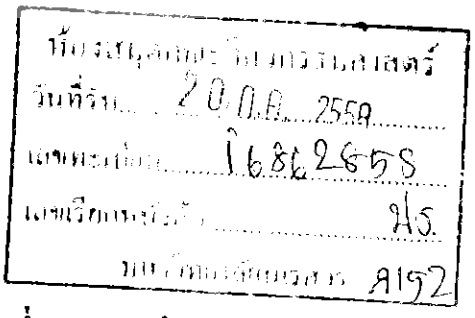
พัฒนาระบบเพื่อเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล

DATA LOGGING AND MONITORING FOR DIGITAL POWER METER

นายกรกช วิชัยทา รหัส 53362433

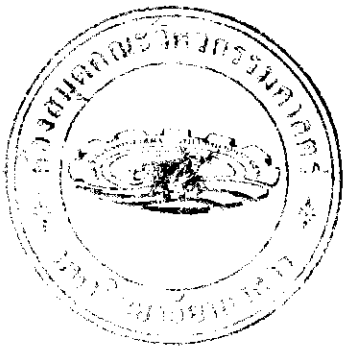


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2556



ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน พัฒนาระบบเพื่อเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล

ผู้ดำเนินโครงงาน นายกรกช วิชัยทา รหัส 53362433

ที่ปรึกษาโครงงาน ดร. ปิยดนัย ภาชนะพรรณ์

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2556

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน
(ดร. ปิยดนัย ภาชนะพรรณ์)

.....กรรมการ
(ดร. สุพรรณนิกา วัฒนะ)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. พนัส นัถฤทธิ์)

ชื่อหัวข้อโครงงาน	พัฒนาระบบเพื่อเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล
ผู้ดำเนินโครงงาน	นายกรกช วิชัยทา รหัส 53362433
ที่ปรึกษาโครงงาน	ดร. ปิยดนัย ภาชนะพรรณ์
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2556

บทคัดย่อ

มิเตอร์ไฟฟ้าแบบดิจิตอล เป็นเครื่องมือที่ได้รับการพัฒนาความสามารถให้วัดค่า และแสดงผลค่าทางไฟฟ้าออกมาได้หลายรูปแบบ โครงงานนี้เป็นการพัฒนาระบบเพื่อเก็บค่าและแสดงผลจากข้อมูลที่ มิเตอร์วัดได้ ซึ่งได้แก่ ค่า กำลังไฟฟ้าจริง, กำลังไฟฟ้าจินตภาพ, กำลังไฟฟ้าปรากฏ และ ตัวประกอบกำลังไฟฟ้า ออกมายังหน้าจอคอมพิวเตอร์และจัดเก็บในรูป text file ซึ่งพัฒนาบนพื้นฐานของโปรแกรม Virtual Basic และมีการสื่อสารระหว่างมิเตอร์และคอมพิวเตอร์ผ่านสาย UTP บนมาตรฐานการสื่อสารRS-485 บนโปรโตคอลชนิด MODBUS

Project title	Data Logging and Monitoring for Digital Power Meter
Name	Mr. Korakot Wichaita ID. 53362433
Project advisor	Dr. Piyadanai Pachanapan, Ph.D.
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic year	2013

Abstract

Digital power meter is a tool, that was developed to measure and display the electrical data. This project is the implementation of the computer software to collect the data and indicate. The measured data which is including real power, reactive power, apparent power and power factor to the monitor and logging the data into text file. This software was developed based on Virtual Basic program. Moreover, the communication system between digital power meter and computer is using the UTP cable via the communicative standard RS-485 on protocol MODBUS.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยการดูแลจาก ดร. ปิยดนัย ภาชนะพรรณ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานที่คอยให้คำปรึกษาและแนะนำ ขั้นตอนในการทำ โครงงานเกี่ยวกับการออกแบบโปรแกรมเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล นอกจากนั้นยังให้การตรวจทานเล่มปริญญานิพนธ์ ผู้ดำเนินโครงงานจึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ดร. สุพรรณนิกา วัฒนะ และ ผศ.ดร. พนัส นัถฤทธิ์ เป็นคณะกรรมการในการสอบโครงงานที่ให้คำ แนะนำ ชี้แนะแนวทาง ข้อคิดเห็นต่างๆและรวมถึงการตรวจรูปเล่มปริญญานิพนธ์ที่เป็นประโยชน์ในโครงงานนี้ ทำ ให้โครงงานนี้ออกมาสมบูรณ์ยิ่งขึ้น

ขอขอบคุณเพื่อนๆ สาขาคอมพิวเตอร์ และ สาขาไฟฟ้ากำลัง รุ่นที่ 14 ทุกคนที่คอยช่วยเหลืองานของข้าพเจ้า เพราะเป็นการกระตุ้นงานของข้าพเจ้าเองเพื่อให้ข้าพเจ้าได้ทำ โครงงานในครั้งนี้จนบรรลุวัตถุประสงค์ตามที่ข้าพเจ้าได้ตั้งเป้าหมายไว้และ ท้ายที่สุดขอกราบขอบพระคุณบิดามารดาผู้เป็นที่รักและเคารพของข้าพเจ้าผู้ที่คอยให้กำ ลังใจและให้โอกาสในการทา โครงงานมาโดยตลอดเวลาจนทำให้ประสบความสำเร็จดังทุกวันนี้

นายกรกช วิชัยทา

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบัน มีการใช้มิเตอร์ไฟฟ้าแบบดิจิตอล (Digital power meter) มากขึ้นโดยการต่อเข้ากับเครื่องคอมพิวเตอร์ตั้งโต๊ะ พกพา หรือเครื่องคอมพิวเตอร์ขนาดเล็ก มีการใช้งานในอุตสาหกรรมต่างๆอย่างแพร่หลาย แต่พบว่าซอฟต์แวร์เองนั่นมีราคาค่อนข้างสูงพอสมควร และไม่สามารถปรับแก้ไขด้วยตนเอง

ดังนั้น จึงได้มีการจัดทำโครงงานเพื่อพัฒนาซอฟต์แวร์นี้ขึ้นมาเพื่อที่จะเป็นเครื่องมือสำหรับเก็บข้อมูล และแสดงค่าทางไฟฟ้า โดยนำข้อมูลจาก มิเตอร์ผ่านทางระบบสื่อสารเชื่อมต่อเข้ากับคอมพิวเตอร์และจัดการข้อมูลเพื่อแสดงผลออกมาทางหน้าจอคอมพิวเตอร์ให้ง่ายและสะดวกต่อการจดบันทึกค่าที่ต้องการ สามารถนำไปพัฒนาสำหรับต่อเข้าเครื่องคอมพิวเตอร์ขนาดเล็กได้ซึ่งสามารถ ต่อเข้ากับระบบสื่อสารแบบใช้สาย หรือไร้สาย เพื่อที่จะส่งข้อมูลเข้าสู่ส่วนกลาง ซึ่งโครงงานนี้เป็นการเริ่มต้นบันทึกค่าต่างๆทางไฟฟ้า เพื่อที่จะเป็นต้นแบบให้ผู้สนใจ ได้ต่อเติมปรับปรุง และพัฒนาให้ดีขึ้นกว่าเดิม

1.2 วัตถุประสงค์ของโครงงาน

พัฒนาซอฟต์แวร์เพื่อเก็บค่า และ แสดงผลแบบติดตาม (Data logging and Monitoring) ที่สามารถรับค่าจาก มิเตอร์ไฟฟ้าแบบดิจิตอลระบบ 1 เฟส และแสดงผลบนหน้าจอคอมพิวเตอร์ สามารถเก็บค่าทุกๆ ช่วงเวลาระดับวินาที โดยใช้ภาษา Visual basic [1] ในการออกแบบโปรแกรมเพื่อรองรับการใช้งานกับคอมพิวเตอร์ขนาดเล็กเช่น Raspberry pi [2] ในอนาคตต่อไป ซึ่งรองรับกับภาษา Visual basic

1.3 ขอบเขตของโครงงาน

1. สร้างชุดทดสอบสำหรับการตรวจวัดค่าทางไฟฟ้าจาก มิเตอร์
2. ต่อ มิเตอร์ ผ่านระบบสื่อสารเข้ากับคอมพิวเตอร์
3. จับสัญญาณจาก มิเตอร์เพื่อนำเอาค่า กำลังไฟฟ้าจริง (P) กำลังไฟฟ้าปรากฏ (S) กำลังไฟฟ้าจินตภาพ (Q) ค่าตัวประกอบกำลังทางไฟฟ้า (Power Factor) มาเก็บและประมวลผล

4. จัดเก็บข้อมูลและแสดงผลออกทางหน้าจอ

1.4 ขั้นตอนและแผนการดำเนินงาน

1. สร้างชุดทดลองเพื่อตรวจวัดค่าทางไฟฟ้าโดยนำ (หม้อแปลงกระแส Current Transformer, CT) มาติดตั้งเพื่อวัดค่า ต่อ หม้อแปลงกระแส ตรวจวัดผ่านมิเตอร์ไฟฟ้าแบบดิจิตอล แบบ1เฟส และต่อเข้ากับคอมพิวเตอร์
2. ศึกษาการเขียนโปรแกรม เพื่อที่จะติดต่อและเก็บค่าจาก มิเตอร์
3. เขียนโปรแกรมโดยใช้โปรแกรม Visual basic เพื่อบันทึกหาค่าทางไฟฟ้าต่างๆ และแสดงผลบนหน้าจอคอมพิวเตอร์
4. ทำการทดลองจดบันทึกค่าโดยทิ้งไว้ ช่วงเวลาหนึ่งเพื่อทดสอบความถูกต้องของค่าที่ได้
5. ตรวจสอบหาข้อผิดพลาดในการทดลอง แล้วนำมาแก้ไข และทดลองซ้ำ
6. สรุปผล และจัดทำรูปเล่มโครงงาน

1.5 แผนปฏิบัติงานตลอดระยะเวลาทำโครงงาน

โครงงาน ระบบเก็บข้อมูลและแสดงผล จากอุปกรณ์วัดค่าทางไฟฟ้าแบบดิจิตอล

การปฏิบัติงาน	เดือน								
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. ศึกษาค้นคว้าข้อมูล	←	—	—	→					
2. ออกแบบชุดทดสอบ				↔					
3. ต่อชุดทดสอบ					↔				
4. พัฒนา ซอฟต์แวร์เก็บข้อมูลและแสดงผลจากมิเตอร์						↔	↔		
5. ทำการทดลอง สร้างต้นแบบโปรแกรมเพื่อจัดเก็บข้อมูลและแสดงผลที่ได้จากมิเตอร์								↔	
6. สรุปผล และจัดทำรูปเล่ม								←	→

1.6 งบประมาณที่ใช้

รายละเอียดงบประมาณที่ของโครงงานมีดังนี้

1) ค่าพิมพ์เอกสาร	200 บาท
2) จัดทำเล่มปริญญานิพนธ์	800 บาท
รวมเป็นเงินทั้งสิ้น(หนึ่งพันบาทถ้วน)	1,000 บาท

หมายเหตุ: ค่าใช้จ่ายถัวเฉลี่ยทุกรายการ

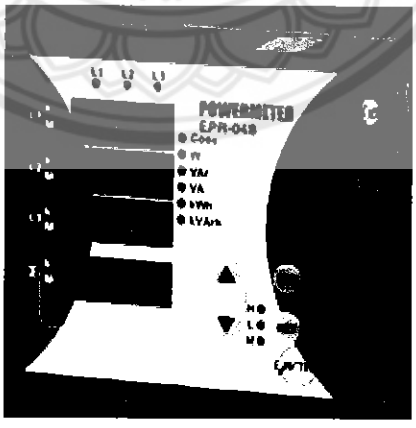
บทที่ 2

ทฤษฎีหลักการและเอกสารที่เกี่ยวข้อง

ในการจัดทำโครงงาน พัฒนาระบบเพื่อเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอลประกอบด้วยส่วนต่างๆดังต่อไปนี้ คือ มิเตอร์ไฟฟ้าแบบดิจิตอล EPR-04s จะต่อเข้ากับระบบสื่อสารอนุกรม RS485 ผ่านโปรโตคอล MODBUS ลักษณะข้อมูลที่ส่งผ่านออกมาจากมิเตอร์ จะมาในรูปแบบของเลขฐาน 16 (Hexadecimal format) และจะแปลงจาก RS485 มาเป็น RS232 ซึ่งมีตัวเชื่อมต่อแบบ DB9 เพื่อที่จะสามารถต่อเข้ากับคอมพิวเตอร์ เพื่อแสดงผลรายละเอียดของอุปกรณ์ในเบื้องต้นจะถูกอธิบายในบทนี้

2.1 มิเตอร์ไฟฟ้าแบบดิจิตอล ERP-04S

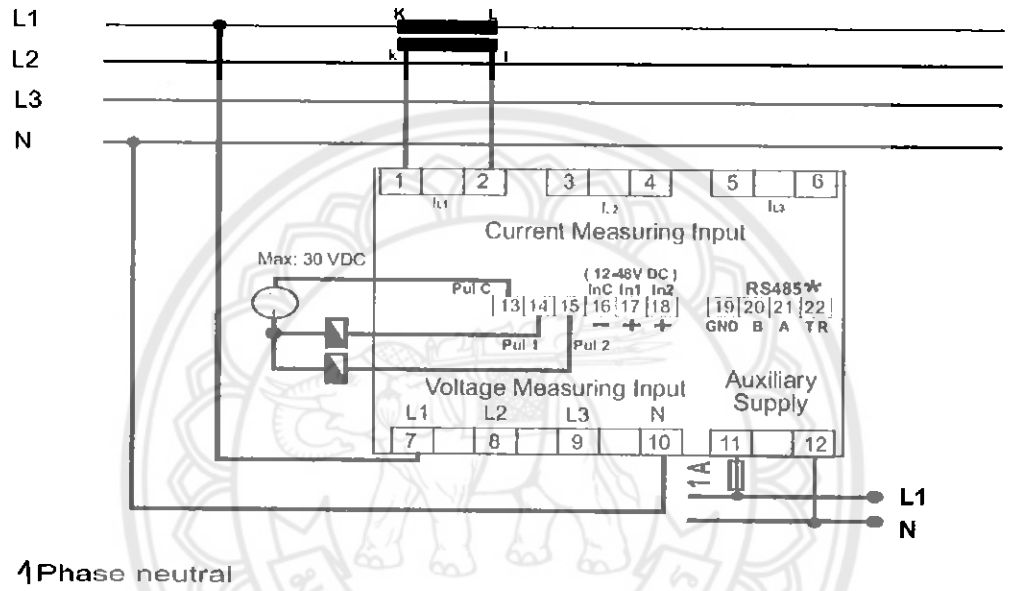
มิเตอร์ไฟฟ้าแบบดิจิตอล ERP-04S [3] เป็นอุปกรณ์วัดค่าพลังงานทางไฟฟ้าแบบดิจิตอลที่สามารถแสดงการวัดต่อเฟสและรวมผลทั้งสามเฟสได้ในหน้าจอเดียวกัน สามารถวัดค่าพลังงานทางไฟฟ้าได้ เช่น ค่าตัวประกอบกำลังทางไฟฟ้า (*p.f*) ค่ากำลังไฟฟ้าปรากฏ (*S*) ค่ากำลังไฟฟ้าจริง (*P*) ค่ากำลังไฟฟ้าจิตภาพ (*Q*) มีคุณภาพในการวัดค่าพลังงานไฟฟ้า จึงสามารถทดแทนการใช้มิเตอร์แบบจานหมุนได้อย่างมีประสิทธิภาพ และค่าพลังงานไฟฟ้า ที่ถูกบันทึกไว้ใน มิเตอร์ไฟฟ้าแบบดิจิตอล จะยังคงอยู่แม้ไฟดับ



รูปที่ 2.1 มิเตอร์ไฟฟ้าแบบดิจิตอล รุ่น EPR-04S [3]

2.1.1 การเชื่อมต่อระบบมิเตอร์เข้ากับระบบไฟฟ้า

การเชื่อมต่อมิเตอร์ไฟฟ้าแบบดิจิตอลเข้าในระบบเพื่อการวัดค่าจะต่อมิเตอร์เข้ากับระบบสื่อสารอนุกรม RS485 ที่เป็นโปรโตคอลแบบ MODBUS ที่จะส่งข้อมูลผ่านสายส่งแบบอนุกรม 2 เส้นไปยังตัวแปลงสัญญาณสื่อสารอนุกรม RS232 ที่มีตัวเชื่อมแบบ DB9 ตัวผู้ ที่จะต่อเข้ากับคอมพิวเตอร์ผ่านตัวเชื่อมต่อแบบ DB9 ตัวเมีย การเชื่อมต่อมิเตอร์เข้ากับระบบการวัดค่าสามารถต่อได้ทั้งระบบ 1 เฟส และ 3 เฟส แต่ในการเก็บข้อมูลเพื่อจัดทำโครงงานนี้จะเป็นการต่อวัดผลแบบ 1 เฟส



รูปที่ 2.2 แสดงการต่อมิเตอร์แบบ 1 เฟส [3]

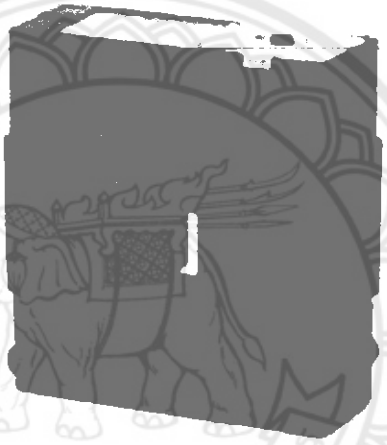
จากภาพจะเป็นส่วนด้านหลังของมิเตอร์ L1 หมายถึงสายไฟฟ้า เฟสที่ 1 จะต่อเข้ากับ Input ในภาพคือหมายเลข 7 และ N คือสายนิวทรัลของระบบ ในภาพคือหมายเลข 10 เส้นประ คือการต่อสายดิน อุปกรณ์ K-L คือหม้อแปลงกระแส (CT) จะต่ออนุกรมกับสายไฟฟ้าเฟสที่ 1 เชื่อมต่อกับหมายเลข 1 และ 2 ตามภาพที่แสดงการต่อหม้อแปลงกระแสไม่จำเป็นต้องใช้ไฟเลี้ยง สามารถนำสายไฟสอดผ่านตัวหม้อแปลงได้เลย และต่อจากขั้วของหม้อแปลงกระแสเข้าสู่มิเตอร์ได้ดังภาพที่แสดง

2.1.2 หม้อแปลงกระแส

หม้อแปลงกระแส (CT) [4] คือเครื่องมือวัดชนิดหนึ่ง ที่อาศัยหลักการเหนี่ยวนำของสนามแม่เหล็กไฟฟ้า ที่เกิดจากการไหลของกระแสไฟฟ้า ผ่านลวดตัวนำ เช่น สายไฟฟ้า การไหลของกระแสไฟฟ้าจะทำให้เกิดฟลักซ์แม่เหล็กขึ้นมารอบๆตัวนำ เมื่อต่อหม้อแปลงกระแสเข้ากับตัวนำไฟฟ้า ฟลักซ์แม่เหล็กจะตัดผ่านแกนเหล็กในด้านแรงสูงของหม้อแปลงกระแส จะทำให้เกิดการเหนี่ยวนำ ออกมาเป็นค่ากระแสและแรงดันที่มีค่าต่ำลง ทางด้านแรงต่ำของหม้อแปลงกระแส

ซึ่งสามารถนำไปป้อนให้กับเครื่องมือวัดต่าง ๆ เช่นมิเตอร์ไฟฟ้าได้ เนื่องจากเครื่องมือวัดส่วนใหญ่ไม่สามารถใช้กับค่าพิกัดกระแสสูงๆได้

หม้อแปลงกระแสที่ใช้ในการทดลองนี้ มีอัตราทดของค่ากระแสด้านแรงสูงและแรงต่ำเป็น 100/5 หมายความว่า ถ้ากระแสในลวดตัวนำมีค่า 100 A. จะเกิดการเหนี่ยวนำออกมาเป็นกระแสในด้านแรงต่ำของหม้อแปลงกระแสเท่ากับ 5 A. แต่เนื่องจากค่ากระแสจากการต่อภาระในการทดลองมีค่าน้อยหากเทียบกับขนาดหม้อแปลงที่มี จึงต้องพันสายไฟด้านแรงสูงของหม้อแปลงกระแสเพิ่มขึ้นเป็น 4 รอบ ทำให้อัตราทดกระแสมีค่าเพิ่มขึ้นจากเดิม 100/5 เป็น 25/5 A (อัตราทดของ CT จึงมีค่าเท่ากับ 5)



รูปที่ 2.3 หม้อแปลงกระแสที่ใช้ในการทดลอง [4]

2.2 มาตรฐาน ระบบสื่อสารที่เกี่ยวข้อง

มาตรฐาน RS232 และมาตรฐาน RS485 เป็นมาตรฐานสื่อสารอนุกรม ที่กำหนดขึ้นมาโดยกลุ่มพันธมิตร อุตสาหกรรมอิเล็กทรอนิกส์ในประเทศสหรัฐอเมริกา EIA (Electronic Industries Alliance) ในปี พ.ศ.2512 RS232 จะเป็นเป็นมาตรฐานที่นิยมใช้มากกว่า RS485 เริ่มต้นจากความต้องการที่จะกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมพิวเตอร์กับโมเด็มในสมัยนั้น RS232 จะมีระยะทางการเชื่อมต่อน้อยกว่า RS485 ระยะทางการส่งข้อมูลสูงสุดของ RS232 จะอยู่ที่ 900 เมตร และที่ระยะนี้ ความเร็วในการรับส่งจะอยู่ที่ 2.4 kbps ถ้าหากต้องการความเร็วที่มากกว่านั้นโดยความเร็วสูงสุดที่ RS232 สามารถทำได้ คือ 19.2 kbps แต่จะต้องต่อสายสัญญาณให้สั้นลงเหลือเพียง 15 เมตร RS232 สามารถทำงานได้ทั้งแบบ Half duplex และ Full duplex การส่งข้อมูลบนมาตรฐาน

RS232 จะใช้สายที่เป็นรูปแบบเฉพาะของ RS232 คือสายที่มีหัว DB9 ติดตั้งอยู่ และการเชื่อมต่อ จะเป็นการเชื่อมแบบ จุดต่อจุด ไม่สามารถต่อหลายจุดบนสายส่งเส้นเดียวกันได้

การใช้มาตรฐาน RS232 เชื่อมต่อระบบสื่อสารในโรงงานที่มีการสวิตชิ่งค่ากระแสสูงๆ จะทำให้เกิดสัญญาณรบกวน เนื่องจาก RS232 จะเทียบสัญญาณจากค่ากราวด์ เมื่อเกิดการสวิตชิ่ง จะทำให้ค่ากราวด์ถูกรบกวน ทำให้ค่ากราวด์เปลี่ยนไป การเทียบสัญญาณอาจจะมีการผิดพลาดได้ ภายหลังจึงนิยมใช้มาตรฐาน RS485 มากกว่า

มาตรฐาน RS485 เป็นหนึ่งในมาตรฐานการสื่อสารแบบอนุกรม (Serial Communication) เป็นระบบบัสที่พัฒนาต่อมาจาก RS422 และ RS-232 เพื่อตอบสนองต่อความต้องการใช้งาน ที่ต้องการเชื่อมต่ออุปกรณ์หลายๆตัวบนเครือข่ายเดียวกันเข้าด้วยกัน โดยมีระยะทางการสื่อสารที่ไกลขึ้น และมีความเร็วรับส่งข้อมูลที่สูงขึ้น เมื่อเทียบกับมาตรฐานการสื่อสาร RS-232 และ RS422

RS485 เป็นการสื่อสารอนุกรมของมิเตอร์ไฟฟ้าแบบดิจิตอลรุ่น EPR-04S ภายในจะมีการติดต่อสื่อสารโดยใช้โปรโตคอล MODBUS ที่จะส่งค่าพารามิเตอร์ต่างๆไปที่คอมพิวเตอร์ แต่การที่จะเชื่อมต่อเข้ากับคอมพิวเตอร์ได้นั้น ต้องมีการแปลงจาก RS485 ไปเป็น RS 232 เพื่อที่จะเข้ากับพอร์ตขนานของคอมพิวเตอร์ได้ การส่งข้อมูลจากมิเตอร์ไปยังคอมพิวเตอร์ จะเป็นการส่งข้อมูลแบบอนุกรม โดยเป็นการส่งแบบไม่ประสานเวลา (Asynchronous) ลักษณะการส่งจะมีสายสัญญาณ 2 เส้น คือ RXD TXD และ RS485 จะไม่เทียบสัญญาณจากค่ากราวด์เหมือน RS232 แต่จะเทียบสัญญาณระหว่างสาย 2 เส้นที่ใช้ส่ง ทำให้ RS-485 ทำงานได้แบบ Half duplex นั่นคือการสื่อสารข้อมูล 2 ทิศทางจะต้องแยกการรับหรือการส่งข้อมูลออกจากกัน ไม่สามารถรับส่งข้อมูลได้เวลาเดียวกันยกตัวอย่างเช่น วิทยุสื่อสาร ส่วนแบบ Full duplex จะเป็นการรับ-ส่งข้อมูลได้ในเวลาเดียวกัน ยกตัวอย่างเช่นโทรศัพท์

