เครื่องตรวจจับอุณหภูมิและความชื้นแบบไร้สาย

WIRELESS THERMOMETER AND HUMIDITY DETECTOR

โดย

นาย ชัยมงคล ฉันทานุกูล

นาย อภิวิชญ์ อมรรัตนพันธ์

นายอะดุง สิงหพันธุ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญานิพนธ์ปีการศึกษา 2551

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **เครื่องตรวจจับอุณหภูมิและความชื้นแบบไร้สาย**

WIRELESS THERMOMETER AND HUMIDITY DETECTOR

ผู้จัดทำ

1.	นาย ชัยมงคล	ฉันทานุกูล	48010195
2.	นาย อภิวิชญ์	อมรรัตนพันธ์	48011071
3.	นายอะดุง	สิงหพันธุ์	48011096

(รศ.ดร.กอบชัย เดชหาญ)

(ดร. สิรภพ ตู้ประกาย)

เครื่องตรวจจับอุณหภูมิและความชื้นแบบไร้สาย

WIRELESS THERMOMETER AND HUMIDITY DETECTOR

โดย	นาย ชัยมงคล	ฉันทานุกูล	48010195
	นาย อภิวิชญ์	อมรรัตนพันธ์	48011071
	นายอะดุง	สิงหพันธุ์	48011096
อาจารย์ที่ปรึกษา	รศ.ดร.กอบชัย	เดชหาญ	
	ดร. สิรภพ	ตู้ประกาย	

บทคัดย่อ

โครงงานนี้เสนอการสร้างเครื่องวัดอุณหภูมิและความชื้นแบบไร้สายโดยอุปกรณ์ชิ้นนี้สามารถใช้ประโยชน์ในโรงงานหรือสภาพภูมิอากาศ เพื่อตรวจสอบความผิดปกติที่อาจเกิดขึ้นในจุดต่างๆและสามารถนำข้อมูลอุณหภูมิในช่วงเวลาต่างๆ มาแสดงผ่านทางหน้าจอคอมพิวเตอร์และเก็บบันทึกไว้ในฐานข้อมูล

ABSTRACT

This project presents thermometer and humidity wireless which can be useful in the plant or forecast the weather for detect any changing in that area , it also can send and display information on monitor and recorded in computer 's database.

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 แนวคิดของโครงงาน

การตรวจสภาวะแวดล้อมอุณหภูมิภายในโรงงาน ได้มีการพัฒนาเครื่องมือ และระบบต่างๆ โดยอาศัยเทคโนโลยีเข้ามาช่วย เพื่อให้รู้อุณหภูมิของโรงงานในแต่ละจุดว่าเป็นอย่างไร เมื่อมีการตรวจสอบก่อนทำให้สามารถเตรียมพร้อมที่จะรับมือกับสถานการณ์และสามารถหาแนวทางป้องกันการเกิดเพลิงไหม้

สภาวะอุณหภูมิภายในโรงงานนั้นใช้เป็นข้อมูลในการอ้างอิงคุณภาพของโรงงานอีกทั้งความปลอดภัยของแต่ละโรงงานได้ เราสามารถนำอุปกรณ์ชิ้นนี้มาระบุหรืออ้างอิงการตรวจสอบโรงงานหรือการป้องกันอุบัติเหตุภายในโรงงาน โดยการตรวจสอบวัดอุณหภูมิ

โดยแนวคิดในการทำอุปกรณ์ตรวจจับอุณหภูมิและความชื้นสัมพัทธ์ไร้สาย แบ่งเป็น 2 ส่วน ส่วนแรกทำการทดสอบ ตัวรับ-ส่งข้อมูลไร้สาย TRW-2.4 GHz ว่ารับ-ส่งข้อมูลกันได้หรือไม่ ส่วนที่สองทำการทดสอบตัวเซนเซอร์อุณหภูมิและความชื้นสัมพัทธ์ต่อแบบ serial port เข้าคอมพิวเตอร์ โดยดูว่าทาง Hyperterminal อ่านข้อมูลจากตัวเซนเซอร์ได้หรือไม่ จากนั้นก็นำทั้งสองส่วนมารวมกัน จะได้ 1 อุปกรณ์ของตัวรับ กับ 1 สถานีของอุปกรณ์ตัวส่งแล้วทำซ้ำอีกสถานีโดยใส่ดีเลย์เวลาในการส่งจะได้สถานีที่ 2ของอุปกรณ์ตัวส่ง จากนั้นก็ประมวลผลทางคอมพิวเตอร์ด้วยภาษาซีชาร์ป

1.2 ความมุ่งหมายและจุดประสงค์ของการทำโครงงาน

1.2.1 เพื่อศึกษาหลักการทำงานของตัววัดอุณหภูมิและความชื้นสัมพัทธ์

1.2.2 เพื่อศึกษาลักษณะการเขียนโปรแกรมไมโครคอนโทรลเลอร์และภาษา C#

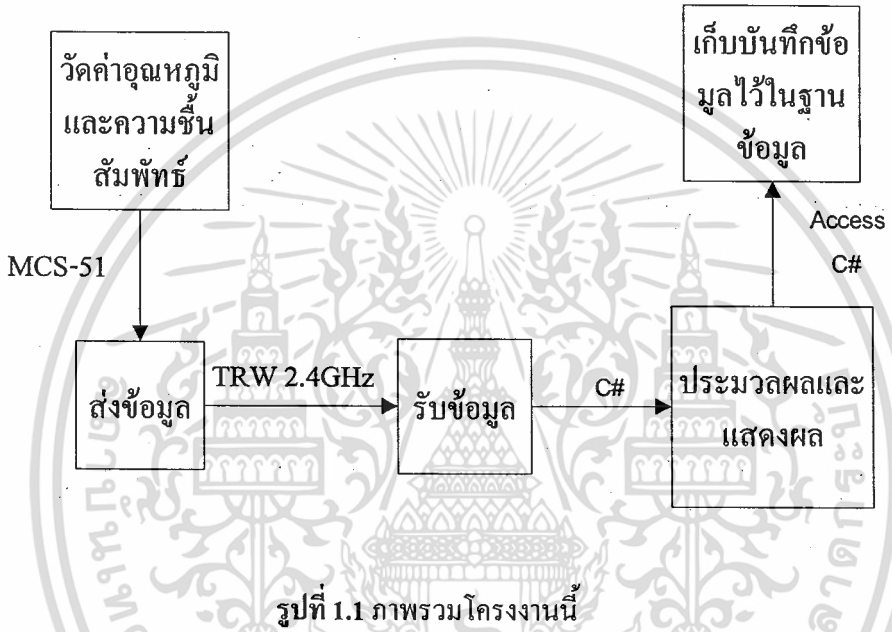
1.2.3 เพื่อศึกษาการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก

1.2.4 เพื่อฝึกทักษะในการสั่งให้คอมพิวเตอร์ทำงานตามที่ได้ออกแบบไว้

1.2.5 เพื่อศึกษากระบวนการส่งสัญญาณแบบไร้สาย

1.2.6 เพื่อสร้างระบบติดตามผลและส่งข้อมูลอุณหภูมิและความชื้นสัมพัทธ์ จากความรู้ได้ได้ศึกษาและฝึกทักษะที่กล่าวมาข้างต้น

ภาพรวมโครงงานเป็นการรับ/ส่งค่าอุณหภูมิและความชื้นสัมพัทธ์แบบไร้สาย โดยสถานีส่งทั้ง 2 สถานีใช้ MCS-51 เป็นตัวเขียนโปรแกรมควบคุมการทำงาน และส่งออกแบบไร้สายโดยการทำงานของ TRW 2.4 GHz แล้วรับข้อมูลด้วย TRW 2.4 GHz ด้วยเช่นกัน แล้วนำมาเข้าคอมพิวเตอร์ผ่านทางพอร์ตRS-232 นำค่าที่ได้ออกมประมวลผลโดยใช้ภาษา C# รันโดยโปรแกรม Vistual Studio และเก็บฐานข้อมูลบน Microsoft Access

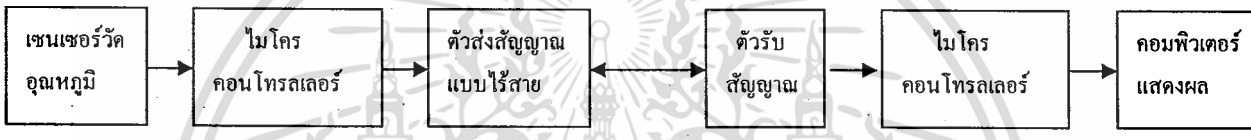


รูปที่ 1.1 ภาพรวมโครงงานนี้

บทที่ 2
ทฤษฎีและหลักการ

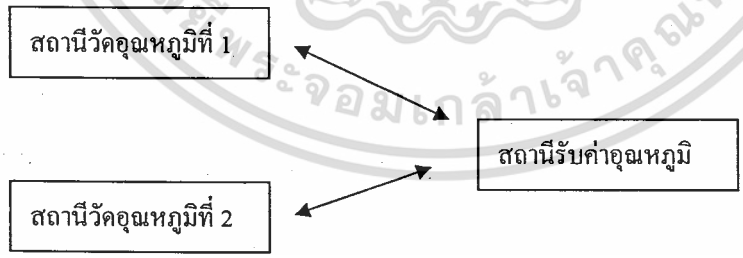
2.1 การทำงานของระบบติดตามผลและส่งข้อมูล

การทำงานของระบบเริ่มต้นจากการวัดค่าอุณหภูมิด้วยเซนเซอร์วัดอุณหภูมิ แสดงผลทางจอ LCD ที่เชื่อมต่ออยู่กับไมโครคอนโทรลเลอร์จากนั้นส่งข้อมูลดังกล่าว ให้ตัวส่งสัญญาณแบบไร้สายเพื่อส่งไปยังตัวรับสัญญาณ ข้อมูลที่รับได้ถูกส่งต่อเข้าไปยังคอมพิวเตอร์ที่ทำหน้าที่คำนวณหาค่าอุณหภูมิและบันทึกค่าดังกล่าว ดังบล็อกไดอะแกรมรูปที่ 2.1



รูปที่ 2.1 บล็อกไดอะแกรมของระบบติดตามผลและส่งข้อมูล

โดยระบบติดตามผลและส่งข้อมูลประกอบด้วย สถานีลูกที่เป็นสถานีวัดอุณหภูมิ 2 สถานี ดังรูปที่ 2.2 การส่งข้อมูลเข้ามาที่สถานีแม่ เป็นสถานีรับค่าอุณหภูมิของแต่ละสถานีที่ใช้วิธีทำการส่งคนละช่วงเวลา เนื่องจากทั้ง 2 สถานีใช้ความถี่เดียวกันในการส่งข้อมูลให้ตัวรับสัญญาณ โดยให้สถานีที่ 1 เป็นผู้ส่งเข้ามาจากนั้น สถานีที่ 2 จึงส่งข้อมูลเข้ามา



รูปที่ 2.2 ชุดอุปกรณ์ในระบบติดตามผลและส่งข้อมูล

2.2 ความชื้นและเสถียรภาพของอากาศ

แม้ว่าองค์ประกอบส่วนใหญ่ของอากาศจะเป็นก๊าซไนโตรเจน และก๊าซออกซิเจน แต่ก๊าซทั้งสองก็มิได้มีอิทธิพลในการเปลี่ยนแปลงสภาพของอากาศ ทั้งนี้เนื่องจากก๊าซทั้งสองมีจุดควบแน่น และจุดเยือกแข็งต่ำมาก อุณหภูมิของอากาศมิได้ต่ำพอที่จะทำให้ก๊าซทั้งสองเปลี่ยนสถานะได้

2.2.1 ความชื้น

ความชื้น (Humidity) หมายถึงจำนวนไอน้ำที่มีอยู่ในอากาศ ความชื้นของอากาศมีการเปลี่ยน แปลงอยู่ตลอดเวลา จะมากหรือน้อย ขึ้นอยู่กับความดันและอุณหภูมิ

ความชื้นสัมพัทธ์ (Relative Humidity) หมายถึง "อัตราส่วนของปริมาณไอน้ำที่มีอยู่จริงในอากาศต่อปริมาณไอน้ำที่จะทำให้อากาศอิ่มตัว ณ อุณหภูมิเดียวกัน" หรือ "อัตราส่วนของความดันไอน้ำที่มีอยู่จริง ต่อความดันไอน้ำอิ่มตัว" ค่าความชื้นสัมพัทธ์แสดงในรูปร้อยละ(%)

ความชื้นสัมพัทธ์ = (ปริมาณไอน้ำที่อยู่ในอากาศ / ปริมาณไอน้ำที่ทำให้อากาศอิ่มตัว) x 100% หรือ

ความชื้นสัมพัทธ์ = (ความดันไอน้ำที่อยู่ในอากาศ / ความดันไอน้ำที่ทำให้อากาศอิ่มตัว) x 100%

2.3 เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

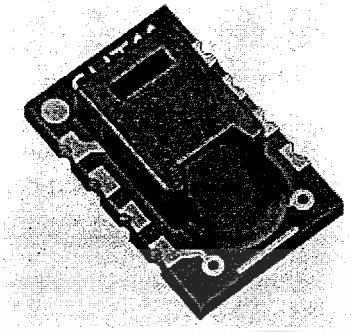
2.3.1 คุณสมบัติของเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์เบอร์ SHT 15

เป็นโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์จาก Sensirion (www.sensirion.com) มีขนาดเล็กและเพื่อความสะดวกในการใช้งานจึงได้ติดตั้งลงบนแผ่นวงจรพิมพ์และต่อคอนเน็กเตอร์ 8 ขา เพื่อให้สามารถติดตั้งลงบนแผ่นต่อวงจรเพื่อทำการทดลองได้ง่าย รวมไปถึงการนำไปประยุกต์ใช้งานจริงด้วยในรูปที่ 2.3 แสดงรูปร่างของโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์ และการจัดขา ส่วนคุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

คุณสมบัติทางด้านเทคนิค

- ทำหน้าที่เป็นทั้งตัววัดอุณหภูมิและความชื้นสัมพัทธ์ได้ในตัวเดียวกัน โดยไม่ต้องใช้อุปกรณ์ต่อเสริมภายนอก
- สามารถวัดความชื้นสัมพัทธ์ได้ในช่วง 0-100 เปอร์เซ็นต์ และอุณหภูมิช่วง -40 ถึง 120 องศาเซลเซียส
- สามารถกำหนดความละเอียดของย่านการวัดได้
- สามารถปรับเทียบได้ ค่าที่วัดได้เป็นค่าทางดิจิตอล
- มีขนาดเล็กและพลังงานต่ำ
- ใช้พลังงานน้อย โดยทำงานในย่านแรงดันไฟเลี้ยง +2.4 ถึง +5.5 โวลต์
- เสถียรภาพในการทำงานสูง
- สามารถใช้งานได้ต่อเนื่องยาวนาน

- ปิดการทำงานตัวเองอัตโนมัติเมื่อไม่มีการใช้งาน



รูปที่ 2.3 รูปร่างเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

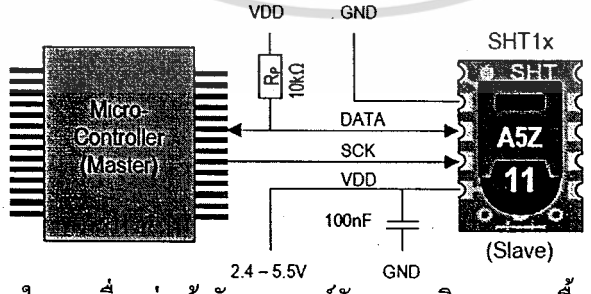
คุณลักษณะของเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์เบอร์ SHT 15 เป็นชิพเดี่ยวที่ใช้ในการตรวจวัดอุณหภูมิและความชื้นสัมพัทธ์ ซึ่งค่าที่วัดได้นั้นมีการสอบเทียบแล้วจึงเป็นค่าที่เชื่อถือได้ ค่าดิจิตอลที่วัดได้จะยังคงมีเสถียรภาพถึงแม้จะมีการใช้งานที่ต่อเนื่องยาวนาน เนื่องมาจากความสามารถของเทคโนโลยี micro – machining ทางด้านการผลิตสมัยใหม่บน ซีมอส (CMOS) ภายในอุปกรณ์ชิ้นนี้มี capacitive – polymer ที่ใช้ในการตรวจวัดอุณหภูมิและความชื้นสัมพัทธ์ โดยค่าที่วัดได้จะเป็นค่าดิจิตอล

2.3.2 การเชื่อมต่อกับเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

การจ่ายไฟให้อุปกรณ์

เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ สามารถทำงานที่แรงดันระหว่าง 2.4 ถึง 5.5 โวลต์ โดยจะใช้เวลา 11 ms จากสภาวะหยุดนิ่ง (sleep) เข้าสู่สภาวะที่พร้อมในการใช้งาน ซึ่งภายในช่วงเวลาดังกล่าวต้องไม่มีการส่งคำสั่งใดๆมายังอุปกรณ์



รูปที่ 2.4 รูปวงจรในการเชื่อมต่อเข้ากับเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

การเชื่อมต่อแบบอนุกรม (เชื่อมต่อแบบโดยตรง 2 สาย)

การเชื่อมต่อแบบอนุกรมของเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์นี้เหมาะสมสำหรับการส่งค่าที่ได้จากการตรวจวัดและการจ่ายพลังงาน โดยไม่สามารถทำงานได้กับรูปแบบการเชื่อมต่อแบบ I²C

- ขาสัญญาณนาฬิกา (SCK) ทำหน้าที่รับสัญญาณนาฬิกาเพื่อกำหนดจังหวะในการสื่อสารข้อมูลระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

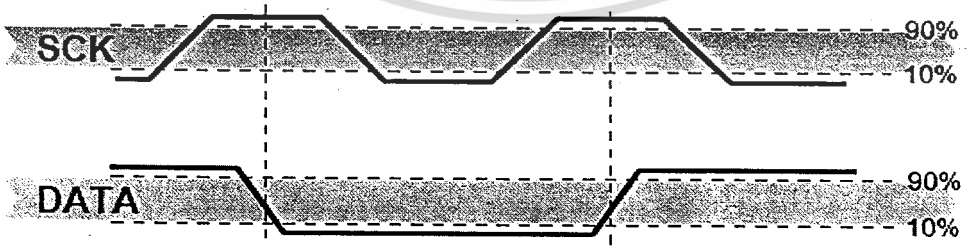
- ขาสัญญาณรับ / ส่งข้อมูล (Data) ขาข้อมูล 3 สถานะ ใช้เป็นขาสัญญาณสำหรับรับ / ส่งข้อมูลระหว่างกันระหว่างอุปกรณ์ โดยการอ่านข้อมูลจะทำที่ขอบขาขึ้นของขาสัญญาณนาฬิกาในระหว่างการส่งสัญญาณสายสัญญาณรับ / ส่งข้อมูล จะต้องคงสภาวะเดิมไว้ในขณะที่ขาสัญญาณนาฬิกามีสถานะ “1” ในการใช้งานควรต่อตัวต้านทาน 220 โอมห์ พูลอัพที่ขานี้ เพื่อหลีกเลี่ยงการแกว่งของสัญญาณที่จะจำทำการอ่านค่าเกิดความผิดพลาด

2.3.3 รูปแบบการสื่อสารข้อมูลของเซนเซอร์วัดอุณหภูมิ

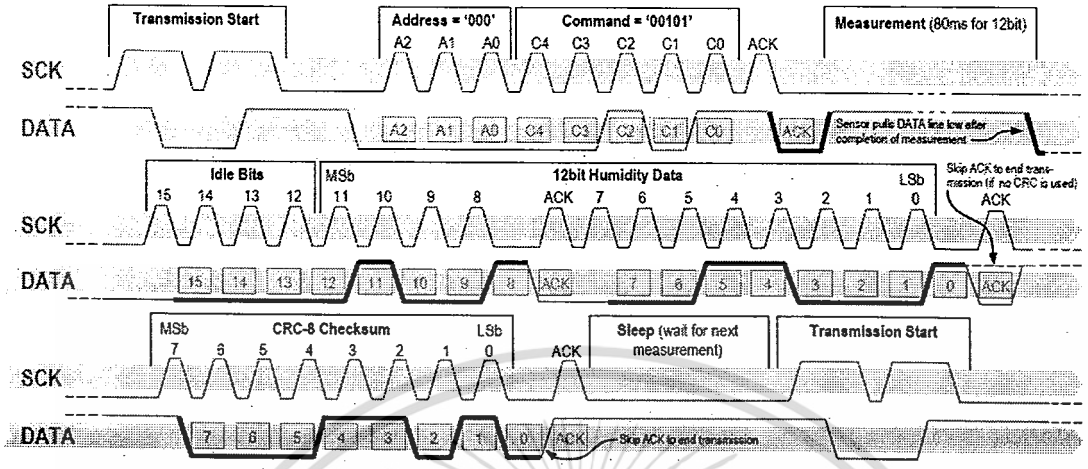
การส่งคำสั่ง

ในสภาวะเริ่มต้นก่อนการส่งข้อมูล คำสั่งจากไมโครคอนโทรลเลอร์ไปยังเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ จำเป็นต้องสร้างรูปแบบสัญญาณกระตุ้นผ่านขาสัญญาณนาฬิกาและขาสัญญาณรับ / ส่งข้อมูล เพื่อให้ตรงกับเงื่อนไขที่เรียกว่า สภาวะเริ่มต้นการส่งสัญญาณ (Transmission start) นั่นคือขาสัญญาณรับ / ส่งข้อมูล ต้องถูกทำให้เป็นลอจิก “0” นานอย่างน้อย 1 ลูกคลื่นของสัญญาณนาฬิกา หลังจากนั้นเซนเซอร์วัดอุณหภูมิ จะทราบทันทีว่าข้อมูลต่อจากนี้เป็นคำสั่ง

โดยลำดับค่าของคำสั่ง จะประกอบด้วยแอดเดรส 3 บิต (เฉพาะ “000” เท่านั้นที่สามารถใช้งานได้) จากนั้นจึงตามด้วยคำสั่ง 5 บิต เซนเซอร์วัดอุณหภูมิจะแสดงว่าได้รับคำสั่งคำสั่งที่ส่งมานั้นเรียบร้อยแล้วโดยการทำให้ขาสัญญาณรับ / ส่งข้อมูล เป็น “0” (ACK bit) หลังจากขอบขาลงของขาสัญญาณนาฬิกา ลูกที่ 8 ของขาสัญญาณนาฬิกา สายสัญญาณรับ / ส่งข้อมูล จะกลับไปเป็น “1” หลังจากขอบขาลงของขาสัญญาณนาฬิกาลูกที่ 9 ของขาสัญญาณนาฬิกา



รูปที่ 2.5 รูปแบบสัญญาณกระตุ้นผ่านขาสัญญาณนาฬิกา และขาสัญญาณรับ / ส่งข้อมูล



รูปที่ 2.6 รูปแบบสัญญาณคำสั่งของเซนเซอร์วัดอุณหภูมิ

ตาราง 2.1 รายละเอียดคำสั่งและข้อมูลสำหรับควบคุมการทำงานของเซนเซอร์

คำสั่ง	ข้อมูลคำสั่ง
สงวนไว้	0000x
อ่านค่าอุณหภูม(Measure Temperature)	000011
อ่านค่าความชื้นสัมพันธ์(Measure Humidity)	00101
อ่านค่ารีจิสเตอร์กำหนดสถานะ(Read Status Register)	00111
สงวนไว้	0101x ถึง 1110x
รีเซตการทำงาน(Soft Reset)ทำให้รีจิสเตอร์กำหนดสถานะกลับไปสู่ค่า Default และต้องใช้เวลาการทำงานอย่างน้อย 11 มิลลิวินาที จึงสามารถรับคำสั่งถัดไปได้	11110

หลังจากสร้างเงื่อนไขสภาวะเริ่มต้นการส่งสัญญาณแล้ว ไมโครคอนโทรลเลอร์สามารถส่งคำสั่งไปยังเซนเซอร์วัดอุณหภูมิ เพื่อกำหนดการทำงานได้ทันที โดยข้อมูลคำสั่งต่างๆ สำหรับการทำงานตามตารางที่กำหนด

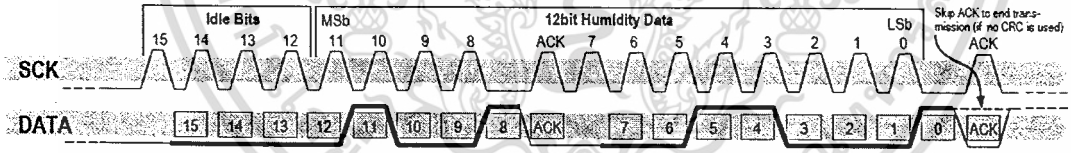
การสั่งงาน

หลังจากที่ส่งคำสั่ง ('00000011' สำหรับวัดค่าอุณหภูมิ) ไมโครคอนโทรลเลอร์จะต้องรอช่วงหนึ่งจนกว่าเซนเซอร์วัดอุณหภูมิ จะวัดค่าเสร็จสิ้น โดยจะใช้เวลา 11/55/210 ms โดยประมาณสำหรับ 8/12/14 บิต (ความยาวของข้อมูลที่วัดได้) โดยเวลาที่ใช้จริงอาจมีการเปลี่ยนแปลงไป ±15 เปอร์เซ็นต์ ขึ้นกับความเร็วของ

ออสซิลเลเตอร์ภายใน ในการสร้างสัญญาณขึ้นหลังจากที่ทำการวัดเสร็จสิ้น เซนเซอร์วัดอุณหภูมิจะทำให้สายสัญญาณรับ / ส่งข้อมูล เป็น “0” และเข้าสู่โหมดหยุดนิ่ง(idle) ไมโครคอนโทรลเลอร์จะต้องรอจนกว่าจะได้รับสัญญาณข้อมูลพร้อม (data ready) จึงจะสามารถเริ่มส่งสัญญาณนาฬิกาในการอ่านข้อมูล ข้อมูลจากการวัดค่าจะถูกเก็บไว้ที่เซนเซอร์วัดอุณหภูมิ จนกว่าจะถูกอ่านออกไปดังนั้นไมโครคอนโทรลเลอร์จึงสามารถทำคำสั่งอื่นไปด้วยได้ขณะที่ทำการอ่านข้อมูลจากเซนเซอร์วัดอุณหภูมิ

ข้อมูลที่ได้จากการวัดความยาว 2 ไบต์ และข้อมูลตรวจสอบความผิดพลาด (CRC) จะถูกส่งออกมาจากเซนเซอร์วัดอุณหภูมิ โดยไมโครคอนโทรลเลอร์จะต้องทำการตอบกลับในแต่ละไบต์ โดยการทำให้สายสัญญาณรับ / ส่งข้อมูล เป็น “0” คำสั่งที่ออกมาบิตแรกจะเป็นบิตสำคัญสูงสุด (เช่น บิตที่สายสัญญาณรับ / ส่งข้อมูล ที่ตำแหน่งสัญญาณนาฬิกาลูกที่ 5 เป็นบิตสำคัญสูงสุดของค่าข้อมูลที่ยาว 12 บิต แต่ที่ตำแหน่งจะไม่ได้เป็นบิตสำคัญสูงสุดของค่าข้อมูลที่ยาว 8 บิต) การสื่อสารระหว่างกันจะสิ้นสุดลงหลังจากบิตตอบกลับ (acknowledge bit) ของข้อมูลตรวจสอบความผิดพลาดถ้าไม่มีการใช้งานบิตตรวจสอบความผิดพลาดไมโครคอนโทรลเลอร์ก็สามารถจบการส่งผ่านข้อมูลหลังไบต์ที่มีบิตสำคัญต่ำสุด ของค่าข้อมูลที่ได้จากการวัด โดยการทำให้เป็น “1”

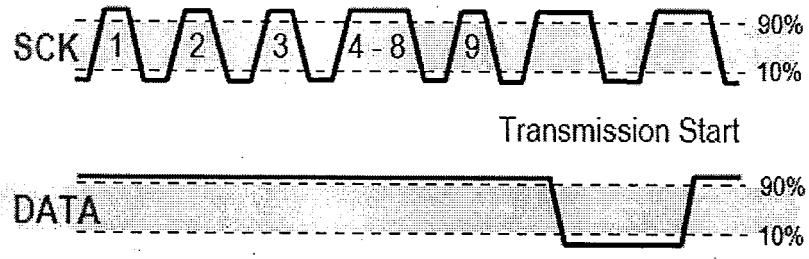
เซนเซอร์วัดอุณหภูมิจะเข้าสู่หยุดนิ่งอัตโนมัติหลังจากการวัดและการส่งข้อมูลเสร็จสิ้น โดยมีข้อระวังในการใช้งานหากไม่ต้องการให้อุปกรณ์มีความร้อนเพิ่มสูงขึ้นจากการใช้งานอีก 0.1 องศาเซลเซียส เซนเซอร์วัดอุณหภูมิควรจะทำงานด้วยเวลาที่ต่ำกว่า 10 เปอร์เซ็นต์ (เช่น กรณีสูงสุดทำการวัดทั้ง 2 ชนิด ต่อ 1 วินาที สำหรับทำการวัดแบบ 12 บิต ที่มีความแม่นยำ)



รูปที่ 2.7 รูปแบบสัญญาณข้อมูลของเซนเซอร์วัดอุณหภูมิ

2.3.4 รีเซตการเชื่อมต่อ (Connection Reset sequence)

เมื่อต้องการเริ่มต้นการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์วัดอุณหภูมิ หรือในกรณีที่การสื่อสารเกิดความผิดพลาด ต้องสร้างสัญญาณรีเซตการเชื่อมต่อขึ้นมาก่อน โดยทำให้ขาสัญญาณรับ / ส่งข้อมูล มีสถานะลอจิกเป็น “1” นานเท่ากับช่วงเวลาที่ป้อนสัญญาณนาฬิกาที่ขาสัญญาณนาฬิกาจำนวน 9 ลูกติดต่อกัน แล้วตามด้วยการสร้างสภาวะเริ่มต้นการส่งสัญญาณก่อนที่จะมีการส่งคำสั่งถัดไป โดยจะการรีเซตแค่การเชื่อมต่อเท่านั้น แต่สถานะของรีจิสเตอร์ที่กำหนดรูปแบบการทำงานยังคงเดิม



รูปที่ 2.8 สัญญาณรีเซตและการสร้างสภาวะเริ่มต้นการส่งสัญญาณ

2.3.5 การคำนวณค่าอุณหภูมิ

ในการอ่านค่าอุณหภูมิจากเซนเซอร์วัดอุณหภูมิ ผู้ใช้สามารถเลือกความละเอียดในการอ่านได้ในแบบ 14 บิต หรือ 12 บิต โดยที่ความละเอียด 14 บิต เป็นค่าตั้งต้น โดยผู้ใช้จำเป็นที่จะต้องอ่านข้อมูลดิบจากเซนเซอร์วัดอุณหภูมิเข้ามาก่อน จากนั้นจึงใช้กระบวนการทางคณิตศาสตร์เพื่อให้ได้ค่าอุณหภูมิออกมา โดยสามารถคำนวณได้จากสมการที่กำหนดมาจาก Sensirion ผู้ผลิตเซนเซอร์วัดอุณหภูมิ ดังนี้

$$\text{Temperature} = d1 + (d2 * SO_T) \qquad (2.1)$$

โดยที่ Temperature คือค่าอุณหภูมิจริง

d1 คือค่าคงที่ ขึ้นอยู่กับไฟเลี้ยงที่ป้อนให้กับขา V_{DD} ของเซนเซอร์วัดอุณหภูมิ

d2 คือค่าคงที่ ขึ้นอยู่กับความละเอียดของอุณหภูมิที่ต้องการจากเซนเซอร์วัดอุณหภูมิ

SO_T คือค่าอุณหภูมิที่อ่านได้จากเซนเซอร์วัดอุณหภูมิ

ตารางที่ 2.2 การกำหนดค่าคงที่ทางอุณหภูมิตัวที่ 1 และตัวที่ 2 เพื่อคำนวณค่าอุณหภูมิจริง

ไฟเลี้ยง	ค่าคงที่ทางอุณหภูมิตัวที่ 1 (d1)	
	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

ความละเอียด	ค่าคงที่ในหน่วยอุณหภูมิตัวที่ 2 (d2)	
	ในหน่วย C	ในหน่วย F
14 บิต	0.01	0.018
12 บิต	0.04	0.072

2.3.6 คำนวณค่าความชื้นสัมพัทธ์

สำหรับการอ่านค่าความชื้นสัมพัทธ์จากเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ สามารถเลือกความละเอียดในการอ่านได้ในแบบ 12 บิต หรือ 8 บิต โดยที่ความละเอียด 12 บิตเป็นค่าตั้งต้นหลักโดยที่จะต้องอ่านข้อมูลดิบจากเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ เข้ามาก่อน จากนั้นจึงใช้กระบวนการทางคณิตศาสตร์เพื่อให้ได้ค่าความชื้นสัมพัทธ์ออกมา โดยสามารถคำนวณได้จากสมการที่กำหนดมาจาก Sensirion ผู้ผลิตเซนเซอร์นี้ดังนี้

$$RH_{true} = (T\,^{\circ}C - 25)*(t_1 + t_2*SO_{RH}) + RH_{linear} \quad (2.2)$$

$$RH_{linear} = C1 + C2*SO_{RH} + C3*SO_{RH}^{2} \quad (2.3)$$

โดยที่ RH_{true} คือค่าความชื้นสัมพัทธ์จริง

$T\,^{\circ}C$ คือค่าอุณหภูมิจริงที่คำนวณได้จากสมการที่

t_1 และ t_2 คือค่าคงที่โดยขึ้นกับความละเอียดของความชื้นสัมพัทธ์ที่ต้องการตามตารางที่

C 1 , C2 และ C3 คือค่าคงที่โดยขึ้นกับความละเอียดของความชื้นสัมพัทธ์ที่ต้องการตามตารางที่

ตารางที่ 2.3 ค่าคงที่ในการแปลงค่าความชื้นสัมพัทธ์ในสมการที่ (2.3)

SO_{RH}	C1	C2	C3
12 บิต	-4	0.0405	-2.8*10-6
8 บิต	-4	0.648	-7.3*10-4

ตารางที่ 2.4 ค่าคงที่ในการแปลงค่าความชื้นสัมพัทธ์ในสมการที่ (2.2)

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

2.4. ทฤษฎีไมโครคอนโทรลเลอร์

โครงสร้างทางฮาร์ดแวร์ของไมโครคอนโทรลเลอร์เบอร์ AT89C52

ไมโครคอนโทรลเลอร์เป็นไมโครคอมพิวเตอร์แบบชิปเดี่ยว ที่มีหน่วยประมวลผลกลางขนาดเล็ก (8บิต – 16 บิต) และหน่วยประมวลผลที่สามารถเข้าถึงข้อมูลแบบบิต หน่วยความจำข้อมูลแบบฟูลดูเพล็กซ์ , วงจรไทเมอร์ (Counter/Timer) ที่อยู่ภายในไอซีที่สามารถต่อกับอุปกรณ์ที่ใช้ในการสร้างวงจรกำหนดสัญญาณ และตัวเก็บประจุก็สามารถใช้งานได้ง่าย

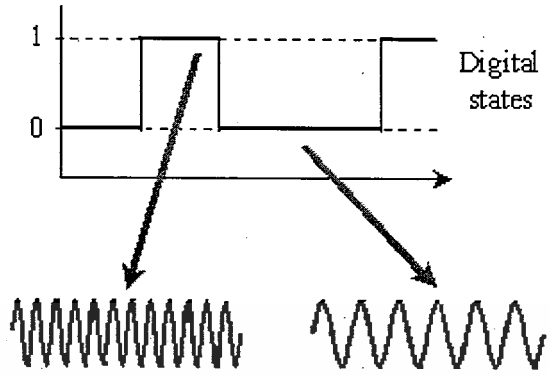
คุณสมบัติของไมโครคอนโทรลเลอร์เบอร์ AT89C52 มีดังนี้

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม 256 ไบต์
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถเป็นได้ทั้งอินพุตและเอาต์พุต
- มีขาพอร์ตสำหรับต่อใช้งาน 32 ขา โดยแบ่งเป็นพอร์ต 0 (8 บิต:P0.0-P0.7) พอร์ต 1 (8บิต:P1.0-P1.7) พอร์ต 2 (8 บิต:P2.0-P2.7) และพอร์ต 3 (8 บิต:P3.0-P3.7)
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 3 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 8 แหล่งกำเนิด

2.4.2 การมอดูเลชั่นแบบดิจิตอล

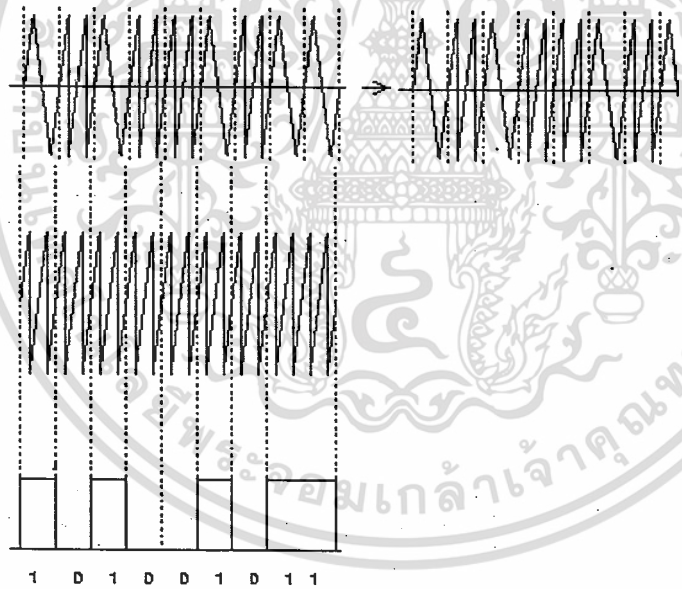
การสื่อสารในปัจจุบันได้นิยมนำเอาการมอดูเลตสัญญาณแบบดิจิตอล (Digital Modulation) มาใช้อย่างแพร่หลาย เนื่องจากระบบดิจิตอลให้ค่าความน่าเชื่อถือสูงกว่าระบบแอนาลอก (Analog System) และมีการรบกวนจากสัญญาณรบกวน (Noise) ต่ำ ซึ่งในปัจจุบัน อุปกรณ์ด้านระบบดิจิตอลได้มีการพัฒนาก้าวหน้าไปมาก ทำให้ต้นทุนลดต่ำลง นอกจากนี้ การมอดูเลตแบบดิจิตอลยังสามารถทำการเข้ารหัส (Encoder) ก่อนทำการมอดูเลตแล้วทำการถอดรหัส (Decoder) หลังการมอดูเลตทำให้การส่งข้อมูลมีความผิดพลาดน้อยลง ในการมอดูเลตสัญญาณดิจิตอล ที่นิยมใช้กันชนิดหนึ่งก็คือ การเปลี่ยนแปลงความถี่ของสัญญาณคลื่นพาห์ตามสัญญาณดิจิตอล (Frequency Shift Keying:FSK)

การเปลี่ยนแปลงความถี่คลื่นพาห์ตามสัญญาณดิจิตอล (Frequency Shift Keying:FSK)วิธีการ FSK นี้คือใช้ความถี่ของเสียงสองความถี่สำหรับแทนสัญญาณ ลอจิก “1” และลอจิก“0” ฝ่ายรับก็พยายามตรวจจับสองความถี่ที่ว่านี้มาแปลงเป็นสัญญาณลอจิกกลับคืน ความถี่ของเสียงทั้งสองเสียงต้องห่างกันพอที่จะแยกออกจากกันได้โดยวงจรอิเล็กทรอนิกส์ และจะต้องไม่ห่างเกินจนตกขอบของความ สามารถของตัวกลางที่จะนำพาไปได้ ดังรูป แสดงหลักการทำงานของ FSK



รูปที่ 2.9 หลักการทำงานของ FSK

รูปคลื่นสัญญาณที่ได้จากวิธีการนี้ สัญญาณดิจิตอลจะควบคุมความถี่ของสัญญาณที่ได้จากวงจรมอดูเลชั่น โดยรูปคลื่นจะมีความถี่สูงเมื่อระดับสัญญาณดิจิตอลเป็น "1" และมีความถี่ต่ำเมื่อระดับสัญญาณเป็น "0" ซึ่งมีอัตราการส่งข้อมูลไม่สูงมากนัก พอๆกับวิธีการ ASK แต่มีข้อดีที่สามารถทนทานต่อสัญญาณรบกวนได้สูงกว่า



รูปที่ 2.10 รูปคลื่นของการมอดูเลตแบบ FSK

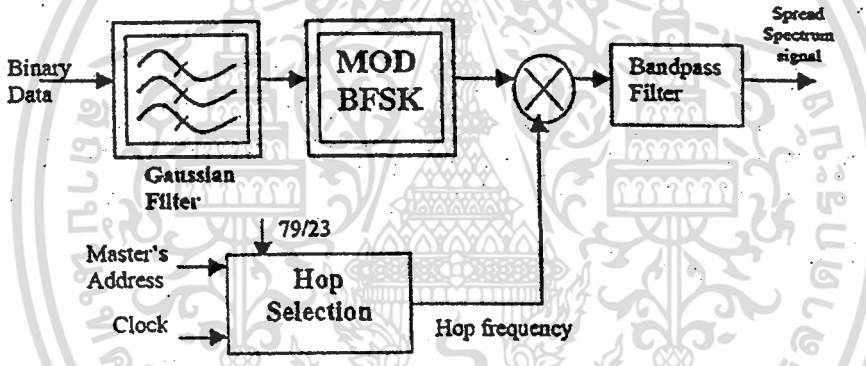
เนื่องจากแถบความถี่คลื่นที่ตัวกลางยอมให้ผ่านไปได้อยู่ในช่วง 300 เฮิรตซ์ ถึง 3400 เฮิรตซ์ จึงสามารถแบ่งความถี่ในย่านนั้นออกเป็น 4 คลื่นเสียงที่สำคัญ สำหรับสถานีส่งสองเสียง สถานีรับสองเสียง เนื่องจากต้องการให้การติดต่อเป็นแบบฟูลดูเพล็กซ์ คือ ทั้งรับและส่งได้ในเวลาเดียวกันจำเป็นจะต้องแยกสถานีออกเป็นสองฝ่าย ออริจิเนลหรือฝ่ายเริ่มการติดต่อ และอีกฝ่ายเรียกว่า คำตอบ (Answer) จะต้องใช้ความถี่

อีกสองความถี่ที่แตกต่างไปจากฝ่ายส่ง (เพื่อป้องกันการรบกวนกันเอง) สำหรับแทนสัญญาณลอจิก “0” และ “1” เช่นเดียวกันจะได้รับและส่งในเวลาเดียวกันเป็นฟูลดูเพล็กซ์ได้

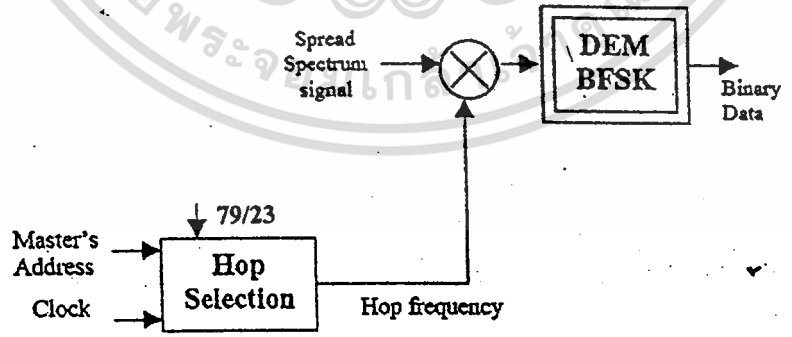
การเปลี่ยนความถี่ตามสัญญาณดิจิตอลแบบเกาส์เซี่ยน (Gaussian Frequency Shift Keying:GFSK)

GFSK พัฒนามาจากเทคนิคการมอดูเลตแบบ FSK เพื่อเพิ่มประสิทธิภาพในการใช้แบนด์วิดท์ โดยข้อเสียของการมอดูเลชั่นแบบ FSK ก็คือ ในขณะที่ข้อมูลเปลี่ยนจาก 0 เป็น 1 หรือ 1 เป็น 0 จะเกิดการเปลี่ยนเฟสของสัญญาณคลื่นพาห์อย่างรวดเร็ว อาจจะสูงขึ้นหรือต่ำลง ซึ่งก็มีผลทำให้ความถี่คลื่น พาห์จริงสูงกว่าหรือต่ำกว่า f0 หรือ f1 ที่กำหนดไว้ ซึ่งจะทำให้แบนด์วิดท์ที่ได้กว้างขึ้น

ดังนั้นเพื่อลดปัญหาดังกล่าวจึงนำเอาสัญญาณข้อมูลที่จะไปทำการมอดูเลตแบบ FSK มาผ่านวงจรกรองแบบเกาส์ (Gaussian Filter) ก่อน ซึ่งจะให้การเปลี่ยนแปลงของสัญญาณข้อมูล เป็นแบบค่อยๆขึ้นหรือค่อยๆลง โดยมีความโค้งเป็นแบบเกาส์เซี่ยนพัลส์ จากนั้นจึงค่อยนำไปมอดูเลตแบบ FSK ก็ทำให้แบนด์วิดท์ที่ได้แคบลงเมื่อเทียบกับการมอดูเลตแบบ FSK และจะทำให้ได้อัตราการส่งข้อมูลสูงขึ้น โดยบล็อกไดอะแกรมของการมอดูเลตและการดีมอดูเลตแบบ GFSK แสดงดังรูปที่ และ รูปที่ 2.11 และ 2.12 ตามลำดับ



รูปที่ 2.11 การมอดูเลตแบบ GFSK



รูปที่ 2.12 การดีมอดูเลตแบบ GFSK

2.5 ตัวส่ง/รับสัญญาณ TRW 2.4GHz

2.5.1 คุณสมบัติของตัวส่ง/ตัวรับสัญญาณ TRW 2.4 GHz

เป็นโมดูลสำเร็จรูปที่ใช้รับ/ส่งข้อมูลในแบบอนุกรม ใช้กับความถี่ 2.4 GHz ปรับแต่งสำเร็จรูปพร้อมกับมีสายอากาศในตัวซึ่งสามารถใช้งานได้ในระยะไกล 280 เมตร (ความเร็วข้อมูล 250 Kbps) หรือระยะ 150 เมตร (ความเร็วข้อมูล 1 Mbps) ในพื้นที่โล่งแจ้ง

- ความถี่ในการใช้งาน 2.4-2.524 GHz
- มีรูปแบบและความเร็วในการรับ - ส่งข้อมูลโดยส่งแบบ GFSK (Gaussian Frequency Shift Keying)
- ทำงานที่ความต่างศักย์ทางไฟฟ้า 3 โวลต์
- กำลังงานเอาต์พุต +4 dBm
- ความเร็วในการรับส่งข้อมูลถึง 1Mbps ,250 Kbps
- ขนาด 20.0 x 36.7 x 2.4 มิลลิเมตร
- ทำงานที่อุณหภูมิ -40 ถึง +85 เซนติเกรด
- ระยะทางในการรับส่งสัญญาณ 280 เมตร (250 Kbps) ,150 เมตร (1Mbps)
- มีสายอากาศรับส่งสัญญาณในตัวส่ง/รับสัญญาณ TRW 2.4GHz

