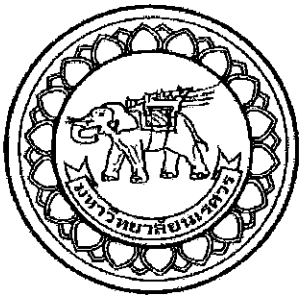


การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ

Processing Unit and Warning System over Mobile Phone

นางสาวศิริโฉม ทองอุบล รหัส 48361882

นายสรายุทธ พันธ์วงค์ รหัส 48361929

นายธงชัย ทุมมี รหัส 48364357

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2551

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ
ผู้ดำเนินโครงงาน	นางสาวศิริโฉม ทองอุบล รหัส 48361882
	นายสรายุทธ พันธ์วงค์ รหัส 48361929
	นายธงชัย ทุมมี รหัส 48364357
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2551

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง)

.....กรรมการ

(ดร.อัครพันธ์ วงศ์กังแห)

.....กรรมการ

(ดร.ศุภวรรณ พลพิทักษ์ชัย)

หัวข้อโครงงาน	การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ			
ผู้ดำเนินงาน	นางสาวศิริโฉม	ทองอุบล	รหัส	48361882
	นายสรายุทธ	พันธ์วงค์	รหัส	48361929
	นายธงชัย	ทุมมี	รหัส	48364357
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง			
สาขาวิชา	วิศวกรรมไฟฟ้า			
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2551			

บทคัดย่อ

โครงงานชิ้นนี้เกิดขึ้นเนื่องจาก การประสบปัญหาภัยธรรมชาติ ซึ่งทำให้เกิดความสูญเสียทั้งด้านทรัพย์สินเงินทองและชีวิต ทำให้ผู้คนต้องล้มตาย เพราะความรุนแรงของภัยธรรมชาติ โดยเฉพาะที่ใกล้ตัวเราที่สุด คือเหตุการณ์พายุดินถล่มที่จังหวัดอุตรดิตถ์ เป็นพายุดินถล่มตามแถบลุ่มเชิงเขา ทำให้เกิดความคิดที่จะประยุกต์การสร้างอุปกรณ์เตือนภัยให้กับประชาชนในเขตใกล้เคียงได้ทราบถึงภัยที่จะเกิดขึ้นอย่างมีประสิทธิภาพ และเพื่อการพัฒนาที่ดีต่อไป เพื่อประมวลผลในการตัดสินใจ แล้วส่งมาทำการประมวลผลในการส่งข้อมูลทางโทรศัพท์เพื่อแจ้งเตือนภัยทางโทรศัพท์มือถือ โดยได้มีการนำไมโครคอนโทรลเลอร์มาใช้ในการควบคุมระบบ อุปกรณ์ที่ใช้จะเป็นบอร์ดโทรศัพท์ ควบคุมระบบโดยผ่านการส่งข้อความ ซึ่งมีการตั้งค่าข้อความตามที่เราต้องการให้ส่งเพื่อเตือนภัย สามารถเก็บข้อมูลและนำมาใช้ได้อย่างมีประสิทธิผล

Project Title	Processing Unit and Warning System over Mobile Phone
Name	Miss Sirichom Thongubol ID 48361882
	Mr Sarayoot Phanwong ID 48361929
	Mr Thongchai Tummee ID 48364357
Project Advisor	Assistant Professor Dr. Yongyut Chonbodeechalermroong
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic Year	2008

ABSTRACT

This project was originated because of facing with natural disaster problems that brought loss of both assets and life. Many people died because of natural disaster violence, especially the closest situation, the landslide in Uttaradit which was a landslide from storms along foot of the hill. The idea of applying a warning alarm was originated for people in the vicinity to efficiently realize, further develop, and make decisions. The data will be collected and will be transmitted to send the data to mobile phones for warning. The micro controller is used to control the equipment system which is a telephone board. The system is controlled by sending messages, which were set as our requirements for sending us the warning alarm. The data can be collected and used efficiently.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง อาจารย์ที่ปรึกษาโครงงาน ที่คอยให้คำปรึกษาและให้ความช่วยเหลือตลอดจนคำแนะนำต่างๆในการทำโครงงานชิ้นนี้ สุดท้าย ต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆพี่ๆทุกคนที่ยังไม่ได้เอ่ยนามที่ให้คำแนะนำและให้การสนับสนุน ผู้จัดทำโครงงานให้สามารถทำโครงงานชิ้นนี้จนสำเร็จลุล่วงไปด้วยดี

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากในปัจจุบันภัยธรรมชาติได้เกิดขึ้นมากมายในหลายๆ ประเทศต่างประสบปัญหาจากภัยธรรมชาติ เช่น เหตุการณ์แผ่นดินไหวทางตอนใต้ของประเทศจีน ซึนามิที่ภาคใต้ของประเทศไทย พายุดินถล่มที่จังหวัดอุตรดิตถ์ เป็นต้น ภัยธรรมชาติเหล่านี้ล้วนก่อให้เกิดความสูญเสียทั้งทางด้านทรัพย์สินเงินทองและชีวิต ทำให้ผู้คนต้องล้มตายเป็นจำนวนมาก

ด้วยเหตุนี้ จึงเกิดความสนใจและเห็นถึงความสำคัญในการป้องกันภัยพิบัติที่จะเกิดขึ้น ดังนั้นจึงเกิดแนวคิดที่จะประยุกต์การสร้างอุปกรณ์เตือนภัยให้แก่ประชาชนในเขตพื้นที่ใกล้เคียงได้ทราบถึงภัยพิบัติที่จะเกิดขึ้นอย่างมีประสิทธิภาพ และเพื่อเป็นการพัฒนาที่ดีต่อไปสำหรับการประมวลผลในการตัดสินใจ แล้วส่งข้อมูลไปยังโทรศัพท์มือถือเพื่อแจ้งเตือนภัย

โครงงานนี้จะมุ่งเน้นในเรื่องการประมวลผลและแจ้งเตือนภัยทางโทรศัพท์มือถือ การสื่อสารแบบไร้สาย

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการรับส่งสัญญาณแบบไร้สาย
2. เพื่อศึกษาการพัฒนาวงจรการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

1. ศึกษาการรับส่งสัญญาณแบบไร้สาย
2. พัฒนาอุปกรณ์การประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ เพื่อสามารถรับรู้ถึงภัยของพายุดินถล่มที่จะเกิดขึ้นล่วงหน้า และสามารถป้องกันและอพยพประชาชนจากเหตุการณ์ได้ทันเวลา ทำให้ไม่สูญเสียชีวิตและทรัพย์สินอย่างที่ผ่านมา

1.4 ขั้นตอนการดำเนินงานและแผนการดำเนินงานตลอดโครงการวิจัย

รายละเอียด	ระยะเวลาดำเนินงาน(เดือน)											
	ปี2551								ปี2552			
	มิ.ย	ก.ค	ส.ค	ก.ย	ต.ค	พ.ย	ธ.ค	ม.ค	ก.พ	มี.ค	เม.ย.	พ.ค
1. รวบรวมข้อมูล	←	→										
2. ศึกษาการทำงานและรวบรวมเนื้อหา			←	→								
3. จัดทำอุปกรณ์และทำการทดลอง					←	→						
4. จัดทำรายงานรวบรวมข้อมูล สรุปผล เข้ารูปเล่มพร้อมทั้งชิ้นงาน เสนอแนะแนวทางพัฒนาต่อ และเตรียมนำเสนอนิทรรศการ									←	→		

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

1. เข้าใจถึงหลักการส่งสัญญาณแบบไร้สาย
2. เพื่อเป็นการแจ้งเตือนให้ประชาชนในบริเวณใกล้เคียงที่เกิดเหตุทราบถึงพายุดินถล่ม
3. สามมารถนำอุปกรณ์ของโครงงานนี้ไปใช้ในการให้บริการในพื้นที่เสี่ยงได้
4. เพื่อเป็นประโยชน์สำหรับประชาชนที่อยู่ในเขตพื้นที่เสี่ยง

1.6 งบประมาณในการทำโครงงาน

1.ถ่ายเอกสารและค่าเข้าเล่มโครงงานฉบับสมบรูณ์	500	บาท
2.ค่าอุปกรณ์ในการทำโครงงาน	2300	บาท
3.ค่าพิมพ์เอกสาร	200	บาท
รวมเป็นเงิน	3000	บาท

หมายเหตุ: ถัวเฉลี่ยทุกรายการ

บทที่2
หลักการและทฤษฎี

จากแนวคิดที่จะสร้างระบบสัญญาณเตือนภัยผ่านทางเครือข่ายโทรศัพท์มือถือ ซึ่งประกอบไปด้วยอุปกรณ์หลัก ๆ คือ บอร์ดโทรศัพท์ซึ่งจะนำไปเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ เพื่อที่จะนำมาเขียนโปรแกรมควบคุมระบบการทำงานของอุปกรณ์ต่าง ๆ โดยหลักการทำงานและทฤษฎีพื้นฐานของไมโครคอนโทรลเลอร์ที่ควรทราบมีดังนี้

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชันของการทำงานต่างๆ ไว้ภายในตัวของมันเอง โดยมีโครงสร้างใกล้เคียงกับคอมพิวเตอร์ คือ ภายในประกอบด้วยหน่วยรับข้อมูลโปรแกรมหน่วยประมวลผล หน่วยความจำ และ หน่วยแสดงผล ซึ่งส่วนประกอบเหล่านี้มีความสมบูรณ์ในตัวของมันเอง ทำให้มีขนาดเล็ก และ สามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวของมัน ซึ่งง่ายต่อการนำไปประยุกต์ใช้งาน

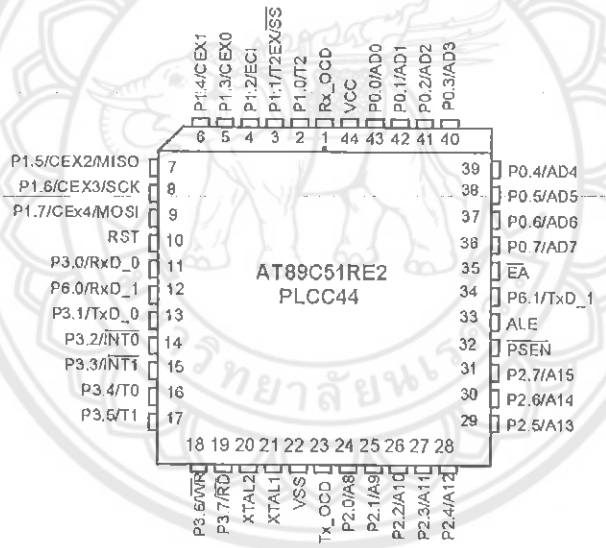
2.1 การเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์

2.1.1 ไมโครคอนโทรลเลอร์รุ่น "CP-JR51RE2 V1.0"

บอร์ดไมโครคอนโทรลเลอร์รุ่นนี้จะใช้ MCU ขนาด 8 บิต ของ Atmel เบอร์ # AT89C51RE2 ซึ่ง MCU ตัวนี้จะบรรจุในตัวถังแบบ PLCC ขนาด 44 ขา จุดเด่นของ MCU เบอร์นี้ คือ มี UART ให้ใช้งาน 2แชนแนล , มีTimer/Counter ขนาด 16 บิต , มีพื้นที่สำหรับ Flash โปรแกรมถึง 128 Kbyte และมีขนาด RAMมากถึง 8 Kbyte ให้ใช้งาน การจัดสรร Port I/O ของบอร์ดที่ได้ต่อขาออกมาไว้ให้ผู้ใช้ได้ใช้งานมีดังนี้ มี Port I/O = 4 Port , Port RS232 = 2Port , Port RS422/485 = 1 Port , Port LCD แบบ 4 bit 1 Port และวงจรสำหรับในส่วนของ RTC ที่ใช้กับ # DS1307ในส่วนของการ Download โปรแกรมลงบอร์ดนั้นจะ Download ผ่านทาง Port RS232 โดยใช้โปรแกรม FlipV3.1.0 เป็นตัว Download และใช้คอมไพเลอร์ Keil µVision3 เป็นตัวพัฒนาโปรแกรมด้วยภาษา C

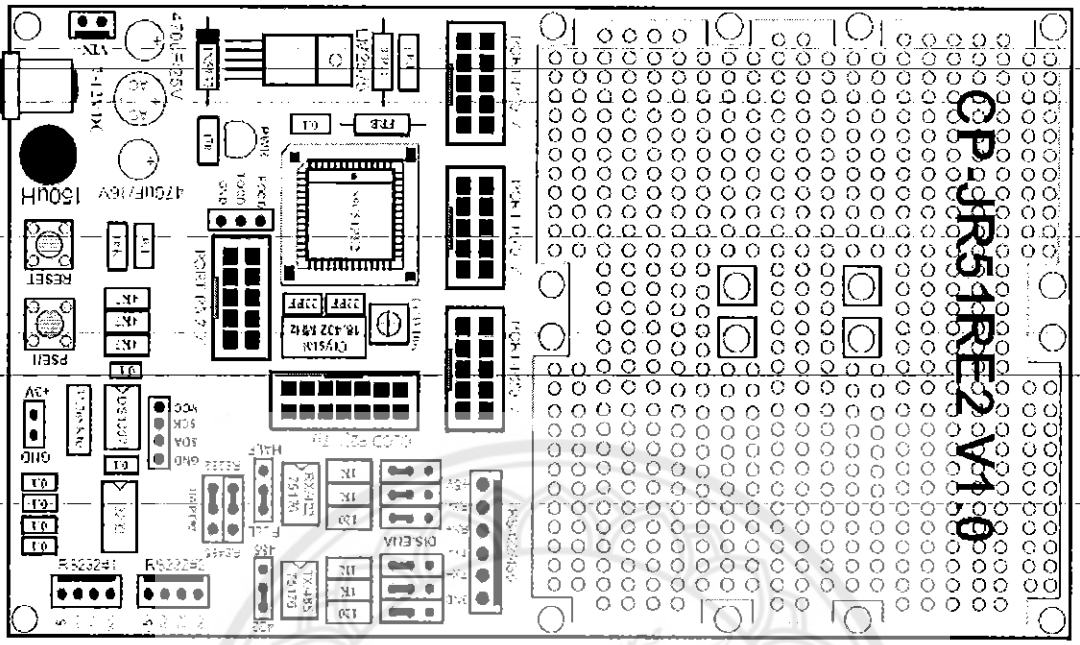
2.1.2 คุณสมบัติของบอร์ด CP-JR51RE2 และ MCU

- MCU เป็นตัวถังแบบ PLCC 44 Pin
- MCU ทำงานที่แรงดัน 2.7 - 5.5 V
- ความถี่ Crystal ที่ใช้งานบนบอร์ด 18.432 MHz
- หน่วยความจำ: Flash 128 KB, RAM 8KB
- การสื่อสารอนุกรมประกอบด้วย SPI 1 แชนแนล และ Uart 2 แชนแนล
- 16 บิต Timer/Counter สำหรับ Timer_0, Timer_1 และ Timer_2
- Watch-Dog Timer 14 bit Counter
- PORT I/O 34 PIN (P0-P3, P6.0, P6.1)
- 11 Interrupt Source ซึ่งกำหนดระดับความสำคัญของ Interrupt ได้ 4 ระดับ
- Download โปรแกรมด้วย Flip V3.1.0 ผ่านทาง RS232
- MCU ทำงานที่อุณหภูมิ – 40 ถึง +85 องศาเซลเซียส



รูปที่1 แสดงโครงสร้าง MCU AT89C51RE2

2. 2โครงสร้าง และการจัดสรร I/O ของบอร์ด CP-JR51RE2



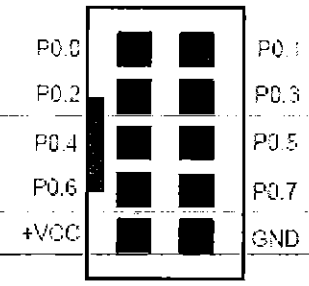
รูปที่ 2 แสดงลักษณะโครงสร้างของบอร์ด CP-JR51RE2 V1.0

2.2.1) แหล่งจ่ายไฟ: สำหรับแหล่งจ่ายไฟของบอร์ดนี้สามารถต่อใช้งานได้ทั้งไฟกระแสตรงและกระแสสลับ โดยป้อนแรงดันไฟตรงหรือไฟสลับที่มีระดับแรงดันประมาณ 9-12 V ให้กับบอร์ด ซึ่งสามารถเลือกต่อกับขั้ว Connector แบบ CPA ขนาด 2 ขา หรือจะต่อผ่านขั้ว Connector สำหรับ Adapter จ่ายไฟก็ได้เช่นกัน โดยจะแสดงผลการทำงานของแหล่งจ่ายไฟให้ทราบด้วย LED "PWR"

2.2.2) สัญญาณนาฬิกา CLOCK: ความถี่ของสัญญาณนาฬิกาที่จะนำไปป้อนให้กับตัว MCU #AT89C51RE2 นั้น \$ตามปกติแล้วสามารถป้อนค่าความถี่ของ Crystalได้ถึง 40MHz ใน Standard Mode (12 Clock / 1 Machine Cycle) แต่ในกรณีที่ให้ MCU ทำงานใน X2 Mode จะสามารถใช้ค่าความถี่สูงสุดได้ที่ 20 MHz โดยสำหรับบอร์ด CP-JR51RE2 นั้นจะกำหนดให้ใช้ค่าความถี่ของ Crystal ที่ป้อนให้กับ MCU ด้วยค่าความถี่ 18.432 MHz เพื่อให้การสื่อสารพอร์ตอนุกรมสามารถหาร Baud Rate ได้ลงตัว

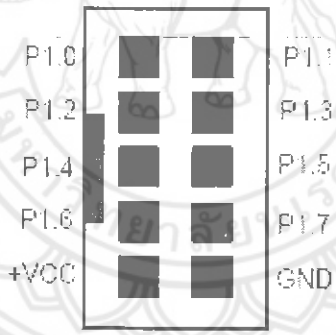
2.2.3) ขั้วต่อ I/O Port: สำหรับบอร์ดนี้จะเป็นการต่อขาสัญญาณ I/O Port ของ MCU เข้ากับขั้วต่อConnector 10 Pinเพื่อให้ผู้ใช้สามารถนำไปต่อใช้งานได้สะดวกมากขึ้นโดยมีการจัดขาสัญญาณดังนี้

- **Port-P0[0..7] :** ขาสัญญาณเหล่านี้ สามารถใช้งานเป็น Input หรือ Output ได้ โดยถูกจัดไว้ที่ขั้วต่อ Connectorขนาด 10 Pin โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก P0 ของ MCU ทั้ง 8 เส้น ลักษณะการจัดขาสัญญาณแสดงดังรูปที่ 2.1



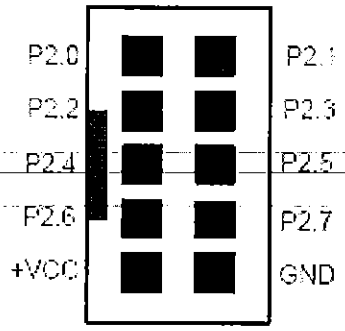
รูปที่ 2.1 แสดงการจัดเรียงขาขั้วต่อ 10 Pin ของ Port P0

- **Port-P1[0..7] :** ขาสัญญาณเหล่านี้ สามารถใช้งานเป็น Input หรือ Output ได้ โดยถูกจัดไว้ที่ขั้วต่อ Connectorขนาด 10 Pin โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก P1 ของ MCU ทั้ง 8 เส้น ลักษณะการจัดขาสัญญาณแสดงดังรูปที่ 2.2



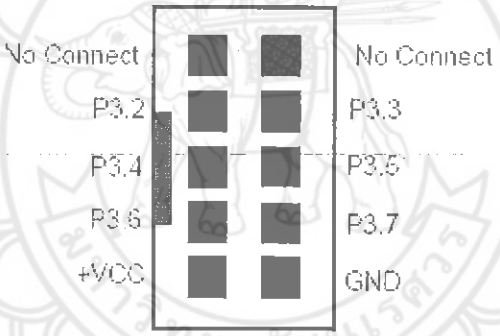
รูปที่ 2.2 แสดงการจัดเรียงขาขั้วต่อ 10 Pin ของ Port P1

- **Port-P2[0..7] :** ขาสัญญาณเหล่านี้ สามารถใช้งานเป็น Input หรือ Output ได้ โดยถูกจัดไว้ที่ขั้วต่อ Connectorขนาด 10 Pin โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก P2 ของ MCU ทั้ง 8 เส้น ลักษณะการจัดขาสัญญาณแสดงดังรูปที่ 2.3



รูปที่ 2.3 แสดงการจัดเรียงขาขั้วต่อ 10 Pin ของ Port P2

Port-P3[2..7] : ขาสัญญาณเหล่านี้ สามารถใช้งานเป็น Input หรือ Output ได้ โดยถูกจัดไว้ที่ขั้วต่อ Connectorขนาด 10 Pin โดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก P3 ของ MCU อยู่ 6 เส้น ส่วนอีก 2 เส้นที่เหลือคือ P3.0 และP3.1นั้นจะถูกต่อไปยัง ขั้วตัว RS232 #1 ลักษณะการจัดขาสัญญาณแสดงดังรูปที่ 2.4



รูปที่ 2.4 แสดงการจัดเรียงขาขั้วต่อ 10 Pin ของ Port P3

***หมายเหตุ** เมื่อจะใช้งานขาสัญญาณใดๆหรือ Port ใดเป็น Input ผู้ใช้จะต้องทำการส่งค่า 0xFF (ทำขาสัญญาณที่จะใช้เป็น Input ให้เป็น 1) ออกไปให้ Pin หรือ Port นั้นก่อนแล้วถึงทำการอ่านข้อมูลเข้ามาได้ แต่ถ้าใช้เป็น Output สามารถส่งข้อมูลออกไปได้เลย

2.2.4) ขั้วต่อ RS232/RS422/RS485: สำหรับบอร์ดนี้ได้จัดสรรขั้วต่อสำหรับการสื่อสารแบบอนุกรมไว้ 3 รูปแบบ โดยรายละเอียดและหน้าที่การใช้งานของแต่ละแบบนั้นจะเป็นดังนี้

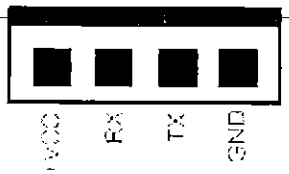
- RS232#1 : ขั้วต่อนี้จะถูกต่อไว้ที่ขั้ว Connector 4 Pin โดยจะทำการเชื่อมต่อสัญญาณมาจาก P3.0(RxD_0) และ P3.1(TxD_0) ซึ่งขั้วต่อนี้นอกจะใช้ในการสื่อสารทาง RS232 แบบปกติแล้ว ยังใช้สำหรับ Download โปรแกรมลงใน MCU ด้วย โดยไม่ต้องทำการ Set Jumper ใด ๆ เพียงแต่เวลาจะ Download โปรแกรม จะต้องทำการกดสวิตช์ PSENและ RESET ดังต่อไปนี้ เพื่อเข้าสู่ Monitor Mode

- กด SW. PSEN ค้างไว้
- ตามด้วยการกด SW. RESET ค้างไว้
- ปล่อย SW. RESET ในขณะที่ SW. PSEN ยังถูกกดค้างอยู่
- ปล่อย SW. PSEN เป็นลำดับสุดท้าย

หลังจาก Download เรียบร้อยให้ทำการกด SW. RESET เพื่อให้ MCU เริ่ม Run โปรแกรมที่เขียน ซึ่ง Port RS232#1 ก็จะเข้าสู่การทำงานแบบปกติ คือ การสื่อสารของ Port ก็จะถูกควบคุมตามโปรแกรมที่ผู้ใช้เขียน

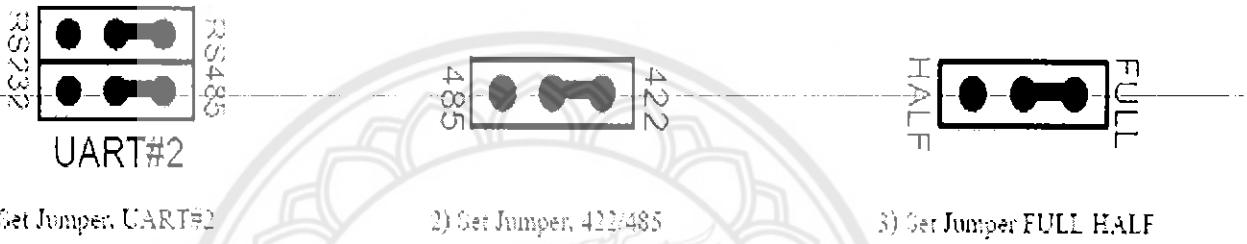
- RS232#2 : สำหรับขั้วต่อนี้ จะใช้สำหรับสื่อสารข้อมูลทาง RS232 เช่นกัน ซึ่งจะแยกอิสระกับขั้วต่อ RS232#1 โดยจะถูกต่อไว้ที่ขั้ว Connector 4 Pin ซึ่งขั้วต่อนี้จะเชื่อมต่อสัญญาณผ่าน Jumper UART#2 ไปยังขาสัญญาณ P6.0(R xD_1) และ P6.1(TxD_1)เนื่องจากขาสัญญาณ P6.0 และ P6.1 นี้จะถูกนำไปใช้งานสำหรับขั้วต่อ RS422/485 ด้วย ดังนั้นเวลาจะใช้งานขั้วต่อ RS232#2 จะต้อง Set Jumper UART#2 มาทางด้าน RS232 ดังแสดงในรูปที่ 2.5

รูปที่ 2.5 แสดงการ Set Jumper UART#2 เมื่อใช้งานขั้วต่อ RS232#2

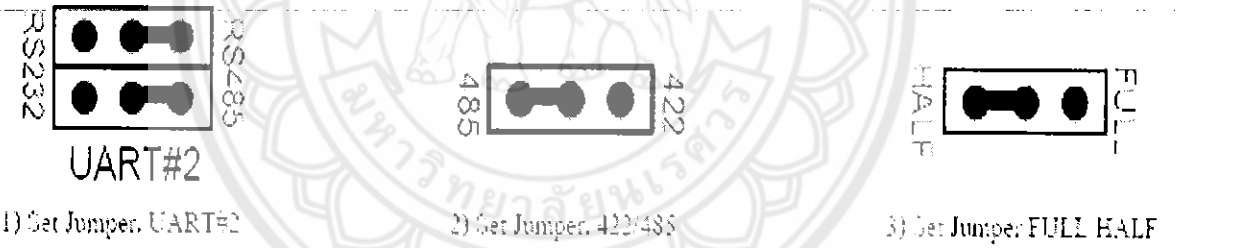


รูปที่ 2.6 แสดงการจัดเรียงขาขั้วต่อ RS232#1 และ RS232#2

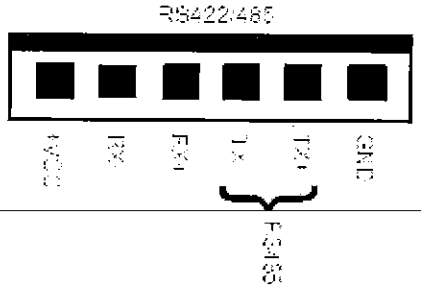
- **RS422/485 :** ขั้วต่อนี้จะใช้สื่อสารข้อมูลแบบ RS422 หรือ RS485 อย่างใดอย่างหนึ่ง ซึ่งการใช้งานขั้วต่อนี้จะต้องทำการต่อ IC Line Driver 75176 จำนวน 2 ตัว สำหรับ RS 422 หรือ 1 ตัวสำหรับ RS485 ลงใน Socket เสียก่อน จากนั้นก็ทำการ Set Jumper UART#2 มาทางด้าน RS485 ดังรูปที่ 2.7 ต่อมาก็ทำการ Set Jumper 422/485 ไปทางด้านที่จะใช้งาน ซึ่งเมื่อเลือกมาทางด้าน 422 จะต้อง Set Jumper FULL/HALF ไปทางด้าน FULL เพื่อใช้ IC Line Driver 2ตัวในการรับข้อมูล(ICตัวใน) และส่งข้อมูล(ICตัวนอก) แต่ถ้าเลือกมาทางด้าน 485 ให้ Set Jumper FULL/HALF มาทางด้าน HALF เพื่อใช้ IC Line Driver ตัวนอกตัวเดียวในการรับและส่งข้อมูล (แบบ 2 Line) โดยใช้ขา P3.7 เป็นตัวควบคุมการรับ-ส่งข้อมูล (P3.7 = 1 : Tx , P3.7 = 0 : Rx)