การส่งข้อมูลบนมาตรฐาน RS485 จะใช้สายสัญญาณแบบสายคู่พันเกลียว (Unshielded Twisted Pair (UTP)) ระยะทางในการส่งข้อมูลสูงสุด จะอยู่ที่ 1200 เมตร ที่ระยะนี้ความเร็วในการส่งข้อมูลจะอยู่ที่ 100 kbs โดยประมาณ หากต้องการความเร็วที่มากกว่านั้น จะต้องใช้สายสัญญาณที่สั้นลง ความเร็วสูงสุด ที่ RS485 สามารถทำได้ จะอยู่ที่ 35 Mbps แต่สายที่ใช้ส่งข้อมูลที่ความเร็วสูงสุดของ RS485 ควรน้อยกว่า 12 เมตร

มาตรฐาน RS485 มีประโยชน์เป็นอย่างมากสำหรับระบบงานที่มีเครื่องวัดหลายตัวเชื่อมต่อบนสายสัญญาณที่เป็นบัสเดียวกัน แต่อย่างไรก็ตามจะต้องมีความระมัดระวังอย่างเป็นพิเศษในการตั้งค่าซอฟต์แวร์เพื่อป้องกันไม่ให้หลายอุปกรณ์ส่งข้อมูลในเวลาเดียวกันวิธีการที่ใช้กันส่วนใหญ่จะกำหนดให้อุปกรณ์หรือจุดเชื่อมต่อตัวอุปกรณ์ตัวหนึ่งทำตัวเป็นตัวแม่ (Master) [5]

Characteristics of RS232, RS422, RS423 and RS485

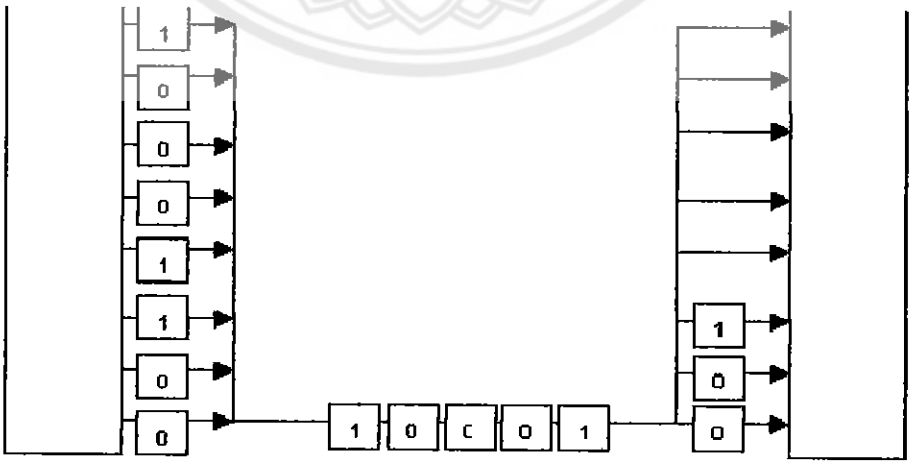
	RS232	RS422	RS485
Differential	no	yes	yes
Max number of drivers	1	1	32
Max number of receivers	1	10	32
Modes of operation	half duplex full duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m
Max speed at 12 m Max speed at 1200 m	20 kbs (1 kbs)	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	n/a	n/a
Receiver input resistance	3..7 kΩ	≧ 4 kΩ	≧ 12 kΩ
Driver load impedance	3..7 kΩ	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV
Receiver input range	±15 V	±10 V	–7..12 V
Max driver output voltage	±25 V	±6 V	–7..12 V
Min driver output voltage (with load)	±5 V	±2.0 V	±1.5 V

รูปที่ 2.4 เปรียบเทียบมาตรฐาน RS232 RS422 และ RS485 [5]

2.3 ทฤษฎีระบบสื่อสารที่เกี่ยวข้อง

2.3.1 การส่งข้อมูลแบบอนุกรม (Serial Transmission)

การส่งข้อมูลแบบอนุกรมนั้น จะเป็นการส่งข้อมูลทีละบิตผ่านสายส่งข้อมูล โดยการเรียงลำดับการส่งจะขึ้นอยู่กับชนิดของอุปกรณ์ โดยที่ตัวส่งจะมีข้อมูลหลายๆชุดหรือหลายๆบิตรวมกัน แต่การส่งนั้นจะส่งได้เพียงทีละ 1บิต ส่งแบบเรียงกันมา ไม่สามารถส่งหลายๆบิตพร้อมกันได้ ดังภาพที่แสดง



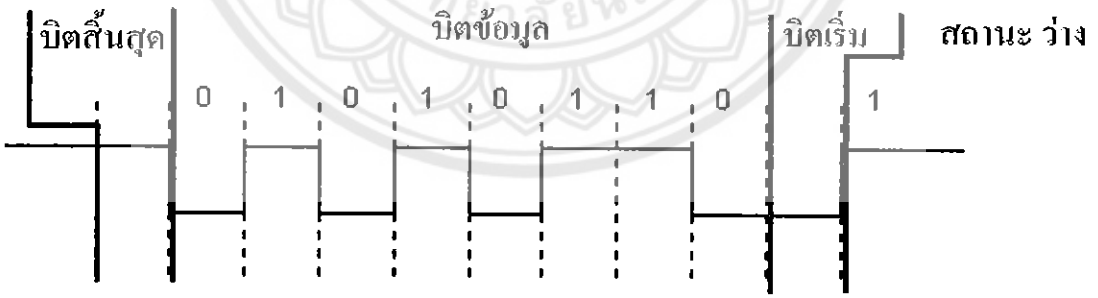
รูปที่ 2.5 การส่งข้อมูลของระบบสื่อสารอนุกรม [5]

จากรูปที่ 2.5 ด้านซ้ายคือต้นทาง และ ด้านขวาคือปลายทาง โดยชุดคำสั่งข้อมูลจะถูกส่งออกจากคอมพิวเตอร์ไปยังมิเตอร์ และเมื่อมิเตอร์รับชุดคำสั่งเข้ามา ก็จะตอบสนองเป็นข้อมูลตามชุดคำสั่งออกไปยังคอมพิวเตอร์ และข้อมูลทั้งหมดนั้นจะถูกส่งผ่านระบบสื่อสารอนุกรม ในชุดข้อมูล อักษรจะประกอบด้วยบิตข้อมูลจำนวน 8 บิต เรียงเป็นลำดับ ข้อมูลจะถูกส่งออกมาทีละบิตระหว่างต้นทางถึงปลายทาง และปลายทางจะรวบรวมบิตเหล่านี้ทีละบิตจนครบ 8 บิต เป็น 1 ตัวอักษร การส่งข้อมูลแบบนี้จะเหมาะสำหรับการส่งระยะทางไกลๆ

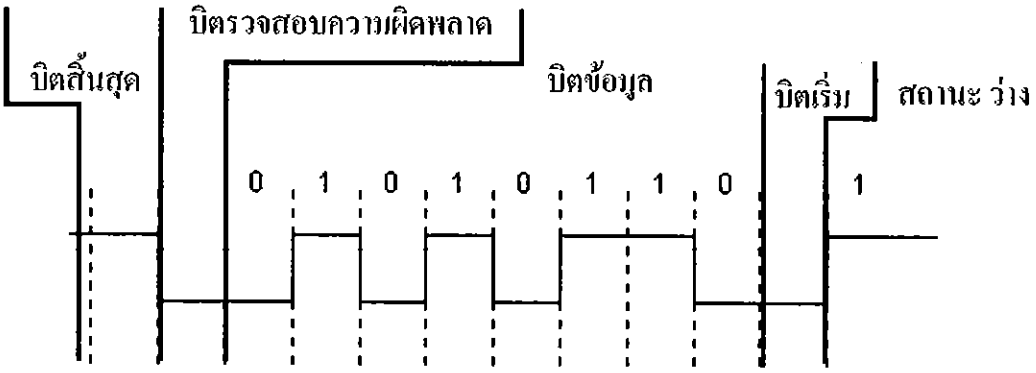
โดยทั่วไปแล้วการส่งข้อมูลนั้นจะประกอบไปด้วยกลุ่มของตัวอักษร ดังนั้นในการส่งข้อมูลแบบอนุกรมนี้จึงเกิดปัญหาขึ้นว่า แล้วต้นทางและปลายทางจะทราบได้อย่างไรว่า จะแบ่งแต่ละตัวอักษรตรงบิตใด จึงเกิดวิธีการสื่อสารข้อมูลขึ้น 2 แบบคือ การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission) และการสื่อสารแบบซิงโครนัส (Synchronous Transmission)

2.3.2 การสื่อสารแบบอะซิงโครนัส และ การสื่อสารแบบซิงโครนัส

การสื่อสารแบบอะซิงโครนัส หรือเรียกอีกอย่างหนึ่งว่าเป็น การสื่อสารแบบระบุจุดเริ่มต้นและจุดสิ้นสุด (Start-Stop Transmission) ลักษณะของสัญญาณที่ใช้ในการติดต่อสื่อสารกันจะประกอบไปด้วย บิตเริ่มต้น (start bit) บิตของข้อมูลที่สื่อสาร (transmission data) จำนวน 8 บิต บิตตรวจข้อผิดพลาด (parity bit) และ บิตสิ้นสุด (stop bit) สำหรับบิตตรวจสอบข้อผิดพลาดจะใช้หรือไม่ใช้ก็ได้ ดังนั้นสัญญาณจึงต้องประกอบด้วยส่วนประกอบอย่างน้อย 3 ส่วนหลักจากรูปที่ 2.6 คือ บิตเริ่ม บิตข้อมูล และ บิตสิ้นสุ



รูปที่ 2.6 การสื่อสารแบบอะซิงโครนัสที่ไม่ได้ใช้บิตตรวจสอบความผิดพลาด [5]



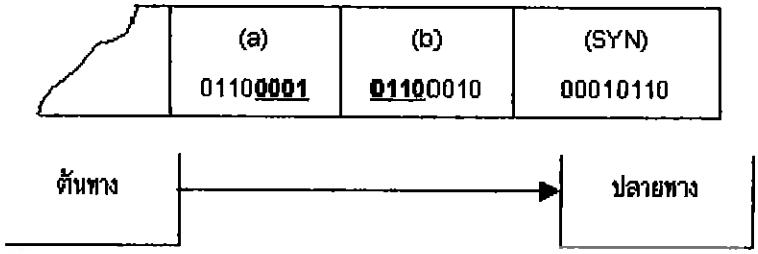
รูปที่ 2.7 การสื่อสารแบบอะซิงโครนัสที่ใช้บิตตรวจสอบความผิดพลาด [5]

จากรูปที่ 2.6 และ 2.7 จะเห็นว่าขณะที่ไม่มีข้อมูลส่งออกมาสถานะของการส่งจะเป็นแบบว่าง (Idle) ซึ่งจะมีระดับของสัญญาณเป็น 1 ตลอดเวลา เพื่อความแน่ใจว่าปลายทาง หรือฝ่ายรับยังคงติดต่อกับต้นทาง หรือฝ่ายส่งอยู่ เมื่อเริ่มจะส่ง ข้อมูลสัญญาณของอะซิงโครนัสจะเป็น 0 ในหนึ่งช่วงสัญญาณนาฬิกา ซึ่งบิตนี้จะเรียกว่าบิตเริ่มต้น ตามหลังของบิตเริ่มต้นจะเป็นบิตข้อมูลสำหรับ 1 ตัวอักษร ตามหลังบิตข้อมูลก็จะเป็นบิตตรวจความถูกต้อง แล้วจะตามด้วยบิตสิ้นสุด ถ้าไม่ใช่บิตตรวจความถูกต้อง ตามหลังบิตข้อมูลก็จะเป็นบิตสิ้นสุดเลย หลังจากนั้นถ้าไม่มีข้อมูลส่งออกมาสัญญาณจะกลับไปอยู่ที่สถานะแบบว่างอีก เพื่อรอการส่งข้อมูลต่อไป จะเห็นว่าการสื่อสารแบบอะซิงโครนัสนี้ มีลักษณะเป็นไปทีละตัวอักษร และสัญญาณที่ส่งออกมา มีบางส่วนเป็นบิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด ทำให้ความเร็วในการส่งข้อมูลต่อวินาทีน้อยลงไป เนื่องจากต้อง สูญเสียช่องทางการสื่อสารให้กับ บิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด (ถ้ามีใช้) ตลอดเวลา

การสื่อสารแบบซิงโครนัส จะทำการจัดกลุ่มของข้อมูลเป็นกลุ่มๆ และทำการส่งข้อมูลทั้งกลุ่มไปพร้อมกันในทีเดียว ซึ่งจะเรียกกลุ่มของข้อมูลนี้ว่า บล็อกของข้อมูล (Block of Data) ซึ่งตัวอักษรตัวแรก และตัวถัดไปที่อยู่ในบล็อกเดียวกันจะไม่มีอะไรมาคั่นเหมือนอย่างแบบอะซิงโครนัส ที่ต้องใช้บิตเริ่มต้น และบิตสิ้นสุดคั่นทุกๆ ตัวอักษร แต่จะมีข้อมูลเริ่มต้นซึ่งเป็นลักษณะของบิตพิเศษที่ส่งมาเพื่อให้รู้ว่านั้นคือ จุดเริ่มต้นของกลุ่มตัวอักษรที่กำลังส่งเรียงกันเข้ามา เช่น อักขระซิง (SYN character) โดยที่อักขระซิงมีรูปแบบบิต คือ 00010110

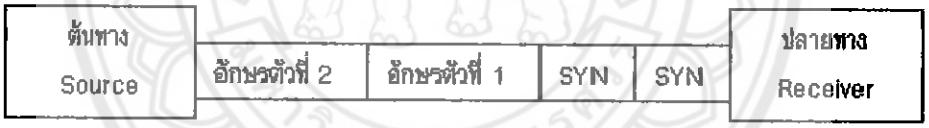
รูปที่ 2.8 ตัวอย่างของการส่งอักษรซิง [5]

จากรูปที่ 2.8 เมื่อปลายทางตรวจพบอักขระซิง หรือ 00010110 แล้วจะทราบได้ทันทีว่าบิตที่ตามมาคือบิตตัวอักษรแต่ละตัว แต่การใช้อักขระซิงเพียงตัวเดียวอาจเกิดข้อผิดพลาดได้ เช่น ถ้าส่งตัวอักษร b และตัวอักษร a ติดต่อกันไป ซึ่งตัวอักษร b มีรูปแบบบิตคือ 01100010 และตัวอักษร a มีรูปแบบบิตคือ 01100001 การส่งจะแสดงได้ดังรูปที่ 2.9

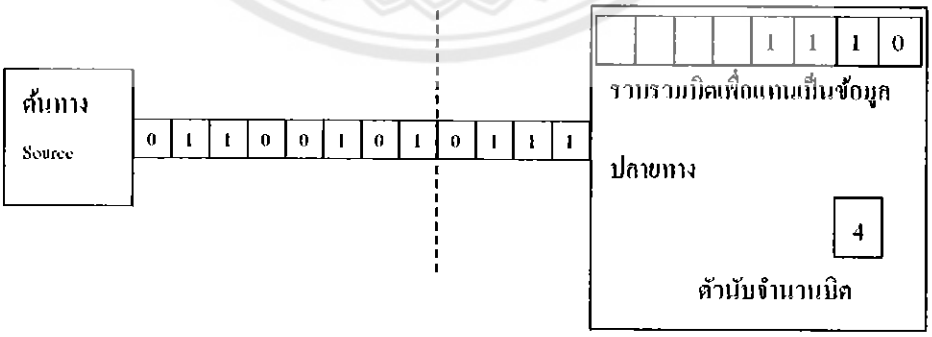


รูปที่ 2.9 ตัวอย่างการส่งอักษรซิงตามด้วยอักษร a และ b [5]

จากรูปจะเห็นว่าเครื่องปลายทางจะตรวจพบอักขระซิงระหว่างบิตของตัวอักษร b และตัวอักษร a ทำให้เข้าใจว่าบิตต่อไปจะเป็นบิตของกลุ่มข้อมูล ซึ่งจะทำให้การรับข้อมูลนั้นเกิดผิดพลาดขึ้นได้ ดังนั้นจึงแก้ปัญหาด้วยการใช้อักขระซิง 2 ตัวต่อกันเป็นลักษณะของบิตพิเศษที่บอกให้ทราบว่าเป็นจุดเริ่มต้นบิตของกลุ่มข้อมูล ตัวอย่างของการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส และการตัดแถวของบิตข้อมูลออกเป็นกลุ่มทีละ 8 บิต เพื่อแทนข้อมูลแสดงได้ดังรูป 2.10

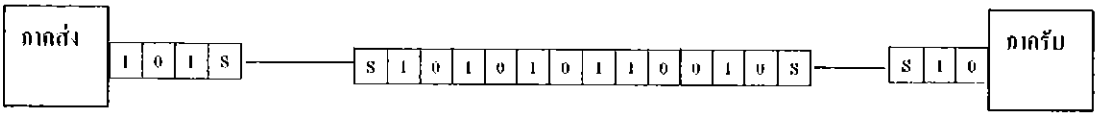


รูปที่ 2.10 ตัวอย่างการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส [5]



รูปที่ 2.11 การตัดแถวของบิตออกเป็นกลุ่ม กลุ่มละ 8 บิต [5]

รูปที่ 2.12 การส่งผ่านข้อมูลแบบซิงโครนัส [5]



รูปที่ 2.13 การส่งผ่านข้อมูลแบบอะซิงโครนัส [5]

จากรูปที่ 2.12 แสดงให้เห็นว่าการส่งผ่านข้อมูลแบบซิงโครนัสนั้นส่วนมากแล้ว ตลอดทางของสายส่งจะใช้ส่งผ่านข้อมูลเต็มตลอดทั้งสาย ส่วนรูปที่ 2.13 แสดงให้เห็นว่าการส่งผ่านข้อมูลแบบอะซิงโครนัสนั้นสายส่งจะขาดความต่อเนื่องของสัญญาณข้อมูลที่ส่งผ่าน หรือถ้ามีสัญญาณข้อมูลที่ส่งผ่านต่อเนื่องกันเต็มตลอดทั้งสายแล้ว ก็จะสูญเสียช่องทางในการส่งไปกับการส่งบิตเริ่มต้น และบิตสิ้นสุดของแต่ละตัวอักษร

ตัวอย่างเช่น กรณีที่ส่งผ่านข้อมูลที่อยู่ในรูปของรหัส ASCII ซึ่งตัวอักษรหนึ่งตัวถูกแทนด้วย 8 บิต ถ้ามีการส่งกลุ่มของข้อมูล 240 ตัวอักษร ในกรณีการส่งผ่านข้อมูลแบบซิงโครนัสมีการใช้ตัวอักขระซิง 3 ตัว และการส่งผ่าน ข้อมูลแบบอะซิงโครนัส ไม่มีการใช้บิตตรวจข้อผิดพลาด ดังนั้นจะสามารถคำนวณหาอัตราส่วนระหว่างการส่งข้อมูลได้ ดังนี้

บิตทั้งหมดของตัวอักษรที่ส่งจะได้ 240 ตัวอักษร x 8 บิต = 1920 บิต แบบซิงโครนัส บิตของตัวอักขระซิงที่ใช้จะได้ SYN 3 ตัว เท่ากับ 3 x 8 บิต = 24 บิตผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 24 = 1944 บิต อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ 1920 หารด้วย 1944 จะได้ประมาณ 99 % ส่วน แบบอะซิงโครนัส บิตเริ่มต้น และบิตสิ้นสุดที่ใช้จะได้ 2 x 240 = 480 บิต ผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 480 = 2400 บิต อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ 1920 หารด้วย 2400 จะได้ประมาณ 80 %

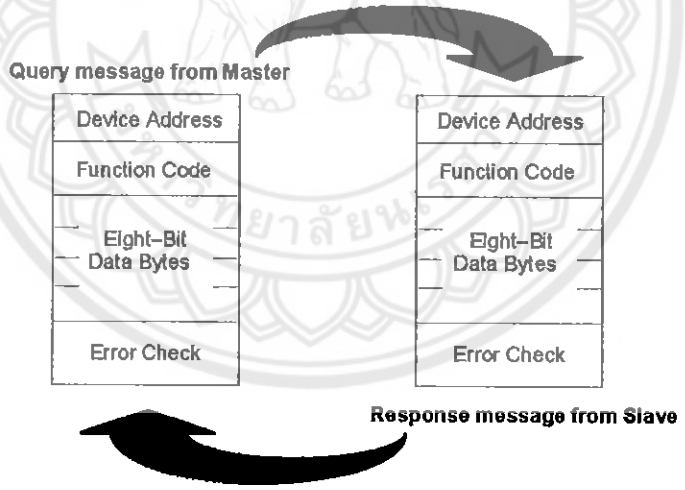
2.3.3 โปรโตคอล MODBUS

โปรโตคอล MODBUS [6] เป็นโปรโตคอลเพื่อสื่อสารข้อมูลอินพุต/เอาต์พุตและริจิสเตอร์ภายใน PLC (Programmable Logic Controller) [7] ซึ่งถูกคิดค้นมาตั้งแต่ปี 1979 โดย Modiconซึ่งเป็นหนึ่งในบริษัทผู้ผลิต PLCในประเทศสหรัฐอเมริกา (ปัจจุบันเป็นบริษัท Schneider Electric)

[8]โปรโตคอล MODBUS เป็นโปรตอลที่เป็นที่ยอมรับอย่างกว้างขวาง เนื่องจาก MODBUS เป็นระบบเปิด เชื่อมต่อและพัฒนาง่ายและไม่มีค่าใช้จ่าย (royalty-Free) อีกทั้งยังแพร่หลายในการนำโปรโตคอลนี้ไปใช้งานในอุปกรณ์อื่นๆ เช่น RTU (Remote terminal Unit), Remote I/O, Digital Power Meter เป็นต้นและในทุกซอฟต์แวร์ SCADA [9] มีความสามารถที่จะสื่อสารกับ MODBUS MODBUS จึงเป็นโปรโตคอลหลักในงานอุตสาหกรรมเพื่อใช้งานในระบบ SCADA และ PLC

MODBUS เป็นโปรโตคอล การสื่อสารรูปแบบง่าย ๆ เป็นรูปแบบการส่งข้อมูลระหว่างอุปกรณ์อิเล็กทรอนิกส์โดยอุปกรณ์ที่ต้องการข้อมูลเรียกว่า ตัวแม่ ส่วนอุปกรณ์ที่ให้ข้อมูลที่ต้องการเรียกว่าตัวลูก โดยใช้รับส่งข้อมูลจากอุปกรณ์ควบคุมกับคอมพิวเตอร์ขนาดเล็ก หรือระบบประมวลผลข้อมูลต่างๆ

การทำงานของ MODBUS เป็นการสื่อสารโดยการส่งข้อมูลไปตามสายสัญญาณ ระหว่างอุปกรณ์โดยวิธีการสื่อสารจะต้องต่อสายสัญญาณอนุกรม ระหว่าง ตัวแม่ (Master) หนึ่งตัวกับ ตัวลูก (Slave) อีกหนึ่งตัว ซึ่งข้อมูลจะถูกส่งต่อเนื่องไปด้วยสัญญาณ 0 หรือ 1 ซึ่งเรียกว่า บิต โดยแต่ละบิตจะอยู่ในรูปแบบแรงดัน โดยเลข 0 จะแทนด้วยแรงดันค่าบวกและ 1 แทนด้วยแรงดันค่าลบ สัญญาณ จะถูกส่งไปด้วยความเร็วที่กำหนดไว้ หรือตามความเร็วทั่วไปที่ 9600bps



รูปที่ 2.14 การติดต่อสื่อสารแบบ Master/Slave [5]

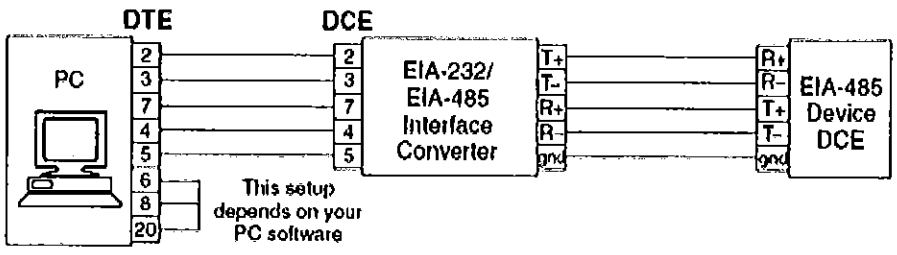
ภายในเฟรมจะส่งบิตข้อมูลรวม 11 บิต คือ บิตเริ่มต้น 1 บิต, ตามด้วยบิตข้อมูล8 บิต, ต่อด้วยบิตตรวจสอบความถูกต้องของข้อมูล 1 บิต และบิตสุดท้ายคือบิตหยุด 1 บิต หากเลือกแบบไม่ให้มีการตรวจสอบความถูกต้องของข้อมูลบิตหยุดจากเดิมมี 1บิต จะเป็นแทน 2 บิตแทน สำหรับการกำหนดให้มีการตรวจสอบความถูกต้องของข้อมูลนั้น สามารถกำหนดบิตตรวจสอบนั้นเป็นแบบคู่หรือคี่ก็ได้หากต้องการ ออกระบบสื่อสารโดยทั่วไปให้สอดคล้องกับอุปกรณ์ที่มีการใช้งาน

อย่างแพร่หลายควรเลือกการตรวจสอบแบบคู่ เพราะแบบคู่นี้ที่สามารถปรับเปลี่ยนเป็นแบบคี่หรือไม่มีการตรวจสอบความถูกต้องของข้อมูลได้อีกด้วย

2.3.4 ระบบสื่อสารอนุกรม RS485 และ RsS232 แบบ Master/lave เพื่อเชื่อมมิเตอร์เข้ากับคอมพิวเตอร์

เป็นการสื่อสารอนุกรมสำหรับคอมพิวเตอร์และอุปกรณ์ต่างๆ โดยการรับ-ส่ง จะเป็นแบบ Half-Duplex โดยในระบบกำหนดให้มีตัวแม่ (Master) 1 ตัวเพื่อคอยจัดคิวการสื่อสารใน ระบบเครือข่ายและ โดยให้อุปกรณ์ที่เหลือเป็น ตัวลูก (Slave) โดย ตัวลูก (Slave) แต่ละตัวจะมี ที่อยู่ของตัวเอง ถ้าหากตัวแม่ต้องการจะสื่อสารกับ ตัวลูก ก็จะทำการส่งโปรโตคอลออกไป โดยในโปรโตคอล จะ ระบุที่อยู่ของ ตัวลูกที่ต้องการจะสื่อสารออกไปด้วย ตัวลูกทุกตัวที่ต่ออยู่ในเครือข่ายจะรับข้อมูลแล้วเช็คดูว่า ที่อยู่ที่ส่งมานั้น เป็นที่อยู่ของตัวเองหรือไม่ ถ้าเป็นของตัวเองก็จะทำการตอบข้อมูลกลับ หรือ ทำงานตามโปรโตคอลที่กำหนด ซึ่งการสื่อสารวิธีนี้ นิยมใช้กันใน งานอุตสาหกรรม ใช้สื่อสารระหว่าง คอมพิวเตอร์ กับเครื่องจักร หรือ อุปกรณ์ เครื่องมือวัดต่างๆ โดยใช้สายสัญญาณเพียง 2 เส้นก็สามารถสื่อสารข้อมูลกันได้ และมาตรฐานมีความเชื่อถือได้และความสามารถที่เพิ่มขึ้นบนการสื่อสารข้อมูล แบบอนุกรมดังต่อไปนี้.