2.5.2 การเชื่อมต่อกับตัวส่ง/รับสัญญาณ TRW2.4GHz

ขาทั้ง 10 ขา ของตัวรับ/ส่ง สัญญาณ TRW 2.4GHz มีการทำงานที่แตกต่างกันโดยสามารถดูได้จากตารางที่ 2.5 และการเซตค่าของขาแต่ละขาสามารถดูได้จากตารางที่ 2.6

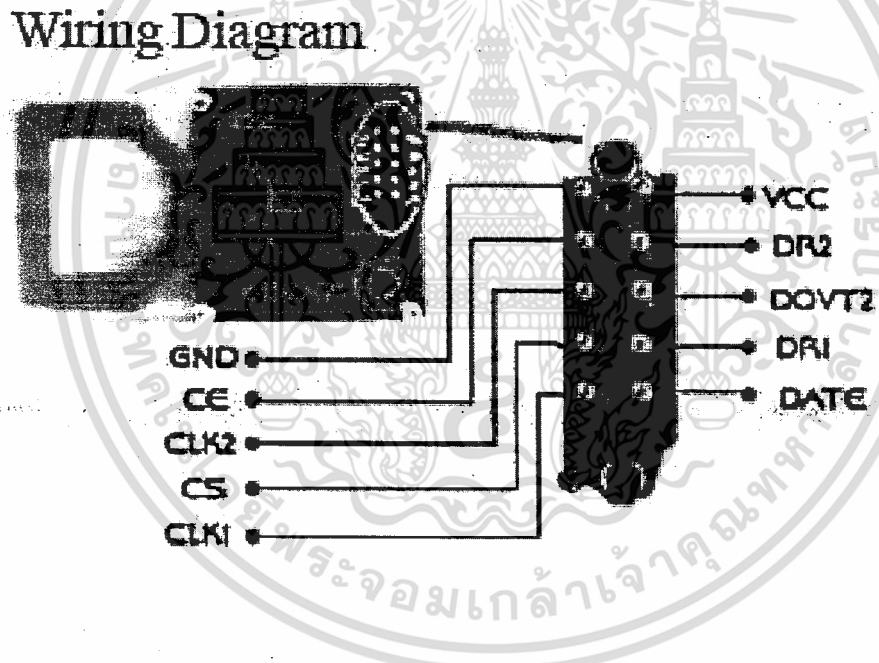
ตารางที่ 2.5 แสดงลักษณะการทำงานของขาตัวส่ง/รับสัญญาณ TRW 2.4GHz

ขา	ชื่อ	ลักษณะการทำงานของขา	รายละเอียด
1	GND	Power	Ground (0v)
2	CE	Input	Chip Enable activates Rx or Tx mode
3	CLK2	I/O	Clock Output/Input for Rx data channel 2
4	CS	Input	Chip Select activates Configuration mode
5	CLK1	I/O	Clock Input (Tx) & I/O (Rx) for data channel 1 3-wire interface
6	DATA	I/O	Rx data channel 1/Tx data input /3 –wire interface
7	DR1	Output	Rx data ready at data channel 1 (Shock Burst Only)
8	DOUT2	Output	Rx data channel 2
9	DR2	Output	Rx data ready at data channel 2 (Shock Burst Only)
10	VCC	Power	Power Supply (+3v DC)

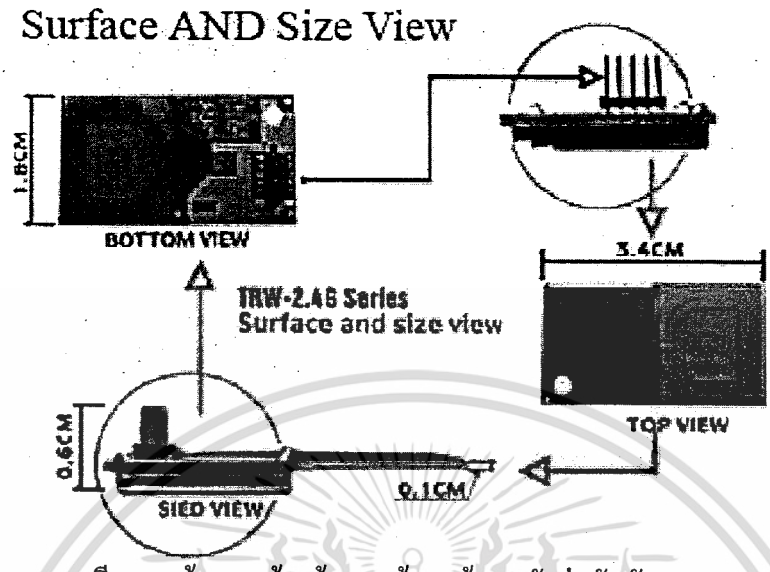
ตารางที่ 2.6 การเซตค่าของแต่ละขาในโหมดต่างๆ

โหมด	PWR_UP	CE	CS
Active (Rx/Tx)	1	1	0
Configuration	1	0	1
Stand by	1	0	1
Power down	0	X	X

รูปร่างและสัญลักษณ์ของตัว ส่ง/รับ สัญญาณTRW 2.4 GHz โดยจะมีขาทั้งหมด 10 ขาโดยแต่ละขากำหนดค่าตามรูปที่ 2.13



รูปที่ 2.13 ลักษณะขาของตัวส่ง/รับสัญญาณ TRW2.4GHz



รูปที่ 2.14 รายละเอียดทางด้านบน ด้านข้างและด้านหน้าของตัวส่ง/รับสัญญาณ TRW2.4GHz

2.5.3 รูปแบบการสื่อสารข้อมูลของตัวส่ง/รับสัญญาณ TRW2.4GHz

โหมดในการใช้งานของตัวส่ง/รับสัญญาณ TRW 2.4GHz มีอยู่ 2 โหมด

1. Shock Burst Mode
2. Direct Mode

โดยในโครงงานนี้ได้กำหนดให้โมดูลความถี่วิทยุมีการทำงานในโหมดชอคเบิร์ส (Shock Burst) ซึ่งมีรายละเอียดดังต่อไปนี้

โหมดชอคเบิร์ส

โหมดชอคเบิร์สเป็นการใช้เทคโนโลยีรับ/ส่งข้อมูลบนชิพแบบเข้าก่อน-ออกก่อน(First in –First out) โดยในการส่งข้อมูลมีทั้งระดับในการส่งบิตข้อมูลมีทั้งความเร็วต่ำและระดับความเร็วสูง เมื่อโมดูลความถี่วิทยุทำงานในโหมดชอคเบร์ส สามารถเพิ่มการเข้าถึงระดับความเร็วในการส่งข้อมูลได้สูง (1 Mbps) โดยใช้ย่านความถี่ 2.4 GHz และต้องใช้ไมโครคอนโทรลเลอร์ความเร็วสูงในการประมวลผลโดยการจัดกระบวนการประมวลผลให้เหมาะสมกับโปรโตคอลบนชิปจะทำให้ได้รับประโยชน์จากโมดูลความถี่วิทยุที่ตามมาดังนี้

- ประหยัดกระแส
- ระบบมีราคาต่ำ(เนื่องจากไมโครคอนโทรลเลอร์มีราคาถูก)
- ลดการชนกันของข้อมูลเมื่อใช้เวลาในการส่งระยะสั้นๆ

รายละเอียดทางไฟฟ้า ตามตารางที่ 2.7 ของตัว รับ/ส่ง สัญญาณ TRW 2.4 GHz ในโหมดชอคเบิร์ส

ตารางที่ 2.7 รายละเอียดทางไฟฟ้า

ชื่อ	ความหมาย	ค่าต่ำสุด	ค่าสูงสุด	หน่วย
	เงื่อนไข			
VDO	แรงดัน	1.9	3.6	V
TEMP	อุณหภูมิ	-40	85	C
	อินพุตดิจิตอล			
V_{IH}	ระดับแรงดันอินพุต HIGHT	VDD-0.3	VDD	V
V_{IL}	ระดับแรงดันอินพุต LOW	Vss	0.3	V
	เอาต์พุตดิจิตอล			
V_{OH}	ระดับแรงดันเอาต์พุต HIGHT (IOH=0.5mA)	VDD-0.3	VDD	V
V_{OL}	ระดับแรงดันเอาต์พุตLOW (IOH=0.5mA)	Vss	0.3	V
	เงื่อนไขต่างๆที่เกี่ยวกับความถี่			
Fop	ช่วงความถี่วิทยุ	2400	2523	MHz
R_{GFSK}	อัตราการส่งข้อมูลในโหมด	>0	1000	Kbps
	กระแสในการใช้งานที่อัตราส่งต่างกันของ 1 ช่องสัญญาณ			
I_{VDD}	250 kbps	18		mA
I_{VDD}	1000kbps	19		mA

หลักการทำงานในโหมดชอคเบิร์ส

เมื่อทำการกำหนดค่าให้โมดูลทำงานในโหมดชอคเบิร์สแล้ว การทำงานของโมดูลในการรับ/ส่งข้อมูล มีหลักการทำงานดังนี้

การส่งข้อมูลในโหมดชอคเบิร์ส

โดยการเชื่อมต่อไมโครคอนโทรลเลอร์กับขา CE,CLK1,DATA ของตัวโมดูล

- เมื่อไมโครคอนโทรลเลอร์ต้องการส่งข้อมูลให้กับโมดูลต้องทำการเซตขา CE ให้อยู่ในสถานะ "1" เพื่อกระตุ้นให้โมดูลทำการประมวลผลข้อมูล
- ทำการเซตไมโครคอนโทรลเลอร์ให้อยู่ในสถานะ "0" เพื่อกระตุ้นให้โมดูลทำการส่งข้อมูล

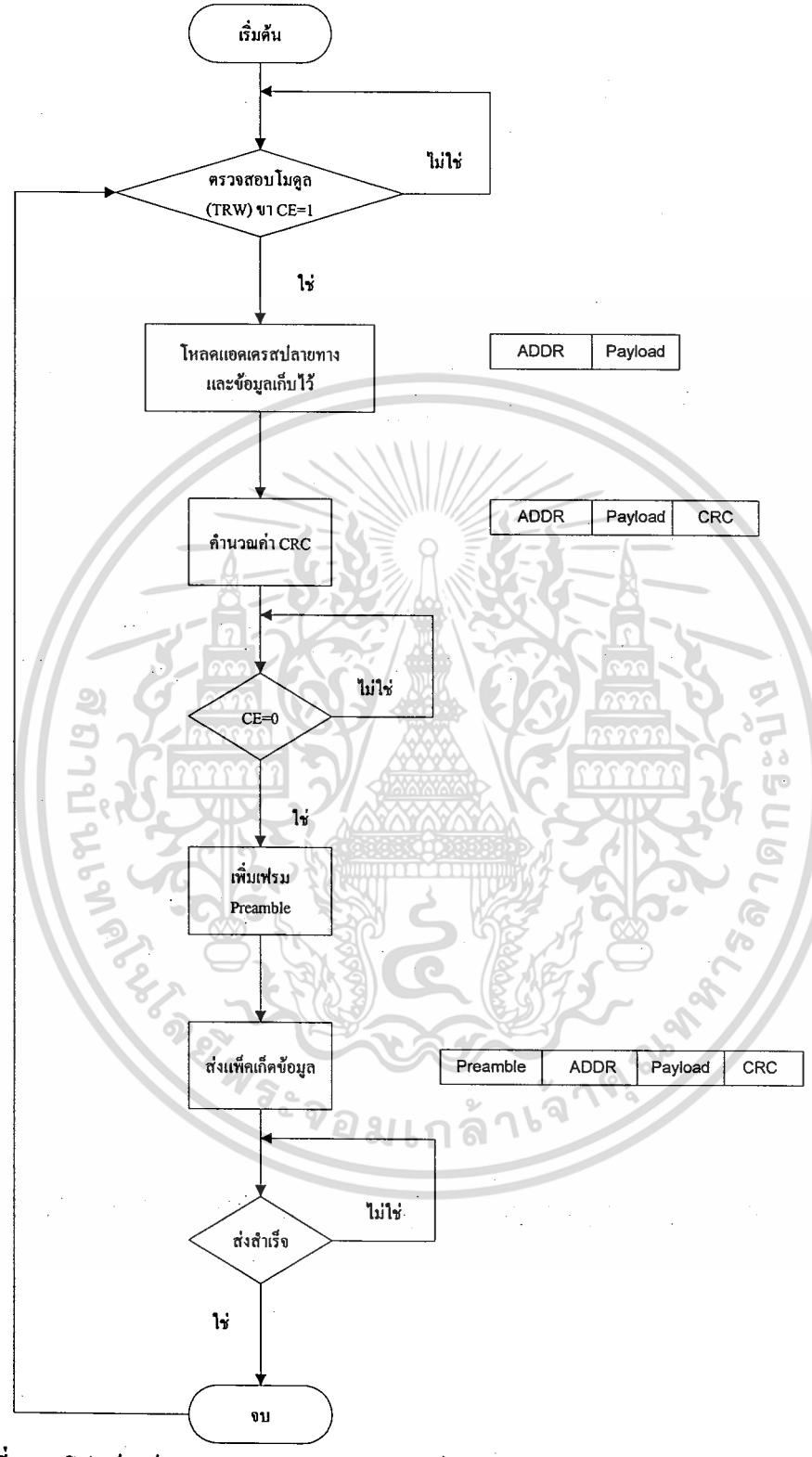
การรับข้อมูลในโหมดชอคเบิร์ส

โดยไมโครคอนโทรลเลอร์ทำการเชื่อต่อกับขา CE,CLK1,DR1 และ DATA (กรณีที่ใช้ช่องสัญญาณเพียงช่องเดียว)

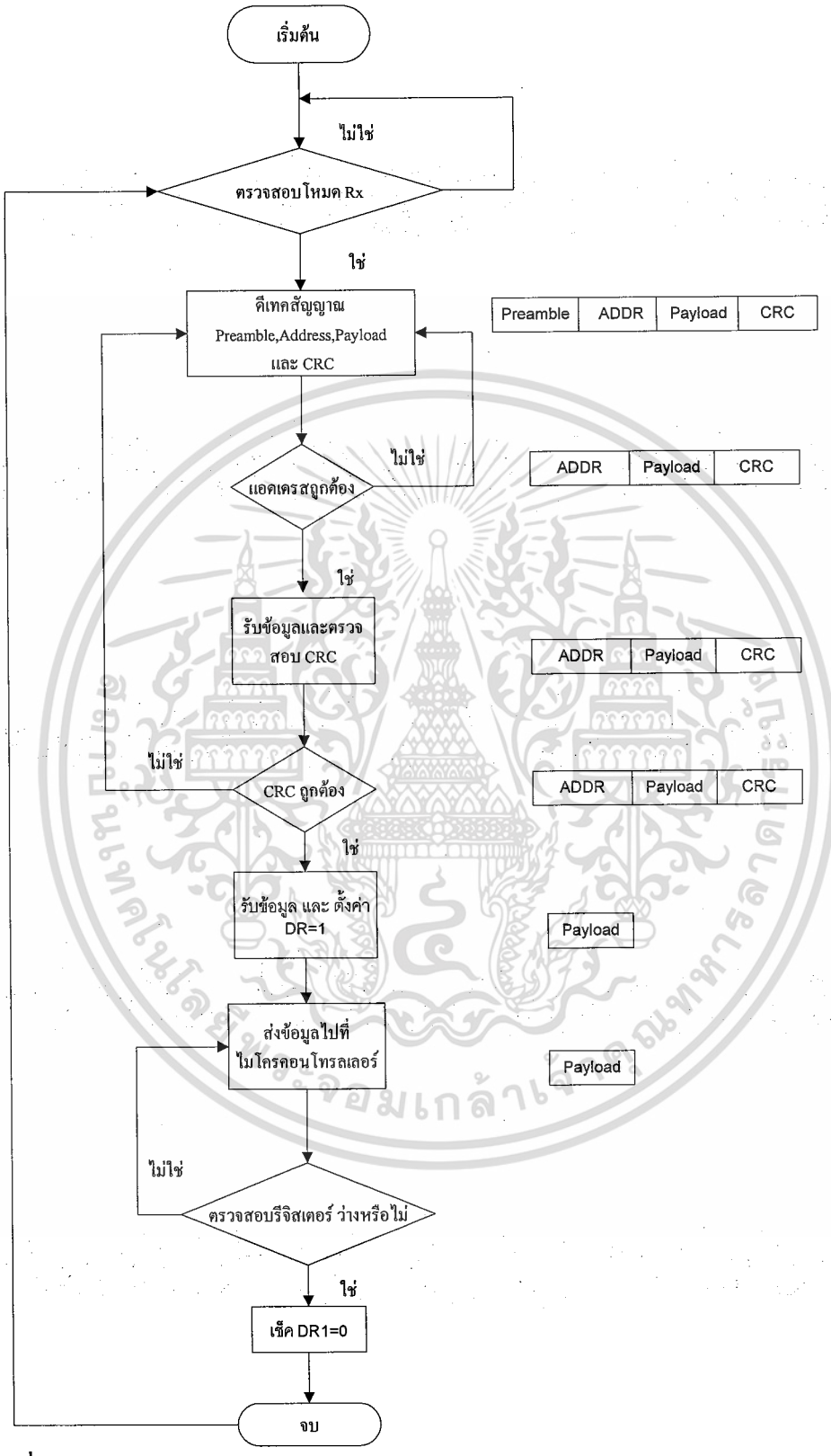
- เมื่อแพคเกตของข้อมูลที่ส่งมาถึงมีแอดเดรสที่ถูกต้องและขนาดของข้อมูลที่เข้ามาครบถ้วนตัวโมดูลทำการเซตค่าให้ขา CE อยู่ในสถานะ "1"
- เมื่อข้อมูลที่รับเข้ามาถูกต้อง(แอดเดรสและCRSถูกต้อง) โมดูลทำการย้าย Preamble , Address และ CRC โดยแจ้งไปยังไมโครคอนโทรลเลอร์ให้ทำการเซตค่าให้ขา DR1 อยู่ในสถานะ "1" และเซตค่าให้ขา CE อยู่ในสถานะ "0" เพื่อบอกว่าขณะนี้ทำการรับข้อมูลอยู่
- ไมโครคอนโทรลเลอร์ทำการเซตค่าต่างๆเพื่อให้รับข้อมูลได้เหมาะสมและเมื่อทำการรับข้อมูลเสร็จเรียบร้อยทำการเซตค่าให้ขา DR1 ให้อยู่ในสถานะ "0" เพื่อเตรียมพร้อมที่จะรับข้อมูลที่เข้ามาใหม่

การส่งสัญญาณแบบชอคเบิร์ส

อธิบายการทำงานของโฟลว์ชาร์ตตัวส่งหลังจากทำการกำหนดรูปแบบ(Configuration)ให้เป็นโหมดส่งสัญญาณแล้ว จากนั้นหลักการทำงานของโมดูลเป็นดังโฟลว์ชาร์ตรูปที่ 2.15 โดยโมดูลตรวจสอบว่าขา CE เป็น "1" หรือไม่ ถ้าใช่ โมดูลรับแอดเดรสปลายทางและข้อมูล จากนั้นทำการคำนวณหา CRC เพื่อเป็นเฟรมที่ใช้ในการตรวจสอบข้อมูลตรวจสอบขา CE ถ้าเป็น "0" แสดงว่าเฟรมข้อมูลและ CRC พร้อมที่จะส่งแล้ว จากนั้นโมดูลทำการเพิ่มเฟรม Preamble เข้าไปแล้วทำการส่งแพคเกตของข้อมูลทั้งหมดและเมื่อส่งข้อมูลเสร็จเรียบร้อยแล้วทำการวนกลับไปยังจุดเริ่มต้นของโฟลว์ชาร์ตใหม่อีกครั้ง



รูปที่ 2.15 โฟลว์ชาร์ตการส่งสัญญาณแบบชอคเบิร์สของตัวส่ง/รับสัญญาณ TRW 2.4GHz



รูปที่ 2.16 โฟลว์ชาร์ตการรับสัญญาณแบบชอคเบิร์สของตัวส่ง/รับสัญญาณ TRW 2.4GHz

อธิบายการทำงานของโฟลว์ชาร์ต ตัวรับ

จากรูปที่ 2.16 หลังจากทำการกำหนดรูปแบบให้เป็นโหมดรับสัญญาณแล้ว โมดูลทำการตรวจจับสัญญาณจากโมดูลตัวส่ง หลังจากตรวจจับสัญญาณพบแล้วทำการตรวจสอบแอดเดรสที่ส่งมาตรงกับแอดเดรสประจำตัวของโมดูลตัวรับที่ทำการตั้งไว้หรือไม่ถ้าใช่ทำการรับข้อมูลกลับ CRC จากนั้นทำการตรวจสอบข้อมูลว่าถูกต้องหรือไม่ ถ้าถูกต้องทำการเซตขา DR1 ให้เป็นสถานะ "1" และทำการโหลดข้อมูลทั้งหมดออกจากรีจิสเตอร์ของโมดูลโดยตรวจสอบว่ารีจิสเตอร์ของโมดูลว่างหรือไม่ เมื่อรีจิสเตอร์ว่างแล้วทำการเซตขา DR1 ให้เป็นสถานะ "0" และวนกลับไปยังจุดเริ่มต้นของโฟลว์ชาร์ต

การทำงานของตัวส่ง/รับสัญญาณ TRW2.4GHz

การเชื่อมต่อกับตัวส่ง/รับสัญญาณ TRW 2.4GHz นั้นทำบนเส้นสัญญาณ 3 เส้นที่เชื่อมต่อไปยังรีจิสเตอร์ของรับ/ส่งสัญญาณ TRW 2.4GHz ค่าการตั้งค่าให้เป็นตัวส่งสัญญาณอาจยาวถึง 18 ไบต์ใน การส่งแบบชอคเบิร์ส ทั้งนี้ต้องทำการตั้งค่าการทำงานของอุปกรณ์ก่อนที่ทำการส่งสัญญาณซึ่งประกอบด้วยขั้นตอนต่อไปนี้

การทำงานของตัวส่งสัญญาณ TRW 2.4GHz

1. การเข้าสู่สแตนด์บายโหมดเพื่อเตรียมอุปกรณ์ให้พร้อมที่จะใช้งาน ทำได้โดยการกำหนดให้ขา DATA1 เป็น "0" ขา CS1 เป็น "0" เป็น "0" ขา CE1 เป็น "0" และขา CLK1 เป็น "0"
2. ส่งค่าที่ใช้ตั้งค่าการทำงานของอุปกรณ์(รายละเอียดจะกล่าวในหัวข้อถัดไป)
3. ส่งข้อมูลที่ต้องการใช้ในการสื่อสารที่ประกอบไปด้วยแอดเดรสของตัวรับสัญญาณและข้อมูลที่ต้องการใช้ในการสื่อสาร ไปยังตัวส่งสัญญาณ TRW 2.4GHz

การทำงานของตัวรับสัญญาณ TRW 2.4 GHz

1. การเข้าสู่สแตนด์บายโหมดเพื่อเตรียมอุปกรณ์ให้พร้อมที่จะใช้งาน ทำได้โดยการกำหนดให้ขา DATA1 เป็น "0" ขา CS1 เป็น "0" เป็น "0" ขา CE1 เป็น "0" และขา CLK1 เป็น "0"
2. ส่งค่าที่ใช้ตั้งค่าการทำงานของอุปกรณ์(รายละเอียดจะกล่าวในหัวข้อถัดไป)
3. ส่งสัญญาณข้อมูลต้องการใช้ในการสื่อสารที่รับได้ออกมาเพื่อนำไปใช้งานต่อไป

2.5.4 การตั้งค่าการใช้งาน

ตารางที่ 2.8 การตั้งค่าการใช้งาน

	ตำแหน่งบิต	จำนวนบิต	ชื่อ	ฟังชั่นการทำงาน
การตั้งค่าการส่งแบบชอคเบิร์ส	143:120	24	TEST	สงวนไว้เพื่อการทดสอบ
	119:112	8	DATA2_W	ความยาวข้อมูลในส่วนเปย์โหลด ช่องสัญญาณที่ 2
	111:104	8	DATA1_W	ความยาวข้อมูลในส่วนเปย์โหลด ช่องสัญญาณที่ 1
	103:64	40	ADDR2	แอดเดรสของตัวรับ ช่องสัญญาณที่ 2
	63:24	40	ADDR1	แอดเดรสของตัวรับ ช่องสัญญาณที่ 1
	23:18	6	ADDR_W	จำนวนบิตของแอดเดรส
	17	1	CRC_L	CRC แบบ 8 หรือ 16
	16	1	CRC_EN	เลือกการทำงานตรวจสอบ CRC
การเชื่อมต่อกับอุปกรณ์ทั่วไป	15	1	RX2_EN	เลือกรับแบบ 2 ช่องสัญญาณ
	14	1	CM	โหมดการติดต่อสื่อสาร
	13	1	RFDR_SB	ความเร็วของข้อมูลที่ส่ง
	12:10	3	XO_F	ความถี่คริสตอล
	9:8	2	RF_PWR	กำลังส่งของสัญญาณวิทยุ
	7:1	7	RF_CH#	ช่องความถี่
	0	1	RXEN	ส่ง หรือ รับ สัญญาณ

ค่าการทำงานบิตแรกที่เลื่อนเข้าไปยังตัวส่ง/รับสัญญาณ TRW 2.4GHz เป็นบิตที่มีความสำคัญสูงสุด โดยเลื่อนเข้าที่ขอบขาขึ้นของสัญญาณนาฬิกา การส่งค่าการทำงานครั้งถัดไปจะทำได้หลังขอบขาลงของสัญญาณขา CS

รายละเอียดค่าการใช้งานตัวส่ง/รับสัญญาณ TRW 2.4GHz

รายละเอียดด้านล่างนี้เป็นเป็นการอธิบายฟังก์ชั่นของทั้ง 144 บิต (บิตที่สำคัญสูงสุด คือบิตที่ 143) ที่ใช้ตั้งค่าการทำงาน

การเชื่อมต่อกับอุปกรณ์ทั่วไป กำหนดที่ บิตที่ 0 ถึง บิตที่ 15

การส่งสัญญาณแบบชอคเบิร์ส กำหนดที่ บิตที่ 0 ถึง บิตที่ 119

การทดสอบ กำหนดที่ บิตที่ 120 ถึง บิตที่ 143

2.6 การตั้งค่าเพื่อทำการส่งแบบชอดเบิร์ส

เริ่มตั้งแต่บิตที่ 16 ถึงบิตที่ 119 ประกอบด้วยส่วนต่างๆของการกำหนดค่าให้รีจิสเตอร์ในการทำงานเพื่อส่งแบบชอดเบิร์สหลังจากที่ป้อนไฟเลี้ยง การตั้งค่าเพื่อส่งแบบชอดเบิร์สจะทำเพียงครั้งเดียวค่าที่ตั้งไว้นั้นจะอยู่ตลอดตราบเท่าที่ป้อนไฟเลี้ยงให้กับอุปกรณ์

การทำงานของเฟสลอคลูป

PLL_CTRL : ใช้ควบคุมการตั้งค่าการทำงานของเฟสลอคลูป สำหรับส่วนทดสอบกำหนดในบิตตำแหน่งที่ 120 ถึง 121

ตารางที่ 2.9 การตั้งค่าการใช้งานเฟดลอคลูป

PLL_CTRL		
บิตที่ 121	บิตที่ 120	การทำงานของเฟสลอคลูป
0	0	ใช้ที่ตัวส่ง/ไม่ใช้ที่ตัวรับ
0	1	ใช้ที่ตัวส่ง/ใช้ที่ตัวรับ
1	0	ไม่ใช้ที่ตัวส่ง/ไม่ใช้ที่ตัวรับ
1	1	ไม่ใช้ที่ตัวส่ง/ใช้ที่ตัวรับ

2.6.1 ความยาวของข้อมูลส่วนเปย์โหลด

DATA2_W : เป็นส่วนที่ใช้ในการกำหนดความยาวของข้อมูลส่วนเปย์โหลดในแพคเกตของช่องสัญญาณที่ 2

บิตตำแหน่งที่ 119 ถึง 112

DATA1_W : เป็นส่วนที่ใช้ในการกำหนดความยาวของข้อมูลส่วนเปย์โหลดในแพคเกตของช่องสัญญาณที่ 1

บิตตำแหน่งที่ 111 ถึง 104

ทั้งนี้จำนวนบิตทั้งหมดในแพคเกตต้องไม่เกิน 256 บิต

จำนวนบิตข้อความสูงสุดในเปย์โหลดสามารถหาได้จาก

ความยาวข้อมูล (บิต) = 256 – ความยาวแอดเดรส – ความยาว CRC

โดยที่ความยาวแอดเดรส เป็นความยาวแอดเดรสของตัวรับสัญญาณที่กำหนดที่บิตที่ 18 ถึง 23

ความยาว CRC ยาว 8 หรือ 16 บิต

ตารางที่ 2.10 การตั้งค่าจำนวนบิตในเปย์โหลด

DATA2_W							
บิตที่ 119	บิตที่ 118	บิตที่ 117	บิตที่ 116	บิตที่ 115	บิตที่ 114	บิตที่ 113	บิตที่ 112

DATA1_W							
บิตที่ 111	บิตที่ 110	บิตที่ 109	บิตที่ 108	บิตที่ 107	บิตที่ 106	บิตที่ 105	บิตที่ 104

2.6.2 แอดเดรสของตัวรับสัญญาณ

ใช้ในการกำหนดแอดเดรสของตัวรับสัญญาณ

ตารางที่ 2.11 การตั้งค่าแอดเดรสของตัวรับสัญญาณ

ADDR2											
103	102	101	…	71	70	69	68	67	66	65	64

ADDR1											
63	62	61	…	31	30	29	28	27	26	25	24

บิตที่ 64 ถึง 103 : ADDR2

แอดเดรสของตัวรับสัญญาณช่องสัญญาณที่ 2 สามารถมีความยาวได้สูงสุดถึง 40 บิต

บิตที่ 24 ถึง 63 : ADDR1

แอดเดรสของตัวรับสัญญาณช่องสัญญาณที่ 1 สามารถมีความยาวได้สูงสุดถึง 40 บิต

2.6.3 ความยาวแอดเดรสของตัวรับสัญญาณ

ใช้ในการกำหนดความยาวแอดเดรสของตัวรับสัญญาณและรูปแบบการทำงานของ CRC

ตารางที่ 2.12 การตั้งค่าความยาวแอดเดรสของตัวรับสัญญาณ

ADDR_W						CRC_L	CRC_EN
บิตที่ 23	บิตที่ 22	บิตที่ 21	บิตที่ 20	บิตที่ 19	บิตที่ 18	บิตที่ 17	บิตที่ 16

บิตที่ 18 ถึง 23 :ADDR_W

กำหนดความยาวแอดเดรสของตัวรับสัญญาณรองรับสูงสุดได้ 40 บิต

บิตที่ 17 : CRC_L

กำหนดความยาวของ CRC ที่ตัวส่ง/รับสัญญาณ TRW 2.4GHz ต้องทำการคำนวณหาโดยสามารถกำหนดความยาวได้ดังนี้

ลอจิก 0 : CRC ยาว 8 บิต

ลอจิก 1 : CRC ยาว 16 บิต

บิตที่ 16 : CRC_EN

เลือกใช้/ไม่ใช้งาน CRC โดยสามารถกำหนดรูปแบบได้ดังนี้

ลอจิก 0 : ไม่ใช้งาน CRC

ลอจิก 1 : ใช้งาน CRC

2.6.4 การติดต่อกับอุปกรณ์โดยทั่วไป

การตั้งค่าสำหรับการติดต่อกับอุปกรณ์โดยทั่วไป

ตารางที่ 2.13 การตั้งค่าสำหรับการติดต่อกับอุปกรณ์โดยทั่วไป

RX2_EN	CM	RFDR_SB	XO_F			RF_PWR	
บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8

บิตที่ 15 : RX2_EN

เป็นบิตที่ใช้ในการกำหนดจำนวนช่องสัญญาณที่ใช้ในการสื่อสาร โดยมีรูปแบบการกำหนดดังนี้

ลอจิก 0 เป็นการกำหนดให้อุปกรณ์ภาครับรับ 1 ช่องสัญญาณ

ลอจิก 1 เป็นการกำหนดให้อุปกรณ์ภาครับรับ 2 ช่องสัญญาณ

โดยที่ การกำหนดให้ตัวรับสามารถรับได้ทั้ง 2 ช่องสัญญาณ ความถี่ของช่องสัญญาณแรกกำหนดในบิตที่ 1 ถึง 7 โดยที่ช่องสัญญาณที่ 2 อยู่เหนือช่องสัญญาณแรกไปอีก 8 ช่องสัญญาณ

บิตที่ 14 : CM (Communication Mode)

ใช้ในการเลือกรูปแบบการสื่อสาร

ลอจิก 1 เป็นการเลือกให้ทำการสื่อสารแบบซอคเบิร์ส

บิตที่ 13 : RFDR_SB

เป็นการกำหนดความเร็วของข้อมูลที่จะใช้ในการส่งสัญญาณ

ลอจิก 0 เป็นการกำหนดส่งข้อมูลด้วยความเร็ว 250 Kbps

ลอจิก 1 เป็นการกำหนดส่งข้อมูลด้วยความเร็ว 1 Mbps

โดยที่การใช้งานที่ความเร็ว 250Kbps ทำให้ตัวรับสัญญาณมีเซนซิติวิตี้ดีขึ้นอีก 10 dB เมื่อเทียบกับการส่งสัญญาณด้วยความเร็ว 1Mbps ทั้งนี้การส่งด้วยความเร็ว 1Mbps จำเป็นต้องมีการต่อคริสตอล 16 MHz เพิ่มเข้าไปในวงจร

บิตที่ 10 ถึง 12

ตารางที่ 2.14 การตั้งค่าสำหรับบิตที่ 10 ถึง 12

บิตที่ 12	บิตที่ 11	บิตที่ 10
0	1	1

บิตที่ 8 ถึง 9 : RF_PWR

เป็นบิตที่ใช้ในการกำหนดกำลังส่งของสัญญาณ โดยสามารถกำหนดได้ตามาตารางดังนี้

ตารางที่ 2.15 การตั้งค่าการกำหนดกำลังส่งของสัญญาณ

กำลังส่งของสัญญาณ		
บิตที่ 9	บิตที่ 8	กำลังส่ง (dBm)
0	0	-20
0	1	-10
1	0	-5
1	1	0

2.6.5 RF channel & direction

ตารางที่ 2.16 การตั้งค่าช่องสัญญาณ และการเลือกให้รับหรือส่งสัญญาณ

RF_CH#							RXEN
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0

บิตที่ 1 ถึง 7 เป็นบิตที่ใช้ในการเลือกตำแหน่งช่องสัญญาณที่จะให้อุปกรณ์ใช้ในการส่งข้อมูลโดยมีวิธีการกำหนดดังนี้

ความถี่ที่ใช้ในการส่งสัญญาณ = 2400MHz + (ตำแหน่งช่องสัญญาณ x 1 MHz)

โดยความถี่ที่ใช้งานอยู่ในช่วง 2400 MHz ถึง 2527 MHz

การกำหนดความถี่ให้ช่องสัญญาณข้อมูลที่ 1 (สัญญาณเอาต์พุตที่ขา 8)

ความถี่ที่ใช้ในการส่งสัญญาณ = 2400MHz + (ตำแหน่งช่องสัญญาณ x 1MHz)

โดยความถี่ที่ใช้งานจะอยู่ในช่วง2400MHz ถึง 2524MHz

การกำหนดความถี่ให้ช่องสัญญาณข้อมูลที่ 2 (สัญญาณเอาต์พุตที่ขา 4)

ความถี่ที่ใช้ในการส่งสัญญาณ = 2400MHz + (ตำแหน่งช่องสัญญาณ x 1 MHz) + 8 MHz

โดยความถี่ที่ใช้งานจะอยู่ในช่วง 2400 MHz ถึง 2524MHz

บิตที่ 0 เป็นบิตที่ใช้ในการกำหนดให้อุปกรณ์ทำหน้าที่เป็นตัวรับหรือตัวส่งสัญญาณ โดยการตั้งค่าการทำงานดังนี้

ลอจิก 0 เป็นการกำหนดให้อุปกรณ์เป็นตัวส่งสัญญาณ

ลอจิก 1 เป็นการกำหนดให้อุปกรณ์เป็นตัวรับสัญญาณ

รายละเอียดรูปแบบแพคเกต

แพคเกตที่ใช้ในการสื่อสารระหว่างตัวส่ง/รับสัญญาณ TRW 2.4GHz ถูกแบ่งออกเป็น 4 ส่วนดังนี้

ตารางที่ 2.16 ไดอะแกรมของแพคเกตข้อมูล

PRE-AMBLE	ADDRESS	PAYLOAD	CRC

1. **Pre-AMBLE** เป็นส่วนที่ตัวส่ง/รับสัญญาณ TRW 2.4GHz ทำการเพิ่มเข้ามาเพื่อใช้ในการส่งสัญญาณ แต่ถ้าทำการส่งแบบชอคเบิร์ส ตัวส่ง/รับสัญญาณ TRW 2.4GHz จะเพิ่มเข้ามาให้อัตโนมัติ
2. **ADDRESS** ส่วนแสดงแอดเดรสของตัวรับสัญญาณ มีความยาวได้ตั้งแต่ 8 ถึง 40 บิต
3. **PAYLOAD** ส่วนของข้อมูลที่ใช้ในการติดต่อสื่อสาร
4. **CRC** ใช้ในการส่งแบบชอคเบิร์ส โดยมีความยาว 8 ถึง 16 บิต

ทั้งนี้สัญญาณเอาต์พุตจากตัวส่ง/รับสัญญาณ TRW 2.4GHz จะเหลือแค่ส่วนของข้อมูลในเปย์โหลด

2.7 การเชื่อมต่อกับเครื่องคอมพิวเตอร์

หากเราต้องการที่จะเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์อื่นๆ หรือคอมพิวเตอร์ด้วยกันมี 2 ทางเลือกนั่นคือ การรับส่งข้อมูลแบบอนุกรมและแบบขนาน

2.7.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นเราสามารถแบ่งออกได้เป็น 2 แบบคือการสื่อสารแบบซิงโครนัสและอะซิงโครนัส

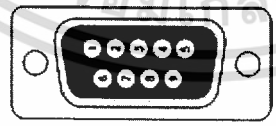
การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลอะซิงโครนัส คือ การรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิกามาร่วมด้วยเหมือนกันกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูลหรือบอดเรต (Baud rate) มีหน่วยเป็นบิตต่อวินาที (Bit per second : bps)

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและการส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูลบอดเรต มาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232

2.7.2 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยกำหนดความยาวของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -15 โวลต์ แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +15 โวลต์ แสดงว่าเป็นช่องว่าง (Space)

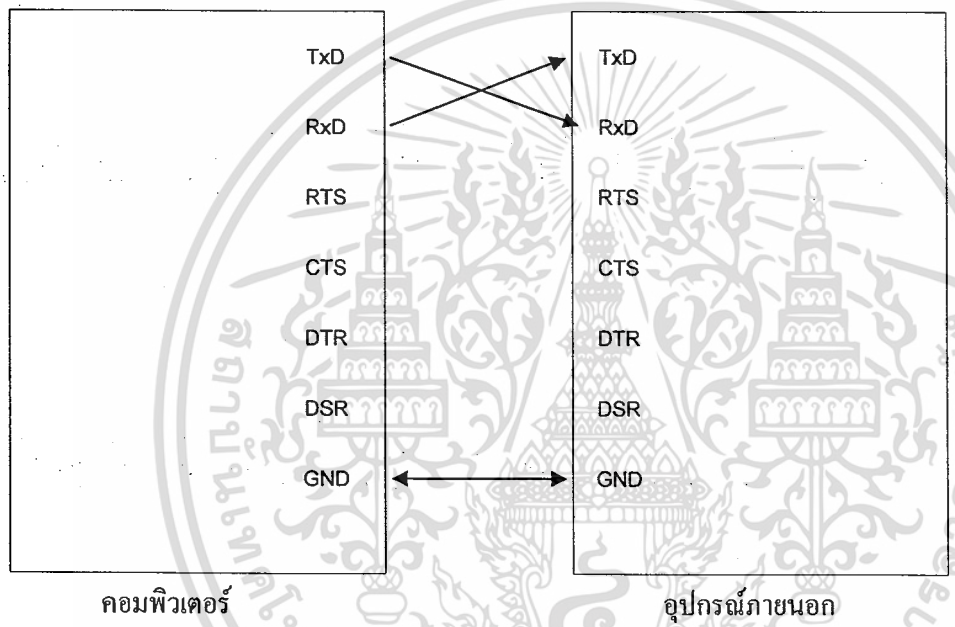


รูปที่ 2.17 การจัดขาของคอนเนคเตอร์พอร์ตอนุกรมมาตรฐาน RS-232 แบบ DB-9

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกดังในรูปที่ 2.18 ลูกศรในรูปแสดงถึงทิศทางของข้อมูลเป็นการเชื่อมต่อในลักษณะที่ใช้สายสัญญาณเพียง 3 สายโดยสายเส้นหนึ่งใช้สำหรับการส่งข้อมูล สายอีกเส้นใช้สำหรับการรับข้อมูล และสายเส้นสุดท้ายจะใช้สำหรับกราวนด์

รายละเอียดของพอร์ตอนุกรม RS-232 ที่ใช้งานมีดังนี้

- Received Data : RD ขานี้ใช้สำหรับเพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บในรีจิสเตอร์บัฟเฟอร์
- Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์โดยที่จะนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป
- Signal Ground ขากราวนด์ของระบบ



รูปที่ 2.18 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232 โดยใช้สายสัญญาณ 3 เส้น

2.7.3 ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม RS-232

มาตรฐานการสื่อสารของข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่าที่ลอจิก "0" จะมีระดับสัญญาณ +3 ถึง +15 โวลต์ส่วนลอจิก "1" จะมีระดับสัญญาณ -3 ถึง -15 โวลต์ระดับสัญญาณนี้ทำให้ไม่สามารถนำขาเอาต์พุตใดๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรงจะต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันเสียก่อน นั่นก็คือวงจรขับแบบมาตรฐาน RS-232 นั้นด้านภาคส่งต้องสามารถเปลี่ยนระดับสัญญาณลอจิกให้เป็นระดับแรงดันตามที่ได้กำหนดไว้และในส่วนของภาครับเองนั้นก็ต้องสามารถตรวจจับระดับแรงดันที่รับเข้ามาแล้วเปลี่ยนกลับให้เป็นสัญญาณลอจิกได้อย่างถูกต้องด้วยเช่นกัน โดยปกติจะใช้ไอซีจำพวก RS-232 Transceiver ที่นิยมมากคือ MAX232 หรือ ICL232 ไอซีกลุ่มนี้จะทำหน้าที่แปลงระดับทีทีแอลให้เป็นสัญญาณในระดับแรงดันของ RS-232 และยังทำหน้าที่แปลงสัญญาณ ในระดับแรงดันของ

RS-232 ให้กลับมาอยู่ในระดับทีทีแอล โดยลอจิก “0” ซึ่งเดิมมีระดับสัญญาณ +3 ถึง +15 โวลต์จะถูกแปลงเป็น 0 โวลต์ส่วนลอจิก “1”ซึ่งมีระดับสัญญาณ -3 ถึง -15 โวลต์ จะแปลงเป็น +5 โวลต์ ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอลได้

2.7.4 การสื่อสารอนุกรมยูเออาร์ที (UART : Universal Asynchronous Receiver-Transmitter)

ยูเออาร์ที เป็นอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส สำหรับการสื่อสารอนุกรมบนเครื่องคอมพิวเตอร์แล้ว ยูเออาร์ทีถือว่าเป็นหัวใจสำคัญของการสื่อสารแบบอนุกรม

หน้าที่หลักของยูเออาร์ทีคือทำหน้าที่ในการแปลงข้อมูลรูปแบบขนาดจากคอมพิวเตอร์เป็นรูป แบบอนุกรมแบบอะซิงโครนัสแล้วส่งออกไปและทำหน้าที่ในการแปลงสัญญาณอนุกรมแบบอะซิงโครนัส ที่ป้อนเข้ามายังยูเออาร์ทีให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์อีกทั้งยังแจ้งข้อมูลอื่นๆให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (Baud rate), รูปแบบการส่งข้อมูลความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้,เฟรมข้อมูล,โอเวอร์รัน)

2.8 การประมวลผลในคอมพิวเตอร์

การประมวลผลในคอมพิวเตอร์จะใช้โปรแกรมที่เขียนขึ้นด้วยภาษาซีชาร์ป

ภาษาซีชาร์ป (C# Programming Language) เป็นภาษาโปรแกรมเชิงวัตถุทำงานบนดอทเน็ตเฟรมเวิร์กพัฒนาโดยบริษัทไมโครซอฟท์ โดยมีรากฐานมาจากภาษาซีพลัสพลัสและภาษาอื่นๆ(โดยเฉพาะภาษาเดลไฟและจาวา) โดยปัจจุบันภาษาซีชาร์ปเป็นภาษามาตรฐานรองรับโดย ECMA และ ISO ภาษาซีชาร์ปซึ่งไมโครซอฟท์นำภาษาซี และซีชาร์ปมาพัฒนาต่อนั้น เป็นภาษาที่ทำให้โปรแกรมรันบนซอฟท์แวร์พื้นฐานของคอมพิวเตอร์ได้ เวลาเขียนโปรแกรมจะง่ายเพราะสามารถใช้รันอุปกรณ์ใดก้อได้ที่มีซอฟท์แวร์พื้นฐานภาษาซีชาร์ปนี้อยู่ภายใต้โครงสร้างเทคโนโลยีดอตเน็ต

ภาษาซีชาร์ปถูกพัฒนาขึ้นโดยเป็นส่วนหนึ่งในการพัฒนาโครงสร้างพื้นฐานของดอตเนตเฟรมเวิร์ค เป็นการนำข้อดีของภาษาต่างๆ(เช่นภาษาเดลไฟ ภาษาซีพลัสพลัส)มาปรับปรุงเพื่อให้มีความเป็นโอโอพี(OOP) อย่างถึงที่สุด ขณะเดียวกันก็ลดความซับซ้อนในโครงสร้างของภาษาลง (เรียบง่ายกว่าภาษาซีพลัสพลัส) และมีเครื่องหมายน้อยลง (เมื่อเทียบกับ Java)

ภาษาเอสคิวแอล (SQL)

เป็นภาษาที่ทำให้ผู้ใช้สามารถทำงานจัดการกับฐานข้อมูลได้ โดยเอสคิวแอลได้รับการออกแบบให้มีการดำเนินการกับข้อมูลแบบโต้ตอบระหว่างผู้ใช้กับคอมพิวเตอร์โดยตรงด้วยการพิมพ์คำสั่งผ่านทางคอมพิวเตอร์และผลลัพธ์ของข้อมูลจากฐานข้อมูลจะปรากฏบนจอภาพทันที