รูปที่ 2.7 แสดงการ Set Jumper เมื่อใช้งานขั้วต่อ RS422



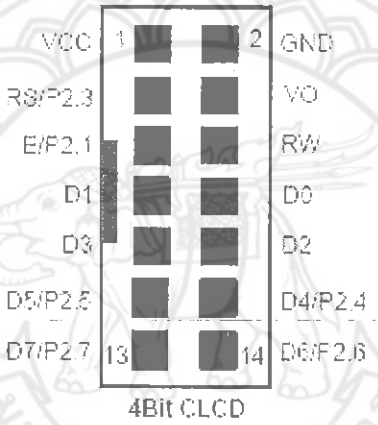
รูปที่ 2.8 แสดงการ Set Jumper เมื่อใช้งานขั้วต่อ RS485



รูปที่ 2.9 แสดงการจัดเรียงขาของขั้วต่อ RS422/485

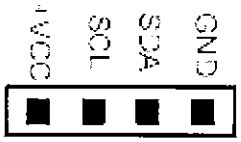
***หมายเหตุ** การใช้งาน ขั้วต่อ RS232#2 หรือ RS422/485 จะต้องเลือกใช้งานขั้วต่อใด ขั้วต่อหนึ่งโดยการ Set Jumper UART#2 ไม่สามารถจะใช้งานขั้วต่อทั้งสองนี้พร้อมกันได้ เนื่องจากได้ใช้ขาสัญญาณ RxD_1(P6.0) และ TxD_1 (P6.1) ของ MCU ร่วมกันอยู่ และในกรณีเลือกใช้งาน RS422/485 จะต้องใส่ IC Line Driver #75176 ใน Socket ทั้ง 2 ตัวด้วย สำหรับ RS422 หรือ 1 ตัวทางริมด้านนอกสำหรับRS485

2.2.5) ขั้วต่อ CLCD: ขั้วต่อนี้จะใช้สำหรับต่อ DOT Matrix LCD โดยถูกจัดไว้ตามขั้วต่อ Connector ขนาด 14 Pinโดยขั้วต่อนี้จะเชื่อมต่อสัญญาณมาจาก Port P2 ของ MCU การจัดวงจรของ Port CLCD นี้จะต่อในลักษณะแบบ 4 บิตมี VR ต่อไว้สำหรับปรับความเข้มของ LCD ให้ด้วย เมื่อผู้ใช้จะใช้งานจะต้องต่อ PIN ของ Modul LCD ให้ตรงกับขาที่กำหนดไว้บน Port ด้วย โดยมีการจัดเรียงขาดังนี้



รูปที่ 2.10 แสดงการจัดเรียงขาของขั้วต่อ CLCD แบบ 4 บิต

2.2.6) Socket DS1307 : สำหรับ Socket 1307 ที่จัดไว้บนบอร์ดนี้ จะเป็นการจัดวงจรไว้สำหรับรองรับการต่อใช้งาน RTC #DS1307 ซึ่งจะเป็นการสื่อสารแบบ I2C เมื่อผู้ใช้จะใช้งานจะต้องนำ IC RTC #DS1307 นี้มาเสียบที่ Socketจากนั้นก็ทำการต่อสาย SDA และ SCL จาก Connector 4 Pin ที่อยู่ข้างๆ ซึ่งได้เชื่อมต่อมาจาก ขา SDA และ SCL ของDS1307 ไว้แล้วไปเข้าขา I/O ของ MCU ที่จะใช้ควบคุม ซึ่ง MCU เบอร์นี้ไม่มีขา I2C ให้ใช้โดยตรงดังนั้นจะต้องสร้างขา SDA และ SCL จาก I/O เอาเอง



รูปที่ 2.11 แสดงการจัดเรียงขาของขั้วต่อ #DS1307 4 PIN

2.3 ET-GSM SIM300CZ

เป็นโมดูลสื่อสารระบบ GSM/GPRS ขนาดเล็ก รองรับระบบการสื่อสาร GSM ความถี่ 900/1800/1900MHz โดยสั่งงานผ่านทางพอร์ตสื่อสารอนุกรม RS232 ด้วยชุดคำสั่ง AT Command สามารถประยุกต์ใช้งานได้มากมายหลายรูปแบบ ไม่ว่าจะเป็นการรับส่งสัญญาณแบบ Voice, SMS, Data, FAX และยังรวมถึงการสื่อสารด้วย Protocol TCP/IP ด้วย

2.3.1คุณสมบัติของบอร์ด ET-GSM SIM300CZ V1.0

• มีสวิตช์แบบ Push-Button สำหรับใช้สั่ง เปิด-ปิด การทำงานของโมดูลภายในบอร์ด

• มี Socket SIM รองรับ SIM Card พร้อมวงจร ESD ป้องกัน SIM เสียหาย

• มีวงจร Regulate แยกอิสระ จำนวน 2 ชุดสามารถใช้กับแหล่งจ่ายภายAdapter ขนาดตั้งแต่ +5V ขึ้นไปสามารถจ่ายกระแสให้กับโมดูล SIM300CZ และอุปกรณ์เชื่อมต่อต่างๆได้อย่างเพียงพอ

•มีวงจรRegulateขนาด 4.2V / 3A สำหรับจ่ายให้กับโมดูล SIM300CZ ได้อย่างเพียงพอ สามารถใช้กับ SIM ของระบบ GSM900MHz แบบ 2-Watt ได้อย่างไม่เกิดปัญหา

• มีวงจร Regulate ขนาด 3.3V / 1A เพื่อจ่ายให้กับวงจรเชื่อมต่อภายนอกโดยไม่ต้องไป ดึงไฟจากตัวโมดูลมาใช้ เพื่อป้องกันปัญหาโมดูลเสียหายจากวงจรภายนอกดึงกระแสเกินพิกัด และสะดวกต่อการออกแบบวงจรเชื่อมต่อเพิ่มเติม ไม่ต้องกังวลว่ากระแสจะไม่พอจ่ายให้กับอุปกรณ์

• มีวงจร Line Driver เพื่อแปลงระดับสัญญาณโลจิกจากโมดูล SIM300CZ ให้เป็น RS232 ระดับมาตรฐานครบทุกเส้นสัญญาณ ทั้งพอร์ตที่ใช้ในการสื่อสารสำหรับสั่งงานโมดูล และ พอร์ตสำหรับใช้ในการพัฒนาโปรแกรม (Debug) สามารถเชื่อมต่อกับพอร์ต RS232 มาตรฐานได้ทันที

• มีLEDแสดงสถานะพร้อมในบอร์ดสำหรับแสดงสถานะของแหล่งจ่ายไฟ สถานะพร้อมทำงานของโมดูล สถานะในการเชื่อมต่อกับNetwork และ สถานะPower-On/Power-OFF ของโมดูล

• มีขั้วสำหรับเชื่อมต่อกับ Handset (ชุดปากพูด และหูฟัง ของโทรศัพท์บ้าน) โดยใช้ขั้วต่อแบบ RJ11 มาตรฐาน พร้อมวงจร Voice Filter สามารถนำชุด Handset ของโทรศัพท์บ้าน ต่อเข้ากับบอร์ดทางขั้วต่อแบบ RJ11 สำหรับใช้พูดคุย โทรออก และ รับสายได้โดยสะดวก

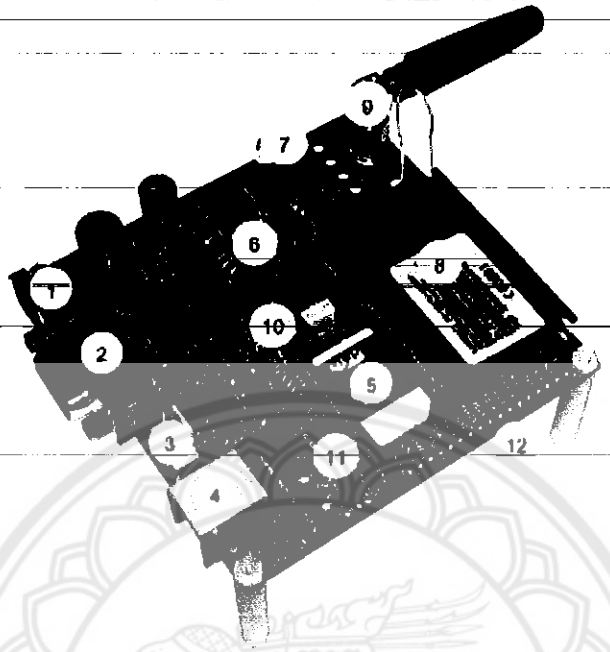
• มี Buzzer พร้อมวงจรขับเพื่อสร้างสัญญาณเสียง ในกรณีมีการโทรเรียกเข้ามายังโมดูล

• มีจุดยึดเสาอากาศ เพื่อใช้เป็นจุดพักสำหรับเชื่อมต่อกับเสาอากาศแบบต่างๆได้โดยสะดวก

• มีขั้วต่อสำหรับติดตั้งโมดูล SIM300CZ พร้อมเสารองและสกรูยึดโมดูลกับตัวบอร์ด

• มีจุดต่อสัญญาณอื่นๆที่เหลือจากโมดูล เช่น Keyboard, Display ,GPIO ,Battery Charger ฯลฯ สำหรับให้ผู้ใช้ต่อขยายไปยังวงจรที่ออกแบบเพิ่มเติมได้โดยง่ายและสะดวก

2.3.2 โครงสร้างของบอร์ด ET-GSM SIM300CZ V1.0



• **หมายเลข 1** เป็น JACK DC-IN แบบมีขั้วโดยมีด้านนอกเป็นขั้วบวก และด้านในเป็น GNDใช้สำหรับรับแหล่งจ่ายไฟจากภายนอกโดยออกแบบให้ใช้กับแหล่งจ่ายไฟซึ่งมีขนาด 5V ขึ้นไปจ่ายกระแสได้ 1A ถึง 3A

• **หมายเลข 2** เป็นขั้วต่อ RS232 (DCE) แบบ DB9 ตัวเมียจะใช้สำหรับเชื่อมต่อเข้ากับสัญญาณ RS232 (DTE) แบบ DB9 ตัวผู้จากคอมพิวเตอร์ PC หรืออุปกรณ์ภายนอกอื่นๆ โดยใช้สาย 9 Pin แบบต่อตรง

• **หมายเลข 3** เป็นขั้วต่อ DEBUG ใช้สำหรับพัฒนา และ DEBUG โปรแกรม สำหรับต่อกับ RS232 ในกรณีที่ต้องการพัฒนาโปรแกรมเพิ่มเติมให้กับโมดูล SIM300CZ เอง

• **หมายเลข 4** เป็นขั้วต่อ RJ11 สำหรับใช้เชื่อมต่อกับชุด Handset ในกรณีที่ต้องการใช้งานโมดูล SIM300CZ เพื่อโทรออกและรับสาย โดยสามารถเชื่อมต่อกับ Handset มาตรฐานได้ทั่วไป

• **หมายเลข 5** เป็น Socket สำหรับติดตั้ง SIM Card ให้กับโมดูล

• **หมายเลข 6** เป็น Switch Push-Button สำหรับใช้ Power-On และ Power-OFF ตัวโมดูล

• **หมายเลข 7** เป็น Buzzer สำหรับสร้างเสียงเรียกเข้าในกรณีที่มีการโทรเข้ามายังโมดูล SIM300CZ

• **หมายเลข 8** เป็น จุดรองรับโมดูล SIM300CZ พร้อมเสาและสกรูสำหรับยึดโมดูลกับบอร์ด

• **หมายเลข 9** เป็น จุดยึด Connector เสาอากาศ GSM/GPRS ย่านความถี่ 900/1800/1900MHz

• **หมายเลข 10** เป็น LED แสดงแหล่งจ่าย VBAT โดยจะติดสว่างเมื่อมีการจ่ายไฟให้บอร์ดแล้ว

• **หมายเลข 11** เป็น LED แสดงสถานะของบอร์ด ซึ่งมีด้วยกัน 3 ดวงคือ

- POWER สีแดง จะติดสว่าง เมื่อโมดูลอยู่ในสถานะ Power-ON
- NETLIGHT สีเหลือง จะกระพริบเมื่อโมดูลอยู่ในสถานะ Power-ON
- STATUS สีเขียว จะติดสว่างเมื่อโมดูลอยู่ในสถานะ Power-ON

• **หมายเลข 12** เป็น จุดต่อสัญญาณเพิ่มเติมในกรณีที่ต้องการประยุกต์ใช้งานโมดูลเพิ่มเติม

2.3.3 คุณสมบัติของโมดูล SIM300CZ

- รองรับความถี่ GSM/GPRS 900/1800/1900MHz
- รองรับ GPRS Multi-Slot Class10 และ GPRS Mobile Station Class B
- รองรับมาตรฐานคำสั่ง AT Command

(GSM 07.07 / 07.05 และคำสั่งเพิ่มเติมจาก SIMCOM)

- รองรับ SIM Applications Toolkit
- ทำงานที่ย่านแรงดัน 3.4V ถึง 4.5V
- รองรับการเชื่อมต่อภายนอก
- ใช้ได้กับ SIM 3V และ 1.8V
- มีวงจร Analog Audio (MIC & Speaker) จำนวน 2 ชุด
- รองรับ 5x5 Keypad Interface & SPI LCD Interface
- มีระบบ RTC พร้อมวงจร Backup
- มีขั้วต่อเสาอากาศภายนอกแบบ Connector และจุดเชื่อมต่อแบบ PAD
- มีระบบ Battery Charge ในตัว

2.3.4 อุปกรณ์แสดงการทำงานของโมดูล SIM300CZ

สำหรับบอร์ด ET-GSM SIM300CZ V1.0 นั้น ได้ออกแบบอุปกรณ์แสดงผลการทำงานของ บอร์ดไว้ในบอร์ดเพื่อใช้แสดงสถานะของการทำงานต่างๆ ให้ผู้ใช้ทราบด้วย คือ

• **Buzzer** ใช้แสดงการทำงานของโมดูลเมื่อมีสายเรียกเข้า โดยการทำงานของ Buzzer นี้จะถูกควบคุมด้วยสัญญาณ BUZZER (Pin23) ของโมดูล SIM300CZ และสามารถปรับระดับความดังของเสียงได้จากคำสั่ง “AT+CRSL” ได้อีกด้วย

• **LED VBAT** ใช้ทำหน้าที่แสดงสถานะของแหล่งจ่ายไฟจากภายนอกที่ต่อมาให้กับบอร์ด โดย LEDนี้จะติดสว่างก็ต่อเมื่อมีการจ่ายไฟให้กับบอร์ดเป็นที่เรียบร้อยแล้ว

• **LED POWER** ใช้แสดงสถานะความพร้อมของโมดูล SIM300CZ ว่าอยู่ในสภานะ Power ON หรือ Power OFF โดย LED ตัวนี้จะถูกควบคุมการทำงานด้วยสัญญาณ VDD_EXT(Pin15) ของโมดูลเมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก “1” โดยถ้า LED Power ติดสว่าง แสดงว่า โมดูล SIM300CZ อยู่ในสถานะ Power ON และพร้อมทำงาน แต่ถ้า LED นี้ดับ แสดงว่าโมดูล อยู่ในสถานะ Power OFF อยู่

• **LED NETLIGHT** ใช้แสดงสถานะของโมดูล ในขณะทำการเชื่อมต่อกับเครือข่ายอยู่ โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณ NETLIGHT (Pin16) ของโมดูล SIM300CZ เมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก "1" โดยเมื่อโมดูลอยู่ในสถานะพร้อมทำงาน LED จะติดกระพริบด้วยค่าความเร็วต่างๆ ซึ่งมีความหมายดังนี้

- OFF แสดงว่าโมดูลอยู่ในสถานะของ Power OFF (ไม่ทำงาน)

- 64mS ON / 800mS OFF แสดงว่า โมดูล SIM300CZ ทำงานปรกติ และไม่ได้อยู่ระหว่างทำการค้นหาเครือข่ายอยู่

- 64mS ON / 3000mS OFF แสดงว่า โมดูล SIM300CZ กำลังทำการค้นหาเครือข่ายเพื่อทำการเชื่อมต่อสัญญาณ

- 64mS ON / 300mS OFF แสดงว่า โมดูล SIM300CZ อยู่ระหว่างการเชื่อมต่อกับเครือข่ายหรืออุปกรณ์อื่นๆด้วย GPRS อยู่

• **LED STATUS** ใช้แสดงสถานะของโมดูล SIM300CZ ว่าพร้อมทำงานหรือไม่ โดย LED ตัวนี้จะถูกควบคุมด้วยสัญญาณ STATUS (Pin19) ของโมดูล SIM300CZ เมื่อทำงานจะมีสถานะทางโลจิกเป็นโลจิก "1" ซึ่งเมื่อ LED นี้ติดสว่าง แสดงว่าโมดูลพร้อมรับคำสั่งต่างๆได้ แต่ถ้า LED ดับ แสดงว่าโมดูลยังไม่พร้อมทำงาน

2.3.5 การสั่ง เปิด และ ปิด การทำงานของโมดูล

ตามปรกติแล้วโมดูล SIM300CZ จะมีโหมดการทำงานอยู่หลายโหมด เราสามารถทำงานสั่งเปิดและ ปิดการทำงานของโมดูลได้ หลายวิธี

• Switch ON/OFF เป็นการสั่ง เปิด และ ปิด การทำงานของโมดูล SIM300CZ ด้วยการกดสวิตช์โดยสวิตช์ตัวนี้ จะเป็นแบบ Push-Button Switch (สวิตช์ กดติด - ปล่อยดับ) โดยเป็นการกำหนดสถานะทางโลจิกให้กับขาสัญญาณ PWRKEY(Pin17) ของโมดูลโดยเมื่อกดสวิตช์จะเป็นโลจิก "0"เมื่อปล่อยสวิตช์จะเป็นโลจิก "1" โดยการทำงานของสวิตช์จะต้องทำการกดสวิตช์ต่อเนื่องกันเป็นเวลานานอย่างน้อย 2000mS (2 วินาที) จึงจะมีผลต่อการทำงานของโมดูล โดยลักษณะการทำงานของสวิตช์ จะเป็นแบบ Toggle กล่าวคือ ถ้าโมดูลอยู่ในสถานะของ Power OFF อยู่ แล้วทำการกดสวิตช์ เป็นเวลาอย่างน้อย 2000mS (2 วินาที) จะเป็นการสั่งให้โมดูลกลับเข้าสู่ Power On หรือพร้อมทำงาน แต่ถ้าหากว่าโมดูลอยู่ในสถานะของ Power ON อยู่แล้วทำการกดสวิตช์ เป็นเวลาอย่างน้อย 2000mS (2 วินาที) แล้วปล่อยจะเป็นการสั่งให้โมดูลหยุดทำงานและกลับเข้าสู่สถานะของ Power OFF (หยุดทำงาน)

ตารางที่ 2.3.5 แสดงสถานะของ LED ในโหมดต่างๆ

LED ต่างๆ	Power-ON	Power-OFF
VBAT (แดง)	ติดสว่าง	ติดสว่าง
POWER (แดง)	ติดสว่าง	ดับ
NETLIGHT (เหลือง)	กระพริบ	ดับ
STATUS (เขียว)	ติดสว่าง	ดับ

หลังจากทำการสั่งPower-ON ในครั้งแรกนั้นก่อนที่จะเริ่มต้นส่งคำสั่งใดๆให้กับโมดูลควรจะรอให้ตัวโมดูลพร้อมเสียก่อน โดยจะมีข้อความ "Call Ready" ปรากฏให้เห็น ในกรณีที่กำหนด Baud rate เป็นแบบAuto Baud rate ไว้ (AT+IPR=0") เมื่อทำการ Power-ON จะได้ผลดังตัวอย่าง

Call Ready

ในกรณีที่กำหนด Baudrate เป็นแบบ Fix Baudrate ไว้(AT+IPR=ค่า Baudrate) เมื่อทำการสั่งให้โมดูPower-ON แต่ละครั้งจะได้ผลดังตัวอย่าง

RDY+

CFUN: 1

+CPIN: READY

Call Ready

2.3.6 การติดต่อสื่อสารกับโมดูล SIM300CZ

การติดต่อสื่อสารกับ โมดูล SIM300CZ ของบอร์ด ET-GSM SIM300CZ นั้น จะเชื่อมต่อผ่านพอร์ตสื่อสารอนุกรม RS232 โดยใช้ขั้วต่อแบบ DB9 ตัวเมีย จัดเรียงสัญญาณตามมาตรฐาน RS232-DCEสามารถนำไปเชื่อมต่อกับสัญญาณ RS232-DTE มาตรฐาน โดยใช้สาย DB9 แบบต่อตรง ได้ทันที โดยสัญญาณทั้งหมดที่ DB9 นี้ได้ผ่านวงจร Line Driver เพื่อแปลงสัญญาณระดับโลจิก จากโมดูล ให้เป็นสัญญาณระดับมาตรฐาน RS232 เป็นที่เรียบร้อยแล้ว ซึ่งถ้าต้องการนำไปเชื่อมต่อกับ RS232(Com Port)ของคอมพิวเตอร์ PC ก็สามารถทำการเชื่อมต่อกันโดยตรงได้ทันที โดยไม่ต้องทำการสลับสายสัญญาณใดๆทั้งสิ้น โดยสัญญาณเชื่อมต่อทางด้านโมดูล SIM300CZ นั้นจะมีทั้งหมด 8 เส้นสัญญาณ ซึ่งในการเชื่อมต่อใช้งานนั้น จะต่อให้ครบทั้ง 8 เส้น หรือ จะเลือกต่อเพียง 3 เส้น (RXD,TXD และ GND) ก็ได้เช่นเดียวกัน โดยสามารถกำหนดได้จากการ Setup ค่า Configuration และคำสั่งใช้งาน โดยสัญญาณการเชื่อมต่อ RS232ด้านโมดูล SIM300CZ จะมีดังนี้

- Pin1 เป็นขา DCD (Data Carrier Detect) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS232 แล้ว ซึ่งตามปรกติจะต่อเข้ากับ DCD Input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC

- Pin2 เป็นขา TXD(Transmit Data) ของ โมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ที่ได้ผ่านการแปลงระดับสัญญาณเป็น RS232 แล้ว ซึ่งตามปรกติจะต่อเข้ากับ RXD (Receive Data)ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin3 เป็นขา RXD (Receive Data) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ สามารถรับสัญญาณระดับ RS232 ได้โดยตรง ซึ่งตามปรกติจะต่อเข้ากับ TXD (Transmit Data) จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin4 เป็นขา DTR(Data Terminal Ready) ของ โมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZซึ่งตามปรกติจะต่อเข้ากับ DTR จากอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin5 เป็นสัญญาณ GND ของโมดูล SIM300CZ ต้องต่อเข้ากับ GND ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin6 ตามปรกติแล้วเป็นสัญญาณ DSR (Data Set Ready) แต่ในกรณีของ SIM300CZ จะไม่ได้ต่อใช้งาน แต่อย่างไรก็ตาม ในบอร์ดได้ทำการป้อนสัญญาณย้อนกลับหรือ Loop Back สัญญาณDTR (Data Terminal Ready) ซึ่งเป็น Output ส่งมาจาก Host หรือ คอมพิวเตอร์ PC กลับไปแทนโดยจะถูกต่อไปเข้ากับสัญญาณ DSR Input ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์
- Pin7 เป็นขาสัญญาณ RTS (Request To Send) ของโมดูล SIM300CZ ซึ่งเป็น Input ของ SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ RTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin8 เป็นขาสัญญาณ CTS (Clear To Send) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZ ซึ่งตามปรกติจะต่อเข้ากับ CTS ของอุปกรณ์ด้าน Host หรือคอมพิวเตอร์ PC
- Pin9 เป็นขาสัญญาณ RI(Ring Indicator) ของโมดูล SIM300CZ ซึ่งเป็น Output จาก SIM300CZซึ่งตามปรกติจะต่อเข้ากับ RI ของอุปกรณ์ด้าน Host หรือ คอมพิวเตอร์ PC

ตารางที่ 2.3.6.1 แสดงการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับ คอมพิวเตอร์ PC

DB9 Female(SIM300CZ)		Signal Direction	DB9 Male(Computer PC)	
Pin	Signal		Signal	Pin
1	DCD	→	DCD	1
2	TXD	→	RXD	2
3	RXD	←	TXD	3
4	DTR	←	DTR	4
5	GND	—	GND	5
6	(DSR)	→	DSR	6
7	RTS	←	RTS	7
8	CTS	→	CTS	8
9	RI	→	RI	9

ตารางที่ 2.3.6.2 แสดงการต่อสายสัญญาณระหว่าง ET-GSM SIM300CZ กับ ไมโครคอนโทรลเลอร์

DB9 Female(SIM300CZ)		Signal Direction	ไมโครคอนโทรลเลอร์
Pin	Signal		Signal
2	TXD	→	RXD
3	RXD	←	TXD
5	GND	—	GND

2.4หลักการรับส่ง SMS ของโทรศัพท์มือถือ

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

2.4 .1โหมดของการรับส่งข้อมูล SMS [1]

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

2.4.2 รูปแบบในการส่งข้อมูลในรูป SMS ผ่าน AT Command

มี 2 รูปแบบ คือ Text Mode และ PDU Mode

1. **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

2. **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

2.1 จะต้องทราบรหัสแอสกีแบบเลขฐาน 16 (Hexadecimal) ของแต่ละอักขระ

2.2 แปลงจากรหัสแอสกีแบบเลขฐาน 16 เป็นรหัสแอสกีแบบเลขฐาน 2 (Binary)

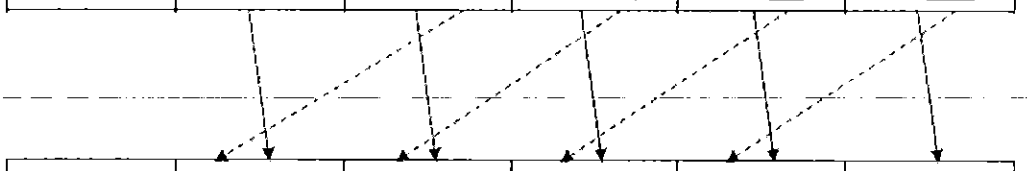
2.3 รหัสแอสกีแบบเลขฐาน 2 มาตัดบิตซ้ายสุดทิ้ง

2.4 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

2.5 แปลงรหัสPDU 8 บิตที่ได้ให้เป็นรหัส PDU แบบเลขฐาน 16 การเข้ารหัส PDUของคำว่า ALERT จะเห็นว่ารหัส PDU ของคำว่าALERT คือ 4166514A05 ดังรูปที่ 2.2

ตารางที่ 2.4 ตัวอย่างการเข้ารหัส PDU ของคำว่า ALERT

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

2.4.3 กลุ่มคำสั่งในการส่ง SMS

- 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

- 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

2.4.4 กลุ่มคำสั่งในการควบคุมและดูสถานะของโทรศัพท์มือถือ

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

- AT+CPBR เป็นคำสั่งที่ใช้อ่านข้อมูลจากสมุดโทรศัพท์ (Read Phone Book Entries)