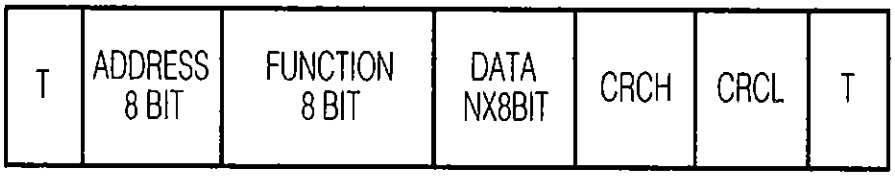
- เพิ่มระยะทางการส่งข้อมูลได้สูงถึง 1200 เมตร (4000 ฟุต ระยะเท่ากันกับมาตรฐาน EIA- 422) [5]
- เพิ่มอัตราความเร็วการส่งข้อมูลได้สูงถึง 10 Mbps (เช่นเดียวกับ EIA-422) [5]
- จำนวนจุดเชื่อมต่ออุปกรณ์สามารถมีได้ถึง 32 จุด ในคู่สายสัญญาณที่ใช้เป็นบัสเดียวกันในการกำหนด ตัวแม่ (Master) ตัวลูก (Slave) ในการจัดทำโครงงานนี้ตัวแม่ จะเป็นคอมพิวเตอร์ ตัวลูกจะเป็นมิเตอร์



รูปที่2.15 การต่อตัวแปลง RS232 กับ RS485 [3]

2.4 ข้อมูลที่มากจาก มิเตอร์ไฟฟ้าแบบดิจิตอล

2.4.1 MODBUS PROTOCOL ของมิเตอร์ไฟฟ้าแบบดิจิตอล รุ่น EPR-04S



รูปที่2.16 ลักษณะข้อมูลที่ออกมาจาก มิเตอร์ไฟฟ้าแบบดิจิตอล [5]

รายละเอียดลักษณะข้อมูลที่ออกจากมิเตอร์มีลักษณะดังรูปที่ 2.16 โดยที่

ค่า T คือเวลาที่สอดคล้องกับข้อมูลต้องไม่ถูกเปลี่ยนแปลงบนบัสการเชื่อมต่อเพื่อให้อุปกรณ์ติดต่อสื่อสารได้รู้ว่า ข้อมูลที่ส่งมาจะสิ้นสุดตอนไหนและข้อมูลที่ส่งมาเริ่มต้นที่ตรงไหน เป็นการบอกให้ทราบส่วนของข้อมูลที่ส่ง ค่า T ต้องไม่ต่ำว่า 3.5 ตัวอักษรที่อัตราการรับ-ส่งที่เลือก ช่วงของ Address จะอยู่ในช่วง 1-247 เป็นค่า Address ของข้อมูลที่ส่งไปทั้งจาก ตัวแม่ (Master) ไป ตัวลูก (slave) และจาก ตัวลูก (slave) ไปยังตัวแม่ (Master) ในการเชื่อมต่อนี้ ให้คอมพิวเตอร์เป็น ตัวแม่ (Master) ส่วนมิเตอร์เป็น ตัวลูก (slave)

ค่า Addressคือค่าที่ระบุตำแหน่งของ รีจิสเตอร์ที่เก็บข้อมูลตัวนั้นเอาไว้ ค่านี้จะบอกว่า รีจิสเตอร์ตัวไหนส่งมา รีจิสเตอร์ตัวนี้ อยู่ที่ไหน

ค่า Function คือค่าที่ตามหลัง Address มาเพื่อระบบว่า ต้องการติดต่อแบบไหน ติดต่อมา เพื่ออ่านค่าที่อยู่ในรีจิสเตอร์นั้น หรือติดต่อมาเพื่อจะเขียนค่าลงในนั้น

ค่า Data Nx8Bit คือข้อมูลธรรมดาที่ส่งออกมาจากมิเตอร์ โดย8บิตจะแทน1ตัวอักษรN แทนจำนวนอักษร

ค่า CRC เป็นวิธีตรวจสอบความผิดพลาด หากใช้ MODBUSโปรโตคอล ค่า CRC นี้ ประกอบไปด้วย 2 ไบต์ ค่า CRC นี้เป็นค่าที่คำนวณมาจากข้อมูลทุกไบต์ ไม่รวมบิตเริ่มบิตหยุดและ บิตตรวจสอบความถูกต้อง โดยที่ตัว ตัวลูก (Slave) ตัวที่ส่งข้อมูลออกมาจะสร้างรหัส CRC แล้วส่ง ตามท้ายไบต์ข้อมูลออกมา หลังจากนั้นเมื่อตัวแม่ (Master) ได้รับเฟรมข้อมูลและถอดข้อมูลออก จากเฟรมแล้วจะทำการคำนวณค่า CRC ตามสูตรเดียวกับตัวลูก (Slave) เพื่อทำการเปรียบเทียบค่า CRC ทั้ง 2 ค่าว่าตรงกันหรือไม่ หากไม่ตรงกันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูล

ค่าพารามิเตอร์ที่ส่งผ่านระบบสื่อสาร จะส่งมาเป็นจำนวน 16 บิตในรูปแบบของ เลขฐาน 16 (Hexadecimal format) เช่น 230V จะส่งมาเป็น 00E6H และค่าตัวประกอบกำลังจะถูกแยกออกเป็น 1.00 ส่วน เช่นถ้าค่าตัวประกอบกำลังมีค่าเท่ากับ 0.98 จะถูกส่งมาเป็น 0062H

ทั้งนี้ข้อกำหนดเฉพาะคือ

1. ใช้สาย 24 AWG หรือ UTP
2. ความต้านทานต่ำว่า 100โอห์ม ต่อกิโลเมตร
3. ลักษณะเฉพาะของค่าอิมพิแดนซ์ ที่ 100 Khz ของ 100 โอห์ม
4. ค่าความจุของการเก็บประจุระหว่างสายกับอากาศต้องน้อยกว่า 60 pF
5. ค่าความจุของการเก็บประจุระหว่างสาย กับสายอื่นๆที่ต่อกับพื้นโลกต้องน้อยกว่า 120 pF
6. ลักษณะของสายที่ต่อ จะต้องพันสายเข้าด้วยกันเป็นเกลียว

การส่งข้อมูลของ MODBUS ของมิเตอร์จะเป็นแบบอะซิงโครนัส (Asynchronous) การรับส่งข้อมูล เป็นการสื่อสารแบบไม่ประสานเวลา เพราะว่าไม่จำเป็นต้อง มีสัญญาณนาฬิกา ร่วมด้วย แต่ต้องทำให้ ตัวส่ง และตัวรับ มีอัตราส่งข้อมูล ที่เท่ากัน รูปแบบข้อมูลแบบอะซิงโครนัสประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น (Start bit) มีขนาด1 บิต
2. บิตข้อมูล (Data) มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบความถูกต้องของข้อมูล (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตหยุด (Stop bit) มีขนาด 1, 1.5, 2 บิตการทดสอบความถูกต้องของค่าพารามิเตอร์ ที่ส่งมาจากตัวมิเตอร์

ค่าพารามิเตอร์ที่ส่งออกมาจากมิเตอร์นั้น ไม่ได้เป็นเลข ฐาน 10 โดยตรง ตัวมิเตอร์จะส่งออกมาในรูปแบบของเลขฐาน 16 เป็นลักษณะข้อมูลตามโปรโตคอล Modbus เราจึงต้องมีการแปลงค่าของข้อมูลที่ส่งออกมานั้น ให้ผู้ใช้งานสามารถอ่านค่าได้อย่างง่าย และการที่มิเตอร์จะส่งข้อมูลมาให้นั้น จะต้องมีการส่งคำสั่งร้องขอไปที่ตัวมิเตอร์ คำสั่งที่ส่งออกไป จะเป็นไปตาม ข้อมูลจำเพาะของตัวมิเตอร์ จะแสดงในรูปที่ 2.17

Available Modbus Function:

03H	READ HOLD REGISTERS
06H	PRESET SINGLE REGISTER
10H	PRESET MULTIPLE REGISTERS

รูปที่ 2.17 ฟังก์ชั่นคำสั่งของมิเตอร์

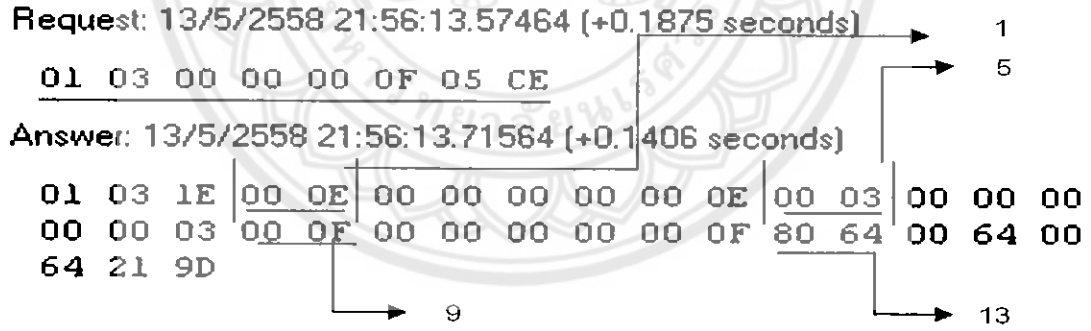
โดยที่

03 หมายถึง การเรียกอ่านค่าในรีจิสเตอร์ และให้รีจิสเตอร์นั้นคงค่าเดิมไว้

06 หมายถึง การปรับตั้งค่ารีจิสเตอร์เพียงตัวเดียวและอ่านข้อมูล

10 หมายถึง การปรับตั้งค่ารีจิสเตอร์มากกว่า 1 ตัวและอ่านข้อมูล

เมื่อมีการส่งการร้องขออ่านข้อมูลภายในตัวมิเตอร์ มิเตอร์จะตรวจสอบลักษณะคำสั่ง แล้วตอบสนองการร้องขอ ออกมาเป็นข้อมูลในลักษณะของ Modbus เช่นกัน ดังรูปที่ 2.18 โดยใช้โปรแกรม monitor device [11] ในการส่งข้อมูลจากชุดทดสอบ ตั้งค่าให้ไม่มีการส่งบิตตรวจสอบความถูกต้อง และฟังก์ชั่นที่ใช้จะเป็นการอ่านค่าจากมิเตอร์เท่านั้น ข้อมูลที่ได้จะเป็นในลักษณะรูปที่ 2.18



รูปที่ 2.18 ลักษณะของข้อมูลที่สื่อสารกันระหว่างคอมพิวเตอร์กับมิเตอร์

จากรูปที่ 2.18 ข้อมูล 01 03 00 00 00 0F คือข้อมูลที่ส่งออกจากคอมพิวเตอร์เป็นลักษณะของคำสั่ง แปลง่ายๆว่า ขออ่านข้อมูลภายใน ตัวลูก (คือมิเตอร์) ที่อยู่หมายเลข 1 โดยขออ่านข้อมูลจากรีจิสเตอร์หมายเลข 00 ถึง รีจิสเตอร์ตัวถัดไปอีก 15 ตำแหน่ง โดยที่ความหมายคำสั่งของแต่ละตำแหน่ง คือ

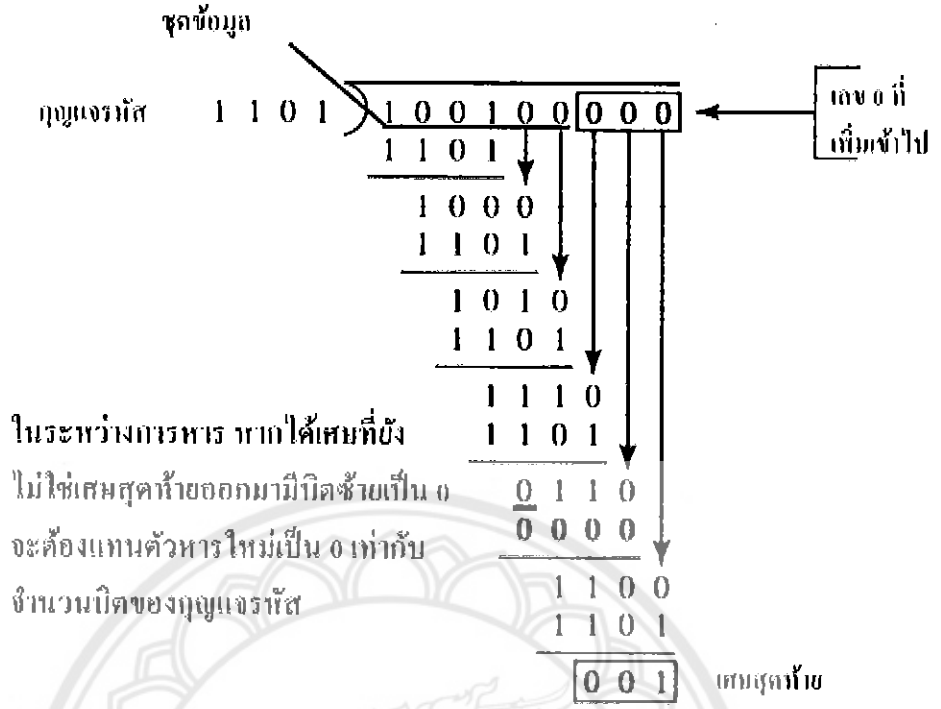
- ตำแหน่งที่ 1 01 คือค่า Address ของมิเตอร์
- ตำแหน่งที่ 2 03 คือฟังก์ชั่นขออ่านค่าจากมิเตอร์ และคงค่านั้นไว้

- ตำแหน่งที่ 3 00 คือ Address ของรีจิสเตอร์ตัวแรกที่ต้องการอ่านค่าเป็นบิตที่มีนัยสำคัญสูงสุด (MSB)
- ตำแหน่งที่ 4 00 คือ Address ของรีจิสเตอร์ที่มีนัยสำคัญต่ำที่สุด (LSB) Address ของรีจิสเตอร์สามารถดูได้จากตารางที่ 2.1
- ตำแหน่งที่ 5 00 คือจำนวนรีจิสเตอร์ที่ต้องการอ่านค่าเป็น (MSB)
- ตำแหน่งที่ 6 0F คือจำนวนรีจิสเตอร์ที่ต้องการอ่านค่าแต่เป็น (LSB) ทั้งหมด 15 ค่า จากตำแหน่ง 00H ถึง 0EH ในตารางที่ 2.1
- ตำแหน่งที่ 7 05 คือค่า CRC 2ไบต์ โดยจะส่ง (LSB) ออกมาก่อน
- ตำแหน่งที่ 8 CE คือ CRC เหมือนกันแต่เป็น (MSB)

CRC สามารถคำนวณด้วยโปรแกรม On-line CRC calculation [10] โดยหลักการคำนวณคือ เมื่อภาคส่งมีข้อมูล ที่ต้องการจะส่ง ชุดหนึ่ง แล้วต้องการใช้การตรวจสอบความถูกต้องด้วย CRC จะต้องกำหนด กุญแจรหัสขึ้นมา 1 ค่า เป็นค่าอะไรก็ได้ แต่ต้องเป็นเลข ฐาน 2 และมีบิตทางซ้ายสุดเป็น 1 เช่น ข้อมูลที่ต้องการจะส่งคือ 100100 และสมมุติกุญแจรหัสเป็น 1101 ให้นับจำนวนบิตของกุญแจรหัสแล้วลบออกไป 1 บิต จากตัวอย่าง จำนวนบิตคือ 4 บิต (1101) ลบออก 1 จะเหลือ 3 บิต ให้แทน 3 บิตเป็นเลข 0 แล้วนำไปต่อท้ายข้อมูลที่จะส่ง ดังนั้นจะได้ข้อมูลทั้งหมดที่จะส่งเป็น 100100000 แล้วนำขึ้นตั้งหารยาวด้วย กุญแจรหัสที่กำหนด ด้วยวิธีการ xor (ซอร์) จะได้เศษออกมาเป็นค่าๆหนึ่ง จากตัวอย่างจะได้เศษเป็น 001 ให้นำเศษที่ได้ ไปแทนเลข 0 จำนวน 3 ตัวที่ต่อเข้ากับข้อมูลที่จะส่งในตอนแรก เพราะฉะนั้น ข้อมูลที่ส่งออกไปจะเป็น 100100001 เมื่อส่งข้อมูลไปหาภาครับ ภาครับจะนำข้อมูลชุดนั้นมา ตั้ง แล้วหารด้วย กุญแจรหัสชุดเดียวกัน ด้วยวิธีการ xor เหมือนกัน ถ้าหารแล้ว ไม่เหลือเศษ แสดงว่าข้อมูลที่ส่งมานั้นไม่มีการผิดพลาดระหว่างทางที่ส่ง หากหารแล้วเหลือเศษ แสดงว่าข้อมูลที่ส่งมามีความผิดพลาดเกิดขึ้น โดย Mosbus เป็น CRC แบบ CRC16 วิธีการตั้งหารและการ xor ดูได้จากรูปที่ 2.19 และ 2.20

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

รูปที่ 2.19 ผลลัพธ์ที่เกิดขึ้นจากการ xor

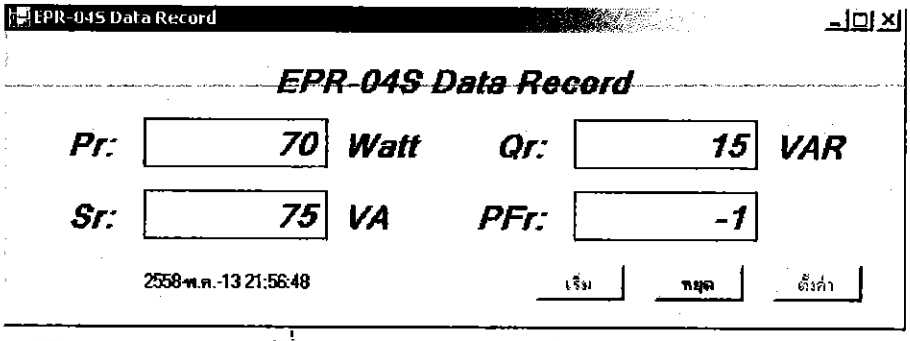


รูปที่ 2.20 วิธีการตั้งหาร

ส่วนข้อมูลด้านล่าง คือข้อมูลที่มิเตอร์ตอบกลับมา 3 ไบต์แรกคือค่า Address 01 และฟังก์ชั่น 03 ตามด้วยจำนวลข้อมูลที่ส่งมา 1E การนับตำแหน่งข้อมูล จะเริ่มนับหลังจากมิเตอร์บอกจำนวนข้อมูลที่ส่งมา นั่นคือนับหลังจาก 1E ครั้งละ 2 ไบต์ เท่ากันทั้งหมด ถ้าอยากจะทราบตำแหน่งข้อมูลว่า แต่ละตำแหน่งมาจากรีจิสเตอร์ตัวไหน และหมายถึงค่าอะไรสามารถดูได้ในตารางที่ 2.1 ส่วน 2 ตำแหน่งสุดท้ายคือ ค่า CRC จำนวน 2 ไบต์คือ 21 9D และจะส่ง LSB ก่อนตามด้วย MSB โดยที่

- ข้อมูลตำแหน่งที่ 1 คือ 00 0E หมายถึงค่ากำลังไฟฟ้าจริง คำนวณ ออกมาเท่ากับ 14 นำไปคูณกับอัตราทด CT VT แล้วคูนต่อด้วย 0.1 ซึ่งอัตราทด CT คือ 5 ถ้าหากไม่ได้ต่อ VT มิเตอร์จะตั้งเป็น 10 เพราะฉะนั้นกำลังไฟฟ้าจริงจะเท่ากับ 14 x 5 x 10 x 0.1 = 70 W .
- ข้อมูลตำแหน่งที่ 5 คือ 00 03 หมายถึง กำลังไฟฟ้าจินตภาพ ใช้สูตรเดียวกัน เพราะฉะนั้นจะได้กำลังไฟฟ้าจินตภาพเท่ากับ 3 x 5 x 10 x 0.1 =15 VAR .
- ข้อมูลตำแหน่งที่ 9 คือ 00 0F หมายถึงค่ากำลังไฟฟ้าปรากฏ ใช้สูตรเดียวกัน เพราะฉะนั้นจะได้กำลังไฟฟ้าปรากฏเท่ากับ 15 x 5 x 10 x 0.1 = 75 VA .
- ข้อมูลตำแหน่งที่ 13 คือ 08 64 หมายถึงค่าตัวประกอบกำลังไฟฟ้า 08 หมายถึงค่ากระแสไฟฟ้าล้าหลังแรงดัน (Lagging) ถ้ามิเตอร์ส่ง 08 มา จะหมายถึง Lagging ส่วนค่า 64

คือค่าตัวประกอบกำลัง คำนวณออกมาเท่ากับ 100 แล้วนำไปหาร 100 เพราะฉะนั้นจะได้ค่าตัวประกอบกำลังเท่ากับ 100/100 = 1 (lagging) เทียบการแสดงผลได้จากรูปที่ 2.1



รูปที่ 2.21 การแสดงผลของโปรแกรม

ตารางที่ 2.1 หมายเลขรีจิสเตอร์และความหมายของข้อมูล [3]

