ระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

Weather Measurement And Monitoring System Using GPRS

โดย

นาย วิริยังค์ พิพัฒน์สกุลโรจน์

นาย ศิวดล บุตะเขียว

นาย ศุภเกียรติ พันธุ์ไพโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

Weather Measurement And Monitoring System Using GPRS

โดย

นาย วิริยังค์ พิพัฒน์สกุลโรจน์ รหัส 48010834

นาย ศิวดล บุตะเขียว รหัส 48010898

นาย ศุภเกียรติ พันธุ์ไพโรจน์ รหัส 48010906

อาจารย์ที่ปรึกษา

ผศ. สุรพล บุญจันทร์

ผศ. นภัทร สระเอี่ยม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

Weather Measurement And Monitoring System Using GPRS

ผู้จัดทำ

1. นาย วิริยังค์ พิพัฒน์สกุลโรจน์ 48010834
2. นาย ศิวดล บุตะเขียว 48010898
3. นาย ศุภเกียรติ พันธุ์ไพโรจน์ 48010906

_____ อาจารย์ที่ปรึกษา

ผศ. สุรพล บุญจันทร์

_____ อาจารย์ที่ปรึกษา

ผศ. นภัทร สระเอี่ยม

ระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

Weather Measurement And Monitoring System Using GPRS

โดย นายวิริยังค์ พิพัฒน์สกุลโรจน์ 48010834

นายศิวดล บุตะเขียว 48010898

นายศุภเกียรติ พันธุ์ไพโรจน์ 48010906

อาจารย์ที่ปรึกษา ผศ. สุรพล บุญจันทร์

ผศ. นภัทร สระเอี่ยม

บทคัดย่อ

โครงงานนี้นำเสนอระบบตรวจวัดและติดตามผลข้อมูลอุตุนิยมวิทยา ซึ่งมีการวัดค่าของอุณหภูมิและความชื้นสัมพัทธ์ของสภาพภูมิอากาศโดยใช้เซนเซอร์วัดอุณหภูมิ และความชื้นสัมพัทธ์ที่ควบคุมด้วยไมโครคอนโทรลเลอร์ และมีการส่งข้อมูลที่ทำการวัดได้ผ่านระบบ General Packet Radio Service (GPRS) ด้วยโมดูลที่ประกอบด้วย TCP/IP stack และโทรศัพท์เคลื่อนที่ซึ่งควบคุมด้วย AT-Command เข้าสู่ Server เพื่อทำการบันทึกลงในฐานข้อมูลและแสดงผลที่หน้าจอคอมพิวเตอร์เป็นระยะๆ โดยข้อมูลดังกล่าวสามารถนำมาใช้ประโยชน์ในการวิเคราะห์ข้อมูลทางอุตุนิยมวิทยา

ABSRRACT

This project presents meteorological monitoring its results which consist of temperature measurement and absolute humidity measurement by using sensor regulated by microcontroller to calculate the results and delivering the information through General Packet Radio Service (GPRS) to the server by utilizing module which consist of TCP/IP stack and mobile phone module managed by AT-Command in order to record the information in database and display to the monitor continuously. These information can also benefit in the application of the analysis of meteorological datas.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

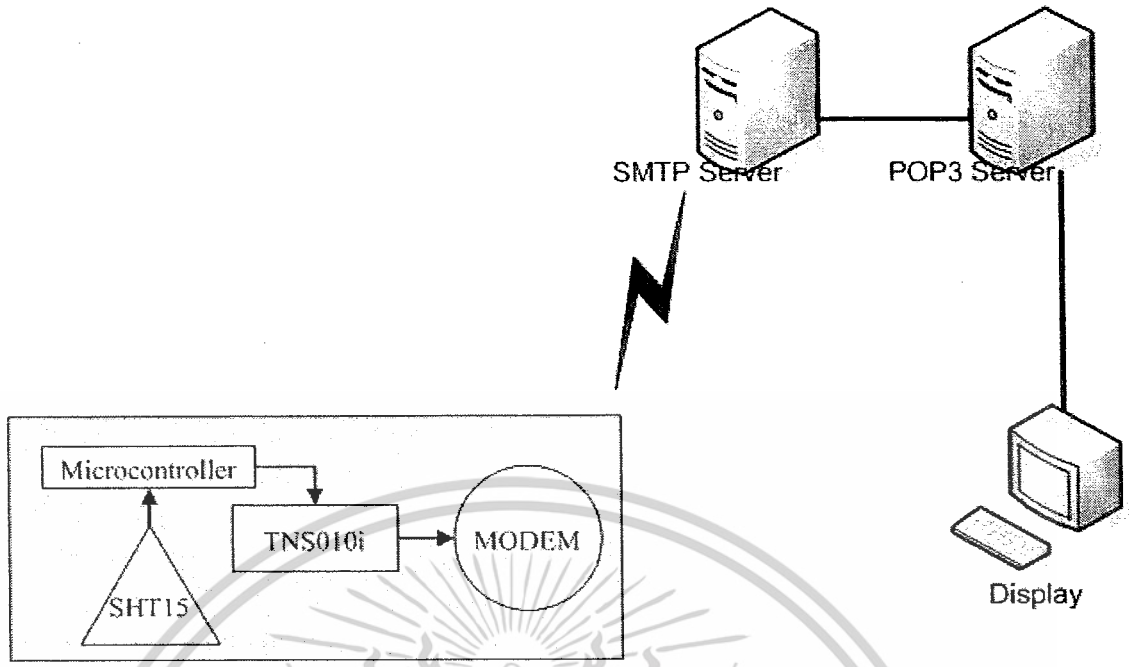
สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

สถานีตรวจวัดอากาศแบบเดิมที่ต้องใช้เจ้าหน้าที่ทำการจดบันทึกข้อมูลทางสภาพอากาศ มักจะประสบปัญหากับปัญหาข้อมูลไม่ครบถ้วนเนื่องจากจะให้คนมาเฝ้าจดบันทึกข้อมูลตลอดเวลา หรือถ้าจะผลัดเปลี่ยนกันก็ดูจะเกินความจำเป็นที่จะต้องลงทุนขนาดนี้ และอาจทำการบันทึกข้อมูลผิดพลาดซึ่งเกดจากความเผอเรอ นอกจากนี้ปริมาณข้อมูลที่จัดเก็บก็ไม่ได้ละเอียดเพียงต่อการที่จะนำมาใช้วิเคราะห์งานระดับลึกๆ ได้ และถ้าหากต้องการวัดสภาพอากาศหลายที่ก็ยิ่งเปลืองทรัพยากรและบุคลากรมากยิ่งขึ้น ทั้งกว่าจะได้ข้อมูลมาใช้งานต้องผ่านกระบวนการอีกหลายขั้นตอน สิ้นเปลืองเวลาและงบประมาณเป็นจำนวนมาก แต่ต่อมาได้มีกลุ่มคนพัฒนาชิ้นงานมาทำให้ความสามารถวัดข้อมูลสภาพอากาศโดยไม่ต้องมีคนไปประจำการตามที่ต่างๆ โดยใช้โมดูลตัวส่ง-ตัวรับแทน แต่จะมีข้อจำกัดในด้านระยะทางระหว่างสถานีรับ-ส่ง ซึ่งทางกลุ่มได้เห็นปัญหานี้จึงได้นำเทคโนโลยี GPRS เข้ามาใช้เป็นตัวกลางในการรับ-ส่งข้อมูล เนื่องจาก GPRS สามารถส่งได้กว้างขวางขึ้นอยู่กับระดับสัญญาณของเครือข่ายผู้ให้บริการโทรศัพท์เคลื่อนที่ ทางกลุ่มจึงได้จัดทำโครงงานระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS เพื่อเป็นแนวทางในการแก้ไขปัญหา

โครงงานชิ้นนี้ จะทำการอ่านค่าอุณหภูมิและความชื้นสัมพัทธ์ที่วัดจากเซนเซอร์ SHT15 แล้วนำค่าที่ได้ส่งไปที่ E-mail Server ด้วยการใช้ TCP/IP Stack Chip ทำหน้าที่ในการส่งผ่านระบบ GPRS โดยเราจะโทรศัพท์เคลื่อนที่ทำหน้าที่แทน modem จากนั้นทำการเขียนโปรแกรม E-mail Client ขึ้นมาเพื่อทำการอ่าน Subject ของ E-mail และเก็บค่าที่ได้ลงฐานข้อมูลแล้วนำมาแสดงผลในรูปของตารางและกราฟ แสดงแบบจำลองของระบบ ดังรูปที่ 1.1



รูปที่ 1.1 แบบจำลองโครงงานระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

โครงงานระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS เป็นการศึกษาการทำงานของไมโครคอนโทรลเลอร์ในเรื่องของการเชื่อมต่อกับอุปกรณ์ภายนอกแบบระบบบัส I2C ของไอซีฐานเวลาจริง DS1307 การส่งข้อมูลแบบอนุกรม ด้วยคำสั่ง AT-Command ของ TCP/IP Stack Chip และส่วนของการสร้างโปรแกรม E-mail Client และการแสดงผลในรูปของตารางและกราฟ

1.3 ขอบเขตการศึกษา

โครงงานระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS ที่นำเสนอนี้จะอาศัยหลักการทำงานที่ใช้ในการวัดค่าของอุณหภูมิ และความชื้นสัมพัทธ์ของสภาพภูมิอากาศที่ควบคุมด้วยไมโครคอนโทรลเลอร์ การส่งข้อมูลที่ทำการวัดได้ผ่านระบบ General Packet Radio Service (GPRS) ด้วยโมดูลที่ประกอบด้วย TCP/IP Stack Chip และโทรศัพท์เคลื่อนที่ที่ทำหน้าที่เป็น Modem ซึ่งควบคุมด้วย AT-Command และการอ่านค่าของอุณหภูมิ ความชื้นสัมพัทธ์ และเวลา จาก E-mail โดยโปรแกรม E-mail Client ที่เขียนขึ้นมาเองและบันทึกลงฐานข้อมูล แล้วทำการดึงข้อมูลจากฐานข้อมูลมาแสดงผลโดยภาษาที่ใช้ในโครงงานนี้คือ C#.NET

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 เทคโนโลยี GPRS

GPRS (General Packet Radio Service) บริการต่างๆที่ผ่านทาง Radio Interface ในระหว่างผู้ใช้ต้นทางและปลายทางซึ่งไม่ว่าจะเป็น Application Server หรือแม้แต่ตัวโทรศัพท์ เคลื่อนที่เองก็ตามจะถูกแปลงเป็น Packet ซึ่งมี IP Address กำกับอยู่ภายใน ซึ่งจะไม่เหมือนเดิมที่เคยใช้กัน เดิมที่ใช้กันคือระบบ Radio Frame ที่ใช้กันในการส่งข้อมูลเสียงพูดบนระบบ GSM

GPRS ไม่ได้เป็นลักษณะที่จะสามารถให้บริการได้ด้วยตัวของระบบเอง แต่ตัวมันเองเป็นเพียงแค่ Bearer ให้กับ Application ต่างๆ ที่ต้องการใช้ความเร็วที่เพิ่มมากกว่าปกติในระบบ GSM ที่เคยรองรับอยู่เดิมมาก่อน และระบบ GPRS จะต้องต่อไปยัง Packet Data Network ที่เป็น IP Network อีกต่อหนึ่ง

ดังนั้นผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ ที่จะเปิดให้ใช้ในระบบ GPRS ได้นั้นจะต้องทำการติดตั้งระบบเครือข่าย ที่ประกอบด้วยหน่วยหลักๆ 2 หน่วยด้วยกันคือ

- SGSN (Serving GPRS Supports Node)
- GGSN (Gateway GPRS Supports Node)

โดยทั้งสองหน่วยหลักขององค์ประกอบนี้จะถูกเชื่อมต่อเข้าด้วยกัน โดยมีอุปกรณ์อื่นๆที่เป็น ตัวช่วยเพื่อไปร่วมใช้ Radio Interface จาก Base Station โดยผ่านตัวควบคุมที่เรียกว่า PCU (Packet Control Unit) ที่ติดตั้งไว้ที่ BSC (Base Station Controller) ทั้งนี้อาจมองได้ว่า GPRS Network เป็นอีก Network หนึ่ง ซึ่งเข้าถึง Mobile Phone ผ่านทาง Radio Interface ของระบบ GSM Network เดิม โดยเป็นบริการที่เกี่ยวเนื่องกับการรับส่งข้อมูลเป็น Packet โดยตรง

ตามทฤษฎีแล้ว GPRS สามารถให้บริการที่ความเร็วสูงสุดถึง 171.2 kbps โดยต้องอาศัยการใช้ช่วงเวลา (timeslot) ทั้งแปดช่วงของทั้งหมดที่มี ซึ่งนั้นหมายถึงความเร็วสูงสุดที่สูงขึ้นถึงสามเท่าของการส่งข้อมูลผ่านสาย บนเครือข่ายโทรศัพท์ปัจจุบัน และสูงขึ้นมากกว่าการเชื่อมต่อแบบ CSD ในเครือข่าย GSM ถึงสิบเท่า

GPRS ยังรองรับการให้บริการในรูปแบบใหม่ที่ไม่สามารถให้บริการได้บนเครือข่าย GSM เดิม เพราะข้อจำกัดด้านความเร็วในการรับส่งข้อมูลในแบบ CSD (9.6 kbps) และข้อจำกัดของขนาดของข้อมูลที่สามารถรับส่งได้ในแบบ SMS (160 ตัวอักษร GPRS) ทำให้สามารถให้บริการในรูปแบบต่างๆ ที่ไม่เคยมีมาก่อนบนเครือข่ายโทรศัพท์เคลื่อนที่

2.1.1 รูปแบบการให้บริการของ GPRS

- Textual And Visual Information บริการนี้เป็นจุดแตกต่างอย่างแรกที่ GPRS เหนือกว่า GSM ทั่วไป โดยสามารถส่งข้อมูลทีเป็นตัวอักษร หรือรูปภาพไปยังโทรศัพท์มือถือ ได้อย่างสะดวกรวดเร็ว ซึ่งจะทำให้ GPRS แทรกซึมเข้าสู่การใช้งานของคนทั่วไปได้ทั้งข่าวความเคลื่อนไหว ข้อมูลที่คนส่วนใหญ่สนใจ รวมทั้งบริการต่างๆ ที่จะเสริมเข้ามาในอนาคต

- Still Images เป็นการส่งภาพนิ่งความละเอียดสูงไปมาระหว่างเครื่องด้วยกันได้ ทำให้สามารถส่งผ่านความรู้สึกดีๆ ผ่านภาพถ่ายหรือการ์ดอวยพรได้เลย รวมทั้งภาพที่ถ่ายได้จากกล้องดิจิตอล ก็สามารถโอนแล้วส่งต่อไปได้ทันที

- Moving Images นอกเหนือจากภาพนิ่งแล้วภาพเคลื่อนไหวก็สามารถส่งต่อกันไปได้เช่นกัน เช่น การประชุมทางไกล หรือ การส่งภาพจากกล้องวงจรปิดไปยัง โทรศัพท์มือถือในกรณีประยุกต์ใช้กับระบบรักษาความปลอดภัย

- Chat เป็นคุณสมบัติที่คงจะถูกใจของผู้รักการคุยแบบไม่ใช้เสียงซึ่งสามารถสนทนากันได้ทั้งแบบเป็นคู่ หรือเป็นกลุ่มได้อย่างสบายใจ ซึ่งจุดเด่นที่สำหรับ สามารถ Chat ได้ทุกที่ที่อยากจะ Chat

- Web Browsing เป็นการเข้าสู่ World Wide Web ด้วยการใช้โทรศัพท์มือถือ ซึ่งความเร็วมีให้เลือกตั้งแต่ 56 Kbps ไปจนถึง 112 Kbps การท่องเว็บจึงไม่ใช่เรื่องยากอีกต่อไป แม้รูปแบบการแสดงผลจะแตกต่างจากการท่องเว็บโดยใช้เครื่องคอมพิวเตอร์อยู่บ้าง

- E-Mail เป็นบริการพื้นฐานที่มีคนนิยมใช้งานมากที่สุดสำหรับการส่งข้อความ โดยจะมีการใช้ในรูปของ SMS (Shorts Message Service)

- File Transfer เป็นบริการโอนถ่ายไฟล์ข้อมูลซึ่งน่าจะใช้งานกันอย่างแพร่หลายขึ้น GPRSเพราะความเร็วดูจะเหนือกว่าการใช้งานผ่านโมเด็ม กับโทรศัพท์พื้นฐานที่เราใช้กันอยู่ในปัจจุบันมาก โดยจะรองรับกับโปรโตคอล FTP และแอพพลิเคชั่นที่อ่านข้อความอย่าง Acrobat Reader

- Audio บริการด้านเสียงของ GPRS จะเหนือกว่าโทรศัพท์มือถือเดิมๆ ที่เรารู้จัก เนื่องจากความคมชัดของสัญญาณเสียงที่เหนือกว่า และยังประยุกต์ใช้ในการเก็บไฟล์เสียงเพื่อนำไปใช้งานในด้านต่างๆ ด้วย เช่น การวิเคราะห์รายละเอียดของเสียงในงานของตำรวจ เป็นต้น

- Remote LAN Access เราสามารถเข้าถึงเครือข่ายคอมพิวเตอร์โดยใช้โทรศัพท์มือถือแทนเบอร์โทรศัพท์กับเครื่องคอมพิวเตอร์ที่บ้านได้อย่างง่ายดาย ซึ่งความเร็วในการส่งถ่ายข้อมูลจะเหนือกว่าโทรศัพท์พื้นฐานทั่วไป

- Vehicle Positioning เป็นความสามารถในการบอกตำแหน่งของยานพาหนะที่เราใช้อยู่ โดยจะสามารถเชื่อมต่อกับดาวเทียม ซึ่งจะสามารถบอกตำแหน่งที่เราอยู่ โดยอ้างอิงกับเครื่องโทรศัพท์มือถือได้อย่างแม่นยำ

2.2 AT-COMMAND

AT Command เป็น protocol ที่ใช้โทรศัพท์เคลื่อนที่ส่วนใหญ่ยึดใช้เป็น protocol หลักในการ interface กับ คอมพิวเตอร์ ซึ่ง AT Command เป็นชุดคำสั่งที่มีไว้สำหรับ modem โดยเฉพาะ แสดงดังตารางที่ 2.1 โดยคอมพิวเตอร์จะมองโทรศัพท์เคลื่อนที่เป็นโมเด็มเหมือนกัน โดยใช้ชุดคำสั่งพื้นฐาน จะถูกกำหนดไว้ในHayes AT Command ซึ่งบริษัท Hayes ได้เป็นผู้คิดค้นชุดคำสั่งนี้ขึ้นมาเพื่อใช้กับโมเด็ม

ของตนเอง และต่อมาได้กลายเป็นมาตรฐานสำหรับผู้ผลิตรายอื่นๆ โดยอาจจะมีชุดคำสั่งขยาย เพื่อใช้เป็นการเฉพาะสำหรับผู้ผลิตรายนั้นก็ได้

การเชื่อมด้านฮาร์ดแวร์นั้นจะนิยมใช้ 2 ช่องทาง คือ

- Serial Port เป็นพอร์ตสำหรับต่อกับอุปกรณ์อินพุตและเอาต์พุต ใช้สำหรับเป็นช่องทางการติดต่อโมเด็มและอุปกรณ์เสริมต่าง ๆ ซึ่งจะมีอยู่ 2 พอร์ต คือ พอร์ตคอม 1 และ พอร์ตคอม 2 สามารถต่อความยาวได้ 6 เมตร

- IrDA (Infrared) การต่อกับคอมพิวเตอร์นั้น OS จะทำการ map port ไปยัง Port COM 3 หรือ COM 4 เพื่อความสะดวกในการใช้งานโดย service ชื่อว่า Virtual COM Port และ IrCOMM

ตารางที่ 2.1 แสดงคำสั่ง AT Command พื้นฐาน

คำสั่ง	หน้าที่
AT	ตรวจเช็คความพร้อมของโทรศัพท์เคลื่อนที่
ATD phone number	สั่งให้โทรศัพท์ต่อไปยัง phone number
ATH	วางสายโทรศัพท์
ATA	รับสาย

2.3 การสื่อสารข้อมูลของไมโครคอนโทรลเลอร์

การสื่อสารข้อมูล คือ ขบวนการในการแลกเปลี่ยนข้อมูลหรือข่าวสาร ซึ่งประกอบด้วยผู้ส่ง (Transmitter) ผู้รับ (Receiver) และตัวกลางในการส่งข้อมูล (Transmission Medium) โดยที่ข้อมูลที่ทำการสื่อสารกันจะอยู่ในรูปของสัญญาณดิจิตอล (Digital) คืออยู่ในรูปของเลขฐานสอง ในรูปรหัสตัวอักษร ตัวเลข หรือเครื่องหมายรหัส ASCII (American Standard Code For Information Interchange)

2.3.1 ประเภทของการสื่อสารข้อมูล (Data Communication) แบ่งออกเป็น 3 ชนิด

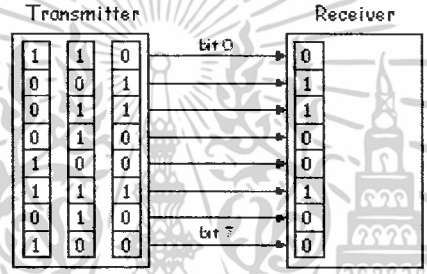
- การสื่อสารแบบทางเดียว (One-Way Transmission หรือ Simplex) การสื่อสารแบบนี้ในเวลาเดียวกันจะสามารถส่งข้อมูลได้เพียงทางเดียวเท่านั้น ถึงแม้ว่าตัวส่งจะมีช่องสัญญาณเหลือก็ตาม ซึ่งมักจะเรียกการสื่อสารแบบทางเดียวนี้ว่า "ซิมเพล็กซ์" ผู้ส่งข้อมูลจะสามารถส่งได้ทางเดียวโดยที่ผู้รับจะไม่สามารถโต้ตอบได้ เช่น การส่งวิทยุกระจายเสียง การแพร่ภาพโทรทัศน์

- การสื่อสารแบบกึ่งทางคู่ (Half-Duplex หรือ Either-Way) การสื่อสารแบบนี้เมื่อผู้ส่งได้ทำการส่งข้อมูลไปแล้ว ผู้รับก็จะรับข้อมูลนั้น หลังจากนั้น ผู้รับก็สามารถปรับมาเป็นผู้ส่งข้อมูลแทน ส่วนผู้ส่งเดิมก็ปรับมาเป็นผู้รับแทนสลับกันได้ แต่ไม่สามารถส่งสัญญาณพร้อมกันในเวลาเดียวกันได้ จึงเรียกการส่งสัญญาณแบบนี้ว่า "ฮาร์ฟดูเพล็กซ์" ได้แก่ วิทยุสนามที่ตำรวจใช้ เป็นต้น

- การสื่อสารแบบทางคู่ (Full-Duplex หรือ Both way Transmission) การสื่อสารแบบนี้สามารถส่งข้อมูลได้พร้อมกันทั้งสองทางในเวลาเดียวกัน เช่น การใช้โทรศัพท์ผู้ใช้สามารถพูดสายโทรศัพท์ได้พร้อม ๆ กัน

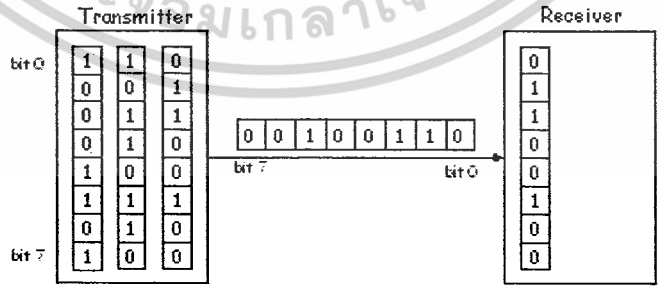
2.3.2 ประเภทของการส่งผ่านข้อมูล (Transmission modes)

- การส่งผ่านข้อมูลแบบขนาน (Parallel Transmission) ข้อมูลขนาด 8 บิตจะถูกส่งออกจากอุปกรณ์ไปยังอุปกรณ์รับพร้อมๆ กัน และช่องสัญญาณที่ใช้ในการรับส่งจะต้องมีอย่างน้อย 8 ช่องสัญญาณ ดังรูปที่ 2.1 ข้อดีคือสามารถรับส่งข้อมูลได้รวดเร็วและเป็นจำนวนมาก ข้อเสียคือไม่เหมาะจะนำไปใช้ในการส่งข้อมูลระยะไกล เนื่องจากสายนำสัญญาณมีราคาแพงและการส่งข้อมูลอาจเกิดความผิดพลาด



รูปที่ 2.1 แสดงการส่งผ่านข้อมูลแบบขนาน

- การส่งผ่านข้อมูลแบบอนุกรม (Serial Transmission) จะส่งข้อมูลออกจากพอร์ต (Port) เรียงกันออกไปทีละบิต และด้านรับจะรับข้อมูลเข้ามาทีละบิต และตรวจสอบบิตที่รับเข้ามาว่าบิตใดเป็นเริ่มต้น และบิตสิ้นสุดการตรวจสอบขึ้นอยู่ กับรูปแบบของรหัสของบิตที่ใช้การส่งผ่านข้อมูลแบบอนุกรมดังรูปที่ 2.2



รูปที่ 2.2 แสดงการส่งผ่านข้อมูลแบบอนุกรม

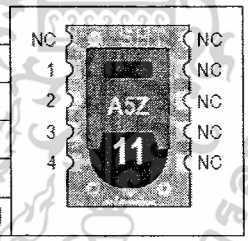
2.4 เซนเซอร์วัดความชื้นสัมพัทธ์และอุณหภูมิ

2.4.1 คุณสมบัติของเซนเซอร์SHT-15

เป็นเซนเซอร์วัดความชื้นสัมพัทธ์และอุณหภูมิซึ่งผลิตจากบริษัท Sensirion แสดงได้ดังรูปที่ 2.3 มีการติดต่อ และส่งข้อมูลด้วยระบบบัส I^2C มีขนาดเล็กจึงสะดวกต่อการใช้งานและการติดตั้งลงบนแผ่นวงจร เพื่อใช้ประยุกต์ในการใช้งานจริงในรูปแบบต่างๆ โดยมีคุณสมบัติด้านเทคนิคดังนี้

- ทำหน้าที่วัดได้ทั้งค่าอุณหภูมิและความชื้นสัมพัทธ์ได้ในตัวถังเดียวกัน
- สามารถวัดความชื้นสัมพัทธ์ได้ในช่วง 0-100% และอุณหภูมิช่วง –40 ถึง120 องศาเซลเซียส
- สามารถกำหนดความละเอียดของย่านการวัดได้
- มีขนาดเล็กและกินพลังงานต่ำ
- ทำงานในย่านแรงดันไฟเลี้ยง +2.4 ถึง +5.5 V
- เสถียรภาพในการทำงานสูง

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial Data, bidirectional
3	SCK	Serial Clock, input only
4	VDD	Source Voltage
NC	NC	Must be left unconnected



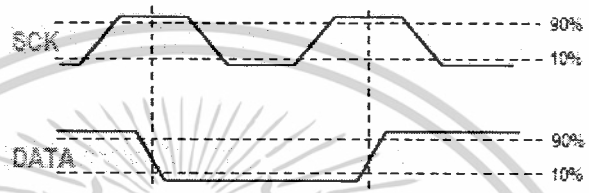
รูปที่ 2.3 แสดงรูปร่างเซนเซอร์ SHT-15 และการจัดขาเพื่อต่อใช้งาน

การสื่อสารในการรับส่งข้อมูลของเซนเซอร์ใช้ระบบบัสบัส I^2C โดยมีขาสัญญาณ 2 ขา ดังนี้

- ขาสัญญาณนาฬิกา (SCK) ทำหน้าที่รับสัญญาณนาฬิกา เพื่อกำหนดจังหวะในการสื่อสารข้อมูล
- ขาสัญญาณรับส่งข้อมูล (DATA) เป็นขาสัญญาณสำหรับรับส่งข้อมูลในการใช้งานควรต่อความต้านทาน 10k พูลอัพที่ขานี้

2.4.2 รูปแบบการสื่อสารข้อมูลของ SHT-15

การส่งคำสั่ง (Sending a Command) ในสภาวะเริ่มต้นก่อนการส่งข้อมูลจากไมโครคอนโทรลเลอร์ไปยัง SHT-15 จำเป็นต้องสร้างรูปแบบสัญญาณกระตุ้นผ่านขาสัญญาณ SCK และ DATA ดังรูปที่ 2.4 เพื่อให้ตรงกับเงื่อนไขที่เรียกว่า Transmission start หรือสภาวะเริ่มต้นการส่งสัญญาณ นั่นคือขา DATA ต้องถูกทำให้เป็นลอจิก "0" นานอย่างน้อย 1 ไซเคิลของสัญญาณนาฬิกา SCK หลังจากนั้น SHT-15 จะทราบทันทีว่า ข้อมูลต่อจากนี้เป็นคำสั่ง



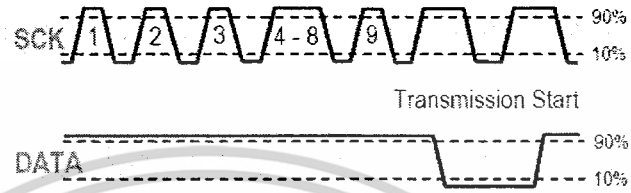
รูปที่ 2.4 แสดงรูปแบบสัญญาณกระตุ้นผ่านขาสัญญาณ SCK และ DATA (Transmission start)

หลังจากสร้างเงื่อนไข Transmission start แล้ว สามารถส่งคำสั่งไปยัง SHT-15 เพื่อกำหนดการทำงานได้ทันที โดยข้อมูลคำสั่งต่างๆสำหรับการทำงานตามตารางที่ 2.2

ตารางที่ 2.2 แสดงรายละเอียดคำสั่งและข้อมูลสำหรับควบคุมการทำงานของเซนเซอร์ SHT-15

คำสั่ง	ข้อมูลคำสั่ง
สงวนไว้	0000x
อ่านค่าอุณหภูมิ (Measure Temperature)	000011
อ่านค่าความชื้นสัมพัทธ์ (Measure Humidity)	00101
อ่านค่ารีจิสเตอร์กำหนดสถานะ (Read Status Register)	00111
สงวนไว้	0101x ถึง 1110x
รีเซตการทำงาน (Soft reset) ทำให้รีจิสเตอร์กำหนดสถานะกลับไปสู่ค่า Default และต้องใช้เวลาในการทำงานอย่างน้อย 11 มิลลิวินาที จึงสมารถรับคำสั่งถัดไปได้	11110

- รีเซตการเชื่อมต่อ (Connection reset sequence) เมื่อต้องการเริ่มต้นการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์ SHT-15 ต้องสร้างสัญญาณรีเซตขึ้นมาก่อน โดยทำให้ขา DATA มีสถานะลอจิกเป็น "1" นานเท่ากับช่วงเวลาที่ป้อนสัญญาณนาฬิกาที่ขา SCK จำนวน 9 ลูกติดต่อกัน ดังรูปที่ 2.5 แล้วตามด้วยการสร้างสภาวะเริ่มต้นการส่งสัญญาณ



รูปที่ 2.5 แสดงสัญญาณรีเซตและการสร้างสภาวะเริ่มต้นการส่งสัญญาณ

- ขั้นตอนการอ่านอุณหภูมิและความชื้นสัมพัทธ์ การอ่านข้อมูลดิบของอุณหภูมิหรือความชื้นสัมพัทธ์นั้น ทำได้ภายหลังจากที่สภาวะเริ่มต้นที่เรียกว่า Transmission start แล้วตามด้วยการส่งข้อมูลคำสั่งอ่านอุณหภูมิหรือความชื้นสัมพัทธ์อย่างใดอย่างหนึ่งไปยัง SHT-15 เซนเซอร์ SHT-15 ต้องใช้เวลาในการประมวลผลเพื่อให้ได้ผลลัพธ์ที่ต้องการ ซึ่งใช้เวลามากหรือน้อยขึ้นอยู่กับความละเอียดของข้อมูลที่ต้องการแสดงในตารางที่ 2.3

ตารางที่ 2.3 แสดงค่าเวลาที่เซนเซอร์ SHT-15 ต้องใช้เวลาในการประมวลผลข้อมูล

ความละเอียดของข้อมูลที่ประมวลผล	เวลาที่เซนเซอร์ SHT-15 ใช้ในการประมวลผล (ผิดพลาด 15%)
14 บิต	210 มิลลิวินาที
12 บิต	55 มิลลิวินาที
8 บิต	11 มิลลิวินาที

2.4.3 การคำนวณค่าอุณหภูมิ

ในการอ่านค่าอุณหภูมิจากเซนเซอร์ SHT-15 ผู้พัฒนาสามารถเลือกความละเอียดในการอ่านได้ในแบบ 14 บิต หรือ 12 บิต โดยที่ความละเอียด บิตเป็นค่าตั้งต้น โดยที่ผู้พัฒนาจำเป็นที่จะต้องอ่านข้อมูลดิบจากเซนเซอร์ SHT-15 เข้ามาก่อน จากนั้นจึงใช้กระบวนการทางคณิตศาสตร์

เพื่อให้ได้ค่าอุณหภูมิออกมา โดยสามารถคำนวณได้จากสมการที่กำหนดมาจาก Sensirion ผู้ผลิตเซนเซอร์ SHT-15 ดังสมการที่ 1

$$\text{Temperature} = d_1 + (d_2 * SO_T) \quad (1)$$

โดยที่ Temperature คือค่าอุณหภูมิจริง d_1 คือค่าคงที่ ขึ้นอยู่กับไฟเลี้ยงที่ป้อนให้กับขา VDD ของ SHT-15 d_2 คือค่าคงที่ขึ้นอยู่กับความละเอียดของอุณหภูมิที่ต้องการจาก SHT-15 SO_T คือค่าอุณหภูมิที่อ่านได้จากเซนเซอร์ SHT-15 โดยค่า d_1 แสดงดังตารางที่ 2.4 และค่า d_2 แสดงดังตารางที่ 2.5

ตารางที่ 2.4 แสดงค่า d_1 เพื่อคำนวณค่าอุณหภูมิที่วัดได้

ไฟเลี้ยง	ค่าคงที่ทางอุณหภูมิตัวที่ 1(d_1)	
	ในหน่วย C	ในหน่วย F
+5V	-40.00	-40.00
+4V	-39.75	-39.50
+3.5V	-39.66	-39.35
+3V	-39.60	-39.28
+2.5V	-39.55	-39.23

ตารางที่ 2.5 แสดงค่า d_2 เพื่อคำนวณค่าอุณหภูมิจริงที่วัดได้

ความละเอียด	ค่าคงที่ทางอุณหภูมิตัวที่ 2 (d_2)	
	ในหน่วย C	ในหน่วย F
14 บิต	0.01	0.018
12 บิต	0.04	0.072