2.4.5 กลุ่มคำสั่งบริการเครือข่าย

- AT+CLIP เป็นคำสั่งที่ใช้ในการแสดงผลหมายเลขโทรศัพท์ที่โทรเข้า (Calling Line Identification Presentation)

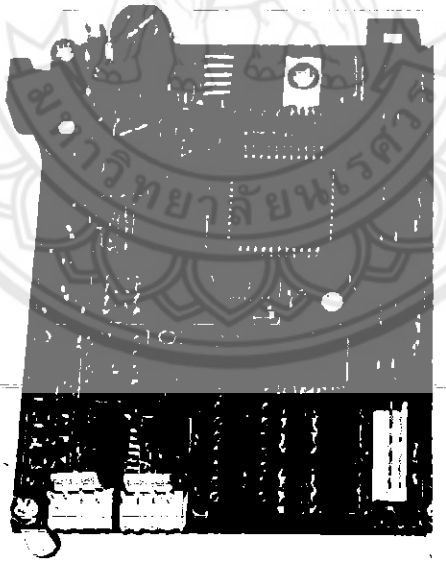
บทที่ 3

วิธีการออกแบบ

ในบทที่ผ่านมาเป็นการศึกษาในเรื่องของที่มาและความสำคัญ รวมถึงหลักการต่างๆ ของการสร้างระบบสัญญาณเตือนภัยผ่านเครือข่ายโทรศัพท์ที่มีการเชื่อมต่อกับ ไมโครคอนโทรลเลอร์ และบอร์ดโทรศัพท์ สำหรับบทนี้จะเป็นการศึกษาถึงการออกแบบและการจัดวางของอุปกรณ์ต่างๆ โดยแบ่งออกเป็น 2 ส่วนด้วยกัน คือ ส่วนของเครื่องควบคุม และส่วนของภาคส่งสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือ ซึ่งรายละเอียดต่างๆ จะแสดงไว้ดังต่อไปนี้

3.1 ส่วนของเครื่องควบคุม

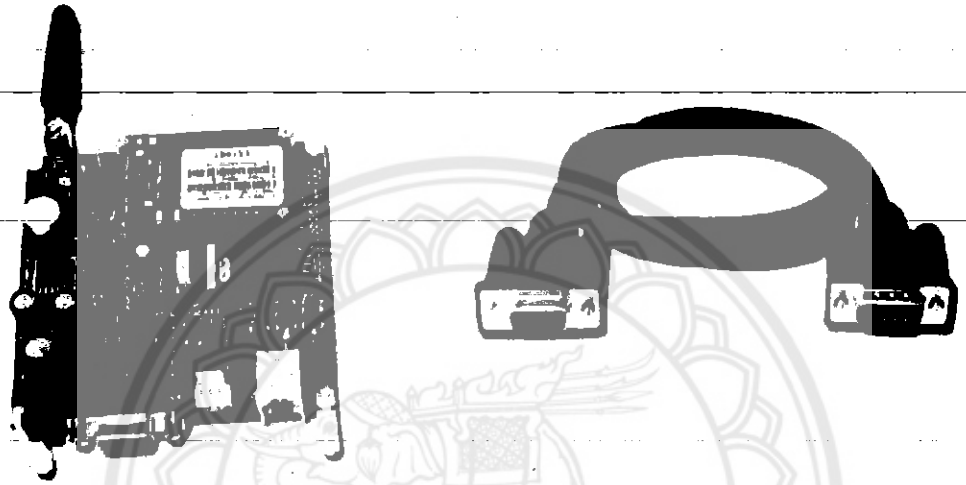
ในส่วนของเครื่องควบคุมจะใช้ ไมโครคอนโทรลเลอร์รุ่น "CP-JR51RE2 V1.0" มีหลักการทำงานดังนี้คือ เมื่อได้รับสัญญาณอินพุทจาก DATA LOGGER ไมโครคอนโทรลเลอร์รุ่น "CP-JR51RE2 V1.0" จะทำการส่งสัญญาณเอาท์พุทออกไปยัง บอร์ดโทรศัพท์ โดยผ่านทาง Port RS232 ที่ ใช้ในการเชื่อมต่อ



รูปที่ 3.1 บอร์ดไมโครคอนโทรลเลอร์รุ่น "CP-JR51RE2 V1.0"

3.2 ส่วนของภาคส่งสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือ

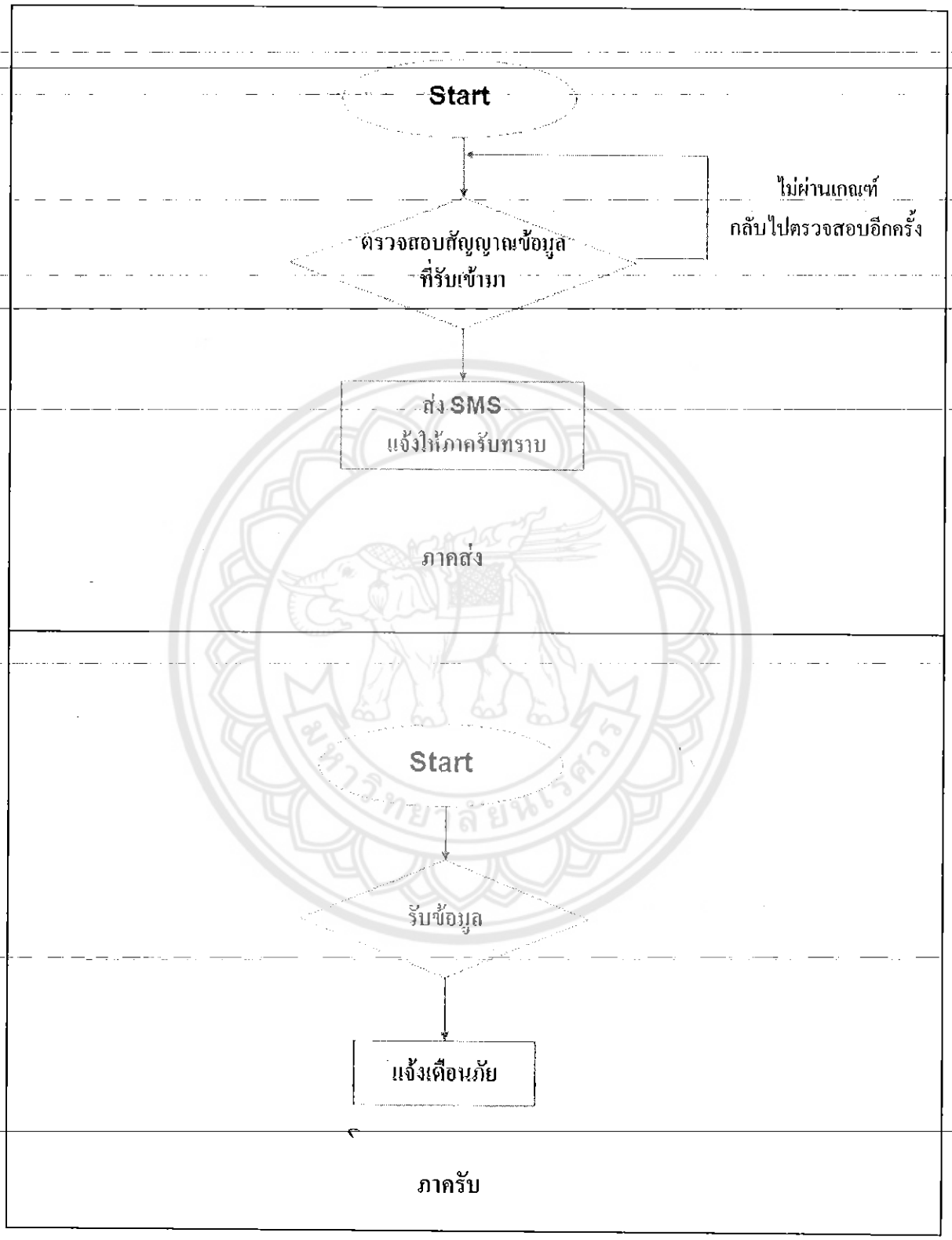
สำหรับในส่วนของภาคส่งสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือนี้ จะใช้บอร์ดโทรศัพท์ ET-GSM SIM300CZ V1.0 ซึ่งออกแบบให้ใช้งานร่วมกับ Controller Board สำหรับงานประยุกต์ที่ต้องควบคุมระบบโดยผ่านการส่งข้อความ ซึ่งมีการตั้งค่าข้อความตามที่เราต้องการให้ส่งเพื่อเตือนภัย



รูปที่ 3.2 บอร์ดโทรศัพท์ ET-GSM SIM300CZ V1.0

ขั้นตอนของการออกแบบและการจัดวางอุปกรณ์ถือว่ามีความสำคัญอย่างยิ่ง เนื่องจากอุปกรณ์แต่ละตัวจะมีหน้าที่และความสำคัญที่แตกต่างกันออกไป และทุกตัวจะต้องทำงานให้มีความสัมพันธ์ซึ่งกันและกัน ทั้งส่วนของเครื่องควบคุม และส่วนของภาคส่งสัญญาณแจ้งเตือนภัยทางโทรศัพท์มือถือ

3.3 Flowchart การทำงานของระบบเตือนภัย



รูปที่ 3.3 Flowchart การทำงานของระบบเตือนภัย

3.4 โหมดการทำงานของระบบเตือนภัย

แบ่งออกเป็น 2 ส่วน ดังนี้

โหมดต้นทาง แบ่งออกเป็น 2 ส่วน ดังนี้

ส่วนแรก คือบอร์ดไมโครคอนโทรลเลอร์ จะทำหน้าที่ในการประมวลผลสัญญาณข้อมูลจาก อินพุทแล้วทำการตัดสินใจว่าข้อมูลที่รับมานั้นมีความปลอดภัยต่ำกว่าเกณฑ์ที่กำหนดไว้หรือไม่ หากต่ำกว่าเกณฑ์ที่กำหนดข้อมูลจะถูกส่งไปยังบอร์ดโทรศัพท์

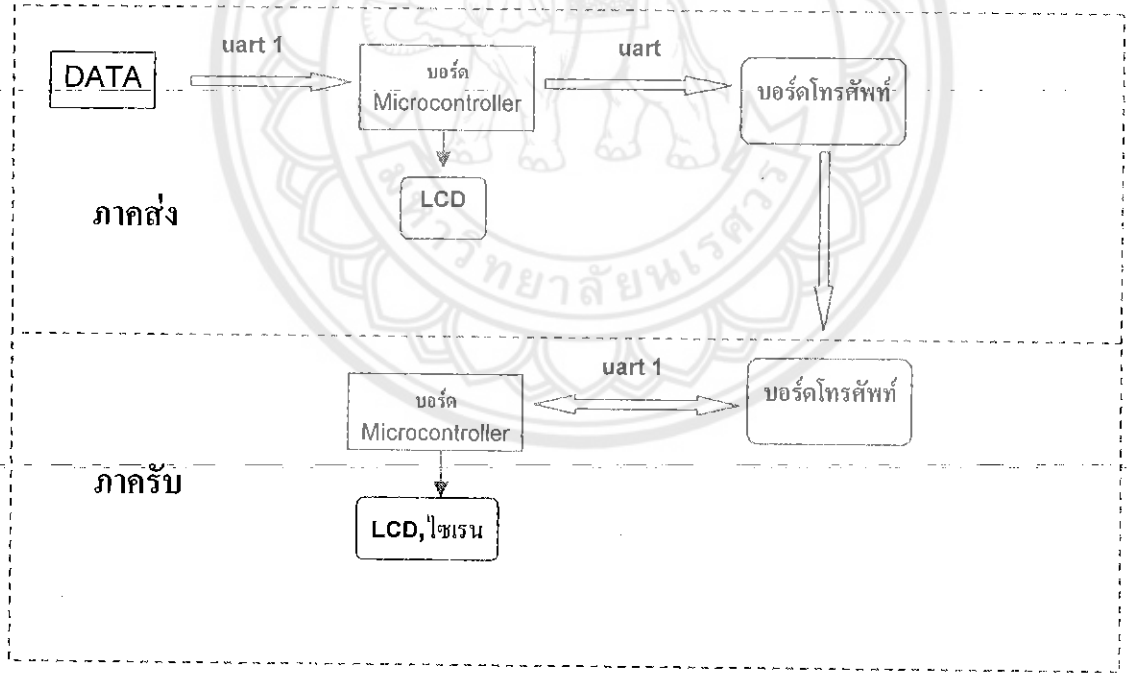
ส่วนที่สอง คือ บอร์ดโทรศัพท์ จะทำหน้าที่ในการแปลงสัญญาณข้อมูลจากบอร์ดไมโครคอนโทรลเลอร์ให้เป็นข้อความ เพื่อที่จะส่งไปยังบอร์ดโทรศัพท์ที่โหมดปลายทาง

โหมดปลายทาง แบ่งออกเป็น 3 ส่วน ดังนี้

ส่วนแรก คือ บอร์ดโทรศัพท์ จะทำหน้าที่ในการรับข้อความ ที่มาจากบอร์ดโทรศัพท์ในโหมดต้นทาง เพื่อทำการแปลงสัญญาณส่งไปยังบอร์ดไมโครคอนโทรลเลอร์ในส่วนที่สอง

ส่วนที่สอง คือ บอร์ดไมโครคอนโทรลเลอร์ จะทำหน้าที่นำข้อมูลที่ได้จากบอร์ดโทรศัพท์เพื่อทำการประมวลข้อมูลและส่งมาเก็บข้อมูล

ส่วนที่สาม คือ LCD ทำหน้าที่แสดงตรวจสอบดูความเคลื่อนไหวของข้อมูลที่ส่งมา



รูปที่ 3.4 โหมดการทำงานของระบบเตือนภัย

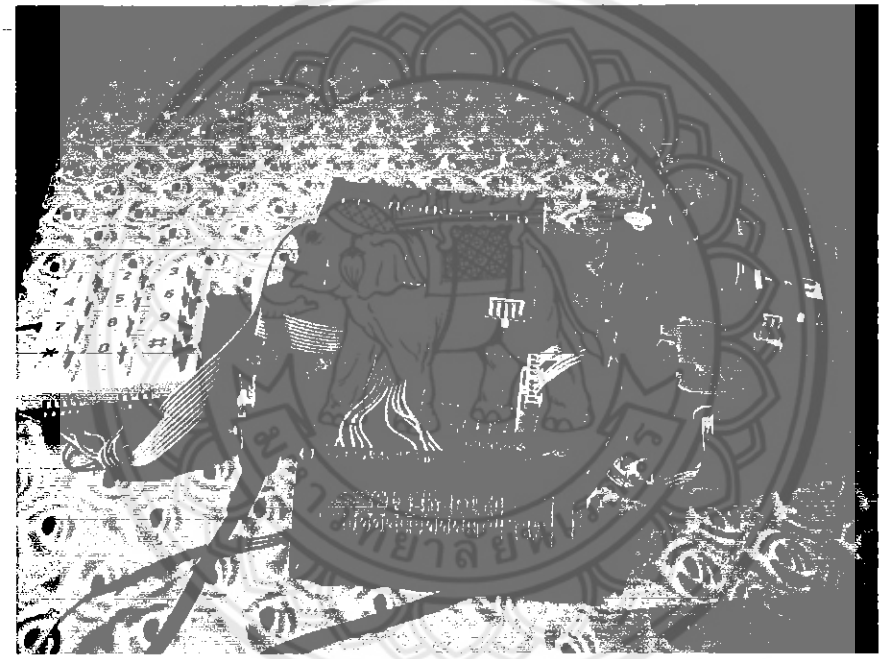
บทที่ 4

ผลการทดลอง

4.1 ระบบการทำงาน

สำหรับระบบการทำงานสามารถแบ่งออกเป็น 2 ส่วนหลักๆ คือ ภาคส่ง ภาครับ

ระบบการทำงานภาคส่ง ประกอบไปด้วย ระบบควบคุมการประมวลผล และส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือไปยังหมายเลขปลายทาง เมื่อต่ออุปกรณ์เข้าด้วยกันแล้ว จากรูปข้างล่างนั้นจะสามารถประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือได้



รูปที่ 4.1 ระบบการทำงานของไมโครคอนโทรลเลอร์ กับ โมดูลโทรศัพท์

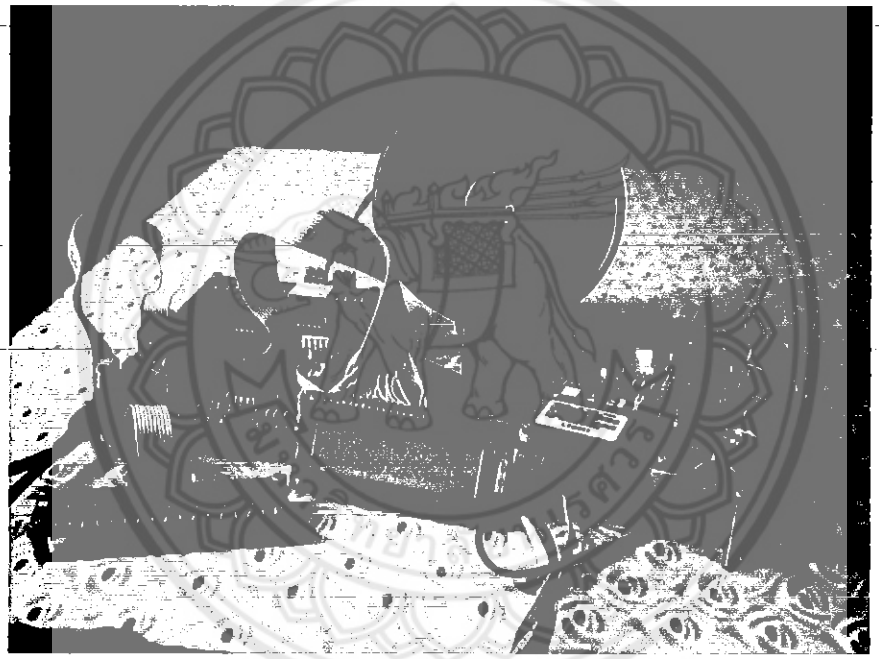
จากรูปที่ 4.1 ประกอบไปด้วยอุปกรณ์ ต่อไปนี้

- ไมโครคอนโทรลเลอร์ ทำหน้าที่ ควบคุมการสั่งงานของโปรแกรม โดยรับอินพุท เข้ามาเก็บไว้และทำการตัดสินใจ ส่งคำสั่งไปยัง โทรศัพท์ มือถือ โดยผ่านพอร์ต RS232 ที่เชื่อมต่อกัน
- โมดูลโทรศัพท์ ทำหน้าที่รับคำสั่ง จากไมโครคอนโทรลเลอร์ และทำการส่งข้อความไปยังหมายเลขปลายทาง
- จอLCD ทำหน้าที่แจ้งสถานะ การรับอินพุท และ การส่งข้อความบนจอ LCD

- EEPROM ทำหน้าที่ โดย EEPROM เป็นหน่วยความจำที่สามารถเขียน - อ่านได้ และสามารถเขียนซ้ำได้หลายๆ ครั้ง และเมื่อ ไม่มีแหล่งจ่ายไฟจ่ายให้กับหน่วยความจำ ข้อมูลก็จะไม่สูญหาย ประโยชน์ที่ได้คือ เก็บค่า อินพุทที่รับเข้ามา และ มีขนาดข้อมูลที่เล็ก

- KEYPAD คือ อุปกรณ์ที่เกิดจากการนำสวิตซ์ หลายๆตัวมาต่อเข้าด้วยกันแบบเมตริกซ์ เสมือนเป็นคีย์บอร์ดขนาดเล็กอันหนึ่ง โดยใช้วิธีการสแกนทำโดยส่งลอจิก 0 ออกทีละแถวจาก R1 ถึง R4 ถ้าไม่มีคีย์ไหนถูกกดคอลัมน์นั้นจะมีค่าเป็นลอจิก 1 แต่ถ้ามีการกดคอลัมน์จะมีค่าเป็น 0

ระบบการทำงานภาครับ ประกอบไปด้วย ระบบควบคุมการรับสัญญาณเตือนภัยทางโทรศัพท์มือถือ จากยังหมายเลย ต้นทาง เมื่อต่ออุปกรณ์เข้าด้วยกันแล้ว จากรูปข้างล่างนั้นจะสามารถอ่านดูข้อความแจ้งเตือนภัยได้



รูปที่ 4.2 ระบบการทำงานของ ไมโครคอนโทรลเลอร์ กับ โมดูลโทรศัพท์

- โมดูลโทรศัพท์จะรับข้อความที่เข้ามาจากหมายเลขต้นทาง และส่งข้อมูลไป ไมโครคอนโทรลเลอร์ โดยผ่านพอร์ต RS232 ที่เชื่อมต่อกัน

- ไมโครคอนโทรลเลอร์ จะนำข้อมูลที่ได้ สั่งให้โปรแกรมอ่านข้อความ แล้วโชว์ที่ LCD เพื่อดูข้อความ และทำการสั่งเตือนภัยด้วยเสียงเป็นเวลา 1 นาที

- จอ LCD ทำหน้าที่แจ้งสถานะ เพื่ออ่านข้อความที่รับเข้ามาและจะโชว์ที่หน้าจอ LCD

4.2 ผลการทดลอง

4.2.1 ผลการทดลองภาคส่ง

เมื่อรับอินพุทเข้ามา 4 อินพุท คือ น้ำฝน น้ำท่า การเคลื่อนตัวของดินและความชื้นของดิน จากนั้นทำการเก็บข้อมูลที่ส่งเข้ามาแล้วตัดสินใจ เมื่อข้อมูลที่รับเข้ามาเกินมาตรฐานที่กำหนดจะมีคำสั่งส่งข้อความเตือนภัยไปยังหมายเลขปลายทาง ทั้งนี้เพื่อความปลอดภัยสูงสุด ทำให้มีหลักเกณฑ์ในการแจ้งเตือนดังนี้

- น้ำท่า เมื่อมีข้อมูลเข้ามา 190 cm ขึ้นไปแจ้งเตือน low
  210 cm ขึ้นไปแจ้งเตือน middle
  230 cm ขึ้นไปแจ้งเตือน High

กรณีที่ 1 น้ำท่าเต็มตลิ่ง 150 cm และเมื่อมีข้อมูลน้ำฝนเข้ามาทำให้น้ำท่าที่เต็มตลิ่งเกิน 150 cm ให้เก็บค่าไว้ แล้วเลื่อนการเตือนของปริมาณน้ำฝนไปหนึ่งขั้นเช่น

- น้ำฝน รับข้อมูลเข้ามาทีละ 5 mm และทำการเก็บข้อมูลไว้โดยที่

ในเวลา 1 ชั่วโมง ถ้า น้ำฝน เกิน	10 mm	ขึ้นไปแจ้งเตือน flood, flood Mid
	20 mm	ขึ้นไปแจ้งเตือน flood, flood High
	30 mm	ขึ้นไปแจ้งเตือน flood, flood High
ในเวลา 1 วัน ถ้า น้ำฝน เกิน	70 mm	ขึ้นไปแจ้งเตือน flood, flood Mid
	85 mm	ขึ้นไปแจ้งเตือน flood, flood High
	100 mm	ขึ้นไปแจ้งเตือน flood, flood High
ในเวลา 3 วัน ถ้า น้ำฝน เกิน	170 mm	ขึ้นไปแจ้งเตือน flood, flood Mid
	185 mm	ขึ้นไปแจ้งเตือน flood, flood High
	200mm	ขึ้นไปแจ้งเตือน flood, flood High

กรณีที่ 2 เมื่อไม่มีข้อมูลน้ำท่าเต็มตลิ่ง 150 cm เข้ามาให้เตือนปริมาณน้ำฝนตามปกติเช่น

- น้ำฝน รับข้อมูลเข้ามาทีละ 5 mm และทำการเก็บข้อมูลไว้โดยที่

ในเวลา 1 ชั่วโมง ถ้า น้ำฝน เกิน	10 mm	ขึ้นไปแจ้งเตือน Flash, flood Low
	20 mm	ขึ้นไปแจ้งเตือน Flash, flood Mid
	30 mm	ขึ้นไปแจ้งเตือน Flash, flood High
ในเวลา 1 วัน ถ้า น้ำฝน เกิน	70 mm	ขึ้นไปแจ้งเตือน Flash, flood Low
	85 mm	ขึ้นไปแจ้งเตือน Flash, flood Mid
	100 mm	ขึ้นไปแจ้งเตือน Flash, flood High

ในเวลา 3 วัน ถ้า น้ำฝน เกิน	170 mm	ขึ้นไปแจ้งเตือน Flash, flood Low
	185 mm	ขึ้นไปแจ้งเตือน Flash, flood Mid
	200 mm	ขึ้นไปแจ้งเตือน Flash, flood High

- การเคลื่อนตัวของดิน เมื่อมีข้อมูลเข้ามามากกว่า 10 cm ขึ้นไปแจ้งเตือน
- ความชื้นของดิน เมื่อมีข้อมูลเข้ามา 50% ขึ้นไปแจ้งเตือน

4.2.2 ผลการทดลองภาครับ

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

แจ้งเตือน Low ไซเรน เสียงดัง 10 วินาที หยุด 50 วินาที

แจ้งเตือน Middle ไซเรน เสียงดัง 20 วินาที หยุด 40 วินาที

แจ้งเตือน High ไซเรน เสียงดัง 40 วินาที หยุด 20 วินาที

และเสียงไซเรนจะหยุดลง เมื่อมีการกดสวิตช์ปุ่มที่ 3

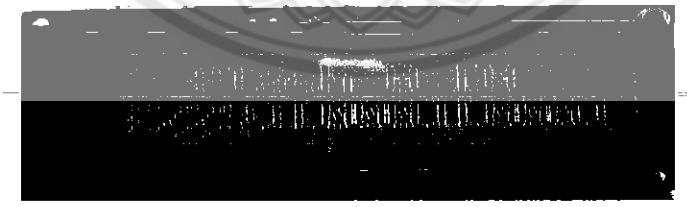
ตัวอย่าง น้ำฝน แสดงเป็น 10:25:00 -18/05/09 Flash ,flood, High 30 mm /1h

กรณีที่ 1 น้ำฝน แสดงเป็น 10:35:00 -18/05/09 Flood ,flood, High 30 mm /1h

น้ำท่า แสดงเป็น 10:45:00 -18/05/09 Flood warning, High 230 cm

ความชื้นของดิน10:55:00 -18/05/09 Soil mois, warning High 50 Per

การเคลื่อนตัวของดิน 11:05:00 -18/05/09 Lanslide warning, High 10 cm



รูปที่ 4.3 แสดงจอ LDC ภาคส่ง

รูปที่ 4.4 แสดงจอ LDC ภาครับ

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการดำเนินงานโครงงาน

จากโครงงานการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือได้ข้อสรุปจากผลการดำเนินงานดังนี้

1. สามารถนำการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือไปประยุกต์ใช้ได้จริงในอนาคต

2. เกิดความรู้ ความเข้าใจในหลักการทำงานของการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือ , การเขียนโปรแกรมภาษา c และการเชื่อมต่อไมโครคอนโทรลเลอร์มากยิ่งขึ้น

3. เกิดความรู้ ความเข้าใจในวิชาพื้นฐานที่มีอยู่เดิม สามารถนำมาใช้และพัฒนาตนเองต่อไป

5.2 ปัญหาและแนวทางการแก้ไข

จากการทำโครงงานการประมวลผลและส่งสัญญาณเตือนภัยทางโทรศัพท์มือถือพบปัญหาและอุปสรรคต่างๆ ดังนี้

1. การดำเนินงานมีความล่าช้า เนื่องจากขาดความรู้ความเข้าใจในการเลือกอุปกรณ์ โดยโทรศัพท์มือถือรุ่นที่มีคุณสมบัติเหมาะสมในการพัฒนาโครงงานหาได้ยาก แก้ไขได้โดยสอบถามผู้ที่เชี่ยวชาญในด้านนี้

2. การดำเนินงานในด้านโปรแกรม บางครั้งเกิดปัญหาข้อผิดพลาดในด้านการพัฒนาโปรแกรมทำให้เกิดความล่าช้าในการดำเนินงาน แก้ไขโดยศึกษาด้วยตนเองและสอบถามผู้ที่เชี่ยวชาญในด้านนี้

5.3 ข้อจำกัดของระบบ

1. ระบบมีระยะเวลาการทำงานจำกัด ขึ้นอยู่กับโปรโมชั่นของโทรศัพท์มือถือ
2. ระบบไม่สามารถส่งข้อความได้ หากไม่มีเครือข่ายบริการของโทรศัพท์มือถือที่ใช้
3. อัตราค่าบริการส่งข้อความมีราคาสูง

5.4 ข้อเสนอแนะในการพัฒนาต่อไป

1. ในอนาคตระบบการสื่อสารไร้สายน่าจะเข้ามามีบทบาทกับชีวิตประจำวันอย่างมาก ดังนั้นการพัฒนา ระบบที่เกี่ยวข้องน่าจะเป็นประโยชน์อย่างมากเช่นกัน

2. ควรเพิ่มอุปกรณ์รับข้อมูลเพิ่มเติม เพื่อให้ระบบยืดหยุ่นและสะดวกต่อการใช้งาน

เอกสารอ้างอิง

(1) คู่มือการใช้งาน ET-GSM SIM300CZ.pdf

(2) คู่มือ_CPJR51RE2_V1.pdf

(3) ประจิน พลังสันติกุล. ชัยวัฒน์ ลิ้มพรจิตรวิไล. MCS-51 microcontroller experiment with Keil C51 compiler. กรุงเทพมหานคร: บริษัท อินโนโวตีฟ เอ็กเพอริเมนต์ จำกัด,2521

(4) อุดม รานอก. ภาษา C สำหรับงานควบคุมไมโครคอนโทรลเลอร์MCS-51. ครั้งที่ 1 .นนทบุรี: ไอดีซีฯ. 2548

(5) นายณรงค์ศักดิ์ เทพช่วย และนายธีรยุทธ หอมจันทร์. **"ระบบแจ้งเตือนภัยผ่านโทรศัพท์มือถือ"** ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อ.เมือง จ.พิษณุโลก 65000

(6) นายโชติชนะ รัตนทิพย์ และนายพิษณุ จันธนะสมบัติ . **"ระบบเตือนภัยในรถยนต์ผ่านโทรศัพท์"**ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น พ.ศ. 2550

(7) นายชายแดน หาญเวชและนายทรงศักดิ์ ร่มจันทร์. **"โทรศัพท์ไร้สายควบคุมอุปกรณ์ไฟฟ้า"** ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น พ.ศ.2546

(8) นายฐาปกิตติ์ ไค่นุ่นน้อย และ นายเอกสรรค์ มลาเซ็ด. **"สถานีวัดปริมาณน้ำฝนทางไกลแบบอัตโนมัติผ่านโทรศัพท์เคลื่อนที่"** ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น พ.ศ.2549

ภาคผนวก ก.