ADDRESS	DESCRIPTION	SIZE (BYTE)	ADDRESS	DESCRIPTION	SIZE (BYTE)
00H	R-ACTIVE POWER	2	1DH	S-REACTIVE POWER MAX	2
01H	S-ACTIVE POWER	2	1EH	S-REACTIVE POWER MIN	2
02H	T-ACTIVE POWER	2	1FH	T-REACTIVE POWER MAX	2
03H	TOTAL ACTIVE POWER	2	20H	T-REACTIVE POWER MIN	2
04H	R-REACTIVE POWER	2	21H	R-APPARENT POWER MAX	2
05H	S-REACTIVE POWER	2	22H	R-APPARENT POWER MIN	2
06H	T-REACTIVE POWER	2	23H	S-APPARENT POWER MAX	2
07H	TOTAL REACTIVE POWER	2	24H	S-APPARENT POWER MIN	2
08H	R-APPARENT POWER	2	25H	T-APPARENT POWER MAX	2
09H	S-APPARENT POWER	2	26H	T-APPARENT POWER MIN	2
0AH	T-APPARENT POWER	2	27H	ACTIVE ENERGY kWh.-1	2
0BH	TOTAL APPARENT POWER	2	28H	ACTIVE ENERGY kWh.-2	2
0CH	R-COS	2	29H	ACTIVE ENERGY kWh.-3	2
0DH	S-COS	2	2AH	ACTIVE ENERGY kWh.-4	2
0EH	T-COS	2	2BH	ACTIVE ENERGY kWh.(EXPORT) -1	2
0FH	WATT DEMAND	2	2CH	ACTIVE ENERGY kWh.(EXPORT) -2	2
10H	V Ar DEMAND	2	2DH	ACTIVE ENERGY kWh.(EXPORT) -3	2
11H	VA DEMAND	2	2EH	ACTIVE ENERGY kWh.(EXPORT) -4	2
12H	WATT DEMAND MAX	2	2FH	REACTIVE ENERGY kVArhL-1	2
13H	V Ar DEMAND MAX	2	30H	REACTIVE ENERGY kVArhL-2	2
14H	VA DEMAND MAX	2	31H	REACTIVE ENERGY kVArhL-3	2
15H	R-ACTIVE POWER MAX	2	32H	REACTIVE ENERGY kVArhL-4	2
16H	R-ACTIVE POWER MIN	2	33H	REACTIVE ENERGY kVArhC-1	2
17H	S-ACTIVE POWER MAX	2	34H	REACTIVE ENERGY kVArhC-2	2
18H	S-ACTIVE POWER MIN	2	35H	REACTIVE ENERGY kVArhC-3	2
19H	T-ACTIVE POWER MAX	2	36H	REACTIVE ENERGY kVArhC-4	2
1AH	T-ACTIVE POWER MIN	2	37H	VOLTAGE TRANS. RATIO	2
1BH	R-REACTIVE POWER MAX	2	38H	CURRENT TRANS. RATIO	2
1CH	R-REACTIVE POWER MIN	2	39H	DEMAND TIME (SEC.)	2

ตารางที่ 2.2 สูตรที่ใช้แปลงข้อมูล [3]

MODBUS REGISTER MAP				
REGISTER	R/W	RANGE	UNIT	MULTIPLIER
L1 PHASE ACTIVE POWER	R	(-18000 - 18000)xCTxVT	Watt	0.1
L2 PHASE ACTIVE POWER	R	(-18000 - 18000)xCTxVT	Watt	0.1
L3 PHASE ACTIVE POWER	R	(-18000 - 18000)xCTxVT	Watt	0.1
L1 PHASE REACTIVE POWER	R	(-18000 - 18000)xCTxVT	Var	0.1
L2 PHASE REACTIVE POWER	R	(-18000 - 18000)xCTxVT	Var	0.1
L3 PHASE REACTIVE POWER	R	(-18000 - 18000)xCTxVT	Var	0.1
L1 PHASE APPARENT POWER	R	(0 -18000)xCTxVT	VA	0.1
L2 PHASE APPARENT POWER	R	(0 -18000)xCTxVT	VA	0.1
L3 PHASE APPARENT POWER	R	(0 -18000)xCTxVT	VA	0.1
L1 PHASE COSφ	R	(-100-100)	-	0.01
L2 PHASE COSφ	R	(-100-100)	-	0.01
L3 PHASE COSφ	R	(-100-100)	-	0.01
TOTAL IMPORT ACTIVE POWER	R	(0 -54000)xCTxVT	Watt	0.1
TOTAL EXPORT ACTIVE POWER	R	(0 -54000)xCTxVT	Watt	0.1
TOTAL INDUCTIVE REACTIVE POWER	R	(0 -54000)xCTxVT	Var	0.1
TOTAL CAPACITIVE REACTIVE POWER	R	(0 -54000)xCTxVT	Var	0.1
TOTAL APPARENT POWER	R	(0 -54000)xCTxVT	VA	0.1
AVERAGE INDUCTIVE COSφ	R	(-1000 -1000)	-	0.001

2.4.2 การตั้งค่า อัตราการส่งข้อมูล (Baud Rate)

Baud Rate คือ อัตราการส่งข้อมูล ความเร็วของการรับ-ส่งข้อมูล นับเป็นเป็นจำนวนบิตต่อวินาที (bps) เช่น 1,200 2,400 4,800 9,600 14,400 19,200 38,400 bit per second เป็นต้นการเลือกอัตราการส่งข้อมูลขึ้นอยู่กับ ชนิดของสายสัญญาณ ระยะทาง และปริมาณสัญญาณรบกวน สำหรับมิเตอร์ไฟฟ้าแบบดิจิตอลรุ่น EPR-04S สามารถเลือก อัตราการส่งข้อมูลได้ระหว่าง 1,200 ถึง 38,400bps แล้วแต่ลักษณะการใช้งานของผู้ใช้และสามารถตั้งค่าให้มีการเช็ค บิตตรวจสอบความถูกต้องให้เป็นการเช็คแบบคู่การเช็คแบบคี่ หรือไม่เช็คเลยก็ได้

2.4.3 การเช็ค บิตตรวจสอบความถูกต้องของข้อมูล (Parity bit)

การเช็คบิตตรวจสอบความถูกต้อง คือ การเพิ่มบิตเข้าไปเข้าไป 1 บิตให้กับทุกๆ 8 บิตของข้อมูลจนกลายเป็น 9 บิต บิตที่เพิ่มขึ้นไม่ใช่ข้อมูล แต่ใส่เพื่อตรวจสอบว่า ข้อมูลมีความผิดพลาดหรือไม่ โดยใช้หลักการนับจำนวนบิตข้อมูลที่มีค่าเป็น 1 ในทุกๆ 8 บิต การเช็คบิตตรวจสอบความถูกต้องนี้แบ่งได้ 2 วิธี คือ บิตตรวจสอบความถูกต้องคู่ หรือ บิตตรวจสอบความถูกต้องคี่

บิตตรวจสอบความถูกต้องคู่ หมายถึง จำนวนบิตที่เป็น 1 เป็นเลขคี่ การคำนวณก็นับจำนวนบิตที่เป็น 1 ในข้อมูลจริง 8 บิตแรกว่ามีจำนวนเป็นเลขคี่หรือเลขคู่ ถ้าได้เลขคี่ให้เติมบิตตรวจสอบความถูกต้องเป็น 0 ถ้านับได้เลขคู่ ก็เติมบิตตรวจสอบความถูกต้องเป็น 1 เพื่อให้จำนวนเป็นเลขคี่ เช่น 101101 มีจำนวนบิต 1 เป็น 4 ตัว ซึ่งเป็นจำนวนคู่ดังนั้น parity bit จะต้องเป็น 1 เพื่อให้จำนวนเป็นคี่ 1101101 หรือ 101100 ก็จะเติมเลข 0 เข้าไป เพื่อที่จะให้เลข 1 เป็นจำนวนคี่อยู่ เช่น 0101100

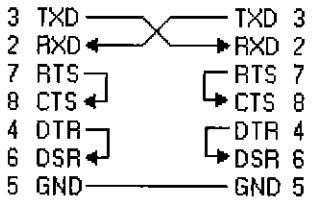
การส่งข้อมูลข้ามกันระหว่างเครื่องคอมพิวเตอร์ โดยที่เครื่องทำหน้าที่เป็นตัวรับข้อมูลจะต้องทราบล่วงหน้าว่าเครื่องที่ส่ง ข้อมูลมาใช้ บิตตรวจสอบความถูกต้องแบบไหนแบบบิตตรวจสอบความถูกต้องคู่ หรือ บิตตรวจสอบความถูกต้องคี่ ถ้าไม่ทราบก็จะทำให้แปลความหมายของข้อมูลที่รับเข้ามาไม่ถูกต้อง เพราะความเข้าใจไม่ตรงกัน

ข้อเสียของบิตตรวจสอบความถูกต้อง การใช้บิตตรวจสอบความถูกต้องคือ เสียเวลา และไม่ได้ประโยชน์มากเท่าที่ควร เพราะไม่สามารถบอกได้ว่าข้อมูลที่ได้มามีความผิดพลาดที่ตำแหน่งไหน และไม่สามารถแก้ไขข้อผิดพลาดได้ บิตตรวจสอบความถูกต้องบอกได้แค่เพียงว่ามีความผิดพลาดเกิดขึ้นเท่านั้น และ ถ้าข้อมูลเกิดผิดพลาด 2 บิต เช่น 10001001 เปลี่ยนเป็น 10101011 ก็ไม่สามารถเช็คข้อผิดพลาดโดยใช้วิธีตรวจสอบความถูกต้องนี้ได้เพราะฉะนั้นการใช้บิตตรวจสอบความถูกต้อง ขึ้นอยู่กับลักษณะข้อมูลและผู้ใช้งานกำหนด มีข้อดีและข้อเสียแตกต่างกัน

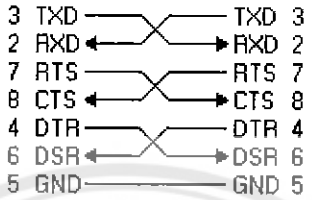
2.4.4 การแปลง RS485 ไปเป็น RS232

การแปลงจาก RS485 ไปเป็น RS232 จะต้องแปลงผ่านหัวแปลง ซึ่งมีจำหน่ายในร้านอุปกรณ์คอมพิวเตอร์ทั่วไป หัวแปลงจะทำหน้าที่แปลงสัญญาณเพื่อที่จะให้ข้อมูลที่มาในรูปแบบของ RS485 ไปเป็นแบบ RS232 เพราะว่า พอร์ตของคอมพิวเตอร์ส่วนใหญ่จะเป็น RS232 พอร์ต RS232 จะมีหัวเชื่อมต่อ DB9 ทำหน้าที่เป็นพอร์ต Input หรือ Output

รูปที่ 2.22 หัวแปลงจาก RS485 ไปเป็น RS232



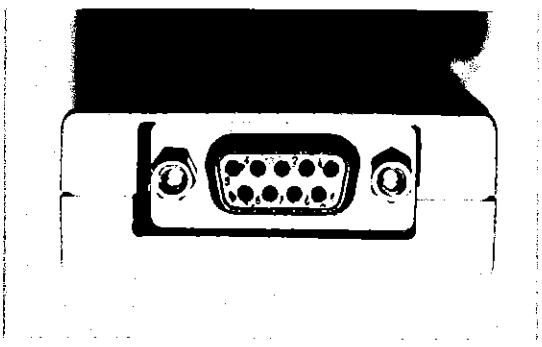
รูปที่ 2.23 การเชื่อมต่ออุปกรณ์อุปกรณ์เข้ากับคอมพิวเตอร์ด้วยหัวเชื่อมDB9ในมาตรฐาน RS232 แบบ3 เส้น



รูปที่ 2.24 การเชื่อมต่ออุปกรณ์เข้ากับคอมพิวเตอร์ด้วยหัวเชื่อมDB9ในมาตรฐาน RS232 แบบ NULL MODEM

2.4.5 การทำงานของขาสัญญาณDB9

ในส่วนของมาตรฐาน RS285 ที่มีหัวเชื่อมแบบDB9จะมีขา RXD TXD และ GND ให้นำขาที่เหมือนกันของทั้งคู่ ต่อเข้าด้วยกันขา TXD เป็นขาที่ในการส่งข้อมูล ขา RXD เป็นขาที่ใช้รับข้อมูล ขา DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน DSR ตรวจสอบว่าพอร์ต ที่ติดต่อด้วย เปิดอยู่หรือไม่เมื่อเปิดพอร์ตอนุกรมขา DTR จะเปิดเพื่อให้อุปกรณ์ทราบว่าต้องการติดต่อด้วย ในขณะเดียวกันก็จะตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่ RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล CTS ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ ต้องการส่งข้อมูลหรือไม่เมื่อต้องการส่งข้อมูลขา RTS จะเปิดและจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จก็จะปิด ในขณะเดียวกันก็จะตรวจสอบขา CTS ว่าอุปกรณ์ต้องการที่จะส่งข้อมูลหรือไม่ส่วนขา GND คือขากราวด์ ใช้เป็นขาอ้างอิงในการเปรียบเทียบระดับแรงดัน



รูปที่ 2.25 หัวเชื่อม DB9 ของตัวแปลง RS485 เป็น RS232

บทที่ 3
วิธีการดำเนินงาน

การออกแบบและการพัฒนาโปรแกรมเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล มีวิธีการดำเนินงาน 4 ขั้นตอนหลักๆ ดังนี้ คือ การศึกษาการทำงาน การสร้างชุดทดสอบ และการทดสอบโปรแกรม

3.1 การศึกษาการทำงาน

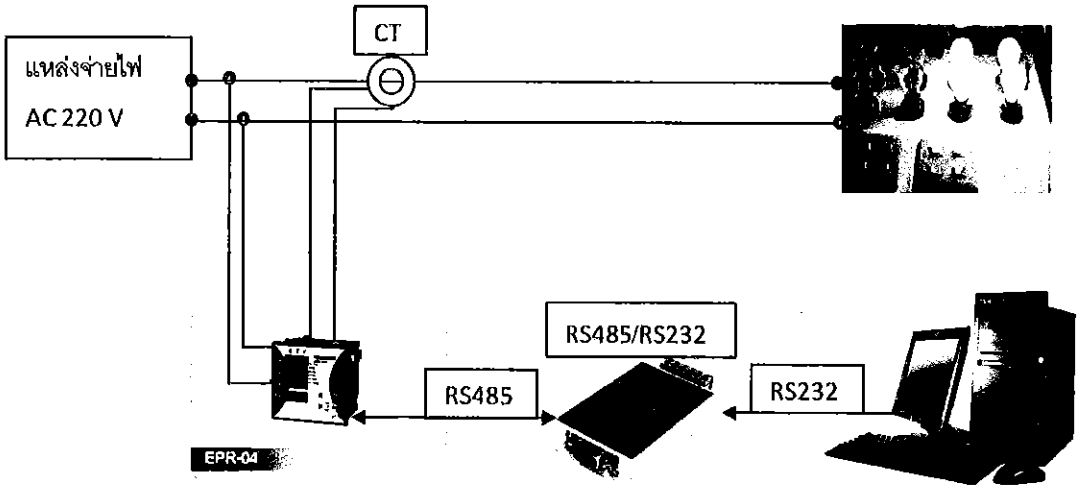
การศึกษาการทำงานแบ่งเป็น 3 ส่วนที่สำคัญ คือ การ ศึกษาการทำงานมิเตอร์ไฟฟ้าแบบดิจิตอล รุ่น EPR-04s การศึกษาการสื่อสารระหว่างมิเตอร์กับคอมพิวเตอร์ และการศึกษาการเขียนโปรแกรมด้วยภาษา Visual basic โดยใช้โปรแกรม Visual basic 2010

3.2 การสร้างชุดทดสอบ

การออกแบบและการสร้างชุดทดสอบ เพื่อ พัฒนาโปรแกรมเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอลนั้น ได้แบ่งการสร้างชุดทดสอบเป็น 2 ส่วนหลักๆคือ ส่วนฮาร์ดแวร์ (Hardware) และ ซอฟแวร์ (Software)

3.2.1 การสร้างชุดทดสอบในส่วนของฮาร์ดแวร์

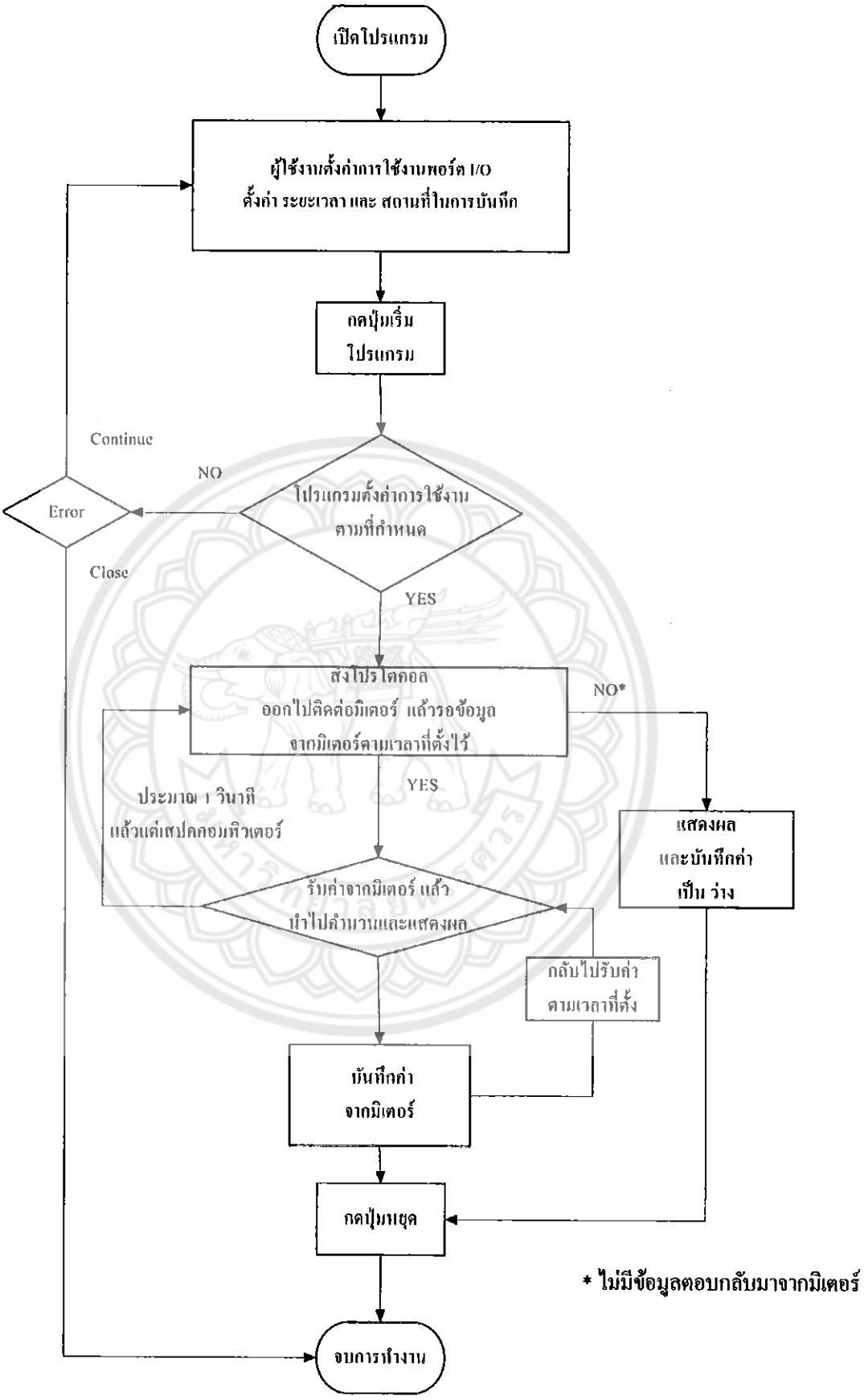
ในส่วนของฮาร์ดแวร์ จะต้องมีการต่อมิเตอร์ไฟฟ้าแบบดิจิตอลและแหล่งจ่ายไฟ เข้ากับภาระทางไฟฟ้า และระบบสื่อสารของตัวมิเตอร์ เพื่อที่จะได้ทำการพัฒนาโปรแกรมเก็บค่าและแสดงผลแบบติดตาม เนื่องจากในที่นี้เป็นการทดสอบในห้องทดลอง ภาระที่ใช้นั้นจึงเลือกเป็นหลอดไฟฟ้าแบบไส้ เพราะว่าหลอดไส้นั้น กินค่าพลังงานต่อหลอดสูงพอสมควร และเมื่อเปิดไปนานๆ จะเกิดความร้อนสะสม ทำให้ตัวประกอบกำลังทางไฟฟ้าเปลี่ยนไป จึงได้เลือกมาทำการทดลอง ส่วนของระบบสื่อสาร จะใช้สายสัญญาณชนิดสายคู่ตีเกลียว มาเป็นสายในการส่งข้อมูล เพราะว่าสายชนิดนี้มีการสูญเสียต่ำ และ การเพี้ยนของข้อมูลจึงได้เลือกมาทำการทดลอง การสร้างชุดทดสอบในส่วนฮาร์ดแวร์จะแสดงในรูปที่ 3.1



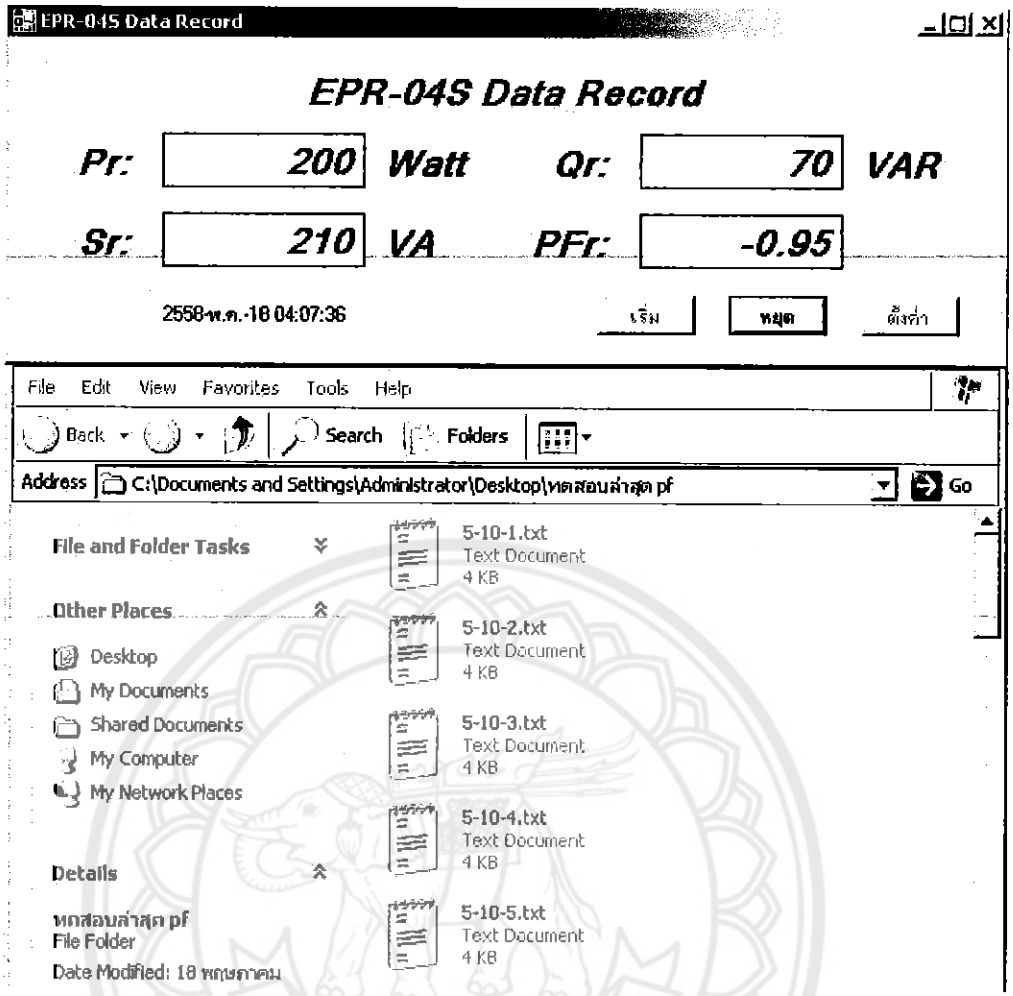
รูปที่ 3.1 บล็อกไดอะแกรมการออกแบบชิ้นงาน

