โปรแกรมแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

MONITOR DISPLAY AND DATA LOGGING SOFTWARE USED FOR DIGITAL ENERGY METER

นายศุภสัณห์ แจ่มแจ้ง รหัส 54364238

นางสาวฐาปนี ดวงเทียน รหัส 54363705

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2557

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน โปรแกรมแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

ผู้ดำเนินโครงงาน นายศุภสัณห์ แจ่มแจ้ง รหัส 54364238

นางสาวฐาปนี ดวงเทียน รหัส 54363705

ที่ปรึกษาโครงงาน ดร. ปิยดนัย ภาชนะพรรณ์

สาขาวิชา วิศวกรรมไฟฟ้า

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2557

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน

(ดร. ปิยดนัย ภาชนะพรรณ์)

.....กรรมการ

(ดร. ชัยรัตน์ พินทอง)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ดร. พนัส นัถฤทธิ์)

ชื่อหัวข้อโครงงาน	โปรแกรมแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล		
ผู้ดำเนินโครงงาน	นายศุภสัณห์	แจ่มแจ้ง	รหัส54364238
	นางสาวฐาปนี	ดวงเทียน	รหัส54363705
ที่ปรึกษาโครงงาน	ดร. ปิยดนัย	ภาชนะพรรณ์	
สาขาวิชา	วิศวกรรมไฟฟ้า		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2557		

บทคัดย่อ

มิเตอร์พลังงานไฟฟ้าแบบดิจิตอล เป็นเครื่องมือที่ได้รับการพัฒนาความสามารถให้วัดค่าและ แสดงผลค่าทางไฟฟ้าออกมาได้หลายรูปแบบ โครงงานนี้เป็นการพัฒนาระบบเพื่อเก็บค่าและแสดงผลจากข้อมูลที่ มิเตอร์วัดได้ ซึ่งได้แก่ค่า พลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า ออกมายังหน้าจอคอมพิวเตอร์โปรแกรมสามารถปรับค่าเวลาในการแสดงค่าพารามิเตอร์ และเวลาในการบันทึกผลได้ โปรแกรมมีการบันทึกข้อมูลลงไฟล์และมีการบันทึกข้อมูลแยกเป็นวันๆ ตามวันที่ (ปี/เดือน/วัน)และจัดเก็บในรูป ของไฟล์ข้อความผลการทดลองจะพบว่าค่าที่โปรแกรมบนหน้าจอแสดงผล ตรงตามหน้าจอมิเตอร์ ไม่มีความคลาดเคลื่อนซึ่งโปรแกรมที่พัฒนาอยู่บนพื้นฐานของโปรแกรม Notepad ++ และ Glade Interface Designer Gtk + 2มีการสื่อสารระหว่างมิเตอร์และคอมพิวเตอร์ผ่านมาตรฐานการสื่อสารRS-485

Project title	The Display and Data Logging Software used with Digital Energy Meter
Name	Mr. Supasan jamjang ID.54364238
	Miss Thapanee Dongthain ID.54363705
Project advisor	Dr. Piyadanai Pachanapan,
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic year	2014

Abstract

This project is the implementation of software to collect data (including Energy (Wh), Voltage (V), Current (A) from the energy digital meter. This software will display the monitoring data on the computer screen. Then, the collected data will be daily logged and stored in form of the text file. The experiment found that the data from monitoring display and in the log file are the same as that is displayed on the meter display. Moreover, this software was developed based on the program Notepad++ and Glade Interface Designer Gtk+2, by using the communication between a digital meter to computer via RS-485.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยการดูแลจาก ดร.ปิยดนัย ภาชนะพรรณ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานที่คอยให้คำปรึกษาและแนะนำ ขั้นตอนในการทำ โครงงานเกี่ยวกับการออกแบบโปรแกรมเก็บค่าและแสดงผลแบบติดตามสำหรับมิเตอร์พลังงานไฟฟ้าแบบดิจิตอล นอกจากนั้น ยังให้การตรวจทานเล่มปริญญานิพนธ์ ผู้ดำเนินโครงงานจึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ดร. ชัยรัตน์ พิมทองและผศ.ดร.พนัส นัถฤทธิ์ เป็นคณะกรรมการในการสอบโครงงานที่ให้คำ แนะนำ ชี้แนะแนวทาง ข้อคิดเห็นต่างๆและรวมถึงการตรวจรูปเล่มปริญญานิพนธ์ที่เป็นประโยชน์ในโครงงานนี้ ทำ ให้โครงงานนี้ออกมาสมบูรณ์ยิ่งขึ้น

ขอขอบคุณเพื่อนๆ สาขาคอมพิวเตอร์ และ สาขาไฟฟ้ากำลัง รุ่นที่ 15 ทุกคนที่คอยช่วยเหลืองานของข้าพเจ้าเพราะเป็นการกระตุ้นงานของข้าพเจ้าเองเพื่อให้ข้าพเจ้าได้ทำ โครงงานในครั้งนี้จนบรรลุวัตถุประสงค์ตามที่ข้าพเจ้าได้ตั้งเป้าหมายไว้และท้ายที่สุดขอกราบขอบพระคุณบิดามารดาผู้เป็นที่รักและเคารพของข้าพเจ้าผู้ที่คอยให้กำลังใจและให้โอกาสในการทา โครงงานมาโดยตลอดเวลาจนทำให้ประสบความสำเร็จดังทุกวันนี้

นายศุภสัณห์ แจ่มแจ้ง

นางสาวฐาปนี ดวงเทียน

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบัน อพาร์ทเมนท์ คอนโด แมนชั่น หอพัก และ อาคาร พลาซ่า ศูนย์การค้าหอพัก สร้างขึ้นอย่างมากมาย การที่จะตรวจยอดการใช้ไฟฟ้าโดยใช้มิเตอร์ไฟฟ้าแบบจานหมุนในการวัดหน่วยค่าไฟ ทำให้เสียเวลาในการทางเดินไปจดค่าไฟ และโอกาสในการจดค่าผิดพลาดมีสูง จึงนำมิเตอร์ไฟฟ้าแบบดิจิตอลเข้ามาใช้ เพื่อสามารถเปิดดูหน่วยค่าไฟได้เลยโดยไม่ต้องเดินไปจดค่าที่ตัวมิเตอร์ เนื่องจากมิเตอร์ไฟฟ้าแบบดิจิตอลสามารถส่งข้อมูลผ่าน RS-485 เพื่อส่งข้อมูลไปเก็บไว้ที่คอมพิวเตอร์ โปรแกรมสามารถบันทึกค่าอัตโนมัติเมื่อเปิดคอมพิวเตอร์ และสามารถตั้งค่าเวลาในการบันทึกข้อมูลได้ด้วย

โครงงานนี้จะพัฒนาโปรแกรมบนคอมพิวเตอร์โดยใช้ภาษาไพธอนให้สามารถแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลโดยเชื่อมผ่าน RS-485 โปรแกรมแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลช่วยประหยัดเวลาในการอ่านหน่วยค่าไฟในกรณีที่ หอพัก อพาร์ทเมนท์ แมนชั่น หรือ คอนโด มีห้องพักหลายห้องหลายชั้น สามารถดึงข้อมูลจากมิเตอร์ไฟฟ้าแบบดิจิตอลแสดงผลผ่านคอมพิวเตอร์ได้อย่างรวดเร็วประหยัดเวลา ในการอ่านและจัดเก็บข้อมูลการใช้พลังงานไฟฟ้าและยังสามารถตรวจสอบการใช้งานพลังงานไฟฟ้าย้อนหลังได้

1.2 วัตถุประสงค์ของโครงงาน

เพื่อพัฒนาโปรแกรมบนภาษาไพธอนให้คอมพิวเตอร์สามารถสื่อสารกับมิเตอร์วัดพลังงานไฟฟ้าชนิดดิจิตอลโดยเชื่อมผ่าน RS-485 พร้อมทั้งแสดงผลและจัดเก็บข้อมูลได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 สามารถนำข้อมูลการใช้พลังงานไฟฟ้าที่บันทึกไว้ไปวาดกราฟเทียบการใช้ไฟฟ้าในและเดือนได้