2.4.4 คำนวณค่าความชื้นสัมพัทธ์

คำนวณค่าความชื้นสัมพัทธ์การอ่านค่าความชื้นสัมพัทธ์จากเซนเซอร์ SHT-15 สามารถเลือกความละเอียดในการอ่านได้ในแบบ 12 บิตหรือ 8 บิต โดยในที่นี้ผู้พัฒนาจะใช้ที่ความละเอียด 12 บิต ซึ่งจะเป็นข้อมูลดิบมาเมื่อผ่านการคำนวณทางคณิตศาสตร์ แล้วจะได้ค่าความชื้นสัมพัทธ์ออกมาโดยสมการที่ 2, 3

$$RH_{true}=(T-25)*(t_1+(t_2*SO_{RH}))+RH_{linear} \quad (2)$$

$$RH_{linear}=c_1+(c_2*SO_{RH})+(c_3*(SO_{RH})^2) \quad (3)$$

โดยที่ RH_{true} คือ ค่าความจริงสัมพัทธ์ค่าจริง

T คือ ค่าอุณหภูมิจริงที่คำนวณได้จากสมการที่ 1

t_1 และ t_2 คือ ค่าคงที่โดยมีค่าขึ้นอยู่กับความละเอียดของค่าความชื้นสัมพัทธ์ที่ต้องการจากเซนเซอร์ SHT-15 แสดงดังตารางที่ 2.5

c_1, c_2 และ c_3 คือค่าคงที่โดยมีค่าขึ้นอยู่กับความละเอียดของค่าความชื้นสัมพัทธ์ที่ต้องการจากเซนเซอร์ SHT-15 แสดงดังตารางที่ 2.6

SO_{RH} คือ ค่าความชื้นสัมพัทธ์ดิบที่อ่านได้จาก SHT-15

ตารางที่ 2.6 แสดงค่าคงที่ t_1, t_2 ที่ความละเอียดต่างๆ

ความละเอียด	ค่าคงที่	
	t_1	t_2
14 บิต	0.01	0.00008
12 บิต	0.01	0.00128

ตารางที่ 2.7 แสดงค่าคงที่ c1, c2, c3 ที่ความละเอียดต่างๆ

ความละเอียด	ค่าคงที่		
	c_1	c_2	c_3
12 บิต	-4	0.648	-2.8×10^{-6}
8 บิต	-4	0.0405	-7.2×10^{-4}

2.5 การเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกกับระบบบัส I^2C

2.5.1 ความรู้เบื้องต้นเกี่ยวกับ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัวจะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรมหรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock line) ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองว่า สาย SDA และ SCL

2.5.2 คุณสมบัติโดยทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุต ของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็กเตอร์เปิด (open-collector)

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่าคือ 7 บิต (7-bit addressing) หรือ 10 บิต (10-bit addressing)

2.5.3 หลักการของบัส I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะขออธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่อยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่ จะอธิบายการทำงานของบัส I^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (Transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (Receiver)

ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการทำงานหรือการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

- การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

- ในระหว่างการถ่ายทอดข้อมูลเมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูงสายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

2.5.4 สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สภาวะ ดังนี้

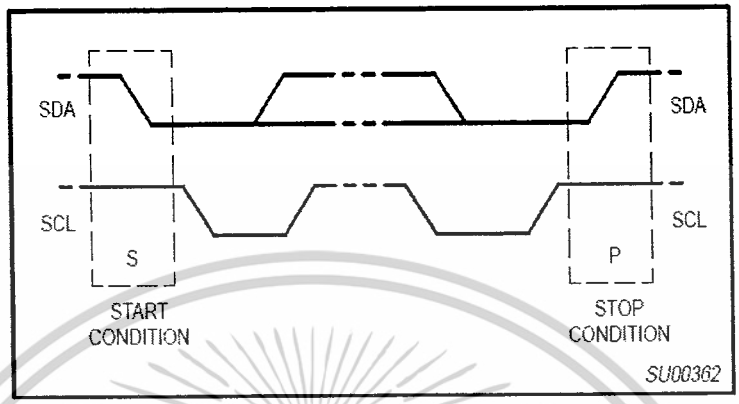
- บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

- เริ่มต้นการถ่ายทอดข้อมูล (start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START) ดังรูปที่ 2.6

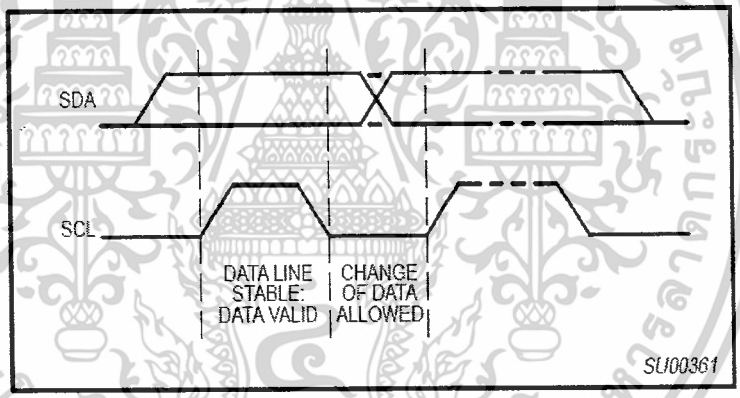
- ข้อมูลดำรงอยู่บนบัส (data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น ดังรูปที่ 2.7

- รับรู้ข้อมูล (acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลใน แต่ละไบต์เรียบร้อยแล้ว ดังรูปที่ 2.7

- หยุดการถ่ายทอดข้อมูล (stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP) ดังรูปที่ 2.6



รูปที่ 2.6 แสดงสภาวะเริ่มและหยุด



รูปที่ 2.7 แสดงสภาวะการส่งข้อมูล

2.5.5 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

ดังนั้นหัวใจสำคัญในอันดับแรกของกาทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดของการอ้างถึงทั้ง 2 รูปแบบ

2.5.6 การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ โดยมีรูปแบบแสดงในรูปที่ 3-6 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น "0" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

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

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (Data)

หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

2.5.7 การอ้างถึงแบบ 10 บิต

ในการอ้างถึงแบบนี้ ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกับแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิต ถัดมาเป็นบิตแอดเดรส ของอุปกรณ์ ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรก ยังคงเป็น การกำหนดว่า ต้องการอ่าน หรือเขียนข้อมูล กับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 3-8 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ 10 บิต

2.5.8 อุปกรณ์ที่ใช้การเชื่อมต่อแบบบัส I^2C

ในปัจจุบัน บัส I^2C ได้รับความนิยมเพิ่มมากขึ้นเรื่อยๆ ด้วยข้อดีที่ชัดเจนคือ ใช้สายสัญญาณเพียง 2 เส้นเท่านั้น และการขยายระบบไมโครคอนโทรลเลอร์ที่มีจำนวนอินพุตเอาต์พุตและหน่วยความจำกัดจำกัดสามารถทำได้ง่ายขึ้นด้วยระบบบัส I^2C เมื่อเป็นเช่นนี้จึงมี

อุปกรณ์เพอริเฟอรัลที่ใช้การเชื่อมต่อแบบ I^2C มากมายจากลหายผู้ผลิตออกมาให้ได้งานกัน ดังมีตัวอย่างต่อไปนี้

ไอซีขยายพอร์ตอินพุตเอาต์พุต (I/O expander) : PCF8574, PCF8582, PCF8584

ไอซีหน่วยความจำอีอีพรอมอนุกรม (Serial EEPROM) : 24Cxx, PCF8570, PCF72/73, PCF8582

ไอซี ADC/DAC: PCF8591

ไอซีรีลไทม์คล็อก (Real-time clock: RTC): PCF8583, PCF8593, PCF8598, 41T56C

ไอซีขับ LCD โมดูล (LCD driver): PCF8466, PCF8576, PCF8577/78, PCF8579, SAA1064

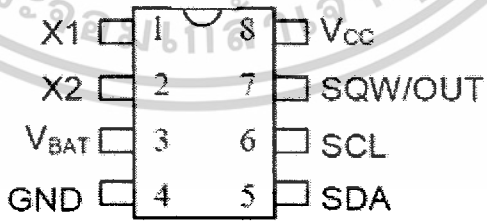
ไอซีกำเนิดสัญญาณ DTMF (DTMF generator): PCD3311/12

2.5.9 การใช้งาน RTC (Real Time Clock) ด้วย DS1307

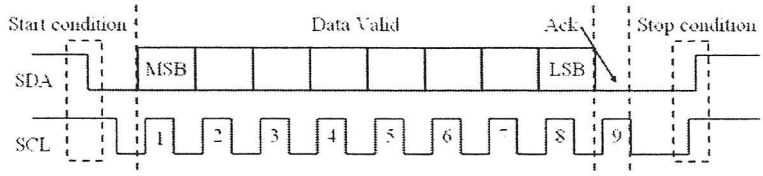
ระบบฐานเวลา เป็นสิ่งสำคัญที่สามารถนำไปใช้ในอุปกรณ์อีเล็กโทรนิกส์ได้หลากหลาย ภายในไมโครคอนโทรเลอร์เองก็มีไทเมอร์เพื่อใช้ในการจับเวลา หรือนำไปใช้เป็นฐานเวลาจริงได้เช่นกัน แต่เนื่องจากไมโครคอนโทรเลอร์สามารถทำงานได้ต่อเมื่อมีไฟเลี้ยงเท่านั้น ดังนั้นการใช้ไทเมอร์ของไมโครคอนโทรลเลอร์ สร้างฐานเวลาจริงจึงไม่เหมาะสมในบางแอปพลิเคชัน

DS1307 เป็น IC ฐานเวลาของดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีบัสรับส่งข้อมูลแบบ I2C ซึ่งเป็นแบบ 2 wire สามารถสื่อสารได้ 2 ทิศทาง (bi-direction bus) ฐานเวลาของ DS1307 นั้นสามารถเก็บข้อมูล วินาที นาที ชั่วโมง วัน วันที่ เดือน และปี ได้

ระบบเวลาสามารถทำงานโหมดรูปแบบ 24 ชั่วโมง หรือ 12 ชั่วโมง AM/PM ก็ได้ ภายมีระบบตรวจจับแหล่งจ่ายไฟ โดยถ้าแหล่งจ่ายไฟหลักถูกตัดไป DS1307 สามารถสวิตซ์ไปใช้ไฟจากแบตเตอรี่ และทำงานต่อไป โดยที่ยังสามารถรักษาข้อมูลไว้ได้ โครงสร้างมีขาทั้งหมด 8 ขา ดังแสดงในรูปที่ 2.8 และมีรายละเอียดการทำงานของขาต่าง ๆ ดังนี้

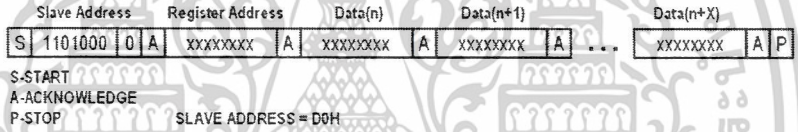


รูปที่ 2.8 ตำแหน่งขาไอซี RTC DS1307



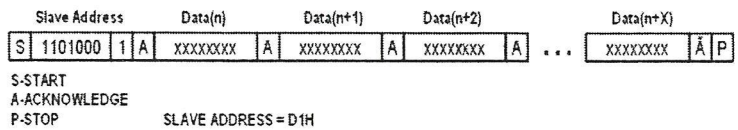
รูปที่ 2.9 การรับส่งข้อมูลผ่านบัส I2C

ในการรับส่งข้อมูลผ่านบัส I2C อุปกรณ์ Master จะเป็นผู้สร้างสัญญาณ Clock บน SDA และเป็นตัวควบคุมสถานะ Start และ Stop เพื่อควบคุมการรับส่งข้อมูลทั้งหมดการส่งข้อมูลไปยังอุปกรณ์ DS1307 ดังแสดงในรูปที่ 2.9 ไมโครคอนโทรลเลอร์ต้องสร้างสภาวะ Start ก่อน จากนั้นต้องส่ง Address ของDS1307 ขนาด 7 บิตซึ่งมีค่าเป็น 1101000 และตามด้วยบิตระบุทิศทางของข้อมูล ในกรณีที่เป็นการเขียนข้อมูลลง DS1307 จะต้องเป็น “0” ดังรูปที่ 2.10 จากนั้นไมโครคอนโทรเลอร์ จะต้องส่งตำแหน่ง Address ภายในรีจิสเตอร์ของ DS1307 ที่ต้องการเขียนข้อมูลลง แล้วจึงค่อย เขียนข้อมูลลง โดยในการส่งข้อมูลแต่ละไบต์จะต้องรอบิต Ack จาก DS1307 ทุกไบต์ เมื่อส่งจน ครบแล้ว ถึงจะสร้างสภาวะ Stop เพื่อกลับสู่สถานะว่าง



รูปที่ 2.10 การเขียนข้อมูลอุปกรณ์ Slave ผ่านบัส I2C

การรับข้อมูลจากอุปกรณ์ Slave ดังแสดงในรูปที่ 2.11 เริ่มแรกไมโครคอนโทรลเลอร์ต้องสร้างสภาวะ Start ก่อน จากนั้นต้องส่ง Address ของ DS1307 ขนาด 7 บิตซึ่งมีค่าเป็น 1101000 และ ตามด้วยบิตระทิศทางของข้อมูล ในกรณีที่เป็นการอ่านข้อมูลจาก DS1307 จะต้องเป็น "1" จากนั้นจึงค่อยรับข้อมูลจากอุปกรณ์ Slave ทีละไบต์ โดยตำแหน่งที่อ่านเข้ามาจะขึ้นอยู่กับตำแหน่งรีจิสเตอร์พอยท์เตอร์ ซึ่งจะเป็นตำแหน่งท้ายสุดที่ได้ทำการเขียนข้อมูลไว้ เมื่ออ่านข้อมูลครบแต่ละไบต์อุปกรณ์ Master ต้องส่ง Acknowledge บิตกลับไปให้อุปกรณ์ Slave ด้วย ในกรณีที่เป็นไบต์สุดท้าย อุปกรณ์ Master ต้องส่ง “not acknowledge” กลับไป



รูปที่ 2.11 การอ่านข้อมูลจากอุปกรณ์ Slave ผ่านบัส I2C

ภายใน DS1307 มีรีจิสเตอร์ภายในใช้เก็บข้อมูลเวลาขนาด 7 ไบต์ 00H-06H ดังแสดงในรูปที่ 2.12 ข้อมูลค่าเวลา และวันที่จะถูกเก็บอยู่ในรูปของเลขฐาน 10 สามารถเลือกได้ว่าให้ทำงานแบบ 12 ชั่วโมง หรือ 24 ชั่วโมง โดยกำหนดที่บิตที่ 6 ที่แอดเดรส 02H โดยถ้าเป็น “1” จะเป็นการทำงานในโหมด 12 ชั่วโมง และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิต 5 ในแอดเดรส 02H นั้นจะใช้แสดงค่า AM/PM โดยถ้าบิตนี้เป็น “1” จะเป็น PM ในกรณีที่แสดงแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่าของหลักสิบในของหน่วยชั่วโมงด้วย

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00-59
	0	10 MINUTES			MINUTES				00-59
	0	12/24	10 HR A/P	10 HR	HOURS				01-12 00-23
	0	0	0	0	0	DAY			1-7
	0	0	10 DATE		DATE				
	0	0	0	10 MONTH	MONTH				01-12
	10 YEAR				YEAR				00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	

รูปที่ 2.12 รีจิสเตอร์ภายในไอซีฐานเวลา DS1307

ที่แอดเดรส 07H เป็นรีจิสเตอร์ควบคุมการทำงานของ SQW/OUT โดยมีรายละเอียดดังนี้

OUT(Outcontrol):ใช้ควบคุมเอาต์พุต

SQWE(Square Wave Enable):ใช้ควบคุมออสซิลเลเตอร์ภายใน DS1307 โดยถ้าบิตนี้เป็น "1" จะเป็นการเปิดออสซิลเลเตอร์

RS(Rate Select):ใช้ควบคุมความถี่ของ Square Wave เมื่อเปิดการทำงานของออสซิลเลเตอร์

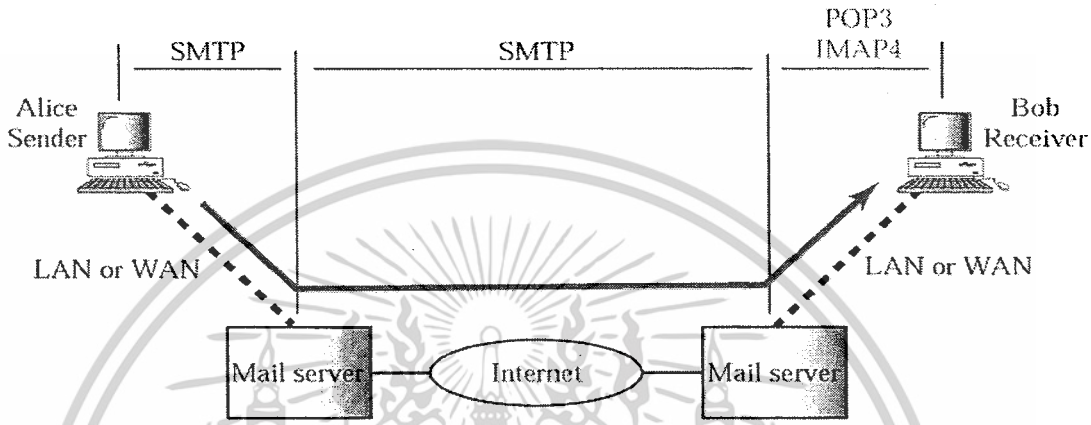
2.6 Mail Server

Mail Server คือ เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นไปรษณีย์ของระบบเครือข่าย ที่บริการรับ-ส่งจดหมายหรือ E-mail ให้แก่สมาชิกในหน่วยงานหรือองค์กร สำหรับติดต่อสื่อสารกันภายในเครือข่ายเดียวกัน หรือติดต่อกับบุคคลอื่นผ่านเครือข่ายอินเทอร์เน็ตก็ได้ ยกตัวอย่างที่ให้บริการ E-mail กับคนทั่วโลก อย่างเช่น Hotmail.com, Yahoo.com, Gmail.com เป็นต้น ซึ่งคอมพิวเตอร์ที่บ้านหรือสำนักงานก็สามารถทำหน้าที่เป็น Mail Server ได้ เพียงแค่ติดตั้งโปรแกรมเซิร์ฟเวอร์ไว้ที่เครื่องเท่านั้น

ประโยชน์ของการมี Mail Server เป็นของตนเองก็เพื่อความคล่องตัวในการจัดการกับ Mail Box ของผู้ใช้งานแต่ละคน โดยสามารถเพิ่มเติมหรือ แก้ไขข้อมูลของผู้ใช้ E-mail ภายในองค์กร จะมี E-mail Address เป็นของตัวเอง รวมทั้งไม่มีข้อจำกัดด้านเนื้อที่เก็บ E-mail

2.7 พอร์ตและโปรโตคอลในการรับส่ง Mail

การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ในเครือข่ายนั้นอาศัยพอร์ต (Port) และโปรโตคอล (Protocol) สำหรับ Mail Server นั้นทำหน้าที่ในการรับและส่ง Mail ให้กับผู้ใช้งานในเครือข่ายซึ่งเป็นการติดต่อสื่อสารกันระหว่างเครื่อง Mail Server กับเครื่องผู้ใช้งาน และระหว่าง Mail Server ของผู้ส่งและผู้รับปลายทางแสดงดังรูปที่ 2.13

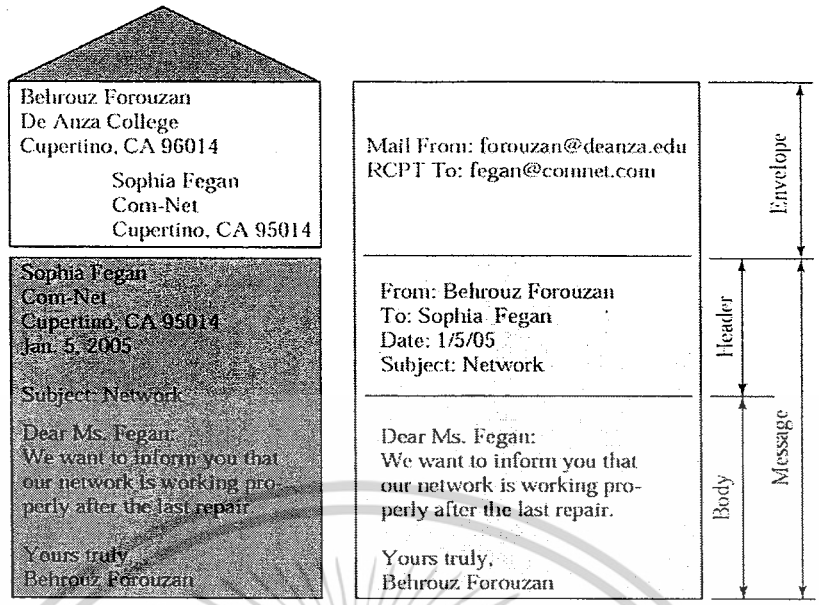


รูปที่ 2.13 แสดงขั้นตอนและโปรโตคอลที่ใช้ในการส่ง E-mail

2.7.1 โพรโตคอล SMTP

E-mail เป็นบริการที่ได้รับความนิยมเป็นอย่างมากในอินเตอร์เน็ต เนื่องจากสามารถส่งข้อมูลได้ ในอินเตอร์เน็ตจะใช้โพร โตคอลที่เรียกว่า SMTP (Simple Mail Transfer Protocol)

การส่ง Mail ก่อนที่จะทำการส่งได้นั้น ผู้ใช้ต้องทำการสร้างเมล์ขึ้นมาก่อนซึ่งลักษณะของจดหมายอิเล็กทรอนิกส์จะคล้ายกับจดหมายทั่วๆ ไปที่เราใช้กันอยู่ทุกวันนี้ ซึ่งจะต้องประกอบไปด้วย 2 ส่วนหลักๆคือ ซองจดหมาย (envelope) และข้อความในจดหมาย (message) ดังรูปที่ 2.14



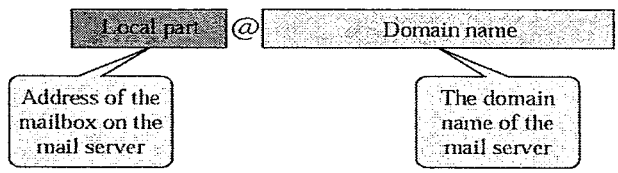
รูปที่ 2.14 รูปแบบ E-mail

ซองจดหมาย (Envelop) ซองจดหมายจะประกอบไปด้วย แอดเดรสของผู้ส่ง แอดเดรสของผู้รับ และส่วนเพิ่มอื่นๆ

ข้อความในจดหมาย (message) ข้อความในจดหมายจะประกอบไปด้วย เฮดเดอร์ (header) และเนื้อความจดหมาย (body) ซึ่งเฮดเดอร์จะเป็นเมสเซจที่จะบ่งบอกถึงผู้ส่ง ผู้รับ หัวข้อจดหมาย และส่วนเพิ่มเติมอื่นๆ ส่วนเนื้อความจดหมาย จะเป็นข้อความที่ผู้ส่งต้องการส่งให้กับผู้รับได้อ่าน

การรับ Mail ระบบอีเมล์นี้จะมีการตรวจสอบกล่องจดหมายหรือ Mail box อยู่เป็นระยะๆ เมื่อมี Mail มาส่งยังผู้ใช้ ระบบจะมีการแจ้งเตือนให้ผู้ใช้ได้รับรู้ โดยจะแสดงรายละเอียดคร่าวๆ ให้ผู้ใช้ เช่น แอดเดรสผู้ส่ง หัวข้อหมาย และเวลาที่ส่งหรือรับ Mail ผู้ใช้สามารถเลือกที่จะอ่านเมล์ที่ส่งมาให้ได้โดยข้างในจะเป็นเนื้อความของจดหมายที่ผู้ส่งต้องการให้ผู้รับได้อ่าน

แอดเดรส แอดเดรสที่ใช้กันในระบบ Mail จะต้องเป็นแอดเดรสเฉพาะของผู้ใช้แต่ละคน ซึ่งจะต้องไม่ซ้ำกัน ระบบแอดเดรสที่ใช้ในโพรโตคอล SMTP จะแบ่งออกเป็น 2 ส่วนคือ โลคอลพาร์ต (local part) และโดเมนเนม (domain name) ซึ่งจะเขียนแยกกันโดยใช้เครื่องหมาย “@” คั่นกลาง ดังรูปที่ 2.15



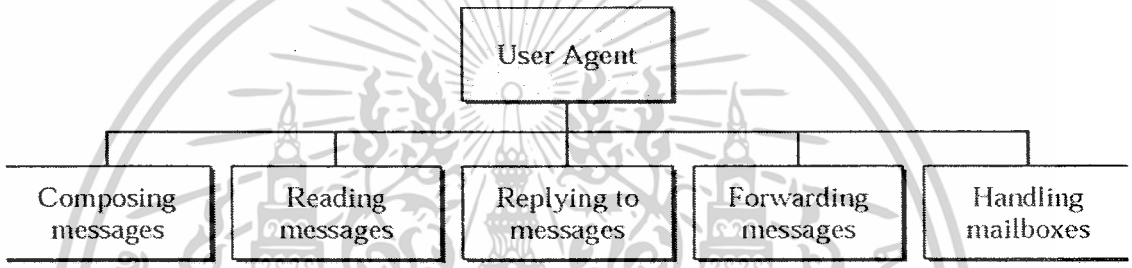
รูปที่ 2.15 E-mail Address

โลคอลพาร์ต (Local Part) จะใช้สำหรับกำหนดชื่อของผู้ใช้ในโดเมนเนมนั้นๆ ซึ่งชื่อของผู้ใช้นี้ในโดเมนหนึ่งๆ จะต้องไม่ซ้ำกัน

โดเมนเนม (Domain Name) ส่วนที่สองจะเรียกว่า โดเมนเนม หรือบางครั้งเรียกว่า mail exchanger โดยปกติจะใช้ชื่อขององค์กรเป็นหลัก

ยูเซอร์เอเจนต์ (User Agent : UA) องค์ประกอบอย่างหนึ่งของระบบไปรษณีย์อิเล็กทรอนิกส์คือ ยูเซอร์เอเจนต์ (UA) หรือบางครั้งเรียกว่า mail reader แต่เพื่อป้องกันการสับจึงของใช้คำว่ายูสเซอร์เอเจนต์

บริการของยูสเซอร์เอเจนต์ ยูสเซอร์เอเจ็นต์เป็นโปรแกรมที่สามารถให้ผู้ใช้เขียน อ่าน ตอบกลับ และส่งต่อจดหมายได้ รูปที่ 2.16 จะแสดงถึงบริการของยูสเซอร์เอเจนต์



รูปที่ 2.16 ยูสเซอร์เอเจนต์

Composing Message ยูสเซอร์เอเจนต์มีหน้าที่บริการให้ผู้ใช้สามารถเขียน Mail ได้ โดยส่วนใหญ่แล้วยูสเซอร์เอเจนต์จะมีเทมเพลต (template) บนหน้าจอเพื่อให้ผู้ใช้สามารถใช้งานได้สะดวกโดยการเติมข้อความลงไปได้

Reading Message หน้าที่อีกอย่างของยูสเซอร์เอเจนต์จะบริการให้ผู้ใช้สามารถที่จะอ่านเมล์ที่ถูกส่งมาให้ได้ เมื่อการสั่งให้ยูสเซอร์เอเจนต์ทำงาน มันจะไปตรวจสอบใน Mail box ว่ามีเมล์มาหรือไม่ ถ้ามีจะแสดงให้ผู้ใช้เห็นว่ามี Mail ใหม่เข้ามา

Replying Message หลังจากที่ได้มีการอ่านเมล์แล้ว ผู้ใช้สามารถใช้ยูสเซอร์เอเจนต์เพื่อตอบกลับได้โดยปกติการตอบกลับจะมีทั้งข้อความของผู้ส่งที่ส่งมาให้ และข้อความที่ผู้รับเขียนใหม่เพิ่มเข้าไป

Forwarding Message ในการตอบกลับนั้นจะเป็นการส่งไปให้กับผู้ที่ส่ง Mail มาให้ แต่การส่งต่อ (forward) จะเป็นการส่ง Mail ที่ได้รับมานั้นไปให้กับผู้ใช้คนอื่นๆ อีก

Handing Mailbox โดยปกติแล้วยูสเซอร์เอเจนต์จะสร้าง Mail box ไว้ 2 box คือ inbox และ outbox โดยที่ inbox จะใช้สำหรับเก็บ Mail ที่ได้รับเข้ามา ส่วน outbox จะใช้สำหรับเก็บเมล์ที่ถูกส่งออกไป

2.7.2 การตรวจสอบคำสั่งการใช้งาน SMTP โดยใช้โปรแกรม telnet

SMTP เป็นโปรโตคอลที่ใช้ในการส่ง Email ซึ่งจะทำงานที่ Port 25 การทำงานของ SMTP นั้นก็จะเหมือนกับ POP3 นั้นก็คือใช้การส่งคำสั่งไปประมวลผล โดยแต่ละคำสั่งนั้นจะลงท้ายด้วย \r\n (Carriage Return กับ Line Feed) สำหรับข้อความตอบกลับมาจาก SMTP จะเป็นตัวเลข 3 หลัก ซึ่งจะบอกผลการทำงานของคำสั่งที่ส่งไป

เราสามารถใช้คำสั่ง ดังตารางที่ 2.7 ทดลองโดยใช้ Telnet ยิงเข้าไปที่ SMTP Server ดังรูปที่ 2.16

รูปที่ 2.17 แสดงผลการส่งคำสั่ง helo, mail from, rcpt to, data, quit

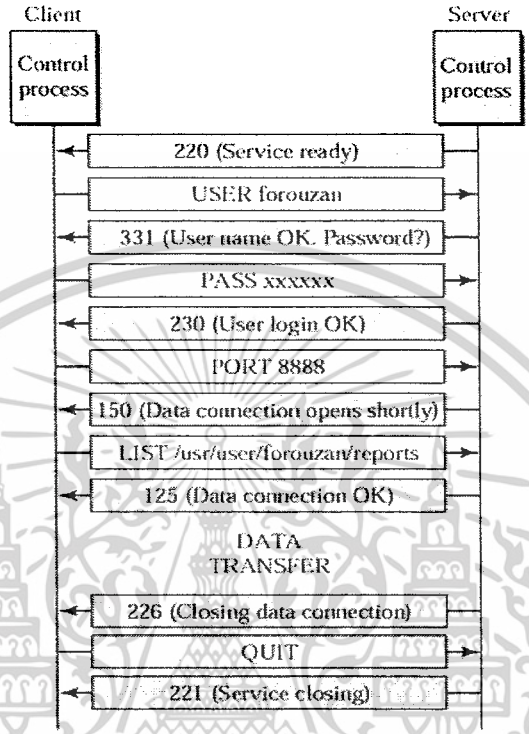
ตารางที่ 2.8 แสดงคำสั่งของ SMTP

คำสั่ง	หน้าที่
HELO<...>	ใช้ทักทายกับ SMTP Server
MAIL FROM:<EMAIL>	เริ่มต้นส่งเมล์ โดยกำหนด Email ต้นทาง
RCPT TO:<EMAIL>	กำหนดผู้รับปลายทาง
DATA	เริ่มต้นส่วนของข้อมูลตั้งแต่ตรงนี้เป็นต้นไป
RSET	ยกเลิกการส่ง
HELP	ขอดูระบบช่วยเหลือ
QUIT	ปิดการเชื่อมต่อและส่งเมล์

2.7.3 โพรโตคอล POP3

Post office Protocol, version 3 (POP3) เป็นโพรโตคอลที่ไม่ค่อยซับซ้อนมากนัก ดังนั้นจึงมีฟังก์ชันในการทำงานที่ค่อนข้างจำกัด ในการใช้งานจะต้องมีการติดตั้งซอฟต์แวร์ POP3 ทั้งที่ไคลเอนต์และเซิร์ฟเวอร์ได้นั้น ไคลเอนต์ (ยูสเซอร์เอเจนต์) จะต้องสร้างการติดต่อกับ Mail Server ก่อนผ่านทาง TCP พอร์ต 110 จากนั้นจะส่งชื่อผู้ใช้และรหัสผ่านเพื่อเป็นการล็อคอินเข้า

ใช้งาน Mail Server เมื่อล็อคอินได้แล้วจึงจะสามารถดูรายการต่างๆ ของ E-mail ใน Mail box ได้ นอกจากนั้นแล้วยังสามารถทำการดาวน์โหลด (download) E-mail จากเซิร์ฟเวอร์มายังเครื่องของผู้ใช้ได้ด้วย ดังรูปที่ 2.18 ซึ่งจะแสดงถึงตัวอย่างของการดาวน์โหลดอีเมล์โดยใช้ POP3

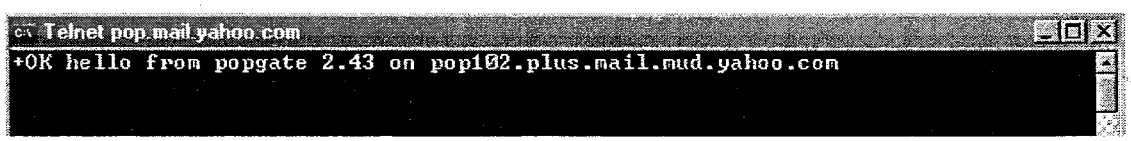


รูปที่ 2.18 POP3

2.7.4 การตรวจสอบคำสั่งการใช้งาน POP3 โดยใช้โปรแกรม telnet

เราสามารถทดสอบการทำงานของคำสั่งต่างๆได้ ดังตารางที่ 2.8 ได้โดยใช้โปรแกรม Telnet ที่มีมากับ Windows เชื่อมต่อเข้าไปยัง POP3 Email Server ทาง Port 110 เมื่อเชื่อมต่อ เข้าไปแล้วเราก็คำสั่ง ต่างๆ เข้าไป POP3 Server ก็จะตอบกลับมา ดังรูปที่ 2.19

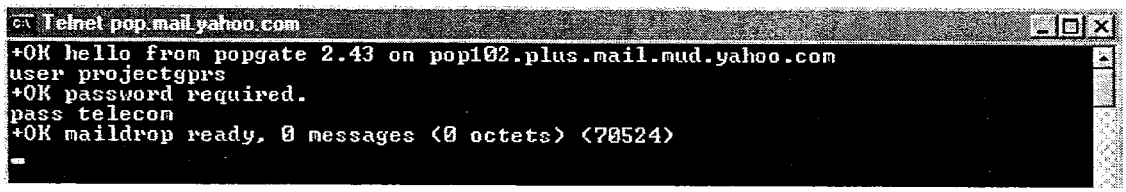
เราใช้โปรแกรม Telnet ทดสอบไปที่ pop.mail.yahoo.com



รูปที่ 2.19 แสดงผลการตอบสนองของ POP3 Server

จากนั้นทำการ Login เข้าสู่ mail server โดยใช้คำสั่ง USER ตามด้วยชื่อ Mail เมื่อ POP3 Server ตอบกลับ "OK" จะให้ระบุพาสเวิร์ด (password) โดยใช้คำสั่ง PASS ตามด้วยรหัสผ่าน