3.2.2 การสร้างชุดทดสอบในส่วนของซอฟต์แวร์

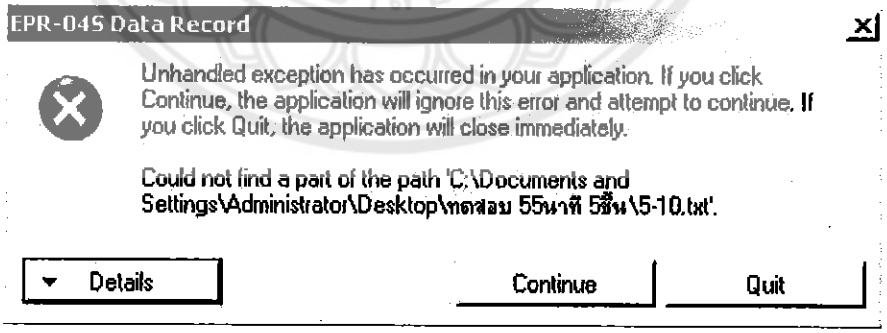
ในการออกแบบการทำงานของโปรแกรมเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอล จะต้องมีการ ส่งข้อมูลระหว่างมิเตอร์ไฟฟ้ากับตัวโปรแกรม ซึ่งจะต้องมีการร้องขอจากโปรแกรมเพื่อที่จะเข้าไปอ่านข้อมูลในตัวมิเตอร์และมิเตอร์จะส่งข้อมูลกลับมา เมื่อมีการร้องขอถูกต้อง การทำงานของโปรแกรมจะเป็นไปตามแผนภาพการทำงานดังที่แสดงในรูปที่ 3.2 การทำงานของโปรแกรม จะถูกอธิบายไว้ในแผนภาพการทำงานจากรูปที่ 3.2 จะเห็นได้ว่าเมื่อเปิดโปรแกรมขึ้นมา จะต้องทำการตั้งค่าการใช้งานเริ่มต้นของพอร์ต I/O เมื่อทำการตั้งค่าเสร็จ ให้กดปุ่มเริ่มทำงาน หลังจากนั้นโปรแกรมจะใช้เวลาในการประมวลผลครู่หนึ่ง และโปรแกรมจะทำการตั้งค่าพอร์ต และ รูปแบบในการบันทึก ตามที่ผู้ใช้ได้กำหนด เมื่อการตั้งค่าถูกต้อง โปรแกรมจะทำการแสดงค่า และ บันทึกผลตามที่ผู้ใช้ตั้งไว้ เมื่อต้องการหยุดการแสดงค่า และ บันทึกผล ผู้ใช้สามารถกดปุ่ม หยุดการทำงานของโปรแกรมได้ทันที หรือจะปรับเปลี่ยนการตั้งค่าใหม่ ก็สามารถทำได้หากการตั้งค่าเริ่มต้นการใช้งานไม่ถูกต้อง โปรแกรมจะประมวลผลหาความผิดพลาดจากการตั้งค่าการสื่อสารไม่ตรงกับมิเตอร์ โปรแกรมไม่แสดง Error แต่จะทำงานต่อ และจะบันทึกค่ากับแสดงผลเป็น 0 หากเป็น Error อย่างอื่น เช่น ไฟล์ที่ทำการบันทึกหายไปขณะบันทึก โปรแกรมจะแสดง Error ออกมา ซึ่งผู้ใช้งานต้องกลับไปแก้ไขด้วยตนเอง ดังรูปที่ 3.4



รูปที่ 3.2 ภาพรวม การทำงานของโปรแกรม

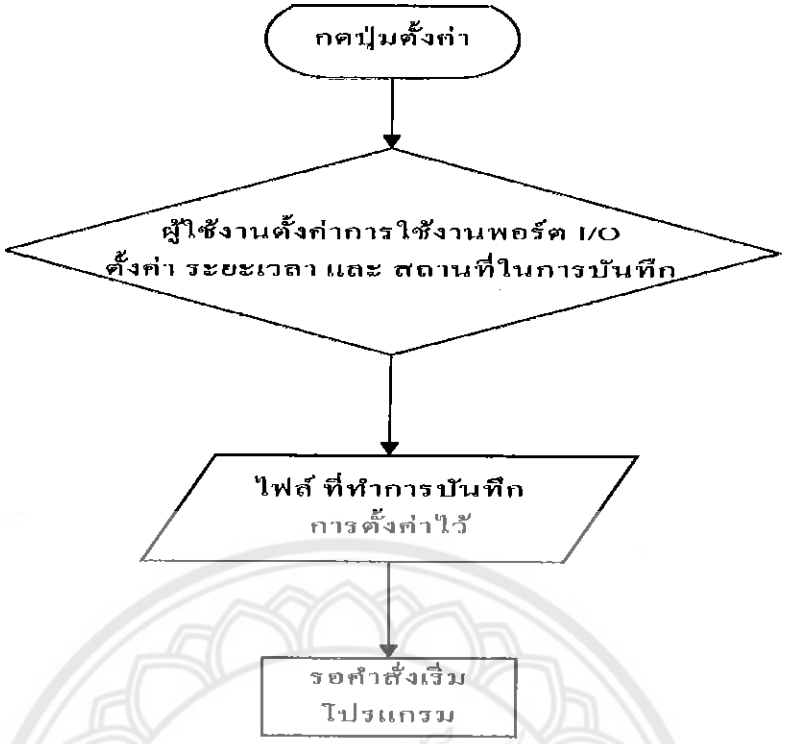


รูปที่ 3.3 ผลจากการทำงานของโปรแกรม

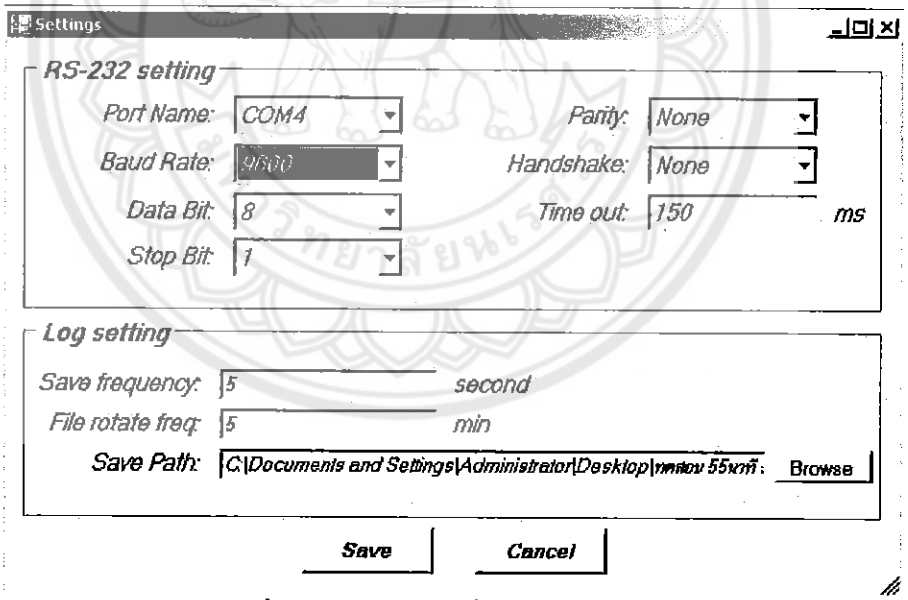


รูปที่ 3.4 Error ของโปรแกรม เมื่อไฟล์ที่ทำการบันทึกหายไปขณะบันทึก

ในการทำงานของโปรแกรม ยังสามารถแบ่งเป็นโปรแกรมย่อยออกเป็น 2 ส่วนหลักๆคือ ส่วนตั้งค่าการใช้งาน และส่วนประมวลผลและบันทึกผล ซึ่งจะมีรายละเอียดการทำงานลึกลงไปอีกระดับมากกว่า ภาพรวมในการทำงานของโปรแกรม ส่วนแรกการตั้งค่ามีลักษณะดังรูปที่ 3.5



รูปที่ 3.5 การตั้งค่าของโปรแกรม



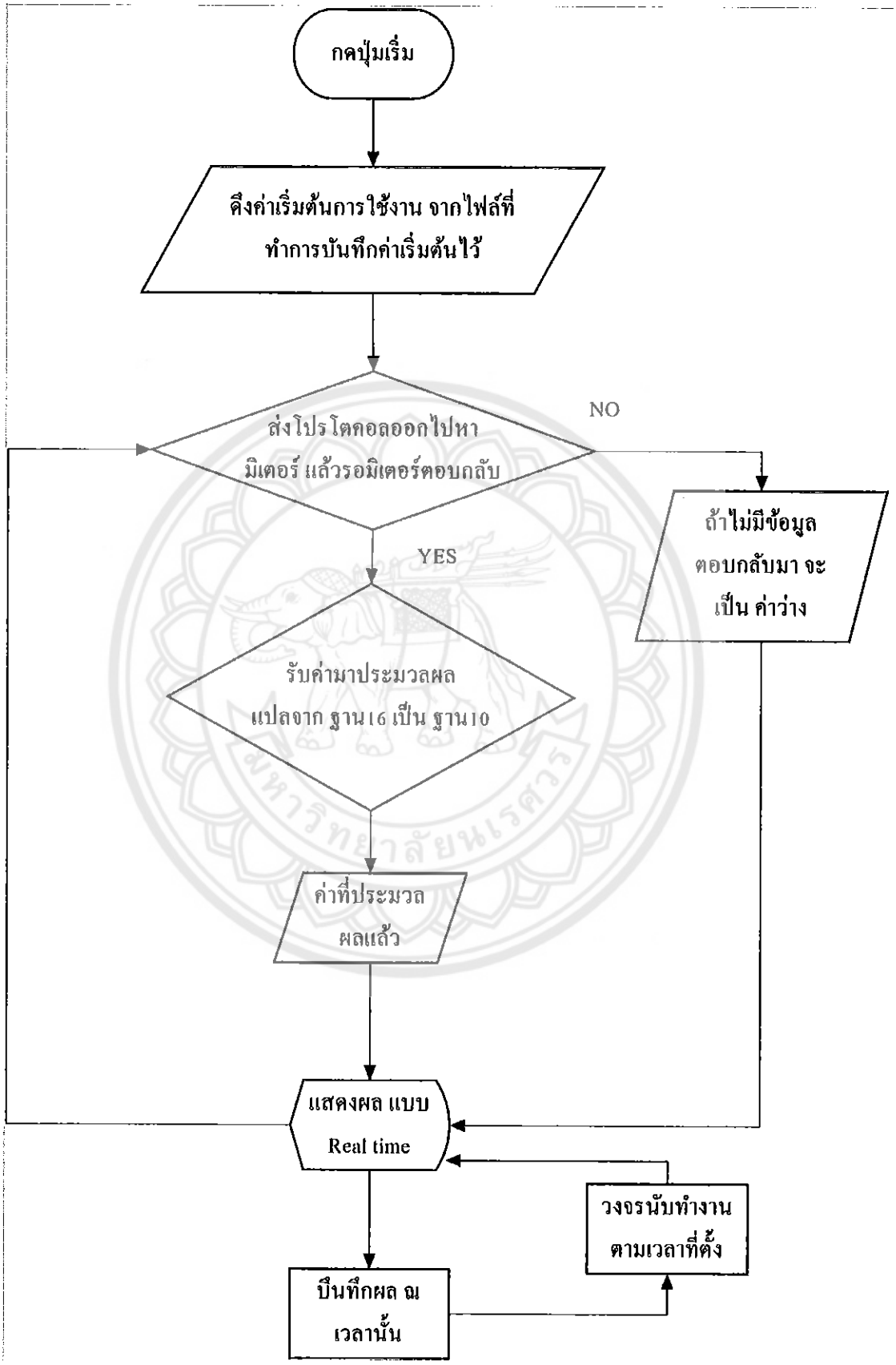
รูปที่ 3.6 หน้าต่างการตั้งค่าของโปรแกรม

ในตัวโปรแกรมนั้น ผู้เขียนออกแบบให้โปรแกรม สามารถตั้งค่าต่างๆ ที่ใช้ในการสื่อสารกับมิเตอร์ รวมทั้งการตั้งค่าเพื่อบันทึกผลของมิเตอร์ได้ ในการปรับตั้งต่างๆ รายละเอียดของการปรับตั้งมีดังนี้

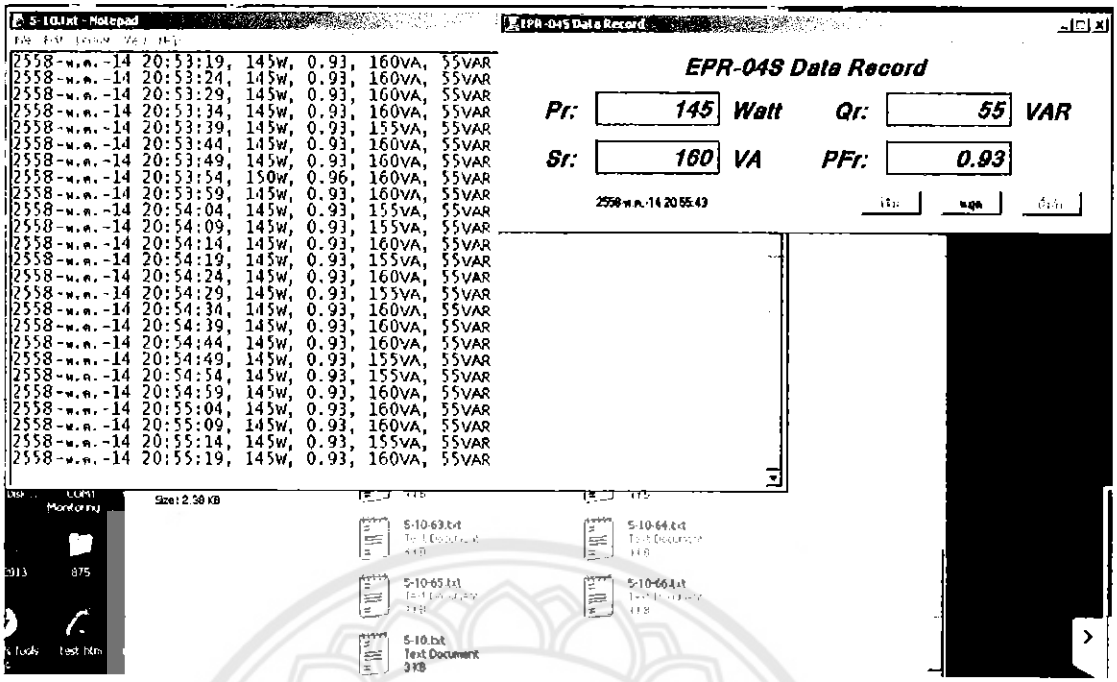
1. Port Name :หมายเลขพอร์ตที่เชื่อมต่อ
2. Baud rate : อัตราการส่งข้อมูล (bit per second)
3. Data bit : ความยาวของชุดข้อมูล
4. Stop bit : บิตหยุด
5. Parity : บิตเช็คความถูกต้อง
6. Hand shake : รูปแบบการเชื่อมต่อสาย เพื่อเช็คความถูกต้อง
7. Time out : การหน่วงเวลาเพื่อรอคำตอบจากมิเตอร์
8. Save Frequency :ระยะเวลาในการบันทึกหนึ่งข้อมูล หน่วยเป็นวินาที
9. File Rotate freq : ระยะเวลาในการขึ้นไฟล์ใหม่ หน่วยเป็นนาที
10. Save Path : สถานที่เก็บไฟล์ในการบันทึกค่า

เมื่อทำการตั้งค่าการใช้งานเริ่มต้น โปรแกรมจะทำการบันทึกค่าเริ่มต้น ลงใน Text file ชื่อ EPR-04s.exe.config เพื่อรอการดึงไปใช้งานในส่วนต่อไป ซึ่งเป็นส่วนของ การส่งโปรโตคอลออกไปหามิเตอร์เพื่อรับข้อมูลและบันทึกผล

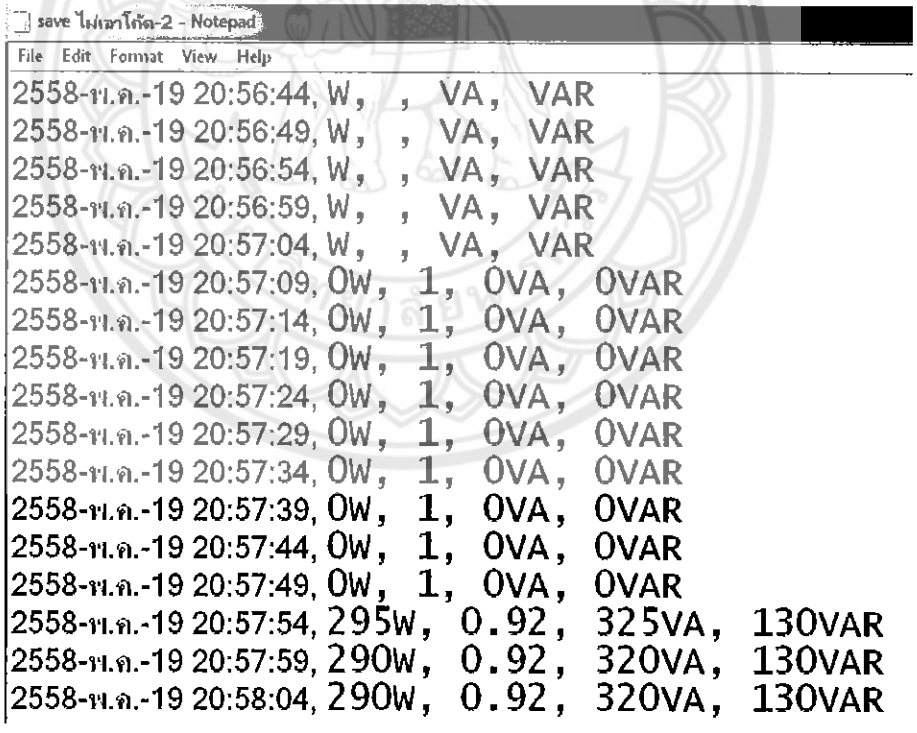
ส่วนที่สอง ส่วนประมวลผลและบันทึกผล ในส่วนนี้ โปรแกรมจะดึงการตั้งค่ามาใช้ แล้วส่งโปรโตคอลออกไปตามรูปแบบของ Modbus หลังจากนั้นจะทำการประมวลผลข้อมูลออกมาเพื่อแสดงผล และบันทึกผล เมื่อทำงานครบตามคำสั่ง โปรแกรมจะส่งข้อความหรือคำสั่งออกไปอีกเพื่อรับค่ามาแสดงผลใหม่ ส่วนการบันทึกนั้น จะดึงค่าออกมาจากการแสดงผล ตามระยะเวลาที่ตั้งไว้ เมื่อบันทึกข้อมูลครบตามระยะเวลา โปรแกรมจะขึ้นไฟล์ใหม่ เพื่อทำการบบันทึกข้อมูลต่อไป



รูปที่3.7 ส่วนแสดงผลและบันทึกผล



รูปที่ 3.8 การบันทึกผลและแสดงผลของโปรแกรม



รูปที่ 3.9 เมื่อไม่มีการตอบกลับจากมิเตอร์ จะบันทึกค่าว่างลงไป

3.3 รายละเอียดการทดสอบชิ้นงาน

การทดสอบชิ้นงาน จะเป็นการทดสอบการทำงานของโปรแกรมและชุดทดสอบ จะแบ่งเป็น 2 การทดสอบ คือ การทดสอบรูปแบบการแสดงผล การทดสอบการจัดเก็บข้อมูล เพื่อที่จะหาว่า โปรแกรมมีความสมบูรณ์มากน้อยเพียงใด รูปแบบการแสดงผลเป็นอย่างไร การจัดเก็บข้อมูลถูกต้องหรือไม่

การทดลองที่ 1 ทดสอบรูปแบบการแสดงผล

การทดสอบรูปแบบการแสดงผลนี้ จะทดสอบเพื่อหาว่า โปรแกรม สามารถแสดงผลของข้อมูลที่มาจากมิเตอร์ได้ถูกต้องหรือไม่ หากการแสดงผลถูกต้อง หน้าจอของมิเตอร์ และการแสดงผลของโปรแกรม จะแสดงข้อมูลชุดเดียวกัน และค่าของข้อมูลตรงกัน

การทดลองที่ 2 ทดสอบการจัดเก็บข้อมูล

การทดสอบนี้ จะทดสอบเพื่อหาว่า เมื่อมีการสั่งให้โปรแกรมจัดเก็บข้อมูล โดยมีการตั้งระยะเวลาในการจัดเก็บหนึ่งข้อมูล และระยะเวลาในการบันทึกข้อมูลในหนึ่งไฟล์แล้ว โปรแกรมสามารถจัดเก็บข้อมูลได้ถูกต้องตามการตั้งค่าหรือไม่ และ ค่าของข้อมูลที่ถูกจัดเก็บ มีความคลาดเลื่อนมากน้อยเพียงใด

หากการจัดเก็บข้อมูลถูกต้อง ข้อมูลจะมีระยะห่างของเวลาตรงตามการตั้งค่า เช่น ตั้งค่าให้เก็บข้อมูลทุกๆ 5 นาที ข้อมูลแรกที่บันทึก จะห่างกับ ข้อมูลต่อไปที่บันทึก 5 นาทีตลอดการจัดเก็บ และ จำนวนของข้อมูล จะมีจำนวนเท่ากับ ระยะเวลาในการบันทึก หารด้วย ระยะเวลาในการจัดเก็บข้อมูลต่อหนึ่งไฟล์บันทึก เช่น ระยะเวลาในการจัดเก็บต่อหนึ่งข้อมูลเท่ากับ 5 นาที และ ระยะเวลาในการจัดเก็บข้อมูลต่อหนึ่งไฟล์มีค่าเท่ากับ 60 นาที จำนวนของข้อมูลต้องมีค่าเท่ากับ 60 หาร 5 เท่ากับ 12 ข้อมูล

บทที่ 4

การทดสอบและผลการทดสอบ

ในบทนี้จะเป็น การออกแบบทดลองต่อชุดทดสอบเข้ากับโปรแกรมที่ได้พัฒนาขึ้นมา เพื่อเก็บค่าและแสดงผล โดยการทดสอบจะแบ่งออกเป็น 3 รูปแบบ คือ รูปการแสดงผล รูปแบบของการเก็บข้อมูล และ ทดสอบการปรับตั้งค่าต่างๆของโปรแกรม รายละเอียดปลีกย่อยของการทดลองมีดังต่อไปนี้

4.1 การทดสอบการแสดงผลทางหน้าจอมอนิเตอร์

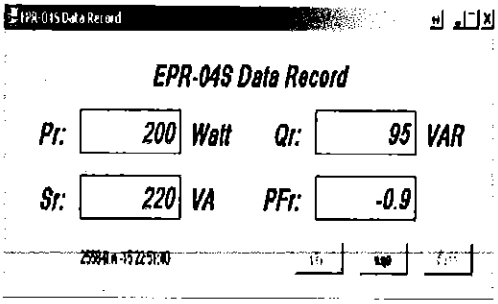
การทดสอบการแสดงผล จะทำโดยการต่อชุดทดสอบ เข้ากับมิเตอร์ และ ต่อสายสัญญาณเข้ากับโปรแกรม เพื่อทดสอบว่า โปรแกรมแสดงผลได้ตรงตามหน้าจอของตัวมิเตอร์หรือไม่ มีความคลาดเคลื่อนมากน้อยเพียงใด โดยที่ ค่าการแสดงผล จะนำมาแสดง 4 ค่าหลักๆ คือ

1. ค่าตัวประกอบกำลังทางไฟฟ้า Pf
2. ค่ากำลังไฟฟ้าจริง P
3. ค่ากำลังไฟฟ้าปรากฏ S
4. ค่ากำลังไฟฟ้าจินตภาพ Q

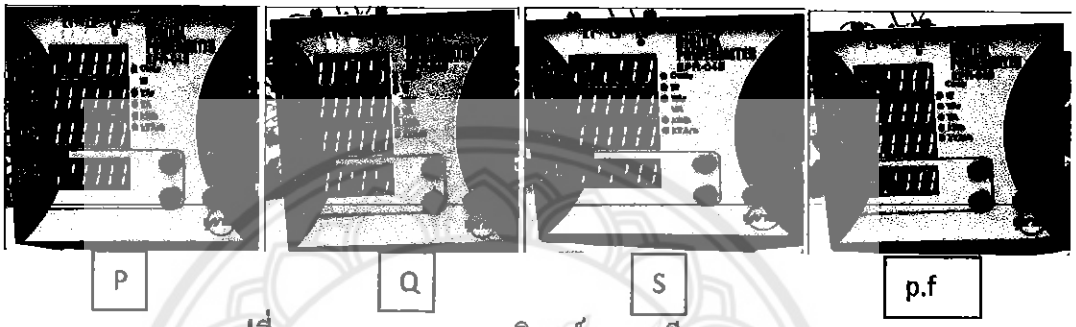
ผลการทดสอบ การแสดงผลของหน้าจอมิเตอร์ เทียบกับ การแสดงผลของโปรแกรม รูปที่ 4.1-4.2