1.3.2 สามารถนำข้อมูลที่ได้มาพัฒนาเพื่อเชื่อมต่อทางอินเตอร์เน็ตให้สามารถตรวจสอบหน่วยค่าไฟแบบออนไลน์ได้ทุกที่

1.4 ขอบเขตการทำโครงงาน

1.4.1 ใช้ภาษาไพธอนพัฒนาโปรแกรมบนคอมพิวเตอร์เพื่อ ให้สื่อสารกับมิเตอร์ได้

1.4.2 แสดงผลการอ่านค่า เวลา วัน เดือน ปี พลังงานไฟฟ้า แรงดันไฟฟ่า กระแสไฟฟ้าจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลบนคอมพิวเตอร์

1.4.3 การบันทึกค่าของข้อมูลที่อ่านได้ตามเวลาที่กำหนด จะบันทึกค่าพารามิเตอร์ต่างๆ ตามการตั้งเวลาและบันทึกลงในไฟล์ log.txt จะแยกบันทึกข้อมูลเป็นรายวันลงในไฟล์ DailyLog

1.5 ขั้นตอนการดำเนินงาน

1.5.1 ออกแบบชุดทดสอบมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

1.5.2 ใช้ ภาษาไพธอนพัฒนาโปรแกรมเพื่อ ให้สื่อสารกับมิเตอร์

1.5.3 ทดสอบโปรแกรมกับชุดทดสอบที่ได้ออกแบบ

1.5.4 เสนอโครงงานให้อาจารย์ที่ปรึกษาตรวจสอบ

1.5.5 ปรับปรุงแก้ไขโครงงาน

1.5.6 เขียนโครงร่างปริญญานิพนธ์

1.5.7 จัดทำรูปเล่มปริญญานิพนธ์

1.5.8 เรียบเรียงปริญญานิพนธ์เป็นรูปเล่มและนำเสนออาจารย์ที่ปรึกษา

1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แสดงแผนการดำเนินงาน

กิจกรรม	ปี 2557-2558							
	ส.ค.	ก.ย.-ต.ค	พ.ย-ธ.ค.	ม.ค-ก.พ	มี.ค-เม.ย	พ.ค	มิ.ย.	ก.ค
1.ศึกษาทำความเข้าใจมิเตอร์ไฟฟ้าแบบดิจิตอล								
2.สร้างชุดทดสอบการทำงานของมิเตอร์ดิจิตอล								
3.ทดสอบการทำงานของมิเตอร์และโปรแกรมที่ได้พัฒนา								
4.เขียนโปรแกรมรับค่าจากมิเตอร์ด้วยภาษา Python								
5.เสนอโครงงานให้อาจารย์ที่ปรึกษาตรวจสอบ								
6.ปรับปรุงแก้ไขโครงงาน								
7.เขียนโครงร่างปริญญานิพนธ์								
8.จัดทำรูปเล่มปริญญานิพนธ์								
9.เรียบเรียงปริญญานิพนธ์เป็นรูปเล่มและนำเสนออาจารย์ที่ปรึกษา								

1.7 ผลที่คาดว่าจะได้รับ

1.เข้าใจการติดต่อสื่อสารระหว่างมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลกับคอมพิวเตอร์

2.มิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลสามารถรับส่งข้อมูลการใช้ไฟฟ้ากับคอมพิวเตอร์ได้เพื่อใช้ แสดงผลบนหน้าจอคอมพิวเตอร์ และ จัดเก็บข้อมูลได้

3. สามารถตรวจสอบข้อมูลการใช้พลังงานไฟฟ้าย้อนหลังได้

1.8 รายละเอียดงบประมาณตลอดโครงงาน

1. ค่าวัสดุสำนักงาน 500 บาท

2.ค่าปริ้นงาน ถ่ายเอกสาร ทำรูปเล่ม 1000 บาท

3. อื่นๆ 500 บาท

รวมเป็นเงิน 2000 บาท (สองพันบาทถ้วน)

หมายเหตุ : ขอถัวเฉลี่ยจ่ายทุกรายการ

บทที่ 2
ทฤษฎีหลักการและเอกสารที่เกี่ยวข้อง

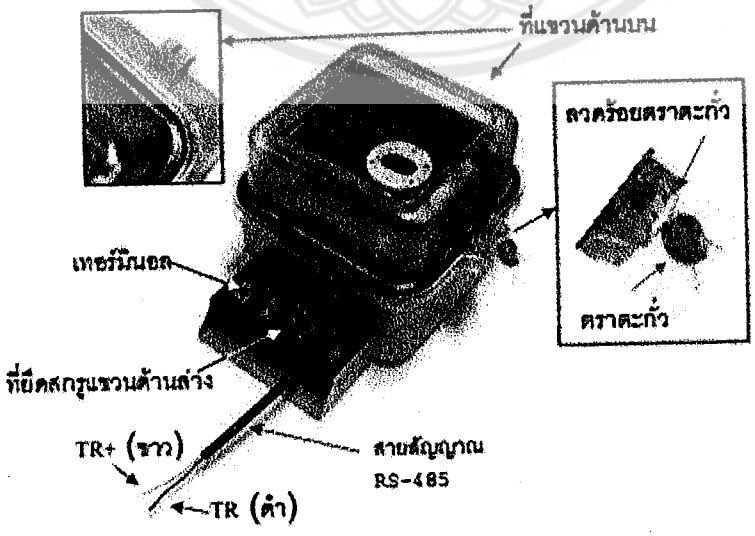
ในการจัดทำโครงงานนี้จะพัฒนาโปรแกรมบนคอมพิวเตอร์โดยใช้ภาษาไพธอนให้สามารถแสดงผลและจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลโดยเชื่อมผ่านมาตรฐาน RS-485จะแสดงผลรายละเอียดของอุปกรณ์ในเบื้องต้นจะถูกอธิบายในบทนี้

2.1 มิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล MITSUBISHI รุ่น S-X 1

คือมิเตอร์อีเล็กทรอนิกส์ แบบหนึ่งเฟส ทำงานโดยอาศัยหลักการประมวลผลแบบดิจิตอล เพื่อวัด และ คำนวณค่าพลังงานไฟฟ้า (kWh) อย่างถูกต้องแม่นยำสามารถวัดการใช้พลังงานไฟฟ้าสำหรับ อพาร์ทเมนท์ คอนโด แมนชั่น หอพัก และ อาคาร พลาซ่า ศูนย์การค้า และ หน่วยงานอื่นๆ ตัวมิเตอร์มีช่องต่อพอร์ต RS-485 ที่สนับสนุนการเชื่อมต่อผ่านทางระบบเครือข่ายคอมพิวเตอร์

2.1.1 คุณสมบัติของมิเตอร์พลังงานไฟฟ้าแบบดิจิตอล

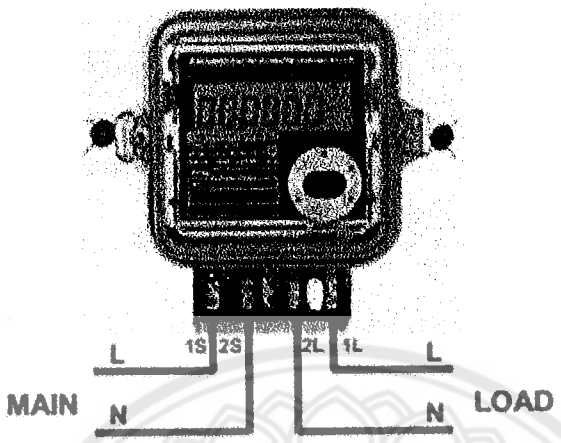
มิเตอร์ไฟฟ้าแบบดิจิตอลแสดงผลค่าพลังงานไฟฟ้า (kWh) บนหน้าจอ ชนิด LCD โดยแสดงตัวเลข 6 หลักพร้อมทศนิยม 1 ตำแหน่งมีความแม่นยำในการวัดสูงถึง CLASS 1 ซึ่งเป็นไปตามมาตรฐานผลิตภัณฑ์มิเตอร์ IEC 62053-21 (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [1])