การลงโปรแกรม

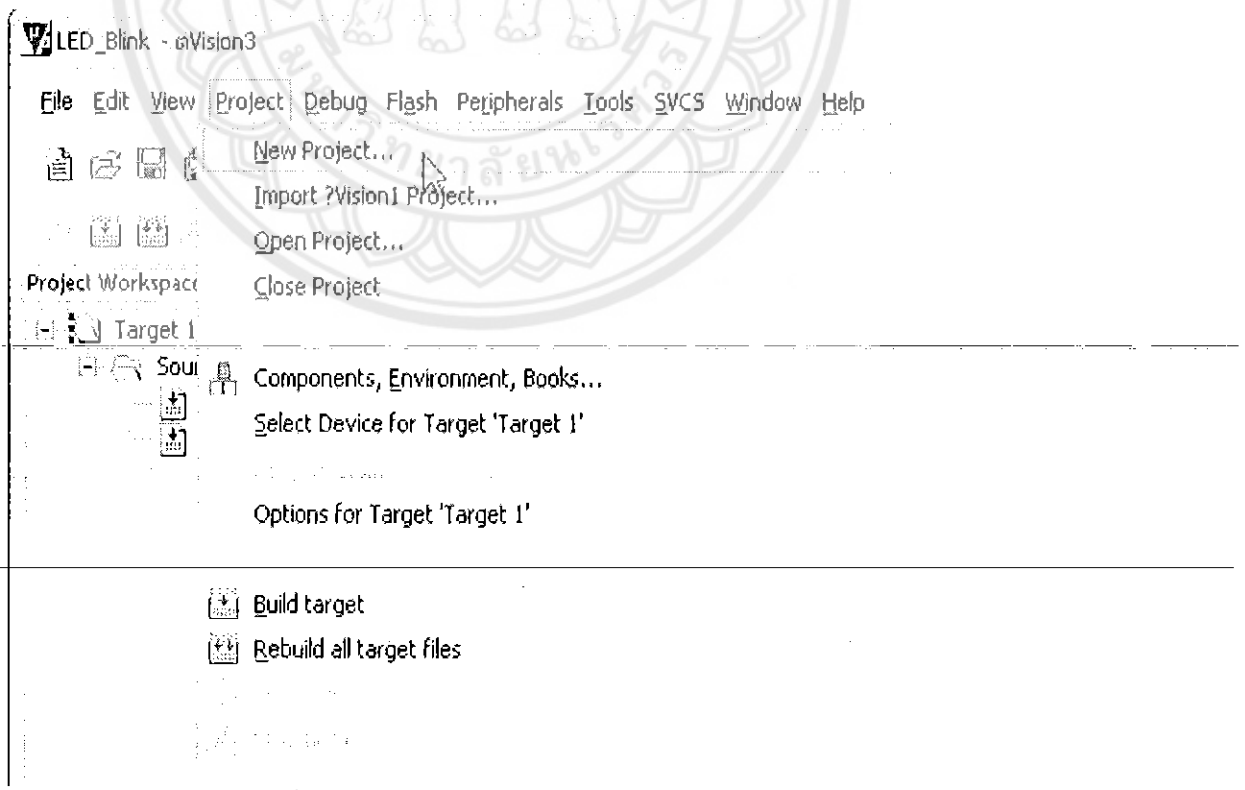
การติดตั้งโปรแกรม Keil Vision3

1) ติดตั้งโปรแกรม Keil Vision3 ลงบนเครื่อง PC เมื่อติดตั้งเสร็จโดยปกติโปรแกรมจะถูกเก็บไว้ที่ C:\Keil

2) ให้ทำการ Copy File: at89c51re2.h ที่ให้มากับแผ่น CD ไปวางไว้ที่ Folder C:\Keil\C51\INC\Atmel เพื่อเอาไว้เรียกใช้ในส่วนของ Include File เมื่อเขียนโปรแกรมจะได้ไม่เสียเวลาประกาศตัวแปรต่างๆอีก

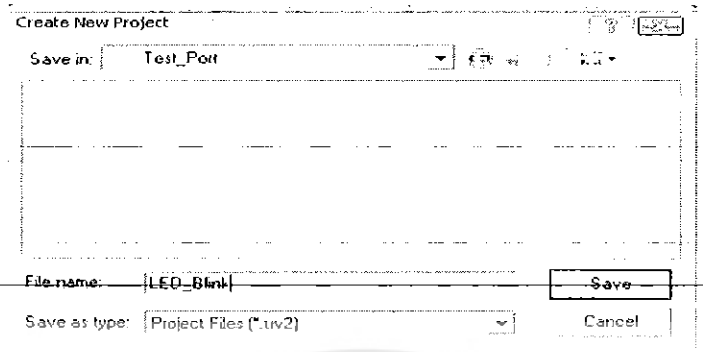
3) Copy File: STARTUP.A51 และ L51_BANK.A51 ที่ให้มากับแผ่น CDไปวางไว้ที่ Folder C:\Keil\C51\LIBซึ่งได้ทำการแก้ไขให้รองรับ MCU เบอร์นี้ไว้แล้ว แต่อาจจะไม่รองรับเบอร์อื่น ดังนั้นก็ควร Copy 2 ไฟล์นี้ ของเดิมที่มาพร้อมกับการติดตั้งเก็บไว้ด้วย เวลาจะเปลี่ยนไปใช้ MCU เบอร์อื่นจะได้นำมาวางทับได้เลย ไม่ต้องเสียเวลาติดตั้งโปรแกรมใหม่

4) เปิดโปรแกรม Keil □Vision3 ( ) ขึ้นมา เลือกที่เมนู Project และเลือก New Project... ดังรูปที่ 3.1.1



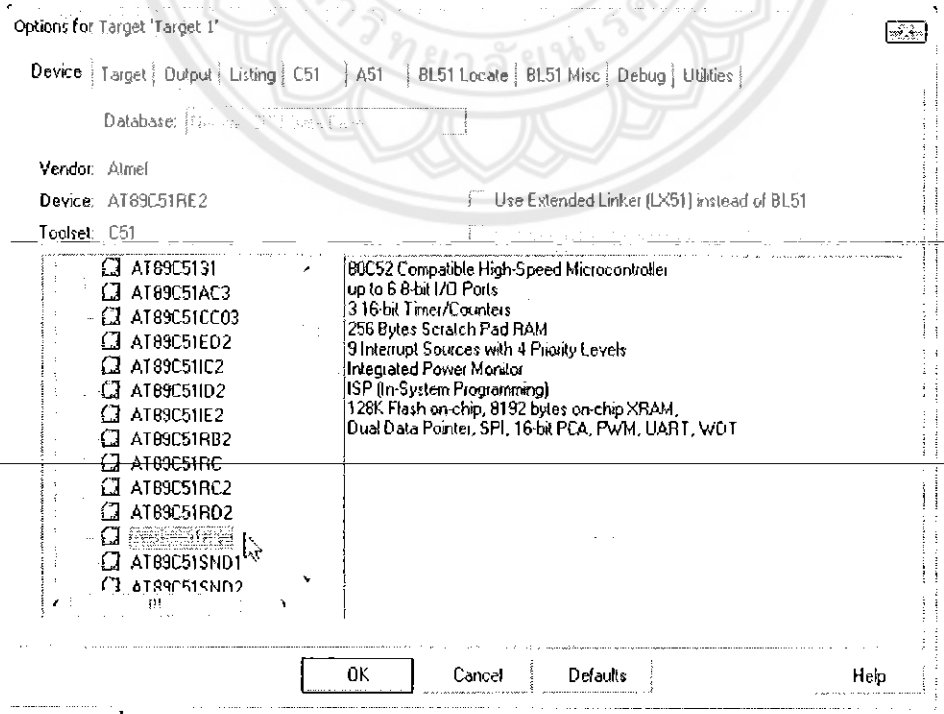
รูปที่ 3.1.1 แสดงการเลือก New Project

5) จะปรากฏหน้าต่างCreate New Projectขึ้นมาดังรูปที่3.1.2ให้เลือก Folderที่จะเก็บProject Fileและตั้งชื่อProject Fileในตัวอย่างจะตั้งชื่อProject fileคือLED_Blinkและเลือกเก็บที่Folder Test Port จากนั้นกดSave

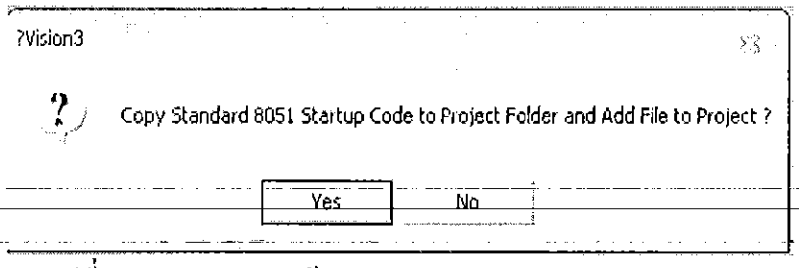


รูปที่ 3.1.2 แสดงหน้าต่าง Create New Project

6) หลังจากกด Save แล้ว จะปรากฏหน้าต่าง Select Device for Target 'Target1' ขึ้นมา ดังรูปที่ 3.1.3 (a) เพื่อให้ผู้ใช้เลือกเบอร์อุปกรณ์ โดยในช่อง Data base ให้ผู้ใช้เลือกที่ Atmel และเลือกที่เบอร์ AT89C51 RE2 ให้สังเกต เมื่อเลือกแล้วเบอร์จะแสดงในช่อง Device: ด้านบน จากนั้นคลิก OK จะมีหน้าต่าง Pop-Up ขึ้นมาดังรูปที่ 3.1.3 (b)ถ้ากด YES จะเป็นการนำ File Start Up (STARTUP.A51) มาตรฐานของ C51 เข้ามายัง Project หรือกด No ก็ได้ถ้าไม่ต้องการนำ File Start Up เข้ามายัง Project ในตัวอย่างนี้จะขอเลือก No

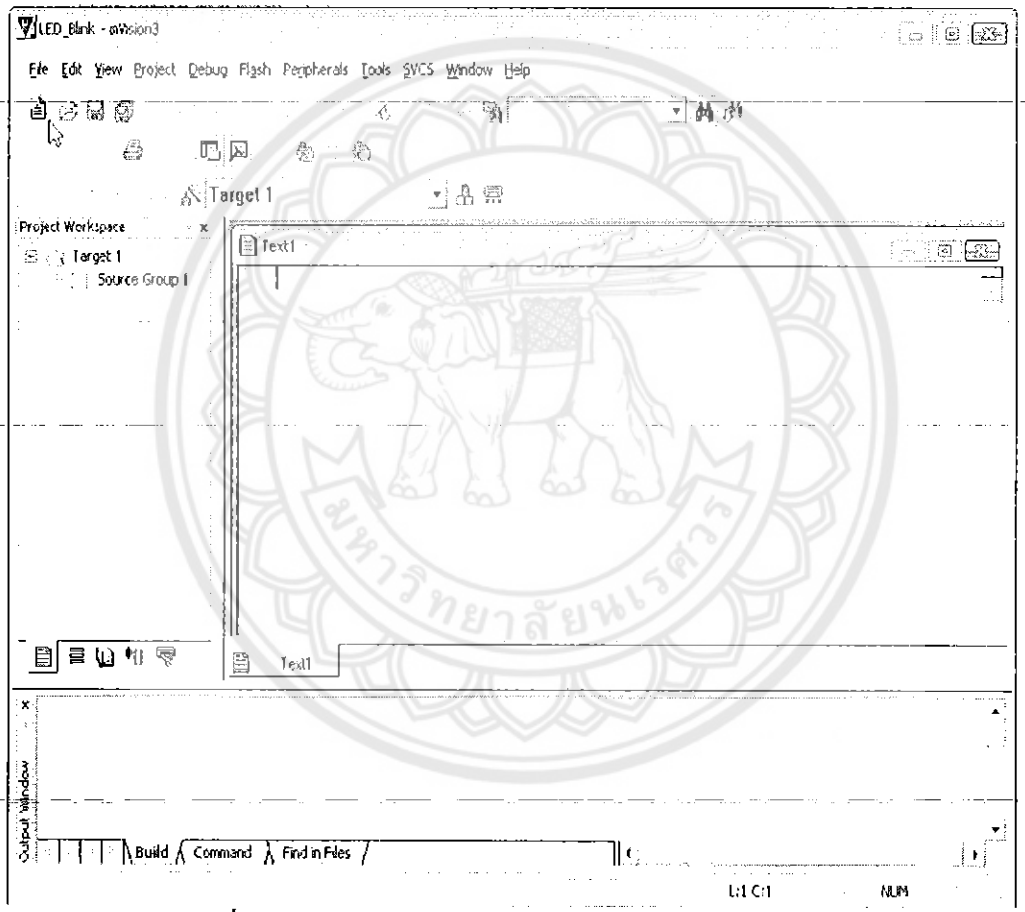


รูปที่ 3.1.3 (a) แสดงหน้าต่าง Select Device for Target 'Target'



รูปที่ 3.1.3 (b) แสดงหน้าต่าง Select Device for Target 'Target'

7) จากนั้นให้คลิกที่ไอคอน Create a New File ( ) จะได้หน้าต่าง Text1 ออกมาดังรูปที่ 3.1.4 ซึ่งจะใช้สำหรับเขียนโปรแกรม

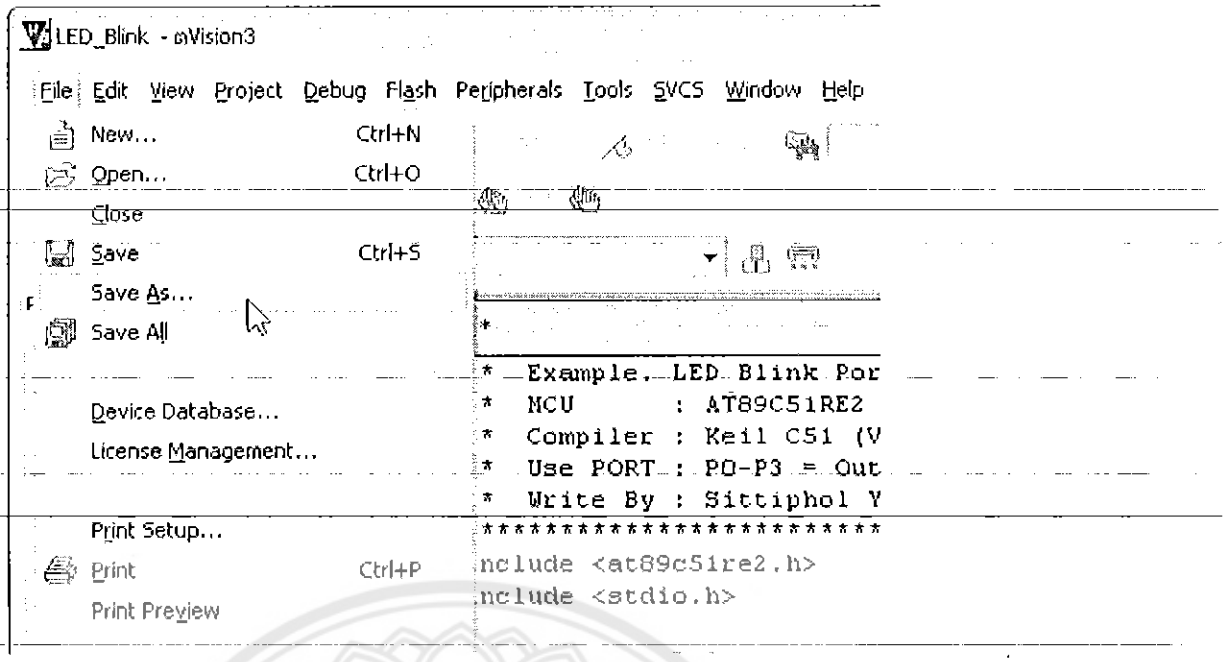


รูปที่ 3.1.4 แสดงหน้าต่าง Text1 สำหรับใช้เขียนโปรแกรม

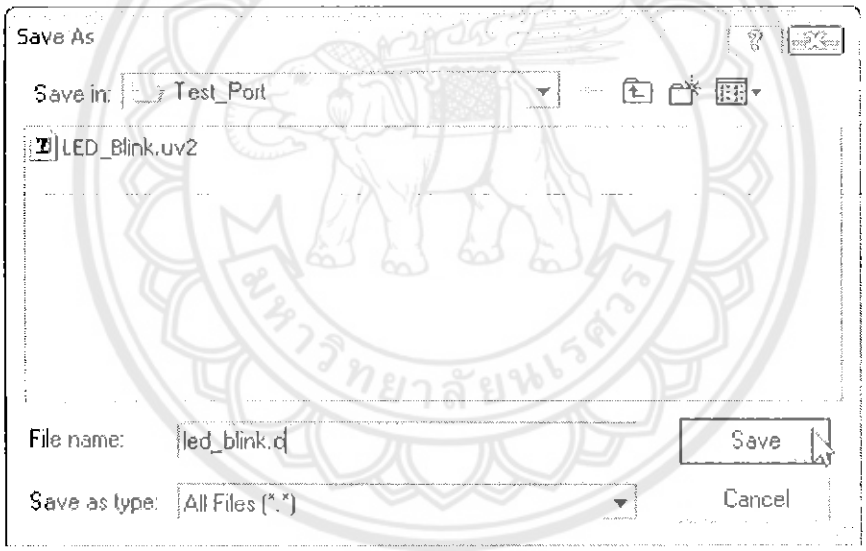
8) ทำการเขียนโปรแกรมตามตัวอย่างด้านล่างลงในหน้าต่าง Text1 เมื่อเขียนโปรแกรมเรียบร้อยก็ให้ไปที่เมนู Fileเลือก save as...(รูป ก.) จะได้หน้าต่างดังรูป ข. ให้เลือก Save ไว้ใน Folder เดียวกับ Project File ในตอนแรกจากนั้นทำการตั้งชื่อ File เป็นนามสกุลจุด C ในตัวอย่างจะตั้งชื่อเป็น led_blink.c และกด Save

ตัวอย่างโปรแกรม led_blink.c

```
/************************************************
* Example. LED Blink Port *
* MCU : AT89C51RE2 *
* Compiler : Keil C51 (V8.05 a) *
* Use PORT : P0-P3 = Output Connect LED *
************************************************/
#include <at89c51re2.h>
#include <stdio.h>
//---------- Delay ----------
void delay(int count)
{ int i , j ;
for(i=0 ; i<=count ; i++)
for(j=0 ; j<=count ; j++) ;
}
main()
{
CKCON0 = 0xFE ; // Set MCU 12 Clock Mode
BMSEL = 0x00 ; // Select Bank 0+Command Bank สำหรับพื้นที่ Flash
//---------- Test Out put port ----------
while(1){
P0 = 0x00 ; // Sent data 0 Out Port P0
P1 = 0x00 ;
P2 = 0x00 ;
P3 = 0x00 ;
delay(200) ;
P0 = 0xFF ; // Sent data 1 Out Port P0
P1 = 0xFF ;
P2 = 0xFF ;
P3 = 0xFF ;
delay(200) ;
}}
```



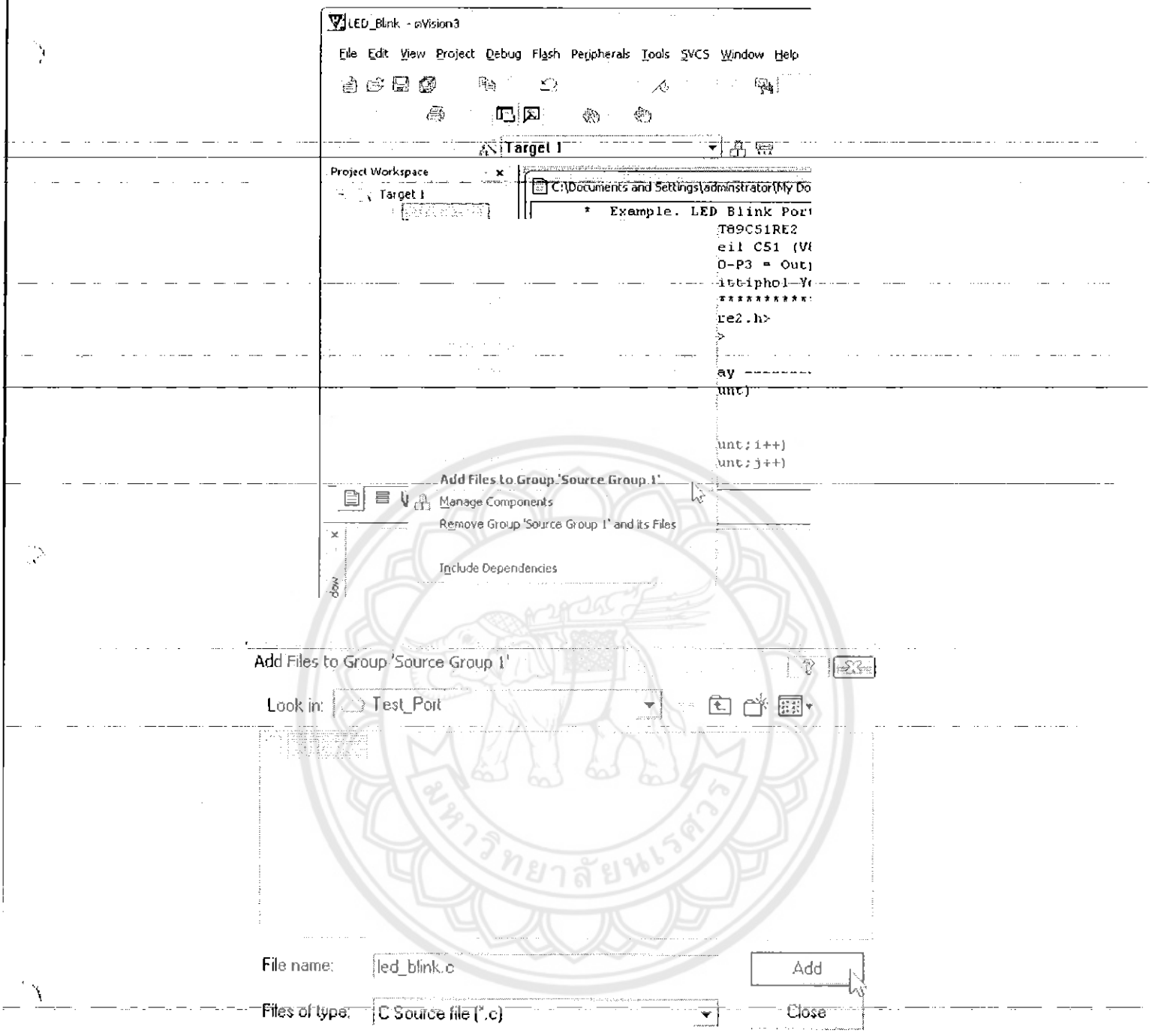
ก) เลือก Save As...



