-Flash-Drive.pdf : http://www.ett.co.th

[4] คู่มือการใช้งาน MPXAZ6115A.pdf : http://www.datashcetcatalog.com

[5] คู่มือการใช้งาน ATMEGA64.pdf : http://www.ett.co.th

[6] คู่มือการใช้งาน GSM Module SAGEM-HILO.pdf : http://www.sagemcom.com

ภาคผนวก ก
ตารางข้อมูลของอุปกรณ์ในวงจร

ก. 1 ตารางข้อมูลของ ATmega

Features

- High-performance, Low-power Atmel AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 64 Kbytes of In-System Reprogrammable Flash program memory
  - 2 Kbytes EEPROM
  - 4 Kbytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Up to 64 Kbytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 1 to 16 Bits
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels
    2 Differential Channels with Programmable Gain (1x, 10x, 200x)
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for Atmel ATmega64L
  - 4.5V - 5.5V for Atmel ATmega64
- Speed Grades
  - 0 - 8 MHz for ATmega64L
  - 0 - 16 MHz for ATmega64

ATMEL

8-bit AVR® Microcontroller with 64K Bytes In-System Programmable Flash

ATmega64
ATmega64L

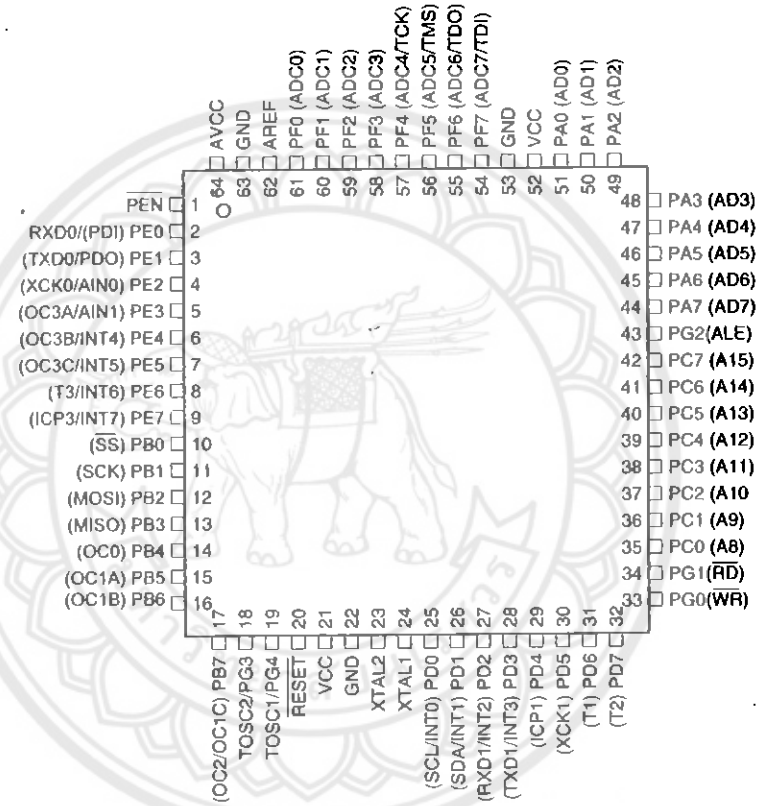
2490Q-AVR-06/10

ATMEL

Pin Configuration

Figure 1. Pinout ATmega64

TQFP/MLF



Note: The bottom pad under the QFN/MLF package should be soldered to ground.

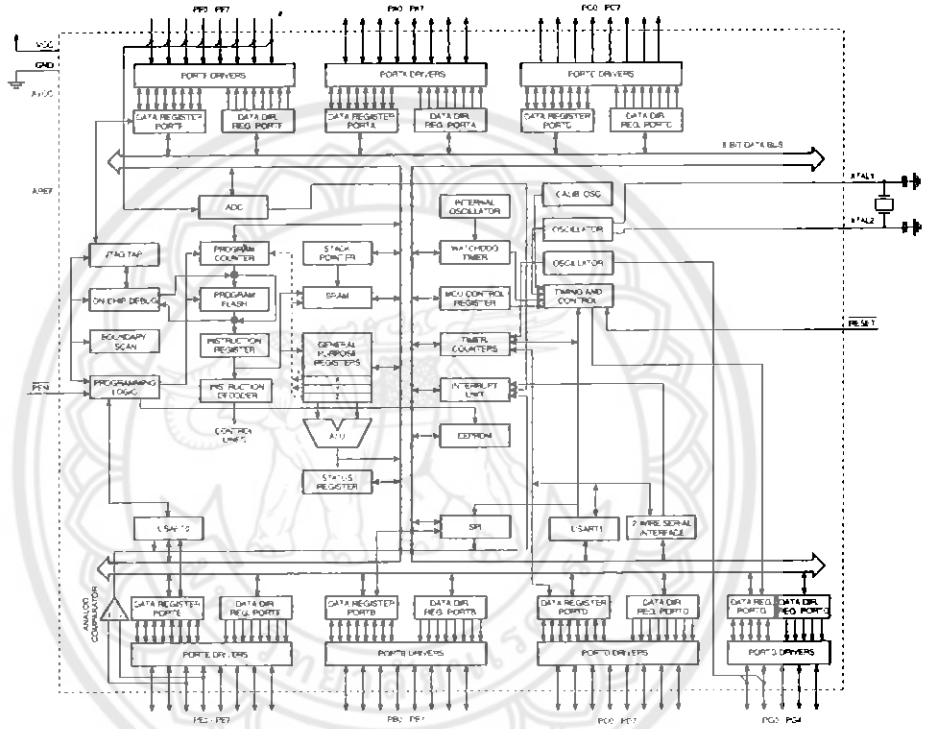
Disclaimer Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega64 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega64 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega64 provides the following features: 64 Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 2 Kbytes EEPROM, 4 Kbytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, two USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and Interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high-density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot Program can use any interface to download the Application Program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega64 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega64 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

ATmega103 and ATmega64 Compatibility

The ATmega64 is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O location reserved in the AVR instruction set. To ensure backward compatibility with the ATmega103, all I/O locations present in ATmega103 have the same location in ATmega64. Most additional I/O locations are added in an Extended I/O space starting from 0x60 to 0xFF (that is, in the ATmega103 internal RAM space). These location can be reached by using LD/LDS/LDD and ST/STS/STD instructions only, not by using IN and OUT instructions. The relocation of the internal RAM space may still be a problem for ATmega103 users. Also, the increased number of Interrupt Vectors might be a problem if the code uses absolute addresses. To solve these problems, an ATmega103 compatibility mode can be selected by programming the fuse M103C. In this mode, none of the functions in the Extended I/O space are in use, so the internal RAM is located as in ATmega103. Also, the extended Interrupt Vectors are removed.

The ATmega64 is 100% pin compatible with ATmega103, and can replace the ATmega103 on current printed circuit boards. The application notes "Replacing ATmega103 by ATmega128" and "Migration between ATmega64 and ATmega128" describes what the user should be aware of replacing the ATmega103 by an ATmega128 or ATmega64.

ATmega103 Compatibility Mode

By programming the M103C Fuse, the ATmega64 will be compatible with the ATmega103 regards to RAM, I/O pins and Interrupt Vectors as described above. However, some new features in ATmega64 are not available in this compatibility mode, these features are listed below:

- One USART instead of two, asynchronous mode only. Only the eight least significant bits of the Baud Rate Register is available.
- One 16 bits Timer/Counter with two compare registers instead of two 16 bits Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port G serves alternate functions only (not a general I/O port).
- Port F serves as digital input only in addition to analog input to the ADC.
- Boot Loader capabilities is not supported.
- It is not possible to adjust the frequency of the internal calibrated RC Oscillator.
- The External Memory Interface can not release any Address pins for general I/O, neither configure different wait states to different External Memory Address sections.
- Only EXTRF and PORF exist in the MCUCSR Register.
- No timed sequence is required for Watchdog Timeout change.
- Only low-level external interrupts can be used on four of the eight External Interrupt sources.
- Port C is output only.
- USART has no FIFO buffer, so Data OverRun comes earlier.
- The user must have set unused I/O bits to 0 in ATmega103 programs.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega64 as listed on page 73.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega64 as listed on page 74.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega64 as listed on page 77. In ATmega103 compatibility mode, Port C is output only, and the port C pins are not tri-stated when a reset condition becomes active.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega64 as listed on page 78.

Port E (PE7..PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega64 as listed on page 81.

Port F (PF7..PF0)

Port F serves as the analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS) and PF4(TCK) will be activated even if a reset occurs.

The TDO pin is tri-stated unless TAP states that shift out data are entered.

Port F also serves the functions of the JTAG interface.

In ATmega103 compatibility mode, Port F is an input port only.

Port G (PG4..PG0)

Port G is a 5-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features.

In ATmega103 compatibility mode, these pins only serves as strobes signals to the external memory as well as input to the 32 kHz Oscillator, and the pins are initialized to PG0 = 1, PG1 = 1, and PG2 = 0 asynchronously when a reset condition becomes active, even if the clock is not running. PG3 and PG4 are Oscillator pins.

RESET	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 19 on page 52. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.
PEN	This is a programming enable pin for the SPI Serial Programming mode. By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. PEN is internally pulled high. The pullup is shown in Figure 22 on page 52 and its value is given in Section "DC Characteristics" on page 325. $\overline{\text{PEN}}$ has no function during normal operation.

ก. 2 ตารางข้อมูลของ MPXAZ 6115 A

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document
by MPXAZ6115A/D

Media Resistant and High Temperature Accuracy Integrated Silicon Pressure Sensor for Measuring Absolute Pressure, On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPXAZ6115A
MPXHZ6115A
SERIES

INTEGRATED PRESSURE SENSOR
15 to 115 kPa (2.2 to 16.7 psi)
0.2 to 4.8 Volts Output

Motorola's MPXAZ6115A series sensor integrates on-chip, bipolar op amp circuitry and thin film resistor networks to provide a high output signal and temperature compensation. The sensor's packaging has been designed to provide resistance to high humidity conditions as well as common automotive media. The small form factor and high reliability of on-chip integration make the Motorola pressure sensor a logical and economical choice for the system designer.

The MPXAZ6115A series piezoresistive transducer is a state-of-the-art, monolithic, signal conditioned, silicon pressure sensor. This sensor combines advanced micromachining techniques, thin film metallization, and bipolar semiconductor processing to provide an accurate, high level analog output signal that is proportional to applied pressure.

Figure 1 shows a block diagram of the internal circuitry integrated on a pressure sensor chip.

Features

- Resistant to High Humidity and Common Automotive Media
- Improved Accuracy at High Temperature
- 1.5% Maximum Error over 0° to 85°C
- Ideally suited for Microprocessor or Microcontroller-Based Systems
- Temperature Compensated from -40° to +125°C
- Durable Thermoplastic (PPS) Surface Mount Package

Application Examples

- Aviation Altimeters
- Industrial Controls
- Engine Control/Manifold Absolute Pressure (MAP)
- Weather Station and Weather Reporting Devices

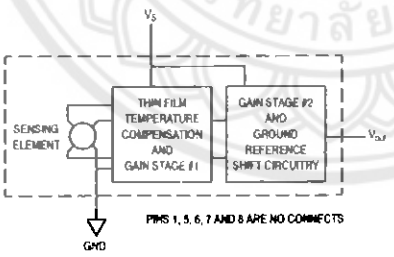


Figure 1. Fully Integrated Pressure Sensor Schematic

SMALL OUTLINE PACKAGE

MPXAZ6115A6U
CASE 482

MPXAZ6115AC6U
CASE 482A

PIN NUMBER			
1	N/C	5	N/C
2	V_S	6	N/C
3	Gnd	7	N/C
4	V_{out}	8	N/C

NOTE: Pins 1, 5, 6, 7, and 8 are internal device connections. Do not connect to external circuitry or ground. Pin 1 is denoted by the notch in the lead.

SUPER SMALL OUTLINE PACKAGE

MPXHZ6115A6U
CASE 1317

MPXAZ6115AC6U
CASE 1317A

REV 1

MOTOROLA intelligence everywhere | digital dna

MPXAZ6115A MPXHZ6115A SERIES

MAXIMUM RATINGS[1]

Parametrics	Symbol	Value	Units
Maximum Pressure (P1 > P2)	P_{max}	400	kPa
Storage Temperature	T_{stg}	-40° to +125°	°C
Operating Temperature	T_A	-40° to +125°	°C
Output Source Current @ Full Scale Output[2]	I_{o^+}	0.5	mAdc
Output Sink Current @ Minimum Pressure Offset[2]	I_{o^-}	-0.5	mAdc

NOTES:
1. Exposure beyond the specified limits may cause permanent damage or degradation to the device.
2. Maximum Output Current is controlled by effective impedance from V_{out} to Gnd or V_{out} to V_S in the application circuit.

OPERATING CHARACTERISTICS (V_S = 5.0 Vdc, T_A = 25°C unless otherwise noted, P1 > P2.)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range		P_{OP}	15	—	115	kPa
Supply Voltage[1]		V_S	4.75	5.0	5.25	Vdc
Supply Current		I_o	—	6.0	10	mAdc
Minimum Pressure Offset[2] @ V_S = 5.0 Volts	(0 to 85°C)	V_{off}	0.133	0.200	0.268	Vdc
Full Scale Output[3] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSO}	4.633	4.700	4.768	Vdc
Full Scale Span[4] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSS}	4.433	4.500	4.568	Vdc
Accuracy[5]	(0 to 85°C)	—	—	—	±1.5	$\%V_{FSS}$
Sensitivity		V/P	—	45.9	—	mV/kPa
Response Time[6]		t_R	—	1.0	—	ms
Warm-Up Time[7]		—	—	20	—	ms
Offset Stability[8]		—	—	±0.25	—	$\%V_{FSS}$

NOTES:
1. Device is ratiometric within this specified excitation range.
2. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
3. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
4. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
5. Accuracy is the deviation in actual output from nominal output over the entire pressure range and temperature range as a percent of span at 25°C due to all sources of error including the following:
   - Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from minimum or maximum rated pressure at 25°C.
   - TcSpan: Output deviation over the temperature range of 0° to 85°C, relative to 25°C.
   - TcOffset: Output deviation with minimum pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.
6. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
7. Warm-up Time is defined as the time required for the product to meet the specified output voltage after the pressure has been stabilized.
8. Offset Stability is the product's output deviation when subjected to 1000 cycles of Pulsed Pressure, Temperature Cycling with Bias Test.