รูปที่ 2.1 ส่วนประกอบของมิเตอร์

2.1.2 การติดตั้งมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

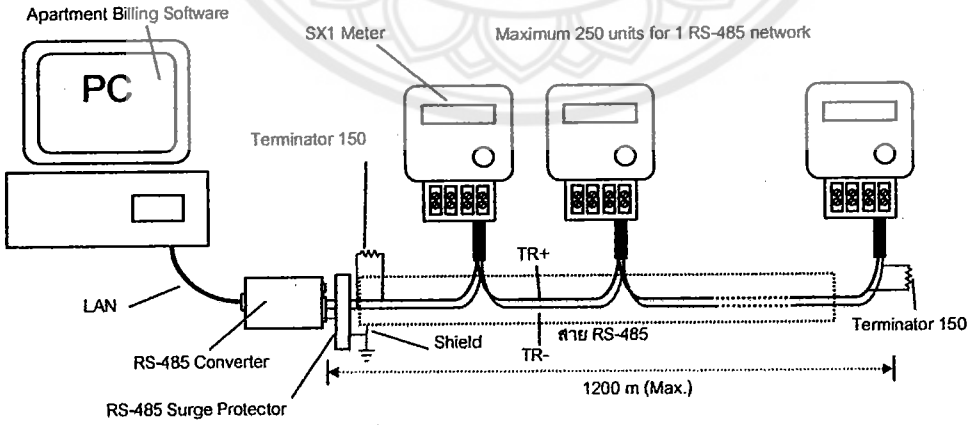
นำมิเตอร์แขนโดยใช้ที่แขวนด้านบนของมิเตอร์ด้วยสกรูขนาดเส้นผ่านศูนย์กลางไม่เกิน 5 มิลลิเมตร แล้วเลื่อนตัวมิเตอร์ลงเล็กน้อยยึดเข้ากันที่แขวนด้านบนและด้านล่าง



รูปที่ 2.2 ตัวอย่างการต่อสายไฟจากแหล่งจ่ายและสายไฟที่ต่อเข้าโหลด

2.1.3 การต่อสายไฟเมนและสายไฟโหลด

ตรวจสอบการต่อสาย ชนิดสาย การต่อสายผิดอาจทำให้มิเตอร์เสียหาย อุปกรณ์ภายนอกเสียหาย เกิดไฟไหม้ไฟฟ้าลัดวงจร การต่อสายจากแหล่งจ่ายและสายไฟที่ต่อเข้าโหลดในขณะที่ยังไม่จ่ายไฟให้มิเตอร์



รูปที่ 2.3 การติดตั้งมิเตอร์เข้ากับเครือข่าย RS-485

2.2 การส่งข้อมูลบนมาตรฐาน RS-485

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

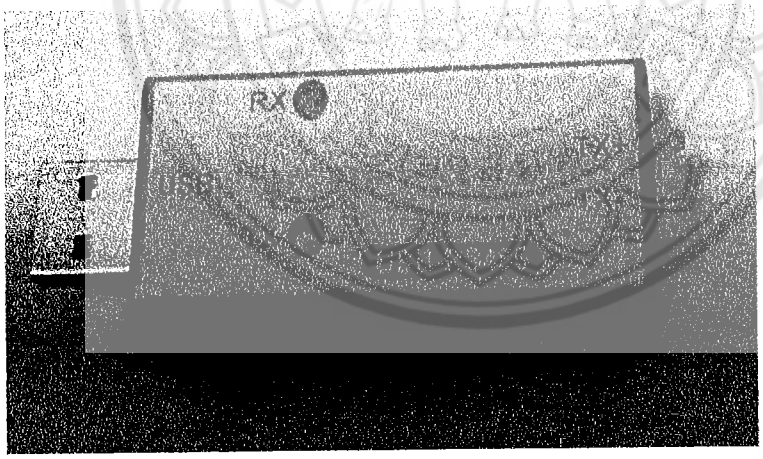
RS485 เป็นการสื่อสารอนุกรมของมิเตอร์ไฟฟ้าแบบดิจิตอลรุ่น EPR-04S ภายในจะมีการติดต่อสื่อสารโดยใช้โปรโตคอล MODBUS ที่จะส่งค่าพารามิเตอร์ต่างๆไปที่คอมพิวเตอร์ แต่การที่จะเชื่อมต่อเข้ากับคอมพิวเตอร์ได้นั้น ต้องมีการแปลงจาก RS485 ไปเป็น RS 232 เพื่อที่จะเข้ากับพอร์ตขนานของคอมพิวเตอร์ได้ การส่งข้อมูลจากมิเตอร์ไปยังคอมพิวเตอร์ 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(สามารถดูรายละเอียดเพิ่มเติมได้ที่ [2])

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	20 kbs	10 Mbs	35 Mbs
Max speed at 1200 m	(1 kbs)	100 kbs	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]

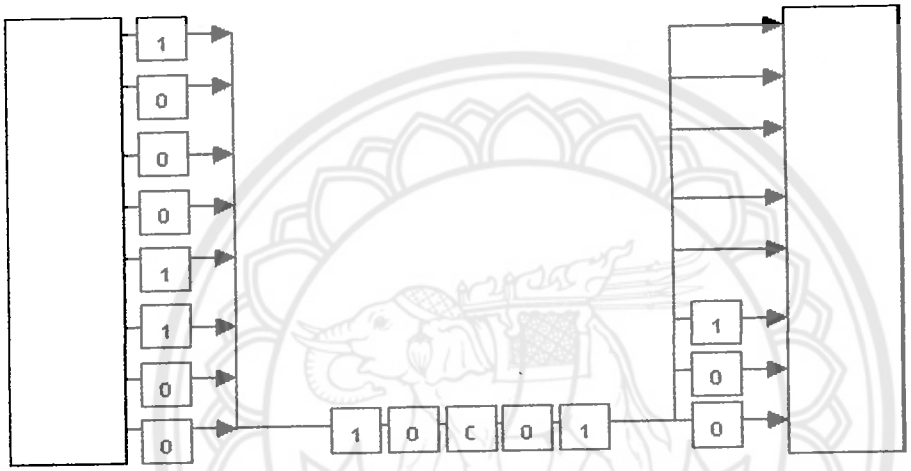


รูปที่ 2.5 มาตรฐานสัญญาณ RS-485

2.3 ทฤษฎีระบบสื่อสารที่เกี่ยวข้อง

2.3.1 การส่งข้อมูลแบบอนุกรม (Serial Transmission)

การส่งข้อมูลแบบอนุกรมนั้น จะเป็นการส่งข้อมูลทีละบิตผ่านสายส่งข้อมูล โดยการเรียงลำดับการส่งจะขึ้นอยู่กับชนิดของอุปกรณ์ โดยที่ตัวส่งจะมีข้อมูลหลายๆชุดหรือหลายๆบิตรวมกัน แต่การส่งนั้นจะส่งได้เพียงทีละ 1บิต ส่งแบบเรียงกันมา ไม่สามารถส่งหลายๆบิตพร้อมกันได้ ดังภาพที่แสดง