บทที่ 3

การออกแบบและการสร้าง

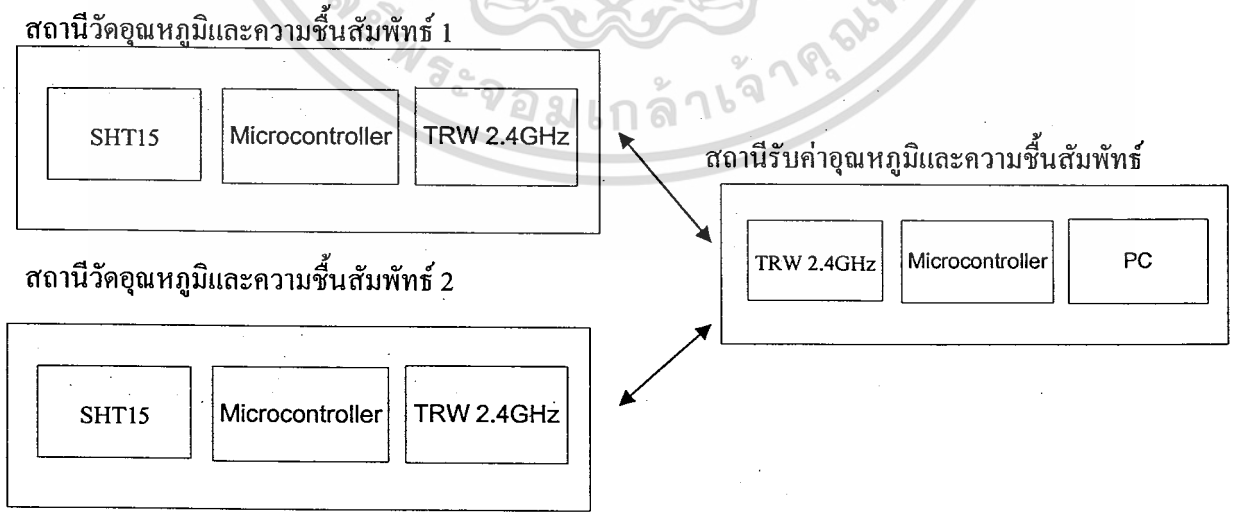
3.1 การออกแบบและการสร้างในส่วนของฮาร์ดแวร์

การออกแบบวงจรฮาร์ดแวร์ของระบบสื่อสารข้อมูลไร้สายประกอบด้วยส่วนต่างๆดังนี้

- ส่วนวัดอุณหภูมิ ใช้เซ็นเซอร์วัดอุณหภูมิ SHT15
- ส่วนส่งสัญญาณแบบไร้สายด้วยตัวส่งสัญญาณ TRW 2.4GHz
- ส่วนรับสัญญาณแบบไร้สายด้วยตัวรับสัญญาณ TRW 2.4GHz
- ส่วนส่งข้อมูลเข้าคอมพิวเตอร์

รูปที่ 3.1 บล็อกไดอะแกรมของระบบติดตามผลและส่งข้อมูลแบบอุณหภูมิและความชื้น

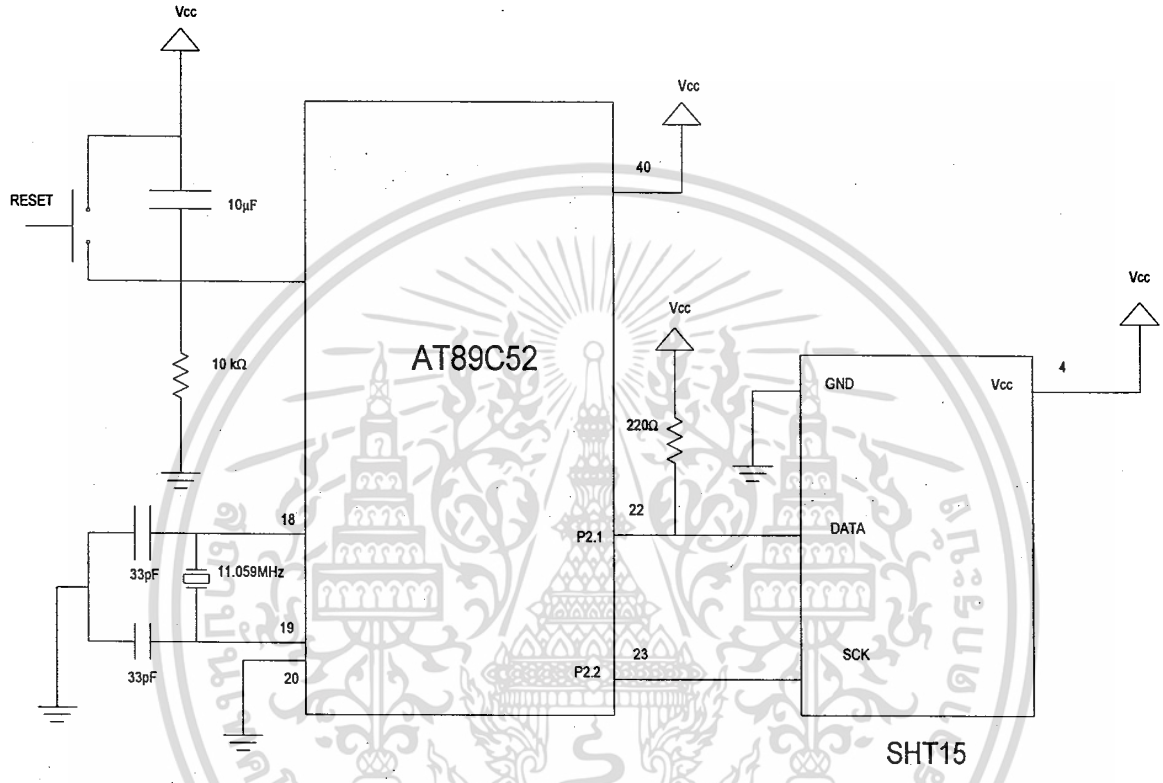
โดยออกแบบให้ระบบติดตามผลอุณหภูมิภายในโรงงานโดยมีสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์ทั้งหมด 2 สถานี และสถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์ 1 สถานี ดังรูปที่ 3.2 ซึ่งสถานีที่ 1 วัดอุณหภูมิและส่งค่ามาให้สถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์ จากนั้นสถานีที่ 2 จึงทำการวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ แล้วจึงส่งมาให้สถานีรับสัญญาณ



รูปที่ 3.2 ชุดอุปกรณ์ในการติดตามผลอุณหภูมิและความชื้นสัมพัทธ์ภายในโรงงาน

3.1.1 วงจรวัดอุณหภูมิ

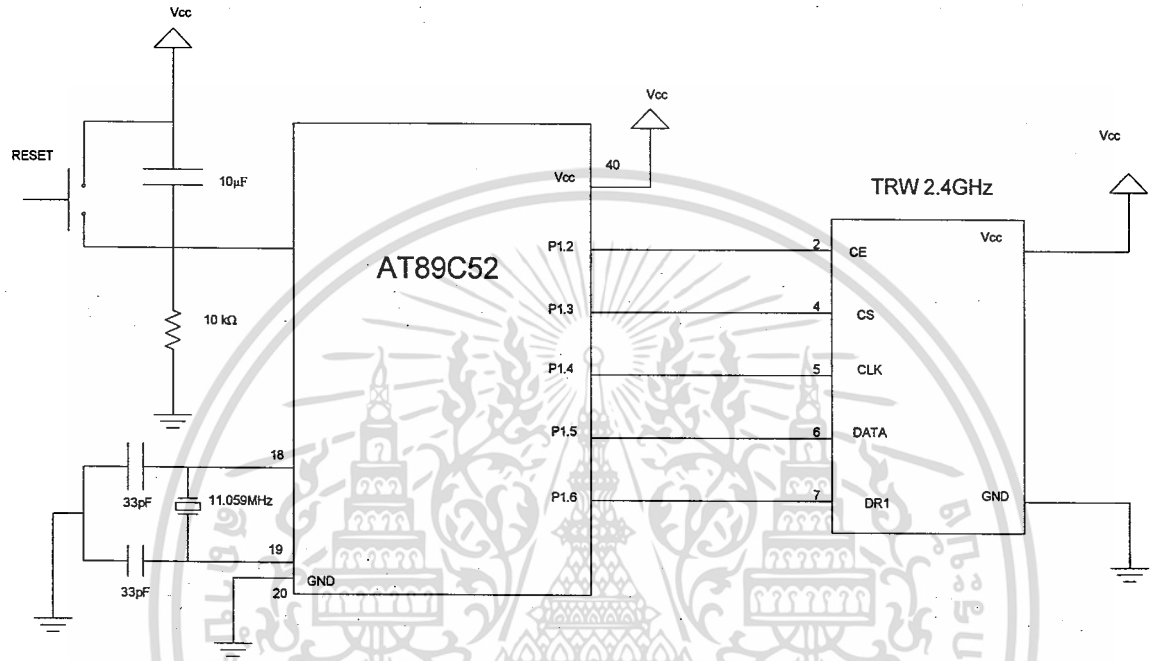
เซ็นเซอร์วัดอุณหภูมิเบอร์ SHT15 มีหน้าที่ในการวัดอุณหภูมิโดยจะทำงานร่วมกับไมโครคอนโทรลเลอร์ ค่าอุณหภูมิที่ได้จะถูกเก็บไว้ในไมโครคอนโทรลเลอร์



รูปที่ 3.3 วงจรวัดอุณหภูมิและความชื้นสัมพัทธ์

3.1.2 วงจรรับ/ส่งสัญญาณด้วย TRW 2.4GHz

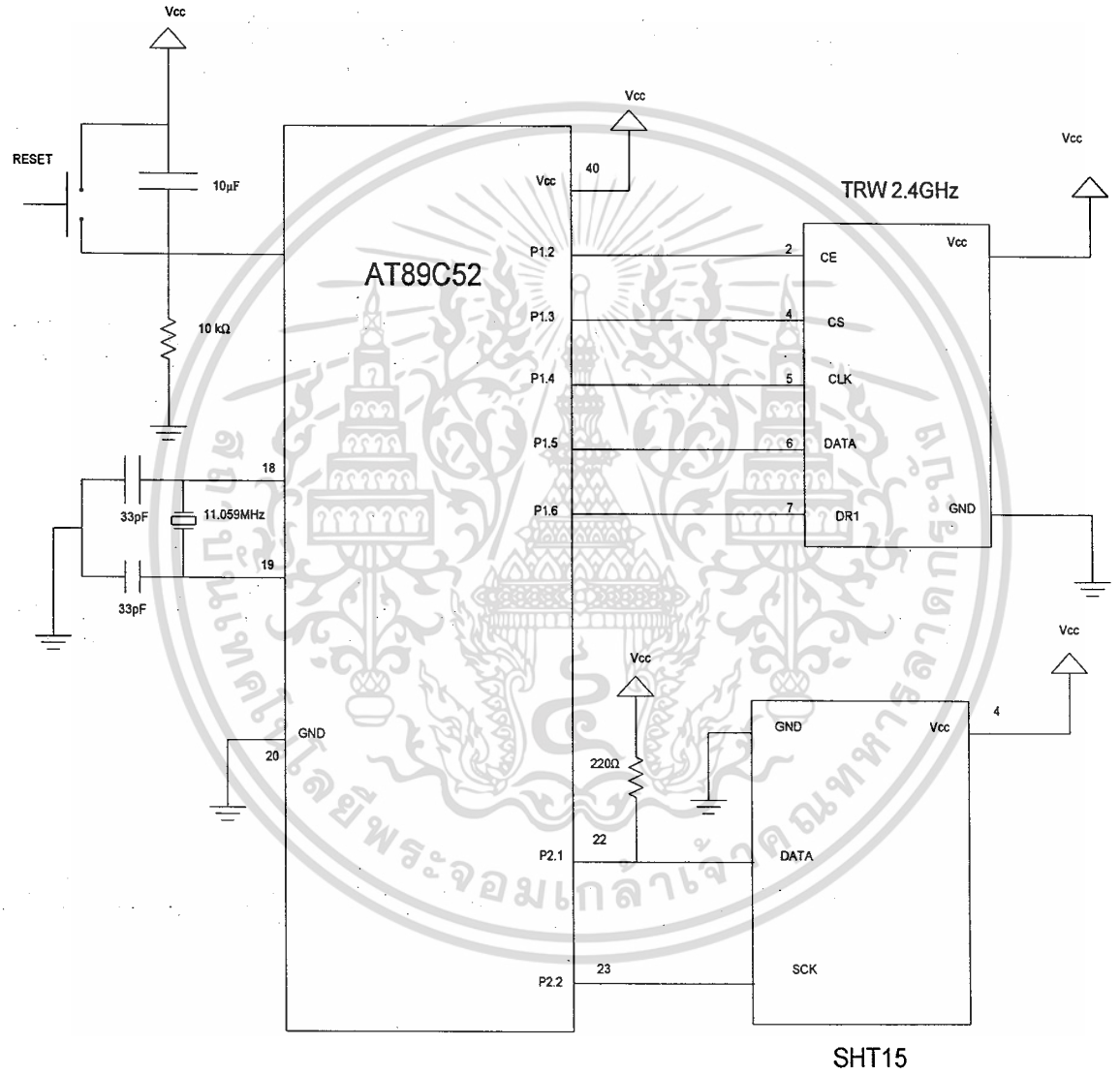
ตัวรับสัญญาณด้วย TRW 2.4GHz มีหน้าที่ในการส่งหรือรับสัญญาณไร้สายผ่านคลื่นวิทยุโดยจะทำงานร่วมกับไมโครคอนโทรลเลอร์



รูปที่ 3.4 วงจรรับ/ส่งสัญญาณด้วย TRW 2.4GHz

3.1.3 วงจรสถานีวัดอุณหภูมิที่ 1 และ สถานีที่ 2

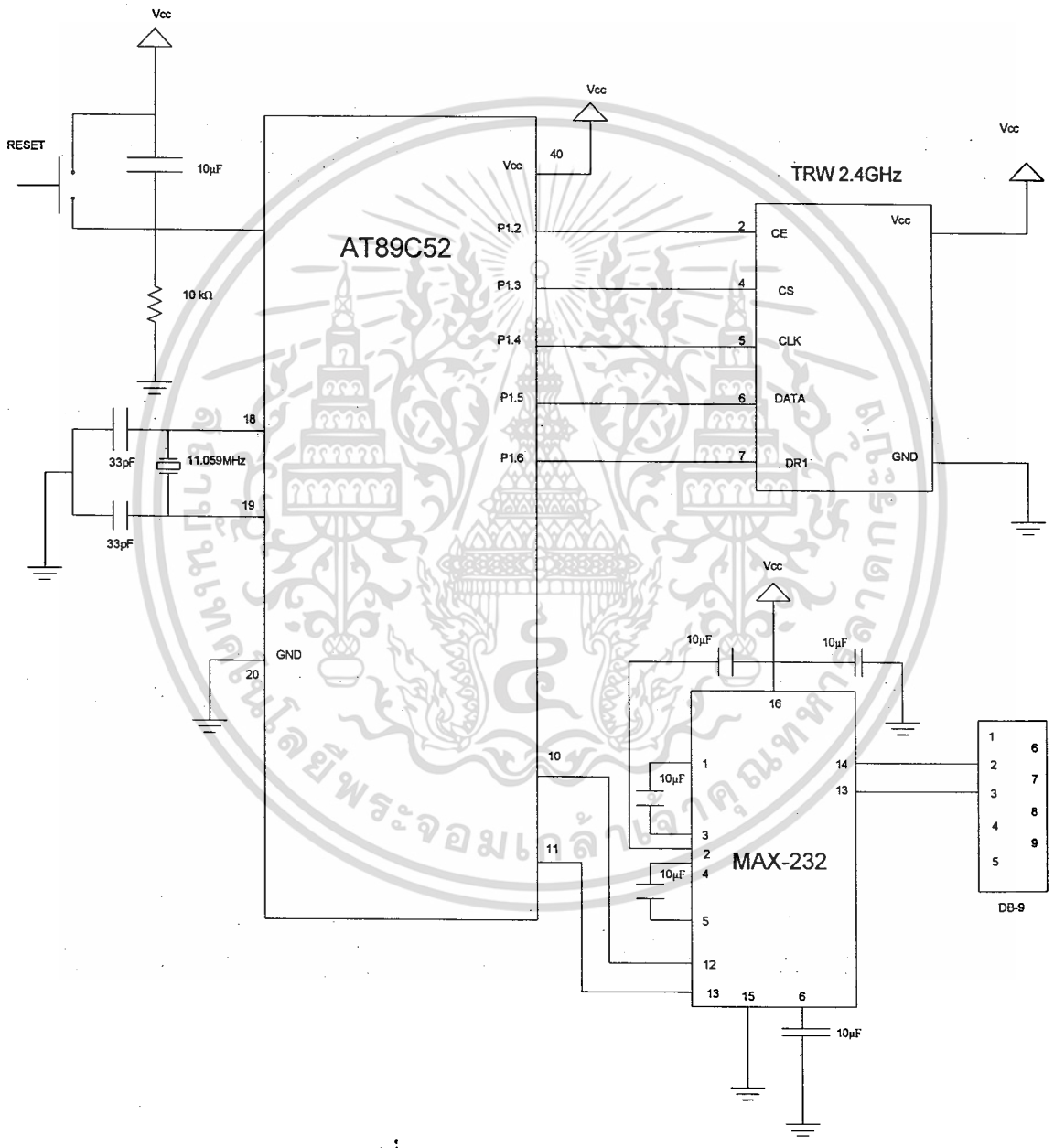
วงจรของสถานีวัดอุณหภูมิจะประกอบด้วยไมโครคอนโทรลเลอร์ที่ควบคุมการทำงานของเซ็นเซอร์วัดอุณหภูมิในการวัดค่าอุณหภูมิ และตัวส่งสัญญาณแบบไร้สายผ่านคลื่นวิทยุย่าน 2.4 GHz ที่ทำการส่งสัญญาณโดยการควบคุมของไมโครคอนโทรลเลอร์



รูปที่ 3.5 วงจรภาคส่งสัญญาณของสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์

3.1.4 วงจรรับของสถานีรับค่าอุณหภูมิ

วงจรของสถานีรับค่าอุณหภูมิประกอบด้วยไมโครคอนโทรลเลอร์ที่ควบคุมการทำงานของตัวรับสัญญาณแบบไร้สายในการรับค่าอุณหภูมิและการควบคุมการส่งข้อมูลให้คอมพิวเตอร์ผ่านไอซี MAX232

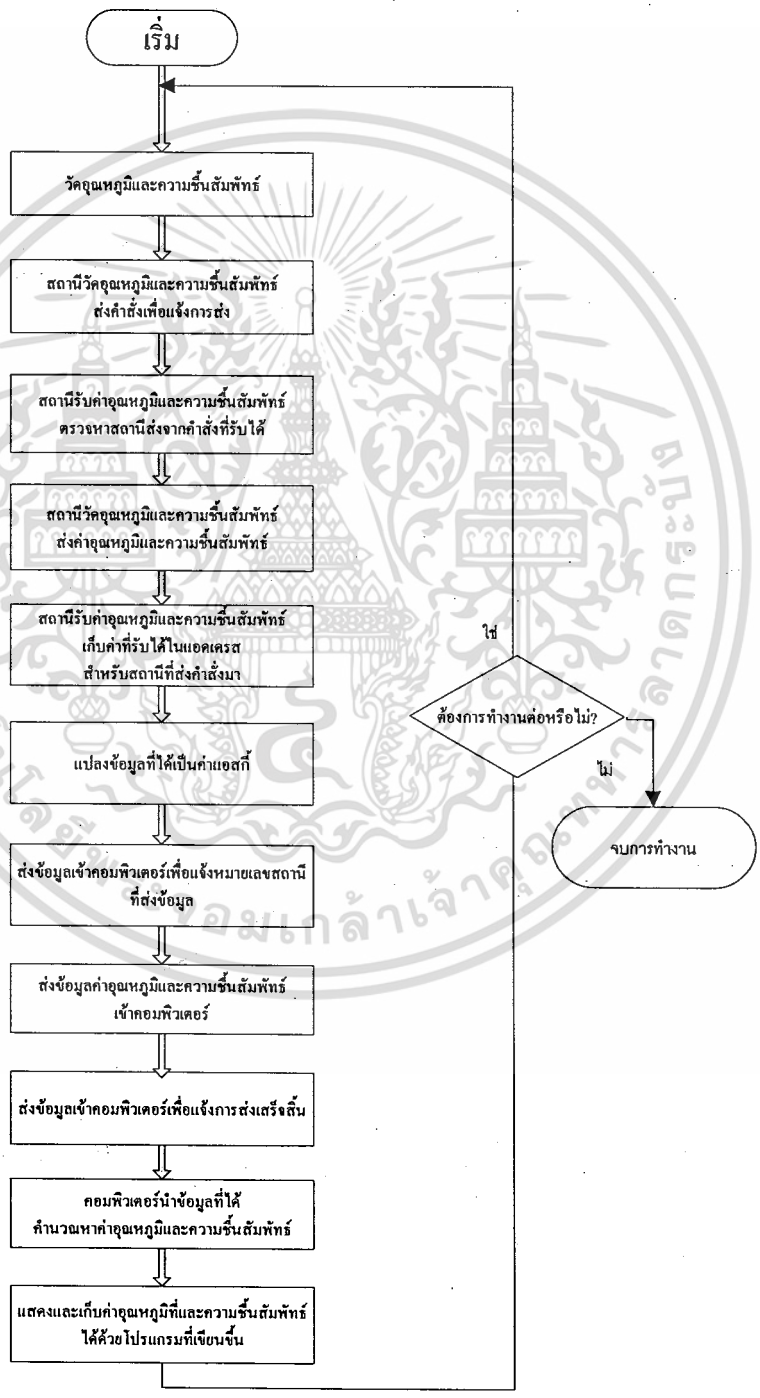


รูปที่ 3.6 วงจรภาครับสัญญาณ

3.2 การออกแบบในส่วนของซอฟแวร์

3.2.1 การทำงานของระบบติดตามผลและส่งข้อมูล อุณหภูมิกับความชื้น

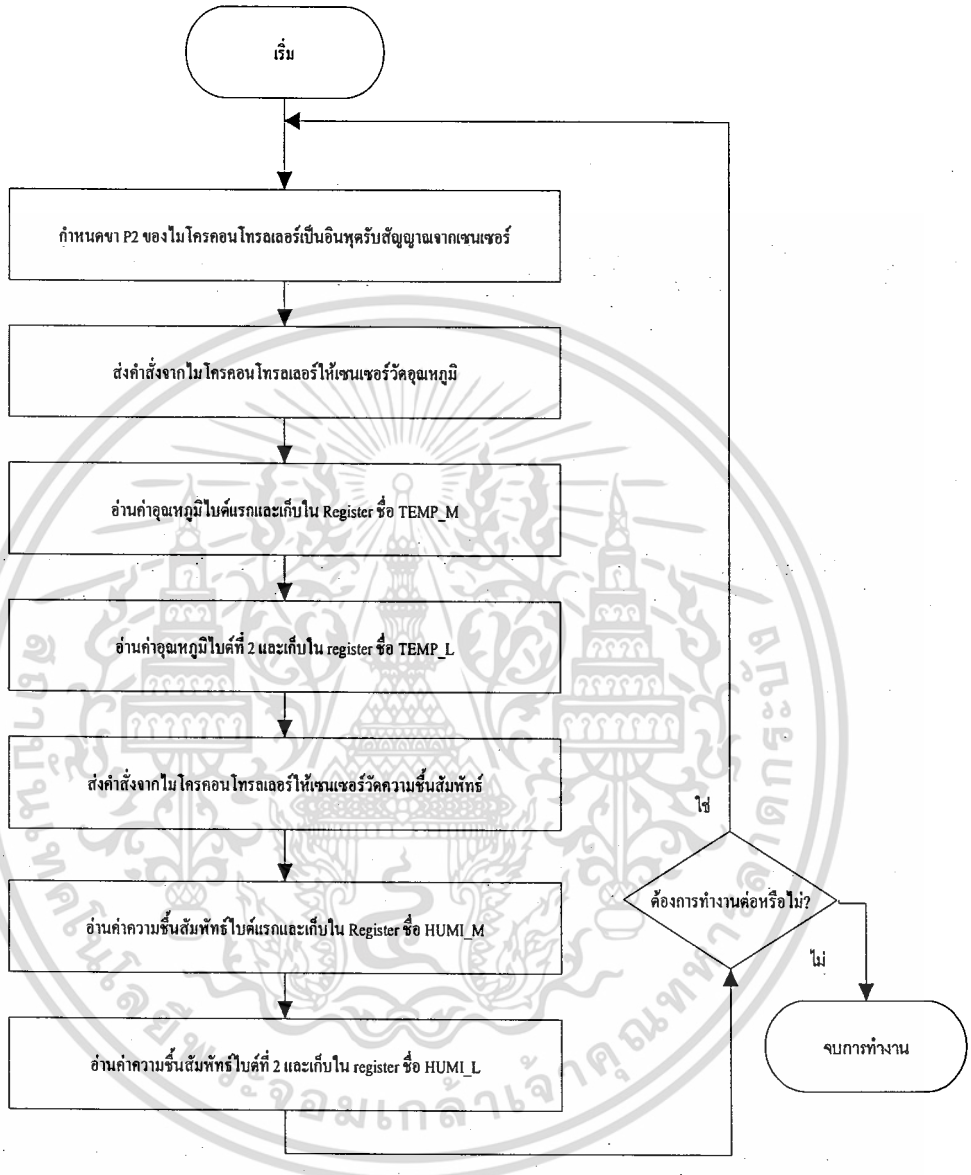
ลำดับการทำงานของระบบติดตามผลและส่งข้อมูลอุณหภูมิกับความชื้นเป็นดังรูปที่ 3.7



รูปที่ 3.7 โฟลว์ชาร์ตการแสดงการทำงานของระบบติดตามผลอุณหภูมิ

3.2.2 การออกแบบซอฟแวร์ติดต่อกับเซนเซอร์วัดอุณหภูมิ

ลำดับขั้นการทำงานของตัวเซ็นเซอร์วัดอุณหภูมิ SHT15 เป็นดังรูปที่ 3.8



รูปที่ 3.8 โฟลว์ชาร์ตการทำงานของเซนเซอร์วัดอุณหภูมิ

3.2.3 การออกแบบการตั้งค่าการทำงานของตัวส่ง/รับสัญญาณด้วย TRW 2.4GHz

กำหนดค่าการทำงานต่างๆทั้ง 18 ไบต์ ซึ่งประกอบด้วยส่วนต่างๆดังนี้

ส่วนทดสอบ

กำหนดให้มีค่า 10001110B 00001000B 000111B

ส่วน PLL_CTRL: ใช้ควบคุมการตั้งค่าการทำงานของเฟดลอคลูป สำหรับส่วนทดสอบ

กำหนดให้เป็น 00 คือ ใช้ที่ตัวส่ง/ไม่ใช้ที่ตัวรับ

ส่วน DATA2_W เป็นส่วนที่ใช้ในการกำหนดความยาวของข้อมูลในแพคเกตของช่องสัญญาณที่ 2

กำหนดให้มีค่า 08H

ส่วน DATA_W เป็นส่วนที่ใช้ในการกำหนดความยาวของข้อมูลในแพคเกตของช่องสัญญาณที่ 1

กำหนดให้มีค่า 08H ส่งครั้งละ 1 ไบต์

ส่วน ADDR2 เป็นแอดเดรสของตัวรับสัญญาณช่องรับสัญญาณที่ 2

กำหนดให้เป็น C0H AAH 55H AAH 55H

ส่วน ADDR1 เป็นแอดเดรสของตัวรับสัญญาณช่องสัญญาณที่ 1

กำหนดให้เป็น AAH 55H AAH 55H AAH

ส่วน ADDR_W ใช้กำหนดความยาวแอดเดรสของตัวรับสัญญาณ

กำหนดให้เป็น 101000B = 40 บิต

ส่วน CRC กำหนดความยาวของ CRC และเลือกใช้/ไม่ใช้งาน CRC

กำหนดให้เป็น 11B คือความยาว 16 บิต ใช้งาน CRC

ส่วน RX2_EN เป็นบิตที่ใช้ในการกำหนดจำนวนช่องสัญญาณที่ใช้ในการสื่อสาร

กำหนดให้เป็นลอจิก 0 ให้อุปกรณ์ภาครับรับ 1 ช่องสัญญาณ

ส่วน CM ใช้ในการเลือกรูปแบบการสื่อสาร

กำหนดให้เป็นลอจิก 1 เลือกให้ทำการสื่อสารแบบชอคเบิร์ส

ส่วน RFDR_SB เป็นการกำหนดความเร็วของข้อมูลที่จะใช้ในการส่งสัญญาณ

กำหนดให้เป็นลอจิก 0 ส่งข้อมูลด้วยความเร็ว 250 Kbps

ส่วน XO_F

กำหนดให้เป็น 011B

ส่วน RF_PWR เป็นบิตที่ใช้ในการกำหนดกำลังส่งของสัญญาณ

กำหนดให้เป็น 11B คือส่งด้วย 0dBm

ส่วน RF_channel กำหนดช่องสัญญาณที่ใช้ในการสื่อสาร

กำหนดให้มีค่า 0011101B = 29 ซึ่งเป็นความถี่ที่ 2429 MHz

ส่วน RXEN เลือกการทำงานให้อุปกรณ์รับ หรือส่งสัญญาณ

ที่ตัวส่งกำหนดให้เป็น 0 เพื่อส่งสัญญาณ

ที่ตัวรับกำหนดให้เป็น 1 เพื่อรับสัญญาณ

ดังนั้นค่าการตั้งค่า 18 ไบต์ ของตัวส่งจะมีค่าดังนี้

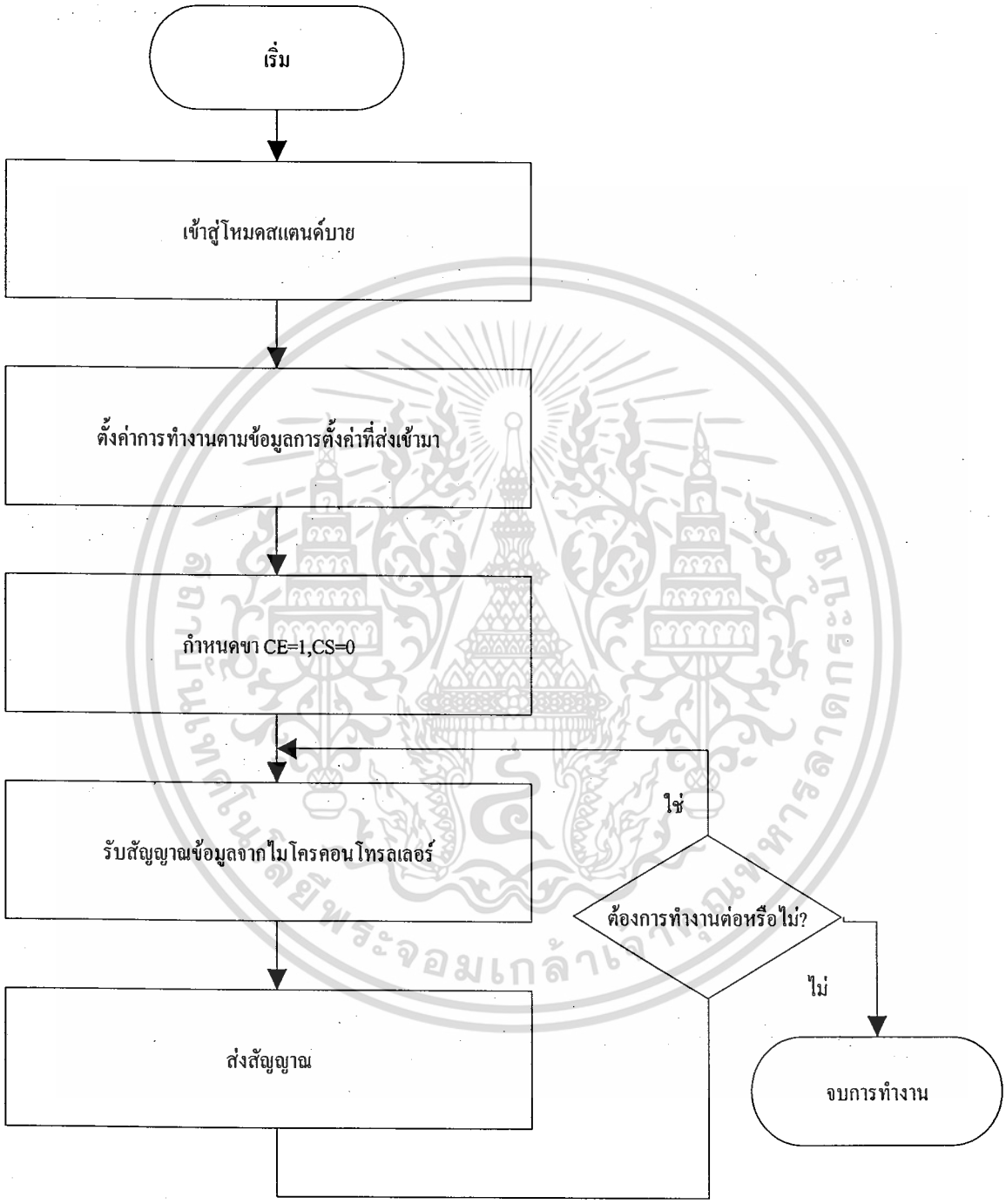
8EH 08H 1CH 08H C0H AAH 55H AAH 55H AAH 55H AAH 55H AAH A3H 4FH 3AH

ดังนั้นค่าการตั้งค่า 18 ไบต์ ของตัวรับจะมีค่าดังนี้

8EH 08H 1CH 08H 08H C0H AAH 55H AAH 55H AAH 55H AAH 55H AAH A3H 4FH 3BH

3.2.4 การทำงานของตัวส่งสัญญาณ

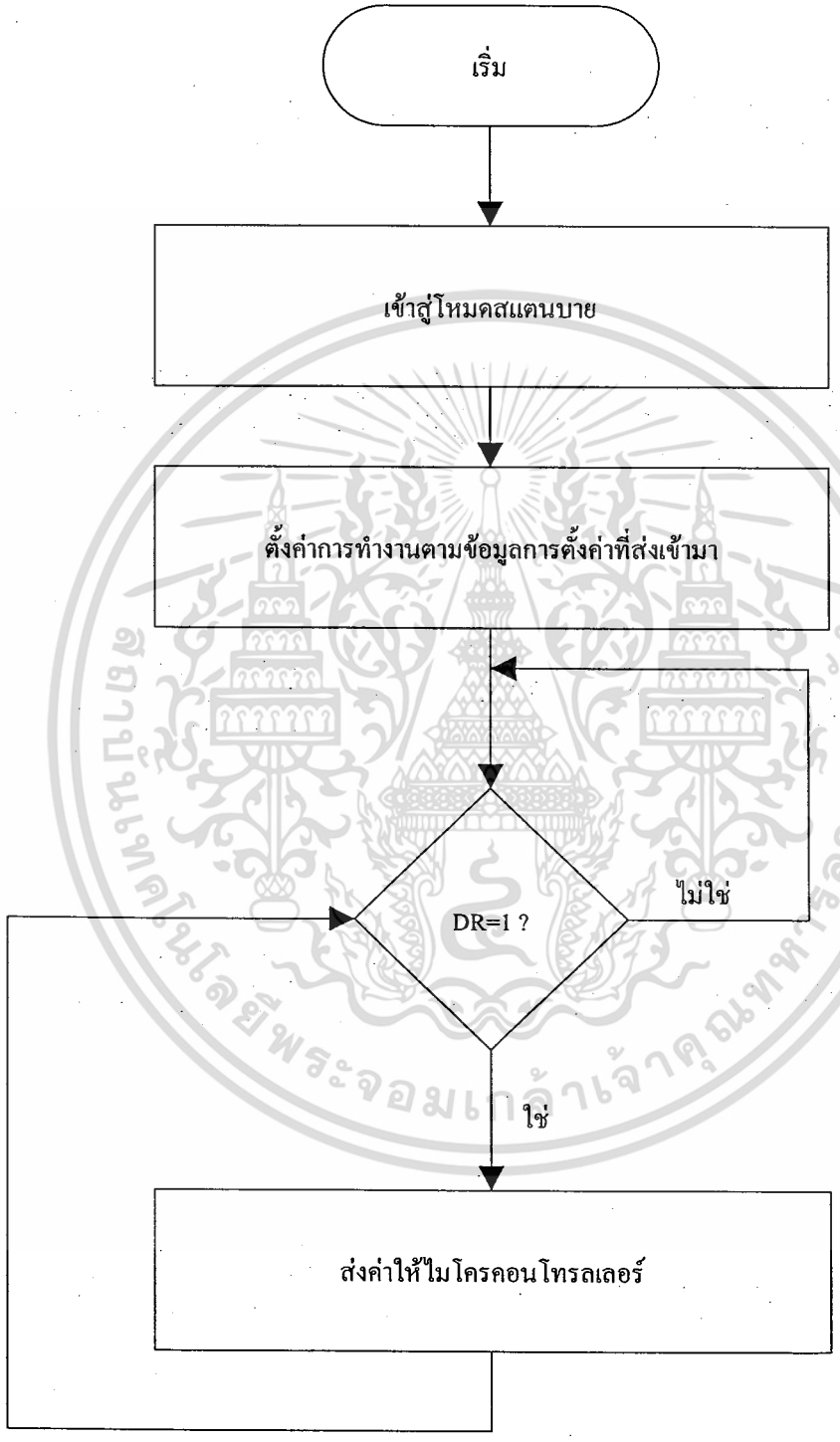
ขั้นตอนการทำงานของตัว ส่ง/รับ สัญญาณ TRW 2.4 GHz มีขั้นตอนดังรูปที่ 3.9



รูปที่ 3.9 โฟลว์ชาร์ตการทำงานของตัวส่งสัญญาณ

3.2.5 การทำงานของตัวรับสัญญาณ

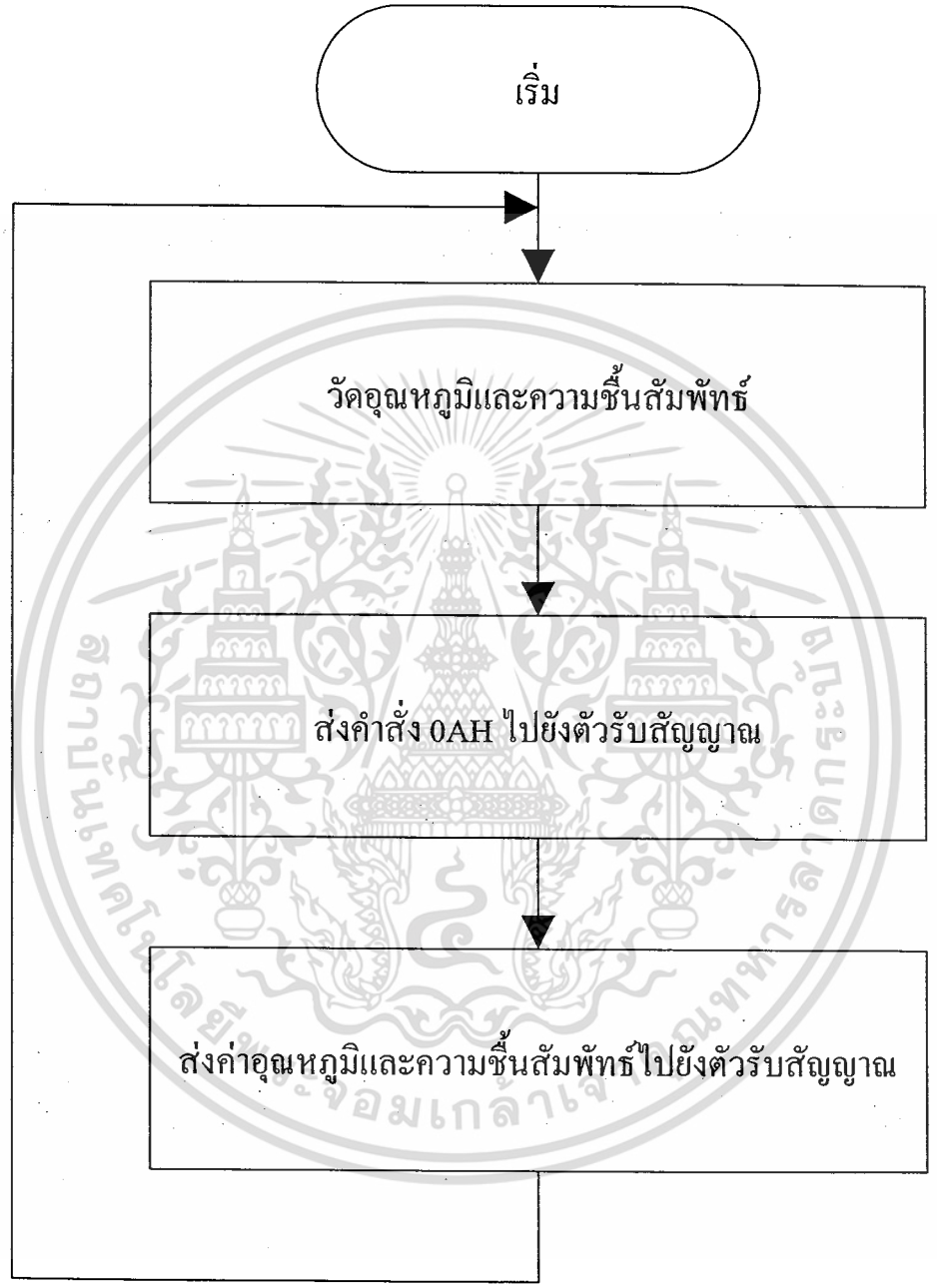
ขั้นตอนการทำงานของตัวรับสัญญาณ TRW 2.4 GHz มีขั้นตอนดังรูปที่ 3.10



รูปที่ 3.10 โฟลว์ชาร์ตการทำงานของตัวรับสัญญาณ

3.2.6 การทำงานของชุดสถานีวัดอุณหภูมิที่ 1

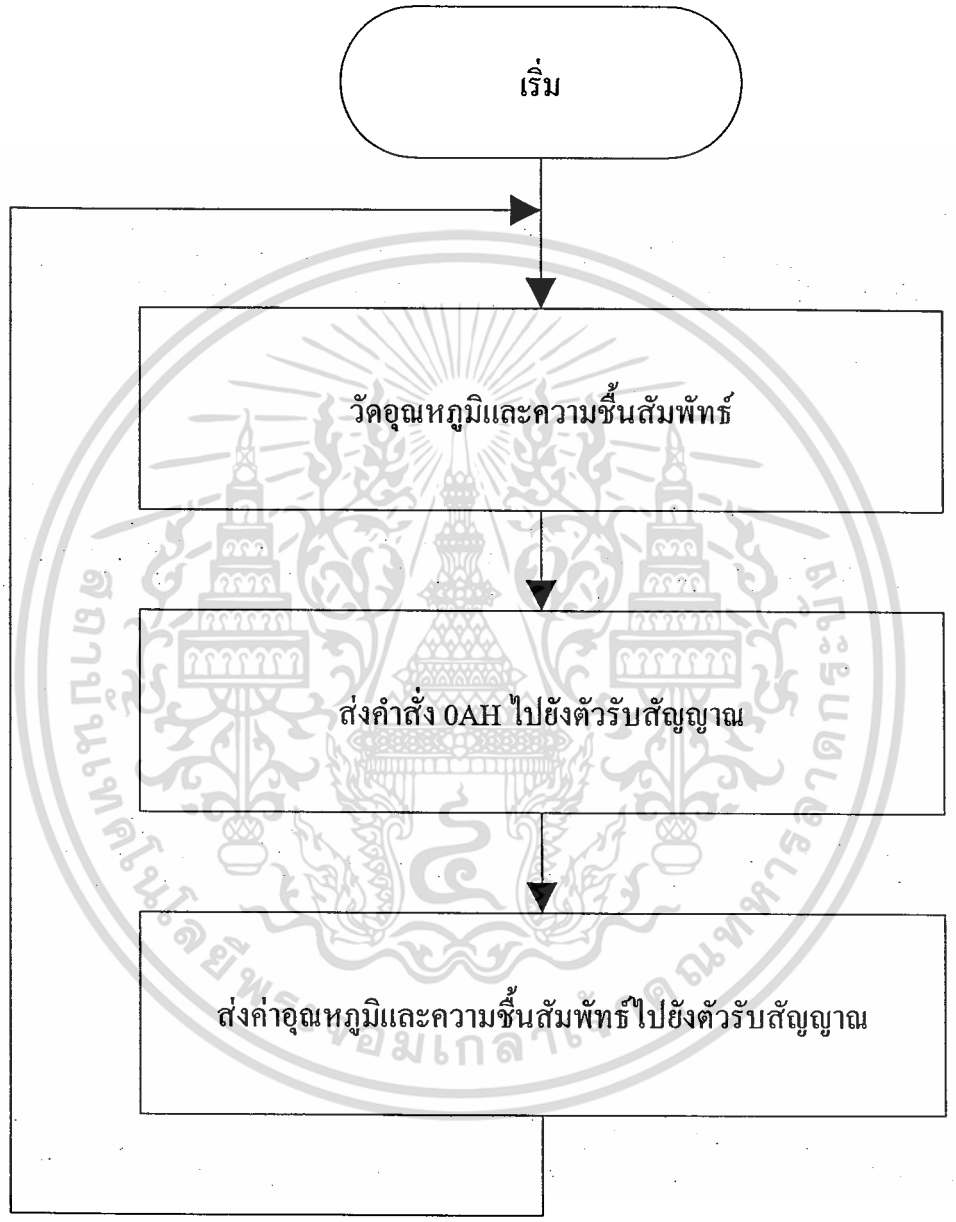
ขั้นตอนการทำงานของชุดสถานีวัดอุณหภูมิที่ 1 มีขั้นตอนดังรูปที่ 3.11