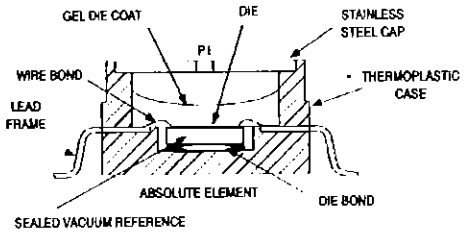


Figure 2. Cross Sectional Diagram SOP (Not to Scale)

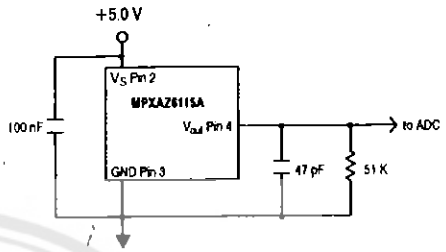


Figure 3. Typical Application Circuit (Output Source Current Operation)

Figure 2 illustrates the absolute sensing chip in the basic Small Outline chip carrier (Case 482).

Figure 3 shows a typical application circuit (output source current operation).

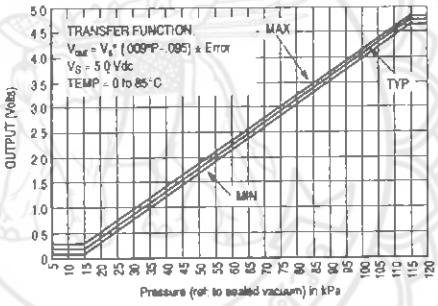


Figure 4. Output versus Absolute Pressure

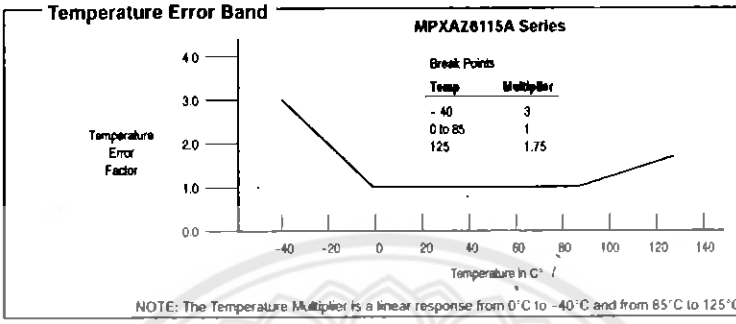
Figure 4 shows the sensor output signal relative to pressure input. Typical minimum and maximum output curves are shown for operation over 0 to 85°C temperature range. The output will saturate outside of the rated pressure range.

A gel die coat isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the sensor diaphragm. The gel die coat and durable polymer package provide a media resistant barrier that allows the sensor to operate reliably in high humidity conditions as well as environments containing common automotive media. Contact the factory for more information regarding media compatibility in your specific application.

Transfer Function (MPXAZ6115A)

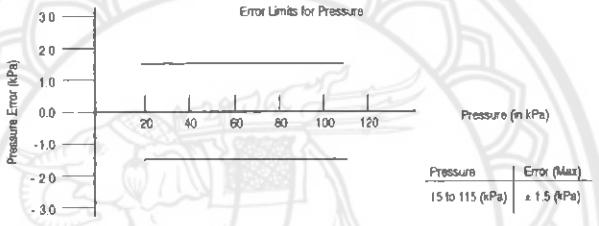
Nominal Transfer Value: $V_{out} = V_S \times (0.009 \times P - 0.095)$
$\pm$ (Pressure Error x Temp. Factor x $0.009 \times V_S$)
$V_S = 5.0 \pm 0.25$ Vdc

Temperature Error Band



NOTE: The Temperature Multiplier is a linear response from 0°C to -40°C and from 85°C to 125°C

Pressure Error Band



ORDERING INFORMATION — SMALL OUTLINE PACKAGE

Device Type	Options	Case No.	MPX Series Order No.	Packing Options	Marking
Basic Element	Absolute, Element Only	482	MPXAZ6115A6U	Rails	MPXAZ6115A
	Absolute, Element Only	482	MPXAZ6115A6T1	Tape and Reel	MPXAZ6115A
Ported Element	Absolute, Axial Port	482A	MPXAZ6115AC6U	Rails	MPXAZ6115A
	Absolute, Axial Port	482A	MPXAZ6115AC6T1	Tape and Reel	MPXAZ6115A

ORDERING INFORMATION — SUPER SMALL OUTLINE PACKAGE

Device Type	Options	Case No.	MPX Series Order No.	Packing Options	Marking
Basic Element	Absolute, Element Only	1317	MPXHZ6115A6U	Rails	MPXHZ6115A
	Absolute, Element Only	1317A	MPXHZ6115A6T1	Tape and Reel	MPXHZ6115A

SURFACE MOUNTING INFORMATION

MINIMUM RECOMMENDED FOOTPRINT FOR SMALL OUTLINE PACKAGE

Surface mount board layout is a critical portion of the total design. The footprint for the semiconductor package must be the correct size to ensure proper solder connection interface between the board and the package. With the correct pad geometry, the packages will self-align when subjected to a solder reflow process. It is always recommended to fabricate boards with a solder mask layer to avoid bridging and/or shorting between solder pads, especially on tight tolerances and/or tight layouts.

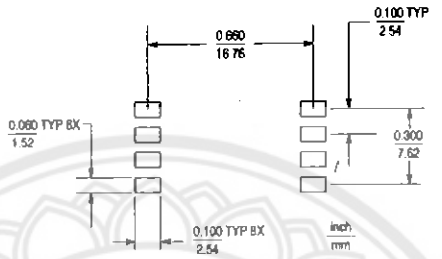


Figure 5. SOP Footprint (Case 482 and 482A)

MINIMUM RECOMMENDED FOOTPRINT FOR SUPER SMALL OUTLINE PACKAGES

Surface mount board layout is a critical portion of the total design. The footprint for the semiconductor package must be the correct size to ensure proper solder connection interface between the board and the package. With the correct pad geometry, the packages will self-align when subjected to a solder reflow process. It is always recommended to fabricate boards with a solder mask layer to avoid bridging and/or shorting between solder pads, especially on tight tolerances and/or tight layouts.

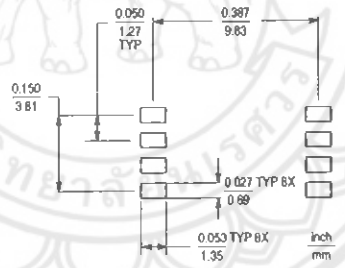
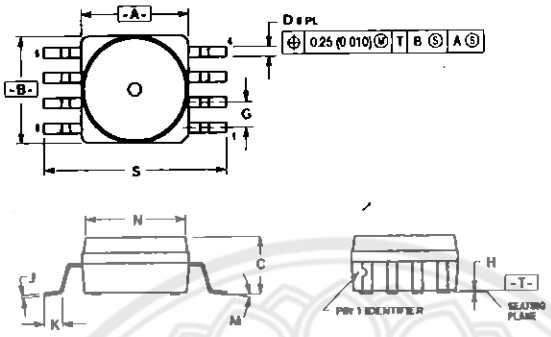


Figure 6. SSOP Footprint (Case 1317 and 1317A)

MPXAZ6115A MPXHZ6115A SERIES

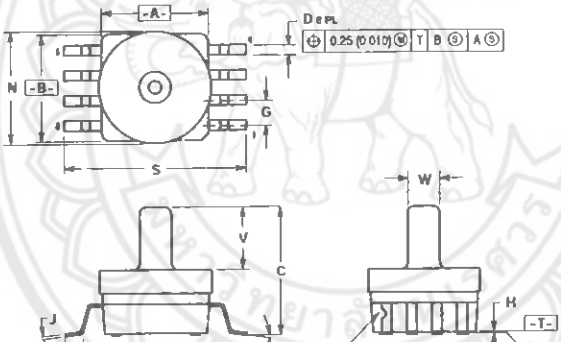
SMALL OUTLINE PACKAGE DIMENSIONS



NOTES
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006).
5. ALL VERTICAL SURFACES 5° TYPICAL DRAFT.

DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.415	0.425	10.54	10.79
B	0.415	0.425	10.54	10.79
C	0.212	0.230	5.38	5.84
D	0.038	0.042	0.96	1.07
G	0.100 BSC		2.54 BSC	
H	0.002	0.010	0.05	0.25
J	0.009	0.011	0.23	0.28
K	0.061	0.071	1.55	1.80
M	0°	7°	0°	7°
N	0.405	0.415	10.29	10.54
S	0.709	0.725	18.01	18.41

CASE 482-01
ISSUE O

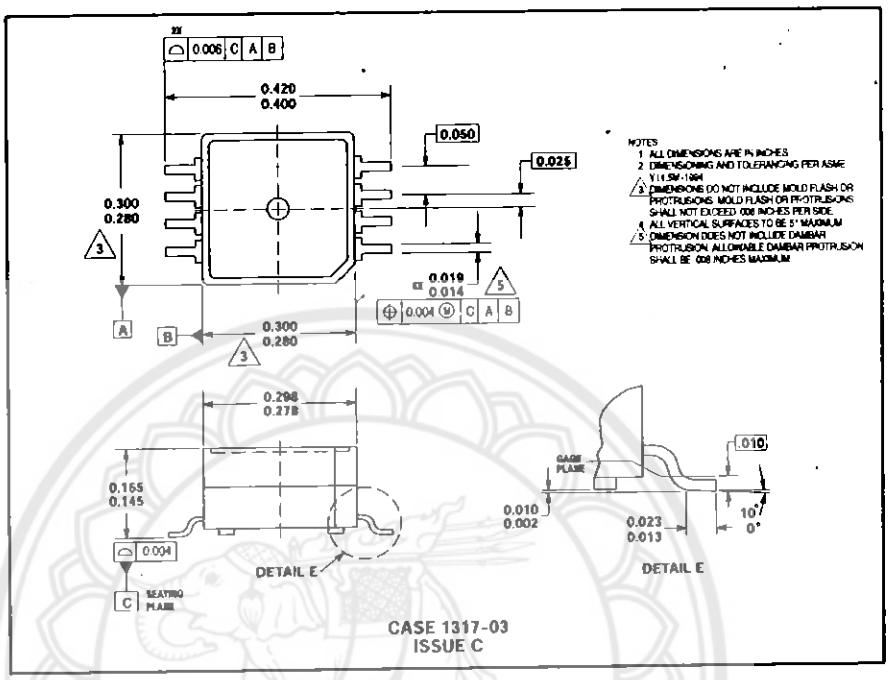


NOTES
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006).
5. ALL VERTICAL SURFACES 5° TYPICAL DRAFT.

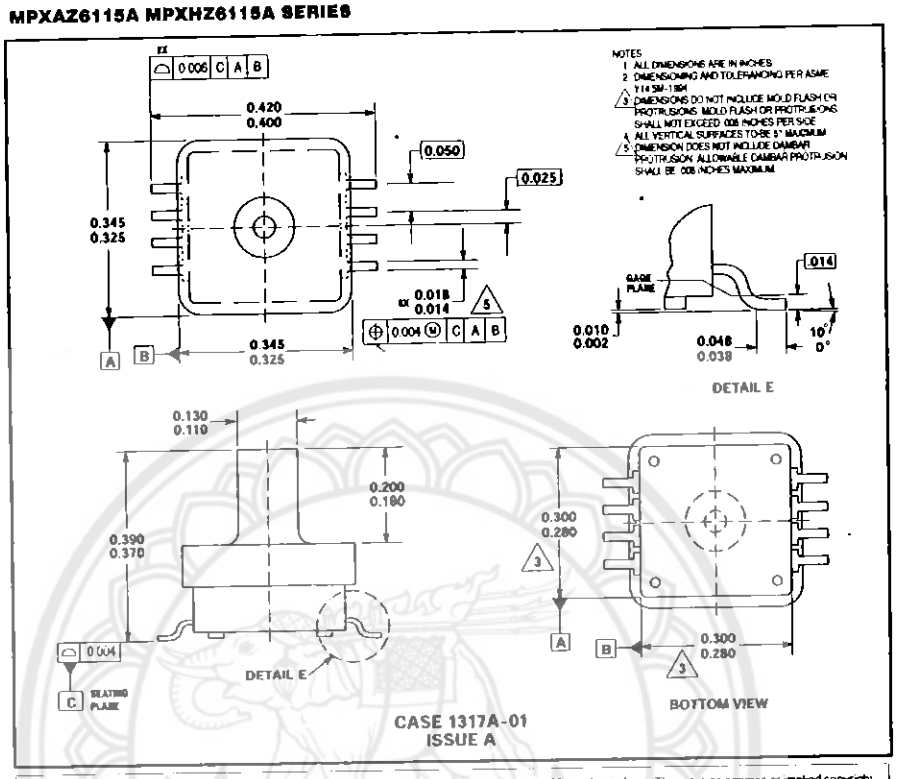
DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.415	0.425	10.54	10.79
B	0.415	0.425	10.54	10.79
C	0.500	0.530	12.70	13.21
D	0.038	0.042	0.96	1.07
G	0.100 BSC		2.54 BSC	
H	0.002	0.010	0.05	0.25
J	0.009	0.011	0.23	0.28
K	0.061	0.071	1.55	1.80
M	5°	7°	0°	7°
N	0.444	0.448	11.28	11.38
S	0.709	0.725	18.01	18.41
V	0.245	0.255	6.22	6.48
W	0.115	0.125	2.92	3.17