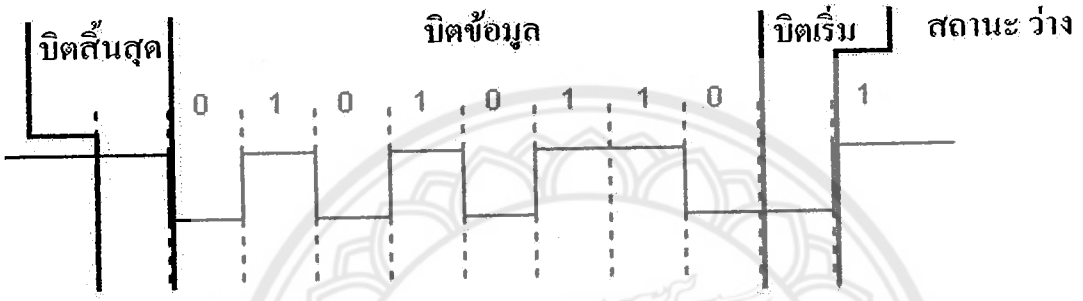
รูปที่ 2.6 การส่งข้อมูลของระบบสื่อสารอนุกรม [2]

จากรูปที่ 2.6 ด้านซ้ายคือต้นทาง และ ด้านขวาคือปลายทาง โดยชุดคำสั่งข้อมูลจะถูกส่งออกจากคอมพิวเตอร์ไปยังมิเตอร์ และเมื่อมิเตอร์รับชุดคำสั่งเข้ามา ก็จะตอบสนองเป็นข้อมูลตามชุดคำสั่งออกไปยังคอมพิวเตอร์ และข้อมูลทั้งหมดนั้นจะถูกส่งผ่านระบบสื่อสารอนุกรม ในชุดข้อมูล อักษรจะประกอบด้วยบิตข้อมูลจำนวน 8 บิต เรียงเป็นลำดับ ข้อมูลจะถูกส่งออกมาทีละบิตระหว่างต้นทางถึงปลายทาง และปลายทางจะรวบรวมบิตเหล่านี้ทีละบิตจนครบ 8 บิต เป็น 1 ตัวอักษร การส่งข้อมูลแบบนี้จะเหมาะสำหรับการส่งระยะทางไกลๆ

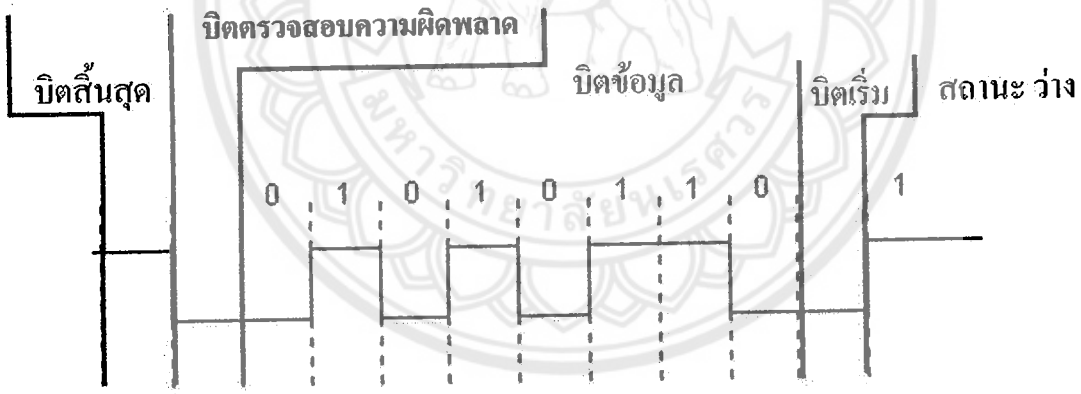
โดยทั่วไปแล้วการส่งข้อมูลนั้นจะประกอบไปด้วยกลุ่มของตัวอักษร ดังนั้นในการส่งข้อมูลแบบอนุกรมนี้จึงเกิดปัญหาขึ้นว่า แล้วต้นทางและปลายทางจะทราบได้อย่างไรว่า จะแบ่งแต่ละตัวอักษรตรงบิตใด จึงเกิดวิธีการสื่อสารข้อมูลขึ้น 2 แบบคือ การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission) และการสื่อสารแบบซิงโครนัส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



รูปที่ 2.7 การสื่อสารแบบอะซิงโครนัสที่ไม่ได้ใช้บิตตรวจสอบความผิดพลาด [2]

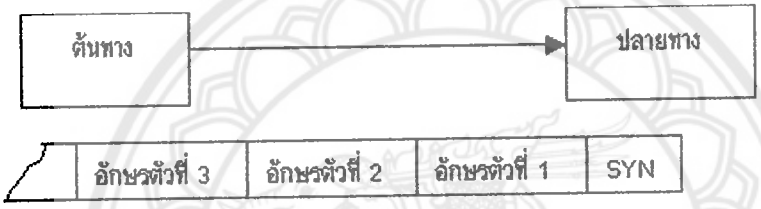


รูปที่ 2.8 การสื่อสารแบบอะซิงโครนัสที่ใช้บิตตรวจสอบความผิดพลาด [2]

จากรูปที่ 2.7 และ 2.8 จะเห็นว่าขณะที่ไม่มีข้อมูลส่งออกมาสถานะของการส่งจะเป็นแบบว่าง (Idle) ซึ่งจะมีระดับของสัญญาณเป็น 1 ตลอดเวลา เพื่อความแน่ใจว่าปลายทาง หรือฝ่ายรับยังคงติดต่อกับต้นทาง หรือฝ่ายส่งอยู่ เมื่อเริ่มจะส่ง ข้อมูลสัญญาณของอะซิงโครนัสจะเป็น 0 ในหนึ่งช่วงสัญญาณนาฬิกา ซึ่งบิตนี้จะเรียกว่าบิตเริ่มต้น ตามหลังของบิตเริ่มต้นจะเป็นบิตข้อมูลสำหรับ 1 ตัวอักษร ตามหลังบิตข้อมูลก็จะเป็นบิตตรวจความถูกต้อง แล้วจะตามด้วยบิตสิ้นสุด ถ้าไม่ใช่บิตตรวจความถูกต้อง ตามหลังบิตข้อมูลก็จะเป็นบิตสิ้นสุดเลย หลังจากนั้นถ้าไม่มีข้อมูลส่งออกมาสัญญาณจะกลับไปอยู่ที่สถานะแบบว่างอีก เพื่อรอการส่งข้อมูลต่อไป จะเห็นว่าการ

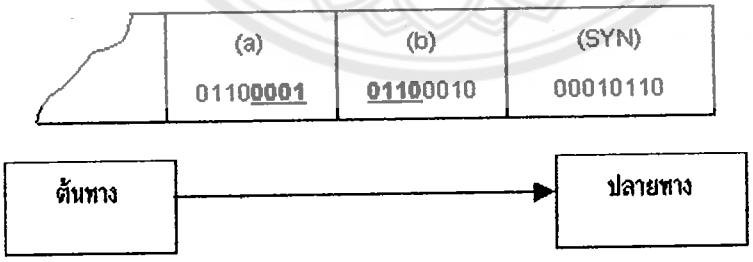
สื่อสารแบบอะซิงโครนัสนี้ มีลักษณะเป็นไปทีละตัวอักษร และสัญญาณที่ส่งออกมา มีบางส่วนเป็นบิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด ทำให้ความเร็วในการส่งข้อมูลต่อวินาทีน้อยลงไป เนื่องจากต้อง สูญเสียช่องทางการสื่อสารให้กับ บิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด (ถ้ามีใช้) ตลอดเวลา

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



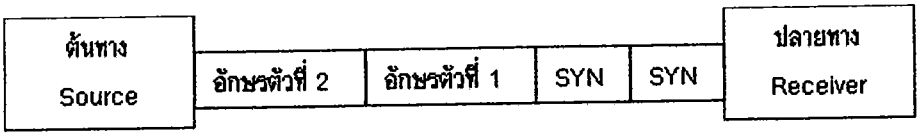
รูปที่ 2.9 ตัวอย่างของการส่งอักษรซิง [2]