เมื่อรหัสผ่านสอดคล้องกับชื่อ Mail จะตอบกลับว่า "+OK" ดังรูปที่ 2.20 แต่ถ้าหากเกิดความผิดพลาด เช่น USER หรือ PASS ไม่สอดคล้องกัน จะตอบกลับด้วยสัญลักษณ์ -ERR



รูปที่ 2.20 แสดงการ Login เข้าสู่ระบบ mail server

ทดสอบการใช้งานคำสั่งต่างๆ ที่ใช้ควบคุมการทำงานของ POP3 ดังรูปที่ 2.21

รูปที่ 2.21 แสดงผลการส่งคำสั่ง stat, list, dele, rset, quit

ตารางที่ 2.9 แสดงคำสั่งของ POP3

คำสั่ง	หน้าที่
USER	ส่งชื่อ Login ของ Account Email
PASS	ส่งรหัสผ่าน
APOP	ส่งทั้งชื่อ Login พร้อมๆ กับรหัสผ่านเลย
STAT	ดูสถานะของ Email ใน Mail Box
LIST	ดูรายการ Email ทั้งหมด
RETR	อ่าน Email ฉบับที่ต้องการ
DELE	ลบ Email
RSET	ยกเลิกการสั่งลบที่ได้สั่งไป
NOOP	ไม่มีอะไร อยู่เฉยๆ
QUIT	ออกจากระบบ

2.8 SMTP Authentication

โดยหลักการแล้ว SMTP Server นั้นทำหน้าที่บริการทั้งรับและส่ง Mail หมายถึง รับ Mail จากเครื่องของผู้ส่งที่ส่งมาจากไหนก็ได้ และส่ง Mail ไปยัง Mail box ของ Mail Address ปลายทางที่ระบุไว้ สังเกตได้ว่า มันไม่ได้สนใจว่า ผู้ส่งเป็นใคร มาจากไหน ถ้าหากเป็นเช่นนี้ ใครๆ ก็สามารถส่ง Mail โดยอาศัยเครื่องเซิร์ฟเวอร์ของท่านได้เลย ดังนั้นโปรแกรม Mail Server ส่วนใหญ่จึงมีออปชั่นให้ท่านกำหนดเงื่อนไขในการส่ง Mail หลายๆแบบ เช่น กำหนดชั่วหมายเลข IP Address ให้เฉพาะคนในวงแลนสามารถส่ง Mail ออกได้ เป็นต้น อย่างไรก็ตามในกรณีที่อยู่นอกบ้านหรือนอกบริษัท แล้วต้องส่ง Mail ผ่าน Mail Server ของตัวเอง จะให้วิธีกำหนดหมายเลข IP Address ไม่ได้เนื่องจากหมายเลขดังกล่าวมันไม่แน่นอน วิธีที่มักใช้กันโดยทั่วไปก็คือ SMTP Authentication หมายถึง การส่ง Mail ต้องผ่านลอกอินระบุชื่อและรหัสผ่าน

2.9 โปรแกรม Email-Client

Mail Client คือโปรแกรมที่ติดตั้งบนเครื่องลูกข่าย หรือเครื่องคอมพิวเตอร์ของผู้ใช้งานแต่ละคน เพื่อทำหน้าที่รับส่ง Mail กับ Mail Server เช่น Outlook Express ที่มีมาให้มาพร้อมกับระบบปฏิบัติการ Windows อยู่แล้ว

2.10 ฐานข้อมูล (Database)

ฐานข้อมูล (Database) คือ วิธีการจัดเก็บข้อมูลที่สัมพันธ์กันอย่างมีระเบียบ ซึ่งจะทำให้ง่ายต่อการใช้งานและค้นหาข้อมูล ซึ่งฐานข้อมูลที่คนส่วนใหญ่คุ้นเคยคือ ฐานข้อมูลเชิงสัมพันธ์ (Relation Database) เป็นรูปแบบการจัดเก็บข้อมูลในฐานข้อมูลที่สัมพันธ์กัน โดยมองข้อมูลในลักษณะของตารางต่างๆ ที่มี

ความสัมพันธ์กันฐานข้อมูลเป็นเพียงวิธีคิดเท่านั้น แต่การใช้งานฐานข้อมูลจะต้องมีองค์ประกอบดังต่อไปนี้

- แอพพลิเคชันฐานข้อมูล (Database Application) แอพพลิเคชันฐานข้อมูล เป็นแอพพลิเคชันที่สร้างไว้ให้ผู้ใช้งานสามารถติดต่อกับฐานข้อมูลได้อย่างสะดวก ซึ่งมีรูปแบบการติดต่อกับฐานข้อมูลแบบเมนู หรือกราฟฟิก โดยไม่จำเป็นต้องมีความรู้เกี่ยวกับฐานข้อมูลเลยก็สามารถเรียกใช้งานฐานข้อมูลได้

- ระบบจัดการฐานข้อมูล (Database Management System) ระบบจัดการฐานข้อมูล (DBMS) เป็นซอฟต์แวร์ที่ทำหน้าที่จัดการข้อมูลในฐานข้อมูล ทั้งการจัดเก็บ การแสดงผล การค้นหา การสำรองข้อมูล ฯลฯ โดยจะเป็นเครื่องมือในการทำงานของผู้บริหารฐานข้อมูล และเป็นตัวกลางที่เชื่อมผ่านระหว่างแอพพลิเคชันฐานข้อมูลที่สร้างขึ้นกับตัวข้อมูลในฐานข้อมูล ตัวอย่างของ DBMS เช่น Microsoft Access, FoxPro, SQL Server, Oracle, Informix, DB2 เป็นต้น

- ดาต้าเบสเซิร์ฟเวอร์ (Database Server) ดาต้าเบสเซิร์ฟเวอร์ เป็นคอมพิวเตอร์ที่คอยให้บริการในการจัดการฐานข้อมูล ซึ่งก็คือ คอมพิวเตอร์ที่ระบบจัดการฐานข้อมูลทำงานอยู่นั่นเอง เพราะฉะนั้นจึงมักจะเป็นคอมพิวเตอร์ที่มีประสิทธิภาพการทำงานสูงกว่าคอมพิวเตอร์ที่ใช้งานทั่วไป

- ข้อมูล (Data) ข้อมูลคือ ตัวเนื้อหาของข้อมูลที่เราใช้งาน ซึ่งจะถูกเก็บในหน่วยความจำของดาต้าเบสเซิร์ฟเวอร์ซึ่งจะถูกเรียกมาใช้งานโดยระบบจัดการฐานข้อมูล

- ผู้บริหารฐานข้อมูล (Database Administrator) ผู้บริหารฐานข้อมูล เป็นคนที่ทำหน้าที่ดูแลข้อมูลในฐานข้อมูลผ่านระบบจัดการฐานข้อมูล ซึ่งจะคอยคุมให้ทำงานเป็นไปอย่างราบรื่น นอกจากนี้ยังทำหน้าที่กำหนดผู้ที่จะมีสิทธิใช้งานฐานข้อมูล กำหนดในเรื่องความปลอดภัยของการใช้งาน พร้อมทั้งดูแลดาต้าเบสเซิร์ฟเวอร์ให้ทำงานอย่างปกติด้วย

2.10.1 ภาษา SQL

SQL ย่อมาจาก Structured Query Language คือภาษามาตรฐานสำหรับทำงานกับฐานข้อมูลเชิงสัมพันธ์ ภาษานี้มีใช้ในระบบฐานข้อมูลยอดนิยมทั้งหลาย ไม่ว่าจะเป็น MySQL, Oracle, Microsoft SQL Server, PostgreSQL, Sybase

คำสั่งในภาษา SQL แบ่งออกเป็น 2 กลุ่มคือ

- คำสั่งในกลุ่ม DDL (Data Definition Language) ประกอบด้วยคำสั่งที่ใช้สร้างฐานข้อมูลสร้างเทเบิล แก้ไขเทเบิล ฯลฯ

- คำสั่งในกลุ่ม DML (Data Manipulation Language) คือคำสั่งที่ใช้ทำงาน กับข้อมูลในฐานข้อมูล

2.10.2 การเขียนโปรแกรมติดต่อฐานข้อมูล

ในการพัฒนาแอพพลิเคชันด้วยเทคโนโลยี ADO.NET นั้นจะใช้งานออบเจ็กต์อยู่หลายๆ ตัวนำมาประกอบกันเป็นแอพพลิเคชัน ดังตารางที่ 2.9

- ออบเจ็กต์จำพวก connection เป็นออบเจ็กต์ที่ทำหน้าที่เชื่อมต่อกับฐานข้อมูล

- ออบเจ็กต์จำพวก data adapter เป็นออบเจ็กต์เหมือนสะพานเชื่อมต่อระหว่างข้อมูลในฐานข้อมูลกับข้อมูลใน data container โดยจะทำหน้าที่หลักในการนำข้อมูลมาเก็บใน data container หรืออัพเดตข้อมูลในฐานข้อมูลที่เปลี่ยนแปลงใน data container ให้ตรงกับฐานข้อมูล

- ออบเจ็กต์จำพวก data container เป็นออบเจ็กต์ที่เก็บและประมวลผลข้อมูลที่นำมาจากฐานข้อมูล เป็นเสมือนหน่วยความจำที่สำเนามาจากฐานข้อมูล ซึ่งตรงนี้ถือเป็นหัวใจของ ADO.NET

ตารางที่ 2.10 แสดงชื่อออบเจ็กต์กับฐานข้อมูลชนิดต่างๆ

ออบเจ็กต์จำพวก	OLEDB	SQL Server	Oracle
Connection	OleDbConnection	SqlConnection	OracleConnection
data adapter	OleDbDataAdapter	SqlDataAdapter	OracleDataAdapter
Data container	DataSet, DataTable, DataRow		

จากแนวคิดข้างต้นเราสามารถสร้างแอพพลิเคชันได้ดังนี้

- ขั้นแรก เชื่อมต่อเข้าสู่ฐานข้อมูล ในการเชื่อมต่อฐานข้อมูลเข้าแอพพลิเคชัน เราสามารถเลือกตัวกลางในการเชื่อมต่อตามชนิดของฐานข้อมูล ปกติเราจะใช้ออบเจ็กต์จำพวก connection ทำหน้าที่นี้

- ขั้นสอง เข้าถึงแหล่งข้อมูล เมื่อเชื่อมต่อเข้าสู่ฐานข้อมูลเสร็จแล้ว ขั้นตอนต่อไปคือการระบุถึงแหล่งข้อมูลหรือตัวข้อมูลที่ได้จากการเชื่อมต่อนั้น ซึ่งเปรียบไปก็เหมือนกับการต่อท่อเข้าไปดูดข้อมูลออกมา

- ขั้นสาม จัดการนำข้อมูลมาแสดงผลหรือประมวลผล เมื่อเข้าถึงแหล่งข้อมูลแล้ว ต่อไปก็เป็นส่วนของการจัดการข้อมูล โดยเราสามารถนำข้อมูลนั้นมาแสดงผลในรูปแบบต่างๆ เช่น ฟอร์ม รายงาน หรือตาราง ขณะเดียวกันก็สามารถลบ เพิ่ม แก้ไขข้อมูลที่ได้มานั้นด้วย

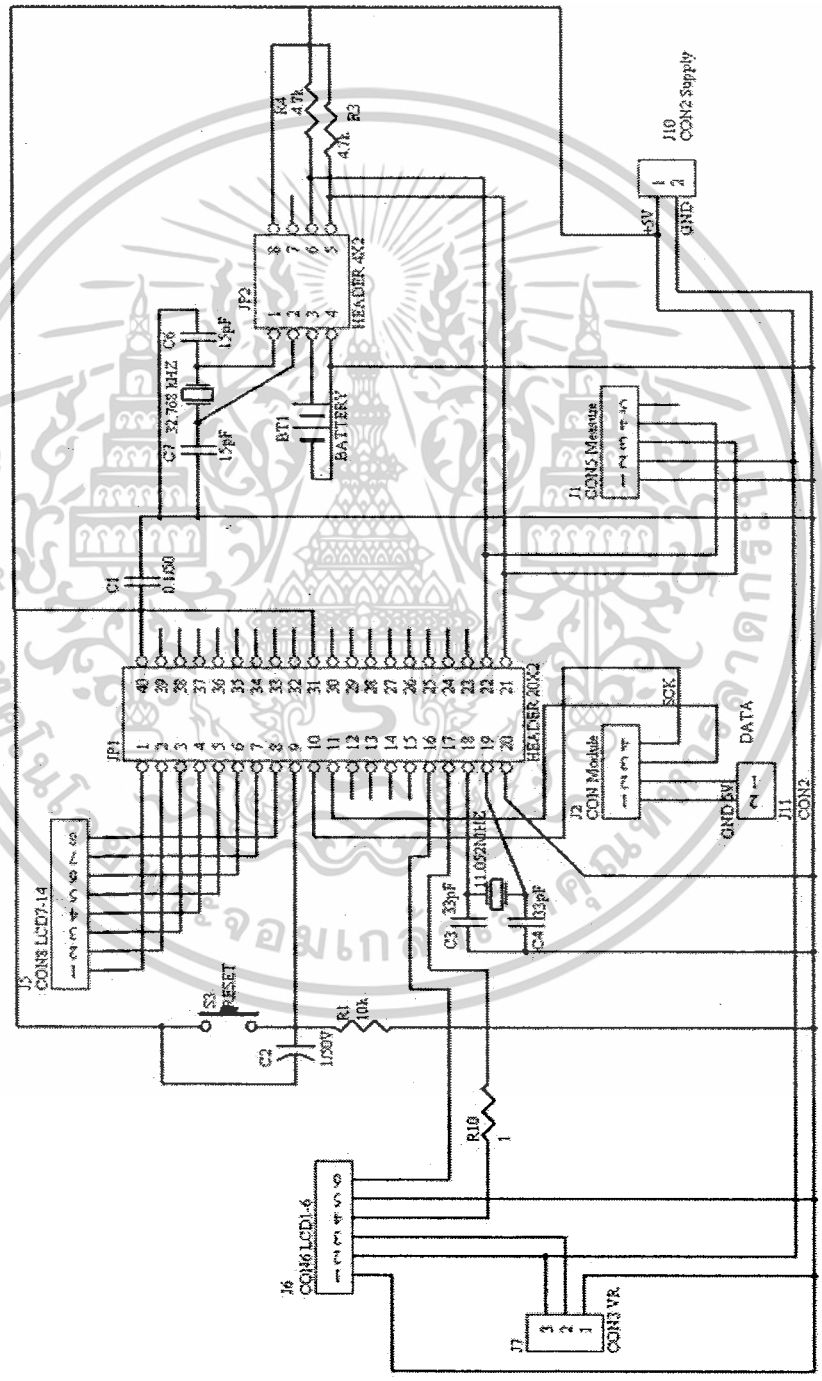
เช่นเดียวกันเราสามารถเลือกใช้ออบเจ็กต์ที่ใช้จัดการข้อมูลได้ว่าจะเป็นแบบไหน โดยแยกตามเกณฑ์ของการเชื่อมต่อเหมือนขั้นตอนที่สอง ซึ่งการเชื่อมต่อแบบแช่นั้นเราสามารถใช้ออบเจ็กต์ DataReader ส่วนการเชื่อมต่อแบบตัดการเชื่อมต่อก็จะใช้ออบเจ็กต์ DataSet เป็นต้น

บทที่ 3

การออกแบบและการสร้าง

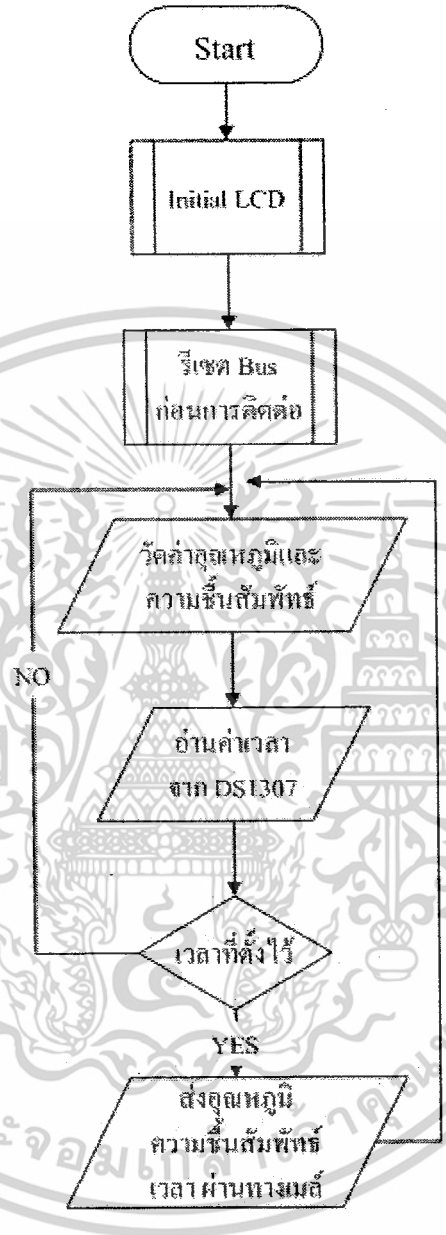
3.1 ส่วนของ Microcontroller

สร้างวงจร Microcontroller พื้นฐานเพื่อใช้ในการเชื่อมต่อกับ SHT-15, DS1307, TNS010i แสดงดังรูปที่ 3.1

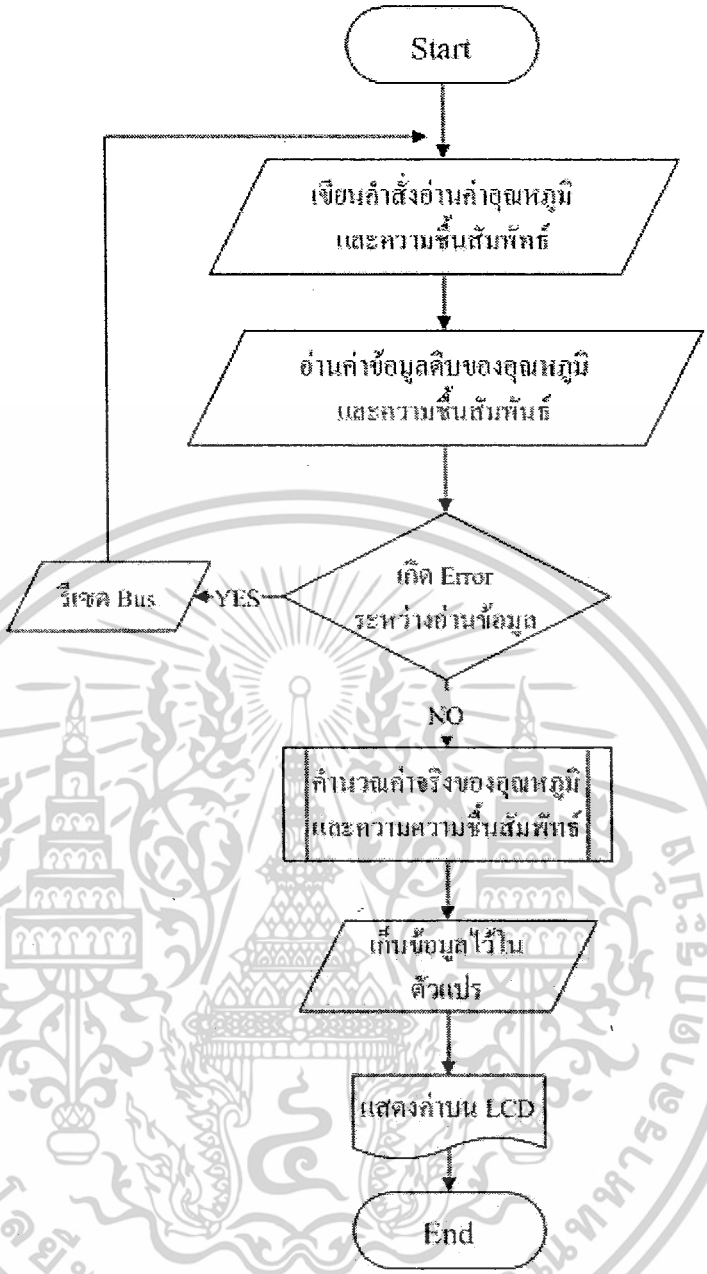


รูปที่ 3.1 แสดงวงจรระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

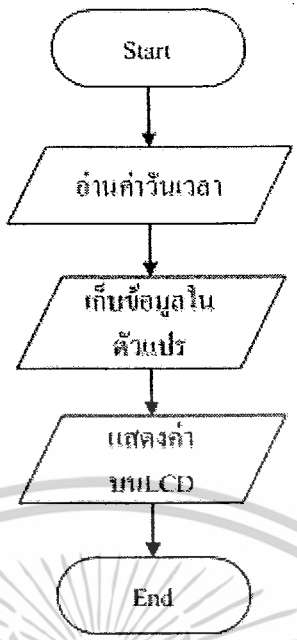
ทำการออกแบบ Flowchart แสดงการทำงานของวงจรระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS แสดงดังรูปที่ 3.2 รูปที่ 3.3 รูปที่ 3.4 รูปที่ 3.5



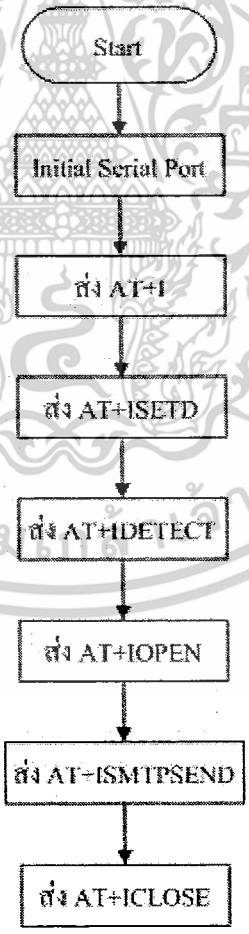
รูปที่ 3.2 Flowchart วงจรระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS



รูปที่ 3.3 Flowchart โปรแกรมย่อยอ่านค่าอุณหภูมิและความชื้นจาก SHT-15



รูปที่ 3.4 Flowchart โปรแกรมย่อยอ่านค่าวันเวลาจาก DS1307



รูปที่ 3.5 Flowchart โปรแกรมย่อยส่ง E-mail

3.2 ส่วนของ Window Application

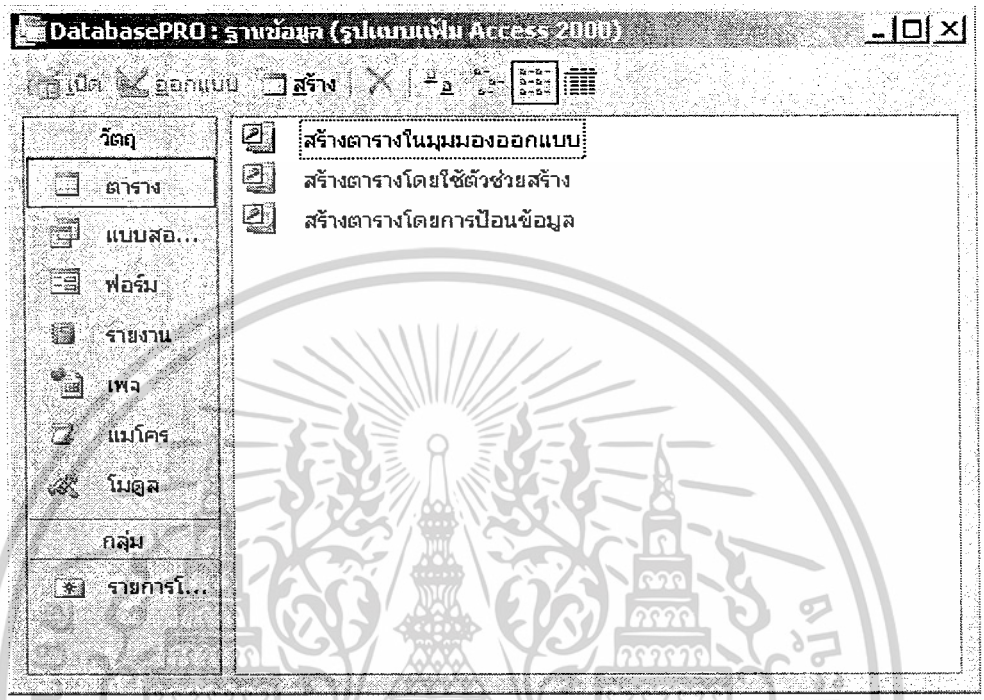
สร้างโปรแกรมอ่าน E-mail ที่เขียนโดย Visual C#.NET โดยใช้คำสั่ง POP3 มาควบคุมการทำงานของโปรแกรม แสดงดังรูปที่ 3.6

รูปที่ 3.6 Flowchart โปรแกรมระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS

3.2.1 การออกแบบฐานข้อมูล

ต้องทำการออกแบบฐานข้อมูลเพื่อง่ายต่อการสร้างในส่วนอื่นๆต่อไป

สร้างฐานข้อมูลชื่อ DatabasePRO แสดงดังรูปที่ 3.7



รูปที่ 3.7 ฐานข้อมูล DatabasePRO

ทำการสร้างตารางชื่อ data ไว้สำหรับเก็บพารามิเตอร์ที่ได้จากการอ่าน Subject ของ E-mail โดยทำการสร้าง Field Place เก็บสถานที่ Temp เก็บอุณหภูมิ Humi เก็บความชื้นสัมพัทธ์ Qty เก็บเวลา OrderDate เก็บวันที่ แสดงได้ดังรูปที่ 3.8

data : ตาราง

	ชื่อเขตข้อมูล	ชนิดข้อมูล
🔑	ID	AutoNumber
	Place	Text
	Temp	Text
	Humi	Text
	Qty	Text
	OrderDate	Text

รูปที่ 3.8 ตาราง buffer

ทำการสร้างตารางชื่อ Setting ไว้สำหรับเก็บพารามิเตอร์ที่ใช้ในการ log in เข้าไปอ่าน E-mail โดยทำการสร้าง Field PopServer เก็บชื่อ POP3 Server Username เก็บชื่อ E-mail Password เก็บรหัสผ่าน TimeInterval เก็บช่วงเวลาที่อ่านจะ E-mail ใหม่ แสดงดังรูปที่ 3.9

Setting : ตาราง

ชื่อเขตข้อมูล	ชนิดข้อมูล
PopServer	Text
Username	Text
Password	Text
TimeInterval	Number

รูปที่ 3.9 ตาราง data

ทำการสร้างตารางชื่อ buffer ไว้สำหรับเก็บค่าที่ได้จากการเลือกวันเดือนปีที่จะแสดงกราฟ โดยทำการสร้าง Field เหมือนกับตาราง data แสดงดังรูปที่ 3.10

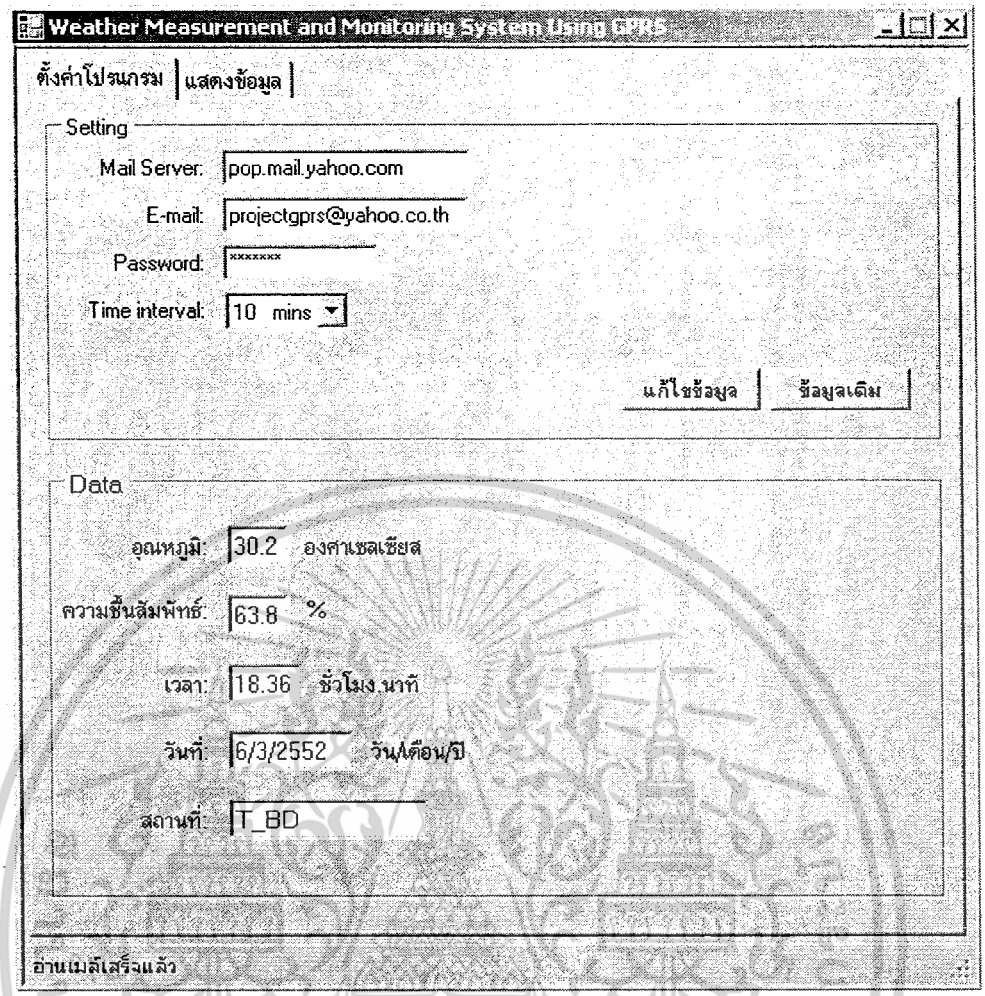
buffer : ตาราง

ชื่อเขตข้อมูล	ชนิดข้อมูล
ID	AutoNumber
Place	Text
Temp	Text
Humi	Text
Qty	Text
OrderDate	Text

รูปที่ 3.10 ตาราง Setting

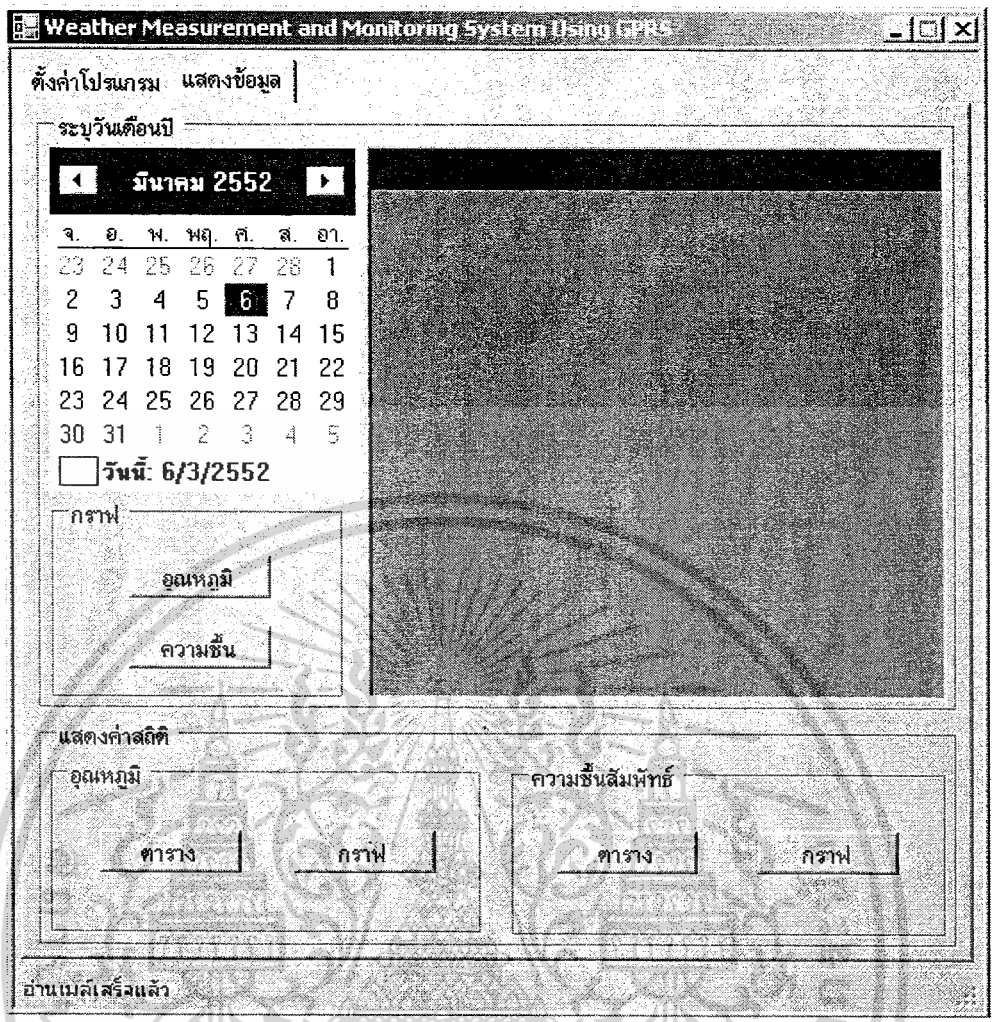
3.2.2 การออกแบบ User Interface

ทำการสร้างเป็น 2 ส่วน โดยส่วนแรกจะเป็นส่วนที่ใช้กรอกข้อมูลที่ใช้ในการ log in เข้าไปอ่าน E-mail และแสดงค่าต่างๆ ที่อ่านได้จาก Subject ของ E-mail โดยนำค่าที่แสดงมาจากตาราง Setting และตาราง data แสดงดังรูปที่ 3.11



รูปที่ 3.11 User Interface ส่วนแรก

โดยส่วนสองจะเป็นส่วนที่ใช้แสดงข้อมูลในรูปของตารางและกราฟ โดยสามารถเลือกวันเดือนปีที่จะแสดงได้ โดยนำค่าที่แสดงมาจากตาราง data และตาราง buffer แสดงดังรูปที่ 3.12