ตารางที่ 4.1 ค่าพลังงาน ต่อ จำนวนภาระต่างๆ

จำนวนหลอดไฟ และ พัดลมตั้งโต๊ะ	Pf	P	S	Q
กรณี 1หลอด และ เปิดพัดลมเบอร์ 1	-0.82	70 W	95 VA	55 VAr
กรณี 2หลอด และ เปิดพัดลมเบอร์2	-0.9	150 W	170 VA	75 VAr
กรณี 3หลอด และ เปิดพัดลมเบอร์ 3	-0.95	190 W	205 VA	75 VAr
กรณีหลอดขนาด 100W 2หลอด	-0.90	200 W	220 VA	95 VAr



รูปที่ 4.1 การแสดงผลของโปรแกรมจากกรณี 100W 2หลอด

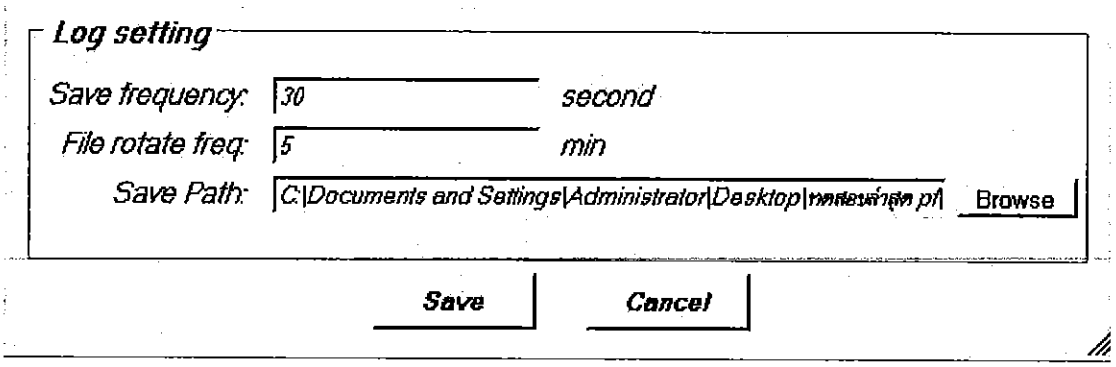


รูปที่ 4.2 การแสดงผลของมิเตอร์จากกรณี 100W 2หลอด

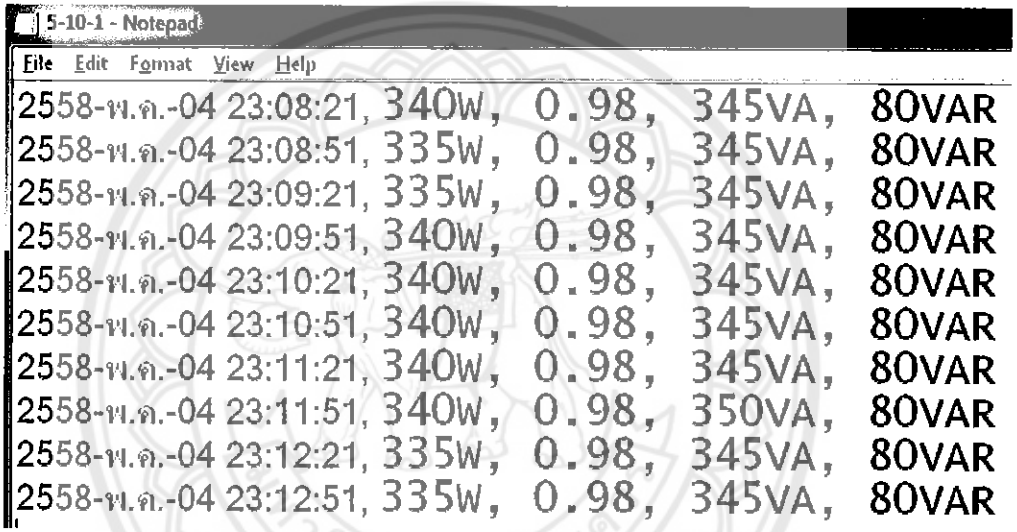
จากการทดสอบการแสดงผลของโปรแกรม จากรูปที่ 4.1 และ 4.2 พบว่า โปรแกรมสามารถที่จะแสดงผลออกมาทางหน้าจอได้อย่างชัดเจน และมีค่ากำลังไฟฟ้าแบบต่างๆ ถูกต้องตามหน้าจอของมิเตอร์ และไม่มีความคลาดเคลื่อนในการแสดงผล และโปรแกรม สามารถแสดงผลแบบติดตามได้ หากมีการเปลี่ยนแปลงของค่าพลังงานทันทีทันใด

4.2 การทดสอบการการจัดเก็บข้อมูล

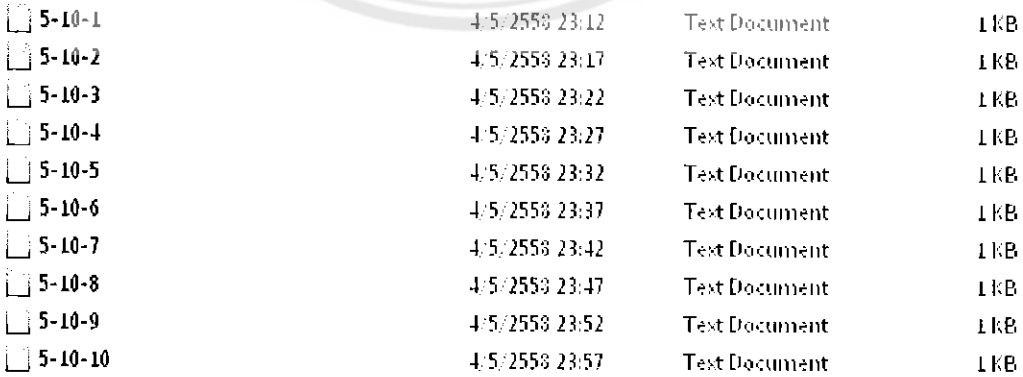
ในการทดสอบการบันทึกค่านั้น จะเป็นการทดสอบ เก็บค่าข้อมูล ที่เวลาต่างกัน เพื่อทดสอบความแม่นยำในการเก็บค่า ตามความห่างของเวลาที่ตั้งไว้ และ จะตั้งค่าโปรแกรม ให้สร้างไฟล์เก็บข้อมูลขึ้นมาใหม่ เมื่อเก็บค่าครบตามเวลาที่ตั้ง เช่น เก็บค่าทุกๆ 30 วินาที เมื่อเก็บครบ 5 นาที จะให้โปรแกรมขึ้นไฟล์บันทึกค่าใหม่ หรือเก็บค่าทุกๆ 1 นาที หากครบ 10 นาทีจะให้โปรแกรมขึ้นไฟล์บันทึกค่าใหม่ ที่ต้องขึ้นไฟล์ใหม่นั้น ก็เพราะจะเป็นการสะดวกต่อดูบันทึกข้อมูลย้อนหลัง และจำนวนข้อมูลจะได้ไม่มีมากจนเกินไป



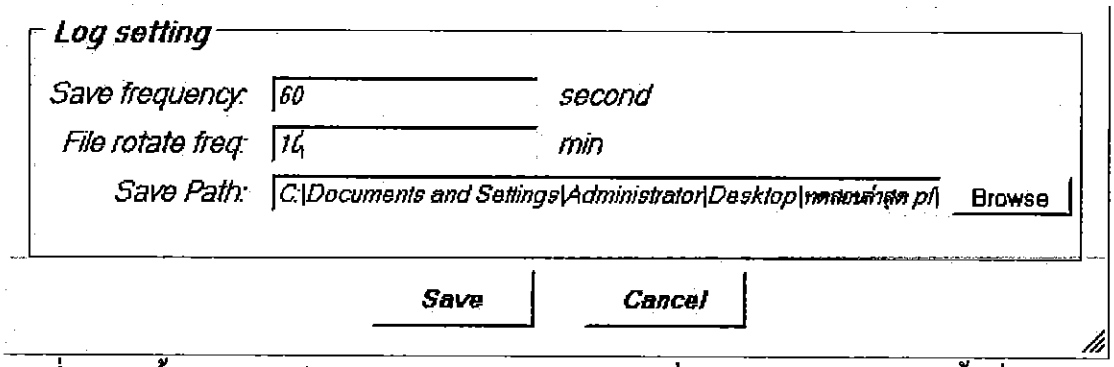
รูปที่ 4.3 การตั้งค่าการบันทึกค่าของโปรแกรมทุกๆ 30 วินาที เมื่อบันทึกค่าครบ 5 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



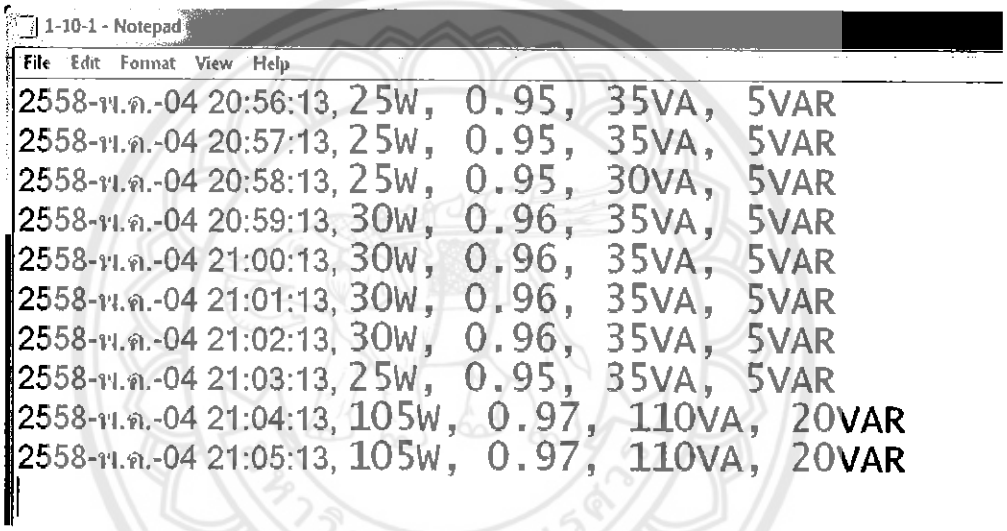
รูปที่ 4.4 การบันทึกค่าของโปรแกรมทุกๆ 30 วินาที เมื่อบันทึกค่าครบ 5 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



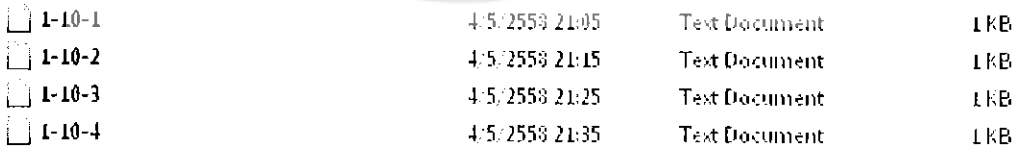
รูปที่ 4.5 การบันทึกค่าของโปรแกรมทุกๆ 30 วินาที เมื่อบันทึกครบ 5 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



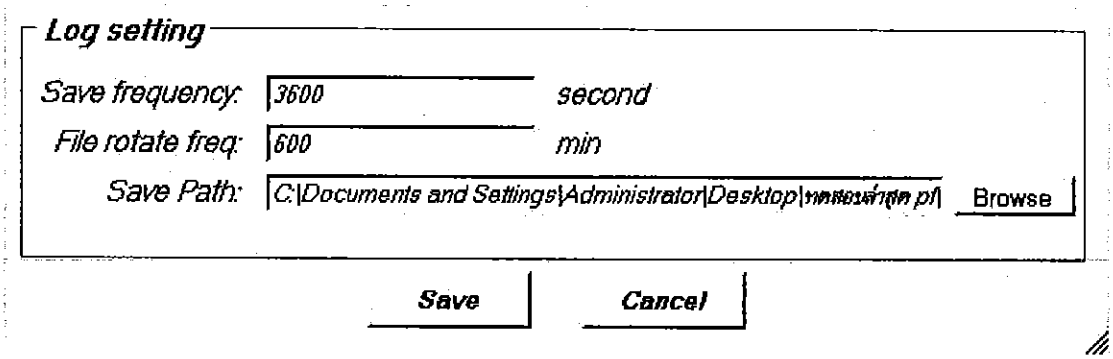
รูปที่4.6 การตั้งค่าการบันทึกค่าของโปรแกรมทุกๆ 1 นาที เมื่อบันทึกครบ 10 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



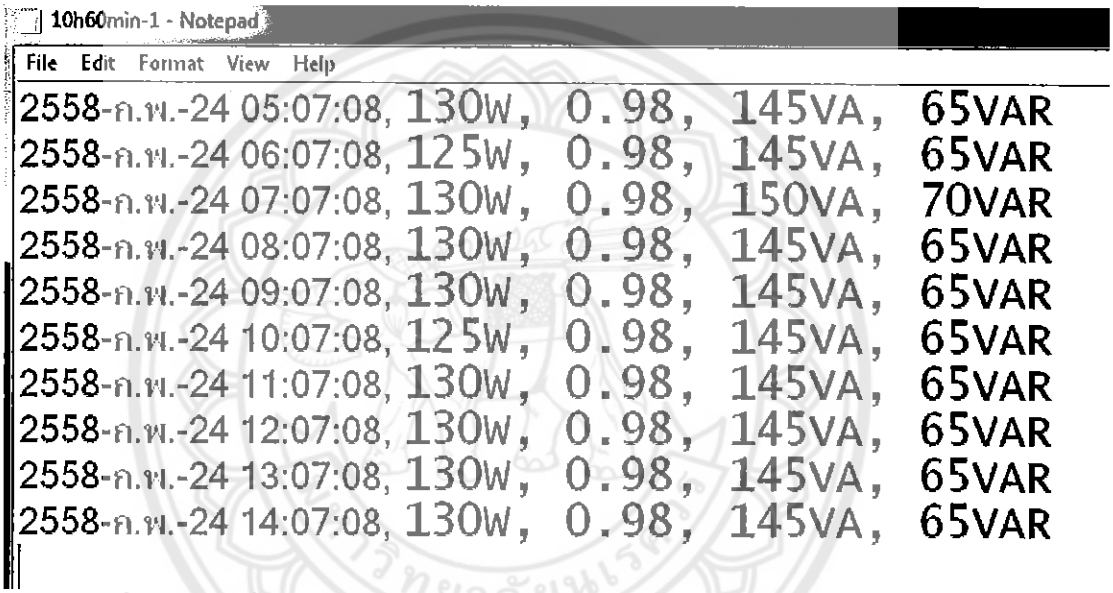
รูปที่ 4.7 การบันทึกค่าของโปรแกรมทุกๆ 1 นาที เมื่อบันทึกค่าครบ 10 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



รูปที่ 4.8 การบันทึกค่าของโปรแกรมทุกๆ 1 นาที เมื่อบันทึกค่าครบ 10 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



รูปที่ 4.9 การตั้งค่าการบันทึกค่าทุกๆ 1 ชั่วโมง เมื่อบันทึกครบ 10 ชั่วโมงให้ขึ้นที่เก็บไฟล์อันใหม่



รูปที่ 4.10 การตั้งค่าการบันทึกค่าทุกๆ 1 ชั่วโมง เมื่อบันทึกครบ 10 ชั่วโมงให้ขึ้นที่เก็บไฟล์อันใหม่

Name	Date modified	Type	Size
10h60min-1	24/2/2553 14:07	Text Document	1 KB
10h60min-2	25/2/2553 0:07	Text Document	1 KB
10h60min-3	25/2/2553 10:07	Text Document	1 KB
10h60min-4	25/2/2553 20:07	Text Document	1 KB
10h60min-5	26/2/2553 6:08	Text Document	1 KB
10h60min-6	26/2/2553 16:08	Text Document	1 KB
10h60min-7	27/2/2553 2:08	Text Document	1 KB
10h60min-8	27/2/2553 12:08	Text Document	1 KB
10h60min-9	27/2/2553 22:08	Text Document	1 KB
10h60min-10	28/2/2553 8:08	Text Document	1 KB
10h60min-11	28/2/2553 18:08	Text Document	1 KB
10h60min-12	1/3/2553 4:08	Text Document	1 KB
10h60min-13	1/3/2553 14:08	Text Document	1 KB
10h60min-14	2/3/2553 0:08	Text Document	1 KB
10h60min-15	2/3/2553 10:08	Text Document	1 KB
10h60min-16	2/3/2553 20:08	Text Document	1 KB
10h60min-17	3/3/2553 6:08	Text Document	1 KB
10h60min-18	3/3/2553 16:08	Text Document	1 KB
10h60min-19	4/3/2553 2:08	Text Document	1 KB

รูปที่ 4.11 การตั้งค่าการบันทึกค่าทุกๆ 1 ชั่วโมง เมื่อบันทึกครบ 10 ชั่วโมงให้ขึ้นที่เก็บไฟล์อันใหม่

ผลการทดสอบ จากรูปที่ 4.3-4.5 เป็นการทดสอบ การบันทึกค่าของข้อมูล เมื่อตั้งการบันทึกค่าทุกๆ 5 วินาที และเมื่อบันทึกครบ 5 นาที ให้ขึ้นที่เก็บไฟล์อันใหม่ พบว่า เมื่อตั้งความละเอียดของเวลาในการบันทึกค่าลงไป ในระดับวินาที โปรแกรม สามารถบันทึกตามเวลาที่ตั้งไว้ได้อย่างถูกต้อง และ ไม่มีความคลาดเคลื่อนเกิดขึ้น

รูปที่ 4.6-4.8 เป็นการทดสอบการบันทึกค่าของข้อมูล เมื่อมีการตั้งการบันทึกค่าทุกๆ 1 วินาที และเมื่อบันทึกครบ 10 นาที ให้ขึ้นที่เก็บไฟล์อันใหม่ พบว่า การบันทึกในระดับนาที โปรแกรมยังมีความสามารถในการบันทึกได้ดี ที่เวลาในระดับนาที และ ไม่มีความคลาดเคลื่อนเกิดขึ้น

รูปที่ 4.9-4.11 เป็นการทดสอบการบันทึกค่าของข้อมูล เมื่อตั้งการบันทึกค่าทุกๆ 1 ชั่วโมง และเมื่อบันทึกครบ 10 ชั่วโมง ให้ขึ้นที่เก็บไฟล์อันใหม่ พบว่า แม้จะมีการตั้งระยะเวลาในการบันทึกที่นานขึ้นเป็นระดับ ชั่วโมง โปรแกรมก็ยังสามารถที่จะบันทึกค่าได้อย่างถูกต้อง และมีความคลาดเคลื่อนของเวลาในการบันทึกในระดับ นาที และในการทดลองนี้เป็นการทดลองที่นานถึง 9 วัน จะเห็นได้ว่าโปรแกรมยังสามารถทำงานต่อเนื่องได้ แม้จะมีความคลาดเคลื่อนในส่วนของเวลาเกิดขึ้น แต่เนื่องจากระยะเวลาที่จำกัด จึงทำการทดลองสูงสุดไว้ที่ 9 วัน เนื่องจากปัญหาความร้อนที่อาจทำให้เกิดอันตรายต่อคอมพิวเตอร์ และ ห้องทดลอง

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลองวัดค่าองค์ประกอบทางไฟฟ้าต่างๆ ของภาระที่ต่อเข้ากับระบบ ประเภทเครื่องใช้ไฟฟ้าในครัวเรือน โดยต่อผ่านมิเตอร์ไฟฟ้าแบบดิจิตอล รุ่น EPR-04s และทำการบันทึกค่าและแสดงผล ผ่านโปรแกรมที่ได้พัฒนาขึ้นมา เมื่อพิจารณาผลการทดลองพบว่า

ค่าพลังงานไฟฟ้าจากผลการทดลอง มีค่าแตกต่างกันออกไป โดยขึ้นอยู่กับภาระต่างๆที่นำมาเชื่อมต่อเข้ากับระบบ ซึ่งค่าพลังงานไฟฟ้าที่โปรแกรมวัดได้นั้น มีค่าใกล้เคียงกับหน้าจอของตัวมิเตอร์ แม้มีการเปลี่ยนแปลงของภาระแบบกะทันหัน การแสดงผลของโปรแกรมยังสามารถแสดงผลแบบติดตามได้อย่างต่อเนื่อง หากมีการตั้งค่าการบันทึกผลแบบละเอียดในระดับวินาที ก็จะสามารถบันทึกผลแบบติดตามได้

ค่ากำลังไฟฟ้าจริง ที่วัดได้จากการทดลอง จะมีปริมาณไม่เท่ากันในทุกการทดลอง เนื่องจากมีการเปลี่ยนค่าของภาระที่นำมาต่อให้มากขึ้นหรือลดลง และ ภาระชนิดหลอดไฟมีการขาดของไส้หลอดเกิดขึ้นแต่ก็ไม่เกิดปัญหากับการแสดงผล และบันทึกผลของโปรแกรม และในค่ากำลังไฟฟ้าปรากฏ กับค่ากำลังไฟฟ้าจินตภาพก็เช่นกัน มีการเปลี่ยนแปลงอยู่ตลอดเวลา และค่าที่เปลี่ยนแปลงตามมาก็จะเป็น ค่าตัวประกอบกำลังทางไฟฟ้า

ค่าตัวประกอบกำลังทางไฟฟ้า โดยส่วนมาก จะมีค่าเป็น ล้าหลัง (Lagging) เนื่องจากมีภาระชนิดขดลวดเหนี่ยวนำ และภาระที่นำมาต่อ มีพัดลม ซึ่งใช้มอเตอร์ในการขับเคลื่อนใบพัด และมีตัวเก็บประจุช่วยในการสตาร์ท ทำให้ค่า ตัวประกอบกำลังทางไฟฟ้าไม่คงที่ ในการทดลอง หากมีการต่อหม้อแปลงกระแสผิดทิศทาง จะทำให้เกิดความคลาดเคลื่อนอย่างมากกับตัวประกอบกำลังได้ เพราะฟลักซ์ของแม่เหล็กไฟฟ้า ที่ตัดกับขดลวดภายในของ หม้อแปลงกระแสจะกลับทิศ ทำให้ได้ค่าตัวประกอบกำลังไฟฟ้าในบางครั้งเป็นค่า Leading และจะทำให้เครื่องหมายของตัวกระกอบกำลังเปลี่ยนไปจาก – เป็น + ทั้งในการบันทึก และแสดงผล แม้กระทั่งการแสดงผลของหน้าจอมิเตอร์

5.2 ประเมินผล

จากการดำเนินงานเทียบกับวัตถุประสงค์ได้ผลดังนี้

1. สามารถพัฒนาโปรแกรมสำหรับอ่านค่าและบันทึกผลแบบติดตามสำหรับมิเตอร์ไฟฟ้าแบบดิจิตอลรุ่น EPR-04s ได้ โดยมีการแสดงค่า กำลังไฟฟ้าจริง กำลังไฟฟ้าปรากฏ กำลังไฟฟ้าจินตภาพ และค่าตัวประกอบกำลังทางไฟฟ้า

2. มีความรู้ความเข้าใจเกี่ยวกับระบบสื่อสาร การสร้างโปรโตคอล การถอดรหัสโปรโตคอลออกมา เพื่อทำการติดต่อสื่อสาร กับติมิเตอร์ไฟฟ้าแบบดิจิตอลได้

3. สามารถใช้องค์ความรู้ที่ได้จากในห้องเรียน ในการฝึกงาน มาใช้งานได้จริง ในการเชื่อมต่อระบบสื่อสารเข้าด้วยกัน

5.3 ปัญหา ข้อเสนอแนะ และแนวทางแก้ไข

1. ปัญหาจากการต่อระบบ เข้ากับภาระที่กินกระแสน้อยเกินไป ทำให้ในตอนแรกไม่สามารถวัดค่าพลังงานไฟฟ้าได้ เพราะขนาดของหม้อแปลงกระแสมีอัตราทดที่มากเกินไป จึงต้องมีการเปลี่ยนขนาดหม้อแปลงกระแส ให้มีขนาดเหมาะสมกับปริมาณค่ากระแสของภาระ และ ถ้าหากเปลี่ยนหม้อแปลงกระแสแล้วค่ากระแสยังไม่พอ ให้ เพิ่มอัตราทดของหม้อแปลง โดยการพันเส้นลวดตัวนำเข้ากับด้านแรงสูงของหม้อแปลงให้มากกว่า 1 รอบ เพื่อทำให้มีกระแสไหลผ่านหม้อแปลงมากขึ้น

2. ปัญหาจากการเชื่อมต่อระบบสื่อสารที่ผิดพลาด ทำให้อุปกรณ์บางส่วนได้รับความเสียหาย จึงต้องทำการซ่อมแซม ทำให้ชุดทดสอบที่ออกมา ไม่มีเรียบร้อยเท่าที่ควรจะเป็น และมีอันตรายจากกระแสไฟฟ้าอยู่มาก โดยเฉพาะจุดเชื่อมต่อ จึงต้องทำการซื้ออุปกรณ์ฉนวนมาใส่ชุดเชื่อมต่อเพื่อความปลอดภัย