จากรูปที่ 2.9 เมื่อปลายทางตรวจพบอักขระซิง หรือ 00010110 แล้วจะทราบได้ทันทีว่าบิตที่ตามมาคือบิตตัวอักษรแต่ละตัว แต่การใช้อักขระซิงเพียงตัวเดียวอาจเกิดข้อผิดพลาดได้ เช่น ถ้าส่งตัวอักษร b และตัวอักษร a ติดต่อกันไป ซึ่งตัวอักษร b มีรูปแบบบิตคือ 01100010 และตัวอักษร a มีรูปแบบบิตคือ 01100001 การส่งจะแสดงได้ดังรูปที่ 2.10

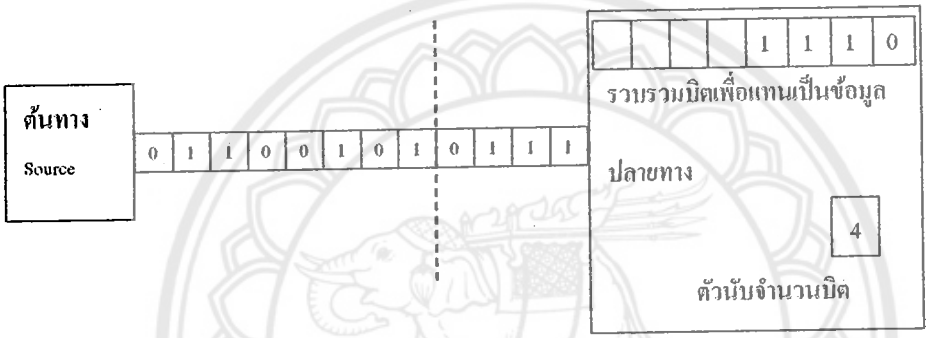


รูปที่ 2.10 ตัวอย่างการส่งอักษรซิงตามด้วยอักษร a และ b [2]

จากรูปจะเห็นว่าเครื่องปลายทางจะตรวจพบอักขระซิงระหว่างบิตของตัวอักษร b และตัวอักษร a ทำให้เข้าใจว่าบิตต่อไปจะเป็นบิตของกลุ่มข้อมูล ซึ่งจะทำให้การรับข้อมูลนั้นเกิดผิดพลาดขึ้นได้ ดังนั้นจึงแก้ปัญหาด้วยการใช้อักขระซิง 2 ตัวต่อกันเป็นลักษณะของบิตพิเศษที่บอกให้ทราบว่าเป็นจุดเริ่มต้นบิตของกลุ่มข้อมูล ตัวอย่างของการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส และการตัดแถวของบิตข้อมูลออกเป็นกลุ่มทีละ 8 บิต เพื่อแทนข้อมูลแสดงได้ดังรูป 2.11

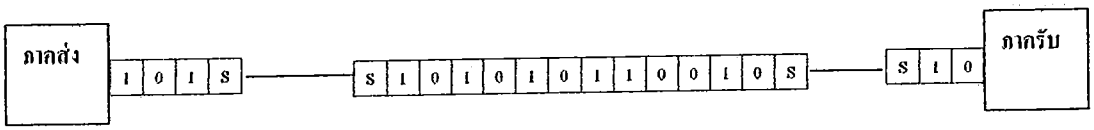


รูปที่ 2.11 ตัวอย่างการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส [2]



รูปที่ 2.12 การตัดแถวของบิตออกเป็นกลุ่ม กลุ่มละ 8 บิต [2]

รูปที่ 2.13 การส่งผ่านข้อมูลแบบซิงโครนัส [5]



รูปที่ 2.14 การส่งผ่านข้อมูลแบบอะซิงโครนัส [2]

จากรูปที่ 2.13 แสดงให้เห็นว่าการส่งผ่านข้อมูลแบบซิงโครนัสนั้นส่วนมากแล้ว ตลอดทางของสายส่งจะใช้ส่งผ่านข้อมูลเต็มตลอดทั้งสาย ส่วนรูปที่ 2.14 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 %

2.4 การรับส่งข้อมูลโปรโตคอลของมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

โปรโตคอล เปรียบเสมือนภาษาที่ใช้ในการสื่อสารในระบบเครือข่ายคอมพิวเตอร์ สำหรับมิเตอร์จะมีคู่มือโปรโตคอลมาพร้อมมิเตอร์

1. RS-485 Protocol

1.1 Specifications

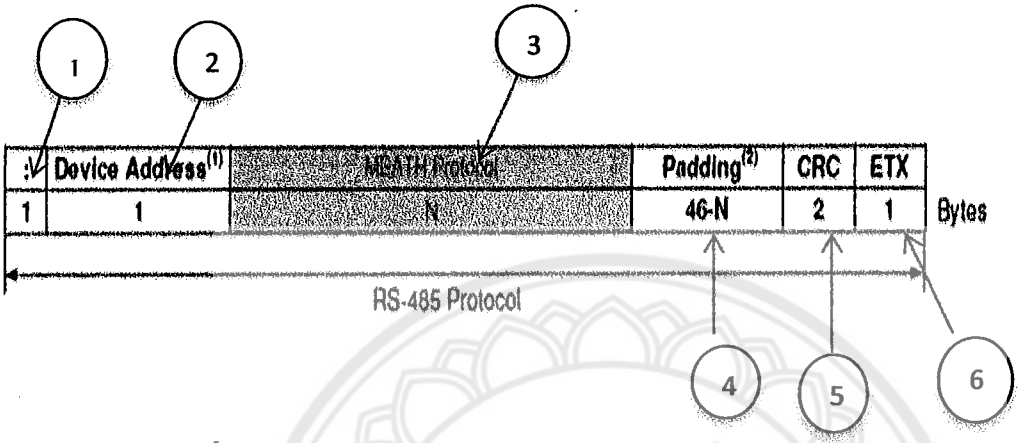
Table 1.1-1 RS-485 communication and RS-485 protocol specifications

Type of transmission	Asynchronous serial bit, Half duplex
Physical interface	RS – 485 (2wires) line driver.
Transmission speed	19,200 bps.
Protocol standard	RS-485 Protocol
Technique	Single Master / Multi Slaves
Packet size	51 bytes (Fixed)
Error detection	CCITT CRC16 (cyclical redundancy check) Polynomial = 0x1021 Non-reflect algorithm (MSB first) Initial value – 0xFFFF Note : Testing value = "123456789" CRC value = 0x29B1 See annex A for more details
Data format	Binary LSB first ST | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | SP ST : Start bit (1bit / logic 0) SP : Stop bit (1bit / logic 1) B0 – B7 : Binary data (8bits)

รูปที่ 2.15 การรับส่งข้อมูลโปรโตคอลของมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

2.4.1 โปรโตคอลมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล SX1

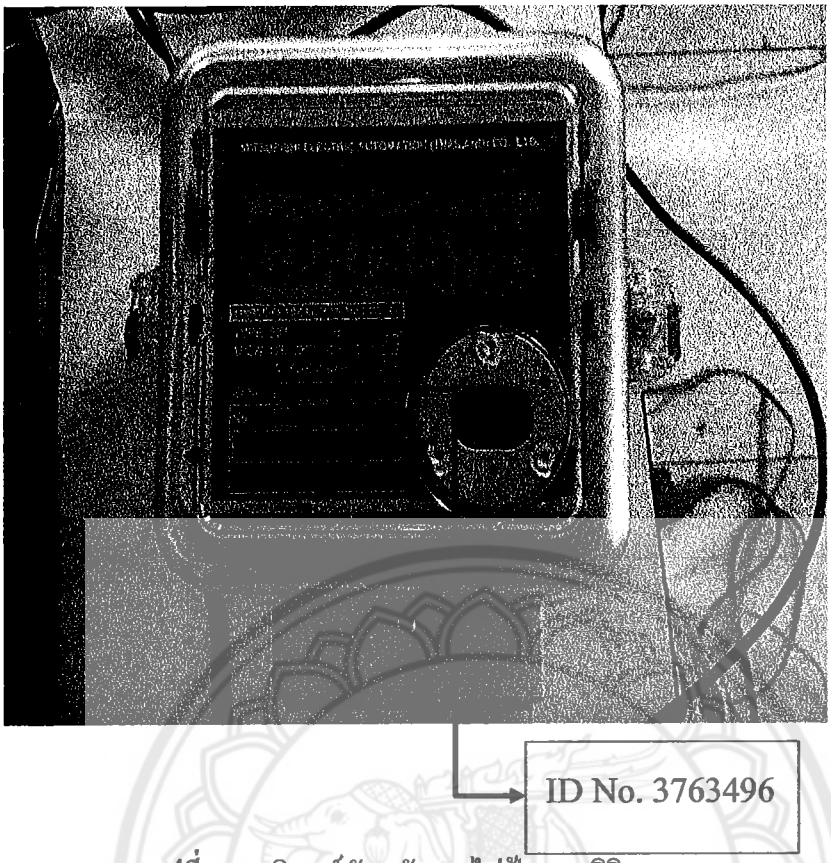
โปรโตคอลของมิเตอร์เป็นเหมือนภาษาที่ใช้ในการเชื่อมต่อและส่งข้อมูลได้อย่างถูกต้อง สำหรับโปรโตคอลของมิเตอร์จะมีทั้งหมด 51 ไบต์ เราจะมาหาความหมายของโปรโตคอลแต่ละตัวว่าจะแปลงเป็นเลข ฐาน 16 ได้อย่างไร (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [3])