รูปที่ 3.12 User Interface ส่วนสอง

บทที่ 4
ผลการทดลอง และวิเคราะห์ผลการทดลอง

4.1 ผลการทดลองวัดสัญญาณอุณหภูมิและความชื้นสัมพัทธ์จากเซนเซอร์ SHT-15

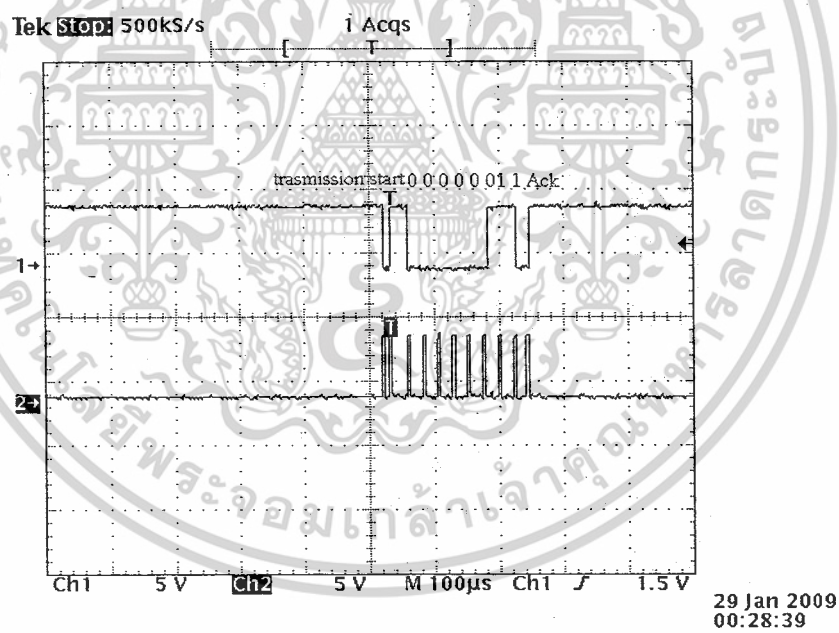
การวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ด้วยเซนเซอร์ SHT-15 นั้นต้องมีการติดต่อแบบระบบบัส I^2C ที่หลังจากที่มีการส่งสัญญาณติดต่อ (Transmission Start) ไปกระตุ้นการทำงานเซนเซอร์แล้วต้องมีการเขียนคำสั่ง พร้อมสัญญาณนาฬิกาเพื่อสั่งให้เซนเซอร์ดังกล่าวทำการวัดค่าค่าอุณหภูมิหรือความชื้นสัมพัทธ์ โดยมีรายละเอียดผลการทดลองดังนี้ดังนี้

4.1.1 การวัดสัญญาณอุณหภูมิจากเซนเซอร์ SHT-15

การทดลองวัดค่าอุณหภูมิต้องมีการส่งคำสั่ง 00000011 จากไมโครคอนโทรลเลอร์ไปยังเซนเซอร์ดังรูปที่ 4.1

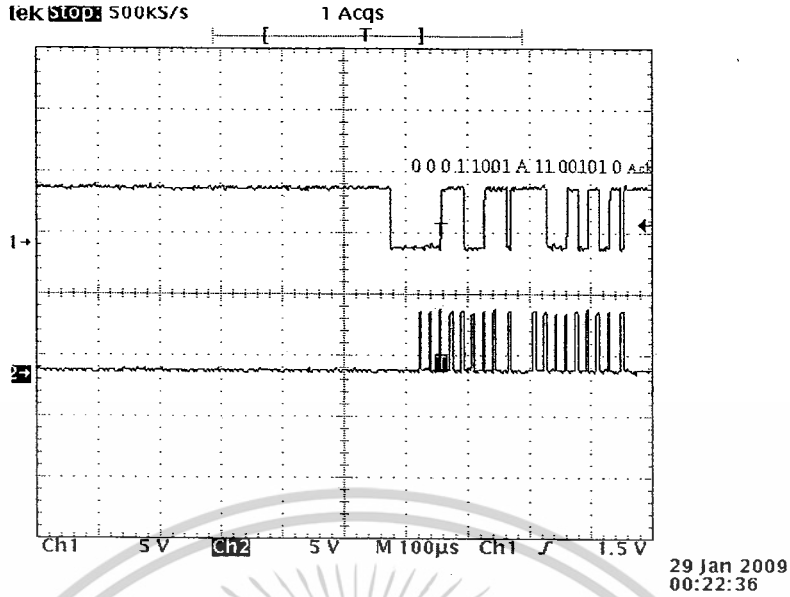
โดย รูปสัญญาณช่องที่ 1 เป็นคำสั่งวัดค่าอุณหภูมิ (00000011)

รูปสัญญาณช่องที่2 เป็นสัญญาณนาฬิกา



รูปที่ 4.1 ผลการทดลองส่งคำสั่งวัดค่าอุณหภูมิจากไมโครคอนโทรลเลอร์ไปเซนเซอร์ SHT-15

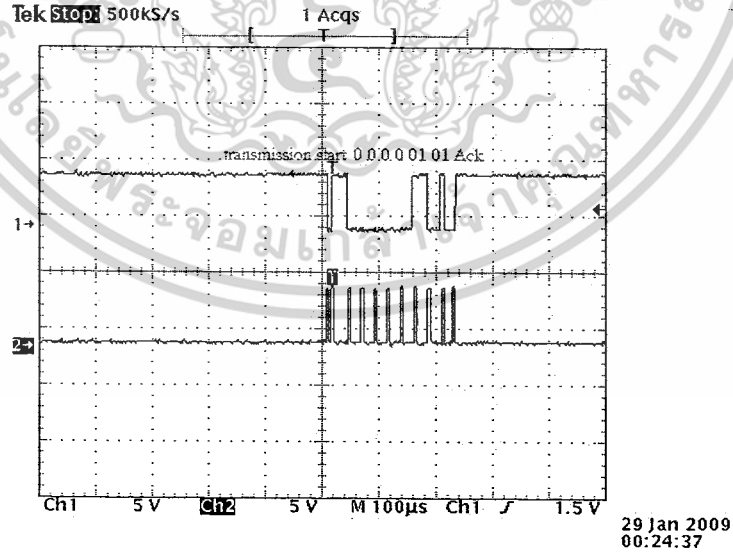
ผลข้อมูลที่เซนเซอร์วัดมาได้จะเป็นข้อมูลดิบที่มีรายละเอียดดังรูปสัญญาณที่แสดงดังรูปที่ 4.2 โดยข้อมูลดังกล่าวต้องผ่านการคำนวณทางคณิตศาสตร์อีกครั้งเพื่อได้ค่าอุณหภูมิจริง



รูปที่ 4.2 ผลการทดลองรับข้อมูลค่าอุณหภูมิจากไมโครคอนโทรลเลอร์ที่วัดด้วย SHT-15

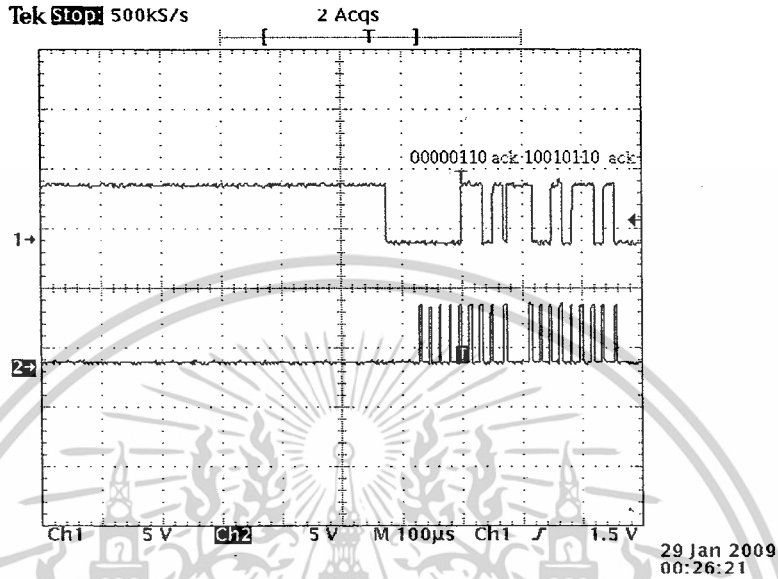
4.1.2 การวัดสัญญาณความชื้นสัมพัทธ์เซนเซอร์ SHT-15

การวัดค่าความชื้นสัมพัทธ์เซนเซอร์ SHT-15 มีการติดต่อกับเซนเซอร์แบบบัสระบบบัส I^2C เช่นเดียวกับการวัดอุณหภูมิแต่ในการส่งคำสั่งในการวัดนั้นคำสั่งวัดความชื้นสัมพัทธ์จะเป็น 00000101แสดงดังรูปที่ 4.3 ซึ่งไมโครคอนโทรลเลอร์จะส่งคำสั่งดังกล่าวไปยังตัวเซนเซอร์



รูปที่ 4.3 ผลการทดลองส่งคำสั่งวัดค่าความชื้นสัมพัทธ์จากไมโครคอนโทรลเลอร์ไป SHT-15

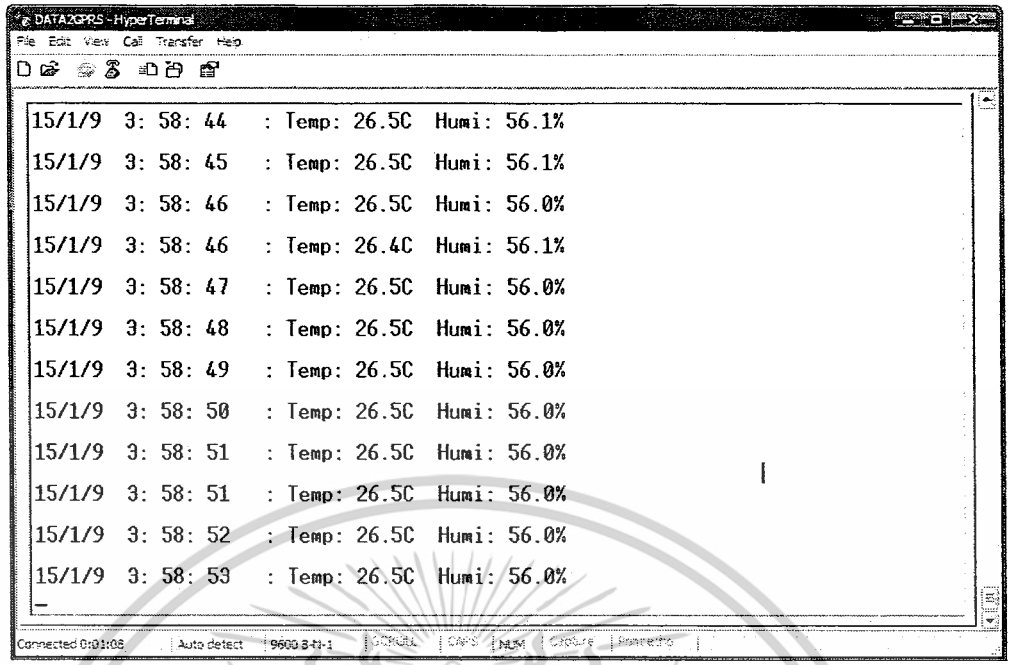
ผลข้อมูลที่เซนเซอร์วัดมาได้จะเป็นข้อมูลความชื้นสัมพัทธ์ดิบที่มีรายละเอียดดัง รูปที่ 4.4 สัญญาณที่แสดงโดยข้อมูลดังกล่าวต้องผ่านการคำนวณทางคณิตศาสตร์อีกครั้งเพื่อได้ค่าความชื้นสัมพัทธ์จริง



รูปที่ 4.4 ผลการรับข้อมูลค่าความชื้นสัมพัทธ์ที่ไมโครคอนโทรลเลอร์รับได้จากวัดด้วย SHT-15

4.2 การทดลองวัดค่าอุณหภูมิและความชื้นสัมพัทธ์จากเซนเซอร์ SHT-15 ณ เวลาต่างๆ

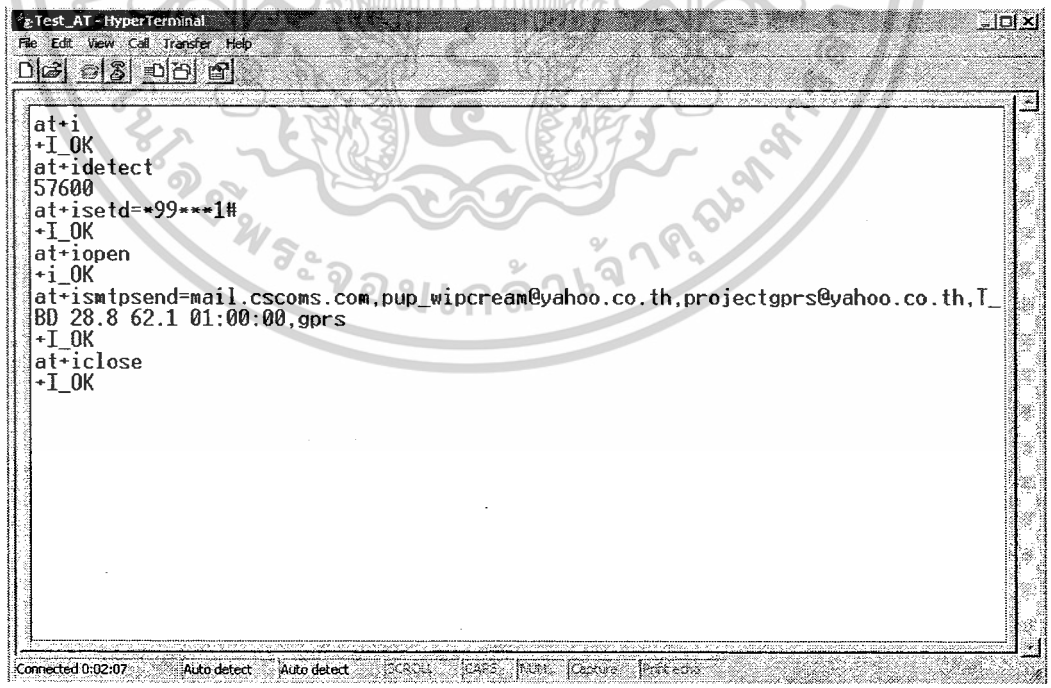
ในการวัดค่าอุณหภูมิและความชื้นสัมพัทธ์เซนเซอร์ SHT-15 นั้นค่าที่รับมายังเป็นข้อมูลดิบจึงต้องมีการเปลี่ยนข้อมูลดังกล่าวให้เป็นค่าที่แท้จริงโดยอาศัยสมการทางคณิตศาสตร์มาคำนวณ ซึ่งในการทดลองนี้ ได้ทำการเขียนโปรแกรมจากไมโครคอนโทรลเลอร์ควบคุมการวัดค่าอุณหภูมิและความชื้นสัมพัทธ์แสดงค่าออกทางหน้าจอคอมพิวเตอร์ด้วยโปรแกรม Hyperterminal ผ่านทางพอร์ตอนุกรมพร้อมทั้งแสดงค่าเวลาที่วัดนั้นๆซึ่งค่าเวลาดังกล่าวได้จากไอซีฐานเวลาจริง (real time) DS1307 แสดงดังรูปที่ 4.5



รูปที่ 4.5 แสดงผลการวัดค่าอุณหภูมิ ความชื้นสัมพัทธ์ และเวลาด้วย SHT-15 และ DS1307

4.3 การทดลองการส่งคำสั่ง AT-Command เพื่อให้ TCP/IP Stack Chip TNS010i ส่ง E-mail

ในการสั่งให้ส่ง E-mail นั้น จะต้องทำการสั่งให้ไมโครคอนโทรลเลอร์ทำการติดต่อที่ 57600 bps แล้วส่งคำสั่งไปตามดังรูปที่ 4.6



รูปที่ 4.6 แสดงผลการส่งคำสั่ง AT-Command ที่ใช้ในการส่ง E-mail

at+i ทดสอบความพร้อมของ TCP/IP chip

at+idetect เช็คค่า baudrate ของโมเดม (โทรศัพท์เคลื่อนที่)

at+isetd=*99***1# กำหนดค่า dial number ของ ISP ในที่นี้คือ *99***1#

at+iopen ทำการเชื่อมต่อกับ ISP และเตรียมใช้งาน TCP/IP socket

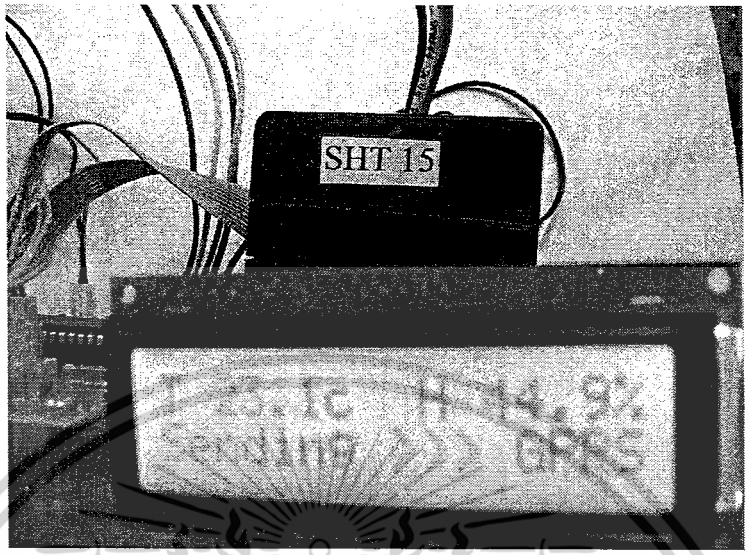
at+ismtpsend=mail.cscoms.com,pup_wipcream@yahoo.co.th,projectgprs@yahoo.co.th,T_BD 28.8 60.3 01:00:01,gprs ทำการส่ง E-mail รูปแบบคำสั่งคือ SMTP Server, Mail Sender, Mail Receiver, Subject, Body

4.4 แสดงผลการทดลองจากชิ้นงานจริง

ช่วงเวลาปกติจะแสดงค่าอุณหภูมิ ความชื้นสัมพัทธ์ วันเดือนปี และเวลา ดังรูปที่ 4.7

รูปที่ 4.7 จอ LCD แสดงสถานะช่วงเวลาปกติ

แต่เมื่อถึงช่วงเวลาที่กำหนดจะแสดงการส่ง E-mail ดังรูปที่ 4.8



รูปที่ 4.8 จอ LCD แสดงสถานะช่วงทำการส่ง E-mail

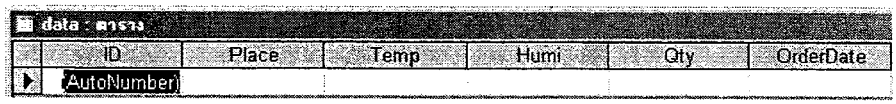
เมื่อส่ง E-mail สำเร็จจะแสดงดังรูปที่ 4.9

รูปที่ 4.9 จอ LCD แสดงสถานะส่ง E-mail สำเร็จ

4.5 การทดลองอ่าน E-mail จากโปรแกรมผ่าน POP3 และเก็บเข้าฐานข้อมูล

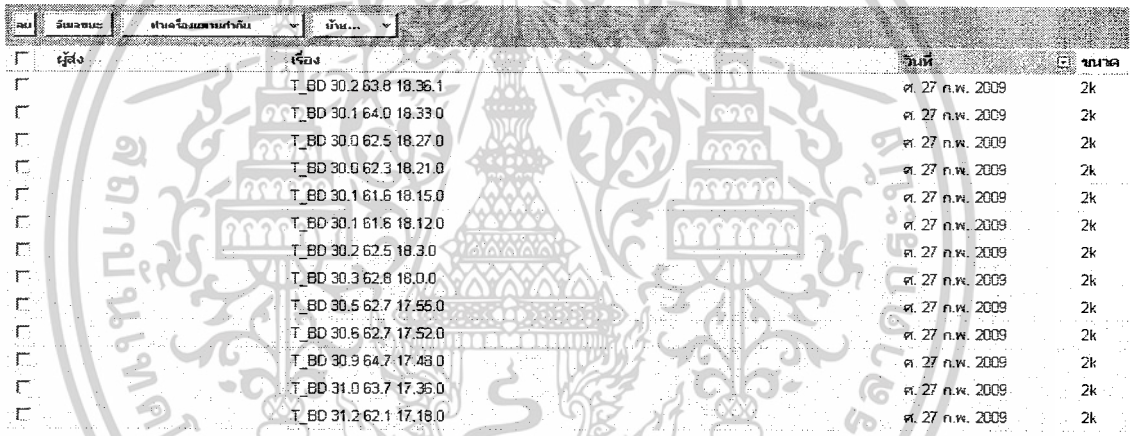
การทดสอบเพื่อให้ทราบว่าทำงานถูกต้อง โดยทำการเปรียบเทียบข้อมูลที่เก็บไว้ในฐานข้อมูลชื่อ DatabasePRO ตารางชื่อ data ก่อนที่จะรันโปรแกรม

เปิดตาราง data จากฐานข้อมูล DatabasePRO แสดงดังรูปที่ 4.10



รูปที่ 4.10 ข้อมูลในตาราง data ก่อนรันโปรแกรม

จากนั้นทำการส่ง E-mail เข้า Mail Server ในที่นี้ใช้ของ yahoo เพราะ yahoo มีบริการ POP3 แสดงดังรูปที่ 4.11



รูปที่ 4.11 E-mail ที่จะทำการอ่านและเก็บเข้าฐานข้อมูล

และทำการรันโปรแกรมข้อความในส่วน Subject จะถูกแยกเข้าสู่ field ต่างๆ ในตาราง data โดยรูปแบบ Subject คือ สถานที่ อุณหภูมิ ความชื้นสัมพัทธ์ เวลา แสดงดังรูปที่ 4.12

data : ตาราง

ID	Place	Temp	Humi	Qty	OrderDate
1071	T_BD	31.2	62.1	17.18	8/3/2552
1072	T_BD	31.0	63.7	17.36	8/3/2552
1073	T_BD	30.9	64.7	17.48	8/3/2552
1074	T_BD	30.6	62.7	17.52	8/3/2552
1075	T_BD	30.5	62.7	17.55	8/3/2552
1076	T_BD	30.3	62.8	18.00	8/3/2552
1077	T_BD	30.2	62.5	18.03	8/3/2552
1078	T_BD	30.1	61.6	18.12	8/3/2552
1079	T_BD	30.1	61.6	18.15	8/3/2552
1080	T_BD	30.0	62.3	18.21	8/3/2552
1081	T_BD	30.0	62.5	18.27	8/3/2552
1082	T_BD	30.1	64.0	18.33	8/3/2552
1083	T_BD	30.2	63.8	18.36	8/3/2552
(AutoNumber)					

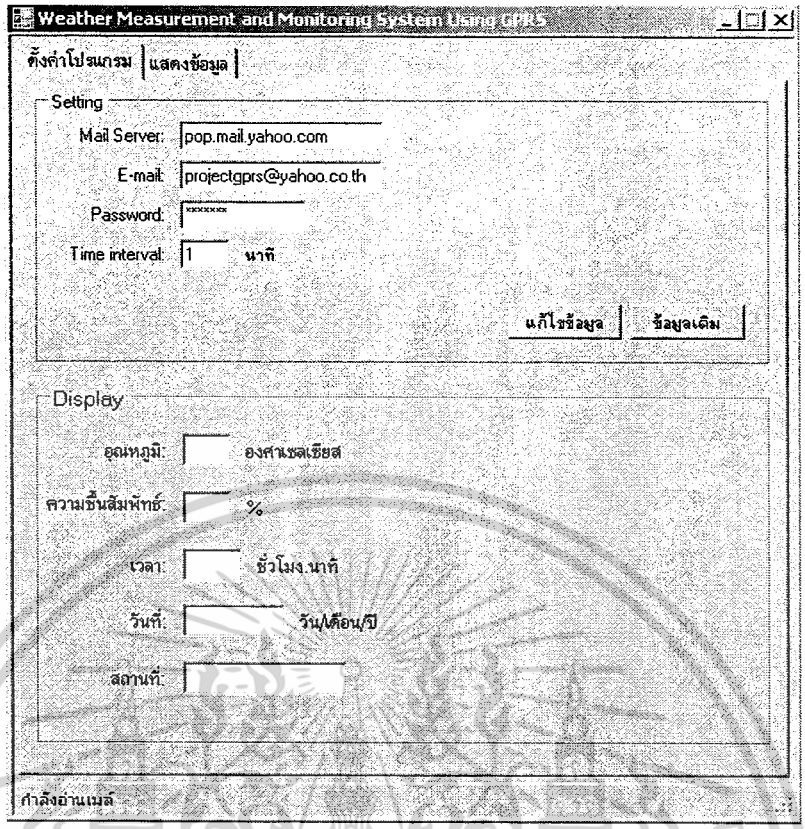
รูปที่ 4.12 ตาราง data หลังจากรันโปรแกรม

หลังจากทำการรันโปรแกรมจะลบ E-mail ที่อยู่ในกล่องรับจดหมายทั้งหมด แสดงดังรูปที่ 4.13

รูปที่ 4.13 กล่องรับจดหมายหลังจากรันโปรแกรม

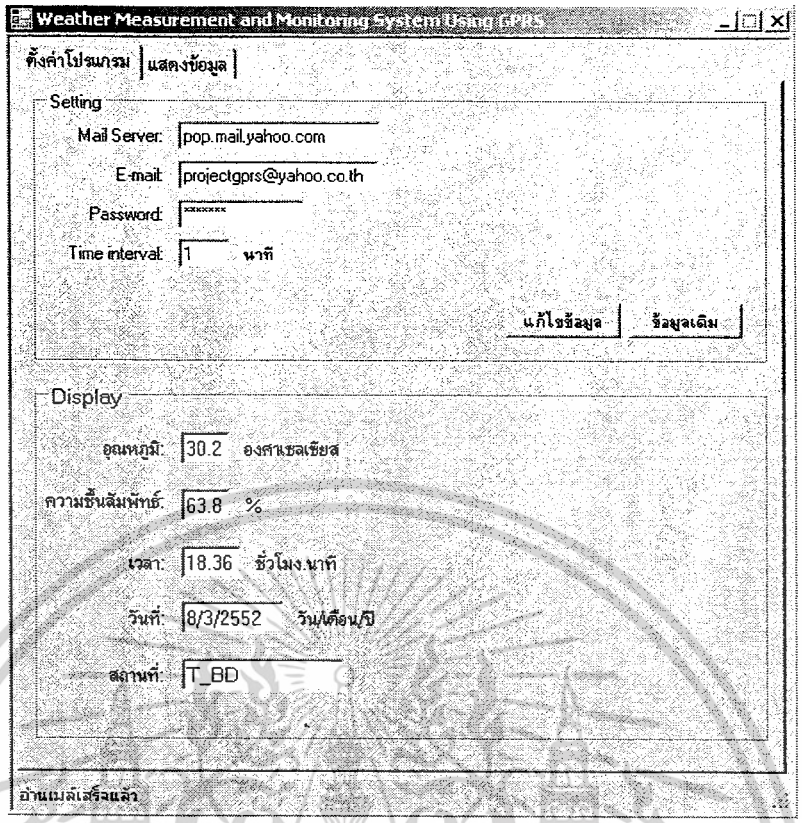
4.6 แสดงผลการทดลองจากการรันโปรแกรม

เมื่อเปิดโปรแกรมขึ้นมาจะทำการอ่าน E-mail อัตโนมัติโดยจะแสดงสถานะว่า กำลังอ่านเมล์ ทางด้านซ้ายล่างของโปรแกรม ดังรูปที่ 4.14



รูปที่ 4.14 สถานะโปรแกรมช่วงกำลังอ่าน E-mail

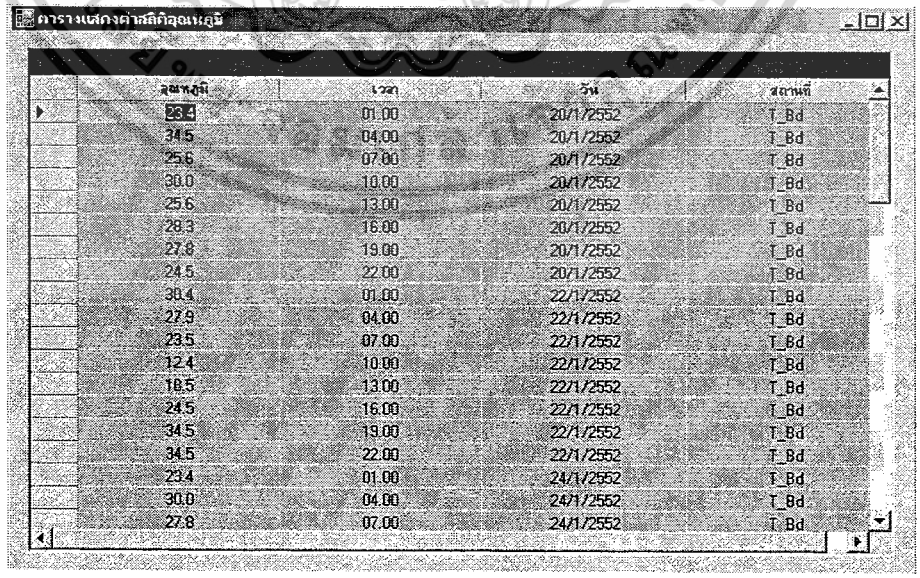
เมื่อทำการอ่าน E-mail เสร็จแล้วจะแสดงสถานะว่า อ่านเมล์เสร็จแล้ว ทางด้านซ้ายล่างของโปรแกรม ดังรูปที่ 4.15



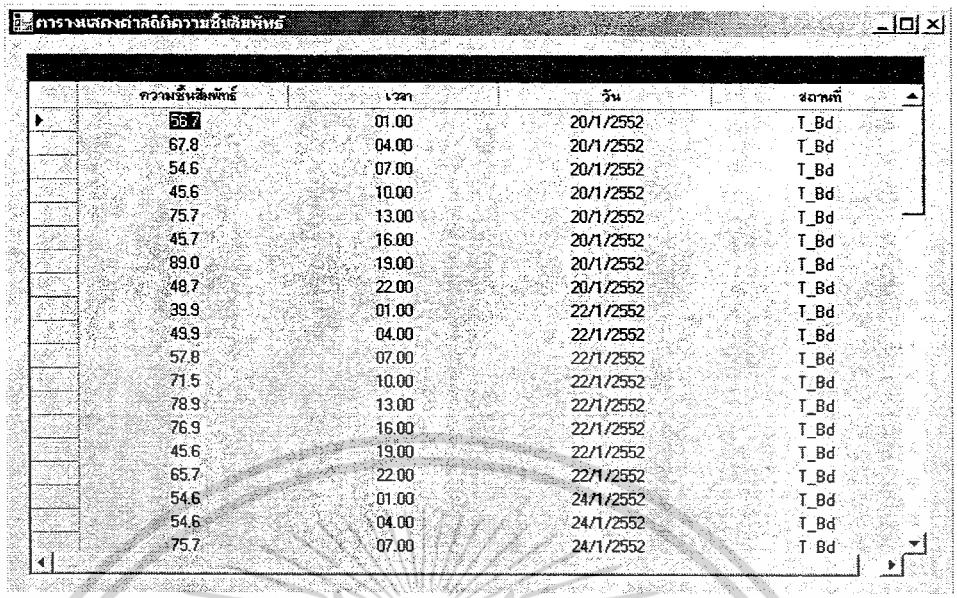
รูปที่ 4.15 สถานะโปรแกรมอ่าน E-mail เสร็จแล้ว

4.7 แสดงผลส่วนแสดงข้อมูล

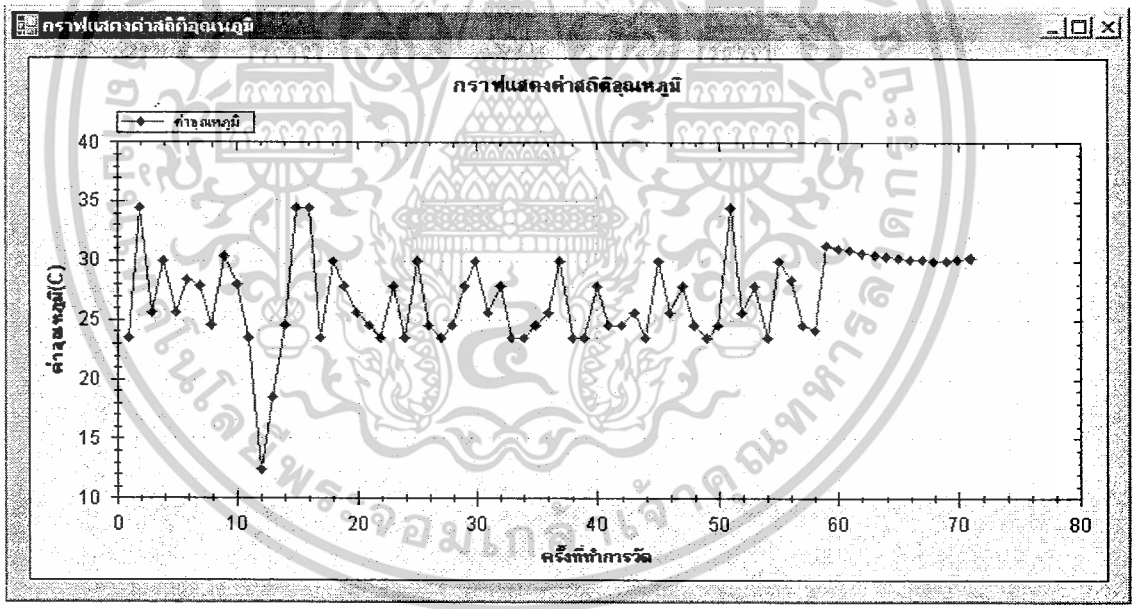
แสดงผลในรูปของตารางและกราฟค่าอุณหภูมิและความชื้นสัมพัทธ์แบบสถิติ (แสดงข้อมูลทั้งหมดที่มีในฐานข้อมูล) ดังรูปที่ 4.16 รูปที่ 4.17 รูปที่ 4.18 รูปที่ 4.19