รูปที่ 3.11 โฟลว์ชาร์ตการทำงานของชุดสถานีวัดอุณหภูมิที่ 1

3.2.7 การทำงานของชุดสถานีวัดอุณหภูมิที่ 2

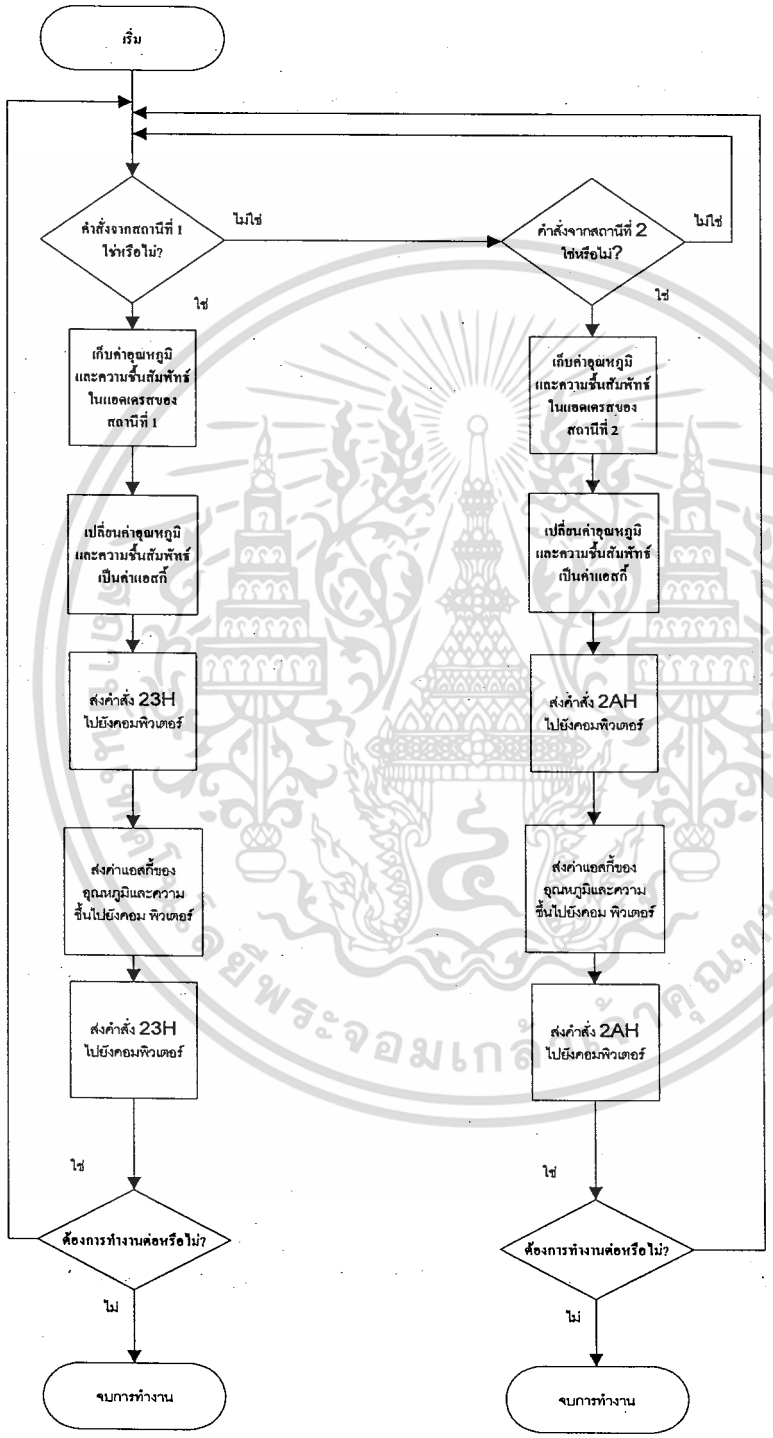
ขั้นตอนการทำงานของชุดสถานีวัดอุณหภูมิที่ 2 มีขั้นตอนดังรูปที่ 3.12



รูปที่ 3.12 โฟลว์ชาร์ตการทำงานของชุดสถานีวัดอุณหภูมิที่ 2

3.2.8 การทำงานของชุดรับสัญญาณ

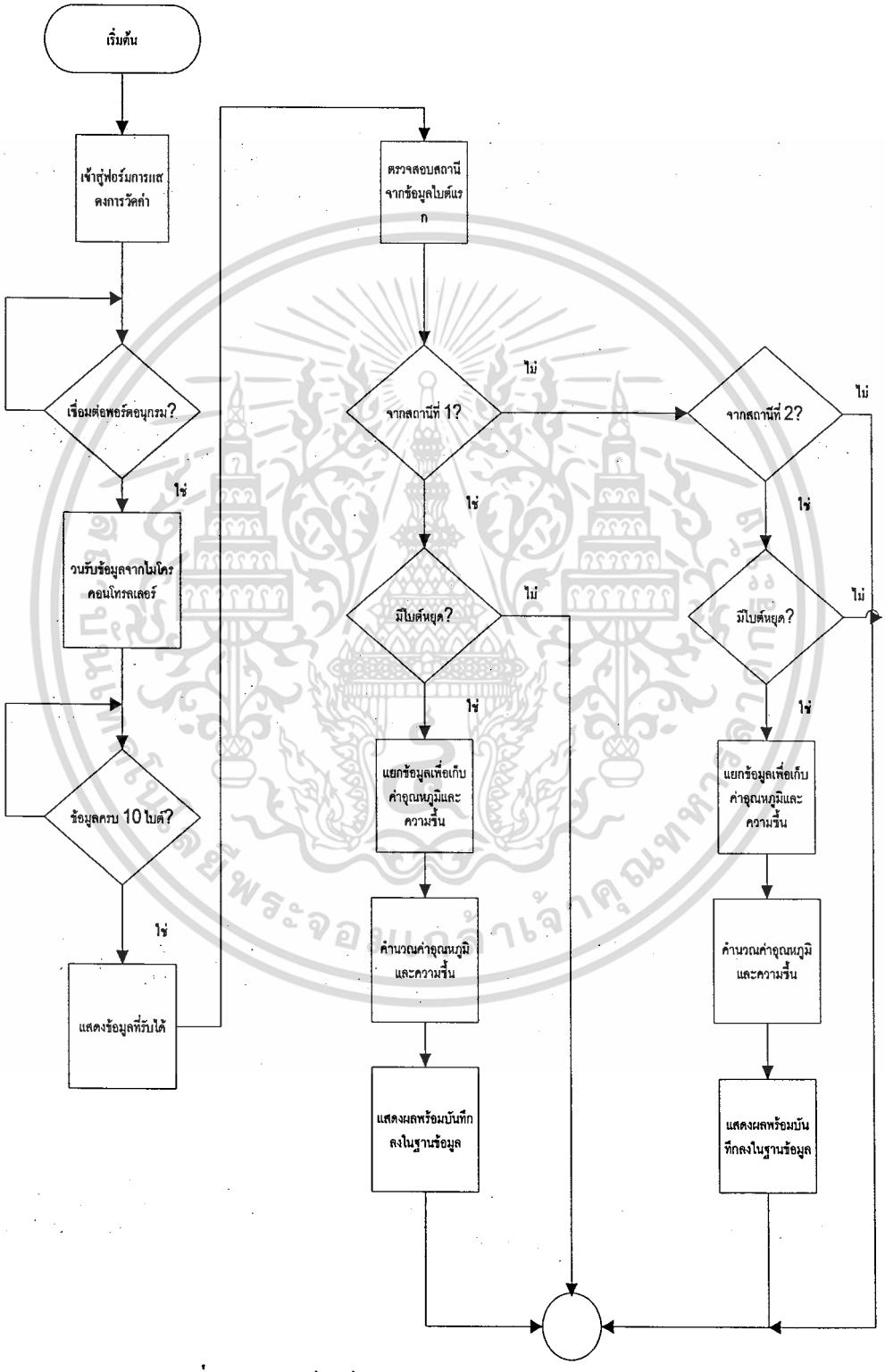
ขั้นตอนการทำงานของชุดสถานีวัดอุณหภูมิที่ 1 และ 2 มีขั้นตอนดังรูปที่ 3.13



รูปที่ 3.13 โฟลว์ชาร์ตการทำงานของชุดรับสัญญาณ

3.3 การประมวลผลในคอมพิวเตอร์

ขั้นตอนการประมวลผลข้อมูลในคอมพิวเตอร์มีขั้นตอนดังรูปที่ 3.14



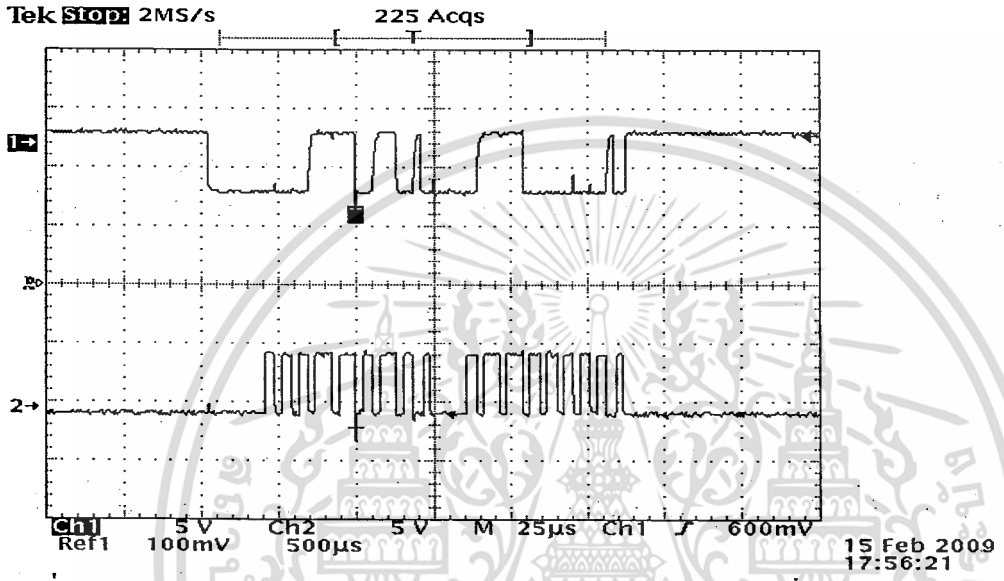
รูปที่ 3.14 โฟลว์ชาร์ตแสดงการประมวลผลในคอมพิวเตอร์

บทที่ 4

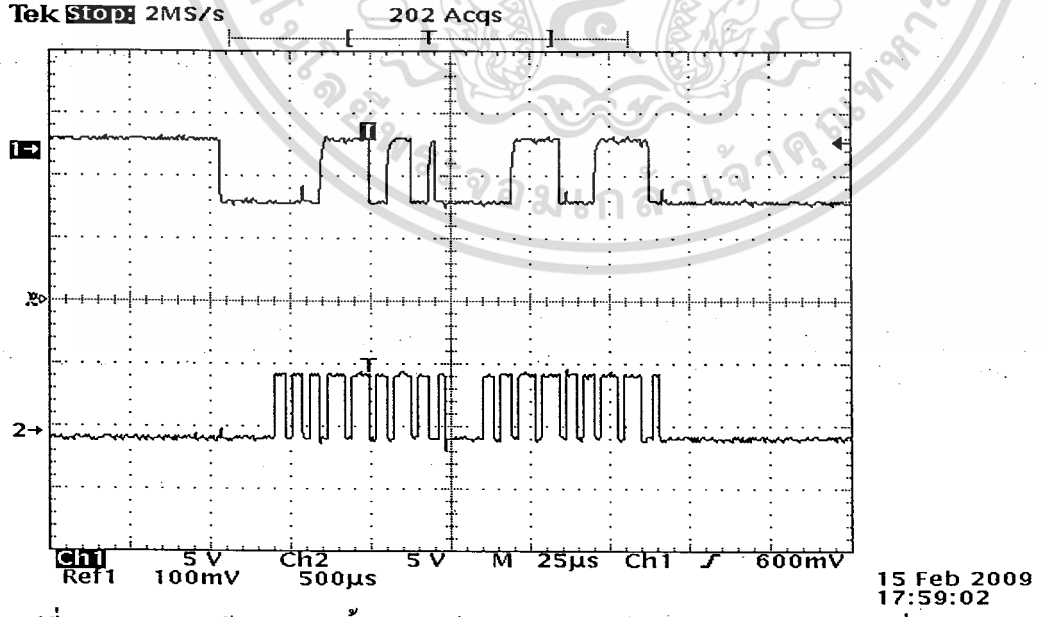
การทดลองและผลการทดลอง

4.1 สัญญาณส่งคำสั่งไปยังเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์

ไมโครคอนโทรลเลอร์ทำการอ่านค่าสัญญาณจากการวัดด้วยเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์



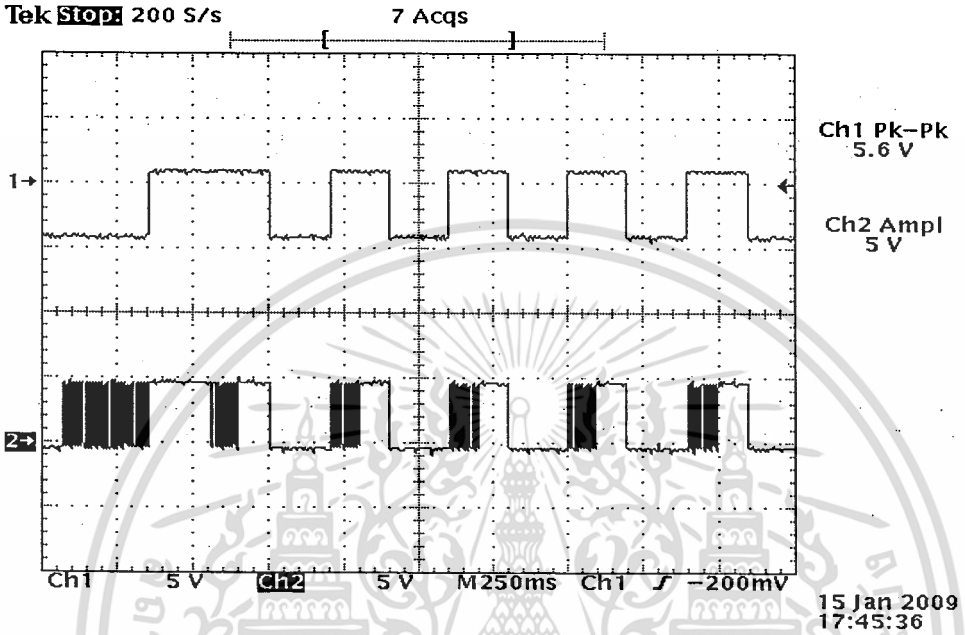
รูปที่ 4.1 สัญญาณข้อมูลอุณหภูมิของตัวเซนเซอร์ SHT-15 (ช่องสัญญาณที่ 1) กับ สัญญาณนาฬิกาของตัวเซนเซอร์ SHT-15 (ช่องสัญญาณที่ 2)



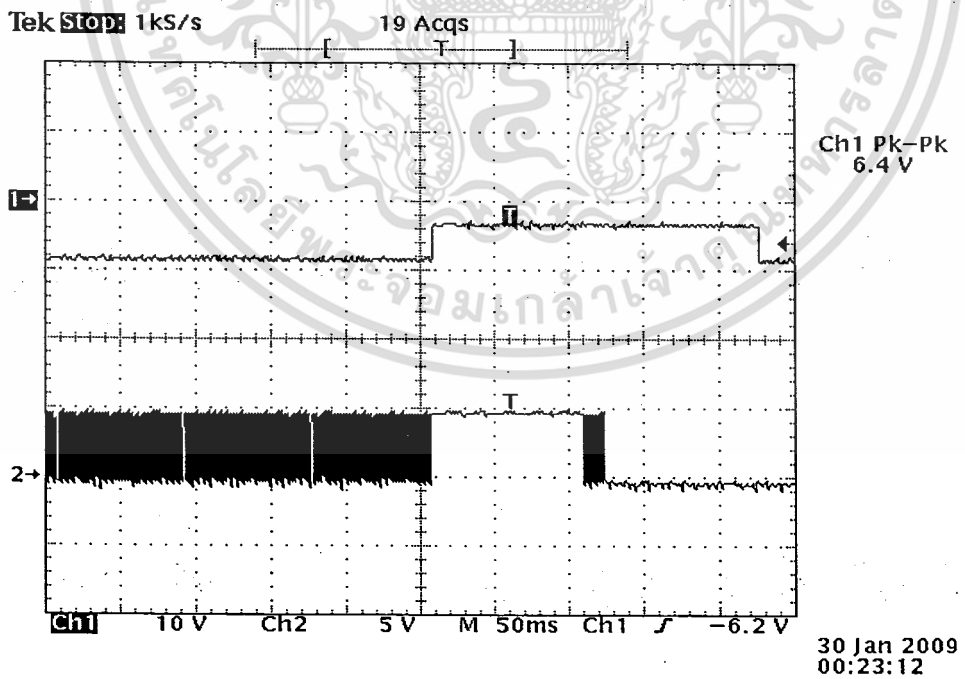
รูปที่ 4.2 สัญญาณข้อมูลความชื้นสัมพัทธ์ของตัวเซนเซอร์ SHT-15 (ช่องสัญญาณที่ 1) กับ สัญญาณนาฬิกาของตัวเซนเซอร์ SHT-15 (ช่องสัญญาณที่ 2)

4.2 สัญญาณที่ใช้ในการควบคุมการทำงานของตัวส่ง/รับสัญญาณ TRW 2.4 GHz

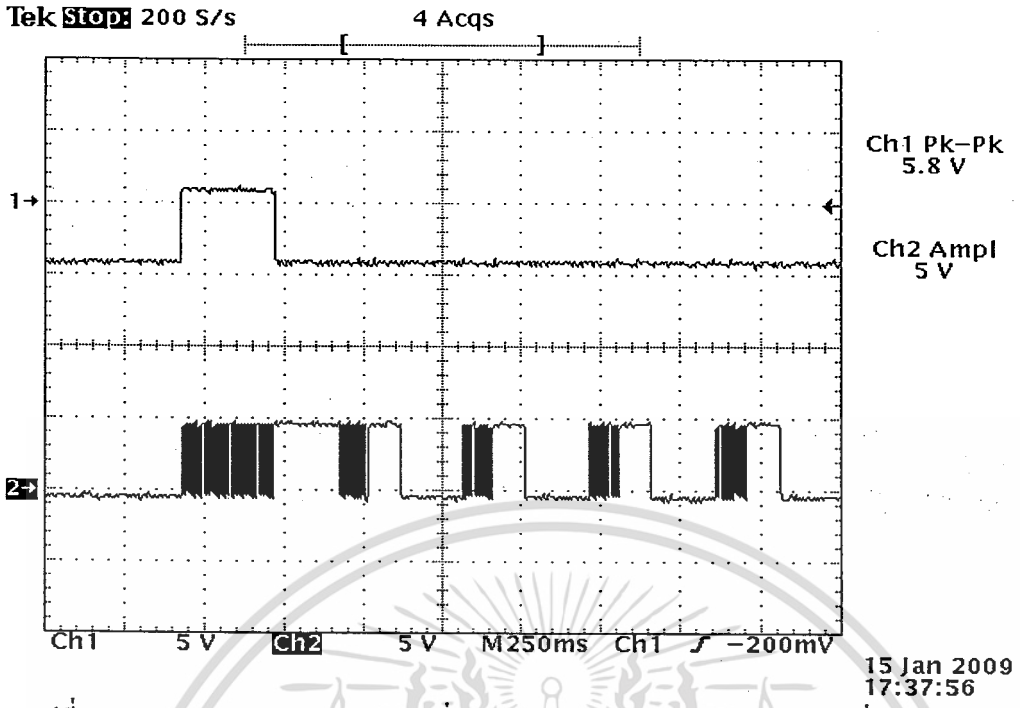
เป็นสัญญาณจากขา TRW 2.4 GHz ของทั้งตัวส่ง/รับ โดยเปรียบเทียบกับขานาฬิกาของ TRW 2.4 GHz เป็นดังรูปที่ 4.3 ถึง รูปที่ 4.11 (ยกเว้นรูปที่ 4.9) ตามลำดับ



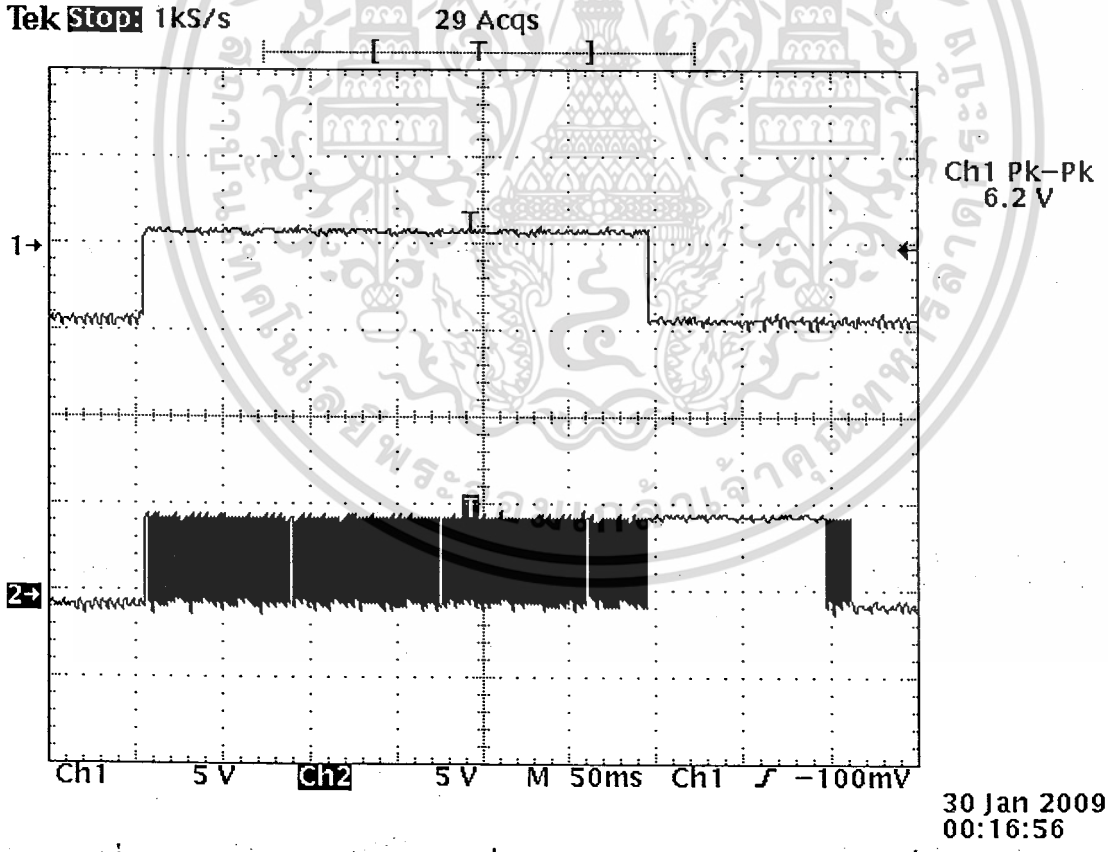
รูปที่ 4.3 สัญญาณขา CE (ช่องสัญญาณที่ 1) กับสัญญาณขา CLK (ช่องสัญญาณที่ 2)ของตัวส่ง



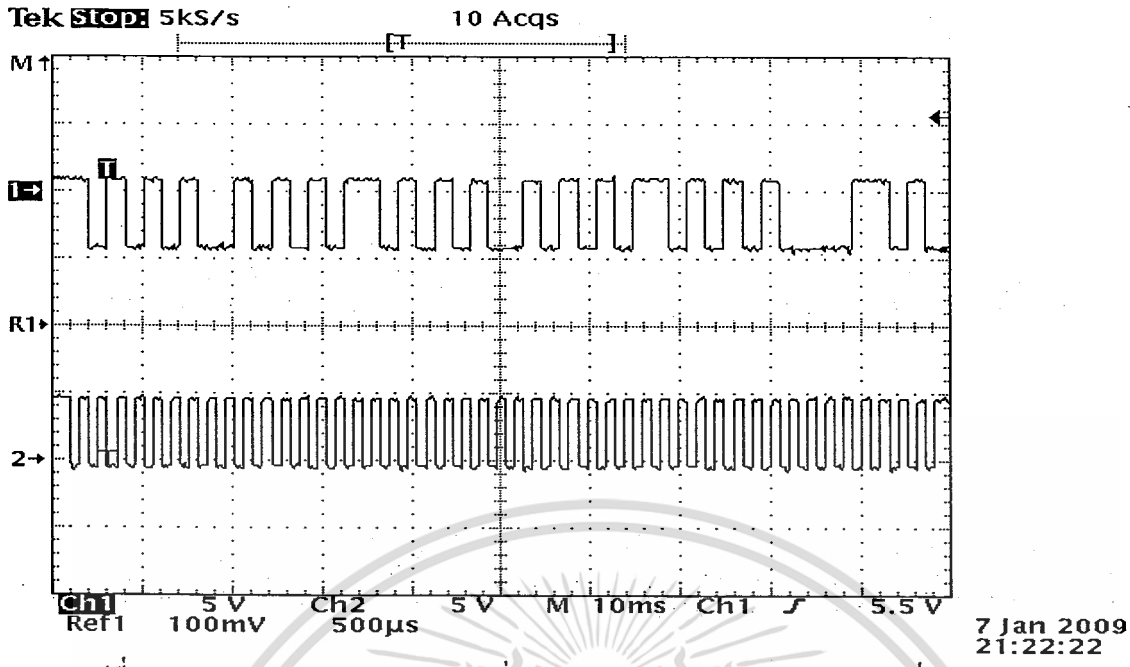
รูปที่ 4.4 สัญญาณขา CE(ช่องสัญญาณที่ 1) กับ สัญญาณขา CLK (ช่องสัญญาณที่ 2)ของตัวรับ



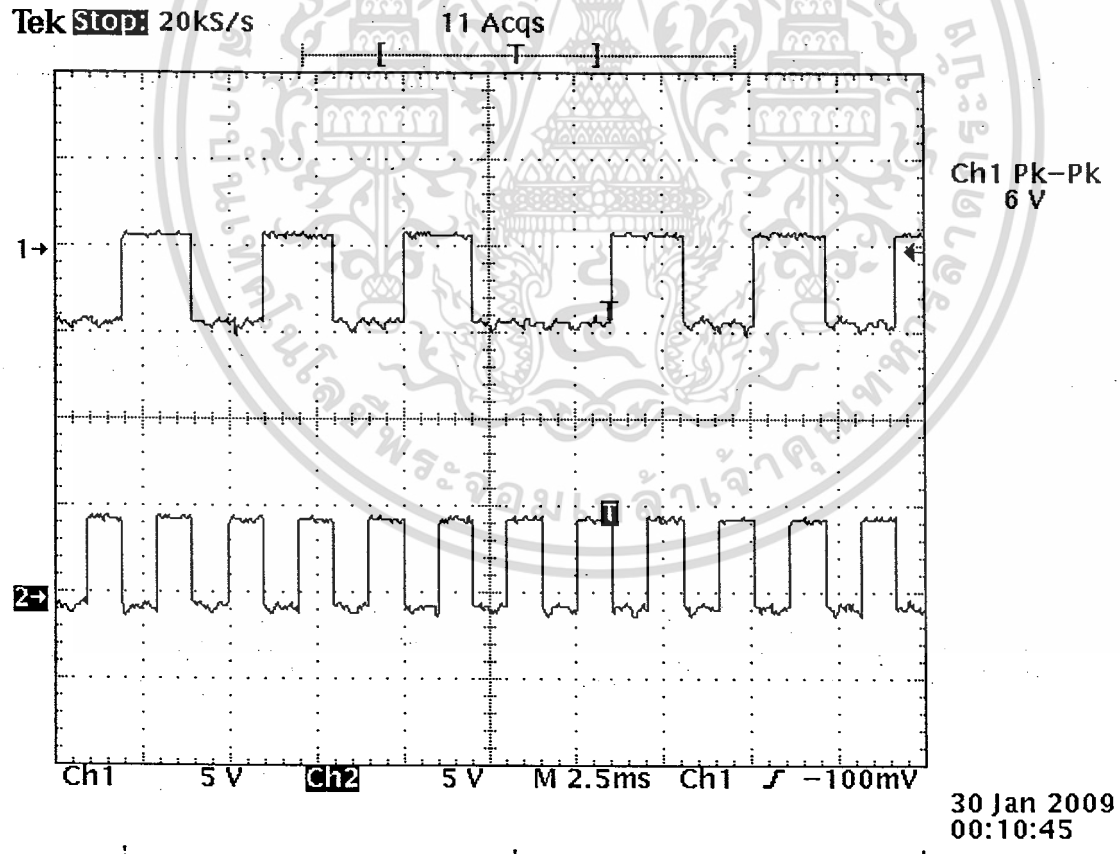
รูปที่ 4.5 สัญญาณขา CS (ช่องสัญญาณที่ 1) กับสัญญาณขา CLK (ช่องสัญญาณที่ 2)ของตัวส่ง



รูปที่ 4.6 สัญญาณขา CS (ช่องสัญญาณที่ 1) กับสัญญาณขา CLK (ช่องสัญญาณที่ 2)ของตัวรับ

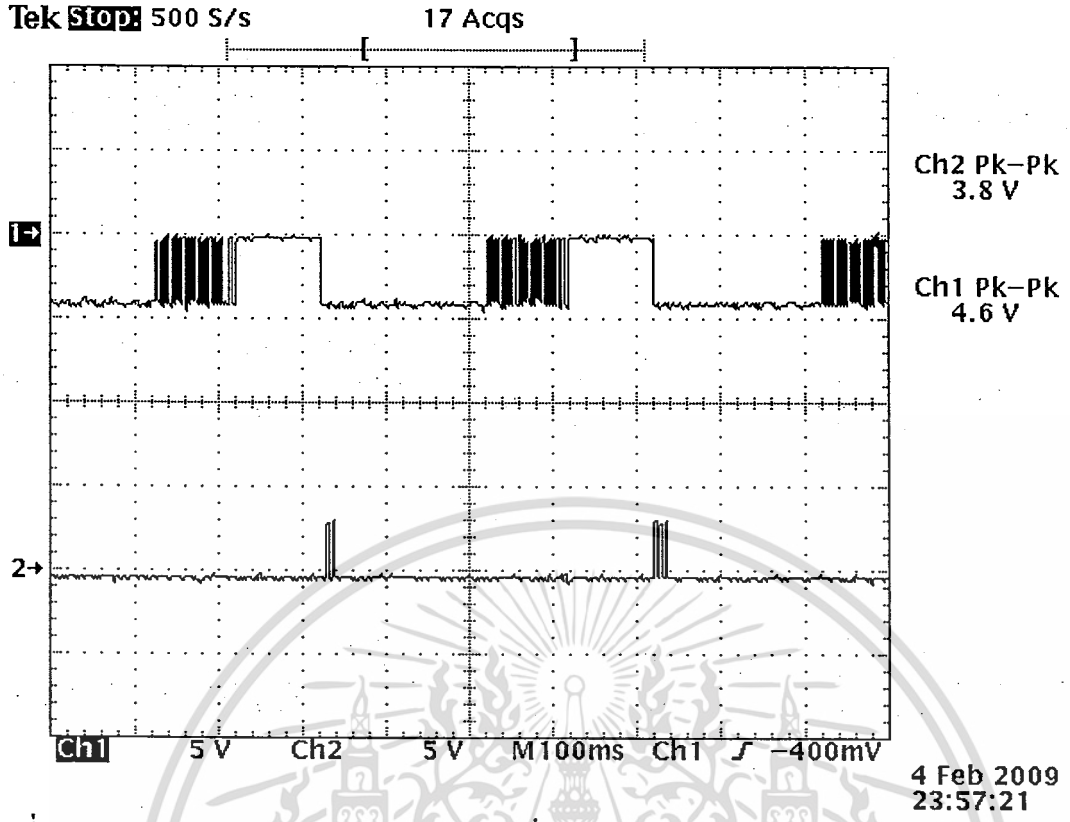


รูปที่ 4.7 สัญญาณขาDAT(ช่องสัญญาณที่ 1) กับ สัญญาณขาCLK(ช่องสัญญาณที่ 2)ของตัวส่ง

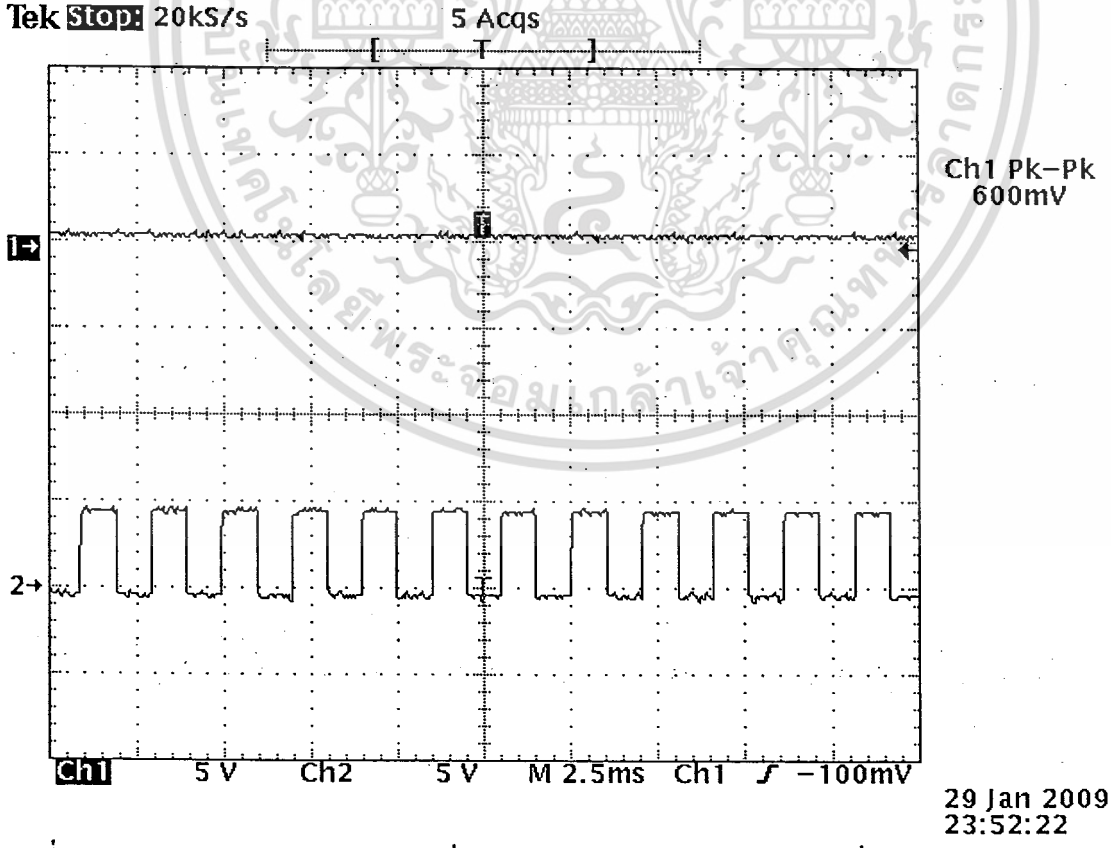


รูปที่ 4.8 สัญญาณขาDAT (ช่องสัญญาณที่ 1) กับสัญญาณขาCLK(ช่องสัญญาณที่ 2)ของตัวรับ

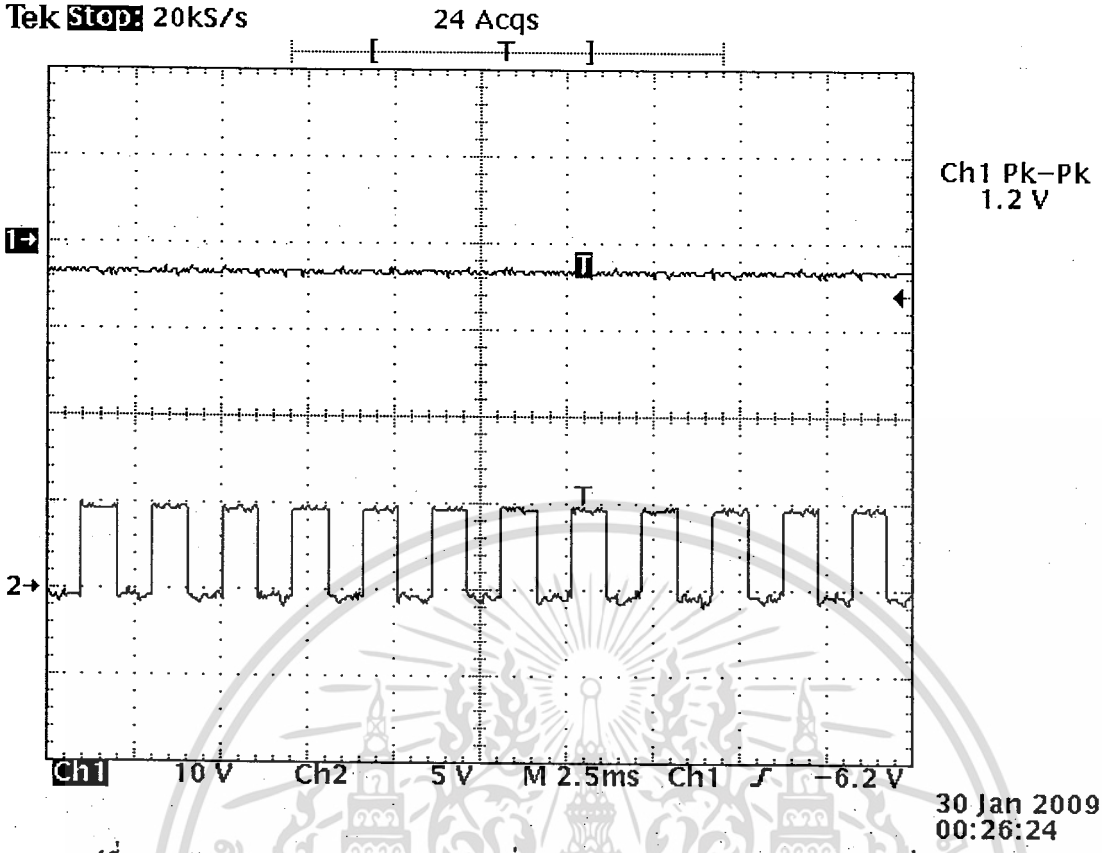
ดังรูปที่ 4.9 เป็นการเปรียบเทียบ สัญญาณข้อมูลตัวส่ง/รับ ของ TRW 2.4 GHz



รูปที่ 4.9 สัญญาณขา DATของตัวส่ง (ช่องสัญญาณที่ 1) กับ สัญญาณขา DATของตัวรับ (ช่องสัญญาณที่ 2)

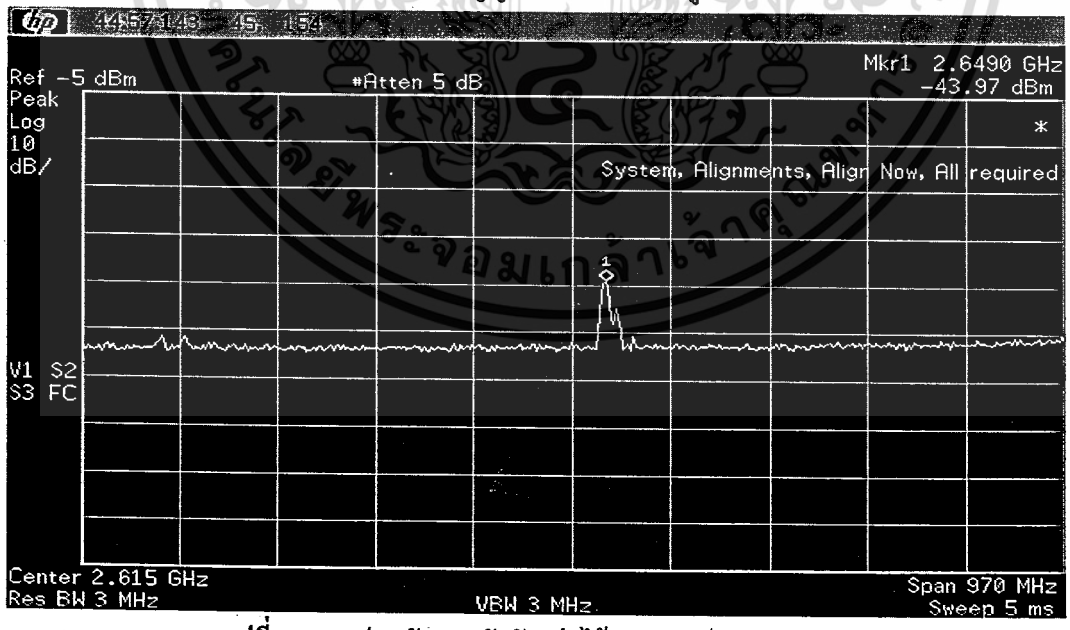


รูปที่ 4.10 สัญญาณขา DR1(ช่องสัญญาณที่ 1)กับสัญญาณขา CLK(ช่องสัญญาณที่ 2)ของตัวส่ง



รูปที่ 4.11 สัญญาณขา DR1(ช่องสัญญาณที่1)กับสัญญาณขาCLK (ช่องสัญญาณที่ 2)ของตัวรับ

ทำการวัดสเปกตรัมของสัญญาณการรับ/ส่งข้อมูล TRW-2.4GHz



รูปที่ 4.12 สเปกตรัมของตัวรับ/ส่งไร้สายของ TRW-2.4GHz

4.3 การส่งข้อมูลเข้าคอมพิวเตอร์

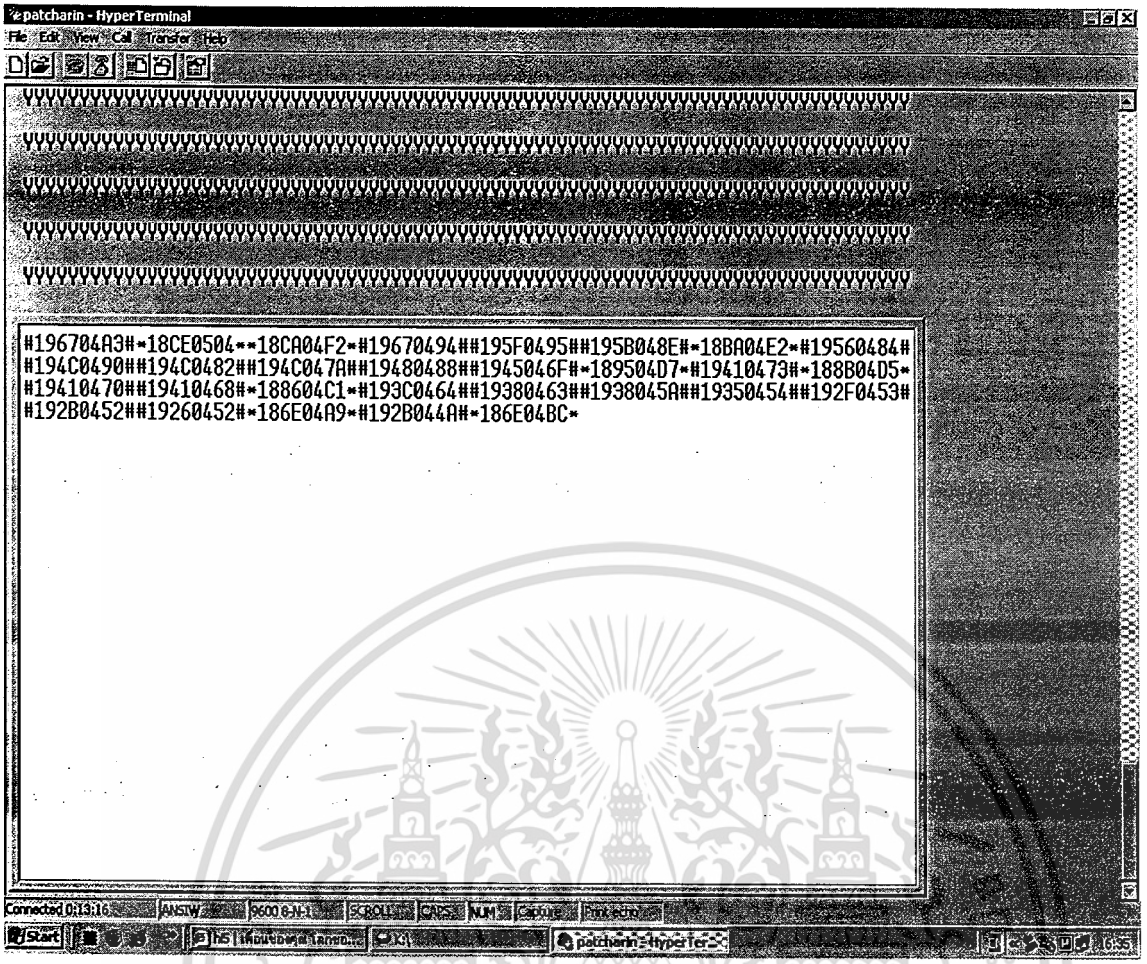
รูปที่ 4.13 เป็นรูปที่แสดงผลการเชื่อมต่อส่งค่าอุณหภูมิและความชื้นให้กับคอมพิวเตอร์ ผลที่ได้ดังกล่าวเป็นผลจากสถานีรับสัญญาณไร้สายที่ส่งให้กับคอมพิวเตอร์ ซึ่งได้ข้อมูลมาจากสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์ทั้ง 2 สถานี

โดยสถานีที่จะส่งข้อมูลเข้ามาทั้งหมด 10 ไบต์ ซึ่งเท่ากับ 10 ตัวอักษรเนื่องจากส่งค่าเข้ามาด้วยค่าแอสกี้แทนค่าที่เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์วัดได้ ประกอบไปด้วย

- ไบต์แรกไบต์เริ่มต้นที่ทำหน้าที่แจ้งว่าเป็นข้อมูลของสถานีที่ 1 ซึ่งกำหนดให้เป็นค่าแอสกี้ของ #
- ไบต์ที่ 2 ถึง ไบต์ที่ 5 เป็นไบต์ค่าแอสกี้ที่แทนค่าอุณหภูมิ
- ไบต์ที่ 6 ถึง ไบต์ที่ 9 เป็นไบต์ค่าแอสกี้ที่แทนค่าความชื้นสัมพัทธ์
- ไบต์ที่ 10 เป็นไบต์ค่าแอสกี้ที่ใช้แจ้งว่าเป็นไบต์สุดท้ายของข้อมูลจากสถานีที่ 1 ซึ่งกำหนดให้เป็นค่าแอสกี้ของ #

สถานีที่ 2 จะส่งข้อมูลเข้ามาทั้งหมด 10 ไบต์ ซึ่งเท่ากับ 10 ตัวอักษร เช่นเดียวกัน ซึ่งประกอบไปด้วย

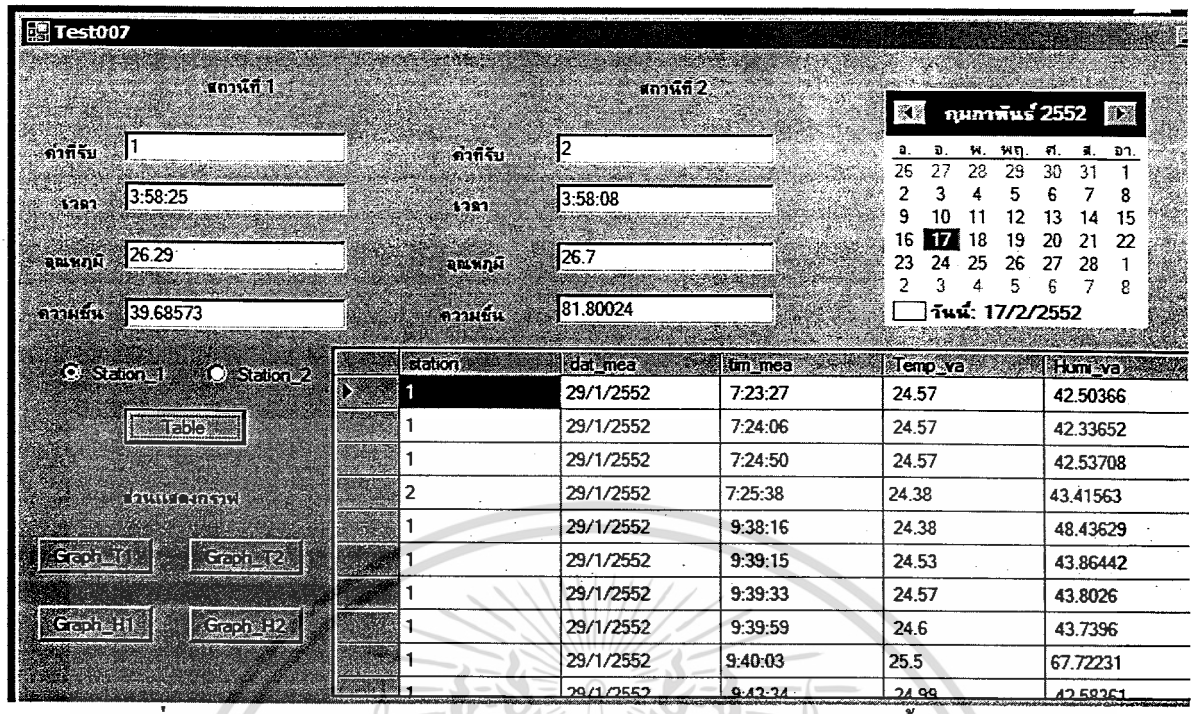
- ไบต์แรกไบต์เริ่มต้นที่ทำหน้าที่แจ้งว่าเป็นข้อมูลของสถานีที่ 2 ซึ่งกำหนดให้เป็นค่าแอสกี้ของ *
- ไบต์ที่ 2 ถึง ไบต์ที่ 5 เป็นไบต์ค่าแอสกี้ที่แทนค่าอุณหภูมิ
- ไบต์ที่ 6 ถึง ไบต์ที่ 9 เป็นไบต์ค่าแอสกี้ที่แทนค่าความชื้นสัมพัทธ์
- ไบต์ที่ 10 เป็นไบต์ค่าแอสกี้ที่ใช้แจ้งว่าเป็นไบต์สุดท้ายของข้อมูลจากสถานีที่ 2 ซึ่งกำหนดให้เป็นค่าแอสกี้ของ *