CASE 482A-01
ISSUE A

SUPER SMALL OUTLINE PACKAGE DIMENSIONS



CASE 1317-03
ISSUE C



HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:
Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573, Japan
81-3-3440-3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre,
2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

HOME PAGE: http://motorola.com/semiconductors

ภาคผนวก ข

รหัสต้นฉบับ (Source Code) ที่ใช้ในโปรแกรม Code Vision AVR

ข.1 รหัสต้นฉบับของภาคส่ง

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.8b Professional
Automatic Program Generator
© Copyright 1998-2006 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/10/2010
Author  : F4CG
Company : F4CG
Comments:

Chip type           : ATmega64
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024
*****************************************************/
#include <mega64.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>

#define  START_READ  835
char first_send=0;
char time_buff[12],index_time=0;
```

```
char  er,st,dummy[8];                    //ประกาศตัวแปรชื่อ Dummy เอาไว้ถ่ายค่าจาก Tempไป
เป็น Integer
 eeprom   int  VAL72 ;       //การเก็บค่าของทั้งหมด 72 ค่า

   char ALARM2=0;
 int cnt_round=0;                  //นับจำนวนรอบที่ Alarm ไปแล้ว
 int  cnt_error=0;                 // นับจำนวน Error

 //********************************** การ Ssan key pad
 ****************************
 #define COL1 PORTA.0
 #define COL2 PORTA.1
 #define COL3 PORTA.2
 #define SW1  PINB.0
 #define ROW1 PINA.3
 #define ROW2 PINA.4
 #define ROW3 PINA.5
 #define ROW4 PINA.6
 #define BUZZER  PORTB.1            // Buzer Port
 #define  DEBUG_232  0

 int ss,i,cnt_read,cnt_save=0;int ADC_temp=0,ADC_old=0;   //ประกาศตัวแปรเป็น integer
 int  time1,round,s_old;
 int cnt_reset,time_clear_reset,sec_cnt=0;
 char Status_rf=0,Status_wt=0,Status_sm=0;

 int round_Set,S1,S2,S3;
 int s_old_buzzer;
 int cnt_buzzer;
 int S1_f,S2_f,S3_f;
 char FLG_L=0;                     // แฟลกของการส่ง LOW
 char FLG_M=0;                     // แฟลกของการส่ง MID
```

```
char FLG_H=0;                    // แฟลกของการส่ง HIGHT
char FLG_SM=0;                   // แฟลกของการส่ง SM
char status_water=0;             // แฟลกของการส่ง status water
char FLG_WT_L=0;                 // แฟลกของการส่ง status water      LOW
char FLG_WT_M=0;                 // แฟลกของการส่ง status water      MID
char FLG_WT_H=0;                 // แฟลกของการส่ง status water      HIGHT
char error_from=0;
char monday_send,day=0;                    //ตัวแปรการส่งของวันจันทร์
int ADC0_f , ADC1_f;

eeprom  int  index_write_sd;               //ตัววิ่งเก็บการบันทึก
eeprom  int  Rain[73],index_rain=0;        //ตัวแปรเก็บน้ำฝน
eeprom  int  Rain_Head[4];                 //เก็บหัวน้ำฝนเพื่อบันทึก
int  per,hour,time0=0,sec=0;
char test=0;
char status_r,RX=0;                        //ตัวแปรเก็บค่าจาก GPS
int time_reset=0;                          //ตัวแปรเก็บค่าการ REset loop
int  h,m,s,dd,mm,yy;                       //ตัวแปรเก็บค่า ชั่วโมง นาที วินาที

char tel[10]={'0','8','4','1','6','8','3','0','3','1'};  // client  tel เบอร์ของตัวลูก

//char te2[10]={'0','8','4','1','6','8','3','0','3','1'};  //  admin     เบอร์ของ Admin

char te2[10]={'0','8','4','9','7','1','4','9','7','5'};  //  admin

eeprom int counter=0;
char status_gps=0;               //ตัวแปรเก็บสถานะของ GPS
char loop=0;
char flg_send1,flg_send2,flg_send3;        //ตัวแปรของ แฟลกซ์ต่างๆ
char flg_send4,flg_send5,flg_send6;        //ตัวแปรของ แฟลกซ์ต่างๆของการส่ง

int ADC0_OLD ,ADC1_OLD,ADC2_OLD; //ตัวแปรของ การถ่ายค่าของ ADC ที่ REad มา
```

```
//
char USB_COMPLETE=0;          //ตัวแปรของการเก็บสถานะของ USB
char kb,temp_lcd[16];         //ตัวแปรของการเก็บคีย์
int  ADC0=0;                  //ตัวแปรของการเก็บค่า ADC0
int  ADC1=0;                  //ตัวแปรของการเก็บค่า ADC1
eeprom  int  ADC2;            //ตัวแปรของการเก็บค่า ADC2 เป็นแบบ EEPROM เปิด
เครื่องมาจะจำที่เดิม
// I2C Bus functions
#asm
  .equ __i2c_port=0x12 ;PORTD          //การติดต่อสื่อสารกับ REAL TIME CLOCK
DS1307
  .equ __sda_bit=0
  .equ __scl_bit=1
#endasm
#include <i2c.h>

// DS1307 Real Time Clock functions
#include <ds1307.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC          //การติดต่อสื่อสารกับ LCD
#endasm
#include <lcd.h>
#define RXB8 1                       //ตัวแปรของการสื่อสาร RS232
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
```

```
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART0 Receiver buffer
#define RX_BUFFER_SIZE0 64
char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0<256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;

// USART0 Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
char status,data;
status=UCSR0A;
data=UDR0;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer0[rx_wr_index0]=data;
  if (data=='K') {  RX=1;  status_gps=1; }     /// if Data ==> OK     RX=1
  else  if (data=='R')     cnt_error ++;

  if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
  if (++rx_counter0 == RX_BUFFER_SIZE0)
```

```
    {
    rx_counter0=0;
    rx_buffer_overflow0=1;
    };
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART0 Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter0==0);
data=rx_buffer0[rx_rd_index0];
if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
#asm("cli")
--rx_counter0;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART0 Transmitter buffer
#define TX_BUFFER_SIZE0 8
char tx_buffer0[TX_BUFFER_SIZE0];

#if TX_BUFFER_SIZE0<256
unsigned char tx_wr_index0,tx_rd_index0,tx_counter0;
#else
```

```
unsigned int tx_wr_index0,tx_rd_index0,tx_counter0;
#endif

// USART0 Transmitter interrupt service routine
interrupt [USART0_TXC] void usart0_tx_isr(void)
{
if (tx_counter0)
  {
  --tx_counter0;
  UDR0=tx_buffer0[tx_rd_index0];
  if (++tx_rd_index0 == TX_BUFFER_SIZE0) tx_rd_index0=0;
  };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART0 Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter0 == TX_BUFFER_SIZE0);
#asm("cli")
if (tx_counter0 || ((UCSR0A & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer0[tx_wr_index0]=c;
  if (++tx_wr_index0 == TX_BUFFER_SIZE0) tx_wr_index0=0;
  ++tx_counter0;
  }
else
  UDR0=c;
#asm("sei")
}
```

```
#pragma used-
#endif

// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 64
char rx_buffer1[RX_BUFFER_SIZE1];

#if RX_BUFFER_SIZE1<256
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
#else
unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
#endif

// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;

// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
{
char status,data;
status=UCSR1A;
data=UDR1;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer1[rx_wr_index1]=data;

  if (data=='D')  USB_COMPLETE=1;

  if ((status_r == 1) && (cnt_read<36) && (data >='0') && (data<='9'))
  {
     if  (cnt_read==0)  lcd_gotoxy(0,1);
     else if  (cnt_read==16)  lcd_gotoxy(0,2);
```

```
      else if  (cnt_read==32)  lcd_gotoxy(0,3);
     // else if  (cnt_read==16)  lcd_gotoxy(0,2);
      lcd_putchar(data);
      cnt_read++;
      if  (cnt_read==2) {lcd_putchar('/');cnt_read=3;}
      else if  (cnt_read==5) {lcd_putchar('/');cnt_read=6;}
      else if  (cnt_read==10) {lcd_putchar(':');cnt_read=11;}
      else if  (cnt_read==13) {lcd_putchar(':');cnt_read=14;}
      else if  (cnt_read==19) {lcd_putchar('-');cnt_read=20;}
      else if  (cnt_read==23) {lcd_putchar('-');cnt_read=24;}
      //else if  (cnt_read==27) {lcd_putchar('-');cnt_read=28;}
   }
   if (++rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
   if (++rx_counter1 == RX_BUFFER_SIZE1)
     {
     rx_counter1=0;
     rx_buffer_overflow1=1;
     };
   };
}

// Get a character from the USART1 Receiver buffer
#pragma used+
char getchar1(void)
{
char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1];
if (++rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#asm("cli")
--rx_counter1;
#asm("sei")
```

```
return data;
}
#pragma used-
// USART1 Transmitter buffer
#define TX_BUFFER_SIZE1 8
char tx_buffer1[TX_BUFFER_SIZE1];

#if TX_BUFFER_SIZE1<256
unsigned char tx_wr_index1,tx_rd_index1,tx_counter1;
#else
unsigned int tx_wr_index1,tx_rd_index1,tx_counter1;
#endif

// USART1 Transmitter interrupt service routine
interrupt [USART1_TXC] void usart1_tx_isr(void)
{
if (tx_counter1)
  {
  --tx_counter1;
  UDR1=tx_buffer1[tx_rd_index1];
  if (++tx_rd_index1 == TX_BUFFER_SIZE1) tx_rd_index1=0;
  };
}

// Write a character to the USART1 Transmitter buffer
#pragma used+
void putchar1(char c)
{
while (tx_counter1 == TX_BUFFER_SIZE1);
#asm("cli")
if (tx_counter1 || ((UCSR1A & DATA_REGISTER_EMPTY)==0))
  {
```

```
  tx_buffer1[tx_wr_index1]=c;
  if (++tx_wr_index1 == TX_BUFFER_SIZE1) tx_wr_index1=0;
  ++tx_counter1;
  }
else
  UDR1=c;
#asm("sei")
}
#pragma used-

// Standard Input/Output functions
#include <stdio.h>

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
      if (time0++ > 20000)
      {
      #asm("cli")
        time0=0;
         if (SW1==0)        // เช็คว่า SW มีการกดหรือไม่
      {
            while(SW1==0);
            ADC2++;  // ถ้ามีให้เพิ่ม ADC1 ทีละ 1 ค่า
      }
      else delay_ms(100);
       #asm("sei")

      }
       if (time1++ > 25000)
       {
```

```
        time1=0;
          rtc_get_time(&h,&m,&s);

          if (s!= s_old_buzzer)          //ทำการถ่ายค่าของการนับ Buzzer
          {
            s_old_buzzer = s;
            cnt_buzzer++;

            if (ALARM2>0)
            {
              ALARM2++;
            }
          }
      }
}
#define ADC_VREF_TYPE 0x00

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
// Declare your global variables here
char scan_keypad (void)
{
  char key=0;
```

```
 PORTA=0b11111111;
 COL1=0; COL2=1;  COL3=1; delay_ms(20);
 if(ROW1==0)    {key='1'; while(ROW1==0);}
 else if(ROW2==0) {key='4';while(ROW2==0);}
 else if(ROW3==0) {key='7';while(ROW3==0);}
 else if(ROW4==0) {key='*';while(ROW4==0);}

 COL1=1; COL2=0;  COL3=1; delay_ms(20);
 if(ROW1==0)    {key='2';while(ROW1==0);}
 else if(ROW2==0)  {key='5';while(ROW2==0);}
 else if(ROW3==0)  {key='8';while(ROW3==0);}
 else if(ROW4==0)  {key='0';while(ROW4==0);}

 COL1=1; COL2=1;  COL3=0; delay_ms(20);
 if(ROW1==0)    {key='3';while(ROW1==0);}
 else if(ROW2==0)  {key='6';while(ROW2==0);}
 else if(ROW3==0)  {key='9';while(ROW3==0);}
 else if(ROW4==0)  {key='#';while(ROW4==0);}
 return key;
}
void read_wt(void)
{
     ADC0=0;
     for(i=0;i<10;i++)
     {
       ADC0 = ADC0+read_adc(0);
     }
     ADC0 = ADC0/10;
     ADC0= ADC0 - START_READ;
     ADC0 = ADC0*1.33;
}
void read_sm(void)
```

```
{
    ADC1=0;
    for(i=0;i<10;i++)
    {
      ADC1 = ADC1+read_adc(1);
    }
    ADC1 = ADC1/10;
    ADC1= ADC1 - START_READ;
    ADC1 = ADC1*1.33;
}
void save_to_sdcard(void)                    //บันทึกลง SD card
{
        lcd_clear();
        lcd_gotoxy(0,1);lcd_putsf("  Save Data to  ");
        lcd_gotoxy(0,2);lcd_putsf("    SD CARD     ");
        rtc_get_date(&dd,&mm,&yy);
        read_sm();
        read_wt();
        putchar1('T');putchar1('P');putchar1('A');putchar1(0x0d);   delay_ms(500); //คำสั่งให้
write data
        putchar1('O');putchar1('P');putchar1('W');putchar1('
');putchar1('D');putchar1('A');putchar1('T');putchar1('A');putchar1('.');putchar1('T');putchar1('X');p
utchar1('T');putchar1(0x0d); delay_ms(500);
        putchar1('W');putchar1('R');putchar1('F');putchar1(' ');putchar1('3');putchar1('4');
putchar1(0x0d);   delay_ms(500); //จำนวน 34 ตัว

        putchar1(dd/10+'0');putchar1(dd%10+'0');putchar1('/');  //3
        putchar1(mm/10+'0');putchar1(mm%10+'0');putchar1('/');  //3
        putchar1(yy/10+'0');putchar1(yy%10+'0');  //2
        putchar1(h/10+'0'); putchar1(h%10+'0'); putchar1(':'); //3
        putchar1(m/10+'0'); putchar1(m%10+'0'); putchar1(':'); //3
        putchar1(s/10+'0'); putchar1(s%10+'0');  //2
```