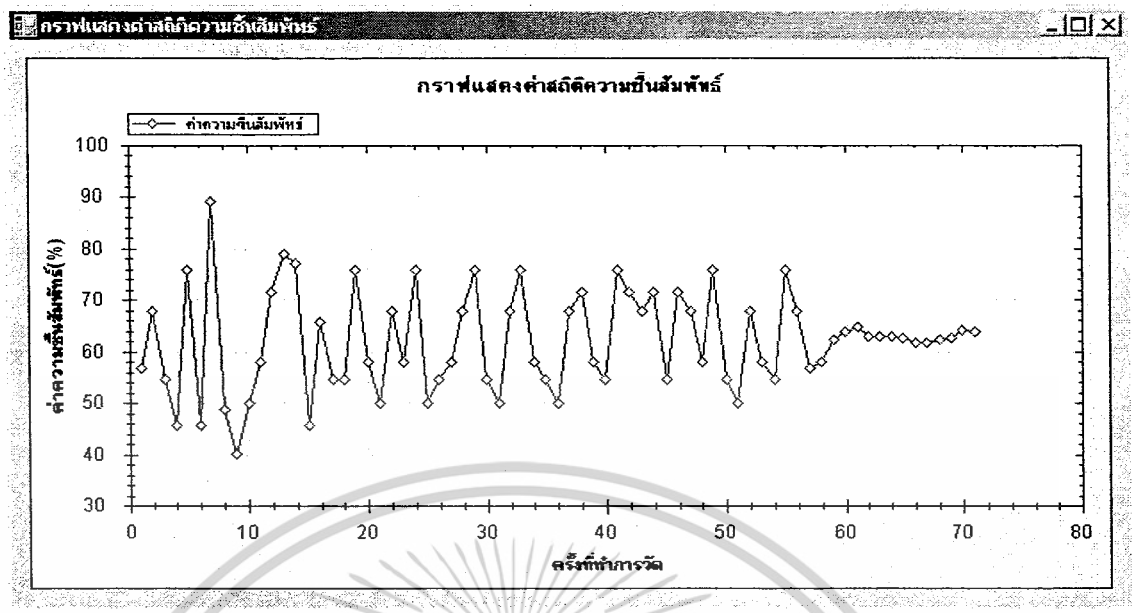
รูปที่ 4.16 ตารางแสดงค่าสถิติอุณหภูมิ



รูปที่ 4.17 ตารางแสดงค่าสถิติความชื้นสัมพัทธ์

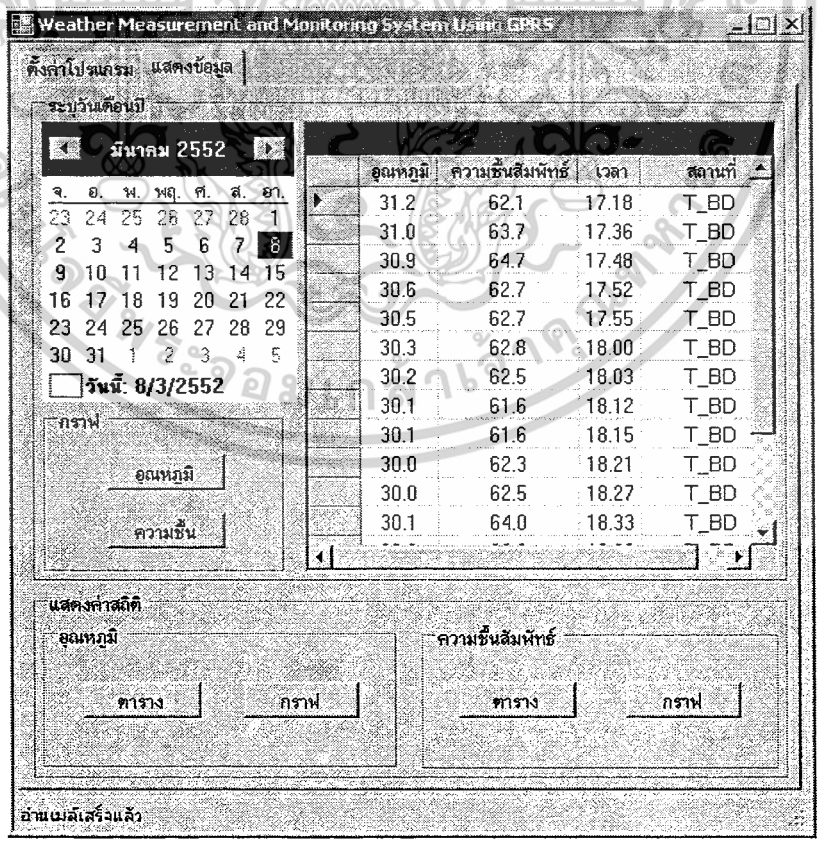


รูปที่ 4.18 กราฟแสดงค่าสถิติอุณหภูมิ

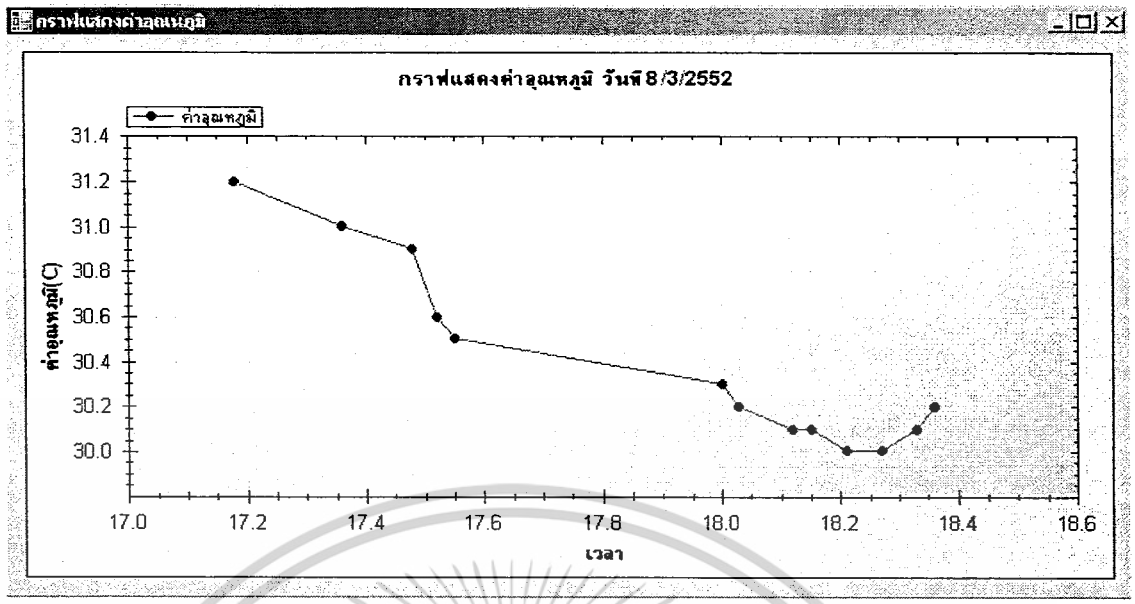


รูปที่ 4.19 กราฟแสดงค่าสถิติความชื้นสัมพัทธ์

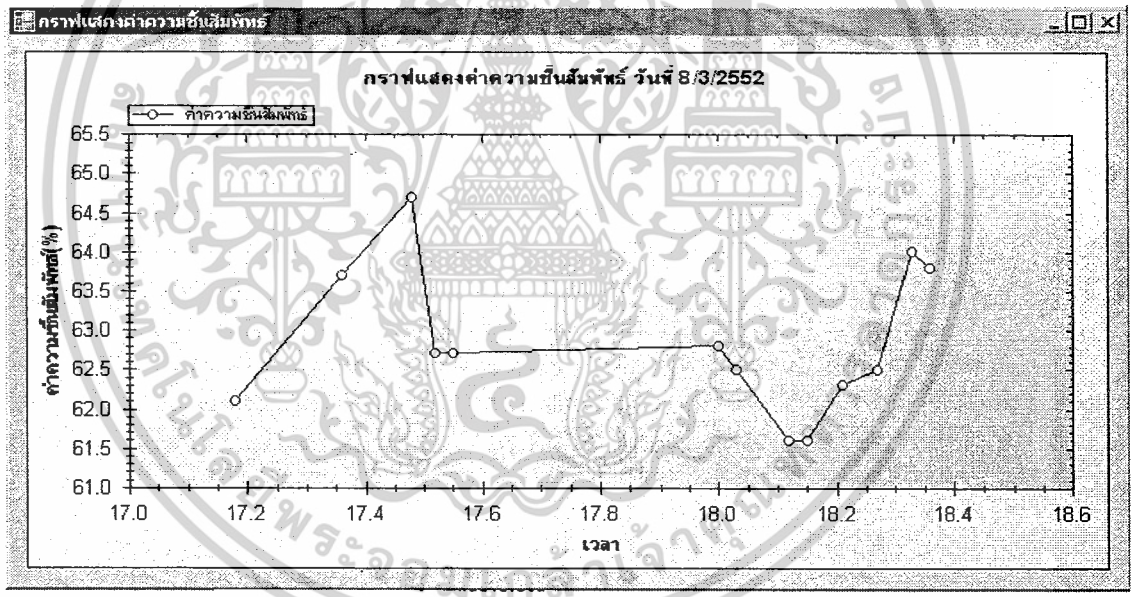
และส่วนแสดงผลในรูปของตารางและกราฟแบบเลือกวันเดือนปี ดังรูปที่ 4.20 รูปที่ 4.21 รูปที่ 4.22



รูปที่ 4.20 ตารางแสดงค่าอุณหภูมิและความชื้นสัมพัทธ์วันที่ 8/3/2552



รูปที่ 4.21 กราฟแสดงค่าอุณหภูมิวันที่ 8/3/2552



รูปที่ 4.22 กราฟแสดงค่าความชื้นสัมพัทธ์วันที่ 8/3/2552

บทที่ 5
สรุปผลและวิจารณ์การทดลอง

5.1 สรุปผล

โครงงานระบบตรวจวัดและติดตามผลข้อมูลสภาพอากาศผ่านระบบ GPRS สามารถใช้งานได้ดังนี้

- ชิ้นงานสามารถแสดงค่าอุณหภูมิ ความชื้นสัมพัทธ์ เวลา และวันเดือนปีได้
- ชิ้นงานสามารถส่ง Mail ที่ตั้งตามเวลาได้ และในระหว่างการส่งจะแสดงสถานะต่างๆ ของโมดูล TCP/IP Stack Chip
- โปรแกรมสามารถอ่าน E-mail จากกล่องรับจดหมายที่ถูกส่งเข้ามาจากชิ้นงานได้
- โปรแกรมสามารถเก็บข้อมูลที่อ่านจาก Subject ของ Mail มาเก็บเข้าฐานข้อมูลได้
- โปรแกรมสามารถนำข้อมูลที่เก็บในฐานข้อมูลมาแสดงในรูปตาราง และกราฟได้

5.2 วิจารณ์ผลการทดลอง

เราสามารถนำชิ้นงานนี้ไปใช้งานในด้านของอุตุนิยมวิทยาได้ เนื่องจากชิ้นงานนี้สามารถใช้งานได้จริง มีค่าใช้จ่ายไม่มากนักเพราะไม่ต้องสร้างระบบการสื่อสารขึ้นมาใหม่เองโดยอาศัยระบบ GPRS ที่ผู้ให้บริการโทรศัพท์เคลื่อนที่นั้นเปิดให้บริการอยู่แล้ว และค่าบริการ GPRS นั้นจะถูกกว่า SMS อยู่มาก สามารถส่งได้ไกลกว่า โมดูลตัวรับ-ส่ง เพราะสามารถใช้งานได้ทุกที่ที่มีคลื่นสัญญาณโทรศัพท์เคลื่อนที่ของผู้ให้บริการนั้นๆ

5.3 ปัญหาและอุปสรรค

- TCP/IP Stack Chip ร้อนง่าย เนื่องมาจากไม่ได้ทำส่วนระบายความร้อนของชิ้นงาน
- E-mail ที่ส่งไปแล้วบางครั้งไม่ถึงผู้รับ เนื่องจากที่ช่วงเวลานั้นๆ มีคนส่ง Mail เป็นจำนวนมาก Mail ที่เก็บอยู่ใน MTA ของ Mail Server จะทิ้ง Mail นั้นๆเมื่อถึงเวลาที่กำหนด
- E-mail ที่ส่งไปถึงกล่องรับจดหมายของผู้รับจะถูกมองเป็นอีเมล์ขยะ เนื่องจากไม่ได้ทำการลงทะเบียนกับที่ Mail Server ของผู้ให้บริการ

บรรณานุกรม

[1] ขจร อนุดิตย์, "การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษา C", Core Function, 2550.

[2] นคร ภักดีชาติ, ชัยวัฒน์ ลิ้นพรจิตรวิไล, "ไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษา C ฉบับ P89V51RD2", อินโนเวตีฟ เอ็กเพอริเมนต์, 2551.

[3] วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, "MCS-51 Microcontroller Theory & Practical Approach : Atmel AT89C5X/AT898XXXX", อินโนเวตีฟเอ็กเพอริเมนต์, 2521.

[4 อำนาจ นุตะมาน, "ติดตั้งและใช้งาน Mail Server ภาคปฏิบัติ", ซีเอ็ดยูเคชั่น, 2551.

[5] นิรุธ อำนวยศิลป์, "Network and Protocols Programming using C/C++", จ.เจริญการพิมพ์, 2548.

[6] ศุภชัย สมพานิช, "คู่มือเขียนโปรแกรมและใช้งาน Visual C# .NET ฉบับสมบูรณ์", ด่านสุทธาการพิมพ์, 2546.

[7] Behrouz A. Forouzan, "Data Communications and Networking", Top Publishing Co., Ltd., 2549.

[8] http://sourceforge.net/project/showfiles.php?group_id=114675

ภาคผนวก

```
/*---------------------------------------------------------------------*/
// Program : Weather Measurement and Monitoring System Using GPRS
// Description : Main file for Microcontroller Phillips P89V51RD2
/*---------------------------------------------------------------------*/
#include<P89V51Rx2.H>
#include<sht_15.h>
#include<ds1307.h>
#include<function.h>
value humi_val,temp_val;
xdata int ans,t_s1,rh_s1,s_t,s_rh;
xdata float rh_true, t_C;
unsigned char idata QBuf[6];

                                /*** แสดงเวลาและวันเดือนปี***/
void displaytime()
{       lcd_origin();
        lcd_command(0xC8);
        send_to_lcd(hour);
        lcd_text(':');
        send_to_lcd(min);
        lcd_text(':');
        send_to_lcd(sec);
        lcd_command(0xC0);
        send_to_lcd(date);
        send_to_lcd(month);
        send_to_lcd(year);
}

                        /***ตรวจสอบค่าเวลาปัจจุบันกับค่าเวลาที่ตั้งไว้***/

void checktime()
{       unsigned int i,k;
 code unsigned char  //time_set[]={0x00,0x05,0x10,0x15,0x20,0x25,0x30,0x35,0x40,0x45,0x50,0x55};
```

```
code unsigned char
time_set[]={0x00,0x03,0x06,0x09,0x12,0x15,0x18,0x21,0x24,0x27,0x30,0x33,0x36,0x39,0x42,0x45,0
x48,0x52,0x55,0x58};
        for(i=0;i<21;i++)
            {
            if( (min==time_set[i])&((sec==0x00)||(sec==0x01)||(sec==0x02)))
                {
                    lcd_origin();

                    for(k=0;k<3;k++)
                    {
                    lcd_puts(0xC0,"              ");
                    lcd_puts(0xC0,"Sending >>> GPRS");
                    }
                    delay(300);
                    lcd_puts(0xC0," Setting Module ");
                    delay(300);

                    Init_Serial(192);
                    printf("at+i\n");
                    delay(300);
                    lcd_puts(0xC0,"              ");
                    lcd_puts(0xC0,"  Detecting...  ");
                    delay(300);
                    printf("at+idetect\n");
                    delay(800);
                    lcd_puts(0xC0,"              ");
                    lcd_puts(0xC0,"  Detected OK!  ");

                    printf("at+isetd=*99***1#\n");
                    delay(1000);

                   lcd_puts(0xC0,"              ");
                   lcd_puts(0xC0,"   Opening...  ");
```

```
                    printf("at+iopen\n");
                    delay(3000);
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0," Open connection ");
                    delay(300);
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0,"  Sending E-mail ");
                    delay(300);
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0,"Transfering Data");
                    delay(300);
printf("at+ismtpsend=mail.cscoms.com,pup_wipcream@yahoo.co.th,projectgprs@yahoo.co.th,T_BD
%2.1f%2.1f  %bx.%bx.%bx,gprs\n",temp_val.f,humi_val.f,hour,min,sec);
printf("T.BD %5.1f%5.1f%2.2bx.%2.2bx.%2.2bx,gprs\n",temp_val.f,humi_val.f,hour,min,sec);
                    delay(8000);
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0,"   Sending OK    ");
                    delay(300);

                    printf("at+iclose\n");
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0,"Close connection");
                    delay(500);
                    lcd_puts(0xC0,"               ");
                    lcd_puts(0xC0," - - SUCCESS - - ");
                    delay(100);
                lcd_puts(0xC0,"               ");
            }
        }
    }
```

/*โปรแกรมหลัก***/**

```
void main()
```

```
{ unsigned char error,checksum;


lcd_init();
delay(100);
//DS1307_wrtime(0x06,0x10,0x00);          //ตั้งเวลาไอซี DS1307
//DS1307_wrdate(0x27,0x02,0x09);


lcd_puts(0x80,"    WEATHER ");
lcd_puts(0xC0,"MonitoringSystem");
delay(500);
lcd_puts(0x80,"     KMITL ");
lcd_puts(0xC0," TELECOM ENG#44 ");
delay(500);
lcd_command(0x01);
lcd_puts(0x80,"T   . c"); // Show temperature value
lcd_puts(0x89,"H   . %"); // Show humidity value
s_connectionreset();
while(1)
  {
    error+=s_measure((unsigned char *)&humi_val.i,&checksum,HUMI);    //วัดค่าความชื้น
    error+=s_measure((unsigned char *)&temp_val.i,&checksum,TEMP);    //วัดค่าอุณหภูมิ

          if (error!=0)s_connectionreset();
          else
          {
                  humi_val.f=(float)humi_val.i;
                  temp_val.f=(float)temp_val.i;
                  calc_sht15(&humi_val.f,&temp_val.f);
           }
           readtime();
          checktime();
          displaytime();
          }
}
```

```
/*---------------------------------------------------------------------*/
// Filename : sht_15.h
// Program : SHT15 HUMIDITY & TEMPERATURE EXAMPLE PROGRAM
// Description : Hardware        89C51RD2
/*---------------------------------------------------------------------*/
#include <intrins.h>
#include <math.h>
#include <stdio.h>
#include <LCD.h>

/***** กำหนดค่าคงที่ที่ใช้ในการติดต่อกับเซ็นเซอร์ SHT 15 ******/

#define noACK 0
#define ACK   1
#define STATUS_REG_W 0x06     //000   0011    0
#define STATUS_REG_R 0x07     //000   0011    1
#define MEASURE_TEMP 0x03     //000   0001    1
#define MEASURE_HUMI 0x05     //000   0010    1
#define RESET       0x1e      //000   1111    0

sbit    DATA = P2^0;                    // กำหนดพอร์ต 2.0 เป็นขาสัญญาณข้อมูล
sbit    SCK =   P2^1;                   // กำหนดพอร์ต 2.1 เป็นขาสัญญาณนาฬิกา

typedef union
{ unsigned int i;
  float f;
} value;

enum {TEMP,HUMI};

/*** เขียนข้อมูลลงระบบบัส I2C และ ตรวจสอบ acknowledge ***/
char s_write_byte(unsigned char value)
{
  unsigned char i,error=0;
```

```
for (i=0x80;i>0;i/=2)
{ if (i & value) DATA=1;
  else DATA=0;
  SCK=1;
  _nop_();
  _nop_();
  _nop_();
  SCK=0;
 }
 DATA=1;
 SCK=1;
 error=DATA;
 SCK=0;
 return error;
}
```

/ อ่านข้อมูลจากระบบบัส I²C และ เซ็นเซอร์ส่ง acknowledge(ack=1) เมื่อส่งข้อมูลครบ ***/***

```
char s_read_byte(unsigned char ack)
{
 unsigned char i,val=0;
 DATA=1;
 for (i=0x80;i>0;i/=2)
 { SCK=1;
   if (DATA) val=(val | i);
   SCK=0;
 }
 DATA=!ack;
 SCK=1;
 _nop_();
 _nop_();
 _nop_();
 SCK=0;
 DATA=1;
 return val;
```

```
}

/*** สร้างสัญญาณ a transmission start ***/
void s_transstart(void)

{
  DATA=1; SCK=0;
  _nop_();
  SCK=1;
  _nop_();
  DATA=0;
  _nop_();
  SCK=0;
  _nop_();
  _nop_();
  _nop_();
  SCK=1;
  _nop_();
  DATA=1;
  _nop_();
  SCK=0;
}

/*** สร้าง communication reset เพื่อรีเซตการเชื่อมต่อระหว่างเซ็นเซอร์กับไมโครคอนโทรลเลอร์***/
void s_connectionreset(void)

{
  unsigned char i;
  DATA=1; SCK=0;
  for(i=0;i<9;i++)
  { SCK=1;
    SCK=0;
  }
  s_transstart();
```

```
}

/*** ทำการวัดค่าอุณหภูมิและความชื้น***/
char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode)
{
 unsigned error=0;
 unsigned int i;

 s_transstart();
 switch(mode){
  case TEMP   : error+=s_write_byte(MEASURE_TEMP); break;
  case HUMI   : error+=s_write_byte(MEASURE_HUMI); break;
  default     : break;
 }
 for (i=0;i<65535;i++) if(DATA==0) break; //รอกระทั่งเซ็นเซอร์ทำการวัดค่าเสร็จ
 if(DATA) error+=1;
 *(p_value)  =s_read_byte(ACK);
 *(p_value+1)=s_read_byte(ACK);
 *p_checksum =s_read_byte(noACK);
 return error;
}


/*** คำนวณค่าอุณหภูมิและความชื้นจากข้อมูลดิบที่รับเข้ามา***/
void calc_sht15(float *p_humidity ,float *p_temperature)
{  code const float C1=-4.0;        // ค่าคงที่ใช้คำนวณค่าความชื้นสัมพัทธ์แบบความละเอียด 12 Bit
   code const float C2= 0.0405;
   code const float C3=-0.0000028;
   code const float T1=-0.01;       // ค่าคงที่ใช้คำนวณค่าความอุณหภูมิแบบความละเอียด 14 Bit
   code  const float T2=0.00008;
  xdata float rh=*p_humidity;          // rh: Humidity 12 Bit
  xdata float t=*p_temperature;        // t: Temperature 14 Bit
  xdata float rh_lin;                  // rh_lin:  Humidity linear
  xdata float rh_true;                 // rh_true: Temperature compensated humidity
```

```
xdata float t_C;                    // t_C   :  Temperature
xda int t_s1,rh_s1,s_t,s_rh;
t_C=t*0.01 - 40;
rh_lin=C3*rh*rh + C2*rh + C1;
rh_true=(t_C-25)*(T1+T2*rh)+rh_lin;
 if(rh_true>100)rh_true=100;
 if(rh_true<0.1)rh_true=0.1;
*p_temperature=t_C;
*p_humidity=rh_true;

/*** แสดงค่าอุณหภูมิและความชื้นสัมพัทธ์บนจอ LCD***/
lcd_init();
 lcd_command(0x01);
lcd_puts(0x80," Temp :    .   C");
 inttolcd(0x88,t_C);
 inttolcd(0x8C,t_s1);
 //lcd_puts(0xC0," Humi :    .   %");
 inttolcd(0xc8,rh_true);
 inttolcd(0xCC,rh_s1);
 //delay(700);

}
```

```
/*-------------------------------------------------------------*/
// Filename : lcd.h
// Program : Module Function for 16*2 LCD display
// Description : Library file for call Function 16*2 LCD display
/*-------------------------------------------------------------*/
#define lcd_clear() lcd_command(1)                  // Macro function for clear display LCD
#define lcd_origin() lcd_command(2)                 // Macro function for set origin LCD
#define NUMBER_OF_DIGITS 16                         // Macro function for set character byte for
convert integer

sbit e  = P3^6;                                     // Define P3.6 for Enable pin of LCD
sbit rs = P3^7;                                     // Define P3.7 for RS pin of LCD

/* Function for delay time ms scale */

void delay(unsigned int ms)
{
  unsigned int x,a;                                 // Keep for counter loop
  for(x=0;x<ms;x++)
  {
  for(a=0;a<908;a++);                               // Loop for delay 1 millisec per unit
  }
}

/*Send command 1 byte to LCD */
void lcd_command(unsigned char com)
{
  rs = 0;
  e = 1;
  P1 = com;
  delay(1);
  e = 0;
  delay(1);
```

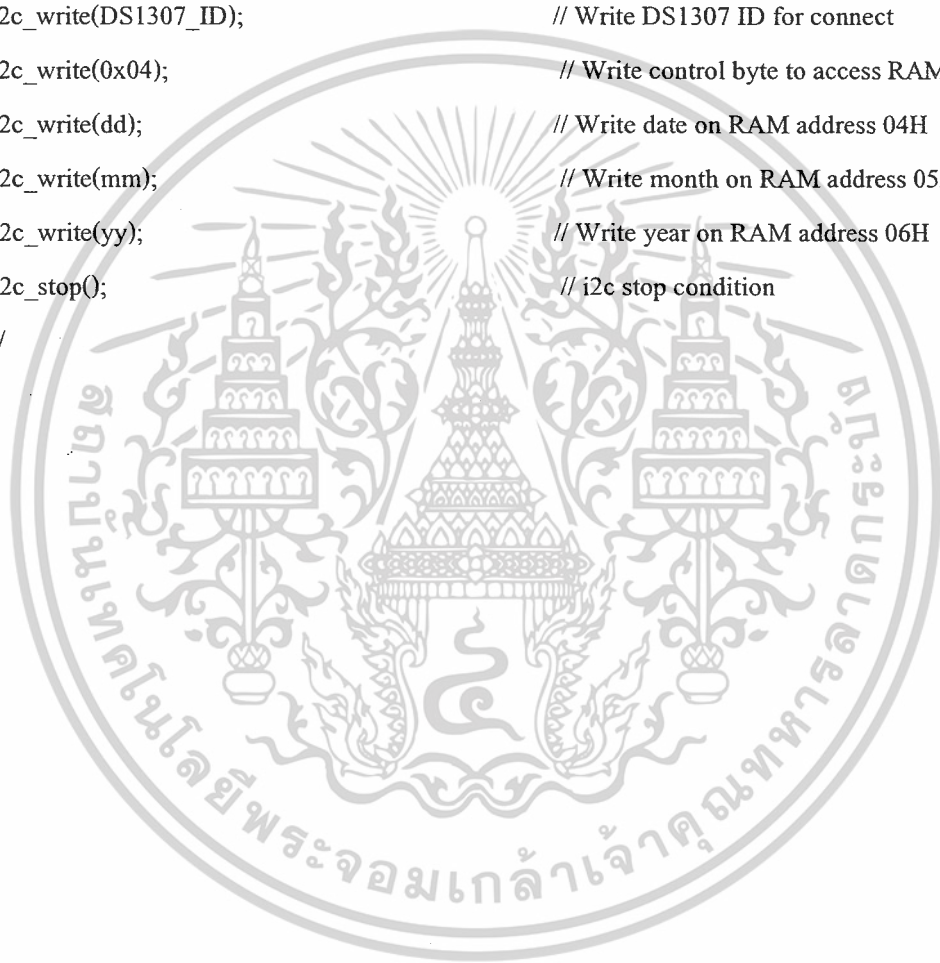
```
}

/* Send character 1 byte to LCD */
void lcd_text(unsigned char text)
{
rs = 1;
e = 1;
P1 = text;
delay(5);
e = 0;
delay(5);
}

/*Display String to LCD */
void lcd_puts(char addr, char *ptr)
{
lcd_origin();                    // Set origin LCD
lcd_command(addr);               // Set address LCD
while(*ptr)                      // check 0(NULL) condition
{
lcd_text(*ptr);                  // Send data of ptr pointer to LCD
ptr++;                           // Increase address 1 time
}
}

/* Convert long integer to ascii for display on LCD */
void _ultoa(unsigned long value, char* string, unsigned char radix)
{
unsigned char index;             // Counter of digit
char buffer[NUMBER_OF_DIGITS];   // Data buffer
index = NUMBER_OF_DIGITS;        // Load counter by digit count

do
{
```

```
buffer[--index] = '0' + (value % radix);                    // Convert configuration by radix(10 or 16)
if ( buffer[index] > '9') buffer[index] += 'A' - '9' - 1; // For base over 10(base 16)
value /= radix;                                             // Div by to calculate into next digit
} while (value != 0);                                       // End for convert?

do
{
*string++ = buffer[index++];                                // Load convert value to string buffer
} while ( index < NUMBER_OF_DIGITS );  // Over of digit count?

*string = 0;                                                // Place null for end string
}

/** Convert long integer to ascii for display on LCD */
void _ltoa(long value_l, char * string_l, unsigned char radix_l)
{
if (value_l < 0 && radix_l == 10)                           // For value < 0 (base.10)
{
*string_l++ = '-';                                          // Load sign '-' for display
_ultoa(-value_l, string_l, radix_l);                        // Convert long integer
}
else
{
_ultoa(value_l, string_l, radix_l);                         // Convert long integer
}
}

/* Function Convert integer display on LCD */
void inttolcd(unsigned char posi, int value)
{
char buff[12];                                              // For keep string send to LCD
_ltoa(value,&buff[0],10);                                   // Convert value base 10
lcd_puts(posi,&buff[0]);                                    // Send integer to LCD
}
```

```
/* Function Initial LCD */
void lcd_init()
{
delay(100);                          // Delay for initial LCD
lcd_command(0x38);                   // on display ,8 bit display ,5*7 dot
lcd_command(0x0C);                   // None cursor
lcd_command(0x01);                   // Clear screen
}
```

```
/*---------------------------------------------------------------------*/
// Filename      : ds1307.h
// Program       : real time
// Description   : Library file for real time
/*---------------------------------------------------------------------*/

#include<i2cV51.h>
#define DS1307_ID 0xD0                          // Define constant DS1307_ID
xdata unsigned char sec,min,hour,date,month,year;   // For keep Date and Time

/****************** Function read data from DS1307 ***************************/
unsigned char DS1307_rd(unsigned char addr)
{
  unsigned char ret;                            // For keep return value
  i2c_start();                                  // i2c start condition
  i2c_write(DS1307_ID);                         // Write DS1307 ID for connect
  i2c_write(addr);                              // Write RAM address on DS1307 for connect
    i2c_start();                                // i2c start condition
  i2c_write(DS1307_ID+1);                       // Write DS1307 ID for Read Mode connect
  ret = i2c_read();                             // Read data and keep to ret
  i2c_stop();                                   // i2c stop condition
  return(ret);                                  // return value
}

/************ Function write hour/min/sec on chip DS1307 *********************/
/*void DS1307_wrtime(unsigned char hh,unsigned char mm,unsigned char ss)
{
  i2c_start();                                  // i2c start condition
  i2c_write(DS1307_ID);                         // Write DS1307 ID for connect
  i2c_write(0x00);                              // Write control byte to access RAM address 00H
  i2c_write(ss);                                // Write sec on RAM address 00H
  i2c_write(mm);                                // Write min on RAM address 01H
  i2c_write(hh);                                // Write hour on RAM address 02H
  i2c_stop();                                   // i2c stop condition
```

```
} */



/********************* Function write date/month/year on chip DS1307 ***********/
/*void DS1307_wrdate(unsigned char dd,unsigned char mm,unsigned char yy)
{
  i2c_start();                          // i2c start condition
  i2c_write(DS1307_ID);                 // Write DS1307 ID for connect
  i2c_write(0x04);                      // Write control byte to access RAM address 04H
  i2c_write(dd);                        // Write date on RAM address 04H
  i2c_write(mm);                        // Write month on RAM address 05H
  i2c_write(yy);                        // Write year on RAM address 06H
  i2c_stop();                           // i2c stop condition
}*/
```



```
/*---------------------------------------------------------------------*/
// Program        : Module Function for i2c bus protocol
// Description    : Library file for call Function control i2c bus communication
// Filename       : i2c.h
/*---------------------------------------------------------------------*/
sbit SDA = P2^0;                                    // Define pin P2.0 for SDA
sbit SCL = P2^1;                                    // Define pin P2.1 for SCL

/********************** Function delay for i2c process ************************/
void i2c_delay(void)
{
  unsigned char i;
  for(i=0;i<10;i++);
}

/********************** Function generate clock for i2c bus *******************/
void i2c_clk(void)
{
  i2c_delay();
  SCL = 1;
  i2c_delay();
  SCL = 0;
}

/********************** Function i2c start condition **************************/
void i2c_start(void)
{
  if(SCL)                                           // If SCL=1 clear SCL
  SCL = 0;                                          // Clear SCL
  SDA = 1;                                          // Set SDA
  SCL = 1;                                          // Set SCL
  i2c_delay();                                      // Delay
  SDA = 0;                                          // Clear SDA
  i2c_delay();                                      // Delay
```

```
  SCL = 0;                                  // Clear SCL
}

/********************* Function i2c stop condition ***************************/
void i2c_stop(void)
{
  if(SCL)                                   // If SCL=1 clear SCL
  SCL = 0;                                  // Clear SCL
  SDA = 0;                                  // Clear SDA
  i2c_delay();                              // Delay
  SCL = 1;                                  // Set SCL
  i2c_delay();                              // Delay
  SDA = 1;                                  // Set SDA
}

/********************* Function write command 1 byte via i2c bus ************/
bit i2c_write(unsigned char dat)
{
  bit data_bit;                             // Bit keep data send to i2c bus
  unsigned char i;                          // Variable for counter
  for(i=0;i<8;i++)                          // For loop 8 time(send data 1 byte)
  {
    data_bit = dat & 0x80;                  // Filter MSB bit keep to data_bit
    SDA = data_bit;                         // Send data_bit to SDA
    i2c_clk();                              // Call for send data to i2c bus
    dat = dat<<1;                           // Shift left 1 time
  }
  SDA = 1;                                  // Set SDA
  i2c_delay();                              // Delay
  SCL = 1;                                  // Set SCL
  i2c_delay();                              // Delay
  data_bit = SDA;                           // For check acknowledge
  SCL = 0;                                  // Clear SCL
  i2c_delay();                              // Delay
```

```
  return(data_bit);                          // If send_bit = 0 i2c OK!
}

/********************** Function read data 1 byte via i2c bus *****************/
unsigned char i2c_read(void)
{
  bit rd_bit;                                // Bit read data from i2c bus
  unsigned char i,dat;                       // For keep counter and data receive
  dat = 0x00;                                // Clear data
  for(i=0;i<8;i++)                           // For loop read data 1 byte
  {
    i2c_delay();                             // Delay
    SCL = 1;                                 // Set SCL
    i2c_delay();                             // Delay
    rd_bit = SDA;                            // Keep for check acknowledge
    dat = dat<<1;                            // Shift left 1 time
    dat = dat | rd_bit;                      // Keep bit data in dat
    SCL = 0;                                 // Clear SCL
  }
  return(dat);
}

/********************** Function Acknowladge by master ************************/
/*void i2c_ACK()
{
  SDA = 0;                                   // Clear SDA
  i2c_delay();                               // Delay
  i2c_clk();                                 // Call for send data to i2c bus
  SDA = 1;                                   // Set SDA
}  */

/********************** Function No-Acknowladge by master *********************/
/*void i2c_NACK()
{
```

```
  SDA = 1;                    // Set SDA
  i2c_delay();                // Delay
  i2c_clk();                  // Call for send data to i2c bus
  SCL = 1;                    // Set SCL
} */
```

frmMain.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using System.Data.OleDb;
using System.Threading;

namespace Pop3ToDB
{
   public partial class frmMain : Form
   {
      private string pop_server, username, password;
      private int TimeInterval;

      private string time(string wordss)
      {
         string newsplit0 = "", newsplit1 = "";
         string[] split = wordss.Split(new Char[] { '.' });
         if (split[0].Length == 1)
         {
            newsplit0 = "0" + split[0] + "";
         }
         else if (split[0].Length == 2)
         {
            newsplit0 = split[0];
         }
         if (split[1].Length == 1)
         {
```

```
        newsplit1 = "0" + split[1] + "";
      }
      else if (split[1].Length == 2)
      {
        newsplit1 = split[1];
      }
      string wordTime = "" + newsplit0 + "." + newsplit1 + "";
      return wordTime;
    }

    private void ReadMail()
    {
      sttbarLabel1.Text = "กำลังอ่านเมล์";
      try
      {
        Pop3 obj = new Pop3();
        obj.Connect(pop_server, username, password);
        ArrayList list = obj.List();
        DB MyDB = new DB();
        foreach (Pop3Message msg in list)
        {
          Pop3Message msg2 = obj.Retrieve(msg);
          string temp = msg2.message.Replace('\n', ' ');
          temp = temp.Replace('\r', ' ');
          string[] WordArray = temp.Split(new char[] { ' ' });
          int x = 0;
          foreach (string Words in WordArray)
          {
            x++;
            if (Words == "Subject:")
            {
              if (MyDB.ConnectDB())
              {
```