รูปที่ 4.13 ข้อมูลค่าอุณหภูมิและความชื้นสัมพัทธ์ของแต่ละสถานีที่ส่งเข้าคอมพิวเตอร์

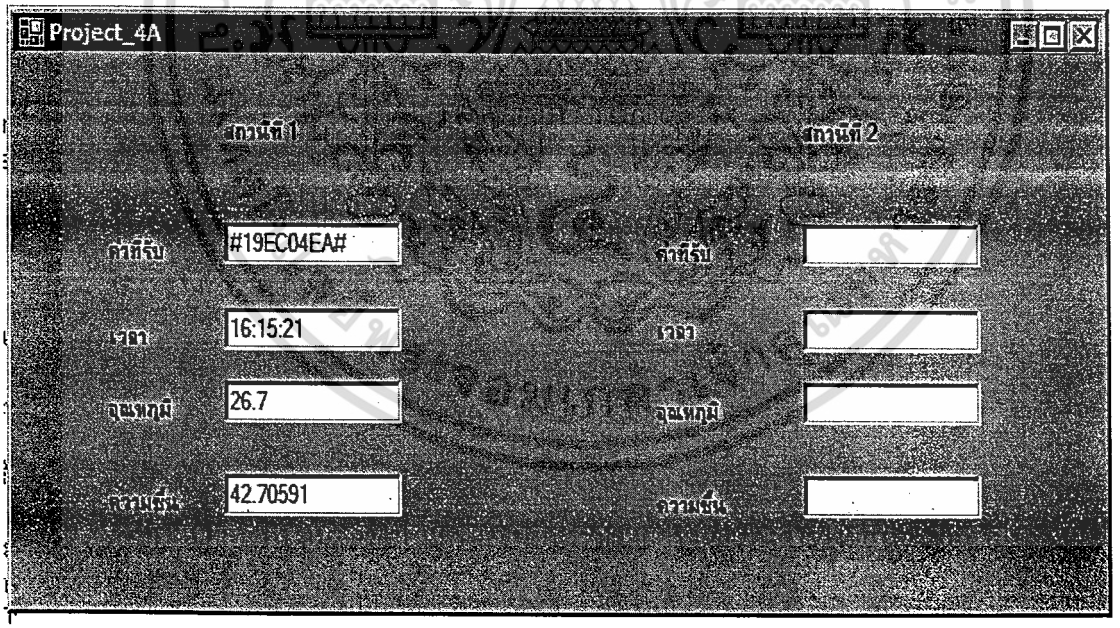
4.4 การประมวลผลในคอมพิวเตอร์

โปรแกรมระบบจะแสดงข้อมูลดังรูปที่ 4.14 เมื่อรับข้อมูลได้จากสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์ที่ 1 และ สถานีที่ 2 ตามลำดับ โดยแสดงหมายเลขสถานีส่งที่ช่อง “สถานี” แสดงค่าอุณหภูมิที่โปรแกรมคำนวณได้ในช่อง “ค่าอุณหภูมิ” และแสดงค่าความชื้นสัมพัทธ์ที่ช่อง “ค่าความชื้น”



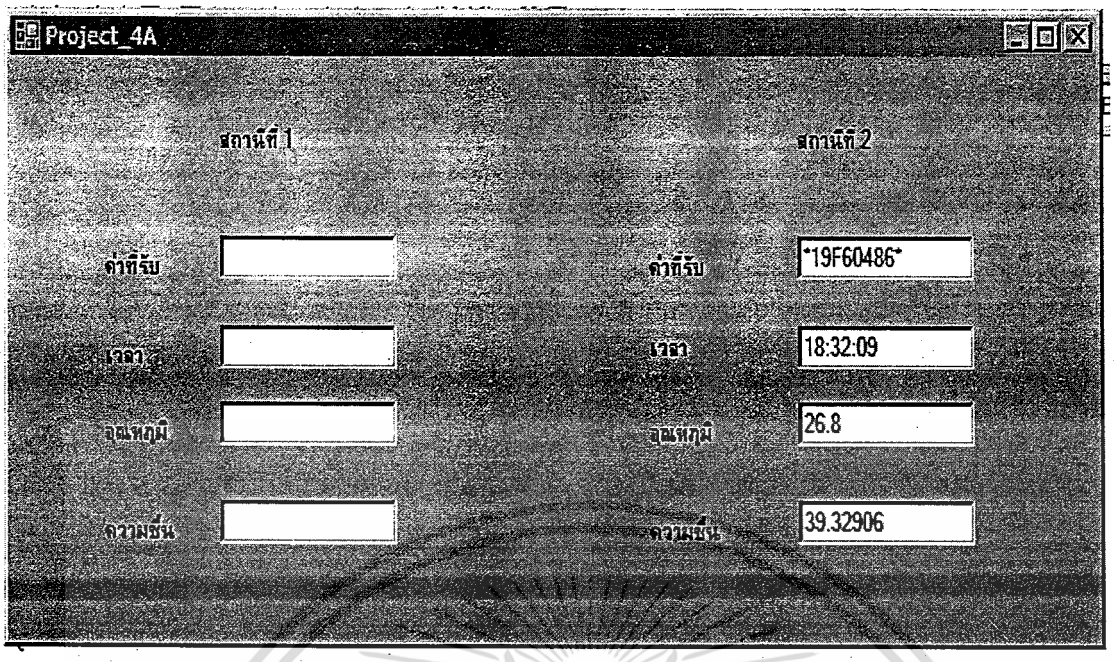
รูปที่ 4.14 การทำงานของโปรแกรมระบบประมวลผลรวมแสดงตารางของทั้ง 2 สถานี

โปรแกรมระบบประมวลผลรับค่าแอสกี้ที่เข้ามาทางสถานีที่ 1 แสดงเวลาที่เข้ามา และคำนวณหาค่าอุณหภูมิและค่าความชื้นสัมพัทธ์



รูปที่ 4.15 การทำงานของโปรแกรมระบบประมวลผล เมื่อรับค่าจากสถานีที่ 1

โปรแกรมระบบประมวลผลรับค่าแอสกี้ที่เข้ามาทางสถานีที่ 2 แสดงเวลาที่เข้ามา และคำนวณหาค่าอุณหภูมิและค่าความชื้นสัมพัทธ์



รูปที่ 4.16 การทำงานของโปรแกรมระบบประมวลผล เมื่อรับค่าจากสถานีที่ 2

เมื่อคำนวณค่าอุณหภูมิและค่าความชื้นสัมพัทธ์เสร็จ ค่าที่ได้จะเก็บฐานข้อมูลใน Microsoft Access ของแต่ละสถานี

Collect | Collect_1 | Collect_2

station	dat_mea	tim_mea	Temp_va	Humi_va
1	15/2/2552	22:50:03	25.12	36.27878
1	15/2/2552	22:52:46	25.1	35.96619
1	15/2/2552	22:53:13	25.06	36.34199
1	15/2/2552	22:53:28	25.06	36.34199
1	15/2/2552	22:53:43	25.06	36.47986
1	15/2/2552	22:54:42	25.06	36.20403
1	15/2/2552	22:55:09	25.06	36.13502
1	15/2/2552	22:55:33	25.06	36.27303
1	15/2/2552	22:55:48	25.06	39.69407
1	15/2/2552	22:56:18	28.28	90.13026
1	15/2/2552	22:56:33	27.39	54.72176
1	15/2/2552	23:09:08	25.12	36.10623
1	15/2/2552	23:09:22	25.1	36.00073
1	15/2/2552	23:10:16	25.03	36.20116
1	15/2/2552	23:10:49	25.06	36.20403
1	15/2/2552	23:11:04	25.06	36.23853
1	15/2/2552	23:11:24	25.1	37.48071
1	15/2/2552	23:11:42	25.12	40.03937
1	15/2/2552	23:12:18	25.25	38.35151
1	15/2/2552	23:14:35	25.12	35.93354
1	15/2/2552	23:15:02	25.1	35.135
1	15/2/2552	23:15:17	25.1	36.13884
1	15/2/2552	23:15:50	25.06	36.20403

ระเบียน: 1 จาก 135 | ไม่มีตัวกรอง | ค้นหา

รูปที่ 4.17 การเก็บฐานข้อมูลใน Microsoft Access ของสถานีที่ 1

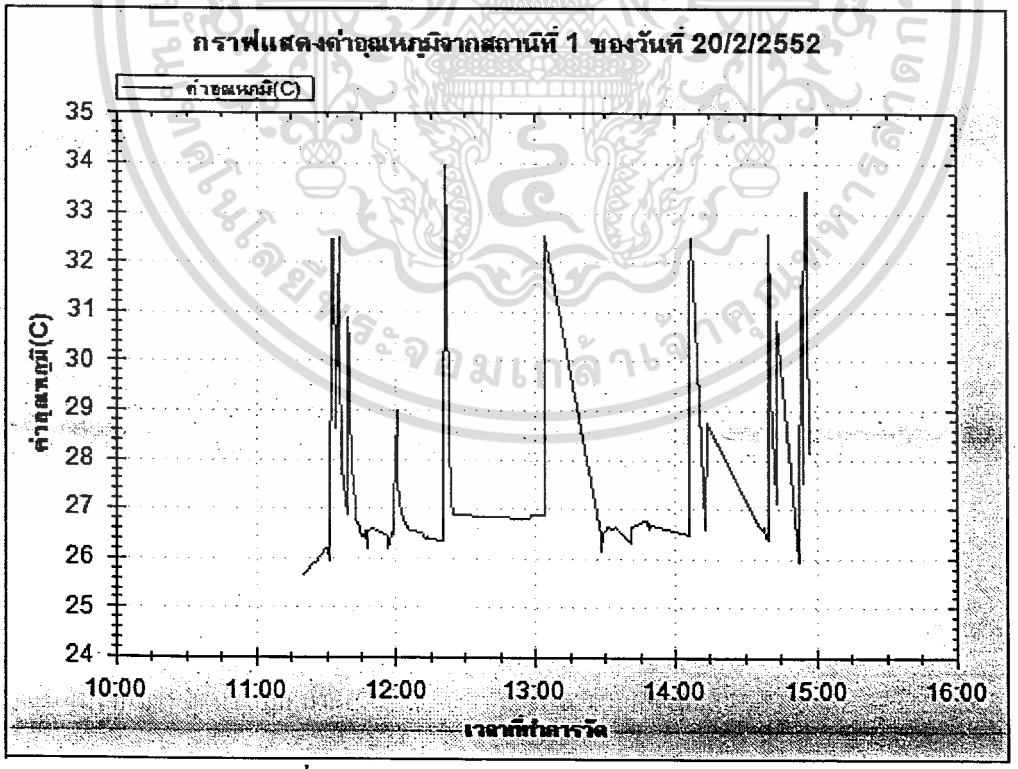
Collect | Collect_1 | Collect_2

station	dat_mea	tim_mea	Temp_va	Humi_va
2	15/2/2552	22:49:41	25.16	35.66074
2	15/2/2552	22:53:46	25.16	35.45304
2	15/2/2552	22:54:00	25.15	35.2153
2	15/2/2552	22:56:38	25.21	35.527
2	15/2/2552	23:11:54	25.06	36.51432
2	15/2/2552	23:12:22	25.21	45.53085
2	15/2/2552	23:12:36	25.3	35.74318
2	15/2/2552	23:15:30	25.16	81.46895
2	15/2/2552	23:15:46	25.16	35.24514
2	15/2/2552	23:16:39	25.16	81.46895
2	15/2/2552	23:16:55	25.16	35.1758
2	15/2/2552	23:17:23	25.16	35.14112
2	15/2/2552	23:18:03	25.16	34.96763
2	15/2/2552	23:20:02	25.16	34.96763
2	15/2/2552	23:20:56	25.1	35.03146
2	15/2/2552	23:21:11	25.12	34.96392
2	16/2/2552	2:54:00	30.84	63.78808
2	16/2/2552	2:55:35	30.84	63.87904
2	16/2/2552	2:56:41	30.7	64.03802
2	16/2/2552	2:57:35	30.7	63.91691
2	16/2/2552	2:59:08	30.66	63.9407
2	16/2/2552	2:59:23	30.66	63.91041
2	16/2/2552	2:59:51	30.66	63.88013

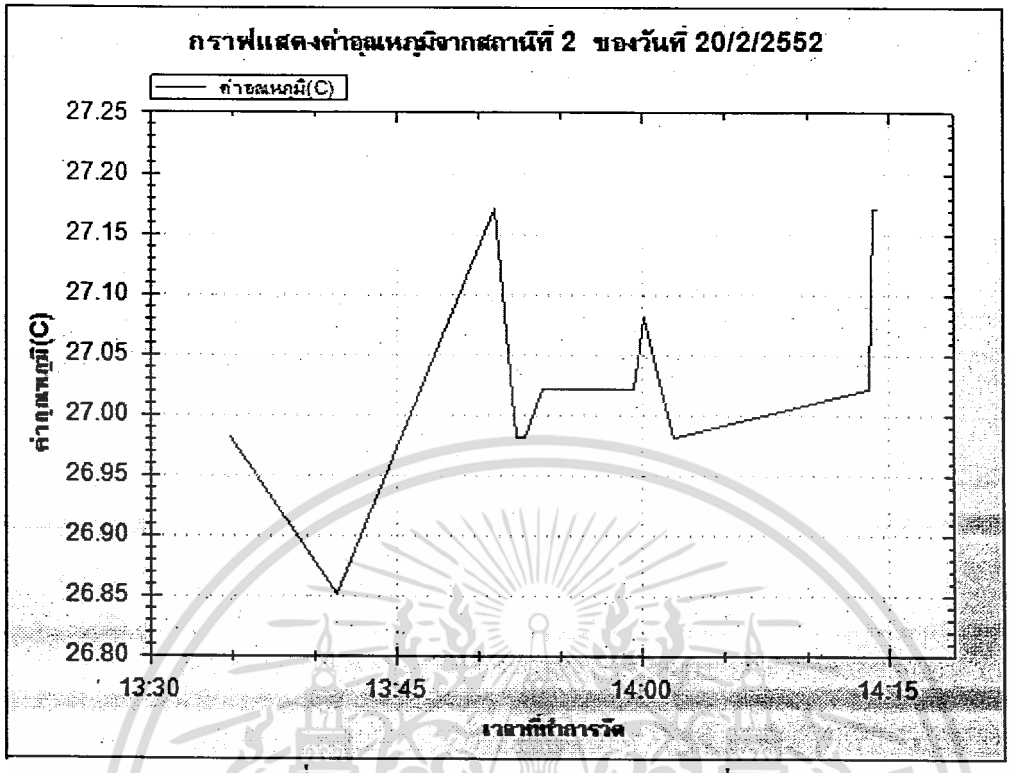
ระเบียน: 7 จาก 86 | ไม่มีตัวกรอง | ค้นหา

รูปที่ 4.18 การเก็บฐานข้อมูลใน Microsoft Access ของสถานีที่ 2

นำค่าอุณหภูมิที่เก็บในฐานข้อมูล มาแสดงผลเป็นกราฟของแต่ละสถานี

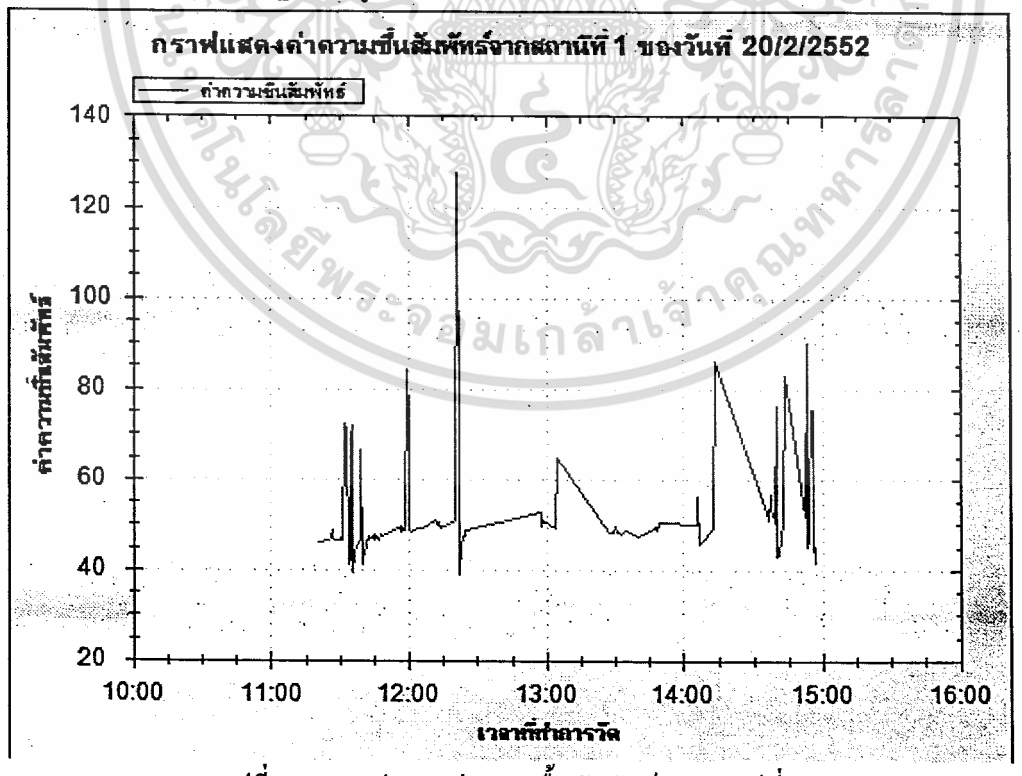


รูปที่ 4.19 กราฟแสดงค่าอุณหภูมิของสถานีที่ 1

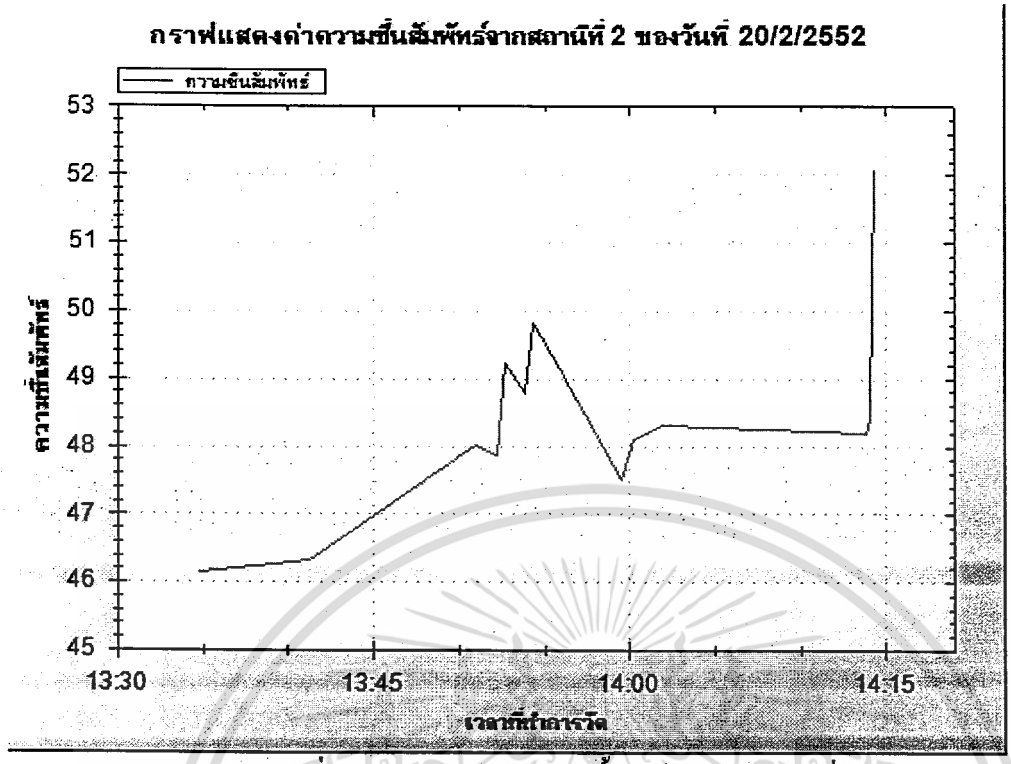


รูปที่ 4.20 กราฟแสดงค่าอุณหภูมิของสถานีที่ 2

นำค่าความชื้นสัมพัทธ์ที่เก็บในฐานข้อมูล มาแสดงผลเป็นกราฟของแต่ละสถานี



รูปที่ 4.21 กราฟแสดงค่าความชื้นสัมพัทธ์ของสถานีที่ 1



รูปที่ 4.22 กราฟแสดงค่าความชื้นสัมพัทธ์ของสถานีที่ 2

4.5 การสอบเทียบอุปกรณ์วัดอุณหภูมิและความชื้นสัมพัทธ์

อุปกรณ์ที่ใช้สอบเทียบมีชื่อว่า HANNA instrument สามารถวัดอุณหภูมิเป็นองศาเซลเซียสและความชื้นสัมพัทธ์เป็นเปอร์เซ็นต์

รูปที่ 4.23 อุปกรณ์สอบเทียบอุณหภูมิและความชื้นสัมพัทธ์

ตารางที่ 4.1 การสอบเทียบอุปกรณ์วัดอุณหภูมิและความชื้นสัมพัทธ์

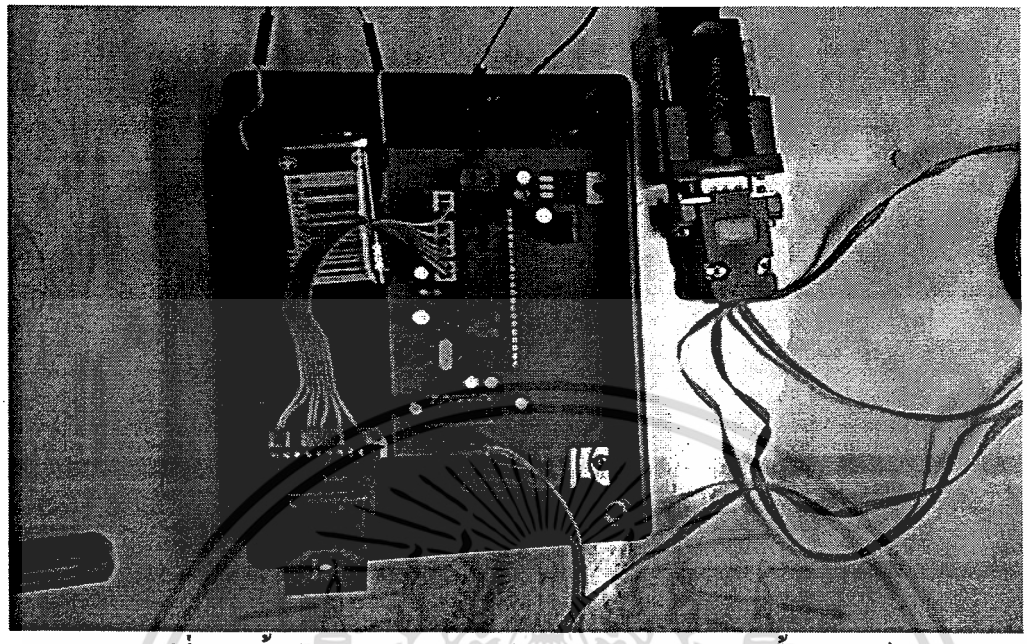
เวลาที่ใช้ในการสอบเทียบ	อุณหภูมิที่วัดได้จากเครื่อง Hunna Instrument (°C)	ความชื้นสัมพัทธ์จากเครื่อง Hunna Instrument (%)	อุณหภูมิที่วัดได้จากเซนเซอร์ SHT 15 (°C)	ความชื้นสัมพัทธ์ที่วัดจาก SHT15 (%)
9.35 am	27.5	74.7	28.04	74.71
9.46 am	27.3	74.2	28.5	76.65
9.47 am.	27.3	74.1	28.04	75.16
9.48 am.	27.3	74.3	28.04	74.86
9.49 am.	27.3	74.6	28	73.65

ตารางที่ 4.2 เปอร์เซ็นต์ความผิดพลาดของอุปกรณ์วัดอุณหภูมิและความชื้นสัมพัทธ์

เวลาที่ใช้ในการสอบเทียบ	เปอร์เซ็นต์ความผิดพลาดของอุณหภูมิ (%)	เปอร์เซ็นต์ความผิดพลาดของความชื้นสัมพัทธ์(%)
9.35 am	1.96	0.01
9.46 am	2.75	3.3
9.47 am.	2.71	1.43
9.48 am.	2.71	0.75
9.49 am.	2.56	0.07
ค่าเฉลี่ย	2.54	1.11

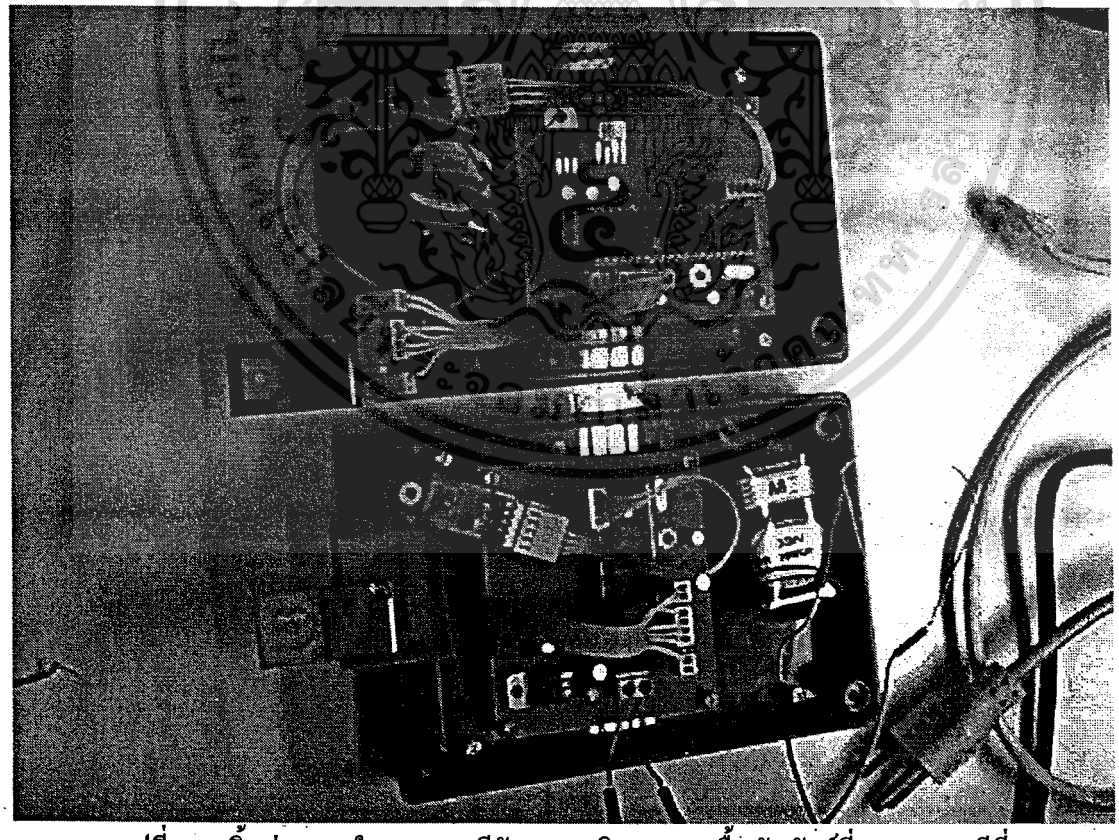
4.6 รูปถ่ายชิ้นงาน

อุปกรณ์สถานีรับที่รับค่าอุณหภูมิและความชื้นสัมพัทธ์ เป็นรหัสแอสกี้มาจาก สถานีส่งที่ 1 และ 2



รูปที่ 4.24 ชิ้นส่วนภายในของสถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์

อุปกรณ์สถานีส่งที่ 1 และ 2 ส่งค่าอุณหภูมิและความชื้นสัมพัทธ์ เป็นรหัสแอสกี้เข้าสถานีรับ



รูปที่ 4.25 ชิ้นส่วนภายในของสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์ที่ 1 และสถานีที่ 2

บทที่ 5
บทวิจารณ์และบทสรุป

สรุปผล

เครื่องตรวจจับอุณหภูมิและความชื้นแบบไร้สายสามารถทำงานได้เป็นผลสำเร็จ โดยสถานีส่งวัดอุณหภูมิและความชื้นสัมพัทธ์ สามารถวัดค่าและส่งมายังสถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์ได้ โดยข้อมูลที่สถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์เป็นข้อมูลที่ถูกต้องจากการเปรียบเทียบกับเครื่อง HANNA instrument ซึ่งเมื่อสถานีรับค่าอุณหภูมิและความชื้นสัมพัทธ์ส่งข้อมูลให้กับคอมพิวเตอร์เพื่อทำการประมวลผลด้วยโปรแกรมที่เขียนขึ้นสำหรับโครงงานนี้ โปรแกรมสามารถคำนวณหาค่าอุณหภูมิและความชื้นสัมพัทธ์พร้อมทั้งสามารถแสดงค่าเป็นแบบตัวเลข ตารางและกราฟ โดยกราฟสามารถแสดงค่าเวลาจริงในปัจจุบันที่บันทึก โดยข้อมูลที่รับมาทั้งหมดเก็บบันทึกผลเป็น Database และสามารถเรียกดูข้อมูลได้ตลอดเวลา

ปัญหาที่เกิดและการแก้ไข

ระยะทางที่สามารถทำการส่งข้อมูลแบบไร้สายยังอยู่ในระยะที่จำกัดและยังไม่สามารถกำหนดเวลาส่งตามที่กำหนดในส่วนของโปรแกรมได้ตรง

การพัฒนาต่อไป

พัฒนาให้รองรับสถานีวัดอุณหภูมิและความชื้นสัมพัทธ์ได้มากกว่า สอง สถานี

พัฒนาความสามารถของโปรแกรมการประมวลผลในคอมพิวเตอร์ให้สามารถทำฟังก์ชั่นอื่นๆได้ เช่น การแสดงข้อมูลจากฐานข้อมูลในหน้าเว็บเพจ

กิตติกรรมประกาศ

สำหรับโครงงานชิ้นนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ต้องขอขอบพระคุณ บิดา และมารดา ของนายชัยมงคล ฉันทานุกูล, นายอภิวิชญ์ อมรรัตนพันธ์, นายอะดุง สิงหพันธุ์ ซึ่งเป็นผู้ที่ให้กำเนิดอีกทั้งยังเป็นผู้คอยให้กำลังใจและคอยสนับสนุนเราในด้านต่างๆเรื่อยมา ขอขอบพระคุณ อ. สิรภพ ตู้ประกาย เป็นอย่างสูงที่คอยให้ความช่วยเหลือทั้งในด้านคำปรึกษา คำแนะนำ รวมทั้งอุปกรณ์ต่างๆในการทำงานตลอดมา

ขอขอบคุณ นางสาวพัชรินทร์ อินโสม, นายคมสันติ์ เลิศบุญยพันธุ์, นายวิริยังค์ พิพัฒน์สกุลโรจน์ เพื่อนๆทุกคนที่ได้ช่วยเหลือ และเป็นกำลังใจให้เสมอมา ขอขอบคุณหนังสือ และตำราต่างๆที่ช่วยให้เราเข้าใจอะไรๆอีกมากมาย และสุดท้าย ขอบคุณโครงงานชิ้นนี้ที่ช่วยสอนอะไรเรามากมายเหลือเกิน สิ่งที่ได้รับจะไม่เป็นเพียงความรู้ทางทฤษฎี แต่จะช่วยให้เราเติบโตขึ้นทั้งทางความคิดและการกระทำ

นายชัยมงคล ฉันทานุกูล
นายอภิวิชญ์ อมรรัตนพันธ์
นายอะดุง สิงหพันธุ์
ผู้จัดทำ

เอกสารอ้างอิง

ศุภชัย สมพานิช . คู่มือการเขียนโปรแกรม Visual C# .NET ฉบับโปรแกรมเมอร์ . ครั้งที่ 1 . นนทบุรี : อินโฟเพรส,2546

สมยศ จุณณะปิยะ .การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51

ไชยวัฒน์ ตระการรัตน์สันติ .Microsoft Access 2000 : สำนักพิมพ์ widebase/จุลภาค ,2550