ข) ทำการตั้งชื่อ File.c และ Save File

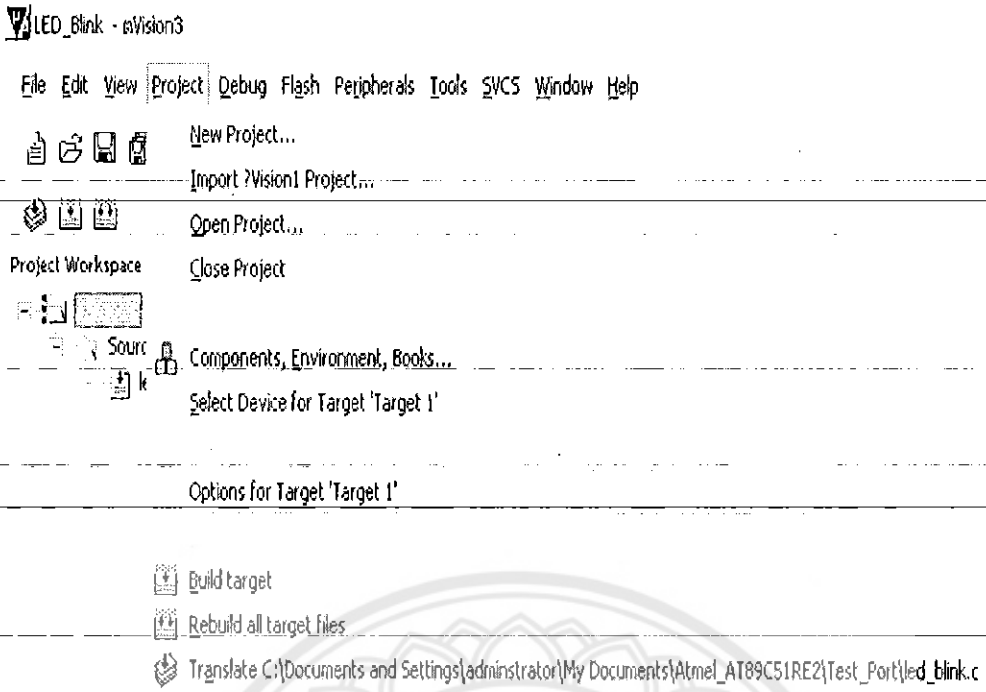
รูปที่ 3.1.5 แสดงหน้าต่างการ Save File . c

9) เมื่อ Save File แล้วให้ทำการ Add File led_blink.c เข้ามายัง Project โดยให้ Double Click หรือ คลิกขวาที่ FolderSource Group 1 ที่อยู่ในหน้าต่างด้านซ้ายมือ จากนั้นเลือก Add Files to Group 'Sourec Group 1' จะปรากฏหน้าต่างให้ Add file ขึ้นมา ให้ผู้ใช้เลือก File led_blink.c แล้วกด Add จากนั้นกด Close เพื่อปิดหน้าต่าง Add File ไฟล์ที่Add ก็จะเข้ามาอยู่ใน Folder Source Group 1



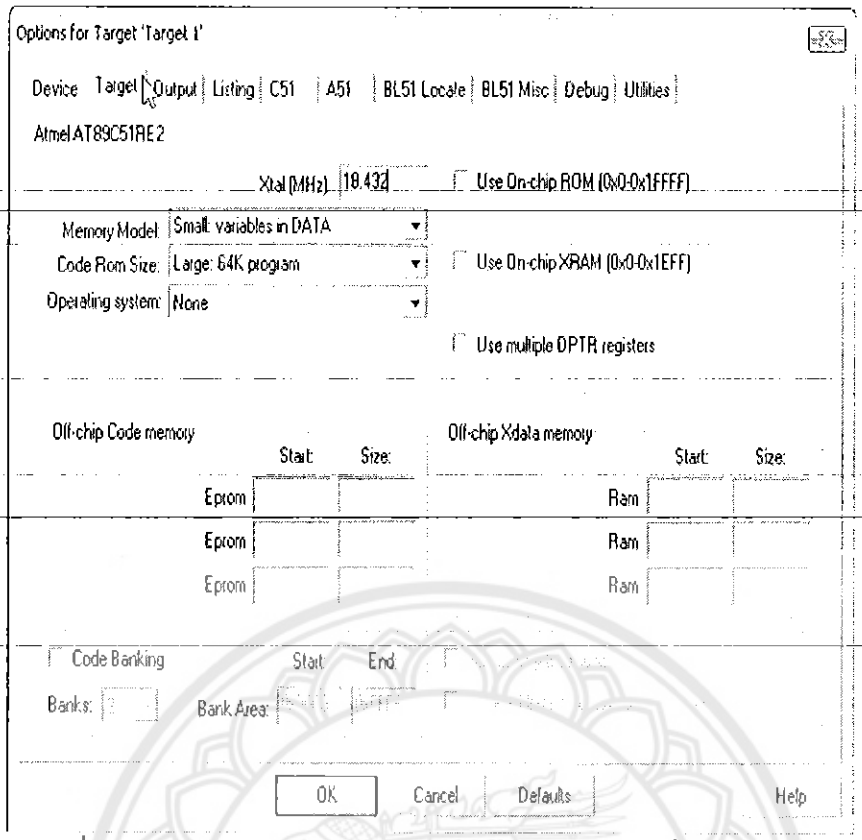
รูปที่ 3.1.6 แสดงหน้าต่างการ Add ไฟล์

10) เมื่อแอดไฟล์ที่เขียนเข้ามาเรียบร้อยแล้วให้ไปคลิกที่ Folder Target 1 ในช่องด้านซ้ายมือให้เป็นแถบสีน้ำเงินจากนั้นไปที่เมนู Project แล้วเลือก Options for Target 'Target 1' ดังแสดงในรูปที่ 3.1.7



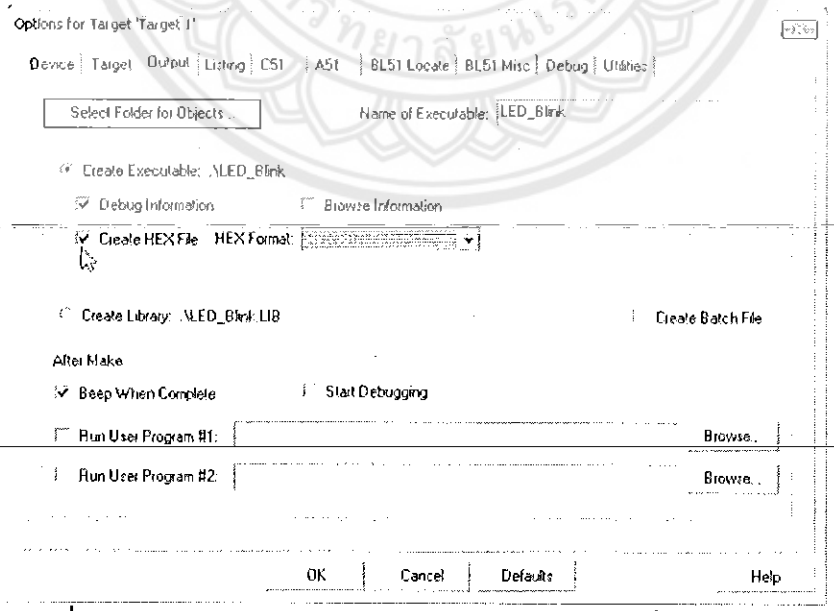
รูปที่ 3.1.7 แสดงการเลือก Option เพื่อ Setup Project

11) จะได้หน้าต่าง Options for Target 'Target1' ออกมา ให้คลิกที่ TAB Target แล้วทำการกำหนดค่าตามรูปที่ 3.1.8ให้สังเกตในช่อง Memory Model : ซึ่งในช่องนี้จะเป็นการเลือกขนาดของ Ram ที่ใช้ในการเขียนโปรแกรม ถ้าผู้ใช้เลือก Small:variables in DATA ตัวแปรที่ ประกาศในโปรแกรม จะถูกเก็บไว้ยังพื้นที่ RAM ภายใน ซึ่งขนาดของตัวแปรที่ประกาศในโปรแกรมจะต้องมีขนาดรวมกันไม่เกิน 128 byte , ถ้าเลือก Compact : variable in PDATAขนาดของตัวแปรที่ประกาศในโปรแกรมจะต้องมีขนาดรวมกันไม่เกิน 256 Byte , และถ้าเลือก Large : variable in XDATA ตัวแปรก็จะถูกเก็บไว้ในพื้นที่ XRAM ดังนั้น ขนาดของตัวแปรที่ประกาศในโปรแกรมจะต้องมีขนาดรวมกันไม่เกิน 8 K byte ซึ่งแล้วแต่ผู้ใช้จะเลือกใช้



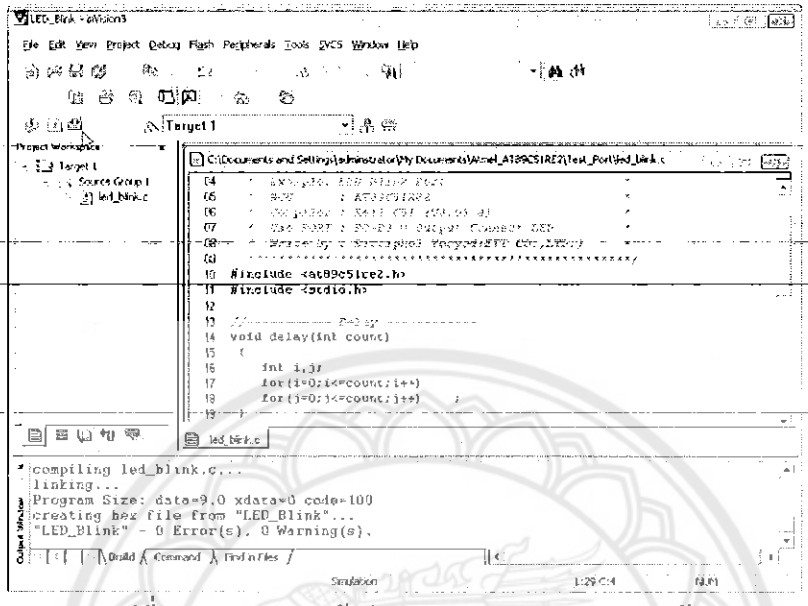
รูปที่ 3.1. 8 แสดงหน้าต่าง Option for Target 'Target 1' ที่ TAB Target

12) คลิกที่ TAB Output แล้วทำการ Tick เครื่องหมายถูกหน้าช่อง Create HEX File ส่วนช่อง HEX Format : ให้เลือกHEX-80 ส่วนช่องอื่นๆก็ให้กำหนดเหมือนในรูป เสร็จแล้วกด OK



รูปที่ 3.1.9 แสดงหน้าต่าง Option for Target 'Target 1' ที่ TAB Output

13) เมื่อ Set ค่าให้กับ Project เรียบร้อยแล้ว ให้ทำการ Compile โปรแกรมเพื่อตรวจสอบว่ามี Error หรือไม่ โดยคลิกที่ไอคอน Rebuild all target files ( ) ถ้าโปรแกรมที่เขียนไม่มีปัญหา ก็จะแสดง Error เป็น 0 อยู่ในหน้าต่างด้านล่าง ดังแสดงในรูปที่ 3.1.10



รูปที่ 3.1.10 แสดงหน้าต่างหลังจาก Compile ผ่านแล้ว

14) เมื่อ Compile เรียบร้อย ก็ให้ทำการ Download โปรแกรมลงใน MCU ได้ โดยใช้โปรแกรม Flip ดูวิธีการDownloadได้ในหัวข้อที่ 4 การ Download โปรแกรม ด้วย FLIP

การ Download โปรแกรม ด้วย FLIP

สำหรับบอร์ด CP-JR51RE2 V1.0 นี้ เราจะโหลดโปรแกรมที่เขียนขึ้นลงไปในตัว MCU โดยใช้โปรแกรม FLIP V3.1.0 Build 1 เป็นตัว Download โดย File ที่ Flip จะใช้ Download นั้นจะต้องเป็น File.hex ซึ่งไฟล์นี้จะได้มาจากในหัวข้อที่ 3 คือเมื่อเราเขียนโปรแกรมเรียบร้อย และ Compile ผ่านแล้ว ตัว Keil uVission3 ก็จะสร้าง File.hexให้กับผู้ใช้ จากนั้นถึงจะใช้ โปรแกรม Flip เป็นตัว Download File.hex นั้นลงบน MCU อีกทีหนึ่งในการ Download ก่อนอื่นผู้ใช้จะต้องทำการติดตั้งโปรแกรม Flip ที่ให้มากับแผ่น CD ลงในเครื่องก่อนซึ่งโปรแกรม Flip นี้จะต้องทำงานร่วมกับ Java ดังนั้นผู้ใช้จะต้องทำการติดตั้ง Java ลงไปในเครื่องด้วย ซึ่งจะสรุปขั้นตอนการ Download ได้ดังนี้

ขั้นตอนการ Download โปรแกรม (Hex File)

4.1) ทำการติดตั้งโปรแกรม Java ที่ให้มากับแผ่น CD ลงในเครื่องก่อนถ้าเครื่องของผู้ใช้ยังไม่มีโปรแกรม Java อยู่

4.2) ทำการติดตั้งโปรแกรม Flip ที่ให้มากับแผ่น CD ลงในเครื่องให้เรียบร้อย

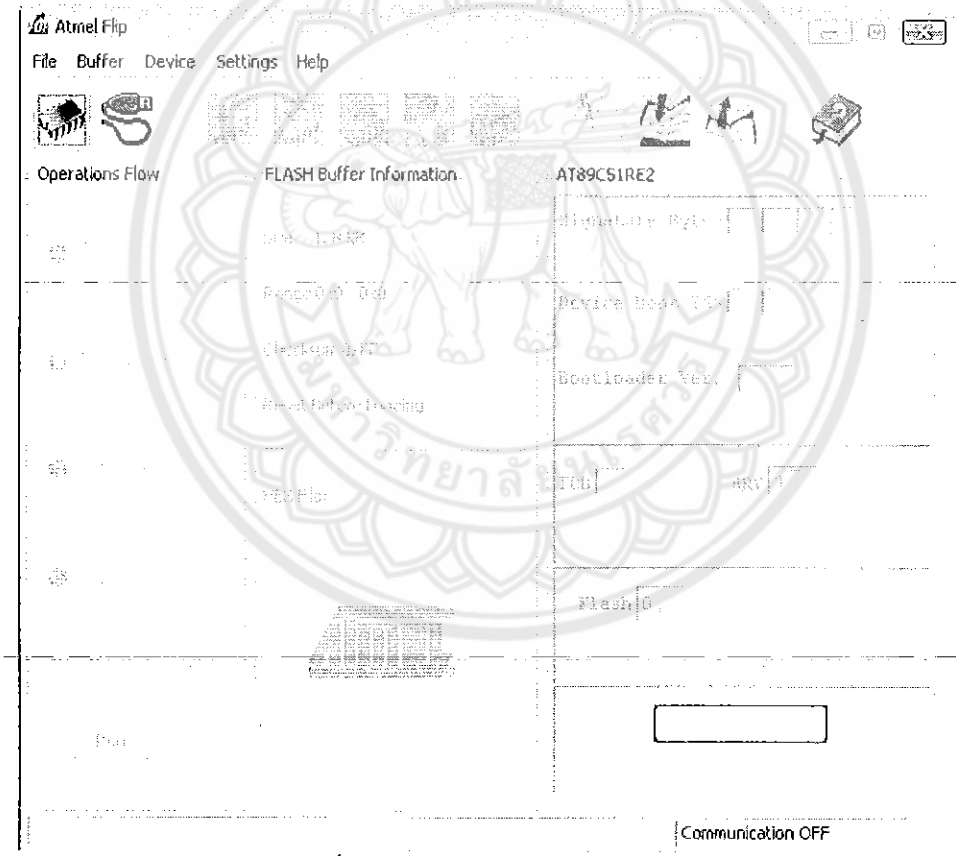
4.3) ต่อสาย RS232 จาก PC เข้ากับขั้วต่อ RS232#1 ของบอร์ด MCU

4.4) ทำการกดสวิตช์ PSEN และ RESET ดังต่อไปนี้เพื่อเข้าสู่ Monitor Mode

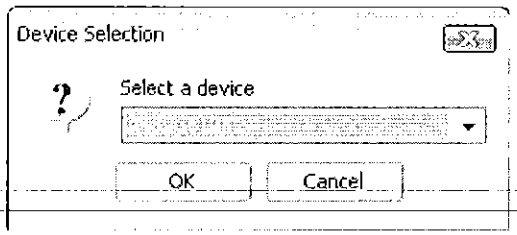
- กด SW. PSEN ค้างไว้
- ตามด้วยการกด SW. RESET ค้างไว้
- ปล่อย SW. RESET ในขณะที่ SW. PSEN ยังถูกกดค้างอยู่
- ปล่อย SW. PSEN เป็นลำดับสุดท้าย

4.5) ให้เปิดโปรแกรม Flip ขึ้นมา [ ] จะได้หน้าต่าง Atmel Flip ดังรูปที่ 4.1 ออกมา

4.6) ให้คลิกที่ไอคอน Select a Target Device [ ] จะได้หน้าต่าง Device Selection ดังรูปที่ 4.2 ในหน้าต่างนี้ให้ผู้ใช้ทำการเลือกเบอร์ MCU ที่ใช้งานในที่นี้คือ #AT89C51RE2 เมื่อเลือกเสร็จให้กด OK.

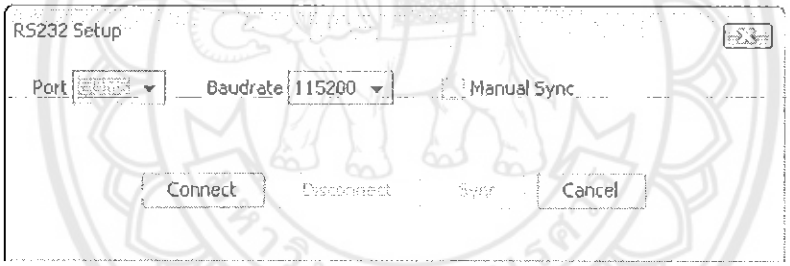


รูปที่ 4.1 แสดงหน้าต่าง Atmel Flip



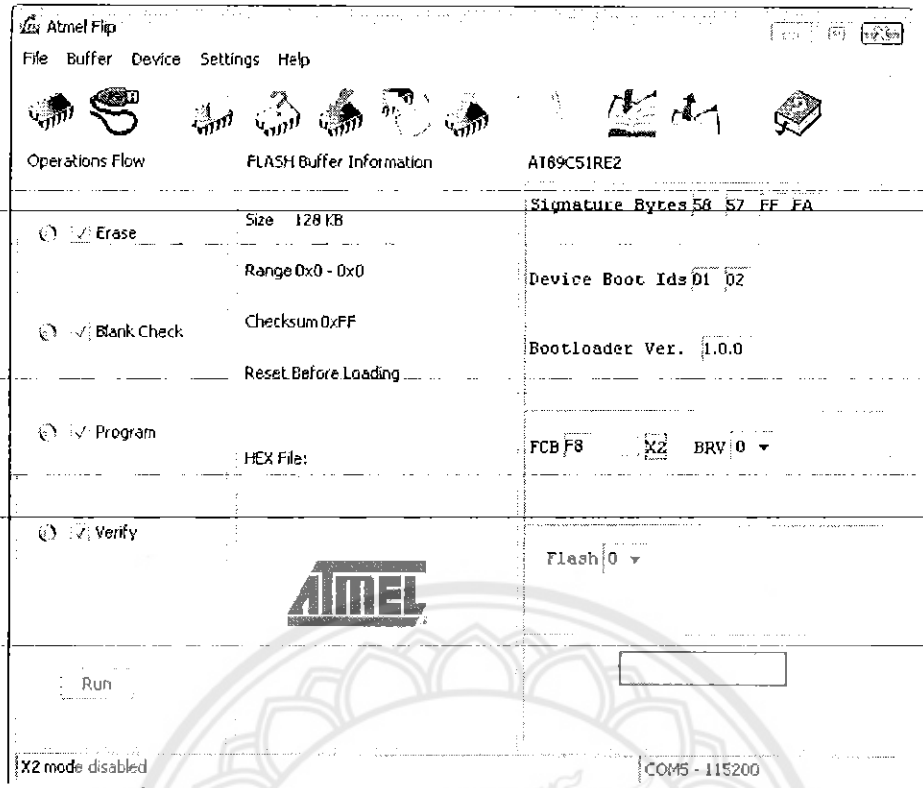
รูปที่ 4.2 แสดงหน้าต่าง Device Selection สำหรับเลือกเบอร์ MCU ที่ใช้งาน

4.7) ให้คลิกที่ไอคอน Select a Communication Medium [ ] เพื่อเลือก Port ในการสื่อสาร เมื่อคลิกแล้วจะได้หน้าต่างรูปเล็กออกมาจากนั้นให้เลือกที่ RS232 ก็จะได้หน้าต่าง RS232 Setup ดังรูปที่ 4.3 จากนั้นให้เลือกComPort ของ PC และเลือก Baud Rate (default =115200) ที่จะใช้ในการ Download โปรแกรม เมื่อเลือกเรียบร้อยแล้วให้คลิก Connect



รูปที่ 4.3 แสดงหน้าต่าง RS232 Setup สำหรับเลือก Com Port และ Baud Rate ในการ Download

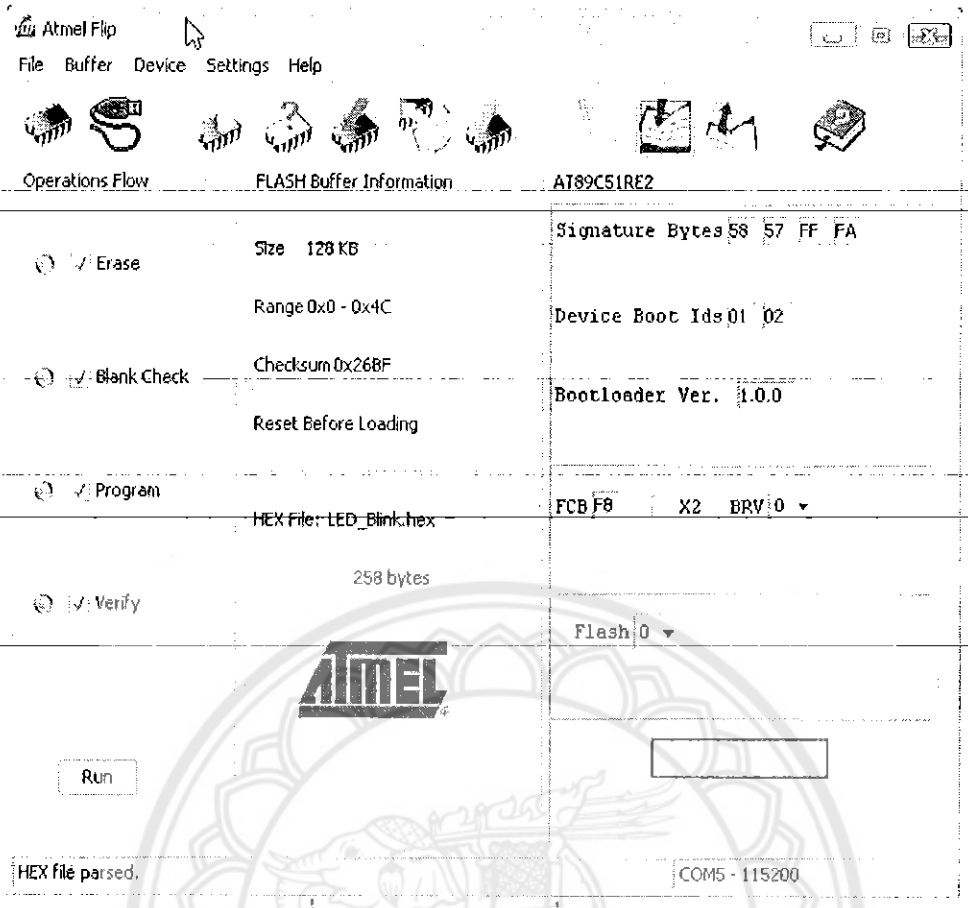
ก่อนที่จะคลิก Connect นั้นต้องแน่ใจว่า MCU ทำงานอยู่ใน Monitor Mode แล้ว ถ้ายังไม่แน่ใจก็ให้ทำการกดSW. PSEN และ RESET ตามขั้นตอนในข้อ 4.4 อีกครั้งหนึ่ง ถ้าการ Connect ผ่าน ไอคอนต่างๆในหน้าต่าง Atmel Flipก็จะถูก Enable ให้ใช้งานได้ ดังแสดงในรูปที่ 4.4 แต่ถ้า Connect ไม่ผ่านก็จะมีหน้าต่างขึ้นมาเตือนดังในรูปที่ 4.5 ซึ่งผู้ใช้จะต้องทำการตรวจสอบ การต่อสาย RS232 และ การเข้าสู่ Monitor Mode ว่าถูกต้องหรือไม่ แล้วค่อยลองทำการConnect ดูใหม่อีกครั้ง



รูปที่ 4.4 แสดงหน้าต่าง Atmel Flip หลังจาก Connect Port สำเร็จ

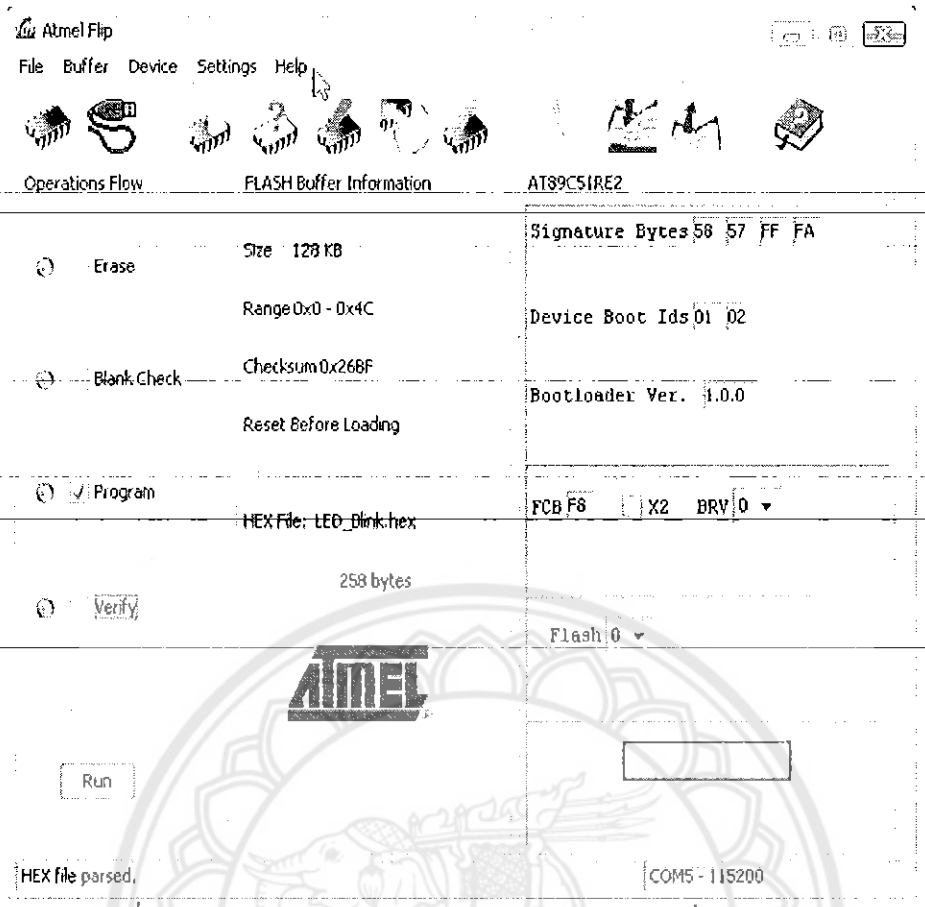
รูปที่ 4.5 แสดงหน้าต่าง Timeout error หลังจาก Connect Port เมื่อ Connect Port ไม่สำเร็จ

4.8) หลังจาก Connect Port ได้เรียบร้อยแล้วให้คลิกที่ไอคอน Load HEX file แล้วทำการเลือก file.Hex จากภายนอกเข้ามายังโปรแกรม Flip ชื่อของ file ที่เลือกเข้ามาก็จะปรากฏที่หน้าต่าง Atmel Flip ในช่อง Hex File : ดังแสดงในรูปที่4.6



รูปที่ 4.6 แสดง Hex File ที่ถูกโหลดเข้า

4.9) เมื่อโหลดไฟล์เข้ามาแล้ว ให้คลิกที่ปุ่ม Run เพื่อ Download โปรแกรมลงไปใน MCU ซึ่งถ้าผู้ใช้ต้องการให้โหลดเร็วขึ้น จากหน้าต่าง Atmel Flip ก่อนที่จะคลิกปุ่ม Run ในช่อง Operations Flow ให้ Tick เลือกเฉพาะช่อง Programเพียงช่องเดียวพอ ส่วนช่องอื่นให้ Tick ออกให้หมด ดังแสดงในรูปที่ 4.7 แล้วถึงคลิกปุ่ม Run



รูปที่ 4.7 แสดงการ Tick ช่อง Program และปุ่ม Run เพื่อ download

4.10) หลังจาก Download เรียบร้อยแล้ว ให้กดที่สวิตช์ RESET ของบอร์ด CP-JR51RE2 V1.0 เพื่อRUN โปรแกรมดูผลการทำงานของโปรแกรมที่เขียนขึ้น เมื่อจะ Download โปรแกรมใหม่ก็ให้ไปเริ่มทำในขั้นตอนที่ 4.4ใหม่

AT COMMANDS

ชุดคำสั่ง AT Commands

1. Call control

Commands	Description
ATA	Answer Command
ATD	Dial Command
ATH	Hang Up Call
ATL	Monitor Speaker Loudness
ATX	Call Progress Monitoring Control
ATO	Return To Online Data Mode
AT + CHUP	Hang Up Call
At + CLCC	List Current Calls
AT + CSNS	Single Numbering Scheme
AT + CSTA	Select type of phone number
AT + CVHU	Voice Hang up
AT + VTD	DTMF tone duration
AT + VTS	DTMF and Tone generation

2. Short Message Services – Point to Point

Command	Description
AT*E2SMSRI	Ring Indicator for SMS
AT+CGSMS	Select Service for MO SMS Messages
AT+CMGF	Message Format
AT+CMGW	Write Message to Memory
AT+CMGC	Send Command
AT+CMGS	Send Messages
AT+CMSS	Send from Storage
AT+CMGD	Delete Message
AT+CMGL	List Message
AT+CMGR	Read Message
AT+CNMI	New Message Indication to TE
AT+CMTI	New Message Indication Unsolicited Response
AT+CPMS	Preferred Message Storage
AT+CSCA	Service Center Address
AT+CSCS	Select Character Set
AT+CSDH	Show Text Mode Parameters
AT+CSMP	Set Text Mode Parameter
AT+CSMS	Select Message Service
AT*E2CMGA	Modify Message Attribute
AT*E2CMGL	List Message, without marking message Read
AT*E2CMGR	Read Message without Read mark

3. Short Message Service – Cell Broadcast

Command	Description
AT+CSCB	Select Cell Broadcast Message Type
AT*EMBOX	Mailbox Numbers
AT*EMWI	Message Waiting Indication

ภาคผนวก ข.