```
                    MyDB.QueryDB("INSERT INTO [data] ( Place, Temp, Humi, Qty, OrderDate )
VALUES ('" + WordArray[x] + "', '" + WordArray[x + 1] + "', '" + WordArray[x + 2] + "', '" +
time(WordArray[x + 3]) + "', '" + DateTime.Today.ToShortDateString() + "');");
                    MyDB.CloseDB();
                    obj.Delete(msg);
                  }
                  break;
                }
              }
            }
            obj.Disconnect();
          }
          catch (Pop3Exception ee)
          {
            MessageBox.Show(ee.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
          }
          sttbarLabel1.Text = "อ่านเมล์เสร็จแล้ว";
        }

        private void DisplaySetting()
        {
          DB MyDB = new DB();
          if (MyDB.ConnectDB())
          {
            MyDB.QueryDB("SELECT * FROM Setting;");
            txtPOPServer.Text = MyDB.pDataTable.Rows[0][0].ToString();
            txtUsername.Text = MyDB.pDataTable.Rows[0][1].ToString();
            txtPassword.Text = MyDB.pDataTable.Rows[0][2].ToString();
            txtInterval.Text = MyDB.pDataTable.Rows[0][3].ToString();
            MyDB.CloseDB();
          }
        }

        private void btnSettingCancel_Click(object sender, EventArgs e)
```

```
{
    DisplaySetting();
}

private void DisplayData()
{
    try
    {
        DB MyDB = new DB();
        if (MyDB.ConnectDB())
        {
            MyDB.QueryDB("SELECT * FROM data;");
            double[] arrTemp = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
            txtTemp.Text = MyDB.pDataTable.Rows[arrTemp.Length - 1][2].ToString();
            double[] arrHumi = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
            txtHumi.Text = MyDB.pDataTable.Rows[arrHumi.Length - 1][3].ToString();
            double[] arrTime = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
            txtTime.Text = MyDB.pDataTable.Rows[arrTime.Length - 1][4].ToString();
            double[] arrDate = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
            txtDate.Text = MyDB.pDataTable.Rows[arrDate.Length - 1][5].ToString();
            double[] arrPlace = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
            txtPlace.Text = MyDB.pDataTable.Rows[arrPlace.Length - 1][1].ToString();
            MyDB.CloseDB();
        }
    }
    catch
    {
    }
}

private void timer2_Tick(object sender, EventArgs e)
{
    DisplayData();
}
```

```
private int mintosec(string p)
{
   int ret;
   ret = int.Parse(p) * 60000;
   return ret;
}

public frmMain()
{
   InitializeComponent();
}

private void frmMain_Load(object sender, EventArgs e)
{
   DB MyDB = new DB();
   if (MyDB.ConnectDB())
   {
      MyDB.QueryDB("SELECT * FROM Setting;");
      pop_server = MyDB.pDataTable.Rows[0][0].ToString();
      username = MyDB.pDataTable.Rows[0][1].ToString(); ;
      password = MyDB.pDataTable.Rows[0][2].ToString();
      TimeInterval = mintosec(MyDB.pDataTable.Rows[0][3].ToString());
      MyDB.CloseDB();
   }
   try
   {
      timer1.Interval = TimeInterval;
   }
   catch
   {
      MessageBox.Show("กรุณาใส่ค่า TimeInterval หรือ ห้ามเติม ศูนย์");
   }
   ThreadStart threadDelegate = new ThreadStart(ReadMail);
```

```
        Thread newThread = new Thread(threadDelegate);
        newThread.Start();
        timer1.Enabled = true;
        DisplaySetting();
        timer2.Enabled = true;
    }

    private void frmMain_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (MessageBox.Show("ยืนยันการออกจากโปรแกรม?", "ยืนยัน", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
            e.Cancel = false;
        else
        {
            timer1.Enabled = false;
            e.Cancel = true;
        }
    }

    private void btnSettingOK_Click(object sender, EventArgs e)
    {
        DB MyDB = new DB();
        if (MyDB.ConnectDB())
        {
            MyDB.QueryDB("UPDATE Setting SET Setting.PopServer = '" + txtPOPServer.Text + "',
Setting.Username = '" + txtUsername.Text + "', Setting.[Password] = '" + txtPassword.Text + "',
Setting.TimeInterval = " + txtInterval.Text + ";");
            MyDB.CloseDB();
            MessageBox.Show("ทำการบันทึกเรียบร้อยแล้ว.", "แจ้ง", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
```

```
{
    ThreadStart threadDelegate = new ThreadStart(ReadMail);
    Thread newThread = new Thread(threadDelegate);
    newThread.Start();
}

private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)
{
    DB MyDB = new DB();
    if (MyDB.ConnectDB())
    {
        MyDB.QueryDB("SELECT Place, Temp, Humi, Qty FROM data WHERE OrderDate ='" +
monthCalendar1.SelectionStart.ToShortDateString() + "';");
        gridSelect.DataSource = MyDB.pDataSet;
        gridSelect.AllowNavigation = false;
        gridSelect.DataMember = "Query";
        MyDB.QueryDB("DELETE FROM buffer;");
        MyDB.QueryDB("SELECT Place, Temp, Humi, Qty, OrderDate FROM data WHERE
OrderDate ='" + monthCalendar1.SelectionStart.ToShortDateString() + "';");
        double[] arrTemp = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
        for (int i = 0; i < arrTemp.Length; i++)
        {
            arrTemp[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Temp"]);
        }
        double[] arrHumi = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];
        for (int i = 0; i < arrHumi.Length; i++)
        {
            arrHumi[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Humi"]);
        }
        string[] arrQty = new string[MyDB.pDataSet.Tables["Query"].Rows.Count];
        for (int i = 0; i < arrQty.Length; i++)
        {
            arrQty[i] = Convert.ToString(MyDB.pDataSet.Tables["Query"].Rows[i]["Qty"]);
        }
```

```
            string[] arrPlace = new string[MyDB.pDataSet.Tables["Query"].Rows.Count];
            for (int i = 0; i < arrPlace.Length; i++)
            {
               arrPlace[i] = Convert.ToString(MyDB.pDataSet.Tables["Query"].Rows[i]["Place"]);
            }

            string[] arrOrderDate = new string[MyDB.pDataSet.Tables["Query"].Rows.Count];
            for (int i = 0; i < arrPlace.Length; i++)
            {
               arrOrderDate[i] =
Convert.ToString(MyDB.pDataSet.Tables["Query"].Rows[i]["OrderDate"]);
            }

            for (int i = 0; i < arrTemp.Length; i++)
            {
               MyDB.QueryDB("INSERT INTO [buffer] ( Place, Temp, Humi, Qty, OrderDate )
VALUES ('" + arrPlace[i] + "', '" + arrTemp[i] + "', '" + arrHumi[i] + "', '" + arrQty[i] + "', '" +
arrOrderDate[i] + "');");
            }
            MyDB.CloseDB();
         }
      }

      private void btnTemptb_Click(object sender, EventArgs e)
      {
         tbTemp tbt = new tbTemp();
         tbt.Show();
         tbt.Visible = true;
      }

      private void btnHumitb_Click(object sender, EventArgs e)
      {
         tbHumi tbh = new tbHumi();
         tbh.Show();
```

```
        tbh.Visible = true;
    }

    private void btnHumigr_Click(object sender, EventArgs e)
    {
        grHumi grh = new grHumi();
        grh.Show();
        grh.Visible = true;
    }

    private void btnTempgr_Click(object sender, EventArgs e)
    {
        grTemp grt = new grTemp();
        grt.Show();
        grt.Visible = true;
    }

    private void btnHumiD_Click(object sender, EventArgs e)
    {
        grHumiD grh = new grHumiD();
        grh.Show();
        grh.Visible = true;
    }

    private void btnTempD_Click(object sender, EventArgs e)
    {
        grTempD grt = new grTempD();
        grt.Show();
        grt.Visible = true;
    }
  }
}
```

POP3.cs

```
using System;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Net.Sockets;
using System.Diagnostics;

namespace Pop3ToDB
{
   public class Pop3Exception : System.ApplicationException
   {
      public Pop3Exception(string str): base(str)
      {
      }
   }

   public class Pop3Message
   {
      public long number;
      public long bytes;
      public bool retrieved;
      public string message;
   }

   public class Pop3 : System.Net.Sockets.TcpClient
   {
      public void Connect(string server, string username, string password)
      {
         string message;
         string response;
```

```
    Connect(server, 110);

    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
       throw new Pop3Exception(response);
    }

    message = "USER " + username + "\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
       throw new Pop3Exception(response);
    }

    message = "PASS " + password + "\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
       throw new Pop3Exception(response);
    }
}

public void Disconnect()
{
    string message;
    string response;
    message = "QUIT\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
```

```
        throw new Pop3Exception(response);
    }
}

public ArrayList List()
{
    string message;
    string response;

    ArrayList retval = new ArrayList();
    message = "LIST\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
        throw new Pop3Exception(response);
    }

    while (true)
    {
        response = Response();
        if (response == ".\r\n")
        {
            return retval;
        }
        else
        {
            Pop3Message msg = new Pop3Message();
            char[] seps = { ' ' };
            string[] values = response.Split(seps);
            msg.number = Int32.Parse(values[0]);
            msg.bytes = Int32.Parse(values[1]);
            msg.retrieved = false;
            retval.Add(msg);
```

```
            continue;
        }
    }
}

public Pop3Message Retrieve(Pop3Message rhs)
{
    string message;
    string response;

    Pop3Message msg = new Pop3Message();
    msg.bytes = rhs.bytes;
    msg.number = rhs.number;

    message = "RETR " + rhs.number + "\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
        throw new Pop3Exception(response);
    }

    msg.retrieved = true;
    while (true)
    {
        response = Response();
        if (response == ".\r\n")
        {
            break;
        }
        else
        {
            msg.message += response;
        }
```

```
    }
    return msg;
}

public void Delete(Pop3Message rhs)
{
    string message;
    string response;

    message = "DELE " + rhs.number + "\r\n";
    Write(message);
    response = Response();
    if (response.Substring(0, 3) != "+OK")
    {
        throw new Pop3Exception(response);
    }
}

private void Write(string message)
{
    System.Text.ASCIIEncoding en = new System.Text.ASCIIEncoding();

    byte[] WriteBuffer = new byte[1024];
    WriteBuffer = en.GetBytes(message);

    NetworkStream stream = GetStream();
    stream.Write(WriteBuffer, 0, WriteBuffer.Length);

    Debug.WriteLine("WRITE:" + message);
}

private string Response()
{
    System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
```

```
        byte[] serverbuff = new Byte[1024];
        NetworkStream stream = GetStream();
        int count = 0;
        while (true)
        {
          byte[] buff = new Byte[2];
          int bytes = stream.Read(buff, 0, 1);
          if (bytes == 1)
          {
            serverbuff[count] = buff[0];
            count++;

            if (buff[0] == '\n')
            {
              break;
            }
          }
          else
          {
            break;
          };
        };

        string retval = enc.GetString(serverbuff, 0, count);
        Debug.WriteLine("READ:" + retval);
        return retval;
      }
    }
}
```

DB.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Pop3ToDB
{
   class DB
   {
      private OleDbConnection Conn = new OleDbConnection();
      private OleDbDataAdapter pDataAdapter = null;
      public DataTable pDataTable = null;
      public DataSet pDataSet = null;
      private string DBPath = "C:/POP3toDB/DatabasePRO.mdb";

      public bool ConnectDB()
      {
         string strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + DBPath + ";Jet
OLEDB:Engine Type=5";
         if (Conn.State == ConnectionState.Open)
            Conn.Close();
         try
         {
            Conn = new OleDbConnection(strConn);
            Conn.Open();
         }
         catch
         {
            MessageBox.Show("ไม่สามารถเชื่อต่อฐานข้อมูล " + DBPath + " ได้.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        return false;
    }
    return true;
}

public void QueryDB(string SQL)
{
    string cmdType = SQL.Substring(0, 3).ToUpper();
    OleDbCommand dbCMD = new OleDbCommand(SQL);
    pDataAdapter = new OleDbDataAdapter();
    pDataTable = new DataTable();
    pDataSet = new DataSet();
    switch (cmdType)
    {
        case "SEL":
            pDataAdapter.SelectCommand = dbCMD;
            pDataAdapter.SelectCommand.Connection = Conn;
            pDataAdapter.SelectCommand.ExecuteScalar();
            pDataAdapter.Fill(pDataTable);
            pDataAdapter.Fill(pDataSet, "Query");
            break;
        case "INS":
            pDataAdapter.InsertCommand = dbCMD;
            pDataAdapter.InsertCommand.Connection = Conn;
            pDataAdapter.InsertCommand.ExecuteScalar();
            break;
        case "UPD":
            pDataAdapter.UpdateCommand = dbCMD;
            pDataAdapter.UpdateCommand.Connection = Conn;
            pDataAdapter.UpdateCommand.ExecuteScalar();
            break;
        case "DEL":
            pDataAdapter.DeleteCommand = dbCMD;
            pDataAdapter.DeleteCommand.Connection = Conn;
```

```
                pDataAdapter.DeleteCommand.ExecuteScalar();
                break;
            }
        }

        public void CloseDB()
        {
            pDataTable.Clear();
            Conn.Close();
        }
    }
}
```

tbTemp.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;

namespace Pop3ToDB
{
    public partial class tbTemp : Form
    {
        public tbTemp()
        {
            InitializeComponent();
        }

        private void tbTemp_Load(object sender, EventArgs e)
        {
            DB MyDB = new DB();
            if(MyDB.ConnectDB())
            {
                MyDB.QueryDB("SELECT * FROM data");
            }
            DataGrid1.DataSource = MyDB.pDataSet.Tables["Query"];

            CustomDatagrid();
        }

        private void CustomDatagrid()
```

```
{
  DataGridTableStyle grdTS = new DataGridTableStyle();
  grdTS.MappingName = "Query";
  grdTS.BackColor = Color.LightSalmon;
  grdTS.AlternatingBackColor = Color.LightSalmon;

  DataGridColumnStyle cs = new DataGridTextBoxColumn();
  cs.MappingName = "Temp";
  cs.HeaderText = " อุณหภูมิ";
  cs.Width = 150;
  cs.Alignment = HorizontalAlignment.Center;
  grdTS.GridColumnStyles.Add(cs);

  DataGridColumnStyle cs1 = new DataGridTextBoxColumn();
  cs1.MappingName = "Qty";
  cs1.HeaderText = " เวลา";
  cs1.Width = 150;
  cs1.Alignment = HorizontalAlignment.Center;
  grdTS.GridColumnStyles.Add(cs1);

  DataGridColumnStyle cs2 = new DataGridTextBoxColumn();
  cs2.MappingName = "OrderDate";
  cs2.HeaderText = " วัน";
  cs2.Width = 150;
  cs2.Alignment = HorizontalAlignment.Center;
  grdTS.GridColumnStyles.Add(cs2);

  DataGridColumnStyle cs3 = new DataGridTextBoxColumn();
  cs3.MappingName = "Place";
  cs3.HeaderText = " สถานที่";
  cs3.Width = 150;
  cs3.Alignment = HorizontalAlignment.Center;
  grdTS.GridColumnStyles.Add(cs3);
```

```
            DataGrid1.TableStyles.Add(grdTS);
        }

        private void SetSize()
        {
            DataGrid1.Location = new Point(10, 10);
            DataGrid1.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
        }

        private void tbTemp_Resize(object sender, EventArgs e)
        {
            SetSize();
        }
    }
}
```

tbHumi.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;

namespace Pop3ToDB
{
    public partial class tbHumi : Form
    {
        public tbHumi()
        {
            InitializeComponent();
        }

        private void tbHumi_Load(object sender, EventArgs e)
        {
            DB MyDB = new DB();
            if (MyDB.ConnectDB())
            {
                MyDB.QueryDB("SELECT * FROM data;");
            }
            DataGrid2.DataSource = MyDB.pDataSet.Tables["Query"];

            CustomDatagrid();
        }

        private void CustomDatagrid()
```

```
{
    DataGridTableStyle grdTS = new DataGridTableStyle();
    grdTS.MappingName = "Query";
    grdTS.BackColor = Color.LightBlue;
    grdTS.AlternatingBackColor = Color.LightBlue;

    DataGridColumnStyle cs = new DataGridTextBoxColumn();
    cs.MappingName = "Humi";
    cs.HeaderText = " ความชื้นสัมพัทธ์";
    cs.Width = 150;
    cs.Alignment = HorizontalAlignment.Center;
    grdTS.GridColumnStyles.Add(cs);

    DataGridColumnStyle cs1 = new DataGridTextBoxColumn();
    cs1.MappingName = "Qty";
    cs1.HeaderText = " เวลา";
    cs1.Width = 150;
    cs1.Alignment = HorizontalAlignment.Center;
    grdTS.GridColumnStyles.Add(cs1);

    DataGridColumnStyle cs2 = new DataGridTextBoxColumn();
    cs2.MappingName = "OrderDate";
    cs2.HeaderText = " วัน";
    cs2.Width = 150;
    cs2.Alignment = HorizontalAlignment.Center;
    grdTS.GridColumnStyles.Add(cs2);

    DataGridColumnStyle cs3 = new DataGridTextBoxColumn();
    cs3.MappingName = "Place";
    cs3.HeaderText = " สถานที่";
    cs3.Width = 150;
    cs3.Alignment = HorizontalAlignment.Center;
    grdTS.GridColumnStyles.Add(cs3);
```

```
        DataGrid2.TableStyles.Add(grdTS);
    }

    private void SetSize()
    {
        DataGrid2.Location = new Point(10, 10);
        DataGrid2.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
    }

    private void tbHumi_Resize(object sender, EventArgs e)
    {
        SetSize();
    }
  }
}
```

grTemp.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Data.OleDb;

namespace Pop3ToDB
{
    public partial class grTemp : Form
    {
        public grTemp()
        {
            InitializeComponent();
        }

        double[] arr;

        private void grTemp_Load(object sender, EventArgs e)
        {
            {
                CreateGraph(zgl);
                SetSize();
                timer1.Enabled = true;
            }
        }

        private void CreateGraph(ZedGraphControl zgc)
```

```
{
   DB MyDB = new DB();
   if (MyDB.ConnectDB())
   {
      MyDB.QueryDB("SELECT * FROM data;");
   }

   arr = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

   for (int i = 0; i < arr.Length; i++)
   {
      arr[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Temp"]);
   }

   GraphPane myPane = zgc.GraphPane;

   myPane.Title.Text = "กราฟแสดงค่าสถิติอุณหภูมิ";
   myPane.XAxis.Title.Text = "ครั้งที่ทำการวัด";
   myPane.YAxis.Title.Text = "ค่าอุณหภูมิ(C)";

   PointPairList list = new PointPairList();

   for (int x = 0; x < arr.Length; x++)
   {
      list.Add(x + 1, arr[x]);
   }

   myPane.XAxis.MajorGrid.IsVisible = true;
   myPane.YAxis.MajorGrid.IsVisible = true;
   myPane.XAxis.MajorGrid.Color = Color.LightGray;
   myPane.YAxis.MajorGrid.Color = Color.LightGray;

   LineItem myCurve = myPane.AddCurve("ค่าอุณหภูมิ", list, Color.Red, SymbolType.Diamond);
   myCurve.Symbol.Fill = new Fill(Color.Black);
```

```
        myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);
        myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);
        zgc.AxisChange();
    }

    private void timer1_Tick_1(object sender, EventArgs e)
    {
        CreateGraph(zgl);
    }

    private void SetSize()
    {
        zgl.Location = new Point(10, 10);
        zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
    }

    private void grTemp_Resize(object sender, EventArgs e)
    {
        SetSize();
    }
  }
}
```

grHumi.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Data.OleDb;

namespace Pop3ToDB
{
   public partial class grHumi : Form
   {
      public grHumi()
      {
         InitializeComponent();
      }

      double[] arr;

      private void grHumi_Load(object sender, EventArgs e)
      {
         {
            CreateGraph(zgl);
            SetSize();
            timer1.Enabled = true;
         }
      }

      private void CreateGraph(ZedGraphControl zgc)
```

```
{
    DB MyDB = new DB();
    if (MyDB.ConnectDB())
    {
        MyDB.QueryDB("SELECT * FROM data;");
    }

    arr = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

    for (int i = 0; i < arr.Length; i++)
    {
        arr[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Humi"]);
    }

    GraphPane myPane = zgc.GraphPane;

    myPane.Title.Text = "กราฟแสดงค่าสถิติความชื้นสัมพัทธ์";
    myPane.XAxis.Title.Text = "ครั้งที่ทำการวัด";
    myPane.YAxis.Title.Text = "ค่าความชื้นสัมพัทธ์(%)";

    PointPairList list = new PointPairList();

    for (int x = 0; x < arr.Length; x++)
    {
        list.Add(x + 1, arr[x]);
    }

    myPane.XAxis.MajorGrid.IsVisible = true;
    myPane.YAxis.MajorGrid.IsVisible = true;
    myPane.XAxis.MajorGrid.Color = Color.LightGray;
    myPane.YAxis.MajorGrid.Color = Color.LightGray;

    LineItem myCurve = myPane.AddCurve("ค่าความชื้นสัมพัทธ์", list, Color.Blue,
SymbolType.Diamond);
```

```
            myCurve.Symbol.Fill = new Fill(Color.White);
            myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);
            myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);
            zgc.AxisChange();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            CreateGraph(zgl);
        }

        private void SetSize()
        {
            zgl.Location = new Point(10, 10);
            zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
        }

        private void grHumi_Resize(object sender, EventArgs e)
        {
            SetSize();
        }
    }
}
```

grTempD.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Data.OleDb;

namespace Pop3ToDB
{
   public partial class grTempD : Form
   {
      public grTempD()
      {
         InitializeComponent();
      }

      double[] arrTime,arrTemp;

      private void grTempD_Load(object sender, EventArgs e)
      {
         {
            CreateGraph(zgl);
            SetSize();
         }
      }

      private void CreateGraph(ZedGraphControl zgc)
      {
```

```
DB MyDB = new DB();
if (MyDB.ConnectDB())
{
   MyDB.QueryDB("SELECT * FROM buffer;");
}

arrTemp = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

for (int i = 0; i < arrTemp.Length; i++)
{
   arrTemp[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Temp"]);
}

arrTime = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

for (int i = 0; i < arrTime.Length; i++)
{
   arrTime[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Qty"]);
}

GraphPane myPane = zgc.GraphPane;

try
{
   myPane.Title.Text = "กราฟแสดงค่าอุณหภูมิ วันที่ " + MyDB.pDataTable.Rows[0][5] + "";
}
catch
{
}

myPane.XAxis.Title.Text = "เวลา";
myPane.YAxis.Title.Text = "ค่าอุณหภูมิ(C)";

PointPairList list = new PointPairList();
```

```
        for (int x = 0; x < arrTemp.Length; x++)
        {
           list.Add(arrTime[x], arrTemp[x]);
        }

        myPane.XAxis.MajorGrid.IsVisible = true;
        myPane.YAxis.MajorGrid.IsVisible = true;
        myPane.XAxis.MajorGrid.Color = Color.LightGray;
        myPane.YAxis.MajorGrid.Color = Color.LightGray;

        LineItem myCurve = myPane.AddCurve( "ค่าอุณหภูมิ", list, Color.Red,SymbolType.Circle );
        myCurve.Symbol.Fill = new Fill(Color.Black);
        myPane.Chart.Fill = new Fill(Color.White, Color.LightPink, 45F);
        myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);
        zgc.AxisChange();
      }

      private void SetSize()
      {
        zgl.Location = new Point(10, 10);
        zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
      }

      private void grTempD_Resize(object sender, EventArgs e)
      {
        SetSize();
      }
    }
}
```

grHumiD.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Data.OleDb;

namespace Pop3ToDB
{
   public partial class grHumiD : Form
   {
      public grHumiD()
      {
         InitializeComponent();
      }

      double[] arrHumi,arrTime;

      private void grHumiD_Load(object sender, EventArgs e)
      {
         {
            CreateGraph(zgl);
            SetSize();
         }
      }

      private void CreateGraph(ZedGraphControl zgc)
      {
```

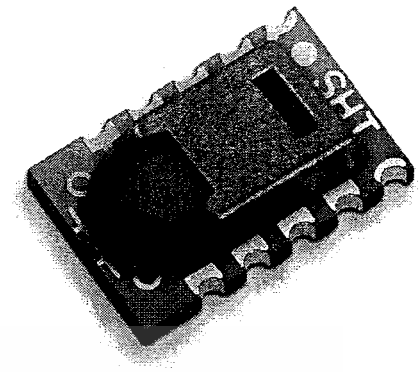
```
DB MyDB = new DB();
if (MyDB.ConnectDB())
{
  MyDB.QueryDB("SELECT * FROM buffer;");
}

arrHumi = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

for (int i = 0; i < arrHumi.Length; i++)
{
  arrHumi[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Humi"]);
}

arrTime = new double[MyDB.pDataSet.Tables["Query"].Rows.Count];

for (int i = 0; i < arrTime.Length; i++)
{
  arrTime[i] = Convert.ToDouble(MyDB.pDataSet.Tables["Query"].Rows[i]["Qty"]);
}

GraphPane myPane = zgc.GraphPane;

try
{
  myPane.Title.Text = "กราฟแสดงค่าความชื้นสัมพัทธ์ วันที่ " + MyDB.pDataTable.Rows[0][5]
+ "";
}
catch
{
}
  myPane.XAxis.Title.Text = "เวลา";
myPane.YAxis.Title.Text = "ค่าความชื้นสัมพัทธ์(%)";

PointPairList list = new PointPairList();
```

```
        for (int x = 0; x < arrHumi.Length; x++)
        {
           list.Add(arrTime[x], arrHumi[x]);
        }

        myPane.XAxis.MajorGrid.IsVisible = true;
        myPane.YAxis.MajorGrid.IsVisible = true;
        myPane.XAxis.MajorGrid.Color = Color.LightGray;
        myPane.YAxis.MajorGrid.Color = Color.LightGray;

        LineItem myCurve = myPane.AddCurve("ค่าความชื้นสัมพัทธ์", list, Color.Blue,
SymbolType.Circle);
        myCurve.Symbol.Fill = new Fill(Color.White);
        myPane.Chart.Fill = new Fill(Color.White, Color.LightPink, 45F);
        myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);
        zgc.AxisChange();
      }

      private void SetSize()
      {
        zgl.Location = new Point(10, 10);
        zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
      }

      private void grHumiD_Resize(object sender, EventArgs e)
      {
        SetSize();
      }
    }
}
```

Datasheet SHT1x (SHT10, SHT11, SHT15)

Humidity and Temperature Sensor

- Fully calibrated
- Digital output
- Low power consumption
- Excellent long term stability
- SMD type package – reflow solderable



Product Summary

SHT1x (including SHT10, SHT11 and SHT15) is Sensirion's family of surface mountable relative humidity and temperature sensors. The sensors integrate sensor elements plus signal processing on a tiny foot print and provide a fully calibrated digital output. A unique capacitive sensor element is used for measuring relative humidity while temperature is measured by a band-gap sensor. The applied CMOSens® technology guarantees excellent reliability and long term stability. Both sensors are seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC).

Each SHT1x is individually calibrated in a precision humidity chamber. The calibration coefficients are programmed into an OTP memory on the chip. These coefficients are used to internally calibrate the signals from the sensors. The 2-wire serial interface and internal voltage regulation allows for easy and fast system integration. The tiny size and low power consumption makes SHT1x the ultimate choice for even the most demanding applications.

SHT1x is supplied in a surface-mountable LCC (Leadless Chip Carrier) which is approved for standard reflow soldering processes. The same sensor is also available with pins (SHT7x) or on flex print (SHTA1).

Dimensions

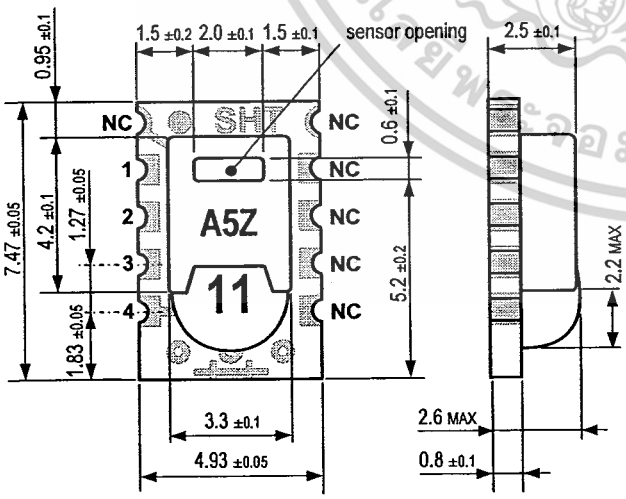


Figure 1: Drawing of SHT1x sensor packaging, dimensions in mm (1mm = 0.039inch). Sensor label gives "11" for SHT11 as an example. Contacts are assigned as follows: 1:GND, 2:DATA, 3:SCK, 4:VDD.

Sensor Chip

SHT1x V4 – for which this datasheet applies – features a version 4 Silicon sensor chip. Besides a humidity and a temperature sensor the chip contains an amplifier, A/D converter, OTP memory and a digital interface. V4 sensors can be identified by the alpha-numeric traceability code on the sensor cap – see example "A5Z" code on Figure 1.

Material Contents

While the sensor is made of a CMOS chip the sensor housing consists of an LCP cap with epoxy glob top on an FR4 substrate. The device is fully RoHS and WEEE compliant, thus it is free of Pb, Cd, Hg, Cr(6+), PBB and PBDE.

Evaluation Kits

For sensor trial measurements, for qualification of the sensor or even experimental application of the sensor there is an evaluation kit *EK-H2* available including sensor, hard and software to interface with a computer.

For more sophisticated and demanding measurements a multi port evaluation kit *EK-H3* is available which allows for parallel application of up to 20 sensors.

ensor Performance

elative Humidity

arameter	Condition	min	typ	max	Units
esolution [1]		0.4	0.05	0.05	%RH
		8	12	12	bit
curacy [2] HT10	typical		±4.5		%RH
	maximal	see Figure 2			
curacy [2] HT11	typical		±3.0		%RH
	maximal	see Figure 2			
curacy [2] HT15	typical		±2.0		%RH
	maximal	see Figure 2			
epeatability			±0.1		%RH
eplacement		fully interchangeable			
ysteresis			±1		%RH
onlinearity	raw data		±3		%RH
	linearized		<<1		%RH
esponse time [3]	τ (63%)		8		s
perating Range		0		100	%RH
ong term drift [4]	normal		< 0.5		%RH/yr

Temperature

Parameter	Condition	min	typ	max	Units
Resolution [1]		0.04	0.01	0.01	°C
		12	14	14	bit
Accuracy [2] SHT10	typical		±0.5		°C
	maximal	see Figure 3			
Accuracy [2] SHT11	typical		±0.4		°C
	maximal	see Figure 3			
Accuracy [2] SHT15	typical		±0.3		°C
	maximal	see Figure 3			
Repeatability			±0.1		°C
Replacement		fully interchangeable			
Operating Range		-40		123.8	°C
		-40		254.9	°F
Response Time [6]	τ (63%)	5		30	s
Long term drift			< 0.04		°C/yr

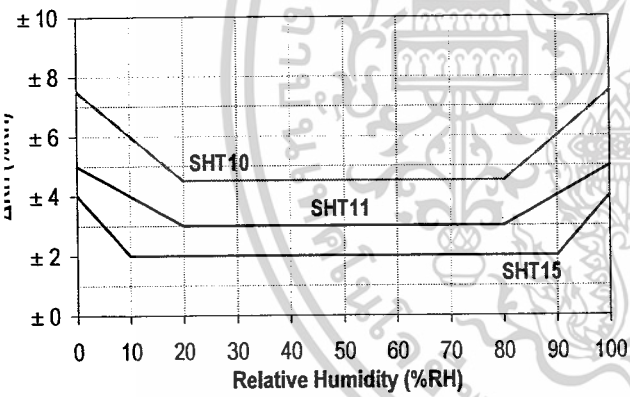


Figure 2: Maximal RH-accuracy at 25°C per sensor type.

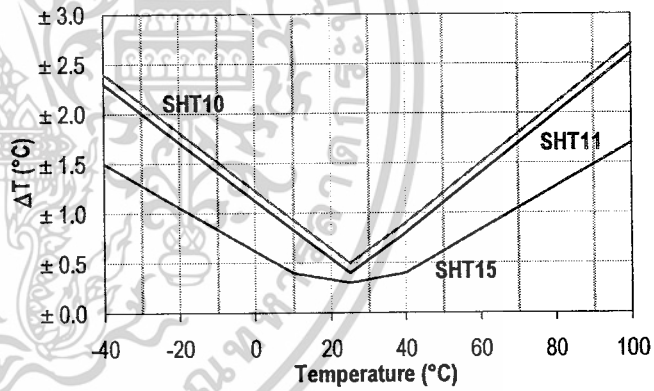


Figure 3: Maximal T-accuracy per sensor type.

Electrical and General Items

Parameter	Condition	min	typ	max	Units
Source Voltage		2.4	3.3	5.5	V
Power Consumption [5]	sleep		2	5	µW
	measuring		3		mW
	average		150		µW
Communication	digital 2-wire interface, see Communication				
Storage	10 – 50°C (0 – 125°C peak), 20 – 60%RH				

Packaging Information

Sensor Type	Packaging	Quantity	Order Number
SHT10	Tape & Reel	2000	1-100218-04
SHT11	Tape & Reel	100	1-100051-04
	Tape & Reel	400	1-100098-04
	Tape & Reel	2000	1-100524-04
SHT15	Tape & Reel	100	1-100085-04
	Tape & Reel	400	1-100093-04

[1] The default measurement resolution of is 14bit for temperature and 12bit for humidity. It can be reduced to 12/8bit by command to status register.

[2] Accuracies are tested at Outgoing Quality Control at 25°C (77°F) and 3.3V. Values exclude hysteresis and non-linearity.

[3] Time for reaching 63% of a step function, valid at 25°C and 1 m/s airflow.

[4] Value may be higher in environments with high contents of volatile organic compounds. See Section 1.3 of Users Guide.

[5] Values for VDD=5.5V at 25°C, average value at one 12bit measurement per second.

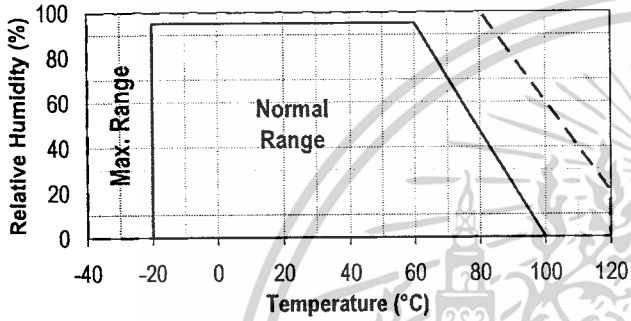
[6] Response time depends on heat capacity of and thermal resistance to sensor substrate.

sers Guide SHT1x

Application Information

1 Operating Conditions

ensor works stable within recommended normal range – ee Figure 4. Long term exposures to conditions outside ormal range, especially at humidity >80%RH, may mporarily offset the RH signal (+3 %RH after 60h). After turn to normal range it will slowly return towards alibration state by itself. See Section 1.4. "Reconditioning rocedure" to accelerate eliminating the offset. Prolonged xposure to extreme conditions may accelerate ageing.



igure 4: Operating Conditions

.2 Soldering instructions

or soldering SHT1x standard reflow soldering ovens may e used. The sensor is qualified to withstand soldering rofile according to IPC/JEDEC J-STD-020C with peak emperatures at 260°C during up to 40sec including Pb-ree assembly in IR/Convection reflow ovens.