ภาคผนวก

Program

- Assembly
- Visual C#.NET

```
;*****************************************
;       program for SLAVE 1             ; การกำหนดค่าของขาในสถานีที่ 1
;*****************************************
CE      BIT     P1.2
CS      BIT     P1.3
CLK     BIT     P1.4
DAT     BIT     P1.5
DR1     BIT     P1.6
LDATA           BIT P2.1
LCLK            BIT P2.2
CRESET          EQU 1EH
M_TEMP              EQU 03H
M_HUMI              EQU 05H
COUNTR              EQU 40H
ERROR           EQU 41H
COMMAND             EQU 42H
VALUE           EQU 43H
TEMP_M              EQU 44H
TEMP_L          EQU 45H
HUMI_M              EQU 46H
HUMI_L          EQU 47H

SLAVE1_T1       EQU 30H
SLAVE1_T2       EQU 31H
SLAVE1_T3       EQU 32H
SLAVE1_T4       EQU 33H
SLAVE1_H1       EQU 34H
SLAVE1_H2       EQU 35H
SLAVE1_H3       EQU 36H
SLAVE1_H4       EQU 37H
```

```
        ORG     0000H
;****************************************
                MOV     SCON,#50H               ; การทำงานของส่วนหลัก
                MOV     PCON,#00H
                MOV     TMOD,#20H
                MOV     TH1,#0FDH
                SETB    TR1
        MOV     P2,#0FFH
MAIN:           MOV     SLAVE1_T1,#00H
                MOV     SLAVE1_T2,#00H
                MOV     SLAVE1_T3,#00H
                MOV     SLAVE1_T4,#00H
                MOV     SLAVE1_H1,#00H
                MOV     SLAVE1_H2,#00H
                MOV     SLAVE1_H3,#00H
                MOV     SLAVE1_H4,#00H
                CALL    DELAY_1s
                CALL    S_CONRESET
                MOV     ERROR,#00
                MOV     COMMAND,#M_TEMP
                CALL    S_COMMAND
                CALL    S_READ
                MOV     TEMP_M,VALUE
                CALL    S_READ
                MOV     TEMP_L,VALUE
                CALL    DELAY_1s
                MOV     COMMAND,#M_HUMI
                CALL    S_COMMAND
                CALL    S_READ
                MOV     HUMI_M,VALUE
                CALL    S_READ
                MOV     HUMI_L,VALUE
                MOV     A,ERROR
                CJNE    A,#00H,MAIN
```

```
        CALL    INIT
        CALL    SETMODE_TX
        MOV             A,#0AH          ;       ACK TO RX = START BYTE
        CALL    SEND_TRW
        MOV     A,TEMP_M
        CALL    SEND_TRW
        MOV     A,TEMP_L
        CALL    SEND_TRW
        MOV     A,HUMI_M
        CALL    SEND_TRW
        MOV     A,HUMI_L
        CALL    SEND_TRW
        CALL    DELAY_10s               ;               DELAY FOR 10 sec
        LJMP    MAIN
;*****************************************************
;  CONNECTION RESET
;*****************************************************
S_CONRESET:     SETB    LDATA
                CLR     LCLK
                MOV     COUNTR,#9
AA:             SETB    LCLK
                NOP
                CLR     LCLK
                DJNZ    COUNTR,AA
                LCALL   T_START
                RET
;*****************************************************
;  TRANSMITTION START
;*****************************************************
T_START:
                SETB    LDATA
                NOP
                CLR     LCLK
                NOP
```

```
                SETB    LCLK
                NOP
                CLR     LDATA
                NOP
                CLR     LCLK
                NOP
                SETB    LCLK
                NOP
                SETB    LDATA
                NOP
                CLR     LCLK
                NOP
                RET
;*****************************************************
;  COMMAND WRITE LOOP
;*****************************************************
S_COMMAND:      MOV     ERROR,#00
                LCALL   T_START
                MOV     A,COMMAND
SHT15WD:        MOV     R7,#8
                CLR     C
SHT15WD1:       RLC     A
                MOV     LDATA,C
                SETB    LCLK
                NOP
                NOP
                NOP
                CLR     LCLK
                DJNZ    R7,SHT15WD1
                SETB    LDATA
                SETB    LCLK
                JNB     LDATA,BB
                INC     ERROR
BB:             CLR     LCLK
```

```
                MOV     R0,#00
                MOV     A,#00
M_WAIT:         JNB     LDATA,OUT1
                INC     A
                NOP
                NOP
                NOP
                NOP
                NOP
                CJNE    A,#0FFH,M_WAIT
                MOV     A,#00H
                INC     R0
                CJNE    R0,#0FFH,M_WAIT
                JNB     LDATA,OUT1
                INC     ERROR
OUT1:           RET
;***********************************************
S_READ:         MOV     VALUE,#00H
                SETB    LDATA
                SETB    LCLK
                JNB     LDATA,B7
                ORL     VALUE,#80H
B7:             CLR     LCLK
                NOP
                SETB    LCLK
                JNB     LDATA,B6
                ORL     VALUE,#40H
B6:             CLR     LCLK
                NOP
                SETB    LCLK
                JNB     LDATA,B5
                ORL     VALUE,#20H
B5:             CLR     LCLK
                NOP
```

```
        SETB    LCLK
        JNB     LDATA,B4
        ORL     VALUE,#10H
B4:     CLR     LCLK
        NOP
        SETB    LCLK
        JNB     LDATA,B3
        ORL     VALUE,#08H
B3:     CLR     LCLK
        NOP
        SETB    LCLK
        JNB     LDATA,B2
        ORL     VALUE,#04H
B2:     CLR     LCLK
        NOP
        SETB    LCLK
        JNB     LDATA,B1
        ORL     VALUE,#02H
B1:     CLR     LCLK
        NOP
        SETB    LCLK
        JNB     LDATA,B0
        ORL     VALUE,#01H
B0:     CLR     LCLK
        NOP
        CLR     LDATA
        SETB    LCLK
        NOP
        CLR     LCLK
        SETB    LDATA
        RET
;************************************************
INIT:   CALL    DELAY_100ms
        CLR     CE
```

```
        CLR     CS
        CLR     DAT
        CLR     CLK
        CALL    DELAY_100ms
        RET
;*********************************************
;FUNCTION CONFIG FOR TRW TX
;*********************************************************************
SETMODE_TX:  CALL    DELAY_100ms
             CLR     CE
             SETB    CS              ; ENABLE SETTING
             CLR     A
             MOV     R1,#18          ; 18*8 = 144 bit
SETMODE_TX_0:        MOV     DPTR,#CONFIG_TEST     ; Set Header
             PUSH    ACC
             MOVC    A,@A+DPTR
             CALL    WRITE_TRW24
             POP     ACC
             INC     A
             DJNZ    R1,SETMODE_TX_0
             SETB    DAT
             SETB    DR1
             SETB    CE
             CLR     CS
             CALL    DELAY_100ms
             RET
;*********************************************************************
;FUNCTION CONFIG FOR TRW RX
;*********************************************************************
SETMODE_RX:  CALL    DELAY_100ms
             CLR     CE
             SETB    CS
             CLR     A
             MOV     R1,#18                ;18*8 = 144 bit
```

```
SETMODE_RX_0:   MOV   DPTR,#CONFIG_TEST_RX      ;SET HEADER
                PUSH  ACC
                MOVC  A,@A+DPTR
                CALL  WRITE_TRW24
                POP   ACC
                INC   A
                DJNZ  R1,SETMODE_RX_0
                SETB  DAT
                CLR   DR1
                SETB  CE
                CLR   CS
                CALL  DELAY_100ms
                RET
;*********************************************************************
;FUNCTION SEND ADDRESS+DATA TO RECEIVER
;*********************************************************************
SEND_TRW:       CALL  DELAY_100ms
                CLR   CS
                SETB  CE                        ;ENABLE SEND MODE
                PUSH  ACC                       ;SAVE DATA BEFOR
                CLR   A
                MOV   R1,#5                     ;5 BYTE ADDR 4 DATA 1
SEND_TRW_0:     MOV   DPTR,#CONFIG_ADDR1        ;ADDR
                PUSH  ACC
                MOVC  A,@A+DPTR
                CALL  WRITE_TRW24               ;WRITE LOOP
                POP   ACC
                INC   A
                DJNZ  R1,SEND_TRW_0
                POP   ACC                       ;DATA
                CALL  WRITE_TRW24
                CALL  DELAY_100ms
                CLR   CLK
                CLR   CE
```

```
                CLR     DAT
                CALL    DELAY_100ms
                RET
;***********************************************
CLK_TRW:        CLR     CLK
                CALL    DELAY_1ms
                SETB    CLK
                CALL    DELAY_1ms
                RET
;*************************************************************************
;WRITE 8 BIT TO DATA PIN
;*************************************************************************
WRITE_TRW24:    MOV     R0,#8
WRITE_TRW24_0:          JB      ACC.7,WRITE1
                CLR     DAT
                JMP     WRITE_TRW2
WRITE1:                 SETB    DAT
WRITE_TRW2:     CALL    CLK_TRW
                RL      A
                DJNZ    R0,WRITE_TRW24_0
                RET
;*************************************************************************
;READ 8 BIT DATA FROM TRW
;*************************************************************************
READ_TRW24:     CLR     A
                MOV     R0,#8
READ_TRW24_0:   RL      A
                SETB    CLK
                CALL    DELAY_1ms
                JB      DAT,READ_1
                CLR     ACC.0
                JMP     READ_TRW24_1
READ_1:                 SETB    ACC.0
READ_TRW24_1:   CLR     CLK
```

```
        CALL    DELAY_1ms
        DJNZ    R0,READ_TRW24_0
        RET
;*************************************************************************
;DELAY TIME 1s
;*************************************************************************
DELAY_1s:       MOV     R5,#0AH
DELAY_1s_1:     CALL    DELAY_100ms
                DJNZ    R5,DELAY_1s_1
                RET

;*************************************************************************
;DELAY TIME 10s
;*************************************************************************
DELAY_10s:      CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        CALL    DELAY_1s
                        RET
;*****************************
DELAY_1ms:      MOV     R6,#0E6H
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6,DELAY_1ms_1
                RET
DELAY_100ms:    MOV     R7,#100
DELAY_100ms_1:          MOV     R6,#0E6H
DELAY_100ms_2:          NOP
```

```
        NOP
        DJNZ    R6,DELAY_100ms_2
        DJNZ    R7,DELAY_100ms_1
        RET
;*************************************
CONFIG_TEST_RX:         DB      8EH,08H,1CH
CONFIG_LEN2_RX:         DB      08H
CONFIG_LEN1_RX:         DB      08H
CONFIG_ADDR2_RX:    DB      0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1_RX:    DB      0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR_RX: DB       0A3H
CONFIG_RF_RX:       DB      6FH
CONFIG_CH_RX:       DB      0BH
CONFIG_TEST:        DB      8EH,08H,1CH
CONFIG_LEN2:        DB      08H
CONFIG_LEN1:        DB      08H
CONFIG_ADDR2:           DB      0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1:           DB      0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR:         DB      0A3H
CONFIG_RF:          DB      6FH
CONFIG_CH:          DB      0AH
END
```

```
;*****************************************
;       program for SLAVE 2
;*****************************************
CE      BIT     P1.2
CS      BIT     P1.3
CLK     BIT     P1.4
DAT     BIT     P1.5
DR1     BIT     P1.6
LDATA           BIT P2.1
LCLK            BIT P2.2
CRESET          EQU 1EH
M_TEMP                  EQU 03H
M_HUMI                  EQU 05H
COUNTR                  EQU 40H
ERROR           EQU 41H
COMMAND                 EQU 42H
VALUE           EQU 43H
TEMP_M                  EQU 44H
TEMP_L          EQU 45H
HUMI_M                  EQU 46H
HUMI_L          EQU 47H
SLAVE2_T1       EQU 30H
SLAVE2_T2       EQU 31H
SLAVE2_T3       EQU 32H
SLAVE2_T4       EQU 33H
SLAVE2_H1       EQU 34H
SLAVE2_H2       EQU 35H
SLAVE2_H3       EQU 36H
SLAVE2_H4       EQU 37H
        ORG     0000H
```

```
;*****************************************
MOV     SCON,#50H
MOV     PCON,#00H
MOV     TMOD,#20H
MOV     TH1,#0FDH
SETB    TR1
MOV     P2,#0FFH
MAIN:   CALL    DELAY_10s               ;               WAIT FOR SLAVE1
MOV     SLAVE2_T1,#00H
MOV     SLAVE2_T2,#00H
MOV     SLAVE2_T3,#00H
MOV     SLAVE2_T4,#00H
MOV     SLAVE2_H1,#00H
MOV     SLAVE2_H2,#00H
MOV     SLAVE2_H3,#00H
MOV     SLAVE2_H4,#00H
CALL    DELAY_1s
CALL    S_CONRESET
MOV     ERROR,#00
MOV     COMMAND,#M_TEMP
CALL    S_COMMAND
CALL    S_READ
MOV     TEMP_M,VALUE
CALL    S_READ
MOV     TEMP_L,VALUE
CALL    DELAY_1s
COMMAND,#M_HUMI
CALL    S_COMMAND
CALL    S_READ
MOV     HUMI_M,VALUE
CALL    S_READ
MOV     HUMI_L,VALUE
MOV     A,ERROR
CJNE    A,#00H,MAIN
```

```
CALL   INIT
CALL   SETMODE_TX
MOV    A,TEMP_M
CALL   SEND_TRW
MOV    A,TEMP_L
CALL   SEND_TRW
MOV    A,HUMI_M
CALL   SEND_TRW
MOV    A,HUMI_L
CALL   SEND_TRW
LJMP   MAIN
;*****************************************************
;  CONNECTION RESET
;*****************************************************
S_CONRESET:   SETB   LDATA
CLR    LCLK
MOV    COUNTR,#9
SETB   LCLK
NOP
CLR    LCLK
DJNZ   COUNTR,AA
LCALL  T_START
RET
;*****************************************************
;  TRANSMITTION START
;*****************************************************
T_START:
SETB   LDATA
NOP
CLR    LCLK
NOP
SETB   LCLK
NOP
CLR    LDATA
```

```
NOP
CLR     LCLK
NOP
SETB    LCLK
NOP
SETB    LDATA
NOP
NOP
RET
;*****************************************************
; COMMAND WRITE LOOP
;*****************************************************
S_COMMAND:   MOV     ERROR,#00
LCALL  T_START
MOV     A,COMMAND
MOV     R7,#8
CLR     C
SHT15WD1:       RLC
MOV     LDATA,C
SETB    LCLK
NOP
NOP
NOP
CLR     LCLK
DJNZ    R7,SHT15WD1
SETB    LDATA
SETB    LCLK
JNB     LDATA,BB
INC     ERROR
BB:             CLR     LCLK
MOV     R0,#00
MOV     A,#00
M_WAIT:                 JNB     LDATA,OUT1
INC     A
```

```
NOP
NOP
NOP
NOP
NOP
CJNE    A,#0FFH,M_WAIT
MOV     A,#00H
INC     R0
CJNE    R0,#0FFH,M_WAIT
JNB     LDATA,OUT1
INC     ERROR
OUT1:           RET
;************************************************
S_READ:                 MOV     VALUE,#00H
SETB    LDATA
SETB    LCLK
JNB     LDATA,B7
ORL     VALUE,#80H
B7:             CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B6
ORL     VALUE,#40H
CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B5
ORL     VALUE,#20H
B5:             CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B4
ORL     VALUE,#10H
CLR     LCLK
```

```
NOP
SETB    LCLK
JNB     LDATA,B3
ORL     VALUE,#08H
B3:             CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B2
ORL     VALUE,#04H
CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B1
ORL     VALUE,#02H
CLR     LCLK
NOP
SETB    LCLK
JNB     LDATA,B0
ORL     VALUE,#01H
CLR     LCLK
NOP
CLR     LDATA
SETB    LCLK
NOP
CLR     LCLK
SETB    LDATA
RET
;************************************************
INIT:   CALL    DELAY_100ms
CLR     CE
CLR     CS
CLR     DAT
CLR     CLK
CALL    DELAY_100ms
```

```
RET
;*********************************************
;FUNCTION CONFIG FOR TRW TX
;*****************************************************************************
SETMODE_TX:  CALL   DELAY_100ms
CLR    CE
SETB   CS              ; ENABLE SETTING
CLR    A
MOV    R1,#18          ; 18*8 = 144 bit
SETMODE_TX_0:       MOV    DPTR,#CONFIG_TEST     ; Set Header
PUSH   ACC
MOVC   A,@A+DPTR
CALL   WRITE_TRW24
POP    ACC
INC    A
DJNZ   R1,SETMODE_TX_0
SETB   DAT
SETB   DR1
SETB   CE
CLR    CS
CALL   DELAY_100ms
;*****************************************************************************
;FUNCTION CONFIG FOR TRW RX
;*****************************************************************************
SETMODE_RX:  CALL   DELAY_100ms
CLR    CE
SETB   CS
CLR    A
MOV    R1,#18                ;18*8 = 144 bit
SETMODE_RX_0:       MOV    DPTR,#CONFIG_TEST_RX          ;SET HEADER
PUSH   ACC
MOVC   A,@A+DPTR
CALL   WRITE_TRW24
POP    ACC
```

```
INC     A
DJNZ    R1,SETMODE_RX_0
SETB    DAT
CLR     DR1
SETB    CE
CALL    DELAY_100ms
RET
;*************************************************************************
;FUNCTION SEND ADDRESS+DATA TO RECEIVER
;*************************************************************************
SEND_TRW:       CALL    DELAY_100ms
CLR     CS
SETB    CE                      ;ENABLE SEND MODE
PUSH    ACC                     ;SAVE DATA BEFOR
CLR     A
MOV     R1,#5                   ;5 BYTE ADDR 4 DATA 1
MOV     DPTR,#CONFIG_ADDR1      ;ADDR
PUSH    ACC
MOVC    A,@A+DPTR
CALL    WRITE_TRW24             ;WRITE LOOP
POP     ACC
INC     A
DJNZ    R1,SEND_TRW_0
POP     ACC                     ;DATA
CALL    WRITE_TRW24
CALL    DELAY_100ms
CLR     CLK
CLR     CE
CLR     DAT
CALL    DELAY_100ms
RET
;*********************************************
CLK_TRW:        CLR     CLK
CALL    DELAY_1ms
```

```
SETB    CLK
CALL    DELAY_1ms
RET
;*************************************************************************
;WRITE 8 BIT TO DATA PIN
;*************************************************************************
        WRITE_TRW24: MOV    R0,#8
WRITE_TRW24_0:       JB     ACC.7,WRITE1
CLR     DAT
        JMP     WRITE_TRW2
WRITE1:              SETB   DAT
WRITE_TRW2:  CALL    CLK_TRW
RL      A
DJNZ    R0,WRITE_TRW24_0
RET
;*************************************************************************
;READ 8 BIT DATA FROM TRW
;*************************************************************************
READ_TRW24:  CLR     A
             MOV     R0,#8
READ_TRW24_0:RL      A
SETB    CLK
CALL    DELAY_1ms
JB      DAT,READ_1
CLR     ACC.0
JMP     READ_TRW24_1
        SETB    ACC.0
READ_TRW24_1:CLR     CLK
CALL    DELAY_1ms
DJNZ    R0,READ_TRW24_0
RET
;*************************************************************************
;DELAY TIME 1s
;*************************************************************************
```

```
DELAY_1s:       MOV     R5,#0AH
DELAY_1s_1:     CALL    DELAY_100ms
DJNZ    R5,DELAY_1s_1
RET
;*************************************************************************
;DELAY TIME 10s
;*************************************************************************
DELAY_10s:      CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
CALL    DELAY_1s
RET
;*********************************
DELAY_1ms:      MOV     R6,#0E6H
DELAY_1ms_1:    NOP
NOP
DJNZ    R6,DELAY_1ms_1
RET
DELAY_100ms:    MOV     R7,#100
        DELAY_100ms_1:          MOV     R6,#0E6H
DELAY_100ms_2:          NOP
NOP
DJNZ    R6,DELAY_100ms_2
DJNZ    R7,DELAY_100ms_1
        RET
;*************************************
CONFIG_TEST_RX:                 DB      8EH,08H,1CH
CONFIG_LEN2_RX:                 DB      08H
```

```
CONFIG_LEN1_RX:          DB    08H
CONFIG_ADDR2_RX:    DB    0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1_RX:    DB    0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR_RX: DB    0A3H
CONFIG_RF_RX:       DB    6FH
CONFIG_CH_RX:       DB    0BH
CONFIG_LEN2:        DB    08H
CONFIG_LEN1:        DB    08H
CONFIG_ADDR2:            DB    0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1:            DB    0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR:          DB    0A3H
CONFIG_RF:          DB    6FH
CONFIG_CH:          DB    0AH
END
```

```
;*****************************************
;       program for MASTER
;*****************************************
CE      BIT     P1.2
CS      BIT     P1.3
CLK     BIT     P1.4
DAT     BIT     P1.5
DR1     BIT     P1.6
        TEMP_M1             EQU 30H
        TEMP_L1             EQU 31H
        HUMI_M1             EQU 32H
        HUMI_L1             EQU 33H
        TEMP_M2             EQU 34H
        TEMP_L2             EQU 35H
        HUMI_M2             EQU 36H
        HUMI_L2             EQU 37H
        CHECK1          EQU 38H
        CHECK2          EQU 39H
        SLAVE1_T1       EQU 40H
        SLAVE1_T2       EQU 41H
        SLAVE1_T3       EQU 42H
        SLAVE1_T4       EQU 43H
SLAVE1_H1   EQU 44H
        SLAVE1_H2       EQU 45H
        SLAVE1_H3       EQU 46H
        SLAVE1_H4       EQU 47H
        SLAVE2_T1       EQU 48H
        SLAVE2_T2       EQU 49H
        SLAVE2_T3       EQU 4AH
        SLAVE2_T4       EQU 4BH
SLAVE2_H1   EQU 4CH
        SLAVE2_H2       EQU 4DH
        SLAVE2_H3       EQU 4EH
        SLAVE2_H4       EQU 4FH
```

```
        ORG     0000H
;****************************************
        MOV     SCON,#50H
        MOV     PCON,#00H
        MOV     TMOD,#20H
        MOV     TH1,#0FDH
        SETB    TR1
        MOV     P2,#0FFH
;*************************************
;       MAIN RX
;*************************************
MAIN:   MOV     TEMP_M1,#00H
        MOV     TEMP_L1,#00H
        MOV     HUMI_M1,#00H
        MOV     HUMI_L1,#00H
        MOV     TEMP_M2,#00H
        MOV     TEMP_L2,#00H
        MOV     HUMI_M2,#00H
        MOV     HUMI_L2,#00H
        MOV     CHECK1,#00H
        MOV     CHECK2,#00H
        MOV     SLAVE1_T1,#00H
        MOV     SLAVE1_T2,#00H
        MOV     SLAVE1_T3,#00H
        MOV     SLAVE1_T4,#00H
        MOV     SLAVE1_H1,#00H
        MOV     SLAVE1_H2,#00H
        MOV     SLAVE1_H3,#00H
        MOV     SLAVE1_H4,#00H
        MOV     SLAVE2_T1,#00H
        MOV     SLAVE2_T2,#00H
        MOV     SLAVE2_T3,#00H
        MOV     SLAVE2_T4,#00H
        MOV     SLAVE2_H1,#00H
```

```
                MOV     SLAVE2_H2,#00H
                MOV     SLAVE2_H3,#00H
                MOV     SLAVE2_H4,#00H
        CALL    DELAY_1ms               ; DELAY FOR VDD OFF TO STANDBY
                CALL    INIT
                CALL    SETMODE_RX
                JNB             DR1,$
                CALL    READ_TRW24
        CJNE    A,#0AH,SLAVE2           ;       CHECK WHICH SLAVE ARE THEY FROM?
                MOV     CHECK1,#00000001B
                CALL    SAVE1                           ; SAVE SLAVE1 DATA
                CALL    CONVERT1
                CALL    TO_COM1
                LJMP    MAIN
SLAVE2:         CJNE    A,#0A0H,MAIN            ;       CHECK WHICH SLAVE ARE THEY FROM?
        MOV     CHECK2,#00000001B
                CALL    SAVE2                           ; SAVE SLAVE1 DATA
                CALL    CONVERT2
                CALL    TO_COM2
                LJMP    MAIN
;*********************************************************************
; SAVE SLAVE#1 DATA
;*********************************************************************
SAVE1:          JNB     DR1,$
                CALL    READ_TRW24
                MOV     TEMP_M1,A
                JNB     DR1,$
                CALL    READ_TRW24
                MOV     TEMP_L1,A
                JNB     DR1,
                CALL    READ_TRW24
                MOV     HUMI_M1,A
                JNB     DR1,$
                CALL    READ_TRW24
```

```
                MOV     HUMI_L1,A
                RET
;*********************************************************************
; SEND SLAVE#1 DATA TO COMPUTER
;*********************************************************************
TO_COM1:        MOV     A,#23H
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_T1
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_T2
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_T3
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_T4
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_H1
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_H2
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_H3
```

```
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,SLAVE1_H4
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                MOV     A,#23H
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                RET
;*************************************************************************
; SAVE SLAVE#2 DATA
;*************************************************************************
SAVE2:          JNB     DR1,$
                CALL    READ_TRW24
                MOV     TEMP_M2,A
                JNB     DR1,$
                CALL    READ_TRW24
                MOV     TEMP_L2,A
                JNB     DR1,$
                CALL    READ_TRW24
                MOV     HUMI_M2,A
                JNB     DR1,$
                CALL    READ_TRW24
                MOV     HUMI_L2,A
                RET
;*************************************************************************
; SEND SLAVE#2 DATA TO COMPUTER
;*************************************************************************
TO_COM2:        MOV     A,#2AH
                MOV     SBUF,A
                JNB     TI,$
```

```
CLR     TI
MOV     A,SLAVE2_T1
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_T2
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_T3
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_T4
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_H1
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_H2
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_H3
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,SLAVE2_H4
MOV     SBUF,A
JNB     TI,$
CLR     TI
MOV     A,#2AH
```

```
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                RET
;*****************************************
; CONVERT SLAVE#1 DATA TO ASCII
;*****************************************
CONVERT1:       MOV     A,TEMP_M1       ; M2 HIGH TO T1
                SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
                CLR     ACC.7
                CLR     ACC.6
                CLR     ACC.5
                CLR     ACC.4
                CALL    CONVERT
                MOV     SLAVE1_T1,A
                MOV     A,TEMP_M1       ; M2 LOW TO T2
                CLR     ACC.7
                CLR     ACC.6
                CLR     ACC.5
                CLR     ACC.4
                CALL    CONVERT
                MOV     SLAVE1_T2,A
                MOV     A,TEMP_L1       ; L2 HIGH TO T3
                SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
                CLR     ACC.7
                CLR     ACC.6
                CLR     ACC.5
                CLR     ACC.4
                CALL    CONVERT
                MOV     SLAVE1_T3,A
                MOV     A,TEMP_L1       ; L2 LOW TO T4
                CLR     ACC.7
                CLR     ACC.6
                CLR     ACC.5
```

```
CLR     ACC.4
CALL    CONVERT
MOV     SLAVE1_T4,A
MOV     A,HUMI_M1       ; M2 HIGH TO H1
SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
CLR     ACC.7
CLR     ACC.6
CLR     ACC.5
CLR     ACC.4
CALL    CONVERT
MOV     SLAVE1_H1,A
MOV     A,HUMI_M1       ; M2 LOW TO H2
CLR     ACC.7
CLR     ACC.6
CLR     ACC.5
CLR     ACC.4
CALL    CONVERT
MOV     SLAVE1_H2,A
MOV     A,HUMI_L1       ; L2 HIGH TO H3
SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
CLR     ACC.7
CLR     ACC.6
CLR     ACC.5
CLR     ACC.4
CALL    CONVERT
MOV     SLAVE1_H3,A
MOV     A,HUMI_L1       ; L2 LOW TO H4
CLR     ACC.7
CLR     ACC.6
CLR     ACC.5
CLR     ACC.4
CALL    CONVERT
MOV     SLAVE1_H4,A
RET
```

```
;*******************************************
; CONVERT SLAVE#2 DATA TO ASCII
;*******************************************
CONVERT2:   MOV   A,TEMP_M2     ; M2 HIGH TO T1
            SWAP  A             ; HIGH NIBBLE TO LOW NIBBLE
            CLR   ACC.7
            CLR   ACC.6
            CLR   ACC.5
            CLR   ACC.4
            CALL  CONVERT
            MOV   SLAVE2_T1,A
            MOV   A,TEMP_M2     ; M2 LOW TO T2
            CLR   ACC.7
            CLR   ACC.6
            CLR   ACC.5
            CLR   ACC.4
            CALL  CONVERT
            MOV   SLAVE2_T2,A
            MOV   A,TEMP_L2     ; L2 HIGH TO T3
            SWAP  A             ; HIGH NIBBLE TO LOW NIBBLE
            CLR   ACC.7
            CLR   ACC.6
            CLR   ACC.5
            CLR   ACC.4
            CALL  CONVERT
            MOV   SLAVE2_T3,A
            MOV   A,TEMP_L2     ; L2 LOW TO T4
            CLR   ACC.7
            CLR   ACC.6
            CLR   ACC.5
            CLR   ACC.4
            CALL  CONVERT
            MOV   SLAVE2_T4,A
            MOV   A,HUMI_M2     ; M2 HIGH TO H1
```

```
        SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
        CLR     ACC.7
        CLR     ACC.6
        CLR     ACC.5
        CLR     ACC.4
        CALL    CONVERT
        MOV     SLAVE2_H1,A
        MOV     A,HUMI_M2       ; M2 LOW TO H2
        CLR     ACC.7
        CLR     ACC.6
        CLR     ACC.5
        CLR     ACC.4
        CALL    CONVERT
        MOV     SLAVE2_H2,A
        MOV     A,HUMI_L2       ; L2 HIGH TO H3
        SWAP    A               ; HIGH NIBBLE TO LOW NIBBLE
        CLR     ACC.7
        CLR     ACC.6
        CLR     ACC.5
        CLR     ACC.4
        CALL    CONVERT
        MOV     SLAVE2_H3,A

        MOV     A,HUMI_L2       ; L2 LOW TO H4
        CLR     ACC.7
        CLR     ACC.6
        CLR     ACC.5
        CLR     ACC.4
        CALL    CONVERT
        MOV     SLAVE2_H4,A
        RET
;*********************************************************
CONVERT:    CJNE    A,#00H,AS1
            MOV     A,#30H
```

```
        RET
AS1:    CJNE    A,#01H,AS2
        MOV     A,#31H
        RET
AS2:    CJNE    A,#02H,AS3
        MOV     A,#32H
        RET
AS3:    CJNE    A,#03H,AS4
        MOV     A,#33H
        RET
AS4:    CJNE    A,#04H,AS5
        MOV     A,#34H
        RET
AS5:    CJNE    A,#05H,AS6
        MOV     A,#35H
        RET
AS6:    CJNE    A,#06H,AS7
        MOV     A,#36H
        RET
AS7:    CJNE    A,#07H,AS8
        MOV     A,#37H
        RET
AS8:    CJNE    A,#08H,AS9
        MOV     A,#38H
        RET
AS9:    CJNE    A,#09H,AS10
        MOV     A,#39H
        RET
AS10:   CJNE    A,#0AH,AS11
        MOV     A,#41H
        RET
AS11:   CJNE    A,#0BH,AS12
        MOV     A,#42H
        RET
```

```
AS12:       CJNE    A,#0CH,AS13
            MOV     A,#43H
            RET
AS13:       CJNE    A,#0DH,AS14
            MOV     A,#44H
            RET
AS14:       CJNE    A,#0EH,AS15
            MOV     A,#45H
            RET
AS15:       CJNE    A,#0FH,FINISH
            MOV     A,#46H
            RET
FINISH:     RET
;*************************************************************************
;       SHOW SLAVE#1 DATA
;*************************************************************************
SHOW1:      MOV     P0,#11001111B       ; SHOW SLAVE1 DATA
            CALL    DELAY_1s
            CALL    DELAY_1s
            MOV     P0,TEMP_M1
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
            MOV     P0,TEMP_L1
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
            MOV     P0,HUMI_M1
            CALL    DELAY_1s
            CALL    DELAY_1s
            CALL    DELAY_1s
```

```
                CALL    DELAY_1s
                MOV     P0,HUMI_L1
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                RET
;*********************************************************************************
; SHOW SLAVE#1 DATA ASCII
;*********************************************************************************
SHOW_ASCII1:    MOV     P0,#11000011B       ; SHOW SLAVE1 DATA
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_T1
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_T2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_T3
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_T4
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_H1
```

```
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_H2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_H3
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE1_H4
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                RET
;*************************************************************************
;       SHOW SLAVE#2 DATA
;*************************************************************************
SHOW2:          MOV     P0,#11110011B           ; SHOW SLAVE1 DATA
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,TEMP_M2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,TEMP_L2
                CALL    DELAY_1s
                CALL    DELAY_1s
```

```
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,HUMI_M2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,HUMI_L2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                RET
;*************************************************************************
; SHOW SLAVE#2 DATA ASCII
;*************************************************************************
SHOW_ASCII2:    MOV     P0,#11000011B       ; SHOW SLAVE2 DATA
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE2_T1
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE2_T2
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                MOV     P0,SLAVE2_T3
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
                CALL    DELAY_1s
```

```
        MOV     P0,SLAVE2_T4
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        MOV     P0,SLAVE2_H1
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        MOV     P0,SLAVE2_H2
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        MOV     P0,SLAVE2_H3
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        MOV     P0,SLAVE2_H4
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        CALL    DELAY_1s
        RET
;*************************************************
INIT:   CALL    DELAY_100ms
        CLR     CE
        CLR     CS
        CLR     DAT
        CLR     CLK
        CALL    DELAY_100ms
        RET
```

```
;*************************************************
;FUNCTION CONFIG FOR TRW RX
;*************************************************************************
SETMODE_RX:  CALL   DELAY_100ms
             CLR    CE
             SETB   CS
             CLR    A
             MOV    R1,#18                  ;18*8 = 144 bit
SETMODE_RX_0:        MOV    DPTR,#CONFIG_TEST_RX        ;SET HEADER
             PUSH   ACC
             MOVC   A,@A+DPTR
             CALL   WRITE_TRW24
             POP    ACC
             INC    A
             DJNZ   R1,SETMODE_RX_0
             SETB   DAT
             CLR    DR1
             SETB   DR1
             SETB   CE
             CLR    CS
             CALL   DELAY_100ms
             RET
;*************************************************************************
;WRITE 8 BIT TO DATA PIN
;*************************************************************************
WRITE_TRW24: MOV    R0,#8
WRITE_TRW24_0:       JB     ACC.7,WRITE1
             CLR    DAT
             JMP    WRITE_TRW2
WRITE1:              SETB   DAT
WRITE_TRW2:  CALL   CLK_TRW
             RL     A
             DJNZ   R0,WRITE_TRW24_0
             RET
```

```
;*********************************************
CLK_TRW:        CLR     CLK
                CALL    DELAY_1ms
                SETB    CLK
                CALL    DELAY_1ms
                        RET
;*************************************************************************
;READ 8 BIT DATA FROM TRW
;*************************************************************************
READ_TRW24:     CLR     A
                MOV     R0,#8
READ_TRW24_0:RL         A
                SETB    CLK
                CALL    DELAY_1ms
                JB      DAT,READ_1
                CLR     ACC.0
                JMP     READ_TRW24_1
READ_1:                 SETB    ACC.0
READ_TRW24_1:CLR        CLK
                CALL    DELAY_1ms
                DJNZ    R0,READ_TRW24_0
                RET
;*************************************************************************
;DELAY TIME 1ms
;*************************************************************************
DELAY_1ms:      MOV     R6,#0E6H                ;EACH LOOP = 1ms
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6,DELAY_1ms_1
                RET
;*************************************************************************
;DELAY TIME 100ms
;*************************************************************************
DELAY_100ms:    MOV     R7,#100                 ;DO 100 TIMES
```

```
DELAY_100ms_1:      MOV    R6,#0E6H              ;EACH LOOP = 1ms
DELAY_100ms_2:      NOP
                    NOP
                    DJNZ   R6,DELAY_100ms_2
                    DJNZ   R7,DELAY_100ms_1
                    RET
;*************************************************************************
;DELAY TIME 1s
;*************************************************************************
DELAY_1s:           MOV    R5,#0AH
DELAY_1s_1:         CALL   DELAY_100ms
                    DJNZ   R5,DELAY_1s_1
                    RET
;****************************************
CONFIG_TEST_RX:                 DB      8EH,08H,1CH
CONFIG_LEN2_RX:                 DB      08H
CONFIG_LEN1_RX:                 DB      08H
CONFIG_ADDR2_RX:    DB          0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1_RX:    DB          0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR_RX:  DB          0A3H
CONFIG_RF_RX:       DB          6FH
CONFIG_CH_RX:       DB          0BH
CONFIG_TEST:        DB          8EH,08H,1CH
CONFIG_LEN2:        DB          08H
CONFIG_LEN1:        DB          08H
CONFIG_ADDR2:                   DB      0C0H,0AAH,55H,0AAH,55H
CONFIG_ADDR1:                   DB      0AAH,55H,0AAH,55H,0AAH
CONFIG_NUMADDR:                 DB      0A3H
CONFIG_RF:          DB          6FH
CONFIG_CH:          DB          0AH
END
```

ฟอร์มการบันทึกค่าและแสดงตาราง

```
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 System.IO.Ports;
using System.Globalization;
using System.Threading;
using System.Data.OleDb;

namespace Project4a
{
  public partial class frmCal : Form
  {
    SerialPort port;
    string t, t2, t1, t3, hum1, hum2;

    public frmCal()
    {
      InitializeComponent();
      port = new SerialPort();
      port.PortName = "COM1";
      port.BaudRate = 9600;
      port.Parity = Parity.None;
      port.DataBits = 8;
      port.StopBits = StopBits.One;
      port.Open();
      //attach a method to be called when there is data waiting in the port's buffer
      port.DataReceived += new SerialDataReceivedEventHandler(serial_DataReceived);
    }
    private void frmCal_Load(object sender, EventArgs e)
    {

    }
      private void serial_DataReceived(object sender,
                    System.IO.Ports.SerialDataReceivedEventArgs e)
            {
                    //Event for receiving data
                    //Read the buffer to text box.
                    this.Invoke(new EventHandler(DisplayText));
            }

            private string a, c = "";
            private string rx;
            private void DisplayText(object sender, EventArgs e)
            {

                    // นำข้อมูลที่อยู่ในอินพุท บัฟเฟอร์ทั้งหมดออกมา
                    rx = rx + port.ReadExisting();
                    Checking();
            }
```

```
private string q, q1;
private void Checking()

{
        if(rx.Length == 10)
        {
                a = rx;
                c = c + a;
                string b = c.Substring(0,1);
                string b1 = c.Substring(9,1);
                {
                        if((b == "#")  &&  (b1 == "#"))
                        {
                                t = t + c;
                                txtRx1.Text = t;
                                q = "1";
                                meas1();
                        }
                        else if ((b == "*") && (b1 == "*"))
                        {
                                t3 = t3 + c;
                                txtRx2.Text = t3;
                                q1 = "2";
                                meas2();
                        }

                        else
                        {
                                MessageBox.Show("ข้อมูลไม่ถูก");
                        }
                }
        }
}


private void meas1()
{

        string r = c.Remove(0,1);
        string r1 = r.Remove(8,1);
        string c1 = r1.Remove(4,4);
        string t1 = c1;
        int i = Int32.Parse(t1,NumberStyles.HexNumber);
        float y = Convert.ToSingle(i);
        cal_t1(y);

        string c2 = r1.Remove(0,4);
        string h = c2;
        int j = Int32.Parse(h,NumberStyles.HexNumber);
        float g = Convert.ToSingle(j);
        cal_hu1(g);
        txtTim1.Text = DateTime.Now.ToLongTimeString();
        AddtodataBase1();
AddtodataBase3();
```

```
        }
        private void cal_t1(float x)
        {
                temp1 = (0.01f * x) - 39.66f;
                t1 = temp1.ToString();
txtTemp1.Text = t1;
                m = txtTemp1.Text;
        }


        private float temp1;
        private void cal_hu1(float humi_d)

        {
                const float c1 = -4.0f;              //for 12 bit
                const float c2 = 0.0405f;
                const float c3 = -0.0000028f;
                const float t1 = 0.01f;              //for 14 bit 3v
                const float t2 = 0.00008f;
                float rh_linear, rh_true;

                rh_linear = c1 + (c2 * humi_d) + (c3 * humi_d * humi_d);
                rh_true = (temp1 - 25) * (t1 + t2 * humi_d) + rh_linear;
                hum1 = rh_true.ToString();
                txtHumi1.Text = hum1;
                 k = txtHumi1.Text;
        }

        private void meas2()
        {
                string r = c.Remove(0,1);
                string r1 = r.Remove(8,1);
                string c1 = r1.Remove(4,4);
                string t1 = c1;
                int i = Int32.Parse(t1,NumberStyles.HexNumber);
                float y = Convert.ToSingle(i);
                cal_t2(y);

                string c2 = r1.Remove(0,4);
                string h = c2;
                int j = Int32.Parse(h,NumberStyles.HexNumber);
                float g = Convert.ToSingle(j);
                cal_hu2(g);
                txtTim2.Text = DateTime.Now.ToLongTimeString();
AddtodataBase2();
AddtodataBase4();
        }

        private void cal_t2(float x)
        {
                temp2 = (0.01f * x) - 39.66f;
                t2 = temp2.ToString();
                txtTemp2.Text = t2;
                m = txtTemp2.Text;
```

```
        }

        private float temp2;
        private void cal_hu2(float humi_d)

        {
                const float c1 = -4.0f;                 //for 12 bit
                const float c2 = 0.0405f;
                const float c3 = -0.0000028f;
                const float t1 = 0.01f;                 //for 14 bit 3v
                const float t2 = 0.00008f;
                float rh_linear, rh_true;

                rh_linear = c1 + (c2 * humi_d) + (c3 * humi_d * humi_d);
                rh_true = (temp2 - 25) * (t1 + t2 * humi_d) + rh_linear;
                hum2 = rh_true.ToString();
                txtHumi2.Text = hum2;
                 k = txtHumi2.Text;
        }

        private void ClearAll1()

        {
                txtRx1.Text = "";
                txtTim1.Text = "";
                txtTemp1.Text = "";
                txtHumi1.Text = "";
                c = "";
                rx = "";
                t = "";
                t3 = "";

                this.Invoke(new EventHandler(DisplayText));
txtRx1.Text = "1";
txtTim1.Text = DateTime.Now.ToLongTimeString();
                txtTemp1.Text = t1;
txtHumi1.Text = hum1;
        }

        private void ClearAll2()

        {
                txtRx2.Text = "";
                txtTemp2.Text = "";
                txtHumi2.Text = "";
                txtTim2.Text = "";
                c = "";
                rx = "";
                t = "";
                t3 = "";

                this.Invoke(new EventHandler(DisplayText));
txtRx2.Text = "2";
txtTim2.Text = DateTime.Now.ToLongTimeString();
txtTemp2.Text = t2;
txtHumi2.Text = hum2;
```

```
        }

        private string k,m;
        private void AddtodataBase1()

        {
                string sql;
    string strConn = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source
=D:\\project_4a\\data007.mdb;Jet OLEDB:Engine Type=5";
                //กำหนดข้อความเชื่อมต่อฐานข้อมูล
                OleDbConnection conn = new OleDbConnection(strConn);

                string x = DateTime.Now.ToLongTimeString();
                string y = DateTime.Now.ToShortDateString();
                sql = "INSERT INTO Collect(station,dat_mea,tim_mea,Temp_va,Humi_va)" ;
                sql += "VALUES(' " + q + " ',";
                sql += " ' " + y + " ' , ";
                sql += " ' " + x + " ' , ";
                sql += " ' " + m + " ' , ";
                sql += " ' " + k + " ' ) ";

                //สร้างแบบใช้ command(Connection)
                OleDbCommand cmd = new OleDbCommand();


                try
                {
                        //ตรวจสอบสถานะการเชื่อมต่อเดิมของ conn
                        //ถ้ามีการเชื่อมต่อเดิมอยู่
                        if (conn.State == ConnectionState.Open)
                        {
                                conn.Close();//ปิดการเชื่อมต่อเดิมก่อน
                        }

                        conn.Open(); //เปิดการเชื่อมต่อ

                        cmd = new OleDbCommand(sql,conn);
                        cmd.ExecuteNonQuery();
                }
                catch (Exception errToAdd)
                {
                        MessageBox.Show("ไม่สามารถเพิ่มข้อมูลได้" + errToAdd.Message,"
ข้อผิดพลาด");
                }
                finally
                {
                        conn.Close();
                        ClearAll1();
    //      MessageBox.Show("" + k + "");
                }
```

```
            }
    private void AddtodataBase2()
    {
        string sql;
        string strConn = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source
=D:\\project_4a\\data007.mdb;Jet OLEDB:Engine Type=5";
        //กำหนดข้อความเชื่อมต่อฐานข้อมูล
        OleDbConnection conn = new OleDbConnection(strConn);

        string x = DateTime.Now.ToLongTimeString();
        string y = DateTime.Now.ToShortDateString();
        sql = "INSERT INTO Collect(station,dat_mea,tim_mea,Temp_va,Humi_va)";
        sql += "VALUES(' " + q1 + " ',";
        sql += " ' " + y + " ' , ";
        sql += " ' " + x + " ' , ";
        sql += " ' " + m + " ' , ";
        sql += " ' " + k + " ' ) ";

        //สร้างแบบใช้ command(Connection)
        OleDbCommand cmd = new OleDbCommand();


        try
        {
            //ตรวจสอบสถานะการเชื่อมต่อเดิมของ conn
            //ถ้ามีการเชื่อมต่อเดิมอยู่
            if (conn.State == ConnectionState.Open)
            {
                conn.Close();//ปิดการเชื่อมต่อเดิมก่อน
            }

            conn.Open(); //เปิดการเชื่อมต่อ

            cmd = new OleDbCommand(sql, conn);
            cmd.ExecuteNonQuery();
        }
        catch (Exception errToAdd)
        {
            MessageBox.Show("ไม่สามารถเพิ่มข้อมูลได้" + errToAdd.Message, "ข้อผิดพลาด");
        }
        finally
        {
            conn.Close();
            ClearAll2();
        }


    }


    private void AddtodataBase3()

                {
                        string sql;
```

```
        string strConn = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source
=D:\\project_4a\\data007.mdb;Jet OLEDB:Engine Type=5";
                        //กำหนดข้อความเชื่อมต่อฐานข้อมูล
                        OleDbConnection conn = new OleDbConnection(strConn);

                        string x = DateTime.Now.ToLongTimeString();
                        string y = DateTime.Now.ToShortDateString();
                        sql = "INSERT INTO Collect_1(station,dat_mea,tim_mea,Temp_va,Humi_va)"
;
                        sql += "VALUES(' " + q + " ' ,";
                        sql += " ' " + y + " ' , ";
                        sql += " ' " + x + " ' , ";
                        sql += " ' " + m + " ' , ";
                        sql += " ' " + k + " ' ) ";

                        //สร้างแบบใช้ command(Connection)
                        OleDbCommand cmd = new OleDbCommand();


                        try
                        {
                                //ตรวจสอบสถานะการเชื่อมต่อเดิมของ conn
                                //ถ้ามีการเชื่อมต่อเดิมอยู่
                                if (conn.State == ConnectionState.Open)
                                {
                                        conn.Close();//ปิดการเชื่อมต่อเดิมก่อน
                                }

                                conn.Open(); //เปิดการเชื่อมต่อ

                                cmd = new OleDbCommand(sql,conn);
                                cmd.ExecuteNonQuery();
                        }
                        catch (Exception errToAdd)
                        {
                                MessageBox.Show("ไม่สามารถเพิ่มข้อมูลได้" + errToAdd.Message,"
ข้อผิดพลาด");

                        }
                        finally
                        {
                                conn.Close();
                                ClearAll1();
    //      MessageBox.Show("" + k + "");
                        }
      }


    private void AddtodataBase4()
    {
      string sql;
      string strConn = "Provider = Microsoft.Jet.OLEDB.4.0; Data Source
=D:\\project_4a\\data007.mdb;Jet OLEDB:Engine Type=5";
      //กำหนดข้อความเชื่อมต่อฐานข้อมูล
      OleDbConnection conn = new OleDbConnection(strConn);
```

```
    string x = DateTime.Now.ToLongTimeString();
    string y = DateTime.Now.ToShortDateString();
    sql = "INSERT INTO Collect_2(station,dat_mea,tim_mea,Temp_va,Humi_va)";
    sql += "VALUES(' " + q1 + " ',";
    sql += " ' " + y + " ' , ";
    sql += " ' " + x + " ' , ";
    sql += " ' " + m + " ' , ";
    sql += " ' " + k + " ' ) ";

    //สร้างแบบใช้ command(Connection)
    OleDbCommand cmd = new OleDbCommand();


    try
    {
      //ตรวจสอบสถานะการเชื่อมต่อเดิมของ conn
      //ถ้ามีการเชื่อมต่อเดิมอยู่
      if (conn.State == ConnectionState.Open)
      {
        conn.Close();//ปิดการเชื่อมต่อเดิมก่อน
      }

      conn.Open(); //เปิดการเชื่อมต่อ

      cmd = new OleDbCommand(sql, conn);
      cmd.ExecuteNonQuery();
    }
    catch (Exception errToAdd)
    {
      MessageBox.Show("ไม่สามารถเพิ่มข้อมูลได้" + errToAdd.Message, "ข้อผิดพลาด");
    }
    finally
    {
      conn.Close();
      ClearAll1();
      //      MessageBox.Show("'" + k + "'");
    }

}
int station;
string strConn;
int go;
OleDbConnection Conn = new OleDbConnection();
OleDbDataAdapter da;
DataSet ds = new DataSet();
bool IsFind = false;
private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
    dgvAlldata.DataSource = null;
    ds.Clear();
```

```
        strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:/project_4a/data007.mdb;Jet
OLEDB:Engine Type=5";
        if (Conn.State == ConnectionState.Open)
        {
          Conn.Close();
        }
        Conn.ConnectionString = strConn;
        Conn.Open();
        string sqlEmp;

          sqlEmp = "SELECT station,dat_mea,tim_mea,Temp_va,Humi_va FROM Collect ";
          sqlEmp += " WHERE dat_mea like '%" + monthCalendar1.SelectionStart.ToString("d/M/yyyy")
+ "%' AND [station] = ' " + station.ToString() + " '";

          if (IsFind == true)
          {
            ds.Tables["Collect"].Clear();
          }
          da = new OleDbDataAdapter(sqlEmp, Conn);
          da.Fill(ds, "Collect");

          if (ds.Tables["Collect"].Rows.Count != 0)
          {
            IsFind = true;
            dgvAlldata.ReadOnly = true;

          }
          else
          {
            IsFind = false;

          }

        dgvAlldata.DataSource = ds.Tables["Collect"];
      }

      private void radioButton1_CheckedChanged(object sender, EventArgs e)
      {
        station = 1;
      }

      private void radioButton2_CheckedChanged(object sender, EventArgs e)
      {
        station = 2;
      }

      private void dgvAlldata_CellContentClick(object sender, DataGridViewCellEventArgs e)
      {

      }

      private void button2_Click(object sender, EventArgs e)
      {
        Form2 a = new Form2(monthCalendar1.SelectionStart);
          a.Show();
          a.Visible = true;
```

```
    }

    private void button3_Click(object sender, EventArgs e)
    {
      Form3 b = new Form3(monthCalendar1.SelectionStart);
      b.Show();
      b.Visible = true;
    }

    private void button4_Click(object sender, EventArgs e)
    {
      Form4 c = new Form4(monthCalendar1.SelectionStart);
      c.Show();
      c.Visible = true;
    }

    private void button5_Click(object sender, EventArgs e)
    {
      Form5 d = new Form5(monthCalendar1.SelectionStart);
      d.Show();
      d.Visible = true;
    }
  }
}
```

ฟอร์มแสดงกราฟอุณหภูมิสถานีที่ 1

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

namespace Project4a
{
  public partial class Form2 : Form
  {
    double[] arr;
    DateTime date;
    public Form2(DateTime date)
    {
      InitializeComponent();
      this.date = date;
    }

    private void Form2_Load(object sender, EventArgs e)
    {
      CreateGraph(zgl);
      SetSize();
    }

    private void CreateGraph(ZedGraphControl zgc)
```

```
    {
        string constr = "Provider=Microsoft.Jet.OleDb.4.0;" +
            "Data Source = D:\\project_4a\\data007.mdb";
        OleDbConnection conn = new OleDbConnection(constr);

        string sqlresult = "SELECT * FROM Collect_1 Where dat_mea like '%" +
date.ToString("d/M/yyyy") + "%'";
        OleDbDataAdapter daTable = new
        OleDbDataAdapter(sqlresult, conn);
        DataSet dset = new DataSet();
        daTable.Fill(dset, "dat_mea");

        arr = new double[dset.Tables["dat_mea"].Rows.Count];

        for (int i = 0; i < arr.Length; i++)
        {
            arr[i] = Convert.ToDouble(dset.Tables["dat_mea"].Rows[i]["Temp_va"]);
        }
        //get a reference to the GraphPane

        zgc.GraphPane = new GraphPane();
        GraphPane myPane = zgc.GraphPane;

        //Set the titles and axis labels
        myPane.Title.Text = "กราฟแสดงค่าอุณหภูมิจากสถานีที่ 1 ของวันที่ " + date.ToString("d/M/yyyy")
+ "";
        myPane.XAxis.Title.Text = "เวลาที่ทำการวัด";
        myPane.YAxis.Title.Text = "ค่าอุณหภูมิ(C)";

        PointPairList list = new PointPairList();

        for (int x = 0; x < arr.Length; x++)
        {
            string[] time = dset.Tables["dat_mea"].Rows[x]["tim_mea"].ToString().Trim().Split(':');
            XDate xtime = new XDate(this.date.Year, this.date.Month, this.date.Day, int.Parse(time[0]),
int.Parse(time[1]), int.Parse(time[2]));
            list.Add(xtime, arr[x]);
        }

        //Add gridlines to the plot
        myPane.XAxis.MajorGrid.IsVisible = true;
        myPane.YAxis.MajorGrid.IsVisible = true;
        myPane.XAxis.MajorGrid.Color = Color.LightGray;
        myPane.YAxis.MajorGrid.Color = Color.LightGray;

        myPane.XAxis.Type = AxisType.Date;

        //Generate a blue curve with circle symbols
        LineItem myCurve = myPane.AddCurve("ค่าอุณหภูมิ(C)", list, Color.Blue, SymbolType.None);

        //filling the symbol with white
        myCurve.Symbol.Fill = new Fill(Color.White);

        //Fill the axis background with a color
        myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);
```

```
            //Fill the pane background with a color gradient
            myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);

            //Calculate the Axis Scale Ranges
            zgc.AxisChange();
        }

        private void Form2_Resize(object sender, EventArgs e)
        {
            SetSize();
        }

        private void SetSize()
        {
            zgl.Location = new Point(10, 10);
            //Leave a small margin around the outside of the control
            zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
        }
        private void timer1_Tick_1(object sender, EventArgs e)
        {
            CreateGraph(zgl);
            SetSize();
            zgl.Invalidate();
        }
    }
}
```

ฟอร์มแสดงกราฟอุณหภูมิสถานีที่ 2

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

namespace Project4a
{
    public partial class Form3 : Form
    {
        double[] arr;
        DateTime date;
        public Form3(DateTime date)
        {
            InitializeComponent();
            this.date = date;
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            CreateGraph(zgl);
            SetSize();
        }
```

```
    private void CreateGraph(ZedGraphControl zgc)
    {
        string constr = "Provider=Microsoft.Jet.OleDb.4.0;" +
            "Data Source = D:\\project_4a\\data007.mdb";
        OleDbConnection conn = new OleDbConnection(constr);


        string sqlresult = "SELECT * FROM Collect_2 Where dat_mea like '%" +
date.ToString("d/M/yyyy") + "%'";
        OleDbDataAdapter daTable = new
        OleDbDataAdapter(sqlresult, conn);
        DataSet dset = new DataSet();
        daTable.Fill(dset, "mydat");

        arr = new double[dset.Tables["mydat"].Rows.Count];

        for (int i = 0; i < arr.Length; i++)
        {
            arr[i] = Convert.ToDouble(dset.Tables["mydat"].Rows[i]["Temp_va"]);
        }
        //get a reference to the GraphPane

        zgc.GraphPane = new GraphPane();
        GraphPane myPane = zgc.GraphPane;

        //Set the titles and axis labels
        myPane.Title.Text = "กราฟแสดงค่าอุณหภูมิจากสถานีที่ 2  ของวันที่ " + date.ToString("d/M/yyyy")
+ "";
        myPane.XAxis.Title.Text = "เวลาที่ทำการวัด";
        myPane.YAxis.Title.Text = "ค่าอุณหภูมิ(C)";

        PointPairList list = new PointPairList();

        for (int x = 0; x < arr.Length; x++)
        {
            string[] time = dset.Tables["mydat"].Rows[x]["tim_mea"].ToString().Trim().Split(':');
            XDate xtime = new XDate(this.date.Year, this.date.Month, this.date.Day, int.Parse(time[0]),
int.Parse(time[1]), int.Parse(time[2]));
            list.Add(xtime, arr[x]);
        }

        //Add gridlines to the plot
        myPane.XAxis.MajorGrid.IsVisible = true;
        myPane.YAxis.MajorGrid.IsVisible = true;
        myPane.XAxis.MajorGrid.Color = Color.LightGray;
        myPane.YAxis.MajorGrid.Color = Color.LightGray;

        myPane.XAxis.Type = AxisType.Date;


        //Generate a blue curve with circle symbols
        LineItem myCurve = myPane.AddCurve("ค่าอุณหภูมิ(C)", list, Color.Blue, SymbolType.None);

        //filling the symbol with white
        myCurve.Symbol.Fill = new Fill(Color.White);
```

```
            //Fill the axis background with a color
            myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);

            //Fill the pane background with a color gradient
            myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);

            //Calculate the Axis Scale Ranges
            zgc.AxisChange();
        }

        private void Form3_Resize(object sender, EventArgs e)
        {
            SetSize();
        }

        private void SetSize()
        {
            zgl.Location = new Point(10, 10);
            //Leave a small margin around the outside of the control
            zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
        }


        private void timer1_Tick_1(object sender, EventArgs e)
        {
            CreateGraph(zgl);
            SetSize();
            zgl.Invalidate();
        }
    }
}
```

ฟอร์มแสดงกราฟความชื้นสัมพัทธ์สถานีที่ 1

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

namespace Project4a
{
    public partial class Form4 : Form
    {
        double[] arr;
        DateTime date;
        public Form4(DateTime date)
        {
            InitializeComponent();
            this.date = date;
        }
```

```
        private void Form4_Load(object sender, EventArgs e)
        {
            CreateGraph(zgl);
            SetSize();
        }

        private void CreateGraph(ZedGraphControl zgc)
        {
            string constr = "Provider=Microsoft.Jet.OleDb.4.0;" +
                "Data Source = D:\\project_4a\\data007.mdb";
            OleDbConnection conn = new OleDbConnection(constr);

            string sqlresult = "SELECT * FROM Collect_1 Where dat_mea like '%" +
date.ToString("d/M/yyyy") + "%'";
            OleDbDataAdapter daTable = new
            OleDbDataAdapter(sqlresult, conn);
            DataSet dset = new DataSet();
            daTable.Fill(dset, "mydat");

            arr = new double[dset.Tables["mydat"].Rows.Count];

            for (int i = 0; i < arr.Length; i++)
            {
                arr[i] = Convert.ToDouble(dset.Tables["mydat"].Rows[i]["Humi_va"]);
            }

            zgc.GraphPane = new GraphPane();

            GraphPane myPane = zgc.GraphPane;

            //Set the titles and axis labels
            myPane.Title.Text = "กราฟแสดงค่าความชื้นสัมพัทธ์จากสถานีที่ 1 ของวันที่ " +
date.ToString("d/M/yyyy") + "";
            myPane.XAxis.Title.Text = "เวลาที่ทำการวัด";
            myPane.YAxis.Title.Text = "ค่าความชื้นสัมพัทธ์";

            PointPairList list = new PointPairList();

            for (int x = 0; x < arr.Length; x++)
            {
                string[] time = dset.Tables["mydat"].Rows[x]["tim_mea"].ToString().Trim().Split(':');
                XDate xtime = new XDate(this.date.Year, this.date.Month, this.date.Day, int.Parse(time[0]),
int.Parse(time[1]), int.Parse(time[2]));
                list.Add(xtime, arr[x]);
            }