3. ปัญหาในการถอดโปรโตคอลของตัวมิเตอร์ การถอดโปรโตคอลจำเป็นต้องใช้โปรแกรมช่วย แต่เนื่องจากมีลิขสิทธิ์ ของตัวโปรแกรมถอดโปรโตคอล จึงต้องทำการติดตั้งใหม่หลายครั้ง จึงได้แก้ไขโดยการใช้ Free soft ware เข้ามาช่วย

4. สายสัญญาณที่ใช้ในการเชื่อมต่อระบบสื่อสาร มีความเปราะบางมาก ในการเชื่อมต่อ หรือเคลื่อนย้ายอุปกรณ์ทุกครั้ง ต้องระมัดระวังเป็นพิเศษ

5.4 แนวทางในการพัฒนาต่อไป

ผลที่ได้จากการทดลองในโครงการนี้ สามารถนำไปประยุกต์ใช้ในการศึกษาและเป็นแนวทางในการสร้างโปรโตคอล เพื่อ อ่านค่าจากมิเตอร์ไฟฟ้าชนิดอื่นๆ และสามารถประยุกต์จากการวัดผลแบบ 1 เฟส ไปเป็น 3 เฟส ได้ เพื่อใช้ในทางอุตสาหกรรมต่อไป อีกทั้งยังสามารถพัฒนาต่อเพื่อไปใช้กับคอมพิวเตอร์ขนาดเล็กเช่น Raspberry pi เพื่อใช้เป็น Data logger นำไปติดตั้งกับอุปกรณ์ไฟฟ้าขนาดใหญ่ ที่ต้องการวัดค่าและเก็บข้อมูลได้ สามารถนำไปประยุกต์ทำเป็น

แอปพลิเคชัน นำไปใช้ใน สมาร์ทโฟน เพื่อที่จะดูการเปลี่ยนแปลง หรือทำการแจ้งเตือน หากเกิดปัญหาจากค่าพลังงานไฟฟ้า โดยผู้ใช้สามารถควบคุมจากระยะไกลได้

รูปแบบของภาษา Visual basic เป็นภาษาที่ค่อนข้างใช้งานง่ายก็จริง แต่ยังไม่ใช่ ภาษา ที่เป็น Free license ภาษานี้ยังเป็นลิขสิทธิ์ของบริษัท ไมโครซอฟต์ ซึ่งไม่สามารถนำไปใช้ในเชิงธุรกิจได้ ดังนั้นจึงไม่สมควรนำไปค้าขายเชิงธุรกิจ ถ้าหากจะนำไปใช้ในเชิงธุรกิจ ผู้เขียน แนะนำให้พัฒนาต่อ โดยใช้ภาษา Phython เป็นพื้นฐานในการพัฒนา

เอกสารอ้างอิง

[1] http://isareeya.com (2555) โดยคุณ ISAREEYA KEATWUTTIKAN เรื่อง โปรแกรม Visual Basic คืออะไร (Visual Basic language) [ออนไลน์] เข้าถึงได้จาก http://goo.gl/TjrAbl

[2] http://visualgdb.com/ (2558) โดยคุณ sysprogs เรื่อง Developing a Raspberry PI app with Visual Studio [ออนไลน์] เข้าถึงได้จาก http://visualgdb.com/tutorials/raspberry/

[3] http://www.entes.com.tr/ (2558) โดยบริษัท ENTES เรื่อง EPR-04s [ออนไลน์] เข้าถึงได้จาก http://www.entes.com.tr/enerji_olcer_ud_en.asp?livecatID=2&livecataltID=18&urunID=12

[4] http://www.tortech.com.au โดยบริษัท Tortech เรื่อง Split Core Current Transformer DP-23 100/5A [ออนไลน์] เข้าถึงได้จาก http://www.tortech.com.au/toroidal-transformers/split-core-ct-core-current-transformers

[5] http://www.thailandindustry.com (2552) โดยบริษัท Thailand industry เรื่อง การสื่อสารข้อมูลในอุตสาหกรรม[ออนไลน์] เข้าถึงได้จาก http://www.thailandindustry.com/

[6] www.riverplus.com (2556) ,โดยเวปมาสเตอร์ riverplus เรื่อง การสื่อสารแบบModbus Protocol[ออนไลน์] เข้าถึงได้จาก http://riverplusblog.com/2011/08/18/

[7] http://www.rtafshooting.com, (2553) โดย กองทัพอากาศ เรื่อง PLC คืออะไร[ออนไลน์] เข้าถึงได้จาก http://goo.gl/jny90j

[8] http://www.schneider-electric.co.th (2558)โดยบริษัท schneider-electric เรื่อง schneider-electric website [ออนไลน์] เข้าถึงได้จาก http://www.schneider-electric.co.th

[9] http://www.eda.co.th/scada.html, (2552) โดยบริษัท EDA เรื่อง SCADA คืออะไร[ออนไลน์] เข้าถึงได้จาก http://www.eda.co.th/scada.html

[10] http://www.lammertbies.nl (2557) โดยคุณ Lammert bies เรื่อง CRC Calculation [ออนไลน์] เข้าถึงได้จาก http://www.lammertbies.nl/comm/info/crc-calculation.html

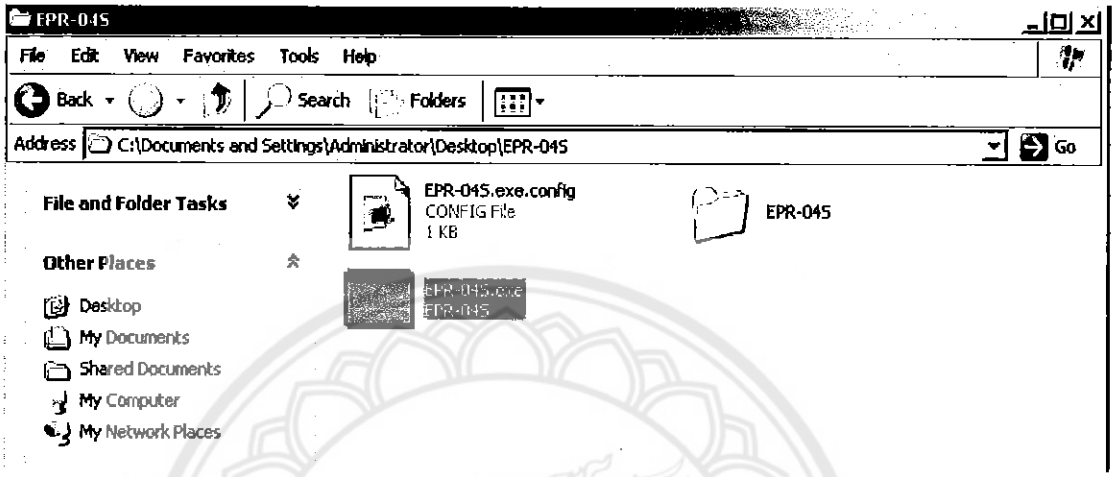
[11] http://free-serial-port-monitor.com (2558) โดย เวปมาสเตอร์ Win7dwnld เรื่อง free serial port monitor [ออนไลน์] เข้าถึงได้จาก http://free-serial-port-monitor.win7dwnld.com/

ภาคผนวก ก.

วิธีการใช้งานโปรแกรมเบื้องต้น

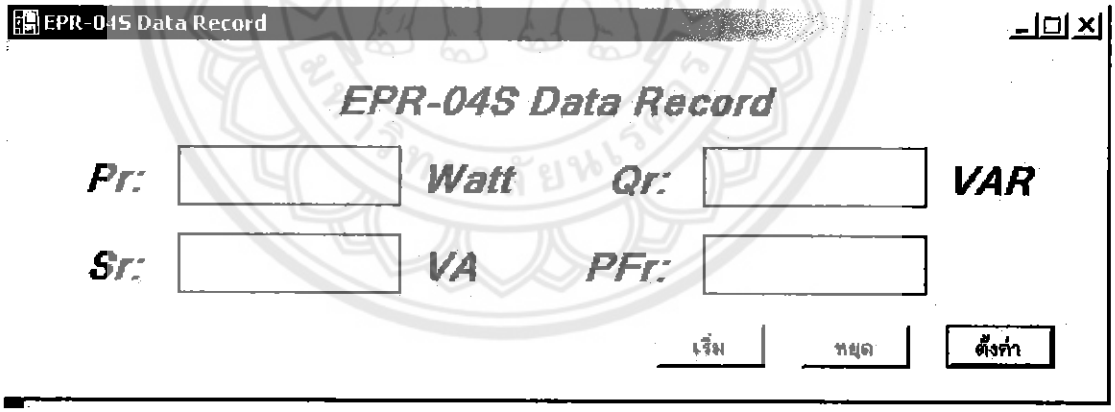
การใช้งานโปรแกรมเบื้องต้น

1. ในแฟ้มจะประกอบไปด้วย ไฟล์เก็บค่า Setting ของโปรแกรม และ แฟ้มเก็บโค้ดของโปรแกรม ให้ คลิกเข้าไปที่ไอคอน EPR-04S.exe เพื่อเข้าสู่โปรแกรม



รูปที่ ก.1 โปรแกรม EPR-04S

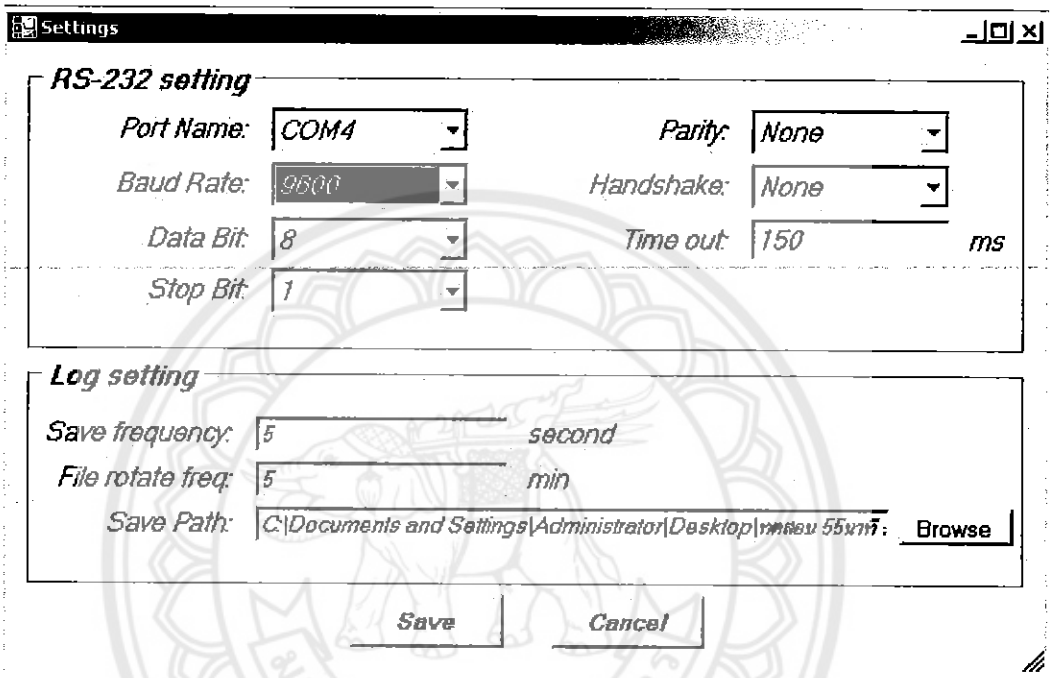
2. เมื่อเข้ามาแล้ว จะมีหน้าต่างโปรแกรมเด้งขึ้นมา ให้กดปุ่มตั้งค่า เพื่อตั้งค่าเริ่มต้นใช้งาน



รูปที่ ก.2 หน้าจอการแสดงผลเมื่อเริ่มใช้งาน

3. เมื่อกดการตั้งค่า ก็จะปรากฏหน้าต่างขึ้นมา ให้สามารถตั้งค่าได้

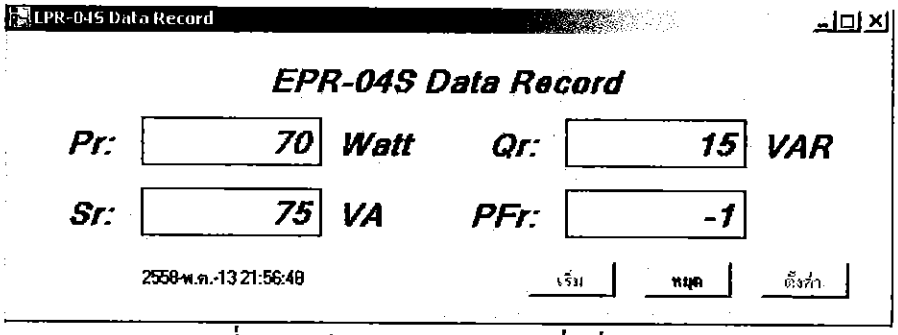
ในส่วนของ RS-232 Setting ถ้าหากไม่ทราบการตั้งค่า ให้ตั้งค่าต่างๆตามรูปภาพที่ ก.3 ในส่วนของ Log setting ผู้ใช้งานสามารถตั้งค่าได้ตามที่ต้องการ แล้วกด Save หลังจากนั้น โปรแกรมจะกลับมาที่หน้าจอเริ่มต้นการใช้งาน ส่วนการดูไฟล์บันทึกค่า สามารถเข้าไป Save Path ที่ตั้งไว้ได้เลย



รูปที่ ก.3 หน้าต่างตั้งค่าของโปรแกรม

4. เมื่อตั้งค่าเสร็จแล้วให้กดปุ่มเริ่ม โปรแกรมก็จะทำงาน และแสดงผลออกมา หากต้องการที่จะเปลี่ยนการตั้งค่าใหม่ หรือเปลี่ยนที่บันทึกไฟล์ ให้กดปุ่มหยุด แล้วกด ปุ่มตั้งค่า แล้วหน้าต่างการตั้งค่าจะเด้งขึ้นมา หากไม่ต้องการเปลี่ยนการตั้งค่าแล้ว สามารถกด Cancel ได้ เพื่อยกเลิก โปรแกรมก็จะกลับไปที่หน้าจอเริ่มต้น

หากต้องการปิดโปรแกรมไปเลย ก็สามารถกดปุ่มหยุด และกด กากบาท ออกไปได้



รูปที่ ก.4 หน้าจอของโปรแกรม เมื่อเริ่มทำงาน

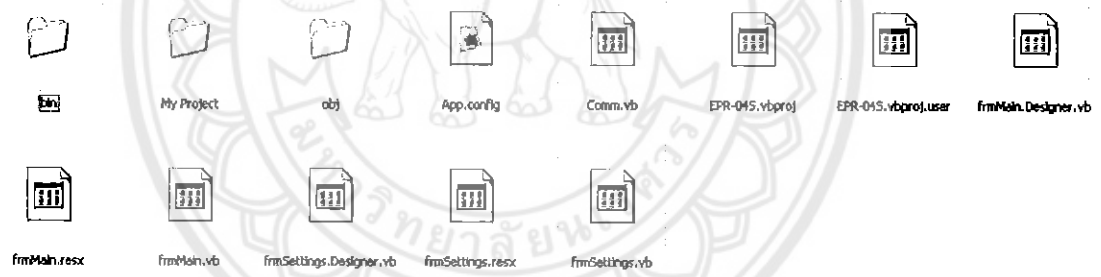
5. ภายในแฟ้ม EPR-04s จะประกอบไปด้วยชุดโค้ดคำสั่งของโปรแกรม สามารถใช้โปรแกรม Notepad++ เปิดดูโค้ดได้ ชุดโค้ดจะมีนามสกุลของไฟล์ทั้ง

.vb คือโค้ดหลักในการทำงานของโปรแกรมโดยที่ Frmsetting.vb คือโค้ดส่วนรับค่าในการตั้งค่าสื่อสารและตั้งสถานที่เก็บไฟล์ Comm.vb คือ โค้ดการดึงค่ามาจากหน้าต่าง Setting และ Frmmian.vb คือ โค้ดหลักในส่วนการทำงานของโปรแกรม

.resx คือโค้ดอ้างอิงคำสั่งที่ใช้ในการทำงานของโปรแกรมโดยที่ Frmsetting.resx คือ ที่เก็บคำสั่งอ้างอิงของส่วนรับค่าในการตั้งค่าสื่อสารและตั้งสถานที่เก็บไฟล์ และ Frmmain.resx คือ ที่เก็บคำสั่งอ้างอิงของส่วนการทำงานของการทำงานหลักของโปรแกรม

.Designer.vb คือ โค้ดที่เก็บการออกแบบหน้าตาของโปรแกรมโดยที่ Frmsetting.Designer คือการออกแบบของหน้าตาของ หน้าต่าง setting และ Frmmain.Designer คือการออกแบบหน้าตาการแสดงผลของโปรแกรม

ส่วนอันที่เหลือ เป็นโครงสร้างของโปรแกรมที่ ถูกกำหนดขึ้นมาตอน Run คำสั่งเพื่อสร้างโปรแกรม EPR-04s.exe หากจะนำไปใช้งาน สามารถลอกสำเนาโปรแกรม EPR-04s.exe ออกไปได้เลย โดยไม่จำเป็นต้องสำเนาส่วนที่เหลือไป



รูปที่ ก.5 ส่วนประกอบของโปรแกรม

ภาคผนวก ข.

รายละเอียดโค้ดคำสั่งในโปรแกรม

ข.1 โค้ด ส่วนรับค่า ในการตั้งค่าการสื่อสาร และ ตั้งสร้างสถานที่เก็บไฟล์ (อยู่ใน Frmsetting.vb)

ในส่วนนี้ จะเป็นการเซ็ทค่าตามที่ผู้ใช้ต้องการ ให้กับช่องรับค่า ต่างๆ เพื่อที่จะนำค่านี้ ไปตั้งรูปแบบการเชื่อมต่อของ พอร์ทสื่อสาร และสถานที่บันทึกไฟล์ และ เมื่อตั้งค่าเสร็จ จะเก็บค่า Config ลงใน Texfile เพื่อรอนำไปใช้เมื่อเริ่มโปรแกรม ดังรูป 3.6 ในบทที่ 3

```
Imports System.IO.Ports

Public Class frmSettings
    Private Sub frmTest_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' เซ็ตค่าให้กับ ComboBox ต่างๆ
        Me.cmbPortName.Items.Clear()
        Me.cmbBuadrate.Items.Clear()
        Me.cmbDataBit.Items.Clear()
        Me.cmbStopBit.Items.Clear()
        Me.cmbParity.Items.Clear()
        Me.cmbHandshake.Items.Clear()

        Me.cmbPortName.Items.AddRange(SerialPort.GetPortNames())
        Me.cmbBuadrate.Items.AddRange(New String() {2400, 4800, 9600, 19200, 38400})
        Me.cmbDataBit.Items.AddRange(New String() {7, 8, 9})
        Me.cmbStopBit.Items.AddRange(New String() {0, 1, 1.5, 2})
        Me.cmbParity.Items.AddRange(New String() {"None", "Odd", "Event"})
        Me.cmbHandshake.Items.AddRange(New String() {"None", "XonXoff", "RTS",
"RTSXonXoff"})

        ' ดึงค่าต่างๆจากไฟล์ config มาใส่ใน ComboBox และ TextBox
        Me.cmbPortName.SelectedIndex = Comm.getConfig("PORT_NAME")
        Me.cmbBuadrate.SelectedIndex = Comm.getConfig("BAUD_RATE")
```

```
        Me.cmbDataBit.SelectedIndex = Comm.getConfig("DATA_BITS")
        Me.cmbStopBit.SelectedIndex = Comm.getConfig("STOP_BITS")
        Me.cmbParity.SelectedIndex = Comm.getConfig("PARITY")
        Me.cmbHandshake.SelectedIndex = Comm.getConfig("HANDSHAKE")
        Me.txtTimeout.Text = Comm.getConfig("TIME_OUT")
        Me.txtSaveFreq.Text = Comm.getConfig("SAVE_FREQ")
        Me.txtLogRotateFreq.Text = Comm.getConfig("LOG_ROTATE_FREQ")
        Me.txtSavePath.Text = Comm.getConfig("SAVE_PATH")

    End Sub

    Private Sub Button3_Click(ByVal sender As Object, ByVal e As EventArgs) Handles
btnCancel.Click
        ' ปิดหน้าต่าง
        Me.Close()
    End Sub

    Private Sub btnBrowse_Click(ByVal sender As Object, ByVal e As EventArgs) Handles
btnBrowse.Click
        Dim saveFileDialog As New SaveFileDialog()

        ' สร้าง file dialog เพื่อให้ user เลือกไฟล์ที่จะบันทึกค่า
        saveFileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
        saveFileDialog.FilterIndex = 1
        saveFileDialog.RestoreDirectory = True

        If saveFileDialog.ShowDialog() = DialogResult.OK Then
            Me.txtSavePath.Text = saveFileDialog.FileName
        End If
    End Sub
```

```
Private Sub btnSave_Click(ByVal sender As Object, ByVal e As EventArgs) Handles
btnSave.Click
    ' เช็คว่าค่าต่างๆที่กรอกถูกต้องหรือไม่
    If Me.cmbPortName.Text = "" Then
        MsgBox("Please select port name!", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If

    If Me.cmbBuadrate.Text = "" Then
        MsgBox("Please select baud rate!", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If

    If Me.cmbDataBit.Text = "" Then
        MsgBox("Please select data bit!", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If

    If Me.cmbStopBit.Text = "" Then
        MsgBox("Please select stop bit", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If

    If Me.cmbParity.Text = "" Then
        MsgBox("Please select parity!", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If

    If Me.cmbHandshake.Text = "" Then
        MsgBox("Please select handshake!", MsgBoxStyle.Critical, "Error")
        Exit Sub
    End If
```

```
If Me.txtTimeout.Text = "" Then
   MsgBox("Please input rs-232 read timeout!", MsgBoxStyle.Critical, "Error")
   Exit Sub
End If

If Me.txtSaveFreq.Text = "" Then
   MsgBox("Please input save frequency!", MsgBoxStyle.Critical, "Error")
   Exit Sub
End If

If Me.txtLogRotateFreq.Text = "" Then
   MsgBox("Please input log rotate frequency!", MsgBoxStyle.Critical, "Error")
   Exit Sub
End If

If Me.txtSavePath.Text = "" Then
   MsgBox("Please input path!", MsgBoxStyle.Critical, "Error")
   Exit Sub
End If

' เขียนค่า Config ต่างๆลงไฟล์ config
Comm.putConfig("PORT_NAME", Me.cmbPortName.SelectedIndex)
Comm.putConfig("BAUD_RATE", Me.cmbBuadrate.SelectedIndex)
Comm.putConfig("DATA_BITS", Me.cmbDataBit.SelectedIndex)
Comm.putConfig("STOP_BITS", Me.cmbStopBit.SelectedIndex)
Comm.putConfig("PARITY", Me.cmbParity.SelectedIndex)
Comm.putConfig("HANDSHAKE", Me.cmbHandshake.SelectedIndex)
Comm.putConfig("TIME_OUT", Me.txtTimeout.Text)
Comm.putConfig("LOG_ROTATE_FREQ", Me.txtLogRotateFreq.Text)
Comm.putConfig("SAVE_FREQ", Me.txtSaveFreq.Text)
Comm.putConfig("SAVE_PATH", Me.txtSavePath.Text)
```

```
    ' กำหนดค่าคอนฟิคใหม่ให้กับตัวแปร
    Comm.portName = Comm.portNameArr(Comm.getConfig("PORT_NAME"))
    Comm.baudRate = Comm.baudrateArr(Comm.getConfig("BAUD_RATE"))
    Comm.dataBits = Comm.dataBitsArr(Comm.getConfig("DATA_BITS"))
    Comm.stopBits = Comm.stopBitsArr(Comm.getConfig("STOP_BITS"))
    Comm.parity = Comm.parityArr(Comm.getConfig("PARITY"))
    Comm.handshake = Comm.handshakeArr(Comm.getConfig("HANDSHAKE"))
    Comm.readTimeout = Comm.getConfig("TIME_OUT")
    Comm.logRotateFreq = Comm.getConfig("LOG_ROTATE_FREQ")
    Comm.saveFreq = Comm.getConfig("SAVE_FREQ")
    Comm.savePath = Comm.getConfig("SAVE_PATH")

    ' ปิดหน้าต่าง
    Me.Close()
  End Sub
End Class
```

ข.2 โค้ดการดึงค่ามาจากหน้าต่าง Setting (อยู่ใน Comm.vb)