รูปที่ 2.16 การรับส่งข้อมูลของ RS-485 ของโปรโตคอลมิเตอร์

หมายเลขที่ 1 : คือ 0x3A เป็นเลขฐาน 16 ที่ถอดรหัสโปโตคอลออกมาจาก (:) ตามตารางที่ 2.1

หมายเลขที่ 2 Device Address มีขนาดข้อมูล 1 ไบต์ หรือ 8 บิต รูปแบบของข้อมูล Binary เป็น เลขที่อยู่ของมิเตอร์ คือ 0x60 เป็นฐาน 16 ที่ถอดรหัสโปรโตคอลจากรหัสเครื่องมิเตอร์คือ ID No. 3763496 ดังรูปที่ 2.17 โดยดูจากเลขหลักที่ 5 6 7 ของหมายเลขมิเตอร์ ถ้าหมายเลขของมิเตอร์หลักที่ 5 เป็นเลขคี่ RS-485 Address จะเป็น 196 แต่ถ้าหมายเลขมิเตอร์หลักที่ 5 เป็นเลขคู่ RS-485 Address จะเป็น 96 คือตัดเลขคู่ตัวนั้นทิ้ง แล้วนำเลข RS-485 Address มาแปลงเป็นฐาน 16 จะเป็น 60



รูปที่ 2.17 มิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

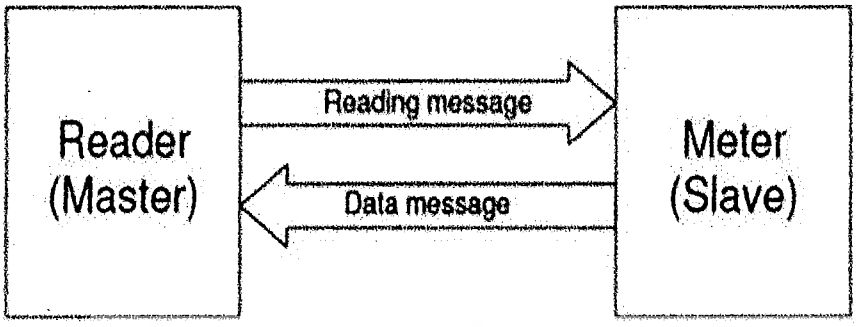
หมายเลขที่ 3 Meath Protocol คือข้อมูลโปรโตคอลของมิเตอร์ส่วนของการแสดงค่าพลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า เพื่อวัดค่าพลังงานงานไฟฟ้า(สามารถดูรายละเอียดเพิ่มเติมได้ที่ [3])

หมายเลขที่ 4 Padding เมื่อหาโปรโตคอลได้ครบทุกตัวแล้ว Padding จะเป็น ไบต์ที่แทนด้วย 0x23 เป็นเลขฐาน 16 เพื่อใส่ใช้ครบ 51 ไบต์ ตามตารางรูปที่ 2.17

หมายเลยที่ 5 CRC ย่อมาจากคำ cyclic redundancy check หมายถึง เทคนิคการตรวจสอบหาข้อผิดพลาดในการบันทึกหรือถ่ายทอดข้อมูล มักจะใช้วิธีการวนทำซ้ำ เพื่อทบทวนดูความแม่นตรง ถ้ามีที่ผิด การทำสองครั้งย่อมจะให้ผลไม่ตรงกันเมื่อถอดรหัสโปโตคอลได้ทั้งหมด 48 ไบต์แล้ว จะใช้ไบต์ที่2-48 มาคำนวณในโปรแกรม ในเว็บไซต์ (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [10]) ดูตัวอย่างการคำนวณได้ในหัวข้อถัดไป

หมายเลขที่ 6 ETX จะแทนด้วยเลขฐาน16 คือ 0x03 เป็นการถอดรหัสจากโปรโตคอลของเครื่องมิเตอร์ ตามตารางที่ 2.1

จากรูปที่ 2.18 คอมพิวเตอร์คือ (Master) เป็นการเชื่อมต่อข้อมูลและส่งไปที่ให้ (Slave) คือมิเตอร์และ (Slave) สามารถมีหลายๆตัวเชื่อมต่อกันเพื่อส่งข้อมูลให้กับ (Master)

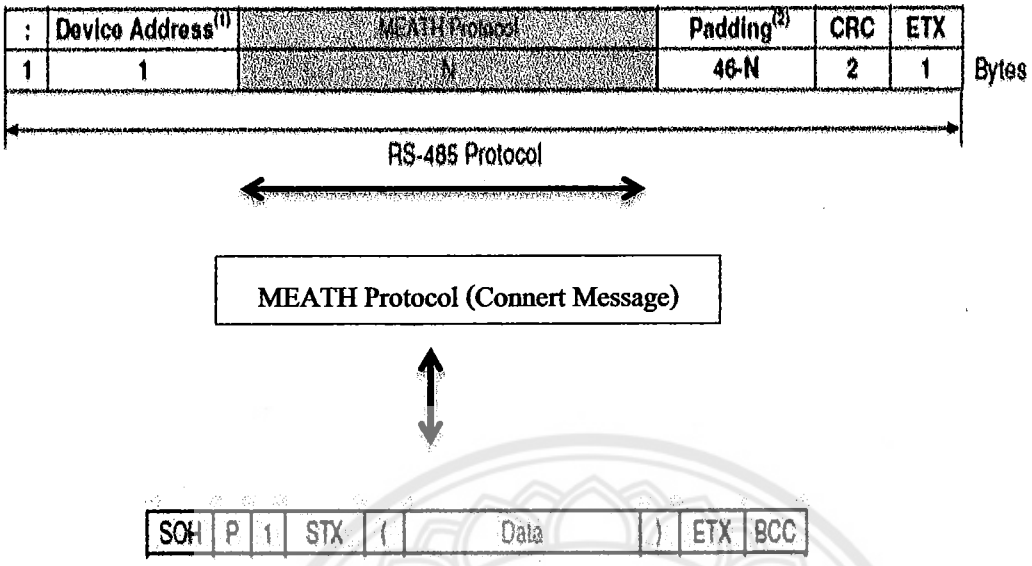


รูปที่ 2.18 หลักการรับส่งข้อมูลระหว่างมิเตอร์

ตารางที่ 2.1 ตารางการแปลง ASCII เลขฐานสิบ เลขฐานแปด และไบนารี (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [9])