```
        putchar1('[');putchar1(ADC2/1000+'0');putchar1(ADC2%1000/100+'0');
putchar1(ADC2%100/10+'0');putchar1(ADC2%10+'0'); putchar1(']'); //6
        putchar1('[');putchar1(ADC0/100+'0');putchar1(ADC0%100/10+'0');
putchar1(ADC0%10+'0');putchar1(']'); //5
        putchar1('[');putchar1(ADC1/100+'0');putchar1(ADC1%100/10+'0');
putchar1(ADC1%10+'0');putchar1(']'); //5
        putchar1(0x0d);putchar1(0x0a);   //2
        delay_ms(400);

        putchar1('C');putchar1('L');putchar1('F');putchar1('
');putchar1('D');putchar1('A');putchar1('T');putchar1('A');putchar1('.');putchar1('T');putchar1('X');p
utchar1('T');putchar1(0x0d); delay_ms(500); //ปิดไฟล์

        //delay_ms(400);
        index_write_sd++;
        //1110101617XXXYYY!\r\n
}
void Send_SMS1(char er,char st)
{
    char loop5=1;
    while(loop5)
    {
        RX=0;       time_reset=0;
        delay_ms(1000);
        while(RX==0)
       {
         printf("AT"); putchar(0x0d);
         delay_ms(1000);
         time_reset++;
         if (time_reset>10) RX=1;
       }
```

```
        RX=0;    time_reset=0;
      while(RX==0)
     {
        printf("AT"); putchar(0x0d);
        delay_ms(1000);
        time_reset++;
   if (time_reset>10) RX=1;
     }
     sprintf(temp_lcd," Sending SMS  ");
     lcd_gotoxy(0,0);lcd_puts(temp_lcd);
     sprintf(temp_lcd,"   To client   ");
     lcd_gotoxy(0,1);lcd_puts(temp_lcd);
    RX=0;time_reset=0;
     while(RX==0)
     {
        printf("AT"); putchar(0x0d);        //ส่ง AT ไปจนกว่าจะตอบ OK กลับมา
        delay_ms(1000);
        time_reset++;
        if (time_reset>10) RX=1;
     }
     delay_ms(1000);
     printf("AT+CMGS ="); //คำสั่งให้ส่ง SMS
     putchar("");

printf("%c%c%c%c%c%c%c%c%c%c",tel[0],tel[1],tel[2],tel[3],tel[4],tel[5],tel[6],tel[7],tel[8],tel
[9]);     // ส่งเบอร์ที่เก็บไว้ด้านบน
     putchar("");
     putchar(0x0d);  putchar(0x0a);  delay_ms(1000);
     putchar('$');
     putchar(h/10+'0'); putchar(h%10+'0'); putchar(m/10+'0'); putchar(m%10+'0');
putchar(s/10+'0'); putchar1(s%10+'0');  delay_ms(400);  // ส่งเวลา  120006  ไม่มี  :  กั้นกลาง
```

```
putchar(dd/10+'0');putchar(dd%10+'0');putchar(mm/10+'0');putchar(mm%10+'0');putchar(yy/10+'
0');putchar(yy%10+'0'); delay_ms(400); //ส่งวันที่  121010  ไม่มี / กั้นกลาง
        if (er==1) putchar('1');
        if (er==2) putchar('2');                    // ส่งปริมาณน้ำช่องที่ 1
        if (er==3) putchar('3');

        putchar(ADC2/1000+'0');putchar(ADC2%1000/100+'0'); putchar(ADC2%100/10+'0');
putchar(ADC2%10+'0'); delay_ms(400);
        putchar(ADC0/100+'0');putchar(ADC0%100/10+'0'); putchar(ADC0%10+'0');
delay_ms(400);
        putchar(ADC1/100+'0');putchar(ADC1%100/10+'0'); putchar(ADC1%10+'0');
delay_ms(400);
       printf("%c",st);        // ส่งปริมาณน้ำช่องที่ 2
        delay_ms(1000);
        putchar('!');
//ส่ง ! เพื่อบอกว่าข้อมูลครบแล้ว
        RX=0;
        putchar(0x1A);

      //  putchar(0x0d); //putchar(0x0a);
        cnt_error=0;

      //  delay_ms(1000);  //ส่ง Ctrl Z เพื่อบอกให้ส่ง SMS
        loop=1;
        time_reset=0;
       while (loop)
       {
              if  (RX==1)  { loop=0;  loop5=0;}          //หาก OK แสดงว่าส่ง ผ่านแล้ว
ออกจาก loop ได้เลย
              if   (cnt_error >=2)   loop=0;             //หาก Error ให้กลับไปส่งใหม่
อีกครั้ง
```

```
                if (time_reset++>10) loop=0;
                delay_ms(1000);
            }


            }


}

        /*
void Send_SMS2(char ch)
{
            sprintf(temp_lcd," Sending SMS 2 ");
            lcd_gotoxy(0,0);lcd_puts(temp_lcd);
            time_reset=0;
            RX=0;
            while(RX==0)
            {
               printf("AT"); putchar(0x0d);  //ส่ง AT ไปยัง Module SMS รอตอบกลับมาด้วย OK
               delay_ms(1000);
               time_reset++;
               if (time_reset>10) RX=1;
            }
                RX=0;time_reset=0;
              while(RX==0)
            {
               printf("AT"); putchar(0x0d);
               delay_ms(1000);
               time_reset++;
               if (time_reset>10) RX=1;
            }
```

```
            delay_ms(1000);
            printf("AT+CMGS =");
            putchar("");
            printf("
08%c%c%c%c%c%c%c%c",te2[2],te2[3],te2[4],te2[5],te2[6],te2[7],te2[8],te2[9]);          //ส่ง
ตามไปด้วยเบอร์โทร
            putchar("");
            putchar(0x0d);delay_ms(1000);

            //putchar('$');
            printf("Time : ");
            putchar(h/10+'0'); putchar(h%10+'0');  putchar(':'); putchar(m/10+'0');
putchar(m%10+'0'); putchar(':');putchar(s/10+'0'); putchar(s%10+'0');
            printf(" Date : ");

putchar(dd/10+'0');putchar(dd%10+'0');putchar('/');putchar(mm/10+'0');putchar(mm%10+'0');putc
har('/');putchar(yy/10+'0');putchar(yy%10+'0');
            printf(" Water 1 [");
            putchar(ADC0/100+'0');putchar(ADC0%100/10+'0'); putchar(ADC0%10+'0');
            printf("]  Water 2 [");
            putchar(ADC1/100+'0');putchar(ADC1%100/10+'0'); putchar(ADC1%10+'0');
            putchar(']');

            printf("  status : ");
            if (ch==1) printf(" SENSOR 1 LOW ");
            else if (ch==2) printf(" SENSOR 1 MID ");
            else if (ch==3) printf(" SENSOR 1 HIGHT ");
            else if (ch==4) printf(" SENSOR 2 LOW ");
            else if (ch==5) printf(" SENSOR 2 MID ");
            else if (ch==6) printf(" SENSOR 2 HIGHT ");
```

```
        putchar(0x1A); delay_ms(1000);    delay_ms(3000);//delay_ms(5000)    // ส่ง 0x1A
เพื่อบอกให้รู้ว่าข้อความเสร็จแล้ว
        delay_ms(3000);



}
        */
void Send_SMS_admin(char er,char st,char per1)
{
        char loop6=1;
        while(loop6)            // วน ลูบ รอ จนกว่าจะส่งเสร็จ
        {
        sprintf(temp_lcd," Sending SMS %i ",error_from);
        lcd_gotoxy(0,0);lcd_puts(temp_lcd);
        time_reset=0;
        RX=0;
        while(RX==0)            //ส่ง AT รอ OK ตอบกลับมา
        {
           printf("AT"); putchar(0x0d);
           delay_ms(1000);
           time_reset++;
         if (time_reset>10) RX=1;
        }
         RX=0;time_reset=0;
         while(RX==0)           //ส่ง AT รอ OK ตอบกลับมา
        {
           printf("AT"); putchar(0x0d);
           delay_ms(1000);
           time_reset++;
           if (time_reset>10) RX=1;
        }
         delay_ms(1000);
```

```
            printf("AT+CMGS =");    // คำสั่งให้ส่ง SMS
            putchar("");
            printf("
08%c%c%c%c%c%c%c%c",te2[2],te2[3],te2[4],te2[5],te2[6],te2[7],te2[8],te2[9]);          //
ส่งเบอร์ไปด้วย 10 หลัก
            putchar("");
            putchar(0x0d);  putchar(0x0a);delay_ms(1000);

            //putchar('$');
            //printf("Time : ");
            //putchar(h/10+'0'); putchar(h%10+'0');  putchar(':'); putchar(m/10+'0');
putchar(m%10+'0'); putchar(':');putchar(s/10+'0'); putchar1(s%10+'0');   delay_ms(200);
            //printf(" Date : ");

//putchar(dd/10+'0');putchar(dd%10+'0');putchar('/');putchar(mm/10+'0');putchar(mm%10+'0');put
char('/');putchar(yy/10+'0');putchar(yy%10+'0');   delay_ms(200);
            printf("Rain (");
            putchar(ADC2/1000+'0');putchar(ADC2%1000/100+'0'); putchar(ADC2%100/10+'0');
putchar(ADC2%10+'0');  delay_ms(400);
            printf(")  mm Water (");
            putchar(ADC0/100+'0');putchar(ADC0%100/10+'0'); putchar(ADC0%10+'0');
delay_ms(400);
            printf(")  cm Soil(");
            putchar(ADC1/100+'0');putchar(ADC1%100/10+'0'); putchar(ADC1%10+'0');
delay_ms(400);
            printf("  cm Flash ");  delay_ms(400);
            if (er==1) printf("Rain),(");
            else if (er==2) printf("Water),(");
            else if (er==3) printf("Soil),(");
            delay_ms(400);
            printf("%c)",st);
```

```
if (per1==1) printf("/1hr");
if (per1==24) printf("/24hr");
if (per1==72) printf("/72hr");
delay_ms(1000);
cnt_error=0; RX=0;
putchar(0x1A);   // putchar(0x0d);
// delay_ms(1000);//delay_ms(5000)  ส่ง 0X1A เพื่อบอกว่าจบโปรแกรมแล้ว
 time_reset=0;
 loop=1;
 // รอใน loop ว่า มี Error ตอบกลับมาหรือไม่
while (loop)
{
   if  (RX==1)  {  loop=0;   loop6=0;}      //หาก OK แสดงว่าส่ง ผ่านแล้ว ออกจาก
loop ได้เลย
        if     (cnt_error >=2)    loop=0;            //หาก Error ให้กลับไปส่งใหม่อีก
ครั้ง
        if (time_reset++>10) loop=0;            //หาก Time out ให้กลับไปส่งใหม่อีก
ครั้ง
       delay_ms(1000);
}
 time_reset=0;
 RX=0;
 while(RX==0)
 {
    printf("AT"); putchar(0x0d);
    delay_ms(1000);
    time_reset++;
    if (time_reset>5) RX=1;
 }

 }
 Send_SMS1(er,st);          //ส่ง SMS ไปยังอีกเบอร์
```

```
}
void read_data_from_sdcard(void)
{
        char i,s =0;
        char loop2=0;
        putchar1('I');putchar1('P');putchar1('A');putchar1(0x0d);  delay_ms(500);  //คำสั่งให้
อ่านข้อมูลจาก SD card
        putchar1('O');putchar1('P');putchar1('R');putchar1('
');putchar1('D');putchar1('A');putchar1('T');putchar1('A');putchar1('.');putchar1('T');putchar1('X');p
utchar1('T');putchar1(0x0d); delay_ms(500);
        if (index_write_sd > 5) s=index_write_sd-5;
        else s=0;
       for (i=s;i<index_write_sd;i++) //37  74     111
        {
      putchar1('S');putchar1('E');putchar1('K');putchar1(' ');       //คำสั่งค้นหาไปยังตำแหน่งที่
ต้องการ
        if  (i*33 < 10)
            {
                putchar1(i*33%10+'0');
            }
            else if (i*33 < 100)
            {
                putchar1((i*33)/10+'0');
                putchar1((i*33)%10+'0');
            }
             else if (i*33 < 9999)
            {
                putchar1((i*33)/100+'0');
                putchar1((i*33)%100/10+'0');
                putchar1((i*33)%10+'0');
            }
            putchar1(0x0d);  delay_ms(500);
```

```
                lcd_clear();
                sprintf(temp_lcd,"index %i/%i [%i]",i+1,index_write_sd,i*33);
                lcd_gotoxy(0,0);lcd_puts(temp_lcd);
                putchar1('R');putchar1('D');putchar1('F');putchar1(' ');putchar1('3');putchar1('3');
putchar1(0x0d);
                status_r = 1;
                cnt_read=0;
                delay_ms(4000);
                status_r = 0;
                loop2=1;
                while(loop2)
                {
                    kb = scan_keypad();  /// รับคำสั่งคีย์แพด
                    if (kb!=0)
                    {
                        if (kb=='#') loop2=0;
                    }
                }
            }
}
int sum_val(char hr)                    //เรียกฟังก์ชั่นการทำงานของการสะสมทุก 1 24 และ 72
ชั่วโมง โดย พารามิเตอร์ hr
{
    int ii,bb;
   char index_monitor=0;
    if (hr==1)
    {
      if  (index_rain==1)          return  Rain[72] - ADC2;
      else        return  ADC2 - Rain[index_rain-1];
    }
    else if (hr==24)  // 14
    {
```

```
                if (index_rain <25)  // 24                                    //  5  hr
69 70 71 72          1  2  3  4  5  6  7  8 9 10
                {
                        index_monitor =  ( index_rain -24)+72;
                }
                else     index_monitor =  ( index_rain -24);

                return   ADC2  -  Rain[index_monitor];
        }
        else  if (hr==72)  // 14
        {
                return       ADC2 - Rain[index_rain+1];
        }
}
void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0xff;
DDRA=0x07;
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x01;
DDRB=0x02;
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
```