Code ของโปรแกรม

ภาคส่ง

```
#include "at89c51re2.h"          // ATMEL:AT89C51RE2 SFR : File
#include <stdio.h>
#include  <stdlib.h>
#include <intrins.h>             // For printf I/O functions
#include "24L04.h"               // EEPROM

bit int1_status;
sbit scl = P3^5;                 // I2C SCL Signal
sbit sda = P3^6;                 // I2C SDA Signal
sbit  row1 = P1^3;               // PORT KEYPAD
sbit  row2 = P1^4;
sbit  row3 = P1^5;
sbit  row4 = P1^6;
sbit  col1 = P1^0;
sbit  col2 = P1^1;
sbit  col3 = P1^2;

int time_cnt=0,time_hr=0,time_day=0;   // TIME
int val_temp=0;                  // LALU
int index_write=0;               // WRITE
int index_read=0;                // READ
int input_1=0;
char t1,t2,t3;                   // ประกาศตัวแปรอินพุท
char Mess[30],send=0,send2=0,send3=0,send4=0;        // ประกาศ Mess
char index_page=0,j_old;                             // ประกาศดัวชี้ข้อมูล
int sum1=0,sum2=0,sum3=0,sum4=0;
```

```
char start_rec=0;
code char TEL1[]={"","+','6','6','8','4','8','1','7','1','6','0','5',""};    // ประกาศเบอร์โทรภาครับ

char uart_buff[25],index_uart;                    // ประกาศการใช้ uart_buff
char temp_232[3];
unsigned char lcdbuf[16+1],kb,index_key;          // ประกาศlcdbuf[16+1],kb,index_key
char j,val1,val2,val3,item;                       // ประกาศ วัน และช่องของข้อมูล
char status_send=0,status_send2=0,status_send3=0,status_send4=0;
                                                  // ประกาศสถานะการส่งข้อความ
char  val4_1,val4_2,val4_3;                       // ประกาศช่องข้อมูล เช่น 1, 5, 6
char  val5_1,val5_2,val5_3;
char  val6_1,val6_2,val6_3;

int val4=0,val5=0,val6=0;
void ds1307_write_byte(unsigned char,unsigned char);   // Setup RTC
unsigned char ds1307_read_byte(unsigned char);         // Read RTC
void i2c_send_byte(unsigned char);                     // Write Byte I2C
unsigned char i2c_get_byte(void);                      // Read Byte I2C
void delay_i2c(void);

char page_index=0;                                // ประกาศตัวชี้ข้อมูล
 /* LCD Interface */

sfr  PORT_LCD = 0xA0;                 // LCD Interface = Port P2
sbit LCD_E  = PORT_LCD^1;             // Enable LCD(Active = "1")
sbit LCD_RW = PORT_LCD^2;             // RW LCD (0=Write,1=Read)
sbit LCD_RS = PORT_LCD^3;             // RS LCD (0=Instruction,1=Data)
sbit LCD_D4 = PORT_LCD^4;             // D4 LCD
sbit LCD_D5 = PORT_LCD^5;             // D5 LCD
sbit LCD_D6 = PORT_LCD^6;             // D6 LCD
sbit LCD_D7 = PORT_LCD^7;             // D7 LCD
```

```
void init_lcd(void);                    // Initial Character LCD(4-Bit Interface)
void gotolcd(unsigned char);            // Set Cursor LCD
void write_ins(unsigned char);          // Write Instruction LCD
void write_data(unsigned char);         // Write Data LCD
void enable_lcd(void);                  // Enable Pulse
char busy_lcd(void);                    // Read Busy LCD Status
void printlcd(void);                    // Display Message LCD
void delay(unsigned long);              // Delay Time Function(1..4294967295)
char i,uart1_buf[64];                   // UART1 Print String Buffer
int data_read=0;

/* pototype  section */
char putchar1(char ch);                 // Put Char To UART-1
char getchar1(void);                    // Get Char From UART-1
void print_uart1(void);                 // Print String to UART1

/*-----------------------------------------
 The main C function.  Program execution Here
-----------------------------------------*/

void delay_ms(unsigned int count)       // ประกาศฟังก์ชั่น delay
{
  unsigned int i;                       // Keil v7.5a
  while(count)                          // การนับ
  {
     i = 115;
    while(i>0) i--;
     count--;
  }
}
char read_kb(void)                      // การประกาศฟังก์ชั่นการเขียน key pad เช่นการกดเรียกดูข้อมูล
{
```

```
int time_kb=350;
row1=1;row2=1;row3=1;row4=1;
col1=0;col2=1;col3=1;//col4=1;
delay_ms(2);
if (row1==0)
{
delay_ms(time_kb);
 col1=1;
 return('1');
}
if (row2==0)
{
 delay_ms(time_kb);
 col1=1;
 return('4');

}
if (row3==0)
{
 delay_ms(time_kb);
 col1=1;
 return('7');

}
if (row4==0)
{
 delay_ms(time_kb);
 col1=1;
 return('*');

}

}
```

```
col1=1; col2=0;   delay_ms(2);
if (row1==0)
{
  delay_ms(time_kb);
  col2=1;
  return('2');

}
if (row2==0)
{
  delay_ms(time_kb);
  col2=1;
  return('5');

}
if (row3==0)
{
  delay_ms(time_kb);
  col2=1;
  return('8');

}
if (row4==0)
{
  delay_ms(time_kb);
  col2=1;
  return('0');

}

col2=1;col3=0;  delay_ms(2);
if (row1==0)
```

```
        {
          delay_ms(time_kb);
          col3=1;
          return('3');

        }
        if (row2==0)
        {
          delay_ms(time_kb);
          col3=1;
          return('6');

        }
        if (row3==0)
        {
          delay_ms(time_kb);
          col3=1;
          return('9');

        }
        if (row4==0)
        {
          delay_ms(time_kb);
          col3=1;
          return('#');
        }
       return(0);
}
void writedataToeeprom(int addr,unsigned char dat)
//ประกาศฟังก์ชั่นการอ่านค่าข้อมูลในหน่วยความจำ
{
     int a1,a2;
```

```
    (int)a1 = (int)((addr)/255);
    (int)a2 = (int)((addr)%255);
    write_2404((int)a1,(int)a2,dat);
    delay_ms(100);
}
char ReaddataFromeeprom(int addr)
//ประกาศฟังก์ชั่นการอ่านค่าข้อมูลในหน่วยความจำ
{
    int a1,a2;
    (int)a1 = (int)((addr)/255);
    (int)a2 = (int)((addr)%255);
    delay_ms(100);
    return(read_2404(a1,a2));
  }
void save_data(int address_start)          // ฟังก์ชั่นการ save data
{
    index_write = (address_start*19)  + 5;      // การเขียนข้อมูลที่ตำแหน่งใดๆ
    writedataToeeprom(index_write,item);  index_write++;
     j = ds1307_read_byte(2);                    // Get Hour
     writedataToeeprom(index_write,j);  index_write++;
     j = ds1307_read_byte(1);                    // Get Minute
    writedataToeeprom(index_write,j); index_write++;
     j = ds1307_read_byte(0);                    // Get Second
    writedataToeeprom(index_write,j);  index_write++;
      j = ds1307_read_byte(4);                   // Get Day
     writedataToeeprom(index_write,j); index_write++;
     j = ds1307_read_byte(5);                    // Get Month
    writedataToeeprom(index_write,j); index_write++;
    j = ds1307_read_byte(6);                     // Get Year
    writedataToeeprom(index_write,j); index_write++;
    writedataToeeprom(index_write,t1);  index_write++;
    writedataToeeprom(index_write,t2);  index_write++;
```

```
        writedataToeeprom(index_write,t3);  index_write++;
        writedataToeeprom(index_write,val4_1);  index_write++;
        writedataToeeprom(index_write,val4_2);  index_write++;
        writedataToeeprom(index_write,val4_3);  index_write++;
        writedataToeeprom(index_write,val5_1);  index_write++;
        writedataToeeprom(index_write,val5_2);  index_write++;
        writedataToeeprom(index_write,val5_3);  index_write++;
        writedataToeeprom(index_write,val6_1);  index_write++;
        writedataToeeprom(index_write,val6_2);  index_write++;
        writedataToeeprom(index_write,val6_3);
    }
    void read_data_lcd(int address_start)       // ฟังก์ชั่นการอ่านข้อมูล
    {
        index_read =  (address_start*19)  + 5;        // การอ่านข้อมูลที่ตำแหน่งใดๆ
        gotolcd(0);                                   // ไปที่ LCD
        sprintf(lcdbuf,"[%i]",(int)ReaddataFromeeprom(index_read)); printlcd();  index_read++;
        sprintf(lcdbuf,"%02BX",ReaddataFromeeprom(index_read));     printlcd();index_read++;
        sprintf(lcdbuf,"%02BX",ReaddataFromeeprom(index_read));     printlcd();index_read++;
        sprintf(lcdbuf,"%02BX",ReaddataFromeeprom(index_read));     printlcd();index_read++;
        sprintf(lcdbuf,"%02BX",ReaddataFromeeprom(index_read));     printlcd();index_read++;
        sprintf(lcdbuf,"%02BX",ReaddataFromeeprom(index_read));     printlcd(); index_read++;
        sprintf(lcdbuf,"%02BX|",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"              ");
        gotolcd(0x40);
        printlcd();
        gotolcd(0x40);
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c|",ReaddataFromeeprom(index_read));   printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c|",ReaddataFromeeprom(index_read));   printlcd();index_read++;
```

```
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c|",ReaddataFromeeprom(index_read));   printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c",ReaddataFromeeprom(index_read));    printlcd();index_read++;
        sprintf(lcdbuf,"%c|",ReaddataFromeeprom(index_read));   printlcd();
    }
    void read_data_232(int address_start)
    {
        index_read =  (address_start*19)  + 5;
        printf("_%i_=",(int)ReaddataFromeeprom(index_read)); index_read++;
        printf("%02BX:",ReaddataFromeeprom(index_read));    index_read++;
        printf("%02BX:",ReaddataFromeeprom(index_read));    index_read++;
        printf("%02BX-",ReaddataFromeeprom(index_read));    index_read++;
        printf("%02BX/",ReaddataFromeeprom(index_read));    index_read++;
        printf("%02BX/",ReaddataFromeeprom(index_read));    index_read++;
        printf("%02BX-",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c:",ReaddataFromeeprom(index_read));   index_read++;
        printf("%c:",ReaddataFromeeprom(index_read));   index_read++;
        printf("%c|",ReaddataFromeeprom(index_read));   index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c|",ReaddataFromeeprom(index_read));   index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c|",ReaddataFromeeprom(index_read));   index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c",ReaddataFromeeprom(index_read));    index_read++;
        printf("%c|",ReaddataFromeeprom(index_read));
    }
    void main (void)                    // ประกาศฟังก์ชั่น การเริ่มรันโปรแกรม ครั้งแรกที่เปิดเครื่อง
    {
```

```
char loop=1,uart_data;          // Char Buffer For UART
char telno=3,sec;               // ประกาศเรียกเบอร์โทร
char counter=0,sender=0;
int time_set=0;
/* Initial AT89C51RE2 Start Operate */
CKCON = 0x01;                   // Initial X2 Mode (36.864 MHz)
IE0 = 0x00;                     // Initial Interrupt Control
BMSEL = 0x00;                   // Default Select Flash = Bank-0
AUXR |= 0x01;                   // Inhibit ALE Signal
AUXR &= ~0x02;                  // EXTRAM = 0 (MOVX = Accress XRAM)
AUXR |= 0x1C;                   // XRS[2.0]=111= XRAM Size = 8192 Byte
AUXR1 &= ~0x20;                 // Config U2=0(Multiplex SFR:80H=SCON1)
/* Start of Config AT89C51RE2:UART0,UART1 */
SCON = 0x50;                    // UART0 = Mode 1 (N,8,1)
SCON1 = 0x50;                   // UART1 = Mode 1 (N,8,1)
/* Select Generate Baudrate By Internal-Baud */
TCLK = 0;                       // Disable Timer2 Generate TX Baudrate
RCLK = 0;                       // Disable Timer2 Generate RX Baudrate
BDRCON0 |= 0x0C;    // TBCK:RBCK=1:1 = Used Internal Buad Generate UART0 Baudrate
BDRCON1 |= 0x0C;    // TBCK:RBCK=1:1 = Used Internal Buad Generate UART1 Baudrate
BDRCON0 &= ~0x01;               // SRC0=0 = Select Fosc to Baudrate
BDRCON1 &= ~0x01;               // SRC1=0 = Select Fosc to Baudrate
/* Setup Internal-Baud Generate Baudrate Fast Mode */
/* Support Baudrate : 4800,9600,19200,... ,115200 */
PCON  |= 0x80;                  // UART0:SMOD0 = 1 (Enable Double Baudrate)
BDRCON1 |= 0x80;                // UART1:SMOD1 = 1 (Enable Double Baudrate)
BDRCON0 |= 0x02;                // SPD0=1 = Fast Baudrate Generator
BDRCON1 |= 0x02;                // SPD1=1 = Fast Baudrate Generator
BRL0   = 0x88;                  // Setup UART0 Baudrate 9600BPS
BRL1   = 0x88;                  // Setup UART1 Baudrate 9600BPS
BDRCON0 |= 0x10;                // BRR0=1 = Start Internal Baud1
BDRCON1 |= 0x10;                // BRR1=1 = Start Internal Baud1
```

```
/* Start Keil-C51 Transmit Function */
TI = 1;                                  // Set TI to send First char of UART
TI1 = 1;
/* End of Config AT89C51RE2:UART0,UART1 */

/* Setup UART Interrupt Control */
ES = 0;                                  //Disable UART0 Interupt
IEN1 &= ~0x08;
sec = ds1307_read_byte(0);
ds1307_write_byte(0,sec & 0x7f);         // Disable UART1 Interupt

                    // Disable UART1 Interupt
/*
ds1307_write_byte(2,0x01);
ds1307_write_byte(1,0x10);
ds1307_write_byte(0,0x0);
ds1307_write_byte(4,0x14);
ds1307_write_byte(5,0x05);
ds1307_write_byte(6,0x09);     */
init_lcd();
P1=0xff; delay_ms(1000);
P1=0x00; delay_ms(3000);
gotolcd(0);                              // Set Cursor Line-1
sprintf(lcdbuf,"test sms now!!!!");      // Display Line-1
printlcd();
if  (read_kb()=='0')                     // clear all memory
{
  write_2404(0,0,1);
  printf("clear data on memory success!!!!\r\n");
}
printf("start key\r\n");
while(0)
```

```
{
 kb =  read_kb();
 if   (read_kb()!= 0)                    // clear all memory
 {
   printf("key = %c\r\n",kb);
 }
}
 item = (int)read_2404(0,0);
 index_write=5;                          // เขียนตำแหน่งที่5
 index_read=5;                           // อ่านตำแหน่งที่5
 val1=0;
 val2 = 0;
 val3 = 0;
 val4 =0;
 index_write=5;                          // เขียนตำแหน่งที่5
 index_read=5;                           // อ่านตำแหน่งที่ 5
 printf("start  item  = %i,index write = %i index read =
%i\r\n",(int)item,(int)index_write,(int)index_read);
  P1=0x00; delay_ms(1000);
  P1=0xff; delay_ms(1000);
  loop=1;
  P0=0xff;
  gotolcd(0);
  sprintf(lcdbuf,"  waitting GSM  ");
  printlcd();
  while(loop)
  {
     printf("AT"); putchar(0x0d);
     if (RI)
     {
       RI=0;
       uart_data = _getkey();
```

```
            if (uart_data=='O') {loop=0;}
            else if (uart_data=='K') {loop=0;}
           }
          else delay_ms(1000);
           }
          for(i=0;i<10;i++)                          // show time and date
          {
           gotolcd(0);
          sprintf(lcdbuf,"%02BX:",ds1307_read_byte(2));    printlcd();
          sprintf(lcdbuf,"%02BX:",ds1307_read_byte(1));    printlcd();
          sprintf(lcdbuf,"%02BX  ",ds1307_read_byte(0));   printlcd();
          gotolcd(0x40);
          sprintf(lcdbuf,"   %02BX/",ds1307_read_byte(4)); printlcd();
          sprintf(lcdbuf,"%02BX/",ds1307_read_byte(5));    printlcd();
          sprintf(lcdbuf,"%02BX  ",ds1307_read_byte(6));   printlcd();

        delay_ms(300);
       }
          gotolcd(0);
          sprintf(lcdbuf," Ready to use ");                // ข้อความแสดงบน lcd เมื่อเครื่องพร้อมทำงาน
          printlcd();
          delay_ms(3000);
          printf("at+cmgf=1\r");    delay_ms(100);// เพื่อกำหนดรูปแบบของข้อความเป็น text mode
          status_send=0;
          input_1=0;
          time_cnt=0;
          status_send=0;status_send2=0;status_send3=0;status_send4=0;

          //test
          time_hr=0;
          val6_1='0';
          val6_2='0';
```

```
        val6_3='0';
        val6=0;

    // time_cnt=57;
        while(1)
        {
        //======== send check hardware  every Monday==========
          j = ds1307_read_byte(3);
          if ((j==2) && (status_send==0))
          {
            status_send=1;
            printf("AT+CMGS=");                    // คำสั่งส่ง SMS
             for(i=0;i<14;i++) putchar(TEL1[i]);
            putchar(0x0d); delay_ms(500);
            printf("@@@@@@@@");                    // ข้อความที่ส่งออกไป
            putchar(0x1A);
          }
          if (j!=2)  status_send=0;
          //**********************เงื่อนไข**********************
         j = ds1307_read_byte(1);
        if (j!=j_old)
        {
          (int)time_cnt++;
           if (time_cnt >=60) {time_hr++;  time_cnt=0;  // การแสดงเวลาใน lcd
           status_send=0;status_send2=0;status_send3=0;status_send4=0;}
          gotolcd(0);
          sprintf(lcdbuf," %iday %ihr %imin ",time_day,time_hr,time_cnt);   // Display Line-1
          printlcd();
          delay_ms(3000);   gotolcd(0);
          sprintf(lcdbuf,"   flash [%i]  ",input_1,uart_buff[3]);
           printlcd();
           gotolcd(0x40);
```

```
            sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
      ",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
            printlcd();
      /*********************กรณีที่ 1 น้ำท่า**********************
            j_old = j;
          }
      send=0;
          if   (time_hr == 1)                  //ในเวลา 1 ชั่วโมง
      {
          if (val4 <150)                       // normal น้ำท่าน้อยกว่า 150
          {
           if   (input_1 >=30)       // มากกว่าเท่ากับ30 mm ขึ้นไปแจ้งเตือน Flash, flood High
           {
           sprintf(Mess,"Flash ,flood, High %i mm/1h  ",input_1);
           send=1;
             }
           else if   (input_1 >=20)  // มากกว่าเท่ากับ20 mm ขึ้นไปแจ้งเตือน Flash, flood middle
           {
           sprintf(Mess,"Flash ,flood, Middle %i mm/1h  ",input_1);
           send=1;
            }
           else if   (input_1 >=10)  //มากกว่าเท่ากับ10 mm ขึ้นไปแจ้งเตือน Flash, flood low
           {
           sprintf(Mess,"Flash ,flood, Low %i mm/1h  ",input_1);
           send=1;
           }
          }
          else                         // full น้ำท่ามากกว่า 150
          {
           if   (input_1 >=30)    //มากกว่าเท่ากับ30mm ขึ้นไปแจ้งเตือน flood, flood High
           {
```

```
                sprintf(Mess,"flood,flood, High %i mm/1h  ",input_1);
                send=1;
                }
                else if   (input_1 >=20)  //มากกว่าเท่ากับ20mm ขึ้นไปแจ้งเตือน flood, flood High
                {
                sprintf(Mess,"flood,flood, High %i mm/1h  ",input_1);
                send=1;
                }
                else if   (input_1 >=10)  //มากกว่าเท่ากับ10mm ขึ้นไปแจ้งเตือน flood, flood High
                {
                sprintf(Mess,"flood,flood, Middle %i mm/1h  ",input_1);
                send=1;

                }
              }
              time_hr=0;
              time_day++;               //24 hour

            }

            if   (time_day >= 24)
            {
              if (val4 <150)            // normal <150
              {
                if   (input_1 >=100)   // มากกว่าเท่ากับ100mmขึ้นไปแจ้งเตือน Flash, flood High
                {
                  sprintf(Mess,"Flash ,flood, High %i mm/1Day ",input_1);
                  send=1;

                }
                else if   (input_1 >=85)  // มากกว่าเท่ากับ 85mmขึ้นไปแจ้งเตือน Flash, flood Middle
                {
```

```
                sprintf(Mess,"Flash ,flood, Middle %i mm/1Day",input_1);
                send=1;

            }
            else if   (input_1 >=70)  // มากกว่าเท่ากับ 70mmขึ้นไปแจ้งเตือน Flash, flood Low
            {
            sprintf(Mess,"Flash ,flood, Low %i mm/1Day",input_1);
            send=1;
            }
          }
          Else                              // full > 150
          {
            if   (input_1 >=100)   //มากกว่าเท่ากับ 100mmขึ้นไปแจ้งเตือน flood, flood High
            {
              sprintf(Mess,"flood,flood, High %i mm/1Day ",input_1);
              send=1;

            }
            else if   (input_1 >=85)  //มากกว่าเท่ากับ85 mmขึ้นไปแจ้งเตือน flood, flood High
            {
              sprintf(Mess,"flood,flood, High %i mm/1Day",input_1);
              send=1;

            }
            else if   (input_1 >=70)  // มากกว่าเท่ากับ 70 mmขึ้นไปแจ้งเตือน flood, flood Middle
            {
            sprintf(Mess,"flood,flood, Middle %i mm/1Day",input_1);
            send=1;
             }
          }
        }
      if   (time_day >= 72)                 // ในเวลา 3 วัน
```

```
{
    if (val4 <150)                    // normal <150
    {
     if   (input_1 >=200)  //มากกว่าเท่ากับ 200 mm  ขึ้นไปแจ้งเตือน Flash, flood High
     {
     sprintf(Mess,"Flash, flood, High %i mm/3Day ",input_1);
     send=1;

     }
     else if   (input_1 >=185)  //มากกว่าเท่ากับ185 mmขึ้นไปแจ้งเตือน Flash, flood Middle
     {
     sprintf(Mess,"Flash, flood, Middle %i mm/3Day ",input_1);
     send=1;
      }
     else if   (input_1 >=170)  //มากกว่าเท่ากับ 170 mm ขึ้นไปแจ้งเตือน Flash, flood Low
     {
     sprintf(Mess,"Flash, flood, Low %i mm/3Day ",input_1);
     send=1;

     }
    }
    else                              // full >150
    {
     if   (input_1 >=200)  //มากกว่าเท่ากับ 200 mm  ขึ้นไปแจ้งเตือน flood, flood High
     {
     sprintf(Mess,"flood,flood, High %i mm/3Day ",input_1);
     send=1;

     }
     else if   (input_1 >=185)  //มากกว่าเท่ากับ 185 mm  ขึ้นไปแจ้งเตือน flood, flood High
     {
     sprintf(Mess,"flood,flood, High %i mm/3Day ",input_1);
```

```
                send=1;

                }
                else if  (input_1 >=170) //มากกว่าเท่ากับ170 mm ขึ้นไปแจ้งเตือน flood, flood Middle
                {
                sprintf(Mess,"flood,flood, Middle %i mm/3Day ",input_1);
                send=1;
                }
              }
               time_day=0;
            }
            looprx: if(RI1)                  // Check UART1 Receive
             {
              uart_data = getchar1();     // Wait Receive Byte From UART1

             if   ((uart_data=='S') || (uart_data=='e') || (uart_data=='n') || (uart_data=='d') ||
(uart_data=='i') || (uart_data=='n') || (uart_data=='g') || (uart_data=='.'))  goto looprx;
            // ถ้ามีข้อมูลเข้ามาตัวแรกเป็นs,e,n,d,i,g ให้ข้ามไป
            if (uart_data == '$')  start_rec=1;
            if (uart_data == '!')  start_rec=0;

            if ((uart_data > 10) && (start_rec==1))
            {
               if (uart_data=='A')  index_uart=0; // ข้อมูลที่เข้ามาตัวแรกเป็นAเริ่มอ่านข้อมูลได้
              uart_buff[index_uart] =  uart_data;
               (int)index_uart++;
              //$ A000 B0C0 D000 E000 F000 !
              // 0123  4567 89AB CDEF GH01 2
            }
            if (index_uart >=20)                    // มีข้อมูลเข้ามา 20 ตัว
            {
```

```
//val1 =uart_buff[1];
//************************* 1 ************
temp_232[0] = uart_buff[1];
temp_232[1] = uart_buff[2];
temp_232[2] = uart_buff[3];
(int) input_1 = input_1+atoi(temp_232)/10;
//****************************************



val2 =uart_buff[5];
val3 =uart_buff[7];
val4_1=uart_buff[9];          //Dxxx!  8 910 11  12
val4_2=uart_buff[10];
val4_3=uart_buff[11];
temp_232[0] = val4_1;
temp_232[1] = val4_2;
temp_232[2] = val4_3;


val4=0;
val4 = atoi(temp_232);
val5_1=uart_buff[13];         //Exxx!  13 14-16  17
val5_2=uart_buff[14];
val5_3=uart_buff[15];
 temp_232[0] = val5_1;
temp_232[1] = val5_2;
temp_232[2] = val5_3;
val5=0;
val5 = atoi(temp_232);        //แปลงจากตัวหนังสือเป็นตัวเลข


 val6_1=uart_buff[17];        //Fxxx!  18 19-21  22
val6_2=uart_buff[18];
```

```
                val6_3=uart_buff[19];
                temp_232[0] = val6_1;
                temp_232[1] = val6_2;
                temp_232[2] = val6_3;

                val6=0;
                val6 = atoi(temp_232);          //แปลงจากตัวหนังสือเป็นตัวเลข

                t1 = val1;
                t2 = val2;
                t3 = val3;
                index_uart=0;

                gotolcd(0x0);
                sprintf(lcdbuf,"   flash [%i]    ",input_1,uart_buff[3]);
                printlcd();
                gotolcd(0x40);
                sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
            ",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
                printlcd();
                 delay_ms(3000);
                uart_buff[1]='0';

            }
              // Echo Data to UART0
        }
            if ((send==1) && (status_send==0))
            {
                printf("AT+CMGS=");        // การส่งข้อความ SMS
                for(i=0;i<14;i++) putchar(TEL1[i]);
                putchar(0x0d); delay_ms(500);
                j = ds1307_read_byte(2); printf("%02BX:",j);
```

```
                j = ds1307_read_byte(1); printf("%02BX:",j);
                j = ds1307_read_byte(4); printf("%02BX/",j);
                j = ds1307_read_byte(5); printf("%02BX/",j);
                j = ds1307_read_byte(6); printf("%02BX-",j);

                printf("%s",Mess);
                putchar(0x1A);
                gotolcd(0x40);
                sprintf(lcdbuf," Sending sms!!! ");
                printlcd();
                delay_ms(5000);
                status_send=1;
                save_data(item);
                item++;
                write_2404(0,0,item);
                delay_ms(5000);
                    gotolcd(0x0);
                    sprintf(lcdbuf,"  flash [%i]  ",input_1,uart_buff[3]);
                    printlcd();
                    gotolcd(0x40);
                    sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
                    printlcd();
            }
            //********************* sensor 2 น้ำท่าเมื่อมีข้อมูลเข้ามา <190*******************
            if  (val4 >=230)            // มากกว่าเท่ากับ 230 mm  ขึ้นไปแจ้งเตือน High
            {
                sprintf(Mess,"Flood warning, High %c%c%c cm  ",val4_1,val4_2,val4_3);
                send2=1;
            }
            else if  (val4 >=210)       //มากว่าเท่ากับ 210 mm  ขึ้นไปแจ้งเตือน Mid
            {
```

```
            sprintf(Mess,"Flood warning, Mid %c%c%c cm   ",val4_1,val4_2,val4_3);
            send2=1;
        }
        else if  (val4 >=190)        //มากกว่าเท่ากับ 190 mm  ขึ้นไปแจ้งเตือน Low
        {
            sprintf(Mess,"Flood warning, Low %c%c%c cm   ",val4_1,val4_2,val4_3);
            send2=1;
        }
        if  (time_day >= 1)          // เวลามากกว่าเท่ากับ 1วัน
        {
          if  (val4 >=170)           // มากกว่าเท่ากับ 170 cm
          {
            sprintf(Mess,"Flood warning, M %c%c%c cm 1D  ",val4_1,val4_2,val4_3);
            send2=1;
          }
        }
            if ((send2==1) && (status_send2==0))
            {
                printf("AT+CMGS=");
                for(i=0;i<14;i++) putchar(TEL1[i]);

                putchar(0x0d); delay_ms(500);
                j = ds1307_read_byte(2); printf("%02BX:",j);
                j = ds1307_read_byte(1); printf("%02BX:",j);
                j = ds1307_read_byte(4); printf("%02BX/",j);
                j = ds1307_read_byte(5); printf("%02BX/",j);
                j = ds1307_read_byte(6); printf("%02BX-",j);

                printf("%s",Mess);
                putchar(0x1A);
                gotolcd(0x40);
                sprintf(lcdbuf," Sending sms!!! ");
```

```
                printlcd();
                delay_ms(2000);
                status_send2=1;
                save_data(item);
                item++;
                write_2404(0,0,item);
               send2=0;delay_ms(5000);
                        sprintf(lcdbuf,"   flash [%i]  ",input_1,uart_buff[3]);
                        printlcd();
                        gotolcd(0x40);
                        sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
                        printlcd();
            }
            if  (val5 >=50)
            {
              sprintf(Mess,"Soil mois, warning H %c%c%c Per",val5_1,val5_2,val5_3);
              send3=1;
            }

            if ((send3==1) && (status_send3==0))
            {
               printf("AT+CMGS=");    //การส่งข้อความ
                for(i=0;i<14;i++) putchar(TEL1[i]);

               putchar(0x0d); delay_ms(500);
               j = ds1307_read_byte(2); printf("%02BX:",j);
               j = ds1307_read_byte(1); printf("%02BX:",j);
               j = ds1307_read_byte(4); printf("%02BX/",j);
               j = ds1307_read_byte(5); printf("%02BX/",j);
               j = ds1307_read_byte(6); printf("%02BX-",j);
```

```
                printf("%s",Mess);
                putchar(0x1A);
                gotolcd(0x40);
                sprintf(lcdbuf," Sending sms!!! ");
                printlcd();
                delay_ms(2000);
                status_send3=1;
                save_data(item);
                item++;
                write_2404(0,0,item);
                send3=0; delay_ms(5000);
                        sprintf(lcdbuf,"  flash [%i]  ",input_1,uart_buff[3]);
                        printlcd();
                        gotolcd(0x40);
                        sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
                        printlcd();
            }
          //**********************
             if  (val6>=10)                  //ถ้าข้อมูลเข้ามามากกว่าเท่ากับ 10ให้แจ้งเตือน
            {
              sprintf(Mess,"Lanslide warning, H %c%c%c cm  ",val6_1,val6_2,val6_3);
              send4=1;
            }
            if ((send4==1) && (status_send4==0))
            {
               printf("AT+CMGS=");
               for(i=0;i<14;i++) putchar(TEL1[i]);
               putchar(0x0d); delay_ms(500);
               j = ds1307_read_byte(2); printf("%02BX:",j);
               j = ds1307_read_byte(1); printf("%02BX:",j);
               j = ds1307_read_byte(4); printf("%02BX/",j);
```

```
                j = ds1307_read_byte(5); printf("%02BX/",j);
                j = ds1307_read_byte(6); printf("%02BX-",j);
                printf("%s",Mess);
                putchar(0x1A);
                gotolcd(0x40);
                sprintf(lcdbuf," Sending sms!!! ");
                printlcd();
                delay_ms(2000);
                status_send4=1;
                save_data(item);
                item++;
                write_2404(0,0,item);
                send4=0;
                delay_ms(5000);
                        sprintf(lcdbuf,"   flash [%i]  ",input_1,uart_buff[3]);
                        printlcd();
                        gotolcd(0x40);
                        sprintf(lcdbuf," [%c%c%c][%c%c%c][%c%c%c]
",val4_1,val4_2,val4_3,val5_1,val5_2,val5_3,val6_1,val6_2,val6_3);
                        printlcd();
            }

        kb =  read_kb();            //การเรียกใช้ key pad
        if (kb > 0)
        {
         if (kb=='1')               // กด1ดูข้อมูลปัจจุบัน
          {
            read_data_lcd(item-1);
            page_index = item-1;
            read_data_lcd(page_index);
            read_data_232(page_index);
          }
```

```
else if (kb=='2')                    //กด2ดูข้อมูลย้อนหลัง
{
  read_data_lcd(page_index--);
  read_data_232(page_index);
}
else if (kb=='3')                    //กด3 ดูข้อมูลเพิ่ม
{
  read_data_lcd(page_index++);
  read_data_232(page_index);
}
else if (kb=='9')                    //กด9 test การส่งข้อมูล
{
      printf("AT+CMGS=");
      for(i=0;i<14;i++) putchar(TEL1[i]);

    putchar(0x0d); delay_ms(500);
    j = ds1307_read_byte(2); printf("%02BX:",j);
    j = ds1307_read_byte(1); printf("%02BX:",j);
    j = ds1307_read_byte(4); printf("%02BX/",j);
    j = ds1307_read_byte(5); printf("%02BX/",j);
    j = ds1307_read_byte(6); printf("%02BX-",j);

    printf("test send message Flash f",Mess);
    putchar(0x1A);
    gotolcd(0x40);
    sprintf(lcdbuf," Sending sms!!! ");
    printlcd();
    delay_ms(1000);
}
else if (kb=='6')                    //กด 6 test การส่งข้อมูล
{
      printf("AT+CMGS=");
}
```

```
        for(i=0;i<14;i++) putchar(TEL1[i]);

        putchar(0x0d); delay_ms(500);
        j = ds1307_read_byte(2); printf("%02BX:",j);
        j = ds1307_read_byte(1); printf("%02BX:",j);
        j = ds1307_read_byte(4); printf("%02BX/",j);
        j = ds1307_read_byte(5); printf("%02BX/",j);
        j = ds1307_read_byte(6); printf("%02BX-",j);

        printf("Flood warning Low 10 mm",Mess);
        putchar(0x1A);
        gotolcd(0x40);
        sprintf(lcdbuf," Sending sms!!! ");
        printlcd();
        delay_ms(1000);

    }
    else if (kb=='7')                    //กด7 test การส่งข้อมูล
    {
        printf("AT+CMGS=");
        for(i=0;i<14;i++) putchar(TEL1[i]);

        putchar(0x0d); delay_ms(500);
        j = ds1307_read_byte(2); printf("%02BX:",j);
        j = ds1307_read_byte(1); printf("%02BX:",j);
        j = ds1307_read_byte(4); printf("%02BX/",j);
        j = ds1307_read_byte(5); printf("%02BX/",j);
        j = ds1307_read_byte(6); printf("%02BX-",j);

        printf("Flood warning, Middle 20 mm ",Mess);
        putchar(0x1A);
        gotolcd(0x40);
```

```
                sprintf(lcdbuf," Sending sms!!! ");
                printlcd();
                delay_ms(1000);
            }
        else if (kb=='8')                    //กด8 test การส่งข้อมูล
        {
          printf("AT+CMGS=");
           for(i=0;i<14;i++) putchar(TEL1[i]);

          putchar(0x0d); delay_ms(500);
          j = ds1307_read_byte(2); printf("%02BX:",j);
          j = ds1307_read_byte(1); printf("%02BX:",j);
          j = ds1307_read_byte(4); printf("%02BX/",j);
          j = ds1307_read_byte(5); printf("%02BX/",j);
          j = ds1307_read_byte(6); printf("%02BX-",j);

          printf("Flood warning, High 50 mm ",Mess);
          putchar(0x1A);
          gotolcd(0x40);
          sprintf(lcdbuf," Sending sms!!! ");
          printlcd();
          delay_ms(1000);
        }
    }
  }
}
/*****************************/
/* Write Character To UART1 */
/*****************************/
char putchar1 (char c)              //ส่งข้อมูลไปยัง UART1
{  if (c == '\n')                   // If Line Feed(LF)
   {
```

```
  while (!TI1);
  TI1 = 0;
  SBUF1 = 0x0D;                    // Auto Add CR(LF+CR)
 }
  while (!TI1);
  TI1 = 0;                         //ถ้าส่ง เท่ากับ 0
  return (SBUF1= c);
 }
 /*****************************/
 /* Get character From UART1 */
 /*****************************/
 char getchar1 ()                  //รับข้อมูลจาก UART1
 {
  char c;
  while (!RI1);
  c = SBUF1;                       //รีจีสเตอร์ที่เก็บรายการรับ
  RI1 = 0;                         //ถ้ารับเท่ากับ0

  return (c);
 }
 /*****************************/
 /* Print String to UART1 */
 /*****************************/
 void print_uart1(void)            //ส่งทีละประโยค
 {
  char *p;                         // Pointer Buffer
  p = uart1_buf;                   // UART1 Buffer
  do                               // Get char & Print Until null
  {
   putchar1(*p);                   // Write char to UART1
   p++;                            // Next char
  }
```

```
    while(*p != '\0');              // End of ASCII (null)

    return;
}
/*******************************/
/* Initial LCD 4-Bit Interface */
/*******************************/
extern void init_lcd(void)
{
    unsigned int i;                 // Delay Count
    LCD_E = 0;                      // Start LCD Control (Disable)
    LCD_RS = 0;                     // Default Instruction
    LCD_RW = 0;                     // Default = Write Direction
    for (i=0;i<10000;i++);          // Power-On Delay (15 mS)

    PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x30;               // DB5:DB4 = 1:1
    enable_lcd();                   // Enable Pulse
    for (i=0;i<2500;i++);           // Delay 4.1mS

    PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x30;               // DB5:DB4 = 1:1
    enable_lcd();                   // Enable Pulse
    for (i=0;i<100;i++);            // delay 100uS

    PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x30;               // DB5:DB4 = 1:1
    enable_lcd();                   // Enable Pulse
    while(busy_lcd());              // Wait LCD Execute Complete

    PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x20;               // DB5:DB4 = 1:0
```

```
 enable_lcd();                  // Enable Pulse
 while(busy_lcd());             // Wait LCD Execute Complete
 write_ins(0x28);               // Function Set (DL=0 4-Bit,N=1 2 Line,F=0 5X7)
 write_ins(0x0C);               // Display on/off Control (Entry Display,Cursor off,Cursor not
Blink)
 write_ins(0x06);               // Entry Mode Set (I/D=1 Increment,S=0 Cursor Shift)
 write_ins(0x01);               // Clear Display  (Clear Display,Set DD RAM Address=0)
}
/******************/
/* Set LCD Cursor */
/******************/
void gotolcd(unsigned char i)
{
 i |= 0x80;                     // Set DD-RAM Address Command
 write_ins(i);
}
/*****************************/
/* Write Instruction to LCD */
/*****************************/
void write_ins(unsigned char i)
{
 LCD_RS  = 0;                   // Instruction Select
 LCD_RW  = 0;                   // Write Select

 PORT_LCD &= 0x0F;              // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= i & 0xF0;          // Strobe High Nibble Command
 enable_lcd();                  // Enable Pulse

 PORT_LCD &= 0x0F;              // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= (i << 4) & 0xF0;   // Strobe Low Nibble Command
 enable_lcd();                  // Enable Pulse
}
```

```
  while(busy_lcd());              // Wait LCD Execute Complete
}
/*****************************/
/* Write Data(ASCII) to LCD */
/*****************************/
void write_data(unsigned char i)
{
  LCD_RS  = 1;                    // Data Select
  LCD_RW  = 0;                    // Write Select
  PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= i & 0xF0;           // Strobe High Nibble Data
  enable_lcd();                   // Enable Pulse
  PORT_LCD &= 0x0F;               // Clear old LCD Data (Bit[7..4])
  PORT_LCD |= (i << 4) & 0xF0;    // Strobe Low Nibble Data
  enable_lcd();                   // Enable Pulse

  while(busy_lcd());              // Wait LCD Execute Complete
}
/************************/
/* Enable Pulse to LCD */
/************************/
void enable_lcd(void)             // Enable Pulse
{
  unsigned int i;                 // Delay Count
  LCD_E = 1;                      // Enable ON
  for (i=0;i<500;i++);
  LCD_E = 0;                      // Enable OFF
}
/******************/
/* Wait LCD Ready */
/******************/
char busy_lcd(void)
```

```
{
 unsigned int i;          // Dummy Byte Data
 LCD_RS = 0;              // Instruction Select
 LCD_RW = 1;              // Read Direction
 LCD_D7 = 1;              // Prepared Read Busy
 LCD_E = 1;               // Start Read Busy
 for(i=0;i<500;i++);      //Wait LCD Ready
 if(LCD_D7 == 1)          // Verify Busy Flag
 {
  LCD_E  = 0;             // Disable Read
  LCD_RW = 0;             // Default = Write Direction
  return 1;               // LCD Busy Status
 }
 else
 {
  LCD_E = 0;              // Disable Read
  LCD_RW = 0;             // Default = Write Direction
  return 0;               // LCD Ready Status
 }
}
/*****************************/
/* Print Data(ASCII) to LCD */
/*****************************/
void printlcd(void)
{
 char *p;
 p = lcdbuf;

 do                       // Get ASCII & Write to LCD Until null
 {
  write_data(*p);         // Write ASCII to LCD
  p++;                    // Next ASCII
```

```
    }
    while(*p != '\0');          // End of ASCII (null)

    return;
    }
    /*****************************************/
    /* Long Delay Time Function(1..4294967295) */
    /*****************************************/
    void delay(unsigned long i)
    {
     int j=0;
     while(i > 0) {i--;for(j=0;j<114;j++);}     // Loop Decrease Counter
     return;
    }


    void ds1307_write_byte(unsigned char ds1307_address,unsigned char ds1307_data)
    {
     sda = 0;              // I2C Start Condition
     scl = 0;
     delay_i2c();

     i2c_send_byte(0xD0);          // Send ID Code DS1307,Write (1101000+0)
     sda = 1;              // Release SDA
     scl = 1;              // Start ACK Clock
     delay_i2c();
     while(sda) {}
     scl = 0;              // End ACK Clock
     delay_i2c();

     i2c_send_byte(ds1307_address);      // Send DS1307 Address
     sda = 1;              // Release SDA
```

```
    scl = 1;                // Start ACK Clock
    delay_i2c();
    while(sda) {}
    scl = 0;                // End ACK Clock
    delay_i2c();
    i2c_send_byte(ds1307_data);       // Send DS1307 Data
    sda = 1;                // Release SDA
    scl = 1;                // Start ACK Clock
    delay_i2c();
    while(sda) {}
    scl = 0;                // End ACK Clock
    delay_i2c();
    sda = 0;            // Stop Bit(End of Data)

    scl = 1;            // I2C Stop Condition
    delay_i2c();
    sda = 1;

  return;
  }
  /*******************************/
  /* Read Data 1-Byte From DS1307 */
  /*******************************/
  unsigned char ds1307_read_byte(unsigned char ds1307_address)
  {
    unsigned char ds1307_data;     // Dummy Byte
    sda = 0;            // I2C Stat condition
    scl = 0;
    delay_i2c();
    i2c_send_byte(0xD0);          // Send ID Code DS1307,Write (1101000+0)
    sda = 1;              // Release SDA
    scl = 1;                  // Start ACK Clock
```

```
delay_i2c();
while(sda) {}
scl = 0;                // End ACK Clock
delay_i2c();

i2c_send_byte(ds1307_address);      // Send DS1307 Address
sda = 1;                // Release SDA
scl = 1;                // Start ACK Clock
delay_i2c();
while(sda) {}
scl = 0;                // End ACK Clock
delay_i2c();

scl = 1;                // I2C Stop Condition
delay_i2c();
sda = 1;

// New Start For Read //
sda = 0;                // I2C Stat condition
scl = 0;
delay_i2c();

i2c_send_byte(0xD1);            // Send ID Code DS1307,Read (1101000+1)
sda = 1;                // Release SDA
scl = 1;                // Start ACK Clock
delay_i2c();
while(sda) {}
scl = 0;                // End ACK Clock
delay_i2c();

ds1307_data = i2c_get_byte();       // Read 1-Byte From DS1307
```

```
    sda = 1;                    // Send Stop Bit (End of Read Data)
    scl = 1;                    // Start Stop Bit Clock
    delay_i2c();
    scl = 0;                    // End Stop Bit Clock
    delay_i2c();

    scl = 1;                    // I2C Stop Condition
    delay_i2c();
    sda = 1;

    return ds1307_data;
}
/*****************************/
/* Send Data 8 Bit to I2C Bus */
/*****************************/
void i2c_send_byte(unsigned char i)
{
 char j;        // Bit Counter Send

 for(j = 0;j < 8;j++)   // 8-Bit Counter Send Data
 {
  if((i & 0x80) == 0x80)    // Send MSB First
    {sda = 1;}        // Send Data = 1
  else
    {sda = 0;}        // Send Data = 0

  scl = 1;            // Release SDA
  delay_i2c();

  scl = 0;            // Next Bit Send
  delay_i2c();
```

```
    i <<= 1;          // Shift Data For Send (MSB <- LSB)
  }

  return;
}
/*******************************/
/* Get Data 8-Bit From I2C Bus */
/*******************************/
unsigned char i2c_get_byte(void)
{
 unsigned char i;   // Result Byte Buffer
 char j;        // Bit Counter Read Data

 for(j = 0; j < 8; j++)   // 8-Bit Counter Read Data
 {
  i <<= 1;        // Shift Result Save (MSB <- LSB)

  sda = 1;       // Release Data
  scl = 1;            // Strobe Read SDA
  delay_i2c();

  if(sda == 1)
   {
    i |= 0x01;        // Save Bit Data = 1
   }
  else
   {
    i &= 0xFE;        // Save Bit Data = 0
   }

  scl = 0;              // Next Bit Read
  delay_i2c();
```

```
    }

    return i;
}
/*****************************************/
/* Delay For I2C Bus Device Interface */
/*****************************************/
void delay_i2c(void)
{
  unsigned char i;
  i = 2;              // Delay Counter
  while(i > 0) {i--;}        // Loop Decrease Counter
  return;
}
/*********************************/
/* External INT1 Service Function */
/* Trig By RTC:DS1307 Every 1Hz  */
/*********************************/
void external1 (void) interrupt 2 using 1      // External-INT1 : Bank-1
{
  int1_status = 1;              // Set Interrupt Status
}
```