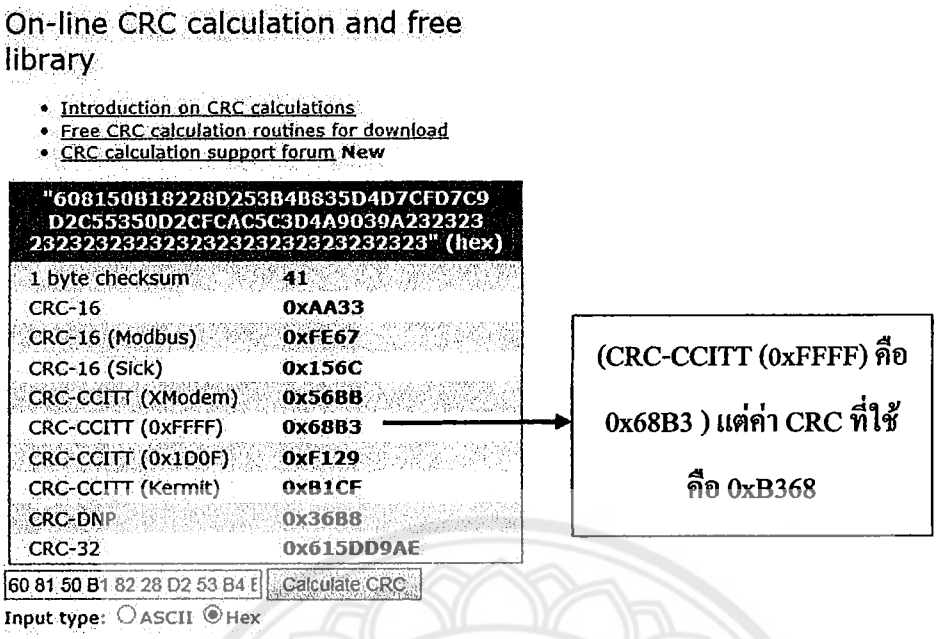
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

➢ 1. โปรโตคอลที่เชื่อมต่อสื่อสาร (Connect Message)



รูปที่ 2.19 โครงสร้างรหัสโปรโตคอลมิเตอร์ส่วนของการเชื่อมต่อสื่อสาร

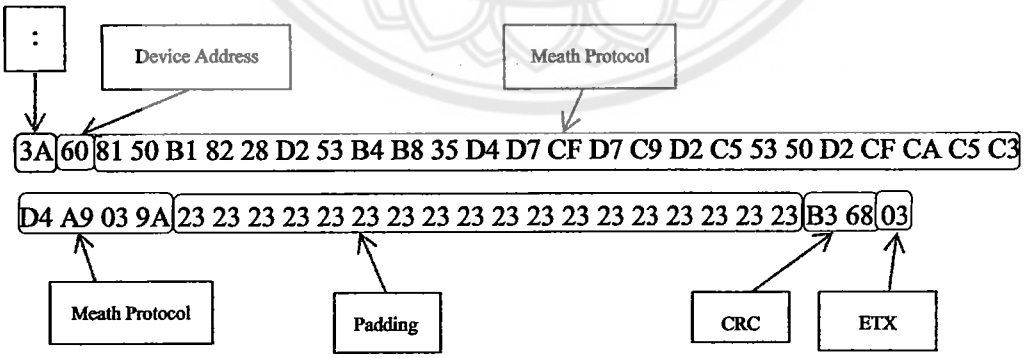
การถอดรหัสโปรโตคอลมิเตอร์ส่วน MEATH Protocol (Connert Message) ในส่วนการเชื่อมต่อสื่อสารกับมิเตอร์จะเป็นโปรโตคอลที่แนบมากับ เอกสารแนบที่ภาคผนวก ค. คือไบต์ที่ 2-30 คือ81 50 B1 82 28 D2 53 B4 B8 35 D4 D7 CF D7 C9 D2 C5 53 50 D2 CF CA C5 C3 D4 A9 03 9A (สามารถดูรายละเอียดเพิ่มเติมได้ที่[3]) เมื่อถอดรหัส MEATH Protocol (Connert Message) ได้ครบแล้ว ต่อไปจะเป็นการคำนวณ CRC ดังรูปที่ 2.20 เลือก Input type : Hex แล้วนำไบต์ที่ 2-48 มาใส่ในช่อง Calculate CRC จะปรากฏค่าCRC แบบกลับไบต์ ที่บรรทัด (CRC-CCITT (0xFFFF) คือ 0x68B3) แต่ค่า CRC ที่ใช้คือ 0xB368



รูปที่ 2.20 โปรแกรมคำนวณ CRC ของการส่งโปรโตคอลพลังงานไฟฟ้าจากเว็บไซต์(สามารถดูข้อมูลเพิ่มเติมได้ที่ [10])

จะได้รหัสโปรโตคอลทั้งหมด 51 ไบต์ ดังต่อไปนี้ รหัส51 ไบต์นี้จะเป็นรหัสที่ส่งไปหามิเตอร์เพื่อเชื่อมต่อให้คอมพิวเตอร์ได้และให้มิเตอร์ส่งข้อมูลกลับมาดังตัวอย่างต่อไปนี้

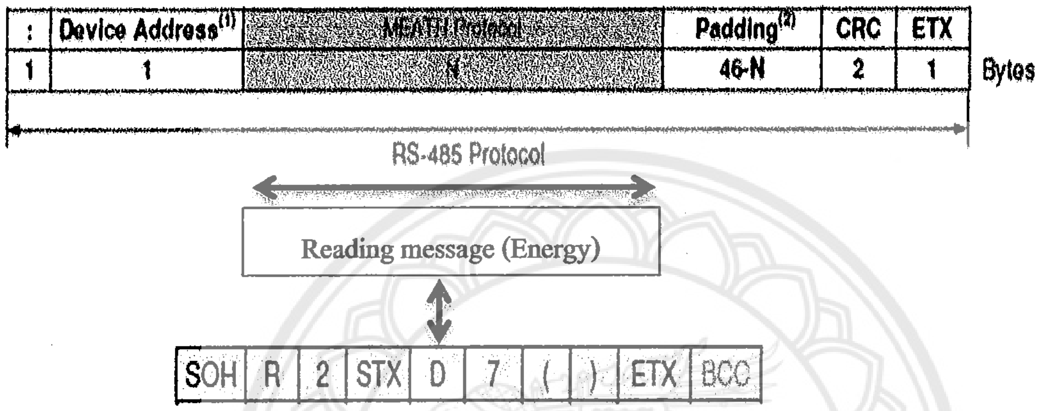
Send : Reading message เป็นรหัสโปรโตคอลที่ส่งไปให้มิเตอร์



Receive : Data message เป็นรหัสโปรโตคอลที่มิเตอร์ส่งข้อมูลกับมา

3A 60 06 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 7B E9 03

➢ 2. โปรโตคอลแสดงค่าพลังงานไฟฟ้า (W/h)



รูปที่ 2.21 โครงสร้างรหัสโปรโตคอลส่วนของการส่งข้อมูลพลังงานไฟฟ้าไปถามมิเตอร์

1. [SOH] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 1 หรือ 01 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 000 0001 แล้วนำมากลับบิตเป็น 1000 000 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 1000 000 เป็น 1000 0001 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0x81
2. [R] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 50 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 101 0100 แล้วนำมากลับบิตเป็น 0100 101 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 0100 101 เป็น 0100 1011 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0xD2
3. [2] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 32 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 110 0100 แล้วนำมากลับบิตเป็น 0100 110 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 0100 110 เป็น 0100 1101 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0xB2