```
DDRC=0x00;
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;
// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;
// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;
// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 16000.000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
```

```
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
```

```
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
```

```
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;
ETIMSK=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0xD8;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud rate: 9600
UCSR1A=0x00;
```

```
UCSR1B=0xD8;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x67;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

BUZZER=0;
lcd_gotoxy(0,0);lcd_putsf("Waiting USB");
putchar1(0x0d);
while (USB_COMPLETE==0);          //รอจนกว่า USB จะพร้อม โดยการรอรับคำสั่ง Ready
มาจาก USB
i2c_init();
delay_ms(2000);
/* initialize the DS1307 RTC */
rtc_init(0,0,0);
printf("Test RS232\r\n");
```

```
//rtc_set_time(23,0,0);         //ตั้งเวลา
//rtc_set_date(15,01,11);       //ตั้งวันที่
//rtc_write(3,7);           // กำหนดการตั้งวันที่  7 = อาทิตย์   1 จันทร์
  test=0;
  RX=0;
  status_gps=0;
    if (test==0)                    //ทำการ initial ค่าของ GPRS
  {
        sprintf(temp_lcd,"  initial GPRS  ");
        lcd_gotoxy(0,0);lcd_puts(temp_lcd);
        sprintf(temp_lcd," Please wait ...");
        lcd_gotoxy(0,1);lcd_puts(temp_lcd);
        delay_ms(2000);
        sprintf(temp_lcd,"  initial SMS   ");
        lcd_gotoxy(0,2);lcd_puts(temp_lcd);
        sprintf(temp_lcd," Please wait ...");
        lcd_gotoxy(0,3);lcd_puts(temp_lcd);
        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d); // รอคำสั่ง OK  มาจาก GPRS
            delay_ms(2000);
        }
        RX=0;
        while(RX==0)
        {
            printf("AT+CMGF = 1");  putchar(0x0d);    //คำสั่งให้ SMS แบบ Charector
            delay_ms(2000);
        }
  }
  sprintf(temp_lcd,"  Initial OK  ");   lcd_gotoxy(0,1);    lcd_puts(temp_lcd);
  lcd_clear();
```

```
    tel[10]=' ';
    te2[10]=' ';
    lcd_gotoxy(0,1); sprintf(temp_lcd,"Tel1=%s",tel); lcd_puts(temp_lcd);
    lcd_gotoxy(0,2);
sprintf(temp_lcd,"Tel2=0%c%c%c%c%c%c%c%c%c",te2[1],te2[2],te2[3],te2[4],te2[5],te2[6],te
2[7],te2[8],te2[9]); lcd_puts(temp_lcd);
    delay_ms(1000);
   loop=1;
   while(loop)  // รอการกดปุ่ม 0 หรือ * เพื่อทำการ Clear data ต่างๆ
   {
     kb = scan_keypad();
     if (kb=='0')    { index_write_sd=0;  index_rain=0;  for(i=0;i<73;i++) Rain[i]=0;}
     else   if (kb=='*')  loop=0;
     else   if (kb=='#')   ADC2=0;
   }
   status_r=0;
   flg_send1=0;
   flg_send2=0;
   flg_send3=0;
   flg_send4=0;
   flg_send5=0;
   flg_send6=0;

   //ADC2=0;
   ADC1=0;
   ADC0=0;
   s_old=0;
   sec_cnt=0;
   cnt_buzzer=0;
   // rtc_write(3,5);

   for (i=0;i<=72;i++)  Rain[i] =   ADC2;          // Claer ค่า น้ำฝน ทุกค่าให้เป็นค่าน้ำฝน
```

```
  index_rain=0;
  S1_f = sum_val(1);  //เก็บค่าปัจจุบันลงในตัวแปร _f เพื่อทำการเช็คในรอบถัดไปหากทำการเปิดเครื่องมาใหม่
  S2_f = sum_val(24);  //เก็บค่าปัจจุบันลงในตัวแปร _f เพื่อทำการเช็คในรอบถัดไปหากทำการเปิดเครื่องมาใหม่
  S3_f = sum_val(72); //เก็บค่าปัจจุบันลงในตัวแปร _f เพื่อทำการเช็คในรอบถัดไปหากทำการเปิดเครื่องมาใหม่
  read_wt();
  ADC0_f=ADC0;  //เก็บค่าปัจจุบันลงในตัวแปร _f เพื่อทำการเช็คในรอบถัดไปหากทำการเปิดเครื่องมาใหม่
  read_sm();
  ADC1_f=ADC1;        //เก็บค่าปัจจุบันลงในตัวแปร _f เพื่อทำการเช็คในรอบถัดไปหากทำการเปิดเครื่องมาใหม่
 // ADC2=1045;
     ALARM2=0;
while (1)
   {
    //lcd_clear();
    rtc_get_time(&h,&m,&s);      sprintf(temp_lcd," Time :%02i:%02i:%02i ",h,m,s);
lcd_gotoxy(0,0);      lcd_puts(temp_lcd);              //อ่านเวลามาเก็บไว้
     rtc_get_date(&dd,&mm,&yy);     sprintf(temp_lcd," Date :%02i/%02i/%02i ",dd,mm,yy);
lcd_gotoxy(0,1);      lcd_puts(temp_lcd);         //อ่านวันที่มาเก็บไว้
    if ( time_clear_reset ++ > 20) //เคลียร์ค่านับให้เป็น 0
    {
      time_clear_reset=0;
       cnt_reset=0;
      }

       // Send_SMS1(er,st);              //ส่ง SMS ไปยังอีกเบอร์
         day = rtc_read(3);  // get day 0-sunday  1 monday
     if ((day == 1) && (monday_send==0))                 // day = 1 วันจันทร์  send data every
monday
```

```
    {
      monday_send=1;
      error_from=11;
      Send_SMS_admin(1,'L','d');
      er = 1;
      st = 'L';
    }
    else if (day>=2)  monday_send=0;
   /*
    if (s!= s_old_buzzer)          //ทำการถ่ายค่าของการนับ Buzzer
    {
      s_old_buzzer = s;
      cnt_buzzer++;
    }  */

    if (s!= s_old)  //ทำการถ่ายค่าของการนับ เวลาในการบันทึกลง SD CARD
    {
      sec_cnt++;
      s_old = s;
    }
    // 0 1 2 3 4 5 6
    // 3 5 7 9
    // 3 2 2 2
    read_wt();  //  อ่านค่าน้ำฝน
    read_sm();    //อ่านค่าความชื้นในดิน
  //  lcd_clear();
  //   sprintf(temp_lcd,"[%c][%c]",Status_rf,status_water);                lcd_gotoxy(0,1);
lcd_puts(temp_lcd);       // โชว์ค่าออก LCD
    sprintf(temp_lcd,"RF[%04i] WT[%03i]",ADC2,ADC0);                lcd_gotoxy(0,2);
lcd_puts(temp_lcd);       // โชว์ค่าออก LCD
    sprintf(temp_lcd,"SM[%03i]ER[%02i/%02i]",ADC1,cnt_round,round_Set);
lcd_gotoxy(0,3);      lcd_puts(temp_lcd);        // แสดงค่า จำนวนครั้งที่ buzzer ดังด้วย
```

```
    // cnt_round=0;
   // sprintf(temp_lcd," %i  %i          ",index_rain,sec_cnt);                lcd_gotoxy(0,2);
lcd_puts(temp_lcd);
    //  sprintf(temp_lcd,"[1]%i[24]%i[72]%i  ",sum_val(1),sum_val(24),sum_val(72));
lcd_gotoxy(0,3);       lcd_puts(temp_lcd);
    //  Status_rf=0;
     //************************** S1 **************************
     S1 = sum_val(1)-S1_f;  // ไปอ่านค่าน้ำฝน ใน 1 ชั่วโมง
     if (S1 >= 35)
     {
        Status_rf ='L';  per=1;
     }
     if (S1 >= 50)
     {
        Status_rf ='M';  per=1;
     }
     if (S1 >= 70)
     {
        Status_rf ='H';  per=1;
     }
     //************************** S24 **************************
     S2 = sum_val(24)-S2_f;   // ไปอ่านค่าน้ำฝน ใน 24 ชั่วโมง
     if (S2 >= 90)
     {
        Status_rf ='L';  per=24;
     }
     if (S2 >= 100)
     {
        Status_rf ='M';  per=24;
     }
     if (S2 >= 110)
     {
```

```
    Status_rf ='H';  per=24;
  }
  //************************** S72 ************************
  S3 = sum_val(72)-S3_f;              // ไปอ่านค่าน้ำฝน ใน 72 ชั่วโมง
  if (S3 >= 185)
  {
    Status_rf ='L';  per=72;
  }
  if (S3 >= 200)
  {
    Status_rf ='M';  per=72;
  }
  if (S3 >= 215)
  {
    Status_rf ='H';  per=72;
  }

   //********************************** write data to  ram
**************************
    delay_ms(900);
    if  (sec_cnt%30==0)  //test 30 only    3600 1 hr          กำหนดเวลาให้ทำการบันทึกลง
EEPROM
   {
      sec_cnt=0;
      index_rain++;    //เพิ่มตัว Counter ลงในตัวแปร index
       ADC_temp =  ADC2 - ADC_old;
       ADC_old = ADC2;
       Rain[index_rain]= ADC2;            //เก็บค่าลงในตัวแปร Array
       if  ( index_rain > 72)      //หากตัวนับมากกว่า 72 ชั่วโมง ให้ทำการนับใหม่
       {
            index_rain=1;
            Rain[index_rain]= ADC2;
```

```
//********************** Program 2 clear all ************************
//     for (i=0;i<=72;i++)  Rain[i] =   ADC2;
}
}

//**********send SMS ********
read_sm();
ADC1 = ADC1 - ADC1_f;
if ((ADC1 >=50) && (FLG_SM==0))  //Check SM
{
        error_from=1;
        Send_SMS_admin(3,'H',0);          // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
        FLG_SM=1;
        cnt_buzzer=0;
        BUZZER=1;
    cnt_round=0;
     er = 3;
    st = 'H';
}
else  if (FLG_SM==0)  // un sm
{
  if  ((Status_rf =='L') && (FLG_L==0))
  {
    FLG_L=1;
    FLG_M=0;
    FLG_H=0;
    read_wt();
    if (ADC0 <50)
    {
      error_from=2;
```

```
            Send_SMS_admin(1,Status_rf,per); // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ Hight และ ตัวแปร PER [ 1,24,72]
             cnt_buzzer=0;
             BUZZER=1;
           cnt_round=0;
             er = 1;
             st = Status_rf;
           }
           else
           {
             error_from=3;
             Send_SMS_admin(2,'M',per);        // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ Hight และ ตัวแปร PER [ 1,24,72]
             cnt_buzzer=0;
             BUZZER=1;
           cnt_round=0;
             er = 2;
             st ='M';
           }
         }
        if ((Status_rf =='M') && (FLG_M==0))
        {
          FLG_M=1;
          FLG_H=0;
          read_wt();
          if (ADC0 <50)
          {
            error_from=4;
             Send_SMS_admin(1,Status_rf,per); // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ Hight และ ตัวแปร PER [ 1,24,72]
            cnt_buzzer=0;
            BUZZER=1;
```

```
            cnt_round=0;
              er = 1;
               st =Status_rf;
            }
            else
            {
               error_from=5;
               Send_SMS_admin(2,'H',per);  // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี พารามิเตอร์
คือ  Hight และ ตัวแปร PER  [ 1,24,72]
               cnt_buzzer=0;
               BUZZER=1;
           cnt_round=0;
            er = 2;
             st ='H';
            }
         }
           if  ((Status_rf =='H') && (FLG_H==0))
        {
          FLG_H=1;
          read_wt();
          if (ADC0 <50)
          {
            error_from=6;
           Send_SMS_admin(1,Status_rf,per);  // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
            cnt_buzzer=0;
            BUZZER=1;
         cnt_round=0;
          er = 1;
            st =Status_rf;
          }
          else
```

```
            {
              error_from=7;
              Send_SMS_admin(2,'H',per);      // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
              cnt_buzzer=0;
              BUZZER=1;
            cnt_round=0;
             er = 2;
              st ='H';
            }
          }
         status_water=0;
         read_wt();
         ADC0 =ADC0 -ADC0_f;
         if (ADC0 >= 65)  status_water='L';
         if (ADC0 >= 80)  status_water='M';
         if (ADC0 >= 90)  status_water='H';
         if  ((status_water=='L') && (FLG_WT_L==0))
         {
           FLG_WT_L=1;
           error_from=8;
           Send_SMS_admin(2,status_water,0);       // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
           cnt_buzzer=0;
           cnt_round=0;
           BUZZER=1;
            er = 2;
             st =status_water;

        //    round=0;
         }
         else if  ((status_water=='M') && (FLG_WT_M==0))
```

```
        {
          FLG_WT_M=1;
          error_from=9;
        Send_SMS_admin(2,status_water,0);   // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
          cnt_buzzer=0;
          BUZZER=1;
    cnt_round=0;
        er = 2;
            st =status_water;
        }
        else if  ((status_water=='H') && (FLG_WT_H==0))
        {
          FLG_WT_H=1;
          error_from=10;
          Send_SMS_admin(2,status_water,0);   // ทำการส่ง SMS ไปยังเบอร์ Admin โดยมี
พารามิเตอร์ คือ  Hight และ ตัวแปร PER  [ 1,24,72]
          cnt_buzzer=0;
          BUZZER=1;
         cnt_round=0;
            er = 2;
            st =status_water;
        }
    }
        if (DEBUG_23)
          {
           printf("S1 = %d S1_f = %d\r\n",S1,S1_f);
           printf("S2 = %d S2_f = %d\r\n",S2,S2_f);
           printf("S3 = %d S3_f = %d\r\n",S3,S3_f);
           printf("Status_rf = %c \r\n",Status_rf);
           printf("adc0 = %d \r\n",ADC0);
           printf("adc1 = %d \r\n",ADC1);
```