            //Add gridlines to the plot
            myPane.XAxis.MajorGrid.IsVisible = true;
            myPane.YAxis.MajorGrid.IsVisible = true;
            myPane.XAxis.MajorGrid.Color = Color.LightGray;
            myPane.YAxis.MajorGrid.Color = Color.LightGray;

            myPane.XAxis.Type = AxisType.Date;
```

```
        //Generate a blue curve with circle symbols
        LineItem myCurve = myPane.AddCurve("ค่าความชื้นสัมพัทธ์", list, Color.Red,
SymbolType.None);

        //filling the symbol with white
        myCurve.Symbol.Fill = new Fill(Color.White);

        //Fill the axis background with a color
        myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);

        //Fill the pane background with a color gradient
        myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);

        //Calculate the Axis Scale Ranges
        zgc.AxisChange();
    }

    private void Form4_Resize(object sender, EventArgs e)
    {
        SetSize();
    }

    private void SetSize()
    {
        zgl.Location = new Point(10, 10);
        //Leave a small margin around the outside of the control
        zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
    }

    private void timer1_Tick_1(object sender, EventArgs e)
    {
        CreateGraph(zgl);
        SetSize();
        zgl.Invalidate();
    }
  }
}
```

ฟอร์มแสดงกราฟความชื้นสัมพัทธ์สถานีที่ 2

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

namespace Project4a
{
  public partial class.Form5 : Form
  {
    double[] arr;
```

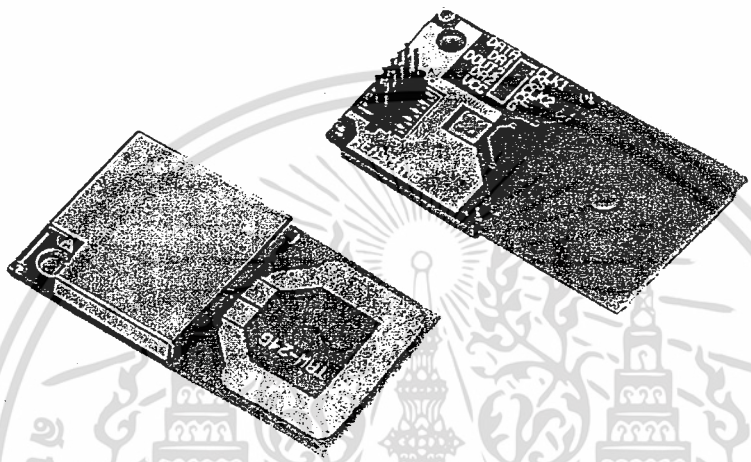
```
DateTime date;
public Form5(DateTime date)
{
  InitializeComponent();
  this.date = date;
}

private void Form5_Load(object sender, EventArgs e)
{
  CreateGraph(zgl);
  SetSize();
}

private void CreateGraph(ZedGraphControl zgc)
{
  string constr = "Provider=Microsoft.Jet.OleDb.4.0;" +
    "Data Source = D:\\project_4a\\data007.mdb";
  OleDbConnection conn = new OleDbConnection(constr);

  string sqlresult = "SELECT * FROM Collect_2 Where dat_mea like '%" +
date.ToString("d/M/yyyy") + "%'";
  OleDbDataAdapter daTable = new
  OleDbDataAdapter(sqlresult, conn);
  DataSet dset = new DataSet();
  daTable.Fill(dset, "mydat");

  arr = new double[dset.Tables["mydat"].Rows.Count];

  for (int i = 0; i < arr.Length; i++)
  {
    arr[i] = Convert.ToDouble(dset.Tables["mydat"].Rows[i]["Humi_va"]);
  }
  //get a reference to the GraphPane

  zgc.GraphPane = new GraphPane();
  GraphPane myPane = zgc.GraphPane;

  //Set the titles and axis labels
  myPane.Title.Text = "กราฟแสดงค่าความชื้นสัมพัทธ์จากสถานีที่ 2 ของวันที่ " +
date.ToString("d/M/yyyy") + "";
  myPane.XAxis.Title.Text = "เวลาที่ทำการวัด";
  myPane.YAxis.Title.Text = "ความชื้นสัมพัทธ์";

  PointPairList list = new PointPairList();

  for (int x = 0; x < arr.Length; x++)
  {
    string[] time = dset.Tables["mydat"].Rows[x]["tim_mea"].ToString().Trim().Split(':');
    XDate xtime = new XDate(this.date.Year, this.date.Month, this.date.Day, int.Parse(time[0]),
int.Parse(time[1]), int.Parse(time[2]));
    list.Add(xtime, arr[x]);
  }

  //Add gridlines to the plot
  myPane.XAxis.MajorGrid.IsVisible = true;
  myPane.YAxis.MajorGrid.IsVisible = true;
```

```
            myPane.XAxis.MajorGrid.Color = Color.LightGray;
            myPane.YAxis.MajorGrid.Color = Color.LightGray;

            myPane.XAxis.Type = AxisType.Date;


            //Generate a blue curve with circle symbols
            LineItem myCurve = myPane.AddCurve("ความชื้นสัมพัทธ์", list, Color.Red, SymbolType.None);

            //filling the symbol with white
            myCurve.Symbol.Fill = new Fill(Color.White);

            //Fill the axis background with a color
            myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45F);

            //Fill the pane background with a color gradient
            myPane.Fill = new Fill(Color.White, Color.FromArgb(220, 220, 255), 45F);

            //Calculate the Axis Scale Ranges
            zgc.AxisChange();
        }

        private void Form5_Resize(object sender, EventArgs e)
        {
            SetSize();
        }

        private void SetSize()
        {
            zgl.Location = new Point(10, 10);
            //Leave a small margin around the outside of the control
            zgl.Size = new Size(this.ClientRectangle.Width - 20, this.ClientRectangle.Height - 20);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            CreateGraph(zgl);
            SetSize();
            zgl.Invalidate();
        }
    }
}
```

TRW-24G

WENSHING®© V1.04

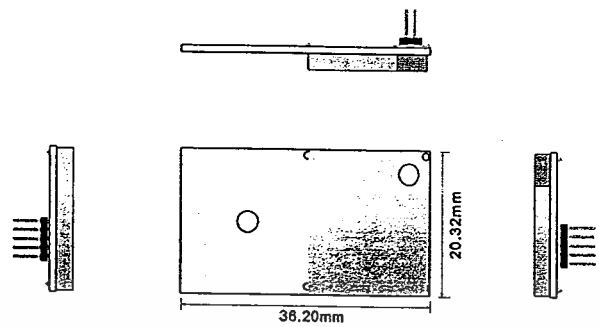
Wireless High Frequency Transceiver Module (RF GFSK)

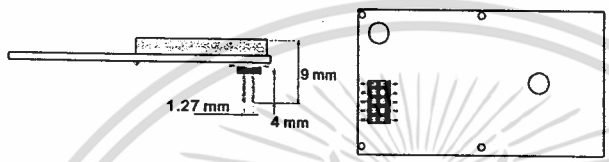


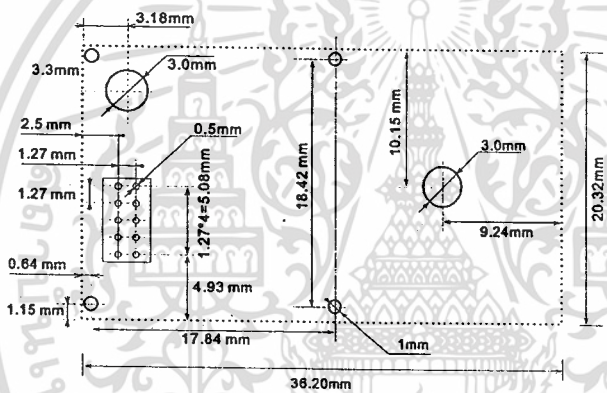
➢ Key Features :

- Frequency Range: 2.4~2.527GHz
- Modulate Mode: GFSK
- Work Voltage: 3V
- Channel: 128
- Output Power: 0dBm
- Data Rate: 1Mbps; 250Kbps
- Operating Temperature: -40~+85 Centigrade
- The longest range : 280m (250Kbps); 150m (1Mbps)
- No dead spaces in reception.
- Built in antenna.
- Competitive price.
- Apply for various type of products: Wireless Joysticks, Wireless Speaker, Wireless Earphone , Wireless Cell phone , Wireless Intercom , Wireless Mouse, Wireless Keyboard and Data Communication.

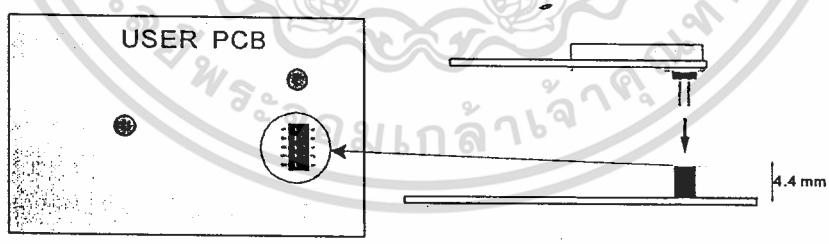
➢ Graph:

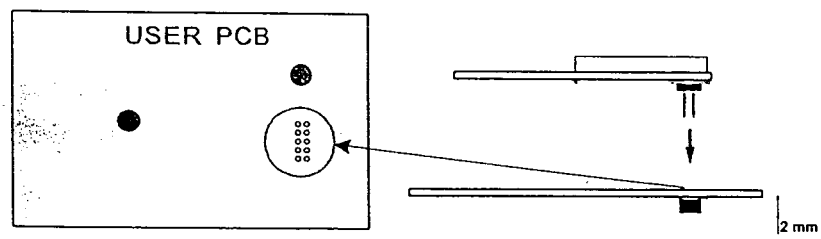






Reference hole position for PCB mounting(Bottom view)



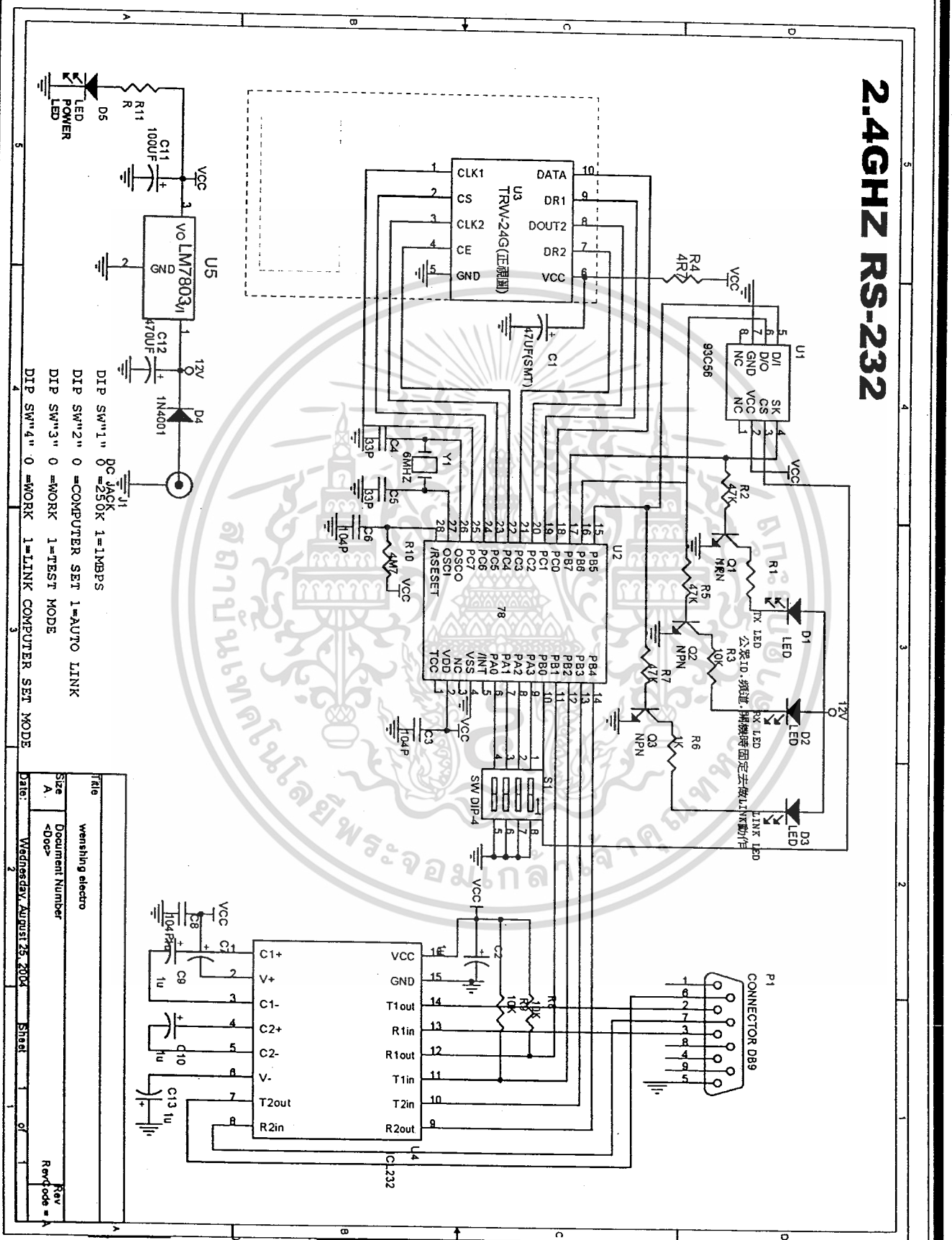


➢ Specification :

Conditions: VDD=+3V,VSS=0V,T_A=-40 centigrade to +85 centigrade

Symbol	Parameter(condition)	Min	Typ	Max	Unit
V_{DD}	Supply voltage	1.9	3.0	3.6	V
TEMP	Operating temperature	-40	+27	+85	Centigrade
f_{OP}	Operating frequency	2400		2527	MHz
R_{GFSK}	Data rate direct mode	250		1000	Kbps
$F_{CHANNEL}$	Channel spacing		1		MHz
I_{VDD}	Supply current one channel 250Kbps		18		mA
I_{VDD}	Supply current one channel 1000Kbps		19		mA
I_{VDD}	Supply current two channels 250Kbps		23		mA
I_{VDD}	Supply current two channels 1000Kbps		25		mA
RX_{SENS}	Sensitivity at 0.1%BER(@250Kbps)		-90		dBm
RX_{SENS}	Sensitivity at 0.1%BER(@1000Kbps)		-80		dBm

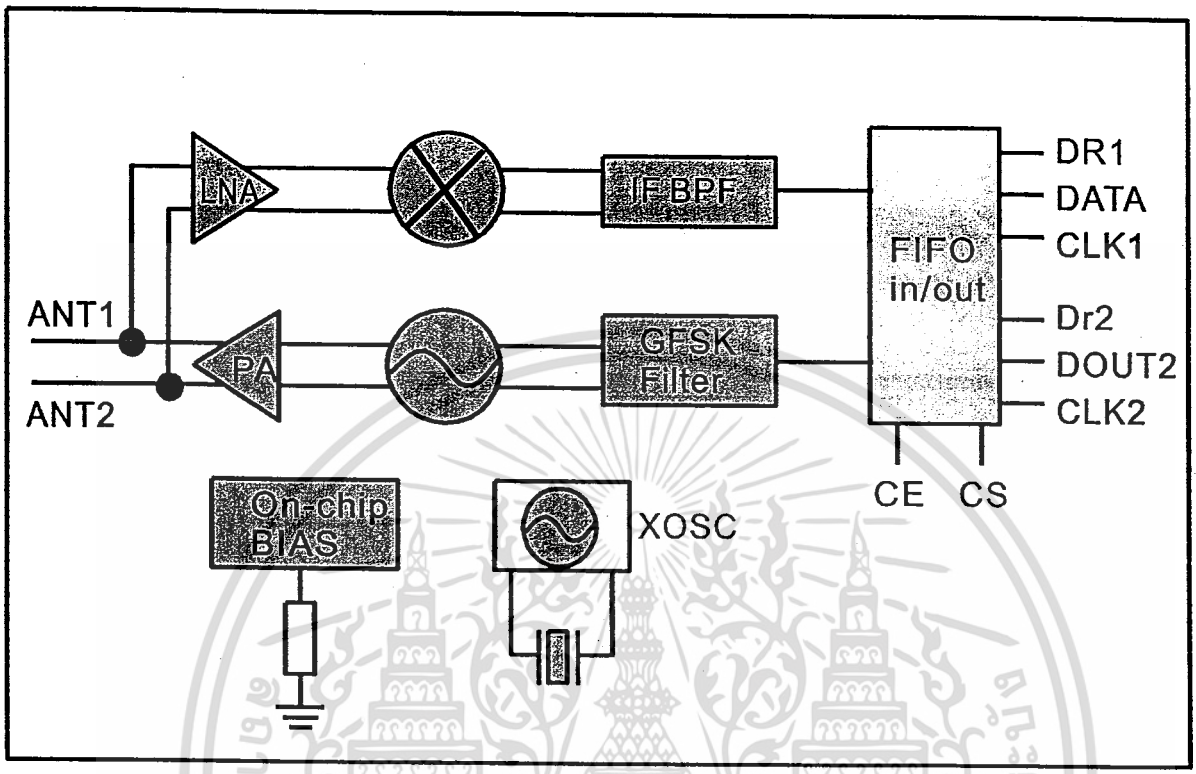
➢ Demo Circuit Diagram :



Conditions: VDD = +3V, VSS = 0V, T_A = - 40°C to + 85°C

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	Operating conditions					
VDD	Supply voltage		1.9	3.0	3.6	V
TEMP	Operating Temperature		-40	+27	+85	°C
	Digital input pin					
V_{IH}	HIGH level input voltage		VDD- 0.3		VDD	V
V_{IL}	LOW level input voltage		Vss		0.3	V
	Digital output pin					
V_{OH}	HIGH level output voltage (I_{OH}=-0.5mA)		VDD- 0.3		VDD	V
V_{OL}	LOW level output voltage (I_{OL}=0.5mA)		Vss		0.3	V
	General RF conditions					
f_{OP}	Operating frequency	1)	2400		2524	MHz
Δf	Frequency deviation			±156		kHz
R_{GFSK}	Data rate ShockBurst™		>0		1000	kbps
$F_{CHANNEL}$	Channel spacing			1		MHz
	Transmitter operation					
P_{RF}	Maximum Output Power	4)		0	+4	dBm
P_{RFC}	RF Power Control Range		16	20		dB
P_{RFCR}	RF Power Control Range Resolution				±3	dB
P_{BW}	20dB Bandwidth for Modulated Carrier				1000	kHz
P_{RF2}	2nd Adjacent Channel Transmit Power 2MHz				-20	dBm
P_{RF3}	3rd Adjacent Channel Transmit Power 3MHz				-40	dBm
I_{VDD}	Supply current @ 0dBm output power	5)		13		mA
I_{VDD}	Supply current @ -20dBm output power	5)		8.8		mA
I_{VDD}	Average Supply current @ -5dBm output power, ShockBurst™	6)		0.8		mA
I_{VDD}	Average Supply current in stand-by mode	7)		12		µA
I_{VDD}	Average Supply current in power down			1		µA
	Receiver operation					
I_{VDD}	Supply current one channel 250kbps			18		mA
I_{VDD}	Supply current one channel 1000kbps			19		mA
I_{VDD}	Supply current two channels 250kbps			23		mA
I_{VDD}	Supply current two channels 1000kbps			25		mA
RX_{SENS}	Sensitivity at 0.1%BER (@250kbps)			-90		dBm
RX_{SENS}	Sensitivity at 0.1%BER (@1000kbps)			-80		dBm
C/I_{CO}	C/I Co-channel			6		dB
C/I_{1ST}	1st Adjacent Channel Selectivity C/I 1MHz			-1		dB
C/I_{2ND}	2nd Adjacent Channel Selectivity C/I 2MHz			-16		dB
C/I_{3RD}	3rd Adjacent Channel Selectivity C/I 3MHz			-26		dB
RX_B	Blocking Data Channel 2			-41		dB

➢ Circuit Description:



➢ ShockBurst™

The ShockBurst™ technology uses on-chip FIFO to clock in data at a low data rate and transmit at a very high rate thus enabling extremely power reduction.

When operation the TRW-24G in ShockBurst™ ,you gain access to the high data rates(1 Mbps)offered by the 2.4GHz band without the need of a costly, high-speed micro controller (MCU) for data processing.

By putting all high speed signal processing related to RF protocol on-chip, the TRW-24G offers the following benefits:

- Highly reduced current consumption.
- Lower system cost (facilitates use of less expensive micro controller).
- Greatly reduced risk of 'on-air' collisions due to short transmission time.

The TRW-24G can be programmed using a simple 3-wire interface where the data rate is decided by the speed of the micro controller.

By allowing the digital part of the application to run at low speed while maximizing the data rate on the RF link, the nRF ShockBurst™ mode reduces the average current consumption in applications considerably.

➢ ShockBurst™ principle:

When the TRW-24G is configured in ShockBurst™, TX or RX operation is conducted in the following way (10 kbps for the example only).

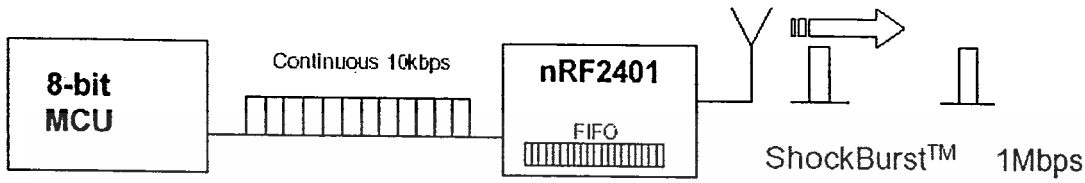


Figure 4 Clocking in data with MCU and sending with ShockBurst™ technology

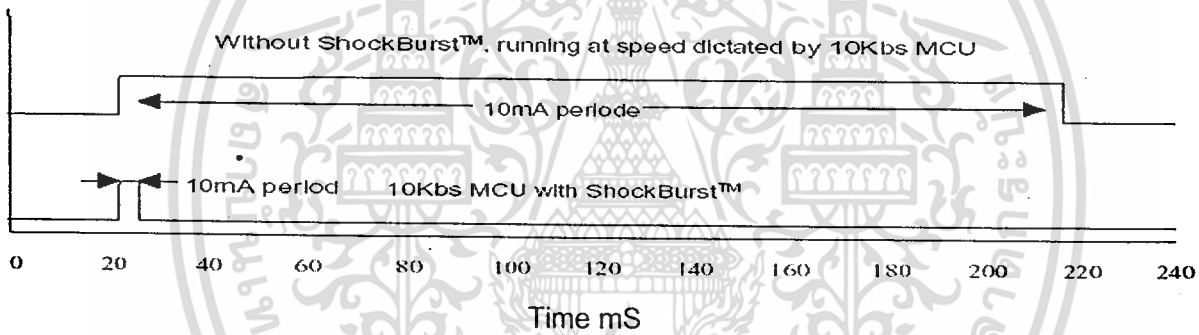


Figure1 Current consumption with & without ShockBurst™ technology

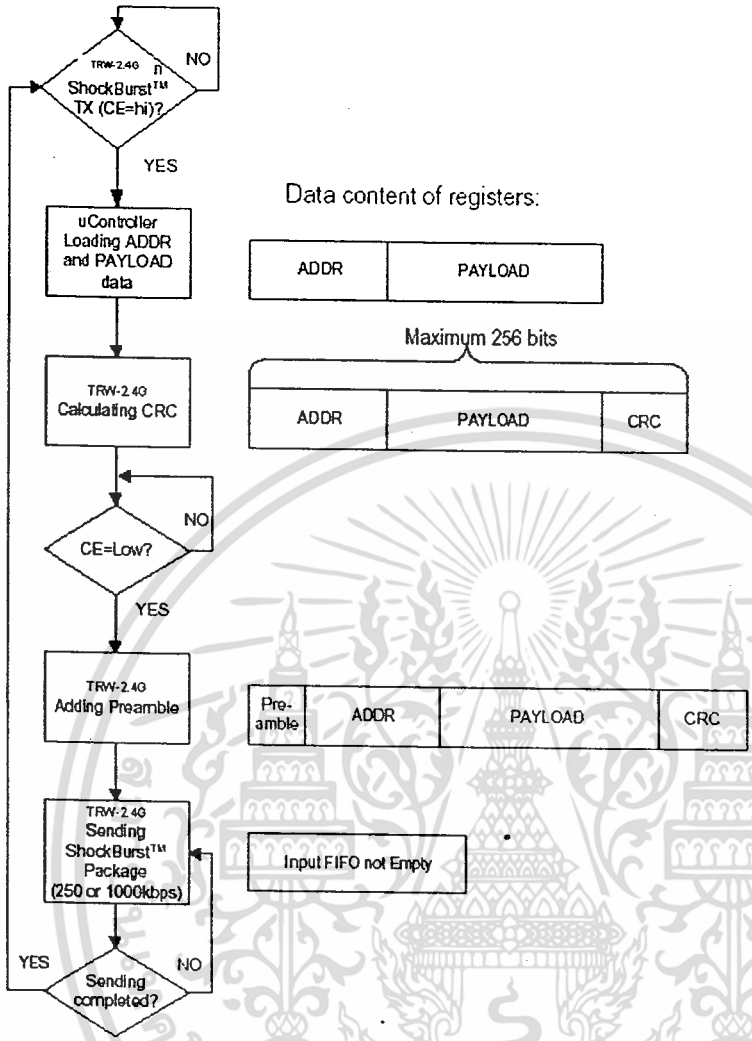


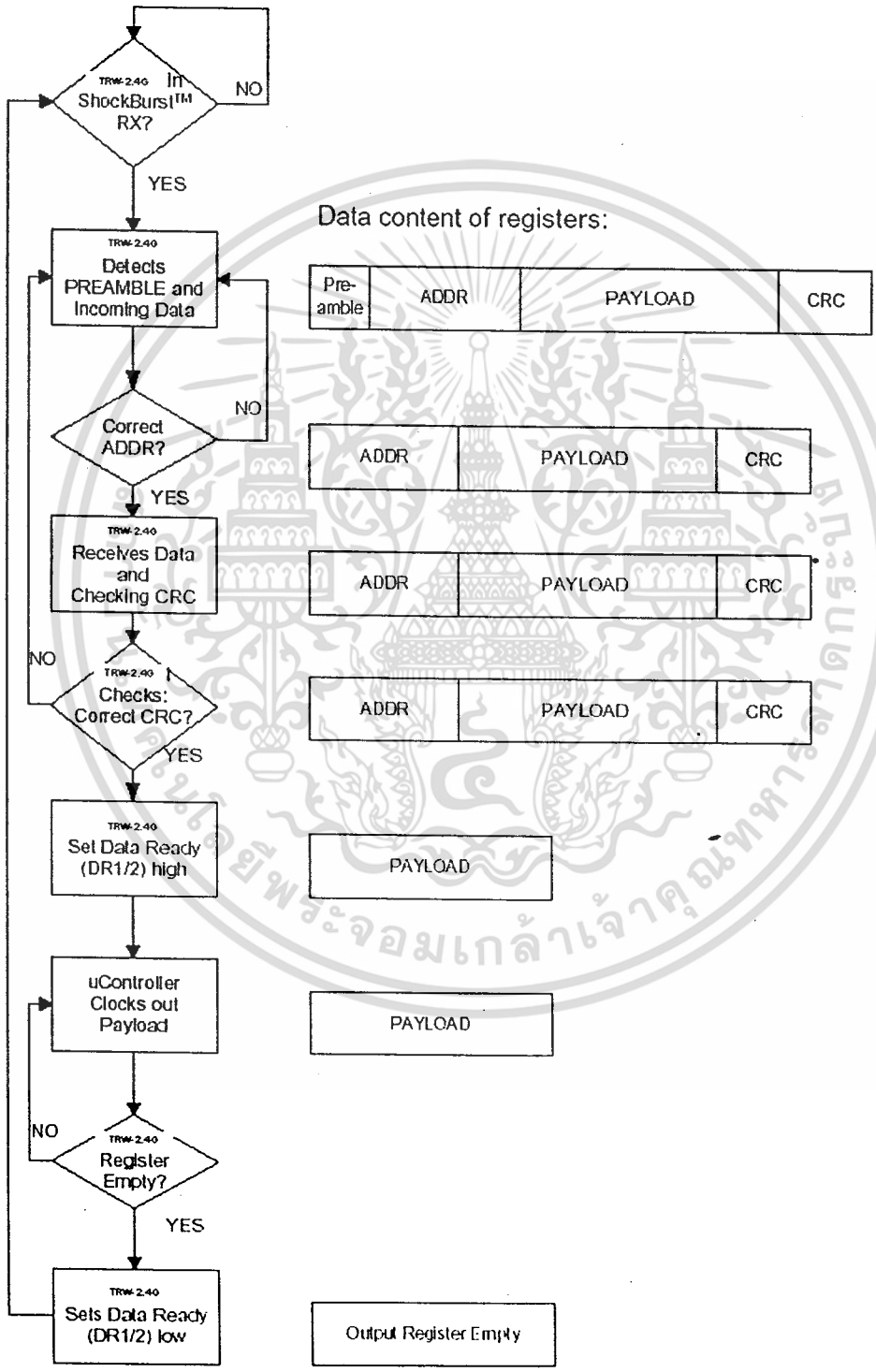
Figure 2 Flow Chart shockBurst™ Transmit of TRW-24G

➢ nRF2401 ShockBurst™ Transmit:

MCU interface pins:CE,CLK1,DATA

1. When the application MCU has data to send, set CE high. This activates TRW-24G on-board data processing.
2. The address of the receiving node(RX address) and payload data is clocked into the TRW-24G. The application protocol or MCU sets the speed <1Mbps(ex:10kbps)>.
3. MCU sets CE low, this activates a TRW-24G ShockBurst™ transmission.
4. TRW-24G ShockBurst™:

- RF front end is powered up.
- RF package is completed (preamble added, CRC calculated).
- Data is transmitted at high speed (250kbps or 1 Mbps configured by user).
- TRW-24G return to stand-by when finished.



➢ TRW-24G ShockBurst™ Receive:

MCU interface pins: CE, DR1, CLK1 and DATA (one RX channel receive)

1. Correct address and size of payload of incoming RF packages are set when TRW-24G is configured to ShockBurst™ RX.
2. To activate RX , set CE high.
3. After 200us settling, TRW-24G is monitoring the air for incoming communication.
4. When a valid package has been received (correct address and CRC found), TRW-24G removes the preamble, address and CRC bits.
5. TRW-24G then notifies (interrupts) the MCU by setting the DR1 pin high.
6. MCU may (or may not) set the CE low to disable the RF front end (low current mode).
7. The MCU will clock out just the payload data at a suitable rate (ex,10 kbps).
8. When all payload data is retrieved TRW-24G sets DR1 low again, and is ready for new incoming data package if CE is kept high during data download. If the CE was set low, a new start up sequence can begin, see Figure 12.

➢ Duoceiver™ Simultaneous Two Channel Receive Mode:

In both ShockBurst™ modes the TRW-24G can facilitate simultaneous reception of two parallel independent frequency channels at the maximum data rate.
This means:

- TRW-24G can receive data from two 1Mbps transmitters (ex: TRW-24G or TRW-24G) 8MHz (8 frequency channels) apart through one antenna interface.
- The output from the two data channels is fed to two separate MCU interfaces.
- Data channel 1:CLK1,DATA,and DR1
- Data channel 2:CLK2,DOUT2,and DR2
- DR1 and DR2 are available only in ShockBurst™.

The TRW-24G DuoCeiver™ technology provides 2 separate dedicated data channels for RX and replaces the need for two, stand alone receiver systems.

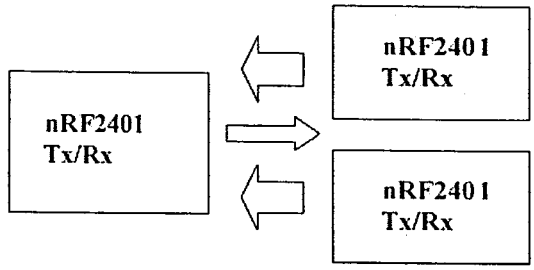


Figure 4 Simultaneous 2 channel receive on TRW-24G

There is one absolute requirement for using the second data channel. For the TRW-24G to be able to receive at the second data channel the frequency channel must be 8MHz higher than the frequency of data channel 1. The TRW-24G must be programmed to receive at the frequency of data channel 1. No time multiplexing is used in TRW-24G to fulfil this function. In direct mode the MCU must be able to handle two simultaneously incoming data packets if it is not multiplexing between the two data channels. In ShockBurst™ it is possible for the MCU to clock out one data channel at a time while data on the other data channel waits for MCU availability, without any lost data packets, and by doing so reduce the needed performance of the MCU.

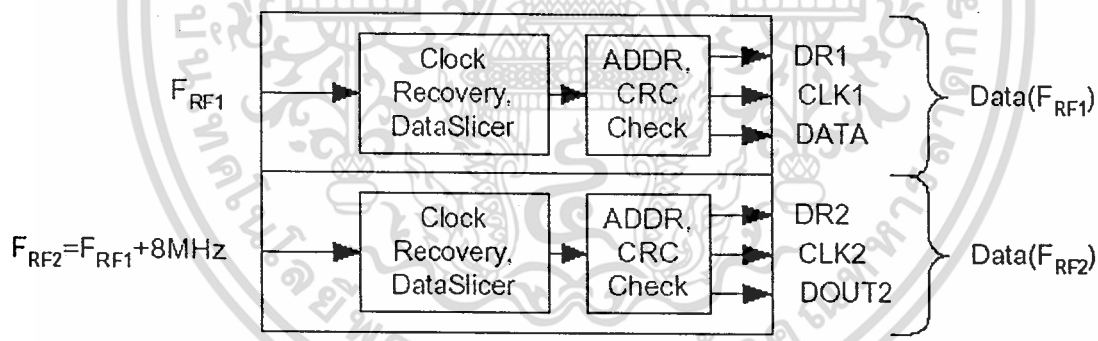


Figure 5 DuoCeiver™ with two simultaneously independent receive channels

➢ Device Configuration:

All configuration of theTRW-24G is done via 3-wire interface to a single configuration register. The configuration word can be up to 15 bytes long for ShockBurst™.

➢ Configuration or ShockBurst™ operation:

The configuration word in ShockBurst™ enables theTRW-24G to handle the RF protocol. Once the protocol is completed and loaded intoTRW-24G only one byte, bit [7:0], needs to be updated during actual operation.

The configuration blocks dedicated to ShockBurst™ is as follows:

- Payload section width: Specifies the number of payload bits in a RF package. This enables theTRW-24G to distinguish between payload data and the CRC bytes in a received package.
- Address width: Sets the number of bits used for address in the RF package, This enables theTRW-24G to distinguish between address and payload data.
- Address (RX Channel 1 and 2): Destination address for received data.
- CRC: Enables TRW-24G on-chip CRC generation and de-coding.

NOTE:

These configuration blocks, with the exception of the CRC, are dedicated for the packages that a TRW-24G is to receive.

In TX mode, the MCU must generate an address and a payload section that fits the configuration of theTRW-24G that is to receive the data.

When using theTRW-24G on-chip CRC feature ensure that CRC is enabled and uses the same length for both the TX and RX devices.

PRE-AMBLE	ADDRESS	PAYLOAD	CRC

Figure 10 Data packet set-up

➢ Configuration word overview:

	Bit position	Number of bits	Name	Function
ShockBurst™ configuration	143:120	24	TEST	Reserved for testing
	119:112	8	DATA2_W	Length of data payload section RX channel 2
	111:104	8	DATA1_W	Length of data payload section RX channel 1
	103:64	40	ADDR2	Up to 5 byte address for RX channel 2
	63:24	40	ADDR1	Up to 5 byte address for RX channel 1
	23:18	6	ADDR_W	Number of address bits (both RX channels).
	17	1	CRC_L	8 or 16 bit CRC
	16	1	CRC_EN	Enable on-chip CRC generation/checking.
General device configuration	15	1	RX2_EN	Enable two channel receive mode
	14	1	CM	Communication mode (Direct or ShockBurst™)
	13	1	RFDR_SB	RF data rate (1Mbps requires 16MHz crystal)
	12:10	3	XO_F	Crystal frequency
	9:8	2	RF_PWR	RF output power
	7:1	7	RF_CH#	Frequency channel
	0	1	RXEN	RX or TX operation

Table 1 Table of configuration words

The configuration word is shifted in MSB first on positive CLK1 edges, New configuration is enabled on the falling edge of CS.

NOTE:

On the falling edge of CS, theTRW-24G updates the number of bits actually shifted in during the last configuration.

Ex:

If theTRW-24G is to be configured for 2 channel RX in ShockBurst™, a total of 120 bits must be shifted in during the first configuration after VDD is applied.

Once the wanted protocol, modus and RF channel are set, only one bit (RXEN) is shifted in to switch between RX and TX.

➢ Configuration Word Detailed Description:

The following describes the function of the 144 bits (bit 143=MSB) that is used to configure the TRW-24G

General Device Configuration: bit [15:0]
ShockBurst™ Configuration: bit [119:0]
Test Configuration: bit [143:120]

MSB	TEST							
D143	D142	D141	D140	D139	D138	D137	D136	
Reserved for testing								
1	0	0	0	1	1	1	0	Default

MSB	TEST														
D135	D134	D133	D132	D131	D130	D129	D128	D127	D126	D125	D124	D123	D122	D121	D120	
Reserved for testing														Close PLL in TX		
0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	Default

DATA2_W								
D119	D118	D117	D116	D115	D114	D113	D112	
Data width channel#2 in # of bits excluding addr/crc								
0	0	1	0	0	0	0	0	Default

DATA1_W								
D111	D110	D109	D108	D107	D106	D105	D104	
Data width channel#1 in # of bits excluding addr/crc								
0	0	1	0	0	0	0	0	Default

ADDR2												
D103	D102	D101	....	D71	D70	D69	D68	D67	D66	D65	D64	
Channel#2 Address RX (up to 40bit)												
0	0	0	...	1	1	1	0	0	1	1	1	Default

ADDR1												
D63	D62	D61	....	D31	D30	D29	D28	D27	D26	D25	D24	
Channel#1 Address RX (up to 40bit)												
0	0	0	...	1	1	1	0	0	1	1	1	Default

ADDR_W						
D23	D22	D21	D20	D19	D18	
Address width in # of bits (both channels)						
0	0	1	0	0	0	Default

CRC		
D17	D16	
CRC Mode 1 – 16bit, 0 – 8bit	CRC 1 – enable; 0 – disable	
0	1	Default

RF-Programming															LSB	
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
Two Ch.	BUF	OD	XO Frequency			RF Power		Channel selection							RXEN	
0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	Default

Table 2 Configuration data word

The MSB bit should be loaded first into the configuration register.
Default configuration word:
h8E08.1C20.2000.0000.00E7.0000.0000.E721.0F04.

➢ ShockBurst™ configuration:

The section B[119:16] contains the segments of the configuration register dedicated to ShockBurst™ operational protocol. After VDD is turned on ShockBurst™ configuration is done once and remains set whilst VDD is present, During operation only the first byte for frequency channel and RX/TX switching need to be changed.

PLL_CTRL

PLL_CTRL		
D121	D120	PLL
0	0	Open TX/Closed RX
0	1	Open TX/Open RX
1	0	Closed TX/Closed RX
1	1	Closed TX/Open RX

Table 10 PLL setting

Bit 121-120:
PLL_CTRL: Controls the setting of the PLL for test purposes. With closed PLL in TX no deviation will be present.

DATAx_W

DATA2_W							
119	118	117	116	115	114	113	112

DATA1_W							
111	110	109	108	107	106	105	104

Table 4 Number of bits in payload

Bit 119-112:
DATA2_W: Length of RF package payload section for receive-channel 2.
Bit 111-104:
DATA1_W: Length of RF package payload section for receive-channel 1.

NOTE:
The total number of bits in a ShockBurst™ RF package may not exceed 256!
Maximum length of payload section is hence given by:

DATAx_W(bits)=256-ADDR_W-CRC

Where:
ADDR_W: length of RX address set in configuration word B [23:18]
CRC: check sum, 8 or 16 bits set in configuration word B [17]
PRE: preamble, 4 or 8 bits are automatically included
Shorter address and CRC leaves more room for payload data in each package.

ADDRx

ADDR2											
103	102	101	....	71	70	69	68	67	66	65	64

ADDR1											
63	62	61	....	31	30	29	28	27	26	25	24

Table 5 Address of receiver #2 and receiver #1

Bit 103-64:
ADDR2: Receiver address channel 2, up to 40 bit.

Bit 63-24:
ADDR1: Receiver address channel 1, up to 40 bit.

NOTE:
Bits in ADDRx exceeding the address width set in ADDR_W are redundant and can be set to logic 0.

ADDR_W & CRC

			ADDR_W			CRC_L	CRC_EN
23	22	21	20	19	18	17	16

Table 6 Number of bits reserved for RX address + CRC setting

Bit 23-18:
ADDR_W: Number of bits reserved for RX address in ShockBurst™ packages.

NOTE:
Maximum number of address bits is 40 (5 bytes). Values over 40 in ADDR_W are not valid.

Bit 17:
CRC_L: CRC length to be calculated by TRW-24G in ShockBurst™.
Logic 0: 8 bit CRC
Logic 1: 16 bit CRC

Bit 16:
CRC_EN: Enables on-chip CRC generation (TX) and verification (RX).
Logic 0: On-chip CRC generation/checking disabled
Logic 1: On-chip CRC generation/checking enabled

NOTE:
An 8 bit CRC will increase the number of payload bits possible in each ShockBurst™ data packet, but will also reduce the system integrigy.

➢ General device configuration:

This section of the configuration word handles RF and device related parameters.

Modes:

RX2_EN	CM	RFDR_SB	XO_F			RF_PWR	
15	14	13	12	11	10	9	8

Table 7 RF operational settings

Bit 15:

RX2_EN:
Logic 0: One channel receive
Logic 1: Two channels receive

NOTE:
In two channels receive, the TRW-24G receives on two, separate frequency channels simultaneously. The frequency of receive channel 1 is set in the configuration word B[7-1], receive channel 2 is always 8 channels (8 MHz) above receive channel 1.

Bit 14:

Communication Mode:
Logic 1: nRF2401 operates in ShockBurst™ mode

Bit 13:

RF Data Rate:
Logic 0: 250 kbps
Logic 1: 1 Mbps

NOTE:
Utilizing 250 kbps instead of 1 Mbps will improve the receiver sensitivity by 10 dB. 1 Mbps requires 16MHz crystal.

Bit 12-10:

D12	D11	D10
0	1	1

Table 8

Bit 9-8:
RF_PWR: Sets TRW-24G RF output power in transmit mode:

RF OUTPUT POWER		
D9	D8	P [dBm]
0	0	-20
0	1	-10
1	0	-5
1	1	0

Table 9 RF output power setting

➢ RF channel & direction:

RF_CH#							RXEN
7	6	5	4	3	2	1	0

Table 10 Frequency channel + RX/TX setting

Bit 7-1:
RF_CH#: Sets the frequency channel the nRF2401 operates on.

The channel frequency in ***transmit*** is given by:

$Channel_{RF} = 2400\ MHz + RF\_CH\# \cdot 1.0\ MHz$
RF_CH # : between 2400MHz and 2527MHz may be set.

The channel frequency in ***data channel 1*** is given by:

$Channel_{RF} = 2400\ MHz + RF\_CH\# \cdot 1.0\ MHz$ (Receive at PIN#8)
RF_CH # : between 2400MHz and 2524MHz may be set.

NOTE:
The channels above 83 can only be utilized in certain territories (ex: Japan)

The channel frequency in ***data channel 2*** is given by:

$Channel_{RF} = 2400\ MHz + RF\_CH\# \cdot 1.0\ MHz$ + 8MHz (Receive at PIN#4)
RF_CH # : between 2408MHz and 2524MHz may be set.

Bit 0:
Set active mode:
Logic 0: transmit mode
Logic 1: receive mode

The data packet for both ShockBurst™ mode and direct mode communication is divided into 4 sections. These are:

1. PREAMBLE	· The preamble field is required in ShockBurst.
2. ADDRESS	· The address field is required in ShockBurst. mode. · 8 to 40 bits length. · Address automatically removed from received packet in ShockBurst.mode
3. PAYLOAD	· The data to be transmitted · In Shock-Burst mode payload size is 256 bits minus the Following :(Address: 8 to 40 bits. + CRC 8 or 16 bits).
4. CRC	· 8 or 16 bits length · The CRC is stripped from the received output data.

Data Package Description:

Figure 7 Data Package Diagram

➢ TRW-24G configuration data is from a high level start.

Example: In ShockBurth launching mode, it is to a passage in the 2410MHz to 1Mbps Rate transmission

Bit143	Bit142	Bit141	Bit140	Bit139	Bit138	Bit137	Bit136
1	0	0	0	1	1	1	0
Bit135	Bit134	Bit133	Bit132	Bit131	Bit130	Bit129	Bit128
0	0	0	0	1	0	0	0
Bit127	Bit126	Bit125	Bit124	Bit123	Bit122	Bit121	Bit120
0	0	0	1	1	1	0	0
Bit119	Bit118	Bit117	Bit116	Bit115	Bit114	Bit113	Bit112
1	1	0	0	1	0	0	0
Bit111	Bit110	Bit109	Bit108	Bit107	Bit106	Bit105	Bit104
1	1	0	0	1	0	0	0
Bit103	Bit102	Bit101	Bit100	Bit99	Bit98	Bit97	Bit96
1	1	0	0	0	0	0	0
Bit95	Bit94	Bit93	Bit92	Bit91	Bit90	Bit89	Bit88
1	0	1	0	1	0	1	0
Bit87	Bit86	Bit85	Bit84	Bit83	Bit82	Bit81	Bit80
0	1	0	1	0	1	0	1
Bit79	Bit78	Bit77	Bit76	Bit75	Bit74	Bit73	Bit72
1	0	1	0	1	0	1	0
Bit71	Bit70	Bit69	Bit68	Bit67	Bit66	Bit65	Bit64
0	1	0	1	0	1	0	1
Bit63	Bit62	Bit61	Bit60	Bit59	Bit58	Bit57	Bit56
1	0	1	0	1	0	1	0
Bit55	Bit54	Bit53	Bit52	Bit51	Bit50	Bit49	Bit48
0	1	0	1	0	1	0	1
Bit47	Bit46	Bit45	Bit44	Bit43	Bit42	Bit41	Bit40
1	0	1	0	1	0	1	0
Bit39	Bit38	Bit37	Bit36	Bit35	Bit34	Bit33	Bit32
0	1	0	1	0	1	0	1
Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
1	0	1	0	1	0	1	0
Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
1	0	1	0	0	0	1	1
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
0	1	1	0	1	1	1	1
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	1	0	1	0	0

Example: ShockBurth in the receive mode, it is to a passage in the 2410MHz to 1Mbps Rate reception.