ภาครับ

```
#include "at89c51re2.h"          // ATMEL:AT89C51RE2 SFR : File
#include <stdio.h>
#include  <stdlib.h>
#include <intrins.h>                    // For printf I/O functions
#include "24L04.h"

sbit scl = P3^5;                 // I2C SCL Signal
sbit sda = P3^6;                        // I2C SDA Signal

char check_part=0;                      //ประกาศการ check_part
char flg=0,index_save,index=0,uart1_buf[100];   //ประกาศใช้ index
char sms_buf[50];                       //รับตัวแปรที่เก็บค่าอ่านมาซิม
char sms_save[50];                      //บันทึกในหน่วยความจำ
int timecnt=0,row=0;                    //ประกาศการนับรีเรย์
char cnt_feed=0;feedback=0;             //ประกาศการเช็คอุปกรณ์เครื่อง และการตอบ
กลับ
int t,index_write=0;                    //ตัวชี้การเก็บ
int index_read=0,cnt_siren2=0;          //ตัวชี้การอ่าน
char sirentime=0;                       //เวลาไซเรน
int cnt_siren=0;                 //นับดูไซเรนดังกี่ครั้ง

sbit sw1  = P1^0;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 0
sbit sw2  = P1^1;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 1
sbit sw3  = P1^2;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 2
sbit sw4  = P1^3;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 3
sbit sw5  = P1^4;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 4
sbit sw6  = P1^5;                       //การใช้สวิตช์ พอร์ตที่ 1 ขา 5
char satatus_alarm=0,RY=0;              //ประกาศสถานะของ alarm ว่าดังอย่างไร เช่น
H M L
char st_message=0;                      //ประกาศข้อความ
```

```
sbit RELAY = P1^7;                          //การใช้ Relay
unsigned char j,val1,val2,val3,val4,item,j1,j2;          //ประกาศการใช้ วันที่,ช่องรับอินพุท
,item

char Datasend=0,loop=0,lcdbuf[16+1];        //ประกาศการใช้ชุดข้อมูล มาที่ LCD Display
Buffer

int  indexcount[5];                         //ประกาศตัวชี้การนับ
char starttel=0,tel[12],indextel=0;         //ประกาศการใช้เบอร์โทรศัพท์
char check_number=0;                        //ประกาศการเช็คเบอร์ว่าตรงกับที่เราตั้งไว้หรือไม่
int page_index=0,time_set;

//code char fix_number1[12]={'+','6','6','8','2','1','6','5','3','4','0','9'};
//code char TEL1[]={"",'+','6','6','8','0','5','1','7','6','7','5','4',""};

 code char fix_number1[12]={'+','6','6','8','2','1','6','5','3','4','0','9'};   //เบอร์โทรในภาคส่ง


int time_check=0;
char telfix[12];
/* pototype  section */
char putchar1(char ch);                     // Put Char To UART-1
char getchar1(void);                        // Get Char From UART-1
void print_uart1(void);                     // Print String to UART1

/* LCD Interface */
sfr  PORT_LCD = 0xA0;                       // LCD Interface = Port P2
sbit LCD_E  = PORT_LCD^1;                   // Enable LCD(Active = "1")
sbit LCD_RW = PORT_LCD^2;                   // RW LCD (0=Write,1=Read)
sbit LCD_RS = PORT_LCD^3;                   // RS LCD (0=Instruction,1=Data)
sbit LCD_D4 = PORT_LCD^4;                   // D4 LCD
```

```
sbit LCD_D5 = PORT_LCD^5;                    // D5 LCD
sbit LCD_D6 = PORT_LCD^6;                    // D6 LCD
sbit LCD_D7 = PORT_LCD^7;                    // D7 LCD

/* pototype  section */
void init_lcd(void);                         // Initial Character LCD(4-Bit Interface)
void gotoled(unsigned char);                 // Set Cursor LCD
void write_ins(unsigned char);               // Write Instruction LCD
void write_data(unsigned char);              // Write Data LCD
void enable_lcd(void);                       // Enable Pulse
char busy_lcd(void);                         // Read Busy LCD Status
void printlcd(void);                         // Display Message LCD
void delay(unsigned long);                   // Delay Time Function(1..4294967295)

/*-------------------------------------------
 The main C function.  Program execution Here
-------------------------------------------*/
void ds1307_write_byte(unsigned char ds1307_address,unsigned char ds1307_data);
unsigned char ds1307_read_byte(unsigned char ds1307_address);


void delay_ms(unsigned int count)            //ประกาศฟังก์ชัน delay_ms
{
  unsigned int i;                            // Keil v7.5a
  while(count)                               //การนับ
{
    i = 115;
                while(i>0) i--;
    count--;
  }
}
```

```
void writedataToeeprom(int addr,char dat)          //ประกาศฟังก์ชันการอ่านค่าข้อมูลใน
หน่วยความจำ
{
        int a1,a2;

        (int)a1 = (int)((addr)/255);
        (int)a2 = (int)((addr)%255);
        write_2404((int)a1,(int)a2,dat);
        delay_ms(100);
}

char ReaddataFromeeprom(int addr)
{
    int a1,a2;
        (int)a1 = (int)((addr)/255);
        (int)a2 = (int)((addr)%255);
        delay_ms(100);
    return(read_2404(a1,a2));
}

void save_data(int address_start)                  //ฟังก์ชันการ save data
{
                int i,ii,x=0;

                index_write = (address_start*42)+1;        //ค่าความกว้างของตัวอักษร

                x=0;

                for(ii=index_write;ii<(index_write+42);ii++)
                {
                 writedataToeeprom(ii,sms_buf[x]); //การเก็บข้อมูลไปยังหน่วยความจำ
```

```
                    if      (sms_buf[x] =='L')      satatus_alarm = 'L'; //ถ้ามีตัว L ให้
alarm เสียงดัง10หยุด50
                    else if (sms_buf[x] =='M')      satatus_alarm = 'M'; //ถ้ามีตัว M ให้
alarm เสียงดัง20หยุด40
                    else if (sms_buf[x] =='H')      satatus_alarm = 'H'; //ถ้ามีตัว H ให้
alarm เสียงดัง40หยุด40
                    x++;
                    }
}

void read_data_lcd(int address_start)           //ฟังก์ชันการอ่านข้อมูลใน LCD และให้
ตัวหนังสือวิ่ง
{
        int xx,y1,iii,ii,x=0;
            gotolcd(0x0);


            j1 = ds1307_read_byte(2);
                j2 = ds1307_read_byte(1);


                //printf(" Time = %02BX:%02BX \r\n",j1,j2);


                if ((j1==0x08) && (j2==0x00) && (RY==0))
                {
                    RY=1;
                    RELAY=1;        // ON
                    delay_ms(800);
                    RELAY=0;

                    printf("Alarm on  08:00\r\n");
```

```
                }
                if (j1==0x09)  RY=0;
```

```
sprintf(lcdbuf,"    ");
printlcd();
```

```
index_read =  (address_start*42)+1;
```

```
sprintf(lcdbuf," index read = %i",address_start);
gotolcd(0); printlcd();

delay_ms(100);
```

```
    gotolcd(0x40);
    sprintf(lcdbuf,"          ");
    printlcd();
    delay_ms(500);
    ii =  index_read;
```

```
    {

     for (xx=ii;xx<=ii+26;xx++)
      {
```

```
        x=xx;
        sprintf(lcdbuf,"          ");

        for(y1=0;y1<16;y1++)
        {
```

```
                lcdbuf[y1]= ReaddataFromeeprom(x);
                x++;
                }

                gotolcd(0x40);

                printlcd();

                printf("status = %c \n",satatus_alarm);

                delay_ms(10);
                //*************************
                if (satatus_alarm == 'L')//ถ้ามีตัว Lเข้ามาให้เสียงดังนับไป10
```