```
            printf("index_rain =%d \r\n",index_rain);
           }
        //=========================================
        //============== test only ==============
       // Status_rf = 'H';
        //round_Set=10;
            if ((FLG_SM==1) && (cnt_round<30))
        {
           if (cnt_buzzer<=30)          BUZZER=1;
           else if (cnt_buzzer<=60)     BUZZER=0;
           else  {cnt_buzzer=0; cnt_round++; }
           round_Set=30;
         }
       // else   round_Set=0;
        if   (  Status_rf=='L')
        {          round_Set=10;
                   if (cnt_buzzer<=10)          BUZZER=1;
                   else if (cnt_buzzer<=60)     BUZZER=0;
                   else  if (cnt_buzzer>60)
                   {
                              cnt_buzzer=0;
                              cnt_round++;
                              if    (cnt_round >=round_Set)      {Status_rf=0;    cnt_round=0;
round_Set=0;}
                   }
        }
        else   if   (  Status_rf=='M')
        {          round_Set=20;
                   if (cnt_buzzer<=20)          BUZZER=1;
                   else if (cnt_buzzer<=60)     BUZZER=0;
                   else  if (cnt_buzzer>60)
                   {
```

```
                    cnt_buzzer=0;
                    cnt_round++;
                    if    (cnt_round >=round_Set)     {Status_rf=0;    cnt_round=0;
round_Set=0;}
                }
    }
  else     if   (( Status_rf=='H')   || (FLG_SM==1) )
    {       round_Set=30;
            if (cnt_buzzer<=30)          BUZZER=1;
            else if (cnt_buzzer<=60)     BUZZER=0;
             else  if (cnt_buzzer>60)
             {
                    cnt_buzzer=0;
                    cnt_round++;
                    if    (cnt_round >=round_Set)     {Status_rf=0;    cnt_round=0;
round_Set=0; }
             }
    }
     else   if   ( status_water=='L')
    {         round_Set=10;
            if (cnt_buzzer<=10)          BUZZER=1;
            else if (cnt_buzzer<=60)     BUZZER=0;
             else  if (cnt_buzzer>60)
             {
                    cnt_buzzer=0;
                    cnt_round++;
                    if    (cnt_round >=round_Set)     {status_water=0;    cnt_round=0;
round_Set=0;}
             }
    }
    else     if   ( status_water=='M')
    {         round_Set=20;
```

```
                if (cnt_buzzer<=20)          BUZZER=1;
                else if (cnt_buzzer<=60)     BUZZER=0;
                 else  if (cnt_buzzer>60)
                 {
                        cnt_buzzer=0;
                        cnt_round++;
                        if    (cnt_round >=round_Set)      {status_water=0;    cnt_round=0;
round_Set=0; }
                 }
    }
   else      if   (  status_water=='H')
    {          round_Set=30;
                if (cnt_buzzer<=30)          BUZZER=1;
                else if (cnt_buzzer<=60)     BUZZER=0;
                 else  if (cnt_buzzer>60)
                 {
                        cnt_buzzer=0;
                        cnt_round++;
                        if    (cnt_round >=round_Set)      {status_water=0;    cnt_round=0;
round_Set=0; }
                 }
    }
     /*
    if ((status_water=='L')&& (cnt_round<10))
    {
       if (cnt_buzzer<=10)          BUZZER=1;
       else if (cnt_buzzer<=60)     BUZZER=0;
       else  {cnt_buzzer=0;  cnt_round++; }
       round_Set=10;
        sprintf(temp_lcd," 5 [%i]  [%i]",cnt_buzzer,cnt_round);
lcd_gotoxy(0,3);      lcd_puts(temp_lcd);        // แสดงค่า จำนวนครั้งที่ buzzer ดังด้วย
    }
```

```
    else if ((status_water=='M')&& (cnt_round<20))
    {
        if (cnt_buzzer<=20)          BUZZER=1;
        else if (cnt_buzzer<=60)     BUZZER=0;
        else  {cnt_buzzer=0;  cnt_round++; }
        round_Set=20;
         sprintf(temp_lcd," 6 [%i]  [%i]",cnt_buzzer,cnt_round);
lcd_gotoxy(0,3);      lcd_puts(temp_lcd);        // แสดงค่า จำนวนครั้งที่ buzzer ดังด้วย
    }
    else if ((status_water=='H')&& (cnt_round<30))
    {
        if (cnt_buzzer<=30)          BUZZER=1;
        else if (cnt_buzzer<=60)     BUZZER=0;
        else  {cnt_buzzer=0;  cnt_round++; }
        round_Set=30;
        sprintf(temp_lcd," 7 [%i] ",cnt_buzzer);                    lcd_gotoxy(0,3);
lcd_puts(temp_lcd);        // แสดงค่า จำนวนครั้งที่ buzzer ดังด้วย
    }
    */
     if (cnt_save++ > 50)  // เวลาที่ นับถอยหลัง เพื่อทำการบันทึกข้อมูลลง SD CARD
      {
        cnt_save=0;
        if   (  (ADC0==ADC0_OLD)  && (ADC1==ADC1_OLD) &&
(ADC2==ADC2_OLD))
        {
        }
        else
        {
                    save_to_sdcard();  // กระโดดไปฟังก์ชั่น SAVE DATA
                    ADC0_OLD = ADC0;
                    ADC1_OLD = ADC1;
                    ADC2_OLD = ADC2;
```

```
    }
}
kb = scan_keypad();              //เช็คว่ามีการกด คีย์แพดหรือไม่
if (kb!=0)
{
   lcd_gotoxy(0,1);lcd_putsf("key ");
   lcd_putchar(kb);  delay_ms(100);

  // if (kb=='1')  Send_SMS();
   if (kb=='#')
   {
     read_data_from_sdcard();       // หากกด # ให้อ่านค่าจาก SD CARD
     cnt_reset=0;
   }
   else if (kb=='2')   // บันทึกเวลา
   {
      lcd_clear();
      sprintf(temp_lcd,"Set day = ",cnt_reset);
      lcd_gotoxy(0,0);
      lcd_puts(temp_lcd);
      delay_ms(200);
      loop=1;
      while(loop)
      {
         kb = scan_keypad();
         if (kb!=0)
         {
            if (kb=='#')
            {
                  rtc_write(3,time_buff[0]-'0');          //บันทึก วัน ลง ใน DS1307

                  memset(dummy,0,sizeof dummy);
```

```
dummy[0] =  time_buff[1];
dummy[1] =  time_buff[2];
dd = atoi(dummy);

memset(dummy,0,sizeof dummy);
dummy[0] =  time_buff[3];
dummy[1] =  time_buff[4];
mm = atoi(dummy);

memset(dummy,0,sizeof dummy);
dummy[0] =  time_buff[5];
dummy[1] =  time_buff[6];
yy = atoi(dummy);

memset(dummy,0,sizeof dummy);
dummy[0] =  time_buff[7];
dummy[1] =  time_buff[8];
h = atoi(dummy);

memset(dummy,0,sizeof dummy);
dummy[0] =  time_buff[9];
dummy[1] =  time_buff[10];
m = atoi(dummy);

memset(dummy,0,sizeof dummy);
dummy[0] =  time_buff[11];
dummy[1] =  time_buff[12];
s = atoi(dummy);
loop=0;
rtc_set_time(h,m,s);        //ตั้งเวลา
rtc_set_date(dd,mm,yy);     //ตั้งวันที่
}
```

```
else if (kb=='*')
{
   loop=0;
}
else
{
   lcd_putchar(kb);
   time_buff[index_time]=kb;  index_time++;
   if (index_time==1)
   {
        lcd_clear();
        sprintf(temp_lcd,"Set date = ",cnt_reset);
        lcd_gotoxy(0,0);
        lcd_puts(temp_lcd);
        lcd_gotoxy(4,1);
   }
    if (index_time==7)
   {
        lcd_clear();
        sprintf(temp_lcd,"Set time = ",cnt_reset);
        lcd_gotoxy(0,0);
        lcd_puts(temp_lcd);
        lcd_gotoxy(4,1);
    }
    if (index_time==3) lcd_putchar('/');
    if (index_time==5) lcd_putchar('/');
    if (index_time==9) lcd_putchar(':');
    if (index_time==11)lcd_putchar(':');
    delay_ms(100);
  }
 }
}
```

```
}
else if (kb=='7')
{
        lcd_clear();
        sprintf(temp_lcd,"Press %d/4 clear",cnt_reset);
        lcd_gotoxy(0,1);
        lcd_puts(temp_lcd);
        delay_ms(200);
    if (cnt_reset++> 3)
    {
        VAL72=0;
        index_rain=0;
   // for(i=0;i<73;i++) Rain[i]=0;
        FLG_SM=0;
        FLG_L=0;
        FLG_M=0;
        FLG_H=0;
        FLG_WT_L=0;
        FLG_WT_M=0;
        FLG_WT_H=0;
        cnt_reset=0;
        lcd_clear();
        sprintf(temp_lcd," Clear data all  ");
        lcd_gotoxy(0,1);
        lcd_puts(temp_lcd);
        delay_ms(200);
        for(i=0;i<73;i++)   Rain[i]=0;

        ADC0_f=ADC0;
        ADC1_f=ADC1;
        S1_f=S1;
        S2_f=S2;
```

```
            S3_f=S3;
        }
}
  else if (kb=='8')  // clear buzzer
{
        BUZZER=0;
        FLG_SM=0;
        Status_rf=0;
        status_water=0;
        round=100;
        BUZZER=0;
        lcd_clear();
        sprintf(temp_lcd," Clear Buzzer off ");
        lcd_gotoxy(0,1);
        lcd_puts(temp_lcd);
        delay_ms(500);
}
 else if (kb=='4')
 {
   Send_SMS_admin(1,'L',0);     cnt_reset=0;
 }
   else if (kb=='9')
 {
            lcd_clear();
            sprintf(temp_lcd,"Total 72 Hr =%i ",VAL72);
             lcd_gotoxy(0,1); lcd_puts(temp_lcd);
            delay_ms(3000);
 }
  else if (kb=='5')
 {
   Send_SMS_admin(1,'M',0);     cnt_reset=0;
 }
```

```
            else if (kb=='6')
            {
             Send_SMS_admin(1,'H',0);cnt_reset=0;
            }
            else if (kb=='0')
            {
                yy=1;
                cnt_reset=0;
                lcd_clear();
                while(yy <= 72)
                {
                    sprintf(temp_lcd,"H%i=%i H%i=%i  ",yy,Rain[yy],yy+1,Rain[yy+1]);
lcd_gotoxy(0,0); lcd_puts(temp_lcd);
                    sprintf(temp_lcd,"H%i=%i H%i=%i  ",yy+2,Rain[yy+2],yy+3,Rain[yy+3]);
lcd_gotoxy(0,1); lcd_puts(temp_lcd);
                    sprintf(temp_lcd,"H%i=%i H%i=%i  ",yy+4,Rain[yy+4],yy+5,Rain[yy+5]);
lcd_gotoxy(0,2); lcd_puts(temp_lcd);
                    sprintf(temp_lcd,"H%i=%i H%i=%i  ",yy+6,Rain[yy+6],yy+7,Rain[yy+7]);
lcd_gotoxy(0,3); lcd_puts(temp_lcd);
                    delay_ms(500);
                    loop=1;
                    while(loop)
                    {
                      kb = scan_keypad();              //เช็คว่ามีการกด คีย์แพดหรือไม่
                      if (kb=='0') loop=0;
                    }
                   yy =yy+8;
                }
            }
        }
//*****************************************************************************
**************
```

```
// ADC0 = read_adc(0);
/*
if ((ADC0 > MIN1) && (flg_send1==0))          //เช็คว่า เซ็นเซอร์ 1 เกิน 900 หรือไม่
{
    Send_SMS1(1);     //ส่ง SMS ให้เบอร์ที่ 1
    flg_send1 =1;
    RX=0;
    while(RX==0)
    {
        printf("AT"); putchar(0x0d);
        delay_ms(2000);
    }
    RX=0;
    while(RX==0)
    {
        printf("AT+CMGF = 1"); putchar(0x0d);
        delay_ms(2000);
    }
     RX=0;
    while(RX==0)
    {
        printf("AT"); putchar(0x0d);
        delay_ms(2000);
    }

    Send_SMS2(1);       //ส่ง SMS ให้เบอร์ที่ 2
  }
ADC0 = read_adc(0);
if  ((ADC0 > MID1) && (flg_send2==0))          //เช็คว่า เซ็นเซอร์ 1 เกิน 900 หรือไม่
{
    Send_SMS1(2);     //ส่ง SMS ให้เบอร์ที่ 1
    flg_send2 =1;
```

```
        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        RX=0;
        while(RX==0)
        {
            printf("AT+CMGF = 1");  putchar(0x0d);
            delay_ms(2000);
        }
        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        Send_SMS2(2);       //ส่ง SMS ให้เบอร์ที่ 2
    }
ADC0 = read_adc(0);
 if  ((ADC0 > MAX1) && (flg_send3==0))        //เช็คว่า เซ็นเซอร์ 1 เกิน 900 หรือไม่
{
        Send_SMS1(3);     //ส่ง SMS ให้เบอร์ที่ 1
        flg_send3 =1;
        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        RX=0;
```

```
            while(RX==0)
            {
                printf("AT+CMGF = 1");  putchar(0x0d);
                delay_ms(2000);
            }
             RX=0;
            while(RX==0)
            {
                printf("AT");  putchar(0x0d);
                delay_ms(2000);
            }
            Send_SMS2(3);        //ส่ง SMS ให้เบอร์ที่ 2
        }

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

        if  ((ADC1 == MIN2) && (flg_send4==0))          // เช็คว่า Sensor 2 เกิน 10 หรือไม่
        {
            Send_SMS1(4);              //ส่ง SMS ให้เบอร์ที่ 1
            flg_send4 =1;
            RX=0;
            while(RX==0)
            {
                printf("AT");  putchar(0x0d);
                delay_ms(2000);
            }
            RX=0;
            while(RX==0)
            {
                printf("AT+CMGF = 1");  putchar(0x0d);
                delay_ms(2000);
```