Bit143	Bit142	Bit141	Bit140	Bit139	Bit138	Bit137	Bit136
1	0	0	0	1	1	1	0
Bit135	Bit134	Bit133	Bit132	Bit131	Bit130	Bit129	Bit128
0	0	0	0	1	0	0	0
Bit127	Bit126	Bit125	Bit124	Bit123	Bit122	Bit121	Bit120
0	0	0	1	1	1	0	0
Bit119	Bit118	Bit117	Bit116	Bit115	Bit114	Bit113	Bit112
1	1	0	0	1	0	0	0
Bit111	Bit110	Bit109	Bit108	Bit107	Bit106	Bit105	Bit104
1	1	0	0	1	0	0	0
Bit103	Bit102	Bit101	Bit100	Bit99	Bit98	Bit97	Bit96
1	1	0	0	0	0	0	0
Bit95	Bit94	Bit93	Bit92	Bit91	Bit90	Bit89	Bit88
1	0	1	0	1	0	1	0
Bit87	Bit86	Bit85	Bit84	Bit83	Bit82	Bit81	Bit80
0	1	0	1	0	1	0	1
Bit79	Bit78	Bit77	Bit76	Bit75	Bit74	Bit73	Bit72
1	0	1	0	1	0	1	0
Bit71	Bit70	Bit69	Bit68	Bit67	Bit66	Bit65	Bit64
0	1	0	1	0	1	0	1
Bit63	Bit62	Bit61	Bit60	Bit59	Bit58	Bit57	Bit56
1	0	1	0	1	0	1	0
Bit55	Bit54	Bit53	Bit52	Bit51	Bit50	Bit49	Bit48
0	1	0	1	0	1	0	1
Bit47	Bit46	Bit45	Bit44	Bit43	Bit42	Bit41	Bit40
1	0	1	0	1	0	1	0
Bit39	Bit38	Bit37	Bit36	Bit35	Bit34	Bit33	Bit32
0	1	0	1	0	1	0	1
Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
1	0	1	0	1	0	1	0
Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
1	0	1	0	0	0	1	1
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
0	1	1	0	1	1	1	1
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	1	0	1	0	1

➢ Important Timing Data:

The following timing applies for operation TRW-24G.

TRW-24G Timing Information:

nRF2401 timing	Max.	Min.	Name
PWR_DWN → ST_BY mode	3ms		Tpd2sby
PWR_DWN→ Active mode (RX/TX)	3ms		Tpd2a
ST_BY → TX ShockBurst™	195µs		Tsby2txSB
ST_BY → TX Direct Mode	202µs		Tsby2txDM
ST_BY → RX mode	202µs		Tsby2rx
Minimum delay from CS to data.		5µs	Tcs2data
Minimum delay from CE to data.		5µs	Tce2data
Minimum delay from DR1/2 to clk.		50ns	Tdr2clk
Maximum delay from clk to data.	50ns		Tclk2data
Delay between edges		50ns	Td
Setup time		500ns	Ts
Hold time		500ns	Th
Delay to finish internal GFSK data		1/data rate	Tfd
Minimum input clock high		500ns	Thmin
Set-up of data in Direct Mode	50ns		Tsdm
Minimum clock high in Direct Mode		300ns	Thdm
Minimum clock low in Direct Mode		230ns	Tldm

Table 11 Switching times for TRW-24G

When the TRW-24G is in power down it must always settle in stand-by(Tpd2sby) before it can enter configuration or one of the active modes.

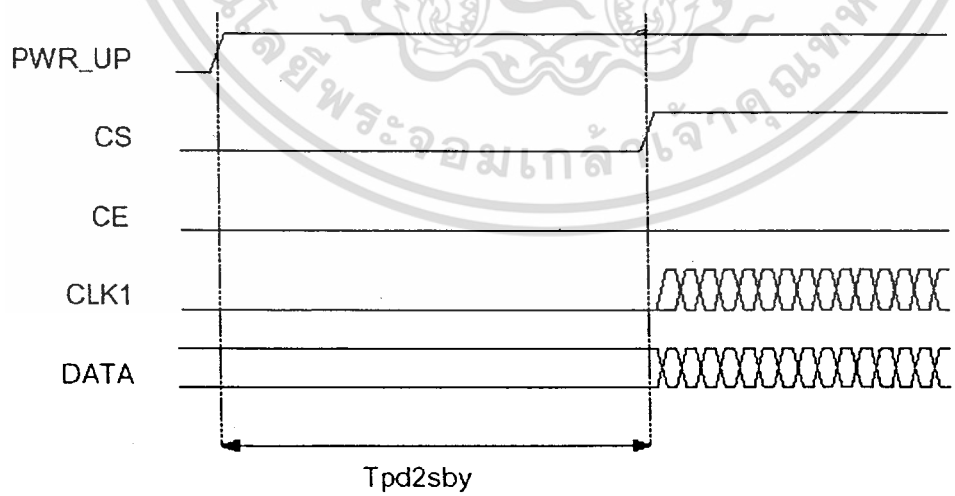


Figure 8 Timing diagram for TRW-24G (or VDD off) to stand by mode.

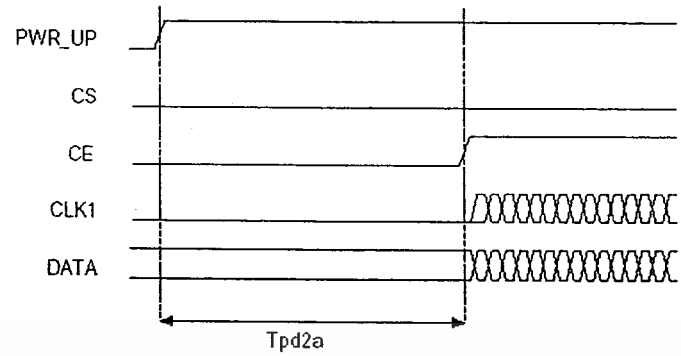


Figure 9 VDD off to active mode

Note that the configuration word will be lost when VDD is turned off and that the device then must be configured before going to one of the active modes. If the device is configured one can go directly from power down to the wanted active mode.

NOTE:

CE and CS may not be high at the same time. Setting one or the other decides whether configuration or active mode is entered.

➢ Configuration mode timing:

When one or more of the bits in the configuration word needs to be changed the following timing apply.

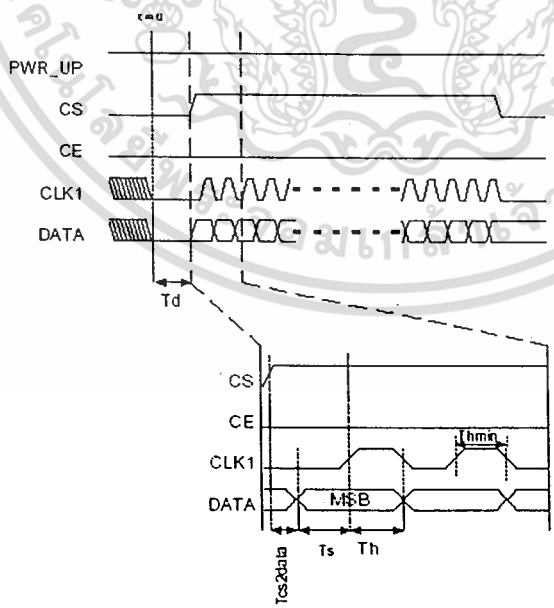


Figure 10 Timing diagram for configuration of TRW-24G

If configuration mode is entered from power down, CS can be set high after Tpd2sby as shown in Figure 8

➢ ShockBurst™ Mode timing:

ShockBurst™ TX:

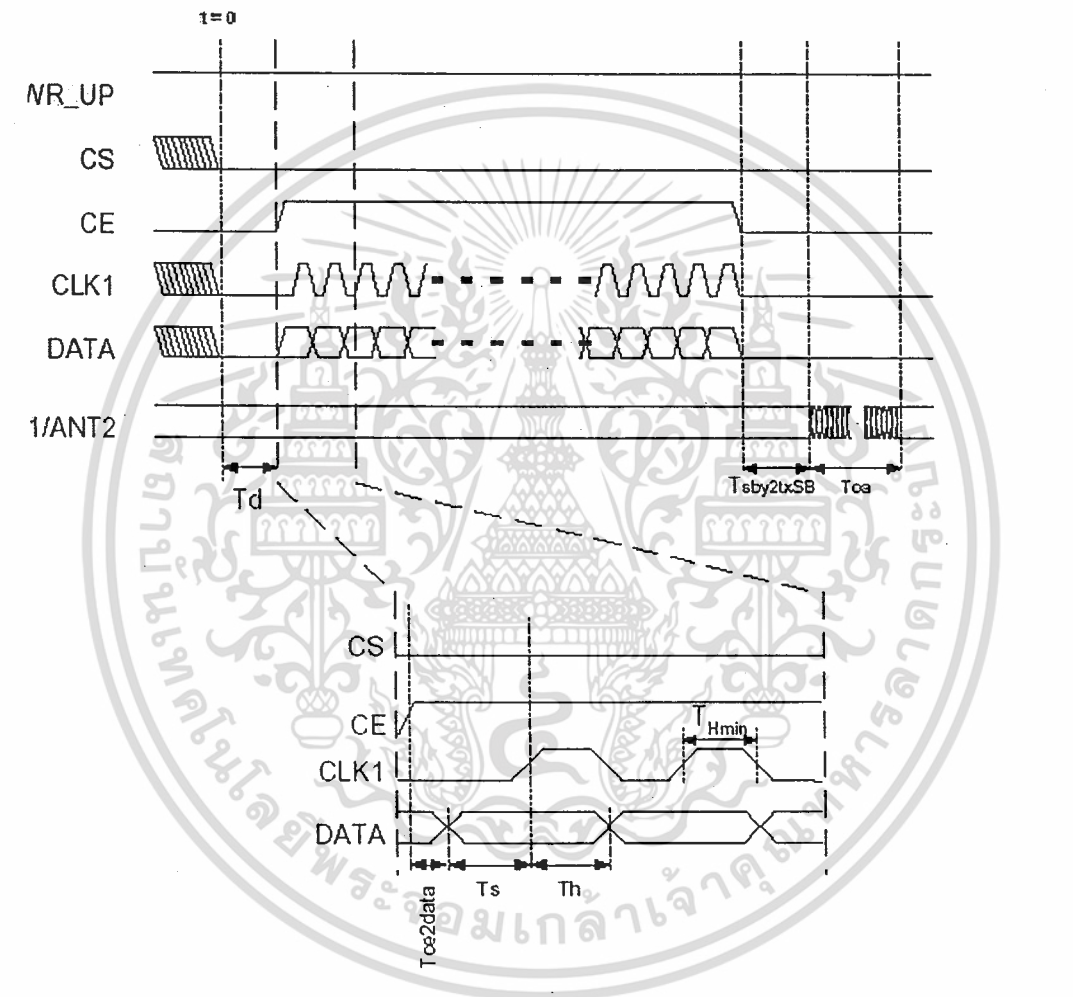


Figure 11 Timing of ShockBurst™ in TX

The package length and the data rate give the delay Toa (time on air), as shown in the equation.

$$T_{OA} = 1/\,datarate \cdot (\#databits + 1)$$

➢ ShockBurst™ RX:

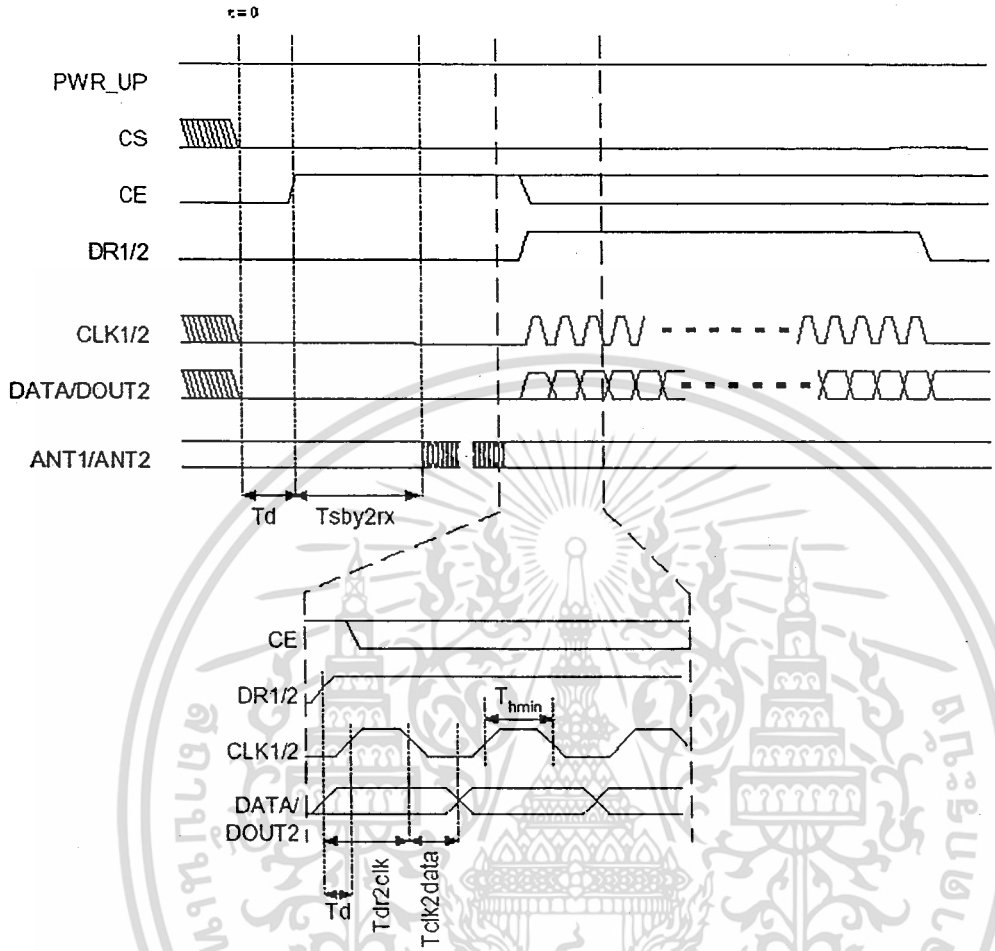


Figure [12] Timing of ShockBurst™ in RX

The CE may be kept high during downloading of data, but the cost is higher current consumption (18mA) and the benefit is no start-up time(200μs) after the DR1 goes low.

➢ Output Power adjustment:

Power setting bits of configuring word	RF output power	DC current consumption
11	0 dBm ±3dB	13.0 mA
10	-5 dBm ±3dB	10.5 mA
01	-10 dBm ±3dB	9.4 mA
00	-20 dBm ±3dB	8.8 mA

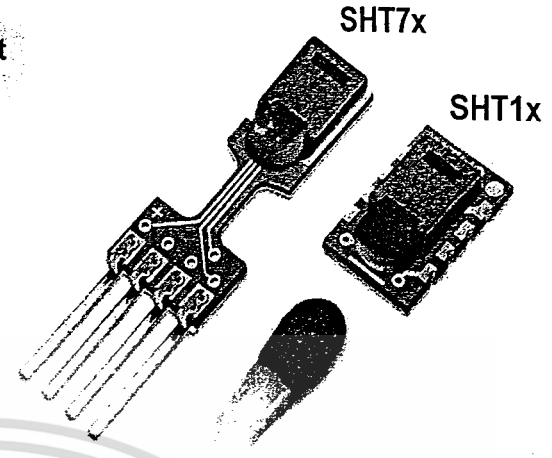
Conditions: VDD= 3.0V, VSS= 0V, T_A= 27℃, Load impedance =400Ω

SENSIRION
THE SENSOR COMPANY

SHT1x / SHT7x

Humidity & Temperature Sensor

Evaluation Kit Available



- Relative humidity and temperature sensors
- Dew point
- Fully calibrated, digital output
- Excellent long-term stability
- No external components required
- Ultra low power consumption
- Surface mountable or 4-pin fully interchangeable
- Small size
- Automatic power down

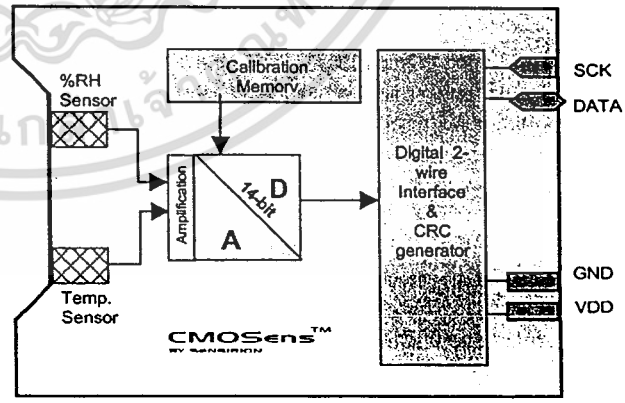
SHT1x / SHT7x Product Summary

The SHTxx is a single chip relative humidity and temperature multi sensor module comprising a calibrated digital output. Application of industrial CMOS processes with patented micro-machining (CMOSens® technology) ensures highest reliability and excellent long term stability. The device includes a capacitive polymer sensing element for relative humidity and a bandgap temperature sensor. Both are seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit on the same chip. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC) at a very competitive price. Each SHTxx is individually calibrated in a precision humidity chamber with a chilled mirror hygrometer as reference. The calibration coefficients are programmed into the OTP memory. These coefficients are used internally during measurements to calibrate the signals from the sensors.
The 2-wire serial interface and internal voltage regulation allows easy and fast system integration. Its tiny size and low power consumption makes it the ultimate choice for even the most demanding applications.
The device is supplied in either a surface-mountable LCC (Leadless Chip Carrier) or as a pluggable 4-pin single-in-line type package. Customer specific packaging options may be available on request.

Applications

_ HVAC
_ Automotive
_ Consumer Goods
_ Weather Stations
_ Humidifiers
_ Dehumidifiers
_ Test & Measurement
_ Data Logging
_ Automation
_ White Goods
_ Medical

Block Diagram



Ordering Information

Part Number	Humidity accuracy [%RH]	Temperature accuracy [K] @ 25 °C	Package
SHT11	±3.0	±0.4	SMD (LCC)
SHT15	±2.0	±0.3	SMD (LCC)
SHT71	±3.0	±0.4	4-pin single-in-line
SHT75	±1.8	±0.3	4-pin single-in-line

1 Sensor Performance Specifications

Parameter	Conditions	Min.	Typ.	Max.	Units
Humidity					
Resolution (2)		0.5	0.03	0.03	%RH
		8	12	12	bit
Repeatability			±0.1		%RH
Accuracy (1) Uncertainty	linearized	see figure 1			
Interchangeability		Fully interchangeable			
Nonlinearity	raw data		±3		%RH
	linearized		<<1		%RH
Range		0		100	%RH
Response time	1/e (63%) slowly moving air		4		s
Hysteresis			±1		%RH
Long term stability	typical		< 0.5		%RH/yr
Temperature					
Resolution (2)		0.04	0.01	0.01	°C
		0.07	0.02	0.02	°F
		12	14	14	bit
Repeatability			±0.1		°C
			±0.2		°F
Accuracy		see figure 1			
Range		-40		123.8	°C
		-40		254.9	°F
Response Time	1/e (63%)	5		30	s

Table 1 Sensor Performance Specifications

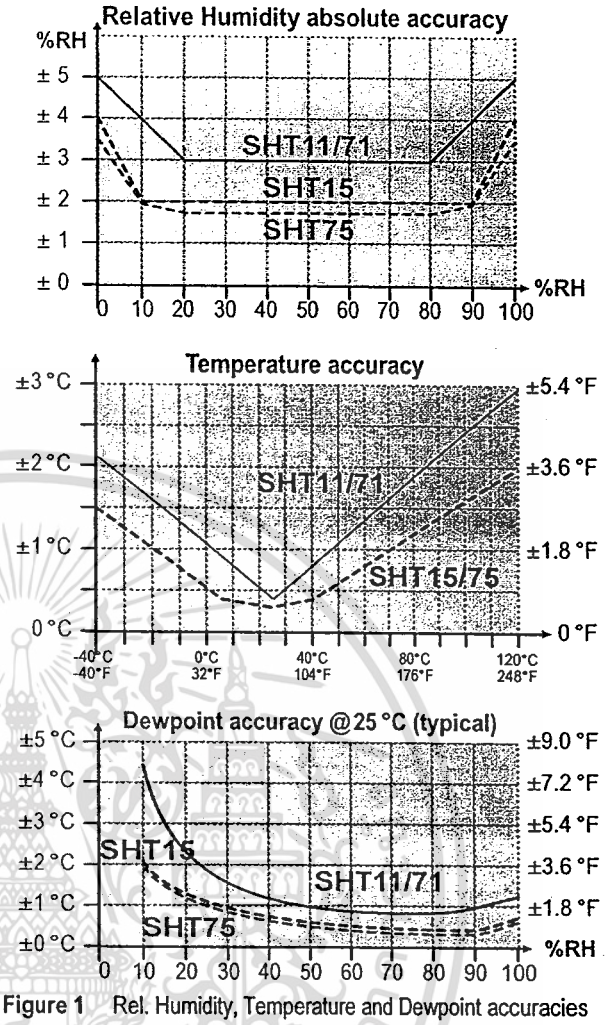


Figure 1 Rel. Humidity, Temperature and Dewpoint accuracies

2 Interface Specifications

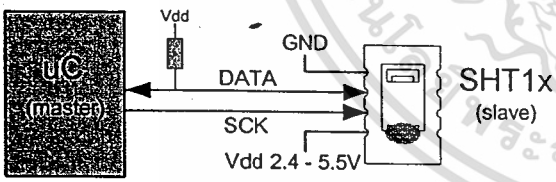


Figure 2 Typical application circuit

2.1 Power Pins

The SHTxx requires a voltage supply between 2.4 and 5.5 V. After powerup the device needs 11ms to reach its "sleep" state. No commands should be sent before that time.
Power supply pins (VDD, GND) may be decoupled with a 100 nF capacitor.

2.2 Serial Interface (Bidirectional 2-wire)

The serial interface of the SHTxx is optimized for sensor readout and power consumption and is not compatible with I²C interfaces, see FAQ for details.

2.2.1 Serial clock input (SCK)

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

2.2.2 Serial data (DATA)

The DATA tristate pin is used to transfer data in and out of the device. DATA **changes after the falling edge** and is **valid on the rising edge** of the serial clock SCK. During transmission the DATA line must remain stable while SCK is high. To avoid signal contention the microcontroller should only drive DATA low. An external pull-up resistor (e.g. 10 kΩ) is required to pull the signal high. (See Figure 2) Pull-up resistors are often included in I/O circuits of microcontrollers. See Table 5 for detailed IO characteristics.

(1) Each SHTxx is tested to be fully within RH accuracy specifications at 25 °C (77 °F) and 48 °C (118.4 °F)
(2) The default measurement resolution of 14bit (temperature) and 12bit (humidity) can be reduced to 12 and 8 bit through the status register.

2.2.3 *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.

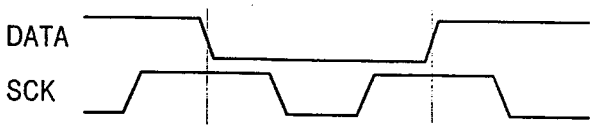


Figure 3 "Transmission Start" sequence

The subsequent command consists of three address bits (only "000" is currently supported) and five command bits. The SHTxx 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 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 2 SHTxx list of commands

2.2.4 *Measurement sequence (RH and T)*

After issuing a measurement command ('00000101' for RH, '00000011' for Temperature) the controller has to wait for the measurement to complete. This takes approximately 11/55/210 ms for a 8/12/14bit measurement. The exact time varies by up to ±15% with the speed of the internal oscillator. To signal the completion of a measurement, the SHTxx pulls down the data line 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 as convenient.

Two bytes of measurement data and one byte of CRC checksum will then be transmitted. The uC 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 the measurement and communication have ended.

Warning: To keep self heating below 0.1 °C the SHTxx should not be active for more than 10% of the time (e.g. max. 2 measurements / second for 12bit accuracy).

2.2.5 *Connection reset sequence*

If communication with the device is lost the following signal sequence will reset its serial interface:
While leaving DATA high, toggle SCK 9 or more times. This must be followed by a "Transmission Start" sequence preceding the next command. This sequence resets the interface only. The status register preserves its content.

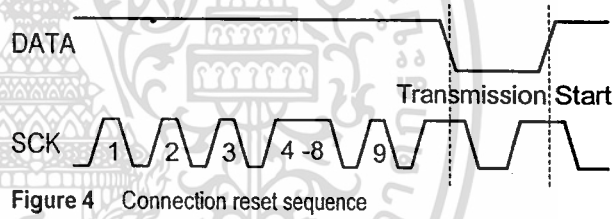


Figure 4 Connection reset sequence

2.2.6 *CRC-8 Checksum calculation*

The whole digital transmission is secured by a 8 bit checksum. It ensures that any wrong data can be detected and eliminated.
Please consult application note "CRC-8 Checksum Calculation" for information on how to calculate the CRC.

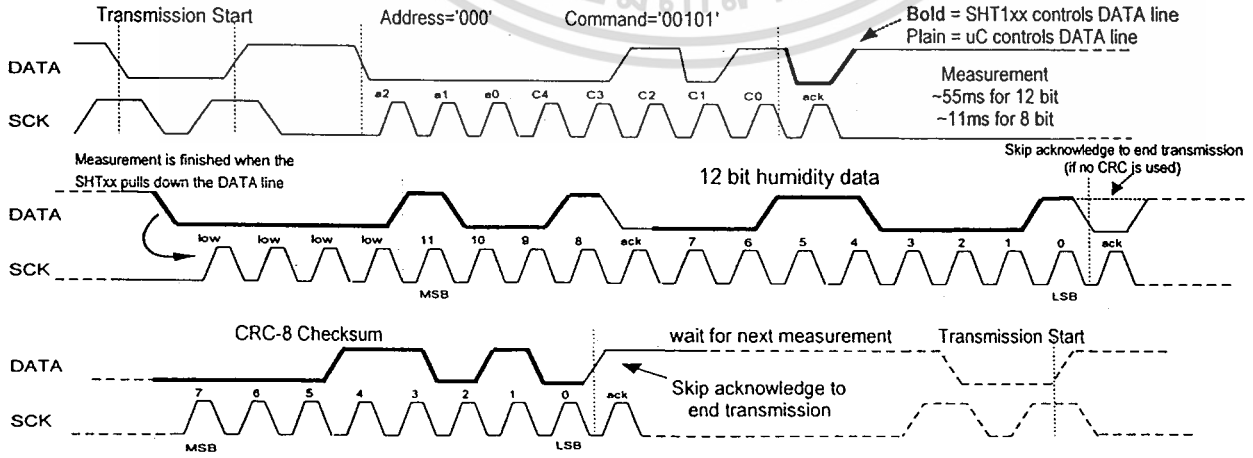


Figure 5 Example RH measurement sequence for value "0000'1001 ' 0011'0001"= 2353 = 75.79 %RH (without temperature compensation)

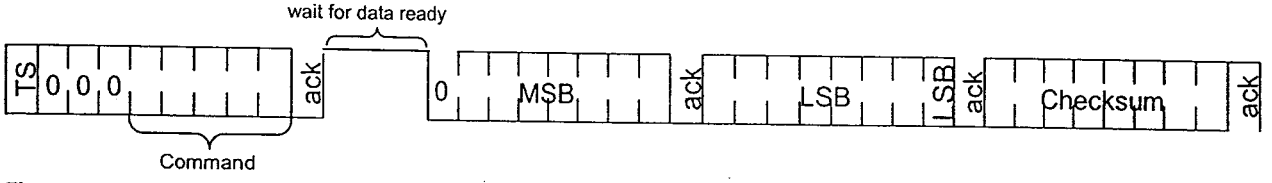


Figure 6 Overview of Measurement Sequence (TS = Transmission Start)

2.3 Status Register

Some of the advanced functions of the SHTxx are available through 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"

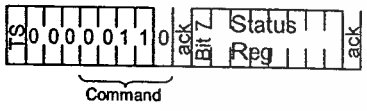


Figure 7 Status Register Write

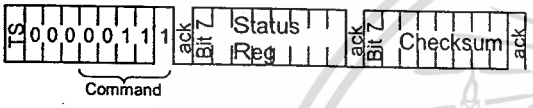


Figure 8 Status Register Read

Bit	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 Temperature resolution '0' = 12bit RH / 14bit Temperature resolution	0	12bit RH 14bit Temp.

Table 3 Status Register Bits

2.3.1 *Measurement resolution*

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

2.3.2 *End of Battery*

The "End of Battery" function detects VDD voltages below 2.47 V. Accuracy is ±0.05 V

2.3.3 *Heater*

An on chip heating element can be switched on. It will increase the temperature of the sensor by 5-15 °C (9-27 °F). Power consumption will increase by ~8 mA @ 5 V.
Applications:
By comparing temperature and humidity values before and after switching on the heater, proper functionality of both sensors can be verified.

- In high (>95 %RH) RH environments heating the sensor element will prevent condensation, improve response time and accuracy

Warning: While heated the SHTxx will show higher temperatures and a lower relative humidity than with no heating.

2.4 Electrical Characteristics[1]

VDD=5V, Temperature = 25 °C unless otherwise noted

Parameter	Conditions	Min.	Typ.	Max.	Units
Power supply DC		2.4	5	5.5	V
Supply current	measuring		550		µA
	average	2[2]	28[3]		µA
	sleep		0.3	1	µA
Low level output voltage		0		20%	Vdd
High level output voltage		75%		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 peak current	on			4	mA
	Tristated (off)		10		µA

Table 4 SHTxx DC Characteristics

	Parameter	Conditions	Min	Typ.	Max.	Unit
F_{SCK}	SCK frequency	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
T_{RFO}	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load 100 pF	30	40	200	ns
T_{CLx}	SCK hi/low time		100			ns
T_V	DATA valid time			250		ns
T_{SU}	DATA set up time		100			ns
T_{HO}	DATA hold time		0	10		ns
T_R/T_F	SCK rise/fall time			200		ns

Table 5 SHTxx I/O Signals Characteristics

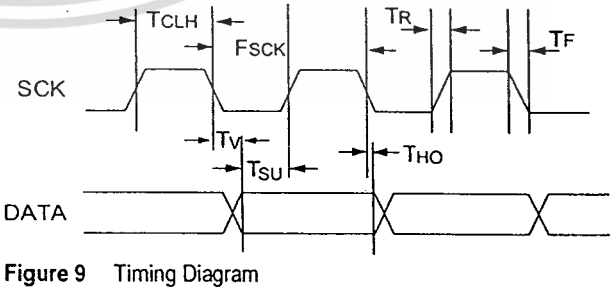


Figure 9 Timing Diagram

1) Parameters are periodically sampled and not 100% tested
2) With one measurement of 8 bit accuracy without OTP reload per second
3) With one measurement of 12bit accuracy per second

3 Converting Output to Physical Values

3.1 Relative Humidity

To compensate for the non-linearity of the humidity sensor and to obtain the full accuracy it is recommended to convert the readout with the following formula[1]:

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^2$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

Table 6 Humidity conversion coefficients

For simplified, less computation intense conversion formulas see application note "RH and Temperature Non-Linearity Compensation".

The humidity sensor has no significant voltage dependency.

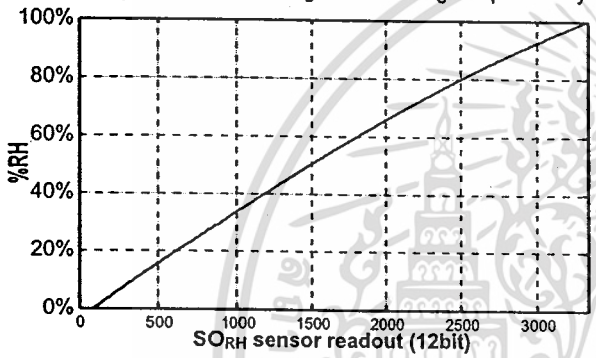


Figure 10 Conversion from SO_{RH} to relative humidity

3.1.1 Humidity Sensor RH/Temperature compensation

For temperatures significantly different from 25 °C (~77 °F) the temperature coefficient of the RH sensor should be considered:

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

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

Table 7 Temperature compensation coefficients

This equals ~0.12 %RH / °C @ 50 %RH

3.2 Temperature

The bandgap PTAT (Proportional To Absolute Temperature) temperature sensor is very linear by design. Use the following formula to convert from digital readout to temperature:

$$Temperature = d_1 + d_2 \bullet SO_T$$

VDD	d_1 [°C]	d_1 [°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

SO_T	d_2 [°C]	d_2 [°F]
14bit	0.01	0.018
12bit	0.04	0.072

Table 8 Temperature conversion coefficients

For improved accuracies in extreme temperatures with more computation intense conversion formulas see application note "RH and Temperature Non-Linearity Compensation".

3.3 Dewpoint

Since humidity and temperature are both measured on the same monolithic chip, the SHTxx allows superb dewpoint measurements. See application note "Dewpoint calculation" for more.

[1] Where SO_{RH} is the sensor output for relative humidity

4 Applications Information

4.1 Operating and Storage Conditions

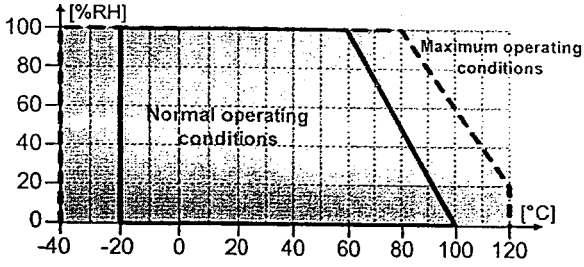


Figure 11 Recommended operating conditions

Conditions outside the recommended range may temporarily offset the RH signal up to ±3 %RH. After return to normal conditions it will slowly return towards calibration state by itself. See 4.3 "Reconditioning Procedure" to accelerate this process. Prolonged exposure to extreme conditions may accelerate ageing.

4.2 Exposure to Chemicals

Chemical vapors may interfere with the polymer layers used for capacitive humidity sensors. The diffusion of chemicals into the polymer may cause a shift in both offset and sensitivity. In a clean environment the contaminants will slowly outgas. The reconditioning procedure described below will accelerate this process. High levels of pollutants may cause permanent damage to the sensing polymer.

4.3 Reconditioning Procedure

The following reconditioning procedure will bring the sensor back to calibration state after exposure to extreme conditions or chemical vapors.
80-90 °C (176-194°F) at < 5 %RH for 24h (baking) followed by
20-30 °C (70-90°F) at > 74 %RH for 48h (re-hydration)

4.4 Temperature Effects

The relative humidity of a gas strongly depends on its temperature. It is therefore essential to keep humidity sensors at the same temperature as the air of which the relative humidity is to be measured.
If the SHTxx shares a PCB with electronic components that give off heat it should be mounted far away and below the heat source and the housing must remain well ventilated.
To reduce heat conduction copper layers between the SHT1x and the rest of the PCB should be minimized and a slit may be milled in between (see figure 13).

4.5 Membranes

A membrane may be used to prevent dirt from entering the housing and to protect the sensor. It will also reduce peak concentrations of chemical vapors. For optimal response times air volume behind the membrane must be kept to a minimum. For the SHT1x package Sensirion recommends the SF1 filter cap for optimal IP67 protection.

4.6 Light

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

4.7 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: All 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. Outgassing of these materials may also contaminate the SHTxx (cf. 4.2). Store well ventilated after manufacturing or bake at 50°C for 24h to outgas contaminants before packing.

4.8 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.
Please see the application note "ESD, Latchup and EMC" for more information.
Power supply pins (VDD, GND) should be decoupled with a 100 nF capacitor if wires are used.

4.9 Qualifications

Extensive tests were performed in various environments. Please contact SENSIRION for detailed information.

Environment	Norm	Results[1]
Temperature Cycles	JESD22-A104-B -40 °C / 125 °C, 1000 cy	Within Specifications
HAST Pressure Cooker	JESD22-A110-B 2.3 bar 125 °C 85 %RH	Reversible shift by +2 %RH
High Temperature and Humidity	JESD22-A101-B 85 °C 85 %RH 1250h	Reversible shift by +2 %RH
Salt Atmosphere	DIN-50021ss	Within Spec.
Condensing Air	-	Within Spec.
Freezing cycles fully submerged	-20 / +90 °C, 100 cy 30min dwell time	Reversible shift by +2 %RH
Various Automotive Chemicals	DIN 72300-5	Within Specifications

Table 9 Qualification tests (excerpt)

4.10 ESD (Electrostatic Discharge)

ESD immunity is qualified according to MIL STD 883E, method 3015 (Human Body Model at ±2 kV)).
Latch-up immunity is provided at a force current of ±100 mA with T_{amb} = 80 °C according to JEDEC 17. See application note "ESD, Latchup and EMC" for more information.

(1) The temperature sensor passed all tests without any detectable drift. Package and electronics also passed 100%

5 Package Information

5.1 SHT1x (surface mountable)

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial data, bidirectional
3	SCK	Serial clock, input
4	VDD	Supply 2.4 - 5.5 V
	NC	Remaining pins must be left unconnected

Table 10 SHT1x Pin Description

5.1.1 Package type

The SHT1x is supplied in a surface-mountable LCC (Leadless Chip Carrier) type package. The sensors housing consists of a Liquid Crystal Polymer (LCP) cap with epoxy glob top on a standard 0.8 mm FR4 substrate. The device is free of lead, Cd and Hg.
Device size is 7.42 x 4.88 x 2.5 mm (0.29 x 0.19 x 0.1 inch)
Weight 100 mg

The production date is printed onto the cap in white numbers in the form wwy. e.g. "351" = week 35, 2001.

5.1.2 Delivery Conditions

The SHT1x are shipped in 12mm tape at 100pcs or 400pcs.. Reels are individually labelled with barcode and human readable labels. The Lot numbers allow full traceability through production, calibration and test.

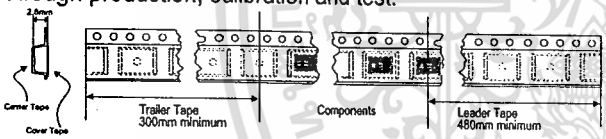


Figure 12 Tape configuration and unit orientation

5.1.3 Soldering Information

Standard reflow soldering ovens may be used. For details please see application note "soldering procedure".

For manual soldering contact time must be limited to 5 seconds at up to 350 °C.
After soldering the devices should be stored at >74 %RH for at least 24h to allow the polymer to rehydrate.
Please consult the application note "Soldering procedure" for more information.

5.1.4 Mounting Examples

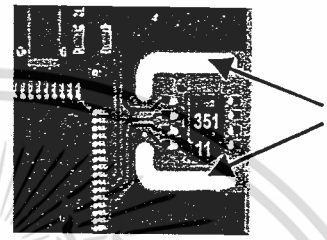


Figure 13 SHT1x PCB Mounting example

The SF1 membrane filter cap is available for optimal IP67 protection. When mounted through a housing the interior can be protected from the environment while still allowing high quality humidity measurements (see example below).

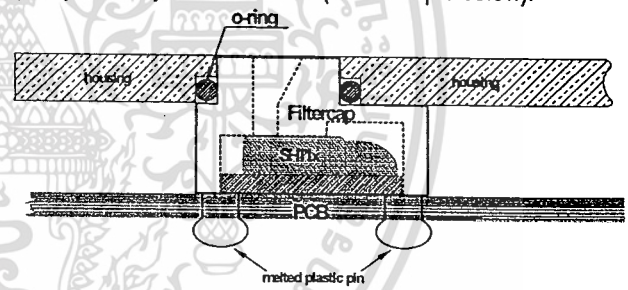


Figure 14 SF1 IP67 filter cap mounting example

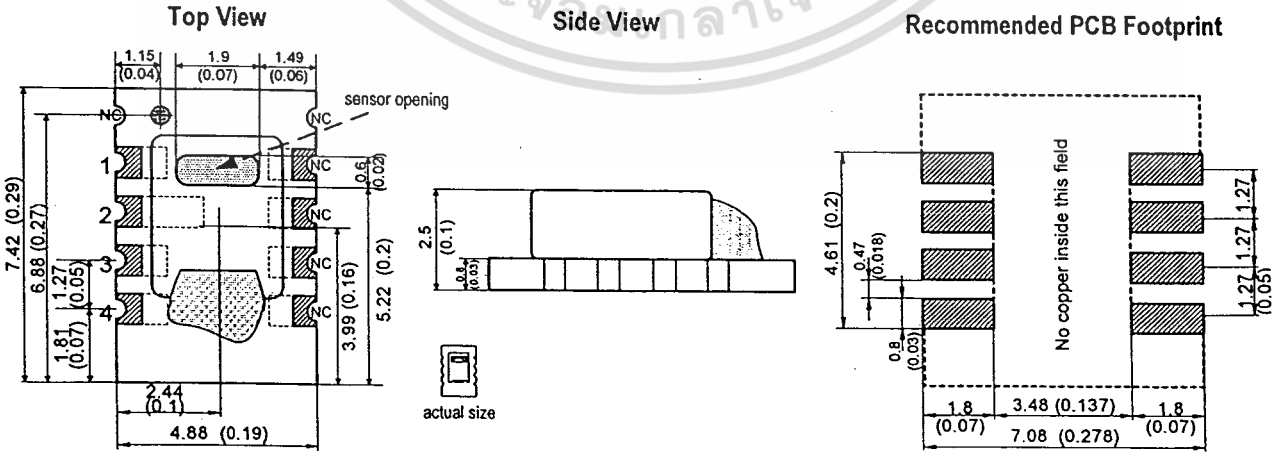


Figure 15 SHT1x drawing and footprint dimensions in mm (inch)

5.2 SHT7x (4-pin single-in-line)

Pin	Name	Comment
1	SCK	Serial clock input
2	VDD	Supply 2.4 - 5.5 V
3	GND	Ground
4	DATA	Serial data bidirectional

Table 11 SHT7x Pin Description

5.2.1 Package type[1]

The device is supplied in a single-in-line pin type package. The sensor housing consists of a Liquid Crystal Polymer (LCP) cap with epoxy glob top on a standard 0.6 mm FR4 substrate. The device is Cd and Hg free.
The sensor head is connected to the pins by a small bridge to minimize heat conduction and response times. The gold plated back side of the sensor head is connected to the GND pin.
A 100nF capacitor is mounted on the back side between VDD and GND.
All pins are gold plated to avoid corrosion. They can be soldered or mate with most 1.27 mm (0.05") sockets
e.g.: Preci-dip / Mill-Max 851-93-004-20-001 or similar
Total weight: 168 mg, weight of sensor head: 73 mg

The production date is printed onto the cap in white numbers in the form wwy. e.g. "351" = week 35, 2001.

5.2.2 Delivery Conditions

The SHT7x are shipped in 32 mm tape. These reeled parts in standard option are shipped with 500 units per 13 inch diameter reel. Reels are individually labelled with barcode and human readable labels.

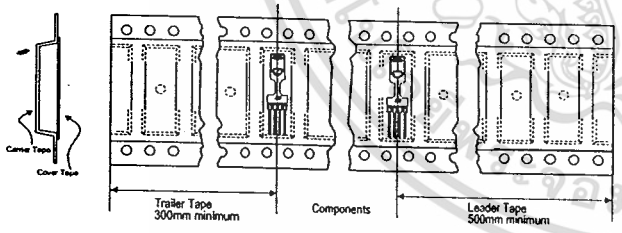


Figure 16 Tape configuration and unit orientation

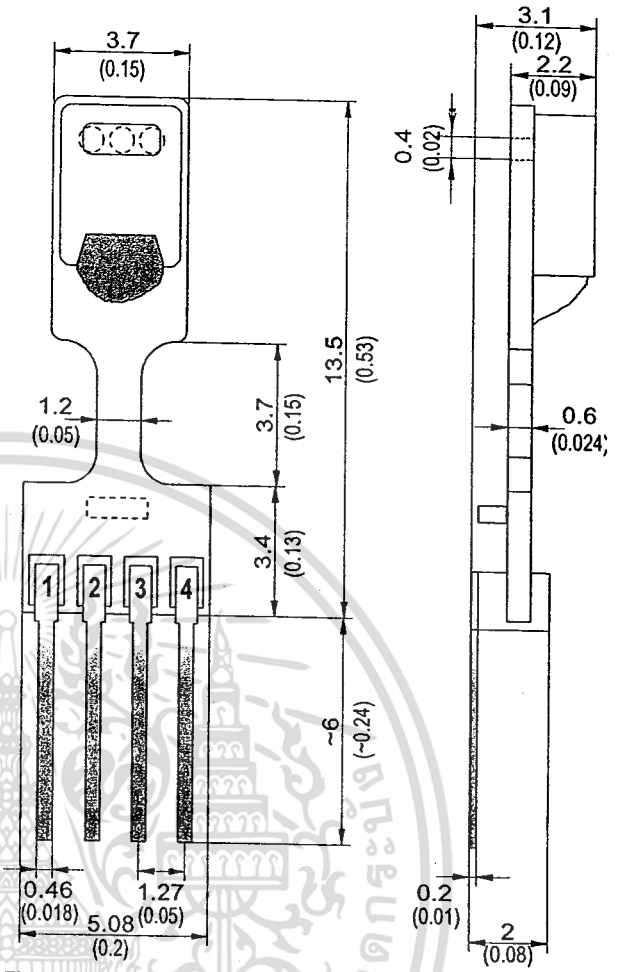


Figure 17 SHT7x dimensions in mm (inch)

5.2.3 Soldering Information[2]

Standard wave SHT7x soldering ovens may be used at maximum 235 °C for 20 seconds.
For manual soldering contact time must be limited to 5 seconds at up to 350 °C.
After wave soldering the devices should be stored at >74 %RH for at least 24 h to allow the polymer to rehydrate.
Please consult the application note "Soldering procedure" for more information.

---

[1] Other packaging options may be available on request.
[2] For maximum accuracy do not solder SHT75!

6 Revision history

Date	Version	Page(s)	Changes
February 2002	Preliminary	1-9	First public release
June 2002	Preliminary		Added SHT7x information
March 2003	Final v2.0	1-9	Major remake, added application information etc. Various small modifications
	V2.01	1-9	Typos, Graph labeling
July 2004	V2.02	1-9	Improved specifications, added SF1 information, improved wording

The latest version of this document and all application notes can be found at:
www.sensirion.com/en/download/humiditysensor/SHT1x_SHT7x.htm

7 Important Notices

7.1 Warning, personal injury

Do not use this product as safety or emergency stop devices or in any other application where failure of the product could result in personal injury. Failure to comply with these instructions could result in death or serious injury.

Should buyer purchase or use SENSIRION AG products for any such unintended or unauthorized application, Buyer shall indemnify and hold SENSIRION AG and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, costs, damages and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that SENSIRION AG was negligent regarding the design or manufacture of the part.

7.2 ESD Precautions

The inherent design of this component causes it to be sensitive to electrostatic discharge (ESD). To prevent ESD-induced damage and/or degradation, take normal ESD precautions when handling this product.

See application note "ESD, Latchup and EMC" for more information.

7.3 Warranty

SENSIRION AG makes no warranty, representation or guarantee regarding the suitability of its product for any particular purpose, nor does SENSIRION AG assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typical" must be validated for each customer applications by customer's technical experts.

SENSIRION AG reserves the right, without further notice, to change the product specifications and/or information in this document and to improve reliability, functions and design.

Headquarters and Sales Office

SENSIRION AG
Eggbühlstr. 14
P.O. Box
CH-8052 Zürich
Switzerland

Phone: + 41 (0)44 306 40 00
Fax: + 41 (0)44 306 40 30
e-mail: info@sensirion.com
http://www.sensirion.com/

Sensirion humidity sensors are available from:

find your local representative at:
www.sensirion.com/reps