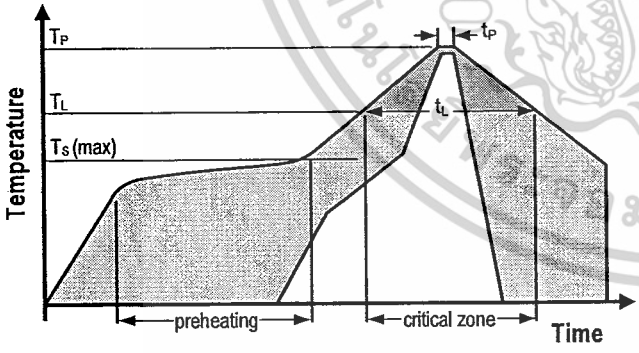


Figure 5: Soldering profile according to JEDEC standard. T_P <= 260°C and t_P < 40sec for Pb-free assembly. T_L < 220°C and t_L < 150sec. Ramp-up/down speeds shall be < 5°C/sec.

For soldering in Vapor Phase Reflow (VPR) ovens the peak conditions are limited to T_P < 233°C during t_P < 30sec and ramp-up/down speeds shall be limited to 10°C/sec. For manual soldering contact time must be limited to 5 seconds at up to 350°C[7].

IMPORTANT: After soldering the devices should be stored at >75%RH for at least 12h to allow the polymer to re-hydrate. Otherwise the sensor may read an offset that slowly disappears if exposed to ambient conditions.

In no case, neither after manual nor reflow soldering, a board wash shall be applied. Therefore it is strongly recommended to use "no-clean" solder paste. In case of application with exposure of the sensor to corrosive gases the soldering pads shall be sealed to prevent loose contacts or short cuts.

For the design of the SHT1x footprint it is recommended to use dimensions according to Figure 7. Sensor pads are coated with 35µm Cu, 5µm Ni and 0.1µm Au.

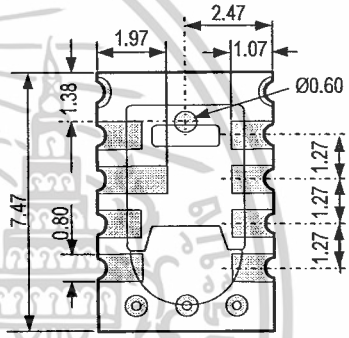


Figure 6: Rear side electrodes of sensor, view from top side.

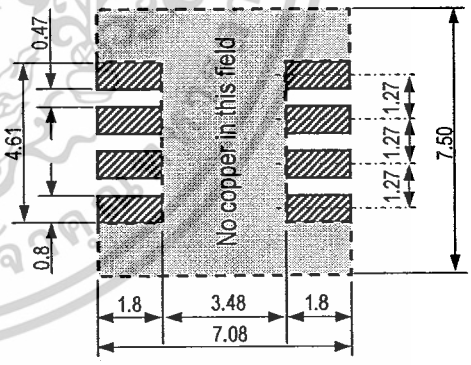


Figure 7: Recommended footprint for SHT1x. Values in mm.

1.3 Storage Conditions and Handling Instructions

It is of great importance to understand that a humidity sensor is not a normal electronic component and needs to be handled with care. Chemical vapors at high concentration in combination with long exposure times may offset the sensor reading.

For these reasons it is recommended to store the sensors in original packaging including the sealed ESD bag at following conditions: Temperature shall be in the range of 10°C – 50°C (0 – 125°C for limited time) and humidity at 20 – 60%RH (sensors that are not stored in ESD bags).

[7] 233°C = 451°F, 260°C = 500°F, 350°C = 662°F

r sensors that have been removed from the original ckaging we recommend to stored them in ESD bags ade of PE-HD[8].

manufacturing and transport the sensors shall be evented of high concentration of chemical solvents and ng exposure times. Out-gassing of glues, adhesive tapes id stickers or out-gassing packaging material such as ubble foils, foams, etc. shall be avoided. Manufacturing ea shall be well ventilated.

or more detailed information please consult the ocument *"Handling Instructions"* or contact Sensirion.

4 Reconditioning Procedure

s stated above extreme conditions or exposure to solvent apors may offset the sensor. The following reconditioning rocedure may bring the sensor back to calibration state:

aking: 100 – 105°C at < 5%RH for 10h
e-Hydration: 20 – 30°C at ~ 75%RH for 12h [9].

.5 Temperature Effects

elative humidity reading strongly depends on mperature. Therefore, it is essential to keep humidity ensors at the same temperature as the air of which the elative humidity is to be measured. In case of testing or ualification the reference sensor and test sensor must how equal temperature to allow for comparing humidity eadings.

the SHT1x shares a PCB with electronic components nat produce heat it should be mounted in a way that revents heat transfer or keeps it as low as possible. Measures to reduce heat transfer can be ventilation, eduction of copper layers between the SHT1x and the est of the PCB or milling a slit into the PCB around the ensor (see Figure 8).

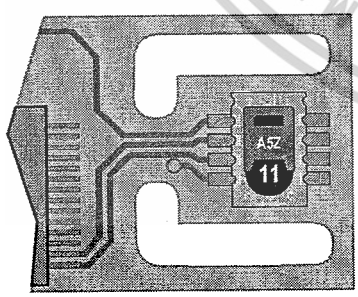


Figure 8: Top view of example of mounted SHT1x with slits milled into PCB to minimize heat transfer.

Furthermore, there are self-heating effects in case the measurement frequency is too high. Please refer to Section 3.3 for detailed information.

[8] For example, 3M antistatic bag, product "1910" with zipper.
[9] 75%RH can conveniently be generated with saturated NaCl solution. 100 – 105°C correspond to 212 – 221°F, 20 – 30°C correspond to 68 – 86°F

1.6 Light

The SHT1x is not light sensitive. Prolonged direct exposure to sunshine or strong UV radiation may age the housing.

1.7 Membranes

SHT1x does not contain a membrane at the sensor opening. However, a membrane may be added to prevent dirt and droplets from entering the housing and to protect the sensor. It will also reduce peak concentrations of chemical vapors. For optimal response times the air volume behind the membrane must be kept minimal. Sensirion recommends and supplies the SF1 filter cap for optimal IP54 protection (for higher protection – i.e. IP67 - SF1 must be sealed to the PCB with epoxy). Please compare Figure 9.

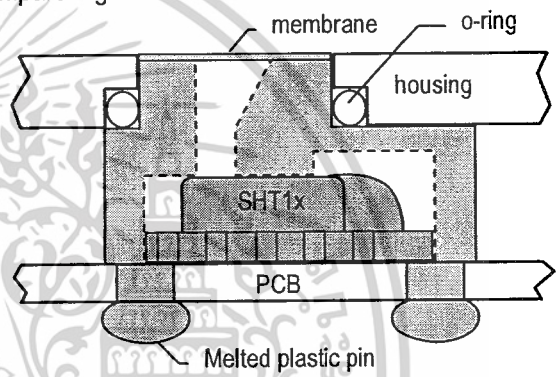


Figure 9: Side view of SF1 filter cap mounted between PCB and housing wall. Volume below membrane is kept minimal.

1.8 Materials Used for Sealing / Mounting

Many materials absorb humidity and will act as a buffer increasing response times and hysteresis. Materials in the vicinity of the sensor must therefore be carefully chosen. Recommended materials are: Any metals, LCP, POM (Delrin), PTFE (Teflon), PE, PEEK, PP, PB, PPS, PSU, PVDF, PVF.

For sealing and gluing (use sparingly): High filled epoxy for electronic packaging (e.g. glob top, underfill), and Silicone. Out-gassing of these materials may also contaminate the SHT1x (see Section 1.3). Therefore try to add the sensor as a last manufacturing step to the assembly, store the assembly well ventilated after manufacturing or bake at >50°C for 24h to outgas contaminants before packing.

1.9 Wiring Considerations and Signal Integrity

Carrying the SCK and DATA signal parallel and in close proximity (e.g. in wires) for more than 10cm may result in cross talk and loss of communication. This may be resolved by routing VDD and/or GND between the two data signals and/or using shielded cables. Furthermore, slowing down SCK frequency will possibly improve signal integrity. Power supply pins (VDD, GND) must be decoupled with a 100nF capacitor if wires are used.

apacitor should be placed as close to the sensor as ossible. Please see the Application Note "ESD, Latchup nd EMC" for more information.

.10 ESD (Electrostatic Discharge)

SD immunity is qualified according to MIL STD 883E, ıethod 3015 (Human Body Model at ±2 kV).

atch-up immunity is provided at a force current of 100mA with T_{amb} = 80°C according to JEDEC78A. See pplication Note "ESD, Latchup and EMC" for more ıformation.

Interface Specifications

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial Data, bidirectional
3	SCK	Serial Clock, input only
4	VDD	Source Voltage
NC	NC	Must be left unconnected

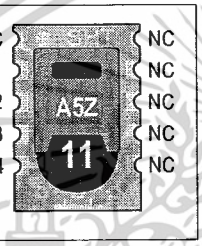


Table 1: SHT1x pin assignment, NC remain floating.

2.1 Power Pins (VDD, GND)

The supply voltage of SHT1x must be in the range of 2.4 – 5.5V, recommended supply voltage is 3.3V. Power supply pins Supply Voltage (VDD) and Ground (GND) must be decoupled with a 100 nF capacitor – see Figure 10.

The serial interface of the SHT1x is optimized for sensor readout and effective power consumption. The sensor cannot be addressed by I²C protocol, however, the sensor can be connected to an I²C bus without interference with other devices connected to the bus. The controller must switch between the protocols.

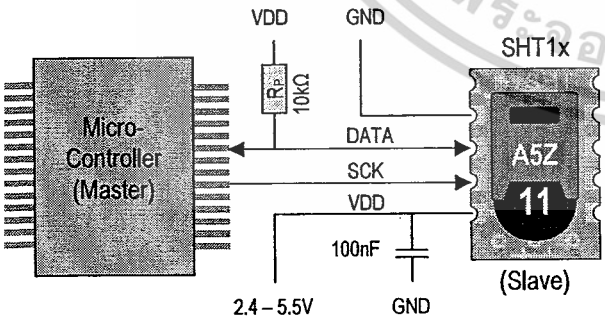


Figure 10: Typical application circuit, including pull up resistor R_P and decoupling of VDD and GND by a capacitor.

2.2 Serial clock input (SCK)

SCK is used to synchronize the communication between microcontroller and SHT1x. Since the interface consists of fully static logic there is no minimum SCK frequency.

2.3 Serial data (DATA)

The DATA tri-state pin is used to transfer data in and out of the sensor. For sending a command to the sensor, DATA is valid on the rising edge of the serial clock (SCK) and must remain stable while SCK is high. After the falling edge of SCK DATA may be changed. For safe communication DATA valid shall be extended T_{SU} and T_{HO} before the rising and after the falling edge of SCK, respectively – see Figure 11. For reading data from the sensor, DATA is valid T_V after SCK has gone low and remains valid until the next falling edge of SCK.

To avoid signal contention the microcontroller must only drive DATA low. An external pull-up resistor (e.g. 10kΩ) is required to pull the signal high – it should be noted that pull-up resistors may be included in I/O circuits of microcontrollers. See Table 2 for detailed I/O characteristic of the sensor.

2.4 Electrical Characteristics

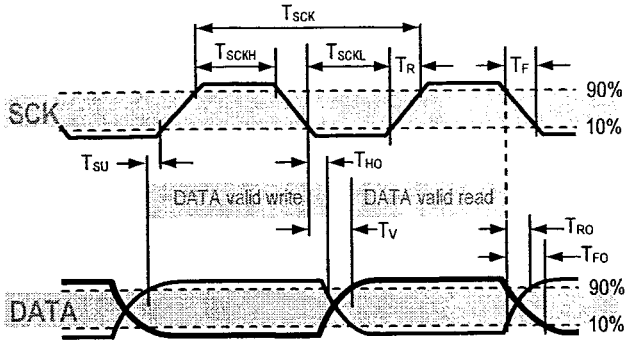
The electrical characteristics such as power consumption, low and high level, input and output voltages depend on the supply voltage. Table 2 gives electrical characteristics of SHT1x with the assumption of 5V supply voltage if not stated otherwise. For proper communication with the sensor it is essential to make sure that signal design is strictly within the limits given in Table 3 and Figure 11.

Parameter	Conditions	min	typ	max	Units
Power supply DC[10]		2.4	3.3	5.5	V
Supply current	measuring		0.55	1	mA
	average[11]	2	28		µA
	sleep		0.3	1.5	µA
Low level output voltage	I_{OL} < 4 mA	0		250	mV
High level output voltage	R_P < 25 kΩ	90%		100%	VDD
Low level input voltage	Negative going	0%		20%	VDD
High level input voltage	Positive going	80%		100%	VDD
Input current on pads				1	µA
Output current	on			4	mA
	Tri-stated (off)		10	20	µA

Table 2: SHT1x DC characteristics. R_P stands for pull up resistor, while I_{OL} is low level output current.

[10] Recommended voltage supply for highest accuracy is 3.3V, due to sensor calibration.

[11] Minimum value with one measurement of 8 bit accuracy without OTP reload per second, typical value with one measurement of 12bit accuracy per second.



igure 11: Timing Diagram, abbreviations are explained in able 3. Bold DATA line is controlled by the sensor, plain DATA ne is controlled by the micro-controller. Both valid times refer to e left SCK toggle.

	Parameter	Conditions	min	typ	max	Units
F_{SCK}	SCK Frequency	VDD > 4.5V	0	0.1	5	MHz
		VDD < 4.5V	0	0.1	1	MHz
T_{SCKx}	SCK hi/low time		100			ns
T_R/T_F	SCK rise/fall time		1	200	*	ns
T_{FO}	DATA fall time	CL = 5pF	3.5	10	20	ns
		CL = 100pF	30	40	200	ns
T_{RO}	DATA rise time		**	**	**	ns
T_V	DATA valid time		200	250	***	ns
T_{SU}	DATA setup time		100	150	***	ns
T_{HO}	DATA hold time		10	15	****	ns

* $T_{R\_max} + T_{F\_max} = (F_{SCK})^{-1} - T_{SCKH} - T_{SCKL}$

** T_{R0} is determined by the R_P*C_{bus} time-constant at DATA line

*** T_{V_max} and T_{SU_max} depend on external pull-up resistor (R_P) and total bus line capacitance (Cbus) at DATA line

**** $T_{H0\_max} < T_V - \max(T_{R0}, T_{F0})$

able 3: SHT1x I/O signal characteristics, CL stands for Output oad, entities are displayed in Figure 11.

3 Communication with Sensor

3.1 Start up Sensor

As a first step the sensor is powered up to chosen supply voltage VDD. The slew rate during power up shall not fall below 1V/ms. After power-up the sensor needs 11ms to get to Sleep State. No commands must be sent before that time.

3.2 Sending a Command

To initiate a transmission, a Transmission Start sequence has to be issued. It consists of a lowering of the DATA line while SCK is high, followed by a low pulse on SCK and raising DATA again while SCK is still high – see Figure 12.

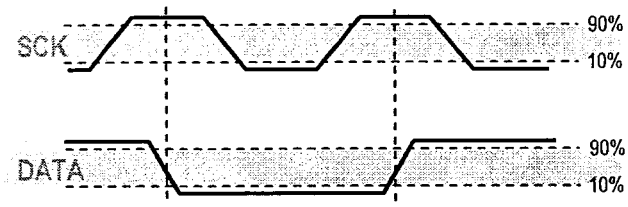


Figure 12: "Transmission Start" sequence

The subsequent command consists of three address bits (only '000' is supported) and five command bits. The SHT1x indicates the proper reception of a command by pulling the DATA pin low (ACK bit) after the falling edge of the 8th SCK clock. The DATA line is released (and goes high) after the falling edge of the 9th SCK clock.

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

Table 4: SHT1x list of commands

3.3 Measurement of RH and T

After issuing a measurement command ('00000101' for relative humidity, '00000011' for temperature) the controller has to wait for the measurement to complete. This takes a maximum of 20/80/320 ms for a 8/12/14bit measurement. The time varies with the speed of the internal oscillator and can be lower by up to 30%. To signal the completion of a measurement, the SHT1x pulls data line low and enters Idle Mode. The controller must wait for this Data Ready signal before restarting SCK to readout the data. Measurement data is stored until readout, therefore the controller can continue with other tasks and readout at its convenience.

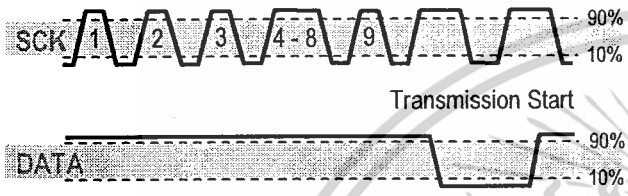
Two bytes of measurement data and one byte of CRC checksum (optional) will then be transmitted. The micro controller must acknowledge each byte by pulling the DATA line low. All values are MSB first, right justified (e.g. the 5th SCK is MSB for a 12bit value, for a 8bit result the first byte is not used).

Communication terminates after the acknowledge bit of the CRC data. If CRC-8 checksum is not used the controller may terminate the communication after the measurement data LSB by keeping ACK high. The device automatically returns to Sleep Mode after measurement and communication are completed.

nportant: To keep self heating below 0.1°C, SHT1x hould not be active for more than 10% of the time – e.g. laximum one measurement per second at 12bit accuracy hall be made.

.4 Connection reset sequence

communication with the device is lost the following signal equence will reset the serial interface: While leaving DATA high, toggle SCK nine or more times – see Figure 3. This must be followed by a Transmission Start equence preceding the next command. This sequence esets the interface only. The status register preserves its ontent.



igure 13: Connection Reset Sequence

.5 CRC-8 Checksum calculation

he whole digital transmission is secured by an 8bit hecksum. It ensures that any wrong data can be detected nd eliminated. As described above this is an additional eature of which may be used or abandoned.

lease consult Application Note "CRC-8 Checksum alculation" for information on how to calculate the CRC.

Status Register

Some of the advanced functions of the SHT1x such as selecting measurement resolution, end of battery notice or using the heater may be activated by sending a command to the status register. The following section gives a brief overview of these features. A more detailed description is available in the Application Note "Status Register".

After the command Status Register Read or Status Register Write – see Table 4 – the content of 8 bits of the status register may be read out or written. For the communication compare Figures 16 and 17 – the assignation of the bits is displayed in Table 5.

Figure 14: Status Register Write

Figure 15: Status Register Read

Examples of full communication cycle are displayed in Figures 15 and 16.

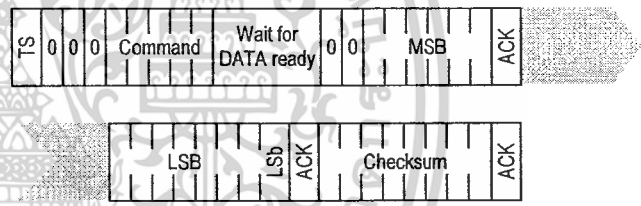


Figure 16: Overview of Measurement Sequence. TS = Transmission Start, MSB = Most Significant Byte, LSB = Last Significant Byte, LSb = Last Significant Bit.

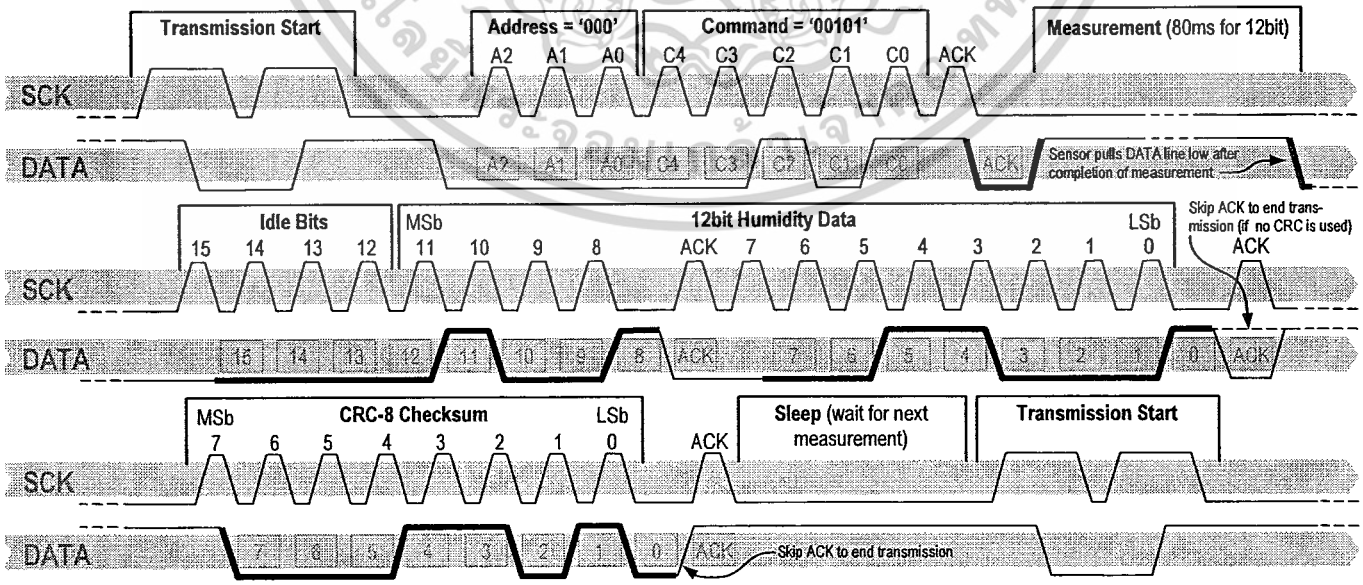


Figure 17: Example RH measurement sequence for value "0000'1001"0011'0001" = 2353 = 75.79 %RH (without temperature compensation). DATA valid times are given and referenced in boxes on DATA line. Bold DATA lines are controlled by sensor while plain lines are controlled by the micro-controller.

t	Type	Description	Default	
7		reserved	0	
6	R	End of Battery (low voltage detection) '0' for VDD > 2.47 '1' for VDD < 2.47	X	No default value, bit is only updated after a measurement
5		reserved	0	
4		reserved	0	
3		For Testing only, do not use	0	
2	R/W	Heater	0	off
1	R/W	no reload from OTP	0	reload
0	R/W	'1' = 8bit RH / 12bit Temp. resolution '0' = 12bit RH / 14bit Temp. resolution	0	12bit RH 14bit Temp.

able 5: Status Register Bits

Measurement resolution: The default measurement esolution of 14bit (temperature) and 12bit (humidity) can e reduced to 12 and 8bit. This is especially useful in high peed or extreme low power applications.

End of Battery function detects and notifies VDD voltages elow 2.47 V. Accuracy is ±0.05 V.

Heater: An on chip heating element can be addressed by writing a command into status register. The heater may ncrease the temperature of the sensor by 5 – 10°C[12] beyond ambient temperature. The heater draws roughly 8mA @ 5V supply voltage.

For example the heater can be helpful for functionality analysis: Humidity and temperature readings before and after applying the heater are compared. Temperature shall ncrease while relative humidity decreases at the same ime. Dew point shall remain the same.

Please note: The temperature reading will display the emperature of the heated sensor element and not ambient temperature. Furthermore, the sensor is not qualified for continuous application of the heater.

4 Conversion of Signal Output

4.1 Relative Humidity

For compensating non-linearity of the humidity sensor – see Figure 18 – and for obtaining the full accuracy of the sensor it is recommended to convert the humidity readout (SO_{RH}) with the following formula with coefficients given in Table 6:

$$RH_{linear} = c_1 + c_2 \cdot SO_{RH} + c_3 \cdot SO_{RH}^{\ 2} \ (\%RH)$$

[12] Corresponds to 9 – 18°F

SO_{RH}	c_1	c_2	c_3
12 bit	-2.0468	0.0367	-1.5955E-6
8 bit	-2.0468	0.5872	-4.0845E-4

Table 6: Optimized V4 humidity conversion coefficients

The values given in Table 6 are newly introduced and provide optimized accuracy for V4 sensors along the full measurement range. The parameter set c_x*, which has been proposed in earlier datasheets, which was optimized for V3 sensors, still applies to V4 sensors and is given in Table 7 for reference.

SO_{RH}	c_1*	c_2*	c_3*
12 bit	-4.0000	0.0405	-2.8000E-6
8 bit	-4.0000	0.6480	-7.2000E-4

Table 7: V3 humidity conversion coefficients, which also apply to V4.

For simplified, less computation intense conversion formulas see Application Note "RH and Temperature Non-Linearity Compensation". Values higher than 99% RH indicate fully saturated air and must be processed and displayed as 100%RH[13]. Please note that the humidity sensor has no significant voltage dependency.

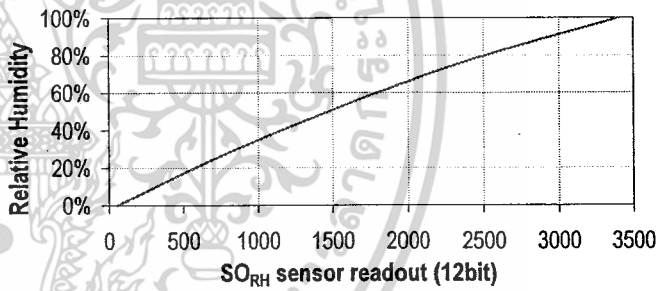


Figure 18: Conversion from SO_{RH} to relative humidity

4.2 Temperature compensation of Humidity Signal

For temperatures significantly different from 25°C (~77°F) the humidity signal requires a temperature compensation. The temperature correction corresponds roughly to 0.12%RH/°C @ 50%RH. Coefficients for the temperature compensation are given in Table 8.

$$RH_{true} = (T_{°C} - 25) \cdot (t_1 + t_2 \cdot SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Table 8: Temperature compensation coefficients[14]

[13] If wetted excessively (strong condensation of water on sensor surface), sensor output signal can drop below 100%RH (even below 0%RH in some cases), but the sensor will recover completely when water droplets evaporate. The sensor is not damaged by water immersion or condensation.

[14] Coefficients apply both to V3 as well as to V4 sensors.

3 Temperature

ne band-gap PTAT (Proportional To Absolute emperature) temperature sensor is very linear by design. se the following formula to convert digital readout (SO_T) temperature value, with coefficients given in Table 9:

$$T = d_1 + d_2 \cdot SO_T$$

VDD	d_1 (°C)	d_1 (°F)
5V	-40.1	-40.2
4V	-39.8	-39.6
3.5V	-39.7	-39.5
3V	-39.6	-39.3
2.5V	-39.4	-38.9

SO_T	d_2 (°C)	d_2 (°F)
14bit	0.01	0.018
12bit	0.04	0.072

able 9: Temperature conversion coefficients[15].

.4 Dew Point

SHT1x is not measuring dew point directly, however dew oint can be derived from humidity and temperature eadings. Since humidity and temperature are both neasured on the same monolithic chip, the SHT1x allows uperb dew point measurements.

or dew point (T_d) calculations there are various formulas o be applied, most of them quite complicated. For the emperature range of -40 – 50°C the following pproximation provides good accuracy with parameters iven in Table 10:

$$T_d(RH,T) = T_n \cdot \frac{\ln\left(\frac{RH}{100\%}\right) + \frac{m \cdot T}{T_n + T}}{m - \ln\left(\frac{RH}{100\%}\right) - \frac{m \cdot T}{T_n + T}}$$

emperature Range	Tn (°C)	m
Above water, 0 – 50°C	243.12	17.62
Above ice, -40 – 0°C	272.62	22.46

Table 10: Parameters for dew point (T_d) calculation.

Please note that "ln(...)" denotes the natural logarithm. For RH and T the linearized and compensated values for elative humidity and temperature shall be applied.

For more information on dew point calculation see Application Note "Dew point calculation".

5 Environmental Stability

If sensors are qualified for assemblies or devices, please make sure that they experience same conditions as the reference sensor. It should be taken into account that response times in assemblies may be longer, hence enough dwell time for the measurement shall be granted. For detailed information please consult Application Note "Qualification Guide".

The SHT1x sensor series were tested according to AEC-Q100 Rev. F qualification test method. Sensor specifications are tested to prevail under the AEC-Q100 temperature grade 2 test conditions listed in Table 11[16]. Sensor performance under other test conditions cannot be guaranteed and is not part of the sensor specifications. Especially, no guarantee can be given for sensor performance in the field or for customer's specific application.

Please contact Sensirion for detailed information.

Environment	Standard	Results[17]
HTSL	125°C, 1000 hours	Within specifications
TC	-50°C - 125°C, 1000 cycles Acc. JESD22-A104-C	Within specifications
UHST	130°C / 85%RH, 96h	Within specifications
THU	85°C / 85%RH, 1000h	Within specifications
ESD immunity	MIL STD 883E, method 3015 (Human Body Model at ±2kV)	Qualified
Latch-up	force current of ±100mA with T_{amb} = 80°C, acc. JEDEC 17	Qualified

Table 11: Qualification tests: HTSL = High Temperature Storage Lifetime, TC = Temperature Cycles, UHST = Unbiased Highly accelerated temperature and humidity Test, THU = Temperature humidity unbiased

6 Packaging

6.1 Packaging type

SHT1x are supplied in a surface mountable LCC (Leadless Chip Carrier) type package. The sensor housing consists of a Liquid Crystal Polymer (LCP) cap with epoxy glob top on a standard 0.8mm FR4 substrate. The device is fully RoHS and WEEE compliant – it is free of of Pb, Cd, Hg, Cr(6+), PBB and PBDE.

[15] Temperature coefficients have slightly been adjusted compared to datasheet SHTxx version 3.01. Coefficients apply to V3 as well as V4 sensors.

[16] Sensor operation temperature range is -40 to 105°C according to AEC-Q100 temperature grade 2.

[17] According to accuracy and long term drift specification given on Page 2.

evice size is 7.47 x 4.93 x 2.5 mm (0.29 x 0.19 x 0.1 ch), see Figure 1, weight is 100 mg.

.2 Traceability Information

ll SHT1x are marked with an alphanumeric, three digit ode on the chip cap (for reference: V3 sensors were abeled with numeric codes) – see "A5Z" on Figure 1. The t numbers allow full traceability through production, alibration and testing. No information can be derived from ie code directly, respective data is stored at Sensirion nd is provided upon request.

abels on the reels are displayed in Figures 19 and 20, ney both give traceability information.

Lot No.: XX0-04-YRRRRTTTT
Quantity: RRRR
RoHS: Compliant

Lot No.

igure 19: First label on reel: XX = Sensor Type (11 for SHT11), 4 = Chip Version (V4), Y = last digit of year, RRRR = number of ensors on reel, TTTT = Traceability Code.

SENSIRION
THE SENSOR COMPANY

Device Type:	1-100PPP-04
Description:	Humidity & Temperature Sensor SHTxx
Part Order No.	1-100PPP-04 or Customer Number
Date of Delivery:	DD.MM.YYYY
Order Code:	45CCCC / 0

Figure 20: Second label on reel: For Device Type and Part Order Number please refer to Table 12, Delivery Date (also Date Code) is date of packaging of sensors (DD = day, MM = month, YYYY = year), CCCC = Sensirion order number.

6.3 Shipping Package

SHT1x are shipped in 12mm tape at 100pcs, 400pcs and 2000pcs – for details see Figure 21 and Table 12. Reels are individually labeled with barcode and human readable labels.

Sensor Type	Packaging	Quantity	Order Number
SHT10	Tape & Reel	2000	1-100218-04
SHT11	Tape & Reel	100	1-100051-04
	Tape & Reel	400	1-100098-04
	Tape & Reel	2000	1-100524-04
SHT15	Tape & Reel	100	1-100085-04
	Tape & Reel	400	1-100093-04

Table 12: Packaging types per sensor type.

Dimensions of packaging tape is given in Figure 21. All tapes have a minimum of 480mm empty leader tape (first pockets of the tape) and a minimum of 300mm empty trailer tape (last pockets of the tape).

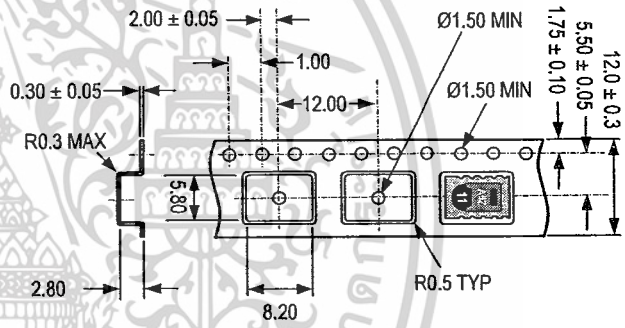


Figure 21: Tape configuration and unit orientation within tape, dimensions in mm (1mm = 0.039inch). The leader tape is at the right side of the figure while the trailer tape is to the left (direction of unreeling).

evision History

ate	Version	Page(s)	Changes
larch 2007	3.0	1 – 10	Data sheet valid for SHTxx-V4 and SHTxx-V3
ugust 2007	3.01	1 – 10	Electrical characteristics added, measurement time corrected
uly 2008	4.0	1 – 11	New release, rework of datasheet
eptember 2008	4.1	3, 4	Adjustment of normal operating range and recommendation for antistatic bag

mportant Notices

arning, Personal Injury

o not use this product as safety or emergency stop devices or in ny other application where failure of the product could result in ersonal injury. Do not use this product for applications other an its intended and authorized use. Before installing, handling, sing or servicing this product, please consult the data sheet and pplication notes. Failure to comply with these instructions could esult in death or serious injury.

the Buyer shall purchase or use SENSIRION products for any nintended or unauthorized application, Buyer shall defend, indemnify nd hold harmless SENSIRION and its officers, employees, ubsidiaries, affiliates and distributors against all claims, costs, amages and expenses, and reasonable attorney fees arising out of, irectly or indirectly, any claim of personal injury or death associated ith such unintended or unauthorized use, even if SENSIRION shall be llegedly negligent with respect to the design or the manufacture of the roduct.

SD Precautions

he inherent design of this component causes it to be sensitive to lectrostatic discharge (ESD). To prevent ESD-induced damage and/or legradation, take customary and statutory ESD precautions when andling this product.
See application note "ESD, Latchup and EMC" for more information.

Varranty

SENSIRION warrants solely to the original purchaser of this product for period of 12 months (one year) from the date of delivery that this roduct shall be of the quality, material and workmanship defined in SENSIRION's published specifications of the product. Within such eriod, if proven to be defective, SENSIRION shall repair and/or eplace this product, in SENSIRION's discretion, free of charge to the Buyer, provided that:

- notice in writing describing the defects shall be given to SENSIRION within fourteen (14) days after their appearance;
- such defects shall be found, to SENSIRION's reasonable satisfaction, to have arisen from SENSIRION's faulty design, material, or workmanship;
- the defective product shall be returned to SENSIRION's factory at the Buyer's expense; and
- the warranty period for any repaired or replaced product shall be limited to the unexpired portion of the original period.