```
        }
         RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        Send_SMS2(4);                //ส่ง SMS ให้เบอร์ที่ 2
}
  if  ((ADC1 == MID2) && (flg_send5==0))          // เช็คว่า Sensor 2 เกิน 10 หรือไม่
{
        Send_SMS1(5);                //ส่ง SMS ให้เบอร์ที่ 1
        flg_send5 =1;

        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        RX=0;
        while(RX==0)
        {
            printf("AT+CMGF = 1");  putchar(0x0d);
            delay_ms(2000);
        }
         RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
```

```
        Send_SMS2(5);              //ส่ง SMS ให้เบอร์ที่ 2
    }
     if  ((ADC1 == MAX2) && (flg_send6==0))          // เช็คว่า Sensor 2 เกิน 10 หรือไม่
    {
        Send_SMS1(6);              //ส่ง SMS ให้เบอร์ที่ 1
        flg_send6 =1;

        RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
         RX=0;
        while(RX==0)
        {
            printf("AT+CMGF = 1");  putchar(0x0d);
            delay_ms(2000);
        }
          RX=0;
        while(RX==0)
        {
            printf("AT");  putchar(0x0d);
            delay_ms(2000);
        }
        Send_SMS2(6);              //ส่ง SMS ให้เบอร์ที่ 2
     }
    */
     };
}
```

ข.2 รหัสต้นฉบับภาครับ

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.8b Professional
Automatic Program Generator
© Copyright 1998-2006 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 10/10/2010
Author  : F4CG
Company : F4CG
Comments:

Chip type           : ATmega64
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024
*****************************************************/
#include <mega64.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
#define   MAX1  800
#define   MAX2  500
#define COL1 PORTA.0
#define COL2 PORTA.1
#define COL3 PORTA.2
#define ROW1 PINA.3
```

```
#define ROW2 PINA.4
#define ROW3 PINA.5
#define ROW4 PINA.6
int cnt_run  =0;
char Val1[4],Val2[4];
#define  BUZZER  PORTD.0
char Status_rf ;
int round;
int cnt_buzzer;
eeprom  int index_save2;
char id_error,st_error,RUN=0;
char TEL_CALL[30];
char DATA_SAVE[80];
char index_pood=0;
int sec,time0;
int cnt_plus,cnt_read,cnt_save=0;
int round_set,cnt_pood=0;
eeprom int index_write;
eeprom char BUFFER_SAVE[200];
int index_save;
int index_read=0;
char test=0;
char status_r,RXD=0;
int time_reset=0;
int  ii,h,m,s,dd,mm,yy;
char tel[10]={'0','8','4','0','4','8','8','3','1','3'};
//char tel[10]={'0','8','3','1','5','4','4','9','1','8'};
char status_gps=0;
char loop=0,flg_send1,flg_send2;
//
char USB_COMPLETE=0;
char kb,temp_lcd[16];
```

```
int  ADC0=0;
int  i,ADC1=0;

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
// USART0 Receiver buffer
#define RX_BUFFER_SIZE0 64
char rx_buffer0[RX_BUFFER_SIZE0];
#if RX_BUFFER_SIZE0<256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif
// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;
// USART0 Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
```

```
{
char status,data;
status=UCSR0A;
data=UDR0;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer0[rx_wr_index0]=data;
  if ((data !=0x0d) && (data!=0x0a))
  {
     if  (cnt_pood < 3) { index_pood=0;   index_save=0; }
      if (cnt_pood ==3)
     {
       TEL_CALL[index_pood] =  data;
       index_pood++;
     }
     if (cnt_pood == 8)
     {
          if (data != 0)
          {
             DATA_SAVE[index_save]= data;
             index_save++;
          }
     }
       if (data=='K')
     {
        RXD=1;  status_gps=1;
     }
     else  if (data=="")
     {
      cnt_pood++;
      if   (cnt_pood==8)
      {
```

```
            cnt_read=0;  rx_wr_index0=0;
            index_save=0;
          }
        }
      }
    if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
    if (++rx_counter0 == RX_BUFFER_SIZE0)
      {
      rx_counter0=0;
      rx_buffer_overflow0=1;
      };
    };
}
#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART0 Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter0==0);
data=rx_buffer0[rx_rd_index0];
if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
#asm("cli")
--rx_counter0;
#asm("sei")
return data;
}
#pragma used-
#endif
// USART0 Transmitter buffer
#define TX_BUFFER_SIZE0 8
```

```
char tx_buffer0[TX_BUFFER_SIZE0];
#if TX_BUFFER_SIZE0<256
unsigned char tx_wr_index0,tx_rd_index0,tx_counter0;
#else
unsigned int tx_wr_index0,tx_rd_index0,tx_counter0;
#endif
// USART0 Transmitter interrupt service routine
interrupt [USART0_TXC] void usart0_tx_isr(void)
{
if (tx_counter0)
  {
  --tx_counter0;
  UDR0=tx_buffer0[tx_rd_index0];
  if (++tx_rd_index0 == TX_BUFFER_SIZE0) tx_rd_index0=0;
  };
}
#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART0 Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter0 == TX_BUFFER_SIZE0);
#asm("cli")
if (tx_counter0 || ((UCSR0A & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer0[tx_wr_index0]=c;
  if (++tx_wr_index0 == TX_BUFFER_SIZE0) tx_wr_index0=0;
  ++tx_counter0;
  }
else
  UDR0=c;
```

```
#asm("sei")
}
#pragma used-
#endif
// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 64
char rx_buffer1[RX_BUFFER_SIZE1];
#if RX_BUFFER_SIZE1<256
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
#else
unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
#endif
// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;
// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
{
char status,data;
status=UCSR1A;
data=UDR1;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer1[rx_wr_index1]=data;
 /*
  if ((status_r == 1) && (cnt_read<21) && (data >='0')&& (data<='9'))
  {
     if  (cnt_read==0)  lcd_gotoxy(0,1);
     else if  (cnt_read==16)  lcd_gotoxy(0,2);
     lcd_putchar(data);
     cnt_read++;
  } */
  if (++rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
```

```
 if (++rx_counter1 == RX_BUFFER_SIZE1)
    {
    rx_counter1=0;
    rx_buffer_overflow1=1;
    };
  };
}
// Get a character from the USART1 Receiver buffer
#pragma used+
char getchar1(void)
{
char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1];
if (++rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#asm("cli")
--rx_counter1;
#asm("sei")
return data;
}
#pragma used-
// USART1 Transmitter buffer
#define TX_BUFFER_SIZE1 8
char tx_buffer1[TX_BUFFER_SIZE1];
#if TX_BUFFER_SIZE1<256
unsigned char tx_wr_index1,tx_rd_index1,tx_counter1;
#else
unsigned int tx_wr_index1,tx_rd_index1,tx_counter1;
#endif
// USART1 Transmitter interrupt service routine
interrupt [USART1_TXC] void usart1_tx_isr(void)
{
```

```
if (tx_counter1)
  {
  --tx_counter1;
  UDR1=tx_buffer1[tx_rd_index1];
  if (++tx_rd_index1 == TX_BUFFER_SIZE1) tx_rd_index1=0;
  };
}
// Write a character to the USART1 Transmitter buffer
#pragma used+
void putchar1(char c)
{
while (tx_counter1 == TX_BUFFER_SIZE1);
#asm("cli")
if (tx_counter1 || ((UCSR1A & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer1[tx_wr_index1]=c;
  if (++tx_wr_index1 == TX_BUFFER_SIZE1) tx_wr_index1=0;
  ++tx_counter1;
  }
else
  UDR1=c;
#asm("sei")
}
#pragma used-
// Standard Input/Output functions
#include <stdio.h>
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
      if (time0++>48000)
      {
```

```
            time0=0;
            sec++;
            cnt_buzzer++;
        }
    }
#define ADC_VREF_TYPE 0x00
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
// Declare your global variables here
char scan_keypad (void)
{
  char key=0;
  PORTA=0b11111111;
  COL1=0; COL2=1;  COL3=1; delay_ms(20);
  if(ROW1==0)    {key='1'; while(ROW1==0);}
  else if(ROW2==0) {key='4';while(ROW2==0);}
  else if(ROW3==0) {key='7';while(ROW3==0);}
  else if(ROW4==0) {key='*';while(ROW4==0);}



  COL1=1; COL2=0;  COL3=1; delay_ms(20);
  if(ROW1==0)    {key='2';while(ROW1==0);}
```

```
  else if(ROW2==0)  {key='5';while(ROW2==0);}
  else if(ROW3==0)  {key='8';while(ROW3==0);}
  else if(ROW4==0)  {key='0';while(ROW4==0);}

  COL1=1; COL2=1;  COL3=0; delay_ms(20);
  if(ROW1==0)    {key='3';while(ROW1==0);}
  else if(ROW2==0)  {key='6';while(ROW2==0);}
  else if(ROW3==0)  {key='9';while(ROW3==0);}
  else if(ROW4==0)  {key='#';while(ROW4==0);}
 return key;
}
 /*
void Send_SMS(char ch)
{
        sprintf(temp_lcd," Sending SMS  ");
        lcd_gotoxy(0,0);lcd_puts(temp_lcd);
        time_reset=0;
        RX=0;
        while(RX==0)
        {
           printf("AT"); putchar(0x0d);
           delay_ms(1000);
           time_reset++;
           if (time_reset>10) RX=1;
        }


        printf("AT+CMGS =");
        putchar("");
printf("%c%c%c%c%c%c%c%c%c%c",tel[0],tel[1],tel[2],tel[3],tel[4],tel[5],tel[6],tel[7],tel[8],tel
[9]);
        putchar("");
```

```
            putchar(0x0d);delay_ms(1000);
            putchar('$');
            putchar(h/10+'0'); putchar(h%10+'0'); putchar(m/10+'0'); putchar(m%10+'0');
putchar(s/10+'0'); putchar1(s%10+'0');
putchar(dd/10+'0');putchar(dd%10+'0');putchar(mm/10+'0');putchar(mm%10+'0');putchar(yy/10+'
0');putchar(yy%10+'0');
            if (ch==1) {putchar(ADC0/100+'0');putchar(ADC0%100/10+'0');
putchar(ADC0%10+'0');  }
            if (ch==2) {putchar(ADC1/100+'0');putchar(ADC1%100/10+'0');
putchar(ADC1%10+'0');  }
            putchar('!');
            putchar(0x1A);  delay_ms(1000);
}  */
 /*
void  read_data_from_eeprom(void)
{
            char i,s =0;
            char loop2=0;
           // putchar1('I');putchar1('P');putchar1('A');putchar1(0x0d);   delay_ms(500);
           // putchar1('O');putchar1('P');putchar1('R');putchar1('
');putchar1('D');putchar1('A');putchar1('T');putchar1('A');putchar1('.');putchar1('T');putchar1('X');p
utchar1('T');putchar1(0x0d); delay_ms(500);
            if  (index_write > 5)  s=index_write-5;
            else s=0;
            for (i=s;i<index_write;i++)
            {

                putchar1('S');putchar1('E');putchar1('K');putchar1(' ');
                if   ((index_write-1)*25 < 10)
                {
                      putchar1((index_write-1)*25%10+'0');
                }
```

```
                else if ((index_write-1)*25 < 100)
                {
                    putchar1(((index_write-1)*25)/10+'0');
                    putchar1(((index_write-1)*25)%10+'0');
                }
                 else if ((index_write-1)*25 < 9999)
                {
                    putchar1(((index_write-1)*25)/100+'0');
                    putchar1(((index_write-1)*25)%100/10+'0');
                    putchar1(((index_write-1)*25)%10+'0');
                }
                putchar1(0x0d);  delay_ms(500);
                lcd_clear();
                sprintf(temp_lcd,"index %i/%i [%i]",i+1,index_write,(index_write-1)*25);
                lcd_gotoxy(0,0);lcd_puts(temp_lcd);

                putchar1('R');putchar1('D');putchar1('F');putchar1(' ');putchar1('2');putchar1('5');
putchar1(0x0d);
                status_r = 1;
                cnt_read=0;
                delay_ms(4000);
                status_r = 0;
                loop2=1;
                while(loop2)
                {
                    kb = scan_keypad();
                    if (kb!=0)
                    {
                        if (kb=='5') loop2=0;
                    }
                }
            }
```

```
} */
void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0xff;
DDRA=0x07;
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;
DDRD.0=1; // 1 output  0 input
// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;
```

```
// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;
// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 16000.000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
```

```
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
```

```
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;
ETIMSK=0x00;
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
```

```
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0xD8;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;
// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud rate: 9600
UCSR1A=0x00;
UCSR1B=0xD8;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x67;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x84;
// LCD module initialization
lcd_init(16);
```

```
// Global enable interrupts
#asm("sei")
   printf("Test RS232\r\n");
   delay_ms(2000);
   test=0;
   RXD=0;
   status_gps=0;
   if (test==0)                          //ทำการ initial module gps
   {
         sprintf(temp_lcd,"  initial GPRS  ");
         lcd_gotoxy(0,0);lcd_puts(temp_lcd);
         sprintf(temp_lcd," Please wait ...");
         lcd_gotoxy(0,1);lcd_puts(temp_lcd);
         delay_ms(2000);
         sprintf(temp_lcd,"  initial SMS   ");
         lcd_gotoxy(0,2);lcd_puts(temp_lcd);
         sprintf(temp_lcd," Please wait ...");
         lcd_gotoxy(0,3);lcd_puts(temp_lcd);
         RXD=0;