จากที่ผู้ใช้ ทำการตั้งค่าในหน้า Setting ค่าที่ตั้งไว้ จะถูกบันทึกลงใน Texfile และ โค้ดในส่วนนี้ จะดึงค่ามาใช้ เพื่อตั้งตั้งค่าพอร์ทการสื่อสารตามที่ผู้ใช้กำหนดในหน้าต่าง Setting เมื่อทำเสร็จ จะบันทึกค่า Config นี้ไว้ เพื่อใช้ในการวน loop คำสั่ง ในการส่ง คำสั่ง จะได้ไม่ต้องมาดึงค่า Config จาก texfile ใหม่ ดังรูปที่ 1 ในส่วนภาคผนวก ก.

```
Imports System.Configuration
Imports System.IO.Ports

Public Class Comm

    Public Shared portNameArr As String() = SerialPort.GetPortNames()
    Public Shared baudrateArr = New String() {2400, 4800, 9600, 19200, 38400}
    Public Shared dataBitsArr = New String() {7, 8, 9}
    Public Shared stopBitsArr = New StopBits() {stopBits.None, stopBits.One,
stopBits.OnePointFive, stopBits.Two}
    Public Shared parityArr = New Parity() {parity.None, parity.Odd, parity.Even}
    Public Shared handshakeArr = New Handshake() {handshake.None, handshake.XOnXOff,
handshake.RequestToSend, handshake.RequestToSendXOnXOff}

    Public Shared portName As String
    Public Shared baudRate As Integer
    Public Shared dataBits As Integer
    Public Shared stopBits As StopBits
    Public Shared parity As Parity
    Public Shared handshake As Handshake
```

```
Public Shared readTimeout As Integer
Public Shared logRotateFreq As Integer
Public Shared saveFreq As Integer
Public Shared savePath As String

Public Shared Sub ReadAllConfig()
    ' อ่านค่าคอนฟิคต่างๆ ถ้าไม่มีค่าที่ตั้งไว้กำหนดค่าเริ่มต้นให้ค่าคอนฟิคโดยอัตโนมัติ
    If Comm.getConfig("PORT_NAME") = "" Then Comm.putConfig("PORT_NAME", "0")
    portName = portNameArr(Convert.ToInt32(Comm.getConfig("PORT_NAME")))

    If Comm.getConfig("BAUD_RATE") = "" Then Comm.putConfig("BAUD_RATE", "2")
    baudRate = baudrateArr(Convert.ToInt32(Comm.getConfig("BAUD_RATE")))

    If Comm.getConfig("DATA_BITS") = "" Then Comm.putConfig("DATA_BITS", "1")
    dataBits = dataBitsArr(Convert.ToInt32(Comm.getConfig("DATA_BITS")))

    If Comm.getConfig("STOP_BITS") = "" Then Comm.putConfig("STOP_BITS", "1")
    stopBits = stopBitsArr(Convert.ToInt32(Comm.getConfig("STOP_BITS")))

    If Comm.getConfig("PARITY") = "" Then Comm.putConfig("PARITY", "0")
    parity = parityArr(Convert.ToInt32(Comm.getConfig("PARITY")))

    If Comm.getConfig("HANDSHAKE") = "" Then Comm.putConfig("HANDSHAKE", "0")
    handshake = handshakeArr(Convert.ToInt32(Comm.getConfig("HANDSHAKE")))

    If Comm.getConfig("TIME_OUT") = "" Then Comm.putConfig("TIME_OUT", "500")
    readTimeout = Comm.getConfig("TIME_OUT")

    If Comm.getConfig("SAVE_FREQ") = "" Then Comm.putConfig("SAVE_FREQ", "30")
    saveFreq = Convert.ToInt32(Comm.getConfig("SAVE_FREQ"))
```

```
        If Comm.getConfig("LOG_ROTATE_FREQ") = "" Then
Comm.putConfig("LOG_ROTATE_FREQ", "60")
        logRotateFreq = Convert.ToInt32(Comm.getConfig("LOG_ROTATE_FREQ"))

        If Comm.getConfig("SAVE_PATH") = "" Then Comm.putConfig("SAVE_PATH",
System.IO.Path.GetDirectoryName(Application.ExecutablePath) & "\EPR-04S-Log.txt")
        savePath = Comm.getConfig("SAVE_PATH")
    End Sub

    Public Shared Function getConfig(key As String) As String
        Dim result As String

        Dim config As Configuration =
ConfigurationManager.OpenExeConfiguration(System.Windows.Forms.Application.ExecutableP
ath)
        If Not config.AppSettings.Settings(key) Is Nothing Then
            result = config.AppSettings.Settings(key).Value
        Else
            putConfig(key, "")
            result = ""
        End If
        getConfig = result
    End Function

    Public Shared Function putConfig(key As String, value As String) As Boolean
        Dim config As Configuration =
ConfigurationManager.OpenExeConfiguration(System.Windows.Forms.Application.ExecutableP
ath)
        Try
            ' เช็คว่ามีค่าคอนฟิคอยู่หรือไม่ ถ้ามีให้อัพเดต แต่ถ้าไม่มีให้เขียนเข้าไปใหม่
            If config.AppSettings.Settings(key) Is Nothing Then
                config.AppSettings.Settings.Add(key, value)
```

```
            Else
                config.AppSettings.Settings(key).Value = value
            End If

            ' บันทึกค่าคอนฟิค
            config.Save()
            config = Nothing
            putConfig = True
        Catch
            putConfig = False
        End Try
    End Function
End Class
```

ข.3 โค้ด หลัก ในส่วนการทำงานของโปรแกรม (อยู่ใน Frmmain.vb)

การทำงานของโปรแกรมจะมีหลายแบบ ในชุดคำสั่งส่วนเดียวกันแบบ คือ มีการตั้งค่าให้พอร์ทสื่อสารของคอมพิวเตอร์ มีการส่งคำสั่งออกไปจากคอมพิวเตอร์ มีการ สร้างแฟ้มงานและบันทึกไฟล์ลงไป มีการแสดงผล มีการคำนวณต่างๆ เช่นคำนวณเวลาในการตั้งค่าให้ timer การคำนวณ CRC การคำนวณการแปลงค่าของข้อมูล ดังรูปที่ 3.3

```
Imports System.IO.Ports
Imports System.IO

Public Class frmMain
    Private WithEvents serial As New SerialPort()

    Dim readPowerCmd = New Byte() {&H1, &H3, &H0, &H0, &H0, &HF, &H5, &HCE}
    Dim readVtCtCmd = New Byte() {&H1, &H3, &H0, &H37, &H0, &H2, &H75, &HC5}

    Dim saveCount As Long = 0
    Dim rotateCount As Long = 0

    Dim saveTime As Long
    Dim rotateTime As Long

    Private Sub btnStart_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnStart.Click
        ' อ่านค่า config ทั้งหมด
        Comm.ReadAllConfig()

        ' สร้างไดเร็คทอรี่สำหรับบันทึก Log (ถ้าไม่สร้างจะทำให้โปรแกรม Error)
        createDirectories(Comm.savePath)


' ค่าเวลาในการสร้าง Log file ขึ้นมาใหม่
```

```
        rotateTime = Comm.logRotateFreq * 60

        ' ค่าเวลาในการบันทึก Log แต่ละครั้ง
        saveTime = Comm.saveFreq

        ' Clear log file เก่าทิ้งโดยการก็อปเก็บไว้แล้วลบ Log file ปัจจุบันทิ้ง
        rotateLog()

        ' ตั้งค่าพอร์ต RS-232
        setSerialPort()

        'รีเซ็ต Counter
        saveCount = 0
        rotateCount = 0

        ' คำนวณเวลาที่เหลือในการบันทึกค่าครั้งถัดไปแล้วตั้งค่าให้กับ Timer
        Timer1.Interval = 3000 - (Environment.TickCount Mod 1000) + 20
        Timer1.Enabled = True

        lblDateTime.Select()

        ' Disable / Enable button
        btnStart.Enabled = False
        btnStop.Enabled = True
        btnSettings.Enabled = False
    End Sub

    Private Sub btnStop_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnStop.Click
        ' หยุดการทำงานของ timer
        Timer1.Enabled = False
```

```
        ' ปิดพอร์ตซีเรียล
        'If serial.IsOpen Then
        serial.Open()
        serial.ReadExisting()
        serial.Close()
        'End If

        ' เคลียร์ค่าบนหน้าจอทั้งหมด
        lblPr.Text = ""
        lblQr.Text = ""
        lblSr.Text = ""
        lblPFr.Text = ""
        lblDateTime.Text = ""

        ' Enable all button
        btnStart.Enabled = True
        btnStop.Enabled = True
        btnSettings.Enabled = True
    End Sub

    Private Sub btnSettings_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSettings.Click
        ' หยุดการทำงานของโปรแกรมก่อนเข้าหน้าตั้งค่า
        If btnStart.Enabled = False Then
            MsgBox("กรุณาหยุดการอ่านค่าจากมิเตอร์ก่อนทำการตั้งค่า", MsgBoxStyle.Exclamation,
"คำเตือน")
            Exit Sub
        End If

        ' โชว์หน้า settings
        frmSettings.ShowDialog()
    End Sub
```

```
' ตั้งค่าพอร์ตซีเรียล
Private Sub setSerialPort()
    " หากพอร์ต RS-232 เปิดอยู่ให้ทำการปิดก่อน
    'If serial.IsOpen Then serial.Close()
    Do While serial.IsOpen
        Application.DoEvents()
    Loop

    ' ตั้งค่าต่างๆให้กับพอร์ต RS-232
    serial.PortName = Comm.portName
    serial.BaudRate = Comm.baudRate
    serial.DataBits = Comm.dataBits
    serial.StopBits = Comm.stopBits
    serial.Parity = Comm.parity
    serial.Handshake = Comm.handshake
    serial.ReadTimeout = Comm.readTimeout
End Sub

' บันทึกค่าบนหน้าจอแสดงผลลงในไฟล์เมื่อครบตามเวลาที่กำหนด
Private Sub saveLog()
    ' เปิด log file เพื่อทำการบันทึกค่า
    Using sw As StreamWriter = New StreamWriter(Comm.savePath, True)
        ' บันทึกค่าต่างๆลงในไฟล์
        sw.WriteLine(lblDateTime.Text & ", " & lblPr.Text & "W, " & Format(lblPFr.Text,
"0.00") & ", " & lblSr.Text & "VA, " & lblQr.Text & "VAR")
    End Using
End Sub

' ก็อปปี้ Log file เก็บไว้เมื่อครบตามเวลาที่กำหนด
' โดยสร้างตัวเลขต่อท้ายจากนั้นลบ Log file เพื่อเก็บ log อันใหม่
Private Sub rotateLog()
```

```
    Dim fileRotate As String = Comm.savePath
    Dim fileNew As String = fileRotate
    Dim fileCount As Integer = 1

    Debug.Print((Environment.TickCount Mod 1000) & " rotate log")

    ' เช็คว่ามีไฟล์อยู่หรือไม่
    If File.Exists(fileRotate) Then
      ' วนลูปหาตัวเลขสุดท้ายเพื่อตั้งชื่อให้ไฟล์
      ' ฟอร์แมตของชื่อไฟล์จะเป็น filename-xx.txt
      ' xx คือ fileCount
      Do While File.Exists(fileNew)
        ' เช็คว่ามี filename-xx.txt อยู่หรือไม่
        ' จนกว่าจะไม่มี filename-xx.txt จึงหยุดแล้วใช้ค่า xx สุดท้ายนี้เป็นชื่อไฟล์ใหม่
        fileNew = Path.GetDirectoryName(fileRotate) & "\" & _
                  Path.GetFileNameWithoutExtension(fileRotate) & "-" & fileCount & _
                  Path.GetExtension(fileRotate)
        fileCount = fileCount + 1
      Loop

      ' ก็อปปี้ log file โดยใช้ชื่อไฟล์ filename-xx.txt ที่หาได้จากด้านบน
      File.Copy(fileRotate, fileNew)

      ' ลบ Log file เก่าออก
      File.Delete(fileRotate)
    End If
  End Sub

  Private Sub PowerDisplay()
    Dim data As Byte()
    Dim Vt As Integer
    Dim Ct As Integer
```

```
' อ่านค่า Vt และ Ct ออกมาเพื่อใช้ในการคำนวณ
data = readVtCt()
' หากพอร์ต RS-232 เปิดอยู่ให้ทำการปิด
If serial.IsOpen Then serial.Close()

' ดึงค่า
If data(0) <> &HFF Then
   ' อ่านค่า Voltage และ Current transformation ratio
   ' ตำแหน่ง Register ต่างๆเทียบจาก Manual
   Vt = Convert.ToInt32((data(3) * &H100) Or data(4))
   Ct = Convert.ToInt32((data(5) * &H100) Or data(6))
Else
   Exit Sub
End If

' อ่านค่า Power จากเครื่อง EPR
data = readPower()
' หากพอร์ต RS-232 เปิดอยู่ให้ทำการปิด
If serial.IsOpen Then serial.Close()

' ถ้าค่าแรกไม่ใช่ 0xFF แสดงว่าค่าที่อ่านถูกต้องหรือไม่มีข้อผิดพลาดขณะอ่านข้อมูลจากเครื่อง
EPR
If data(0) <> &HFF Then
   " อ่านค่า Voltage และ Current transformation ratio
   " ตำแหน่ง Register ต่างๆเทียบจาก Manual
   'Vt = Convert.ToInt32((data(113) * &H100) Or data(114))
   'Ct = Convert.ToInt32((data(115) * &H100) Or data(116))

   " แสดงค่า P, Q และ S ของทุกเฟส
   " จาก User Manual   P = ค่าที่อ่านได้ * Vt * Ct * 0.1
   " ค่าที่อ่านได้จะเป็นเลขฐานสิบหกจำนวน 2 ไบต์
```

```
" ตำแหน่งที่ 1 ของ Array คือ Address ของเครื่อง EPR
" ตำแหน่งที่ 2 คือ Function Code
" ตำแหน่งที่ 3 คือ ความยาวของข้อมูลที่เครื่อง EPR ส่งออกมา(ไม่รวมสามตำแหน่งแรกและ
สองตำแหน่งสุดท้าย)
" เริ่มจากตำแหน่งที่ 3 และ 4 ใน Array data เป็น Pr
" จะต้องแปลงค่าที่อ่านได้เป็นเลขฐานสิบก่อนทำการคำนวน
" สมมติค่าที่ตำแหน่ง 3 และ 4 ใน Array เป็น 0x01 และ 0x23
" ค่าของเลขฐานสิบหกจะเป็น 0x0123
" การคำนวณคือ shift bit ของเลขตำแหน่งที่ 3 ใน Array ไปทางซ้าย 8 ครั้ง ก็คือคูณด้วย
0xFF
" จากนั้นนำมา OR กับเลขในตำแหน่งที่ 4 ใน Array
" สูตรที่ได้ใน VB คือ (data(3) * &H100) Or data(4)
" จากนั้นแปลงให้เป็นฐานสิบโดยใช้ฟังก์ชัน Convert.ToInt32

'lblPtotal.Text = Convert.ToInt32((data(9) * &H100) Or data(10)) * Vt * Ct * 0.1

lblQr.Text = Convert.ToInt32((data(11) * &H100) Or data(12)) * Vt * Ct * 0.1

lblSr.Text = Convert.ToInt32((data(19) * &H100) Or data(20)) * Vt * Ct * 0.1

' ค่า Power Factor
' นำค่าที่อ่านได้มาหาร 100 (คูณ 0.01)
' หากบิตซ้ายสุดเป็น 1 ค่า PF จะติดลบ
' จึงต้องมีการเช็คโดยใช้สูตร data(27) And &H80 <> 0
If data(27) And &H80 <> 0 Then
   lblPFr.Text = Convert.ToInt32(data(28)) * 0.01 * -1
Else
   lblPFr.Text = Convert.ToInt32(data(28)) * 0.01
End If
'If data(29) And &H80 <> 0 Then
'   lblPFs.Text = Convert.ToInt32(data(30)) * 0.01 * -1
'Else
```

```
        '   lblPFs.Text = Convert.ToInt32(data(30)) * 0.01
        'End If
        'If data(31) And &H80 <> 0 Then
        '   lblPFt.Text = Convert.ToInt32(data(32)) * 0.01 * -1
        'Else
        '   lblPFt.Text = Convert.ToInt32(data(32)) * 0.01
        'End If
    End If
End Sub

' อ่านค่าจากเครื่อง Power EPR
Private Function readPower() As Byte()
    Dim readBuff() As Byte
    Dim byteToRead As Integer
    Dim compareCRC As Integer = 0

    Try
        ' เปิดพอร์ต
        serial.Open()

        ' ส่งคำสั่งอ่านค่าไปยังเครื่อง EPR
        serial.Write(readPowerCmd, 0, readPowerCmd.Length)

        'รอ 50ms เพื่อให้ EPR ส่งค่ามาครบ (จริงๆแล้วอาจจะไม่ถึง 50ms )
        For i = 1 To 60
            System.Threading.Thread.Sleep(1)
        Next i

        ' ดึงจำนวนไบต์ที่จะอ่านจากพอร์ต RS-232
        byteToRead = serial.BytesToRead
```

```
        ' กำหนดขนาด Array ใหม่
        ReDim readBuff(byteToRead - 1)

        serial.Read(readBuff, 0, byteToRead)
      Catch ex As Exception
        Debug.Print((Environment.TickCount Mod 1000) & ": Serial Time Out")

        ' หากเกิดข้อผิดพลาดระหว่างการอ่านข้อมูลจากพอร์ต RS-232 ส่งค่า 0xFF กลับไป
        readPower = New Byte() {&HFF}
        Exit Function
      End Try

      ' ถ้าอ่านค่าไม่ได้ให้คืนค่าเป็น 0xFF
      If readBuff.Length <= 0 Or readBuff.Length <> 35 Then
        readPower = New Byte() {&HFF}
        Exit Function
      End If

      ' คำนวนค่า CRC จากค่าที่อ่านได้จากพอร์ต RS-232
      compareCRC = CRC16(readBuff, readBuff.Length - 2)

      ' เปรียบเทียบค่า CRC ที่คำนวณได้กับค่า CRC (สองไบต์สุดท้าย)ที่ส่งมากับข้อมูลจากพอร์ต
RS-232
      ' ถ้าเหมือนกันให้ส่งค่าที่อ่านได้กลับไป แต่ถ้าไม่เหมือนกันส่งค่า 0xFF กลับไป
      If ((compareCRC And &HFF00) \ &HFF) = readBuff(readBuff.Length - 1) And
(compareCRC And &HFF) = readBuff(readBuff.Length - 2) Then
        readPower = readBuff
      Else
        readPower = New Byte() {&HFF}
      End If
    End Function
```

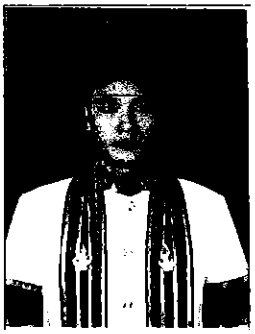
```
' อ่านค่าจากเครื่อง Power EPR
Private Function readVtCt() As Byte()
    Dim readBuff() As Byte
    Dim byteToRead As Integer
    Dim compareCRC As Integer = 0

    Try
        ' เปิดพอร์ต
        serial.Open()

        ' ส่งคำสั่งอ่านค่าไปยังเครื่อง EPR
        serial.Write(readVtCtCmd, 0, readVtCtCmd.Length)

        'รอ 10ms เพื่อให้ EPR ส่งค่ามาครบ
        For i = 1 To 20
            System.Threading.Thread.Sleep(1)
        Next i

        ' ดึงจำนวนไบต์ที่จะอ่านจากพอร์ต RS-232
        byteToRead = serial.BytesToRead

        ' กำหนดขนาด Array ใหม่
        ReDim readBuff(byteToRead - 1)

        serial.Read(readBuff, 0, byteToRead)
    Catch ex As Exception
        Debug.Print((Environment.TickCount Mod 1000) & ": Serial Time Out")

        ' หากเกิดข้อผิดพลาดระหว่างการอ่านข้อมูลจากพอร์ต RS-232 ส่งค่า 0xFF กลับไป
        readVtCt = New Byte() {&HFF}
        Exit Function
    End Try
```

```
    ' ถ้าอ่านค่าไม่ได้ให้คืนค่าเป็น 0xFF
    If readBuff.Length <= 0 Or readBuff.Length <> 9 Then
      readVtCt = New Byte() {&HFF}
      Exit Function
    End If

    ' คำนวนค่า CRC จากค่าที่อ่านได้จากพอร์ต RS-232
    compareCRC = CRC16(readBuff, readBuff.Length - 2)

    ' เปรียบเทียบค่า CRC ที่คำนวณได้กับค่า CRC (สองไบต์สุดท้าย) ที่ส่งมากับข้อมูลจากพอร์ต
RS-232
    ' ถ้าเหมือนกันให้ส่งค่าที่อ่านได้กลับไป แต่ถ้าไม่เหมือนกันส่งค่า 0xFF กลับไป
    If ((compareCRC And &HFF00) \ &HFF) = readBuff(readBuff.Length - 1) And
(compareCRC And &HFF) = readBuff(readBuff.Length - 2) Then
      readVtCt = readBuff
    Else
      readVtCt = New Byte() {&HFF}
    End If
  End Function

  ' ทำการคำนวณค่า CRC16 ของข้อมูล
  ' Reference: http://www.ptcelectronics.com/pdfs/NU-eNod3C-MOD-E-0911_165704-I.pdf
  ' Page: 26
  Function CRC16(ByVal data() As Byte, ByVal length As Integer) As Integer
    Dim returnCRC16 As Integer
    Dim i As Long, j As Long, Odd As Boolean

    returnCRC16 = -1      ' ค่าเริ่มต้นเป็น 0xFFFF
    For i = 0 To length - 1   ' เลื่อนและทำการคำนวณทีละไบต์ตามความยาวของข้อมูล
      ' นำดาต้าในแต่ละไบต์มาทำการ XOR กับตัวแปร returnCRC16
      returnCRC16 = (returnCRC16 And &HFF00) Or ((returnCRC16 And 255) Xor data(i))
```

```
        ' เลื่อนบิต(Shift bit)ไปทางขวาทีละบิตแปดครั้ง
        For j = 1 To 8
          ' เช็คว่าบิตที่จะเลื่อนไปเป็น 1 หรือไม่
          Odd = returnCRC16 And 1

          ' เลื่อนบิตไปทางขวา 1 ครั้ง
          returnCRC16 = ((returnCRC16 And &HFFFE) \ 2) And &H7FFF 'shift right

          ' ถ้าบิตที่เลื่อนไปมีค่าเป็น 1 ให้นำตัวแปร returnCRC16 XOR 0xA001
          If Odd Then returnCRC16 = &HA001 Xor returnCRC16
        Next
      Next
      ' เมื่อคำนวณเสร็จแล้วคืนค่า returnCRC16 กลับไป
      CRC16 = returnCRC16
    End Function

    '
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick 'Handles Timer1.Elapsed
      Timer1.Enabled = False

      'Debug.Print((Environment.TickCount Mod 1000) & " Timer1")

      ' แสดงเวลาบนหน้าจอ
      lblDateTime.Text = Format(Now, "yyyy-MMM-dd HH:mm:ss")
      Application.DoEvents()

      ' นับเวลาเพื่อบันทึกค่าหรือเปลี่ยนไฟล์
      saveCount = saveCount + 1
      rotateCount = rotateCount + 1
```

```
        ' อ่านค่าจากเครื่อง EPR-04S แล้วทำการแสดงผล
        PowerDisplay()
        ' ถ้าถึงเวลาบันทึกค่าให้บันทึกค่า
        If saveCount = saveTime Then
            saveLog()      ' บันทึกค่าที่แสดงบนหน้าจอลง Log file
            saveCount = 0
        End If
        ' ทำการเปลี่ยนไฟล์ที่บันทึก
        If rotateCount >= rotateTime Then
            rotateLog()    ' เปลี่ยน Log file
            rotateCount = 0
        End If
        ' คำนวณเวลาเพื่อบันทึกค่าครั้งถัดไปแล้วตั้งค่าให้กับ Timer
        Timer1.Interval = 1000 - (Environment.TickCount Mod 1000) + 50
        Timer1.Enabled = True
    End Sub

    Private Sub createDirectories(ByVal path As String)
        Dim index As Integer = 0
        Dim directoryName As String = Comm.savePath.Substring(index,
Comm.savePath.IndexOf("\", index) + 1)
        Do While directoryName.Length > 0
            If Not Directory.Exists(directoryName) Then
                Directory.CreateDirectory(directoryName)
            End If
            index = Comm.savePath.IndexOf("\", index) + 1
            directoryName = Comm.savePath.Substring(0, Comm.savePath.IndexOf("\", index) + 1)
        Loop
    End Sub
End Class
```

ประวัติผู้ดำเนินโครงงาน



ชื่อ นายกรกช วิชัยทา

ภูมิลำเนา 38 หมู่ 10 ต. กลางเวียง อ. เวียงสา จ. น่าน

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนสตรีศรีน่าน จ. น่าน
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 5 สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail: **koraoktw53@email.nu.ac.th**