This warranty does not apply to any equipment which has not been installed and used within the specifications recommended by SENSIRION for the intended and proper use of the equipment. EXCEPT FOR THE WARRANTIES EXPRESSLY SET FORTH HEREIN, SENSIRION MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THE PRODUCT. ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE EXPRESSLY EXCLUDED AND DECLINED.
SENSIRION is only liable for defects of this product arising under the conditions of operation provided for in the data sheet and proper use of the goods. SENSIRION explicitly disclaims all warranties, express or implied, for any period during which the goods are operated or stored not in accordance with the technical specifications.
SENSIRION does not assume any liability arising out of any application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. All operating parameters, including without limitation recommended parameters, must be validated for each customer's applications by customer's technical experts. Recommended parameters can and do vary in different applications.
SENSIRION reserves the right, without further notice, (i) to change the product specifications and/or the information in this document and (ii) to improve reliability, functions and design of this product.

Headquarter and Sales Offices

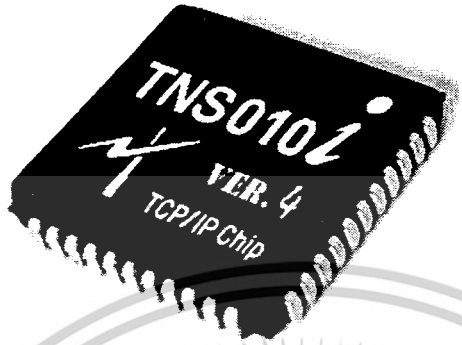
Headquarter
SENSIRION AG
Laubisruetistr. 50
CH-8712 Staefa ZH
Switzerland
Phone: +41 (0)44 306 40 00
Fax: +41 (0)44 306 40 30
info@sensirion.com
http://www.sensirion.com/

Sales Office USA:
SENSIRION Inc.
2801 Townsgate Rd., Suite 240
Westlake Village, CA 91361
USA
Phone: 805 409 4900
Fax: 805 435 0467
michael.karst@sensirion.com
http://www.sensirion.com/

Sales Office Korea:
SENSIRION KOREA Co. Ltd.
#1414, Anyang Construction Tower B/D,
1112-1, Bisan-dong, Anyang-city
Gyeonggi-Province
South Korea
Phone: 031 440 9925~27
Fax: 031 440 9927
info@sensirion.co.kr
http://www.sensirion.co.kr

Sales Office Japan:
SENSIRION JAPAN Co. Ltd.
Postal Code: 108-0074
Shinagawa Station Bldg. 7F,
4-23-5, Takanawa, Minato-ku
Tokyo, Japan
Phone: 03 3444 4940
Fax: 03 3444 4939
info@sensirion.co.jp
http://www.sensirion.co.jp

Find your local representative at: http://www.sensirion.com/reps



TNS010i TCP/IP Stack Chip

PRELIMINARY

Data Sheet and User Manual

Rev. 010-04-10-2005

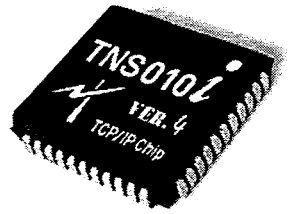
Firmware 4.06

Checkout:

http://www.tcpipchip.com

for details and updates

TNS010i *TCP/IP* stack Chip



Description

TNS010i TCP/IP stack chip is a complete single turnkey product that enables the Internet connection. Not only does it reduce development time to market, the TNS010i allows several real-time efficient transmissions of data and commands between an 8/16/32-bit micro controller (MCU) and the core TCP/IP software engine. With no priori information about the Internet protocol or networking, this device allows the MCU to send and receive web pages, data, and commands in a form of scripting phrases through its built in TCP/IP protocol engine.

Executing only a few simple commands from an MCU through serial pins of TNS010i and you will be able to transmit and receive data to/from web application program. Each command sent across the TNS010i will invoke TNS010i to response back to an MCU with a message. This debugging information helps users keep track of every step during the connection attempt.

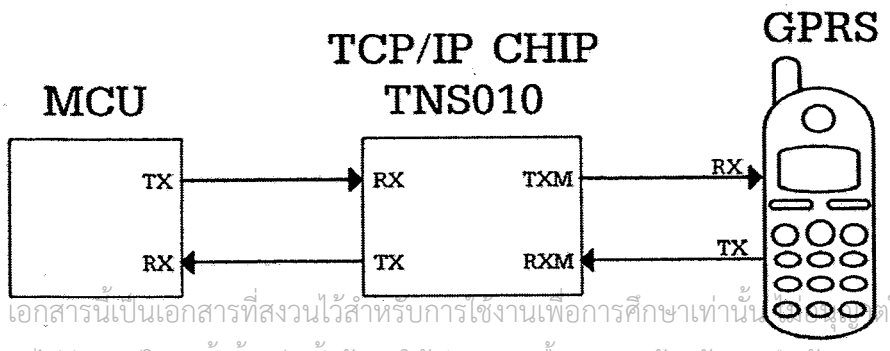
This device can connect to the GPRS phone directly with no additional components.
Even the MCU needs to set-up or read out the GPRS parameters, this device automatically bypasses those AT commands to the phone thus eliminating

- The need for two serial-port hardware on the MCU side. and
- A complex circuit for switching among the three modules: MCU, TNS010i, and GPRS.

Features

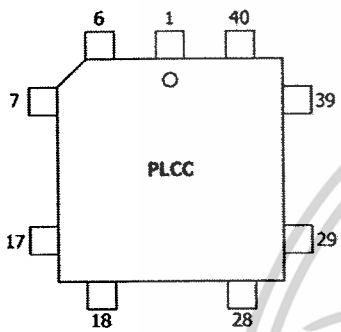
- Support basic protocols including TCP/IP, PPP, DNS ,HTTP,SMTP,POP3 and TCP/IP socket.
- Easy interface to GPRS phone or Modem.
- Input commands length as long as 1,000 bytes.
- No need for external RAM or ROM.
- Minimal components. Requires only one 18.432 MHz Xtal, and 3 Capacitors .
- 19,200 BPS Command communicates with micro controller (MCU)
- Information exchanges rate with GPRS is variable from 1,200 BPS to 57,600 BPS
- low power consumption
- 44-pin PLCC and LQFP package

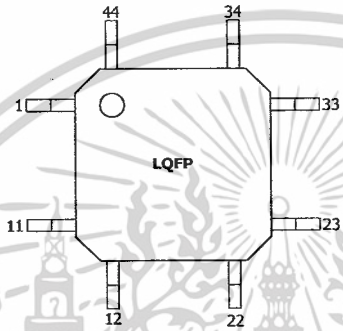
The Connection Diagram



Electrical Characteristics

Symbol	Description	Min	Type	Max	Unit
VIL	Input Low Voltage	-0.5		0.9	V.
VIH	Input High Voltage	1.9		5.5	V.
VIH_	Input High Voltage XTAL1,RST	3.5		5.5	V.
Icc	Power supply current			60	mA
Vcc	Power supply voltage	4.75		5.25	V.





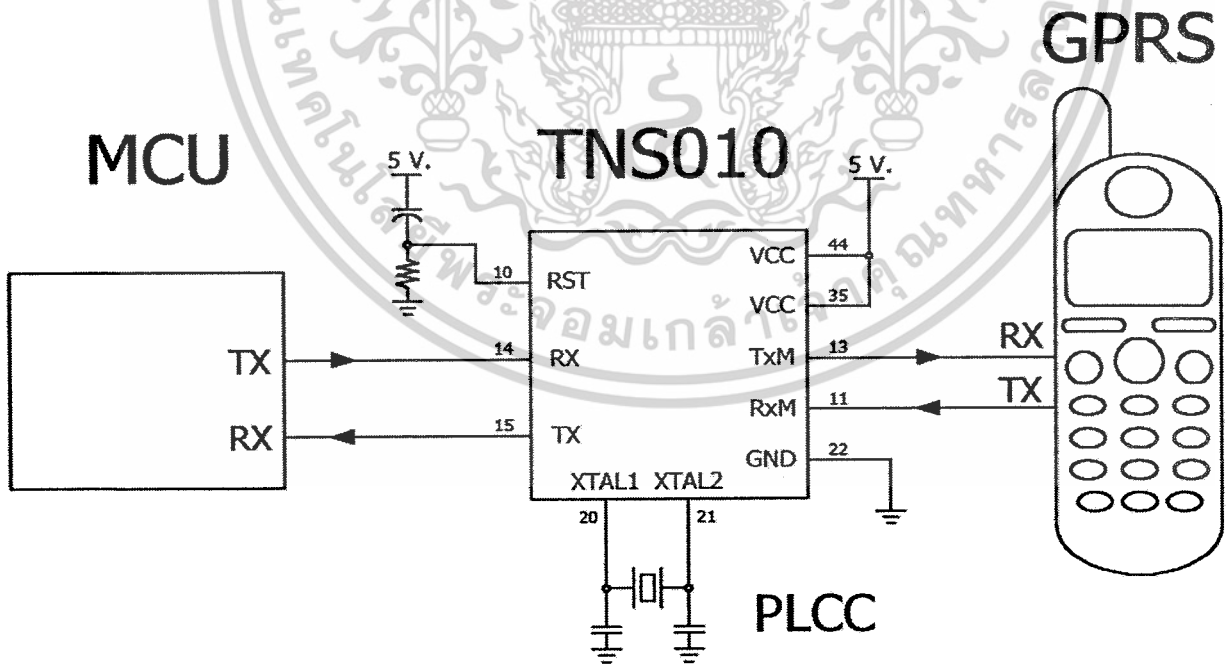
Pin Description

Symbol	PLCC	LQFP	In/Out	Name and Function
Vcc	44,35	38,29	I	Power Supply : power supply voltage for operation
Gnd	22	16	I	Ground : 0 V. reference
XTAL1	21	15	I	Crystal 1: 18.432 MHz. Input oscillator amplifier.
XTAL2	20	14	O	Crystal 2: 18.432 MHz. Output oscillator amplifier.
GPRS/modem	18	12	I	GPRS / Land line modem time out select
CD	2	40	I	Carrier detect
DTR	3	41	O	Data terminal ready
DSR	4	42	I	Data set ready
RTS	5	43	O	Request to send
CTS	6	44	I	Clear to send
RST	10	4	I	Reset : hold high at least 0.22 mS
TxM	13	7	O	Transmit to modem/GPRS phone
RxM	11	5	I	Receive from modem/GPRS phone
Tx	15	9	O	Transmit to controller
Rx	14	8	I	Receive from controller
LED-Connect	9	3	O	LED status shows connected to net
LED-Data trans	19	13	O	LED status shows data transferring

TCP/IP Software Specifications

Maximum at+i command length (include at+i itself)	1000 bytes
Maximum TCP segment size	512 bytes
Serial RX buffer length (Ring buffer)	2,500 bytes
Serial TX buffer length	1,000 bytes
PAP	Support
CHAP	Not support
DNS	Support
HTTP	Support
FTP	Not support
SMTP	Support
POP3	Support
TCP/IP socket	Support

Typical testing circuit



Summary of Commands

- 19,200 BPS command baud rate (MCU (controller) ←→ TCP/IP Chip) N-8-1
- 1,200-57,600 BPS GPRS/modem baud rate. (TCP/IP Chip ←→ GPRS/Modem) N-8-1
- Dark printing characters are main commands and the gray are option commands

Chip test

Command: at+i<CR>
Response: **+I_OK**

Open a PPP connection

Command: at+iopen<CR>
Response:
- Success **+I_OK**

- Not success
+I_ERROR 1 PPP_NOT_ESTABLISH *(can't connect to PPP server)*
+I_ERROR 3 DIAL_NOT_SET *(no dial number set up)*
+I_ERROR 4 MODEM_NOT_RESPONSE *(no modem response)*
+I_ERROR 7 PASSWORD_NOT_GOOD *(invalid password)*

Close a PPP connection

Command: at+iclose<CR>
Response:
- Success **+I_OK**

- Not success
+I_ERROR 4 MODEM_NOT_RESPONSE *(can't disconnect modem)*

Display a current assigned IP address

Command: at+iip<CR>
Response: **CLIENT_IP=x.x.x.x**
SERVER_IP=x.x.x.x
+I_OK

Set up a dialing number

Command:	at+isetd=< ISP dialing number><CR>
Response:	**+I_OK**
Example:	at+isetd=*99***1#<CR>
	+I_OK

Display a dialing number

Command:	at+isetd=?<CR>
Response:	**<DIAL NUMBER>**
	+I_OK
Example:	at+isetd=? <CR>
	991#**
	+I_OK

Set up DNS server's IP address (in case of ISP not automatic DNS assigned)

Command:	at+isetdns=<ip address><CR>
Response:	**+I_OK**
Example:	at+isetdns=203.155.33.1<CR>
	+I_OK

Display DNS server's IP address

Command:	at+isetdns=?<CR>
Response:	**<DNS SERVER IP ADDRESS>**
	+I_OK
Example:	at+isetdns=? <CR>
	203.155.33.1
	+I_OK

Set up PPP user name

Command:	at+isetuser=<username><CR>
Response:	**+I_OK**
Example:	at+isetuser=David<CR>
	+I_OK

Display PPP user name

Command: at+isetuser=?<CR>

Response: **<USER NAME>**
+I_OK

Example: at+isetuser=? <CR>
David
+I_OK

Set up PPP password

Command: at+isetpass=<password><CR>

Response: **+I_OK**

Example: at+isetpass=David password<CR>
+I_OK

Display PPP password

Command: at+isetpass=?<CR>

Response: **<PASSWORD>**
+I_OK

Example: at+isetpass=?<CR>
David password
+I_OK

HTTP Request

Command:
at+ihttp://<url>[<path>][:<port>][<space></h></l></d>]<CR>

Response:
- Success **+I_OK**
<WEB PAGE CONTENT.....
.....
.....
.....>

- Not success
+I_ERROR 2 DNS_IP_NOT_SET
(If specify domain name, must set a DNS IP address first : at+isetdns)
+I_ERROR 5 TCP_TIMEOUT *(server is busy)*
+I_ERROR 0 PPP_CLOSED *(PPP is not established yet, must open with at+iopen)*

Note:
1. /h = display all **header** field.
2. /l = display only **length** field
3. /d = display only **date** field
4. default port number is 80

Example:

```
at+ihttp://www.google.com /l<CR>
+I_OK
Content-Length:2998
<html><head><meta http-equiv="content-type"
content="text/html; charset=windows-
874"><title>Google</title><style><!--
body,td,a,p,.h{font-family:;}
.h{font-size: 20px;}
.q{text-decoration:none; color:#0000cc;}
.....................
.....................
</html>
```

Example:

```
at+ihttp://www.myserver.com:123/doc.htm/d/l<CR>
+I_OK
Date: Mon, 29 Dec 2003 12:29:09 GMT
Content-Length: 3203
<HTML>.....................
.....................
.....................
</HTML>
```

How to request the next HTTP ?

After successfully retrieving the first web page, the successive request needs no further initial set-up commands.
Just send

```
Command:
at+ihttp://<url>[<path>][:<port>][<space></h></l></d>]<CR>
```

In case a connection is broken

1. Close the existing connection first with

```
Command:    at+iclose<CR>
```

2. Then re-open PPP connection with

```
Command:    at+iopen<CR>
```

3. Followed by requesting a web page

```
Command:
at+ihttp://<url>[<path>][:<port>][<space></h></l></d>]<CR>
```

TCP/IP SOCKET

1.OPEN TCP/IP SOCKET

```
Command:    at+isockopen=<url>:<port><CR>
```

Response: **+I_OK**
<RESPONSE/NONE>
Note: <port> is require , no default value

Example: at+sockopen=mail.server.com:25<CR>
+I_OK
220 SMTP Service ready

2.SEND DATA ON TCP/IP SOCKET

```
Command:    at+isock=<data><CR>
```

Response: **+I_OK**
<RESPONSE/NONE>

Example: at+sock= HELO\r\n<CR>
+I_OK
250 HELLO <CR>

3.CLOSE TCP/IP SOCKET

```
Command:    at+isockclose<CR>
```

Response: **+I_OK**

EMAIL SEND/RECEIVE (SMTP & POP3)

1.OPEN -> SEND -> CLOSE Mail

```
Command:
at+ismtpsend=<mailserver>[:<port>],<from>,<to>,<subject>,<data><CR>
```

Response: **+I_OK**
or
+I+ERROR 10 MAIL_ERROR

Note:
- <port> default value is 25.

Example:

at+ismtpsend=mail.server.com,sender@aaa.com,
recv@bbb.com,My subject,Hello how are you?<CR>
+I_OK

2.READ EMAIL(POP3)

2.1 OPEN -> CHECK MAIL

Command:
at+ipop3open=<mailserver>[:<port>],<user_name>,<password><CR>

Response: **< mail number list >**
+I_OK
or
+I+ERROR 10 MAIL_ERROR

Note:
- <port> default value is 110.

Example:
at+ipop3open=pop3.server.com,david,1234<CR>
1,2,3,4,5
+I_OK

2.2 READ EMAIL

Command:
at+ipop3read=<mail_number><CR>

Response: **< mail content >**
+I_OK
or
+I+ERROR 10 MAIL_ERROR

Example:
at+ipop3read=4<CR>
From: sender@aaa.com
Subject: Test

Hello , How are you?
+I_OK

2.3 CHECK MAIL

Command: at+ipop3check<CR>

Response: **< mail number list >**
+I_OK
or
+I+ERROR 10 MAIL_ERROR

Example:
at+ipop3check<CR>
1,2,3,4,5
+I_OK

2.4 DELETE MAIL

Command:	at+ipop3delete=<mail number><CR>
Response:	**< mail number >** **+I_OK** or **+I+ERROR 10 MAIL_ERROR**

Note: mail number

"all" = delete all messages
num1, num2, num3 = delete specific messages
num1-num2 = delete range of messages

Example:

at+ipop3delete=1,2,6<CR>
3,4,5 **//display undelete message**
+I_OK

at+ipop3delete=all<CR>

+I_OK

at+ipop3delete=4-10<CR>
1,2,3 **//display undelete message**
+I_OK

2.5 CLOSE READING EMAIL SERVER

Command:	at+ipop3close<CR>
Response:	**+I_OK** or **+I+ERROR 10 MAIL_ERROR**

Read software version

Command:	at+iver<CR>
Response:	**VERx.xx** **+I_OK**

Example: at+iver <CR>
VER0.07
+I_OK

Bypass to modem's AT commands mode

1st Step :
type **+++** and **<cr>** for entering modem's AT commands mode

Command:	+++<CR>
Response:	**+I_OK**

2nd step :

Case1 : modem is not connecting to internet
Type **at<cr>** to initiate and enter modem's commands mode
OK → response from modem

Case2 : modem is connecting to internet
Type +++ and wait for 2-3 sec. and then type
at<cr> to initiate and enter modem's commands mode
OK → response from modem

From this state you can communicate to modem with it's AT Commands

To quit from modem's AT commands mode and enter TNS010i commands mode

Case1 : modem is not connecting to internet
Type **at+i<cr>** to enter TNS010i's commands mode
+I_OK → response from TNS010i

Case2 : modem is connecting to internet
Type **ato** to enter modem's internet connecting state
CONNECT → response from modem
and then type
at+i<cr> to enter TNS010i's commands mode
+I_OK → the response from TNS010i

Turn on DEBUG mode

Debug mode on = print out some more information when chip is in Processing.

Command:	at+idebug1<CR>
Response:	**+I_OK**

Turn off DEBUG mode

Command:	at+idebug0<CR>
Response:	**+I_OK**

Set up TCP time out

Command: at+isettcpto=<TCP time out value ><CR>
Response: **+I_OK**

Example: at+isettcpto=8<CR> (1-255 Sec.)
+I_OK

Display TCP time out value

Command: at+isettcpto=?<CR>
Response: **<Time-out value>**
+I_OK

Example: at+isettcpto=?<CR>
8
+I_OK

Set up DNS time out

Command: at+isetdnsto=<DNS time out value ><CR>
Response: **+I_OK**

Example: at+isetdnsto=8<CR> (1-255 Sec.)
+I_OK

Display DNS time out value

Command: at+isetdnsto=?<CR>
Response: **<Time-out value>**
+I_OK

Example: at+isetdnsto=?<CR>
8
+I_OK

Set up Modem time out

Command: at+isetmodemto=<time out value ><CR>
Response: **+I_OK**
Example: at+isetmodemto=8<CR> (1-255 Sec.)
+I_OK

Display Modem time out

Command:	at+isetmodemto=?<CR>
Response:	**<Time-out value>**
	+I_OK
Example:	at+isetmodemto=?<CR>
	8
	+I_OK

Detect and automatic set up GPRS/modem interfacing baud rate.

Command:	at+idetect<CR>
Response:	**xxxx**
	+I_OK
Example:	at+idetect<CR>
	9600
	+I_OK

Manual set up GPRS/modem interfacing baud rate

Command:	at+isetbaud=<baud rate><CR>
Response:	**+I_OK**

Available Baud rate list:

57600 BPS.(default)
38400
19200
9600
4800
2400
1200

Example: at+isetbaud=9600<CR>
+I_OK

Display GPRS/modem interfacing baud rate

Command:	at+isetbaud=?<CR>
Response:	**<baud rate value>**
	+I_OK
Example:	at+isetbaud=?<CR>
	9600
	+I_OK

Data counter

Check amount of total data transfer

Command:	at+icount=?<CR>
Response:	**<data transfer counter>** Bytes **+I_OK**
Example:	at+isetbaud=?<CR> **1371** **+I_OK**

Reset data transfer counter

Command:	at+icount=0<CR>
Response:	**+I_OK**
Example:	at+isetbaud=0<CR> **+I_OK**

note: the data counter command available only in ver 4.03 up

Modem's flow Control

Setting Modem's data flow control

Command:	at+icom1ctrl=hard/none<CR>
Response:	**+I_OK**
Example:	at+icom1ctrl=none<CR> **+I_OK**

Read Modem's flow control status

Command:	at+icom1ctrl=?<CR>
Response:	**HARD/NONE** **+I_OK**
Example:	at+icom1ctrl=?<CR> **NONE** **+I_OK**

note : default = none
Dev-010i-B's modem connection is null modem type

Power Down mode (for energy saving)

Put TNS010i into power down mode

Command:	at+ipwrdwn<CR>
Response:	**+I_OK**

**** In power down mode TNS010i will reduce the consume current to 25 uA.**
**** return to active mode by sending a reset pulse.**

GRPS phone / Modem selection

When power on reset. TNS010i will load a modem time out default value according to the logic level below.

GPRS sel. = high(or no connect)
LAND LINE MODEM sel. = low (connect to GND.)

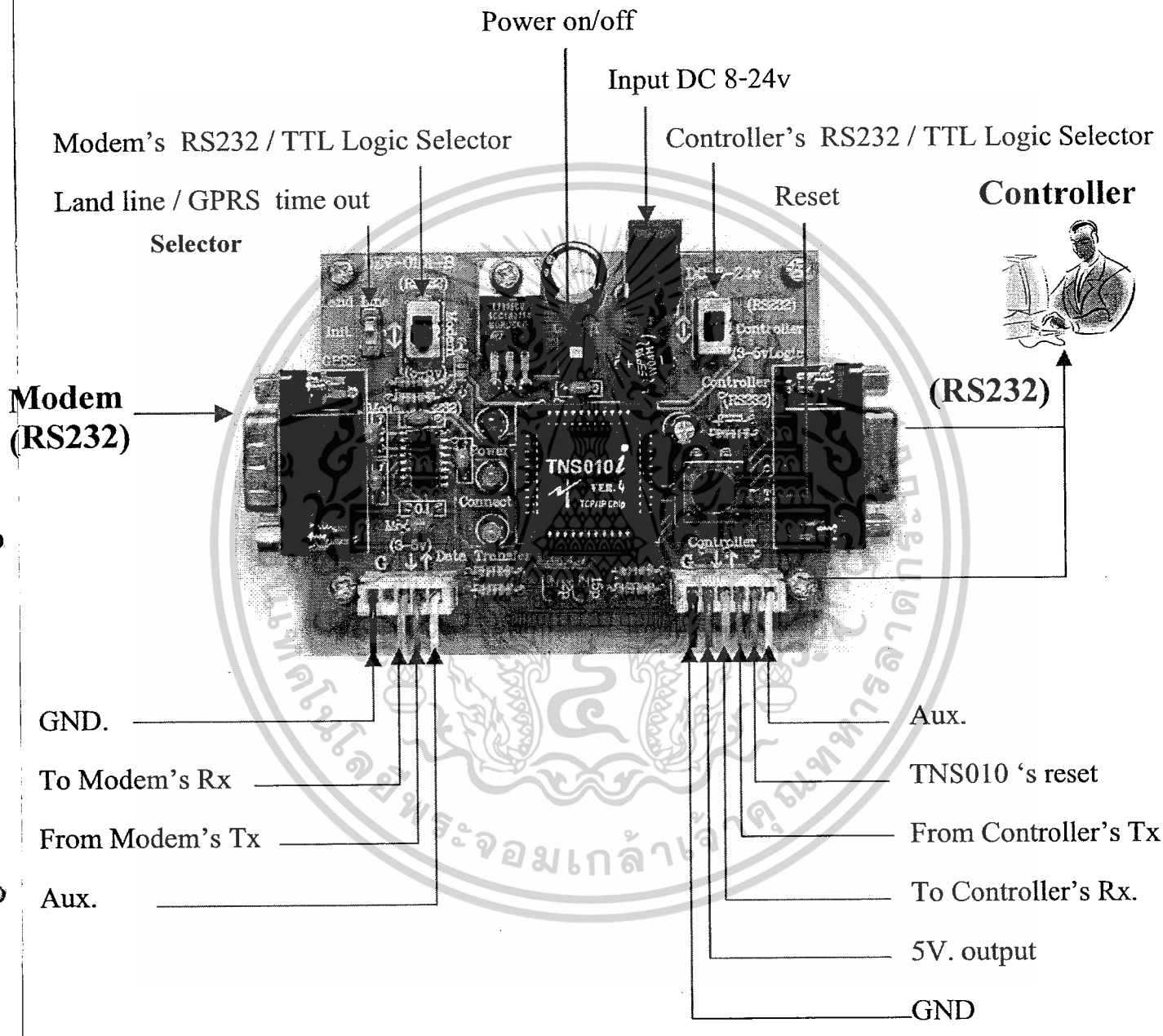
Note:
GPRS modem time out default value = 10 Sec.
Land line modem time out default value = 60 Sec.

you can change these default value by issue
at+isetmodemto=xxx

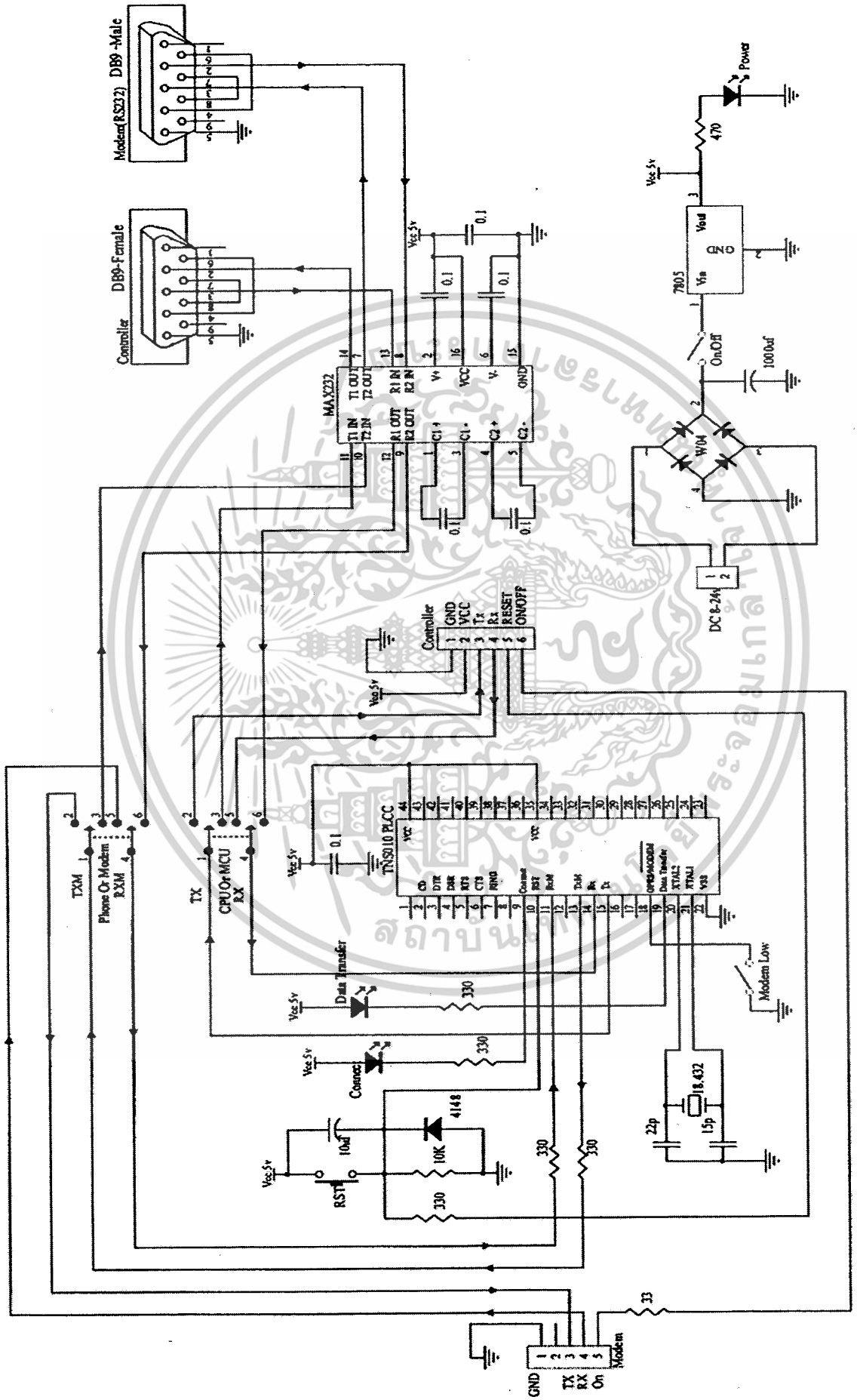
Error message response

```
+I_ERROR 0 PPP_CLOSED
+I_ERROR 1 PPP_NOT_ESTABLISH
+I_ERROR 2 DNS_IP_NOT_SET
+I_ERROR 3 DIAL_NOT_SET
+I_ERROR 4 MODEM_NOT_RESPONSE
+I_ERROR 5 TCP_TIME_OUT
+I_ERROR 6 DNS_TIME_OUT
+I_ERROR 7 PASSWORD_NOT_GOOD
+I_ERROR 8 CONNECT_LOST
```

Development board DEV-010i-B wiring guide

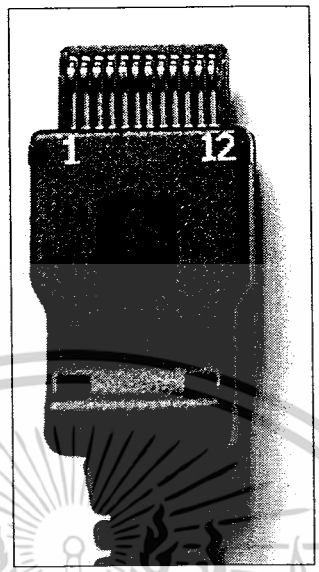


DEV-010i-B schematic



Popular GPRS phone data link details

Siemens S45 connector's pin out



1. Gnd
2. –
3. Charge
4. –
5. Tx (out from phone)
6. Rx (into phone)

Siemens C55

2. Gnd.
3. Tx
4. Rx

Ericsson T65/68/200 connector's pin out

1. Charge
2. Gnd
3. –
4. Gnd
5. –
6. –
7. TX (out from phone)
8. RX (into phone)
9. Power ON (TRIG LOW) **** ONLY ERICSSON T65/T68 ****

Quick Start Guide

TNS010i works well with any GPRS phone or GPRS modem module.
Such as

- Ericsson T65/T68/T200
- Siemens S45/ C55
- Wavecom Integra-GPRS (GPRS module)
- Etc.

** Recommendation for Wavecom integra **

1. Set GPRS modem to baudrate = 57600
   by send AT+IPR=57600 and
2. Save this setting value in it's EEPROM
   by send AT&W
3. Send AT+WOPEN=0

Following is the sample how to start TNS010i with Ericsson/Siemens phone

Testing phone parameters

1. Ensure that the GPRS phone have been activated and tested to verify the connection.

   In case of Ericsson, and Siemens C55 you need to purchase a data link cable (built in RS232 IC.) available from it's dealer.
2. Set the phone parameter according to the recommendations from your GPRS service provider. You may consult a sample web page of how to set at
   http://www.mobilelife.co.th/mobilelife/t/customertools/mobilesetting/manual/gprs/index.htm
   or from your local GRPS 'provider web site
   Note that the parameters may be vary from country to another, and from one provider to another.
3. Add "New Hardware" GPRS phone/modem to your PC (you will need GPRS phone driver that should come with the phone).
4. Add "New Internet Connection". Choose your GPRS phone as a modem
5. Connect the GPRS phone via its data link cable (built in RS232 IC) and connect to Internet by choosing a GPRS modem.
6. Try surfing the web page via your GPRS phone

If you are successful in surfing the web page with the GPRS phone then proceed to the next step.

Testing the TNS010i chip on Window's Hyperterminal program

As an alternative to reduce the overall manufacturing cost and the package size, a data link cable with no RS232IC in its cable can be used given that there is the same logic voltage level to and from the MCU and the GPRS phone.

7. Close all the Internet connections in PC including any web browser. Then remove the data link cable.
8. Open Hyper terminal program in the Windows (N-8-1 19,200 BPS)
9. Connect the GPRS phone to the a new development board 's data link cable(no RS232 IC) and turn the power on.
10. As soon as the development board is powered up, the "I_READY" sign should appear on the Hyper terminal screen.

.. Type the following sequence of commands in its order

→ at+i <ENTER> /* TCP/IP chip response testing */
← I_OK

→ at+idetect<ENTER> /* detect/set modem baudrate */
← 57600 /* modem's baud rate */
← I_OK

Skip the following gray colour steps if your GPRS modem's parameter was set correctly

---

→ +++<ENTER> /* quit TNS010's commands mode */
← I_OK

wait 0-3 Sec. (depend on brand /model of modem)

→ at<ENTER> /* Initiate modem's commands mode */
← OK

→ at+cgdcont=xxx /* xxx = GPRS parameter setting Example = 1,"ip","internet" */
← OK

→ at+i <ENTER> /* Enter TNS010's commands mode */
← I_OK

→ at+isetd=xxx /* xxx = ISP dial number Example = *99***1# */
← I_OK

→ at+iopen /* connect to ISP and open the TCP/IP socket */
← I_OK

→ at+ihttp://xxx /* xxx=url IP / url domain name */
Example =203.130.155.66/test.php?content1=123&content2=345 </l , /d , /h>
← I_OK
← (Data's length or date/month/year or full header or nothing (default) depend on the type of command's suffix (/l,/d,/h))
← <html> .....<html> /* HTTP response data*/

ts Distributor: http://www.es.co.th
cal Support: tcpipchip@yahoo.com