         while(RXD==0)                 //รอจนกว่าจะมี OK กลับมา
         {
            //printf("AT+CMGF = 1");  putchar(0x0d);
            printf("AT");  putchar(0x0d);
            delay_ms(1000);
         }
         RXD=0;
         while(RXD==0)                 //รอจนกว่าจะมี OK กลับมา
         {
            printf("AT+CFUN = 1");  putchar(0x0d);
            delay_ms(1000);
         }
         RXD=0;
```

```
        while(RXD==0)           //รอจนกว่าจะมี OK กลับมา
        {
            printf("AT+CMGF = 1");  putchar(0x0d);
            delay_ms(1000);
        }
    }
    lcd_clear();
    tel[10]=' ';
    lcd_gotoxy(0,1); lcd_puts(tel);
    sprintf(temp_lcd,"  Initial OK  ");
    lcd_gotoxy(0,2);lcd_puts(temp_lcd);
    delay_ms(1000);
    BUZZER=0;delay_ms(500);
   status_r=0;
   flg_send1=0;
   flg_send2=0;
   index_read=0;
   RUN=0;
   cnt_save=0;
while (1)
   {            //for(i=3;i<14;i++)   TEL_CALL[i] = '1';
      if ((Status_rf=='L')&& (round<10))          //หากสถานะของ RF = LOW และจำนวน
ครั้งในการดังไม่ถึง 10 ครั้ง
    {
       if (cnt_buzzer<=10)          BUZZER=1;     // ดัง  10 วินาที
       else if (cnt_buzzer<=50)     BUZZER=0;
       else  {cnt_buzzer=0;  round++;
       if     (round >=10)       Status_rf=0;
       }
      round_set=10;
    }
    else if ((Status_rf=='M')&& (round<20))
```

```
{
    if (cnt_buzzer<=20)          BUZZER=1;
    else if (cnt_buzzer<=60)     BUZZER=0;
    else  {cnt_buzzer=0;  round++;  if    (round >=20)      Status_rf=0;}
    round_set=20;
}
else if ((Status_rf=='H')&& (round<30))
{
    if (cnt_buzzer<=30)          BUZZER=1;
    else if (cnt_buzzer<=60)     BUZZER=0;
    else  {cnt_buzzer=0;  round++;  if    (round >=30)      Status_rf=0; }
    round_set=30;
}
//else   {round_set=0;round=0;}
 kb = scan_keypad();
 if (kb!=0)
 {
          //      $ xxxxxx yyyyyy 1 aaa bbb ccc h !
     if (kb=='*')
    {
       index_save=0;  cnt_save=0;  index_read=0;  index_save2=0;
       lcd_clear();
       sprintf(temp_lcd,"Clear EEprom OK ");
       lcd_gotoxy(0,2); lcd_puts(temp_lcd);

       for (i=0;i<200;i++)
       BUFFER_SAVE[i]=' ';
       delay_ms(500);

    }
    else if (kb=='0')
    {
```

```
                    lcd_clear();
                    if (id_error=='1')
                    {
                        sprintf(temp_lcd," Flash [RF] %c%c%c%c
",DATA_SAVE[13],DATA_SAVE[14],DATA_SAVE[15],DATA_SAVE[16]);
                        lcd_gotoxy(0,1); lcd_puts(temp_lcd);
                     }
                    else if (id_error=='2')
                    {
                        sprintf(temp_lcd," Flash [WT] %c%c%c
",DATA_SAVE[17],DATA_SAVE[18],DATA_SAVE[19]);
                        lcd_gotoxy(0,1); lcd_puts(temp_lcd);
                    }
                     else if (id_error=='3')
                     {
                        sprintf(temp_lcd," Flash [SM] %c%c%c
",DATA_SAVE[20],DATA_SAVE[21],DATA_SAVE[22]);
                        lcd_gotoxy(0,1); lcd_puts(temp_lcd);
                     }

                  if (Status_rf=='L')        {sprintf(temp_lcd," Low ");   BUZZER=1;
RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2); lcd_puts(temp_lcd);}
                    else if (Status_rf=='M')      {sprintf(temp_lcd," Middle ");
BUZZER=1; RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2); lcd_puts(temp_lcd);}
                    else if (Status_rf=='H')      {sprintf(temp_lcd," Hight ");
BUZZER=1; RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2); lcd_puts(temp_lcd);}
                     delay_ms(5000);
        }
        else  if (kb=='2')
        {
              BUZZER=1; delay_ms(500); BUZZER=0;
         }
```

```
        else  if (kb=='8')
        {
              BUZZER=0;
              Status_rf=0;
              round=0;
              round_set=0;
         }
        else  if (kb=='1')  // <<
        {
            if (index_read>0)  index_read--;
            ii = index_read*24;
            lcd_clear();
            sprintf(temp_lcd,"  index %i  ",index_read);  lcd_gotoxy(0,0);  lcd_puts(temp_lcd);
            sprintf(temp_lcd,"Time : ");  lcd_gotoxy(0,1);  lcd_puts(temp_lcd);
            lcd_putchar(BUFFER_SAVE[ii]); lcd_putchar(BUFFER_SAVE[ii+1]);
lcd_putchar(':');
            lcd_putchar(BUFFER_SAVE[ii+2]); lcd_putchar(BUFFER_SAVE[ii+3]);
lcd_putchar(':');
            lcd_putchar(BUFFER_SAVE[ii+4]); lcd_putchar(BUFFER_SAVE[ii+5]);
            sprintf(temp_lcd,"Date : ");  lcd_gotoxy(0,2);  lcd_puts(temp_lcd);
            lcd_putchar(BUFFER_SAVE[ii+6]); lcd_putchar(BUFFER_SAVE[ii+7]);
lcd_putchar('/');
            lcd_putchar(BUFFER_SAVE[ii+8]); lcd_putchar(BUFFER_SAVE[ii+9]);
lcd_putchar('/');
            lcd_putchar(BUFFER_SAVE[ii+10]); lcd_putchar(BUFFER_SAVE[ii+11]);
//lcd_putchar('/');
            lcd_gotoxy(0,3);
            lcd_putchar(BUFFER_SAVE[ii+12]);
            lcd_putchar('|');
            lcd_putchar(BUFFER_SAVE[ii+13]);
lcd_putchar(BUFFER_SAVE[ii+14]);lcd_putchar(BUFFER_SAVE[ii+15]);lcd_putchar(BUFFE
R_SAVE[ii+16]);
```

```
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+17]);
lcd_putchar(BUFFER_SAVE[ii+18]);lcd_putchar(BUFFER_SAVE[ii+19]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+20]);
lcd_putchar(BUFFER_SAVE[ii+21]);lcd_putchar(BUFFER_SAVE[ii+22]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+23]);
                //for(i=ii;i<ii+16;i++)     {lcd_putchar(BUFFER_SAVE[i]);  }
                //lcd_gotoxy(0,3);   for(i=ii+16;i<ii+20;i++)  {lcd_putchar(BUFFER_SAVE[i]);  }

                delay_ms(2500);
            }
             else  if (kb=='3')  //>>
            {
                index_read++;
                ii = index_read*24;
                lcd_clear();
                sprintf(temp_lcd,"  index %i  ",index_read);  lcd_gotoxy(0,0);  lcd_puts(temp_lcd);
                sprintf(temp_lcd,"Time : ");  lcd_gotoxy(0,1);  lcd_puts(temp_lcd);
                lcd_putchar(BUFFER_SAVE[ii]); lcd_putchar(BUFFER_SAVE[ii+1]);
lcd_putchar(':');
                lcd_putchar(BUFFER_SAVE[ii+2]); lcd_putchar(BUFFER_SAVE[ii+3]);
lcd_putchar(':');
                lcd_putchar(BUFFER_SAVE[ii+4]); lcd_putchar(BUFFER_SAVE[ii+5]);
                sprintf(temp_lcd,"Date : ");  lcd_gotoxy(0,2);  lcd_puts(temp_lcd);
                lcd_putchar(BUFFER_SAVE[ii+6]); lcd_putchar(BUFFER_SAVE[ii+7]);
lcd_putchar('/');
                lcd_putchar(BUFFER_SAVE[ii+8]); lcd_putchar(BUFFER_SAVE[ii+9]);
lcd_putchar('/');
                lcd_putchar(BUFFER_SAVE[ii+10]); lcd_putchar(BUFFER_SAVE[ii+11]);
//lcd_putchar('/');
```

```
                lcd_gotoxy(0,3);
                lcd_putchar(BUFFER_SAVE[ii+12]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+13]);
lcd_putchar(BUFFER_SAVE[ii+14]);lcd_putchar(BUFFER_SAVE[ii+15]);lcd_putchar(BUFFE
R_SAVE[ii+16]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+17]);
lcd_putchar(BUFFER_SAVE[ii+18]);lcd_putchar(BUFFER_SAVE[ii+19]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+20]);
lcd_putchar(BUFFER_SAVE[ii+21]);lcd_putchar(BUFFER_SAVE[ii+22]);
                lcd_putchar('|');
                lcd_putchar(BUFFER_SAVE[ii+23]);
                 //lcd_gotoxy(0,2);   for(i=ii;i<ii+16;i++)     {lcd_putchar(BUFFER_SAVE[i]);  }
                //lcd_gotoxy(0,3);   for(i=ii+16;i<ii+20;i++)  {lcd_putchar(BUFFER_SAVE[i]);  }
                delay_ms(2500);
            }
        }
        lcd_clear();
        lcd_gotoxy(0,1);lcd_putsf("  Stand By SMS ");
        sprintf(temp_lcd," Time remain %i ",80-cnt_save);  lcd_gotoxy(0,2); lcd_puts(temp_lcd);
        sprintf(temp_lcd," Alarm %02i/%02i ",round,round_set);  lcd_gotoxy(0,3);
lcd_puts(temp_lcd);
        delay_ms(200);
        if (RUN==1)
        {
            if (cnt_run++ > 50)
            {
                RUN=0;
                BUZZER=0;
                cnt_run=0;
```

```
        }
    }
     if (cnt_save++ > 80)  // 20 sec
    {
            cnt_save=0;
            lcd_clear();
            index_save=0;
            status_r=1;
            cnt_read=0;
            cnt_pood=0;
            RXD=0;
            lcd_clear();
            lcd_gotoxy(0,0); lcd_putsf("Waitting ....");
            printf("AT+CMGR=1\r");
             time_reset=0;
            while(RXD==0)
            {
                if (time_reset++ > 20 ) RXD=1;
                delay_ms(100);
            }
            lcd_gotoxy(0,1);
            TEL_CALL[0]='0';
            TEL_CALL[1]='8';
            for(i=4;i<12;i++)
            {    TEL_CALL[i-2]=TEL_CALL[i];
            }
             if ( TEL_CALL[0] == tel[0])
            if ( TEL_CALL[1] == tel[1])
            if ( TEL_CALL[2] == tel[2])
            if ( TEL_CALL[3] == tel[3])
            if ( TEL_CALL[4] == tel[4])
            if ( TEL_CALL[5] == tel[5])
```

```
                if ( TEL_CALL[6] == tel[6])
                if ( TEL_CALL[7] == tel[7])
                if ( TEL_CALL[8] == tel[8])
                if ( TEL_CALL[9] == tel[9])
                {
                    for(i=0;i<10;i++)
                    lcd_putchar(TEL_CALL[i]);
                   //$ 12020 235465 111 222 333 H !
                    lcd_gotoxy(0,2);   for(i=0;i<16;i++)   {lcd_putchar(DATA_SAVE[i]);
delay_ms(15); }
                    lcd_gotoxy(0,3);   for(i=16;i<24;i++)  {lcd_putchar(DATA_SAVE[i]);
delay_ms(15); }
                   delay_ms(1000);
                      BUFFER_SAVE[index_save2]= DATA_SAVE[1];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[2];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[3];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[4];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[5];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= '0';          index_save2++;
                      BUFFER_SAVE[index_save2]= DATA_SAVE[6];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[7];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[8];  index_save2++;
//
                      BUFFER_SAVE[index_save2]= DATA_SAVE[9];  index_save2++;
//
```

```
                    BUFFER_SAVE[index_save2]= DATA_SAVE[10];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[11];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[12];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[13];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[14];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[15];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[16];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[17];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[18];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[19];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[20];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[21];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[22];  index_save2++;
                    BUFFER_SAVE[index_save2]= DATA_SAVE[23];  index_save2++;
                     id_error = DATA_SAVE[12];
                    Status_rf = DATA_SAVE[23];
                    lcd_clear();
                    if  (id_error=='1')
                    {
                        sprintf(temp_lcd," Flash [RF] %c%c%c%c
",DATA_SAVE[13],DATA_SAVE[14],DATA_SAVE[15],DATA_SAVE[16]);
                        lcd_gotoxy(0,1);  lcd_puts(temp_lcd);
                    }
                     else if  (id_error=='2')
                    {
                        sprintf(temp_lcd," Flash [WT]  %c%c%c
",DATA_SAVE[17],DATA_SAVE[18],DATA_SAVE[19]);
                        lcd_gotoxy(0,1);  lcd_puts(temp_lcd);
                     }
                    else if  (id_error=='3')
                    {
}
```

```
                    sprintf(temp_lcd," Flash [SM]  %c%c%c
",DATA_SAVE[20],DATA_SAVE[21],DATA_SAVE[22]);
                    lcd_gotoxy(0,1);  lcd_puts(temp_lcd);
                }
                if (Status_rf=='L')          {sprintf(temp_lcd," Low  ");   BUZZER=1;
RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2);  lcd_puts(temp_lcd);round=0;}
                else if (Status_rf=='M')     {sprintf(temp_lcd," Middle  ");
BUZZER=1;  RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2);  lcd_puts(temp_lcd);round=0;}
                else if (Status_rf=='H')     {sprintf(temp_lcd," Hight  ");
BUZZER=1;  RUN=1; cnt_buzzer=0;lcd_gotoxy(0,2);  lcd_puts(temp_lcd);round=0;}
                //delay_ms(5000);
              }
             printf("AT+CMGD=1\r");   delay_ms(1000);
            // printf("AT+CMGD=2\r");   delay_ms(1000);
            //  printf("AT+CMGD=3\r");   delay_ms(1000);
             TEL_CALL[0]=0;
             TEL_CALL[1]=0;
             TEL_CALL[2]=0;
             TEL_CALL[3]=0;
             TEL_CALL[4]=0;
             TEL_CALL[5]=0;
             TEL_CALL[6]=0;
             TEL_CALL[7]=0;
             TEL_CALL[8]=0;
             TEL_CALL[9]=0;
             delay_ms(2000);
        }
    };
}
```