แล้วหยุดไปจนกว่าจะนับถึง50

```
                {
                    cnt_siren++;
                    if (cnt_siren <= 10)  RELAY=1;
                    else  RELAY=0;

                    if (cnt_siren >=60)  cnt_siren=0;

                }
                    else if (satatus_alarm == 'M')      //ถ้ามีตัว Mเข้ามาให้
```

เสียงดังนับไป20แล้วหยุดไปจนกว่าจะนับถึง40

```
                    {
                        cnt_siren++;
                        if (cnt_siren <= 20)  RELAY=1;
                        else  RELAY=0;

                        if (cnt_siren >=60)  cnt_siren=0;
```

```
                    }
                    else if (satatus_alarm == 'H')      //ถ้ามีตัว Hเข้ามาให้
เสียงดังนับไป40แล้วหยุดไปจนกว่าจะนับถึง20
                    {
                        cnt_siren++;
                        if (cnt_siren <= 40)  RELAY=1;
                        else  RELAY=0;

                        if (cnt_siren >=60)  cnt_siren=0;

                    }

                    if (check_part==1)
                    {

                        cnt_siren2++;           //เสียงดัง 2 วินาทีเช็ค
อุปกรณ์เครื่อง
                        if (cnt_siren2 <= 50)  RELAY=1;
                        else  RELAY=0;

                        if (cnt_siren2 >=100)
{cnt_siren2=0;check_part=0;}

                    }
                //*****************************************

                    if (sw4==0)             //กดสวิตช์ที่4เพื่อ
หยุดเสียงของalarm
                    {
                        satatus_alarm='0';
                        RELAY=0; cnt_siren2=0;
```

```
}
}
}
}
}
void main (void)                                    //การเริ่มต้นโปรแกรมครั้งแรก
{
    char x,y,i,uart_data;                           //ประกาศตัวแปร x,y,i
  char count_pood=0;                                //ประกาศการนับ
    /* Initial AT89C51RE2 Start Operate */
    CKCON = 0x01;                                   // Initial X2 Mode (36.864
MHz)
  IE0 = 0x00;                                       // Initial Interrupt Control
    BMSEL = 0x00;                                   // Default Select Flash = Bank-0

  AUXR |= 0x01;                                     // Inhibit ALE Signal
    AUXR &= ~0x02;                                  // EXTRAM = 0 (MOVX =
Accress XRAM)
    AUXR |= 0x1C;                                   // XRS[2.0]=111= XRAM Size =
8192 Byte
    AUXR1 &= ~0x20;                                 // Config U2=0(Multiplex
SFR:80H=SCON1)

  /* Start of Config AT89C51RE2:UART0,UART1 */
    SCON = 0x50;                                    // UART0 = Mode 1 (N,8,1)

  SCON1 = 0x50;                                     // UART1 = Mode 1 (N,8,1)
```

```
        /* Select Generate Baudrate By Internal-Baud */
        TCLK = 0;                       // Disable Timer2 Generate TX
    Baudrate
        RCLK = 0;                       // Disable Timer2 Generate RX
    Baudrate
        BDRCON0 |= 0x0C;                // TBCK:RBCK=1:1 = Used Internal
    Buad Generate UART0 Baudrate
        BDRCON1 |= 0x0C;                // TBCK:RBCK=1:1 = Used Internal
    Buad Generate UART1 Baudrate
        BDRCON0 &= ~0x01;               // SRC0=0 = Select Fosc to Baudrate
        BDRCON1 &= ~0x01;               // SRC1=0 = Select Fosc to Baudrate

        /* Setup Internal-Baud Generate Baudrate Fast Mode */
        /* Support Baudrate : 4800,9600,19200,...  ,115200 */
        PCON    |= 0x80;                // UART0:SMOD0 = 1 (Enable Double
    Baudrate)
        BDRCON1 |= 0x80;                // UART1:SMOD1 = 1 (Enable
    Double Baudrate)
        BDRCON0 |= 0x02;                // SPD0=1 = Fast Baudrate Generator
        BDRCON1 |= 0x02;                // SPD1=1 = Fast Baudrate Generator
        BRL0    = 0x88;                 // Setup UART0 Baudrate 9600BPS
        BRL1    = 0x88;                 // Setup UART1 Baudrate 9600BPS
        BDRCON0 |= 0x10;                // BRR0=1 = Start Internal Baud1

        BDRCON1 |= 0x10;                // BRR1=1 = Start Internal Baud1

        /* Start Keil-C51 Transmit Function */
        TI = 1;                         // Set TI to send First char of UART
        TI1 = 1;
        /* End of Config AT89C51RE2:UART0,UART1 */

        /* Setup UART Interrupt Control */
```

```
        ES = 0;                                  // Disable UART0 Interupt
      IEN1 &= ~0x08;                             // Disable UART1 Interupt
        init_lcd();
      /* Print String to UART0 */

      /*
      ds1307_write_byte(2,0x10);
      ds1307_write_byte(1,0x20);
      ds1307_write_byte(0,0x00);
      ds1307_write_byte(4,0x13);
      ds1307_write_byte(5,0x05);
      ds1307_write_byte(6,0x09);  */

       RELAY=1;
       printf("Receive data from sim 300\n");          //แสดงเป็นข้อความ
        printf("Start now\n");

      gotolcd(0);
      sprintf(lcdbuf,"Send command 1/4");              //
      printlcd();


        sprintf (uart1_buf,"at+cfun=1"); print_uart1();      putchar1(0x0d);
        loop=1;
        while(loop)
        {
                    if (RI1)
                    {
                    RI1=0;
                    uart_data = getchar1();
                    if  (uart_data=='O')  loop=0;
                                        else if  (uart_data=='K')  loop=0;
```

```
                    }
                }
                 RELAY=0;
                                                    //printf("%s",uart1_buf);
                delay(500);
              gotolcd(0);
              sprintf(lcdbuf,"Send command 2/4 ");
              printlcd();

          sprintf (uart1_buf,"at+ifc=1,1"); print_uart1();       putchar1(0x0d);
            loop=1;
                while(loop)
                {
                        if (RI1)
                        {
                        RI1=0;
                        uart_data = getchar1();
                        if  (uart_data=='O')  loop=0;
                                        else if  (uart_data=='K')  loop=0;
                        }
                }
                delay(500);
                gotolcd(0);
            sprintf(lcdbuf,"Send command 3/4 ");
            printlcd();
                sprintf (uart1_buf,"at+cmgf=1"); print_uart1();        putchar1(0x0d);

            loop=1;
                while(loop)
                {
                        if (RI1)
                        {
```

```
                RI1=0;
                uart_data = getchar1();
                if (uart_data=='O') loop=0;
                                else if (uart_data=='K') loop=0;
                }
        }
    delay(500);
        gotolcd(0);
    sprintf(lcdbuf,"Send command 4/4  ");
    printlcd();

        sprintf (uart1_buf,"at+csclk=0"); print_uart1();      putchar1(0x0d);
    loop=1;
        while(loop)
        {
                if (RI1)
                {
                RI1=0;
                uart_data = getchar1();
                if (uart_data=='O') loop=0;
                                else if (uart_data=='K') loop=0;
                }
        }


    Datasend=1;
        flg=1;
```

```
            delay(200);
            //P1=0xff;
            row=0;

            for(i=0;i<12;i++)  telfix[i]= fix_number1[i];          // เช็คเบอร์ต้นทางว่า
ตรงกับที่ไว้หรือไม่
            gotolcd(0);                                            // Display Line-1
      sprintf(lcdbuf," waitting clear  ");                         // แสดงทีละประโยค
      printlcd();                                                  //แสดงทั้งประโยค

            for(i=2;i<30;i++)
            {

             printf(" delete %i\r\n",(int)i);
             sprintf (uart1_buf,"at+cmgd=%i",(int)i); print_uart1(); putchar1(0x0d);  //เพื่อสั่งลบ
ข้อความออกจากหน่วยความจำ
             loop=1;
             while(loop)
                    {
             if (RI1)
             {
             RI1=0;
             uart_data = getchar1();
             if  (uart_data=='O')  loop=0;
                     else if  (uart_data=='K')  loop=0;
             }
                    }
                    delay(300);
                    if (sw1==0) i=50;
             }
```

```
                delay(300);

                feedback=0;

                sirentime=0;

                delay_ms(500);

                if(sw1==0)
                {
                 item=0;
                 gotolcd(0);
                        sprintf(lcdbuf," Clear Index !!");          // Display Line-1
                printlcd();                                         // Set Cursor Line-1
                 writedataToeeprom(0,0);
                }


                if(sw6==0)
                {
                        sprintf(lcdbuf," Clear Eeprom !!");         // Display Line-1
                printlcd();
                  for(t=1;t<200;t++)
                  {
                  writedataToeeprom(t,0);
                  delay_ms(20);
                  }
                }

                sirentime=0;
                page_index=0;
```

```
        RELAY=0;

        printf("\f start program now!!!\r\n");
        RY=0;

        gotolcd(0);
        sprintf(lcdbuf,"  waitting GSM  ");
        printlcd();
        delay_ms(1000);

        sprintf (uart1_buf,"at");
        print_uart1();
        putchar1(0x0d);
}
        loop=1;
        while(loop)
        {
                if (RI1)
                {
                RI1=0;
                uart_data = getchar1();
                if  (uart_data=='O')  loop=0;
                                else if  (uart_data=='K')  loop=0;
                }
        }
        time_set=0;
        while(1)                                        // Loop Continue
        {
         printf("start main now  time set %i!!!\r\n",time_set++);
        /*      j1 = ds1307_read_byte(2);
                j2 = ds1307_read_byte(1);

}
```

```
                //printf(" Time = %02BX:%02BX \r\n",j1,j2);


                if ((j1==0x08) && (j2==0x00) && (RY==0))
                {
                        RY=1;
                        RELAY=1;      // ON
                        delay_ms(800);
                        RELAY=0;

                }
                if (j1==0x09)  RY=0;

                */

                if (sw2==0)
                {
                        if (item > 0) page_index--;

                        if  (page_index < 0)   page_index=0;

                }
                if (sw3==0)
                {

                                if (page_index < item )    page_index++;

                }
        gotolcd(0);                                             // Set Cursor Line-1
                sprintf(lcdbuf,"  Read sms %i    ",(int)page_index);          // Display Line-1
                printlcd();
                gotolcd(0x40);                                          // Set Cursor Line-2
```

```
                sprintf(lcdbuf,"Process running ");                    // Display Line-2
                printlcd();

if (item > 0) read_data_lcd(page_index);
                else
                {
                        gotolcd(0);
                                                        // Set Cursor Line-1
                        sprintf(lcdbuf," Not have data  ");             // Display Line-1
                        printlcd();
                                delay_ms(550);
                        }

if (sw4==0)
                {
                satatus_alarm=0;
                RELAY=0;
                check_part=0;

                }

for(i=0;i<30;i++) sms_buf[i]=0;
printf("\r\n");

index=0;
count_pood=0;
flg=0;
time_check=0;
sprintf (uart1_buf,"at+cmgr=1\r\n");
print_uart1();
```

```
        loop=1;
    }   while(loop)
        {
//j1 = ds1307_read_byte(2);
    // j2 = ds1307_read_byte(1);
//printf(" Time = %02BX:%02BX\r\n",j1,j2);
    if(RI1)                                   // Check UART1
Receive
    {
        time_check=0;
        uart_data = getchar1();
        putchar(uart_data); // send data to rs232
if  (uart_data =='K')
        {
            if (index > 0)
            index--;
    }
            sms_buf[index]='';
            index++;
            count_pood=0;
            delay(300);
            flg=1;
}
```

```
                    loop=0;
}

}
else  if  (uart_data =="")
{
    count_pood++;
    if      (count_pood==6)      //6
    {
    }
}
else  if  (uart_data > 0x0d) //&& (uart_data >=0x0a))
{
  if (starttel==1)
    {
        tel[indextel]= uart_data;
        indextel++;

    }
    if ((count_pood ==2)  && (starttel==0))
    {
        starttel=1;
        indextel=0;
    }

    if (count_pood==4) starttel=0;

    if (count_pood >=6)
    {

        if      (uart_data == '@')      cnt_feed++;
```

```
                                        if (cnt_feed >=5)  //5
                                        {
                                                cnt_feed=0;
                                        _feedback=1;
                                        }
                                        sms_buf[index]=uart_data;
                                        index++;
                                }
                        }
                }
                time_check++;
                if (time_check > 10000)   {loop=0;time_check=0;}

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

        }
        delay(8000);
if (index > 5)
                {

                printf("[%i]Data from sim 300 =  ",(int)index);
                for(i=0;i<index;i++) printf("%c",sms_buf[i]);
                index_save=0;

                printf("\r\n phone number in = ");

                check_number=1;
```

```
for(i=0;i<12;i++)
{
        _printf("%c",tel[i]);
        if (tel[i] != telfix[i])   check_number=0;
}
printf("\r\n check number = ");
for(i=0;i<12;i++)
{
        printf("%c",telfix[i]);
}
if (feedback==1)
{
    sprintf(uart1_buf,"at+cmgs=\""); print_uart1();
    for(i=0;i<12;i++)
    {
        putchar1(tel[i]);
    }
    putchar1("");
    //printf("\"");
    putchar1(0x0d); delay(300);
printf (uart1_buf,"send sms check ready to host OK"); print_uart1();
    putchar1(0x1a);

    printf("\r\nsend sms check ready to host  ");
    feedback=0;
```

```
                    check_part=1;
                }

                if(check_number == 0)
                {
                    printf(" Wrong number \r\n");
                    sprintf (uart1_buf,"at+cmgd=1"); print_uart1();
putchar1(0x0d);
                    loop=1;
                    while(loop)
                    {
if (RI1)
{
RI1=0;
uart_data = getchar1();
if  (uart_data=='O')  loop=0;
                        else if  (uart_data=='K')  loop=0;
}
                    }
                    printf("Delete record\r\n");

                }

                if      (check_number==1)
                {
                    printf("\r\nData from save to lcd =  ");

                    for(i=0;i<index;i++)
                    {
```

```
                printf("%c",sms_buf[i]);
            }

            save_data(item);
            read_data_lcd(item);
            page_index = item;
            item++;
            writedataToeeprom(0,item);

            sprintf (uart1_buf,"at+cmgd=1\r\n"); print_uart1();
              loop=1;
              while(loop)
                    {
            if (RI1)
            {
  RI1=0;
  uart_data = getchar1();
  if  (uart_data=='O')  loop=0;
                              else if  (uart_data=='K')  loop=0;
            }
                    }

            printf("Delete record\r\n");
            delay(2000);
        }
    }
```

```
        }
    }
    /*****************************/
    /* Write Character To UART1 */
    /*****************************/
    char putchar1 (char c)                  //ส่งข้อมูลไปยัง Uart1
    {
     if (c == '\n')                         // If Line Feed(LF)
     {
      while (!TI1);                         //ส่งเสร็จหรือยัง
      TI1 = 0;
      SBUF1 = 0x0D;                         // รีจิสเตอร์ที่เก็บรายการส่ง
     }
     while (!TI1);                          //ส่งไปหรือยัง
     TI1 = 0;                               //ถ้าส่งเท่ากับ 0
     return (SBUF1 = c);
    }
    /*****************************/
    /* Get character From UART1 */
    /*****************************/
    char getchar1 ()                        //รับข้อมูลจาก Uart1
    {
     char c;
     while (!RI1);                          //รับเสร็จหรือยัง
     c = SBUF1;                             //รีจิสเตอร์ที่เก็บรายการรับ
     RI1 = 0;                               //ถ้ารับเท่ากับ 0
     return (c);
    }

    /*************************/
    /* Print String to UART1 */
    /*************************/
```

```
void print_uart1(void)                  //ประกาศส่งทีละประโยค
{
 char *p;                               // Pointer Buffer
 p = uart1_buf;                         // UART1 Buffer

 do                                     // Get char & Print Until null
 {
  putchar1(*p);                         // Write char to UART1
  p++;                                  // Next char
 }
 while(*p != '\0');                     // End of ASCII (null)

 return;
}
/*********************************/
/* Initial LCD 4-Bit Interface */
/*********************************/
void init_lcd(void)
{
 unsigned int i;                                // Delay Count

 LCD_E = 0;                                     // Start LCD Control (Disable)
 LCD_RS = 0;                                    // Default Instruction
 LCD_RW = 0;                                    // Default = Write Direction
 for (i=0;i<10000;i++);                         // Power-On Delay (15 mS)

 PORT_LCD &= 0x0F;                      // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= 0x30;                              // DB5:DB4 = 1:1
 enable_lcd();                                  // Enable Pulse
 for (i=0;i<2500;i++);                          // Delay 4.1mS

 PORT_LCD &= 0x0F;                                      // Clear old LCD Data (Bit[7..4])
```

```
    PORT_LCD |= 0x30;                   // DB5:DB4 = 1:1
    enable_lcd();                       // Enable Pulse
    for (i=0;i<100;i++);                // delay 100uS

    PORT_LCD &= 0x0F;                   // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x30;                   // DB5:DB4 = 1:1
    enable_lcd();                       // Enable Pulse
    while(busy_lcd());                  // Wait LCD Execute Complete

    PORT_LCD &= 0x0F;                   // Clear old LCD Data (Bit[7..4])
    PORT_LCD |= 0x20;                   // DB5:DB4 = 1:0
    enable_lcd();                       // Enable Pulse
    while(busy_lcd());                  // Wait LCD Execute Complete

    write_ins(0x28);                    // Function Set (DL=0 4-Bit,N=1 2 Line,F=0 5X7)
    write_ins(0x0C);                    // Display on/off Control (Entry Display,Cursor off,Cursor not Blink)
    write_ins(0x06);                    // Entry Mode Set (I/D=1 Increment,S=0 Cursor Shift)
    write_ins(0x01);                    // Clear Display  (Clear Display,Set DD RAM Address=0)
}

/*********************/
/* Set LCD Cursor */
/*********************/
void gotolcd(unsigned char i)
{
 i |= 0x80;                             // Set DD-RAM Address Command
 write_ins(i);
}
/*****************************/
/* Write Instruction to LCD */
/*****************************/
void write_ins(unsigned char i)
```

```
{
 LCD_RS  = 0;                  // Instruction Select
 LCD_RW  = 0;                  // Write Select

 PORT_LCD &= 0x0F;             // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= i & 0xF0;         // Strobe High Nibble Command
 enable_lcd();                 // Enable Pulse

 PORT_LCD &= 0x0F;             // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= (i << 4) & 0xF0;  // Strobe Low Nibble Command
 enable_lcd();                 // Enable Pulse

 while(busy_lcd());            // Wait LCD Execute Complete
}
/*****************************/
/* Write Data(ASCII) to LCD */
/*****************************/
void write_data(unsigned char i)
{
 LCD_RS  = 1;                  // Data Select
 LCD_RW  = 0;                  // Write Select

 PORT_LCD &= 0x0F;             // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= i & 0xF0;         // Strobe High Nibble Data
 enable_lcd();                 // Enable Pulse

 PORT_LCD &= 0x0F;             // Clear old LCD Data (Bit[7..4])
 PORT_LCD |= (i << 4) & 0xF0;  // Strobe Low Nibble Data
 enable_lcd();                 // Enable Pulse

 while(busy_lcd());            // Wait LCD Execute Complete
}
```

```
/***********************/
/* Enable Pulse to LCD */
/***********************/
void enable_lcd(void)                    // Enable Pulse
{
 unsigned int i;                         // Delay Count
 LCD_E = 1;                              // Enable ON
 for (i=0;i<500;i++);
 LCD_E = 0;                              // Enable OFF
}
/********************/
/* Wait LCD Ready */
/********************/
char busy_lcd(void)
{
 unsigned int i;                         // Dummy Byte Data

 LCD_RS = 0;                             // Instruction Select
 LCD_RW = 1;                             // Read Direction
 LCD_D7 = 1;                             // Prepared Read Busy
 LCD_E = 1;                              // Start Read Busy
 for (i=0;i<500;i++);                    // Wait LCD Ready

 if(LCD_D7 == 1)                         // Verify Busy Flag
 {
  LCD_E  = 0;                            // Disable Read
  LCD_RW = 0;                            // Default = Write Direction
  return 1;                              // LCD Busy Status
 }
 else
 {
  LCD_E = 0;                             // Disable Read
```

```
        LCD_RW = 0;             // Default = Write Direction
        return 0;               // LCD Ready Status
    }
}
/*****************************/
/* Print Data(ASCII) to LCD */
/*****************************/
void printlcd(void)
{
    char *p;

    p = lcdbuf;

    do                          // Get ASCII & Write to LCD Until null
    {
        write_data(*p);         // Write ASCII to LCD
        p++;                    // Next ASCII
    }
    while(*p != '\0');          // End of ASCII (null)

    return;
}
/*******************************************/
/* Long Delay Time Function(1..4294967295) */
/*******************************************/
void delay(unsigned long i)
{
        int j=0;
    while(i > 0) {i--;for(j=0;j<114;j++);}
                        // Loop Decrease Counter
    return;
}
```

```
/*************************************/
/* Delay For I2C Bus Device Interface */
/*************************************/
void delay_i2c(void)
{
 unsigned char i;
 i = 2;                    // Delay Counter
 while(i > 0) {i--;}       // Loop Decrease Counter
 return;
}


/*****************************/
/* Send Data 8 Bit to I2C Bus */
/*****************************/
void i2c_send_byte(unsigned char i)
{
 char j;                   // Bit Counter Send

 for(j = 0;j < 8;j++)      // 8-Bit Counter Send Data
 {
  if((i & 0x80) == 0x80)   // Send MSB First
    {sda = 1;}             // Send Data = 1
  else
    {sda = 0;}       // Send Data = 0

  scl = 1;           // Release SDA
  delay_i2c();

  scl = 0;           // Next Bit Send
  delay_i2c();
```

```
        i <<= 1;                // Shift Data For Send (MSB <- LSB)
    }

    return;
}

void ds1307_write_byte(unsigned char ds1307_address,unsigned char ds1307_data)
{
    sda = 0;                    // I2C Start Condition
    scl = 0;
    delay_i2c();

    i2c_send_byte(0xD0);        // Send ID Code DS1307,Write (1101000+0)
    sda = 1;                    // Release SDA
    scl = 1;                    // Start ACK Clock
    delay_i2c();
    while(sda) {}
    scl = 0;                    // End ACK Clock
    delay_i2c();

    i2c_send_byte(ds1307_address);      // Send DS1307 Address
    sda = 1;                    // Release SDA
    scl = 1;                    // Start ACK Clock
    delay_i2c();
    while(sda) {}
    scl = 0;                    // End ACK Clock
    delay_i2c();
    i2c_send_byte(ds1307_data);     // Send DS1307 Data
    sda = 1;                    // Release SDA
    scl = 1;                    // Start ACK Clock
    delay_i2c();
```

```
  while(sda) {}
  scl = 0;              // End ACK Clock
  delay_i2c();
  sda = 0;              // Stop Bit(End of Data)
  scl = 1;              // I2C Stop Condition
  delay_i2c();
  sda = 1;
  return;
}
/*******************************/
/* Get Data 8 Bit From I2C Bus */
/*******************************/
unsigned char i2c_get_byte(void)
{
 unsigned char i;       // Result Byte Buffer
 char j;                // Bit Counter Read Data

 for(j = 0; j < 8; j++) // 8-Bit Counter Read Data
 {
  i <<= 1;              // Shift Result Save (MSB <- LSB)

  sda = 1;              // Release Data
  scl = 1;              // Strobe Read SDA
  delay_i2c();

  if(sda == 1)
  {
   i |= 0x01;           // Save Bit Data = 1
  }
  else
  {
   i &= 0xFE;           // Save Bit Data = 0
```

```
    }

    scl = 0;            // Next Bit Read
    delay_i2c();
  }

  return i;
}

/*********************************/
/* Read Data 1-Byte From DS1307 */
/*********************************/
unsigned char ds1307_read_byte(unsigned char ds1307_address)
{
  unsigned char ds1307_data;    // Dummy Byte
  sda = 0;                      // I2C Stat condition
  scl = 0;
  delay_i2c();
  i2c_send_byte(0xD0);          // Send ID Code DS1307,Write (1101000+0)
  sda = 1;                      // Release SDA
  scl = 1;                      // Start ACK Clock
  delay_i2c();
  while(sda) {}
  scl = 0;                      // End ACK Clock
  delay_i2c();
  i2c_send_byte(ds1307_address);      // Send DS1307 Address
  sda = 1;                      // Release SDA
  scl = 1;                      // Start ACK Clock
  delay_i2c();
  while(sda) {}
  scl = 0;                      // End ACK Clock
```

```
    delay_i2c();
    scl = 1;                    // I2C Stop Condition
    delay_i2c();
    sda = 1;
    // New Start For Read //
    sda = 0;                    // I2C Stat condition
    scl = 0;
    delay_i2c();
    i2c_send_byte(0xD1);        // Send ID Code DS1307,Read (1101000+1)
    sda = 1;                    // Release SDA
    scl = 1;                    // Start ACK Clock
    delay_i2c();
    while(sda) {}
    scl = 0;                    // End ACK Clock
    delay_i2c();
    ds1307_data = i2c_get_byte();       // Read 1-Byte From DS1307
    sda = 1;                    // Send Stop Bit (End of Read Data)
    scl = 1;                    // Start Stop Bit Clock
    delay_i2c();
    scl = 0;                    // End Stop Bit Clock
    delay_i2c();

    scl = 1;                    // I2C Stop Condition
    delay_i2c();
    sda = 1;

    return ds1307_data;
}
```

ประวัติผู้เขียนโครงงาน

ชื่อ นางสาวศิริโฉม ทองอุบล

ภูมิลำเนา 93 หมู่ 5 ต.ท่าเยี่ยม อ.สากเหล็ก จ.พิจิตร 66160

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนบ้านวังอ้อ
- จบระดับมัธยมศึกษาจากโรงเรียนสากเหล็กวิทยา
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : a-engineer007@hotmail.com

ชื่อ นายสรายุทธ พันธ์วงค์

ภูมิลำเนา 89 หมู่ 2 ต.ห้วยเฮี้ย อ.นครไทย จ.พิษณุโลก 65120

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนบ้านป่าคาย
- จบระดับมัธยมศึกษาจากโรงเรียนนครบางยางพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : tomohogsk@hotmail.com

ชื่อ นายธงชัย ทุมมี

ภูมิลำเนา 48 หมู่ 7 ต.หนองปลาปาก อ.ศรีเชียงใหม่ จ. หนองคาย 43130

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนบ้านเสียว
- จบระดับมัธยมศึกษาจากโรงเรียนวรลาโภนุสรณ์
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : ee_48_ton@hotmail.com