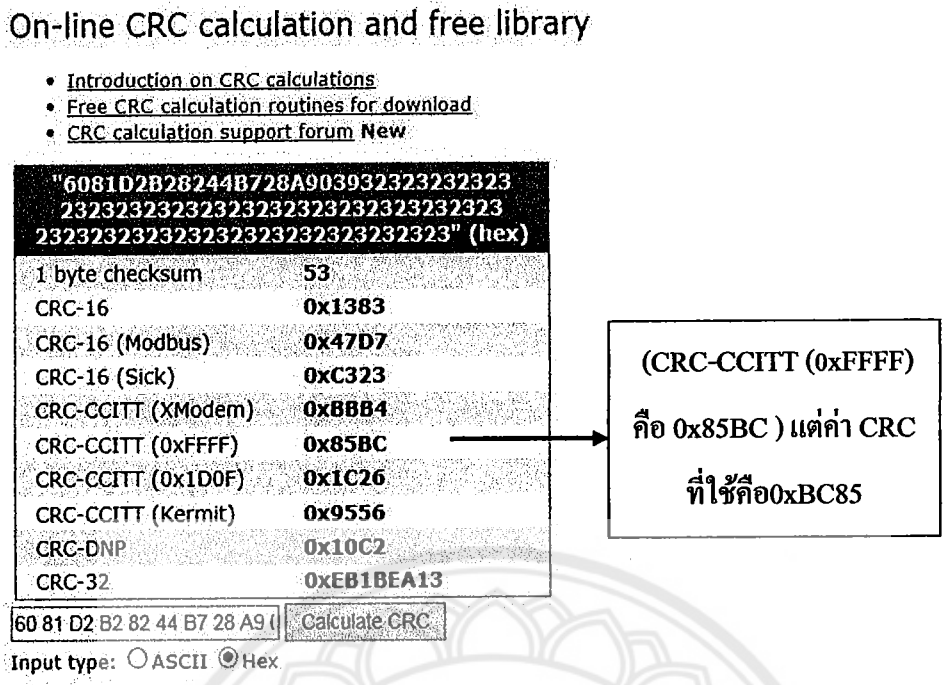
4. [STX] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 2 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 000 0100 แล้วนำมากลับบิตเป็น 0100 000 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 0100 000 เป็น 0100 0001 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0x82
5. [D] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 44 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 001 0010 แล้วนำมากลับบิตเป็น 0010 001 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 0010 001 เป็น 0010 0010 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0x44
6. [7] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 37 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 110 1110 แล้วนำมากลับบิตเป็น 1110 110 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 1110 110 เป็น 1110 1101 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0xB7
7. [( ] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 28 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 010 0001 แล้วนำมากลับบิตเป็น 0001 010 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 0001 010 เป็น 0001 0100 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0x28
8. [ )] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 29 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 010 1001 แล้วนำมากลับบิตเป็น 1001 010 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 1001 010 เป็น 1001 0101 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0xB7
9. [ETX] ดูจากตารางที่ 2.1 ในช่องแปลง ASCII เป็นเลขฐานสิบหก จะพบว่าได้ค่าคือ 3 นำมาแปลงเป็นเลขฐานสอง 7 บิต จะเป็น 000 1100 แล้วนำมากลับบิตเป็น 1100 000 นำมาใส่ตารางที่ 2.2 ในช่อง 0-6 และ ช่อง P จะทำให้เป็นบิตคู่ จาก 1100 000 เป็น 1100 0000 เมื่อได้ครบ 8 บิตจะแปลงเลขฐานสองเป็นเลขฐานสิบหก เป็น 0x93

การถอดรหัสโปรโตคอลมิเตอร์ส่วน MEATH Protocol (Reading message (Power))จะเป็นการถอดรหัสตัวอักษรต่างๆเพื่อคำนวณหาค่า BCC Data ตารางที่ 2.2

ตารางที่ 2.2 ตารางถอดรหัสโปรโตคอลส่วนของ Reading message (Energe)

0	1	2	3	4	5	6	P	P =	parity
1	0	0	0	0	0	0	1	SOH	0x81
0	1	0	0	1	0	1	1	R	0xD2
0	1	0	0	1	1	0	1	2	0xB2
0	1	0	0	0	0	0	1	STX	0x82
0	0	1	0	0	0	1	0	D	0x44
1	1	1	0	1	1	0	1	7	0xB7
0	0	0	1	0	1	0	0	(	0x28
1	0	0	1	0	1	0	1	)	0xA9
1	1	0	0	0	0	0	0	ETX	0x03
1	1	0	0	1	0	0	1	BCC	0x93

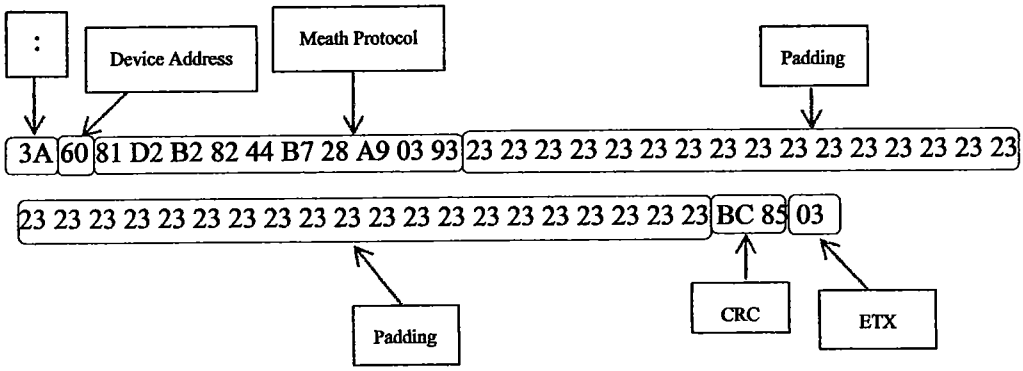
เมื่อคำนวณการถอดรหัสเพื่อหาค่า BCC ได้แล้ว จะมาคำนวณค่า CRC ดังรูปที่ 2.22 เลือก Input type : Hex แล้วนำไบต์ที่ 2-48 มาใส่ในช่อง Calculate CRC จะปรากฏค่าCRC แบบกลับไบต์ที่บรรทัด (CRC-CCITT (0xFFFF) คือ 0x85BC) แต่ค่า CRC ที่ใช้คือ 0xBC85

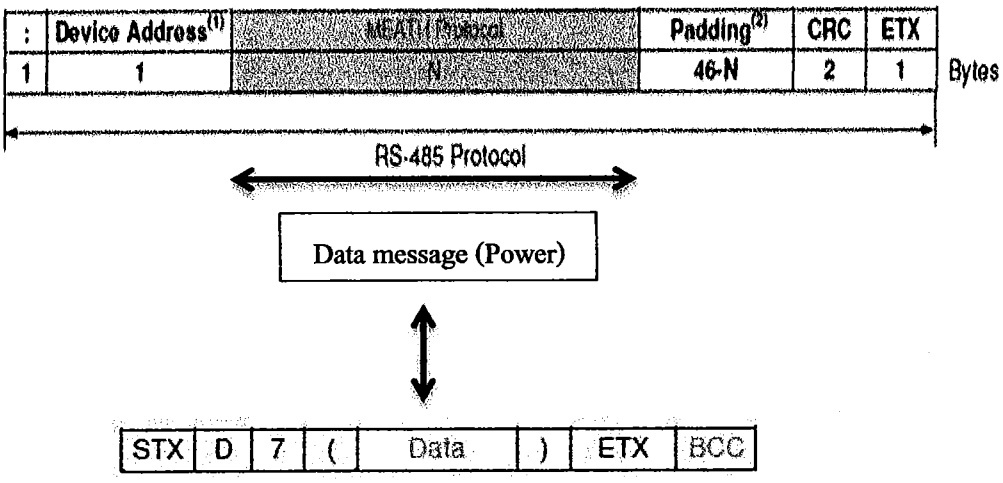


รูปที่ 2.22 โปรแกรมคำนวณ CRCของการส่งโปรโตคอลพลังงานไฟฟ้า จากเว็บไซต์ (สามารถดูข้อมูลเพิ่มเติมได้ที่ [10])

จะได้รหัสโปรโตคอลทั้งหมด 51 ไบต์ ดังต่อไปนี้ รหัส51 ไบต์นี้จะเป็นรหัสที่ส่งไปหามิเตอร์เพื่อเชื่อมต่อให้คอมพิวเตอร์ได้และให้มิเตอร์ส่งข้อมูลกลับมาดังตัวอย่างต่อไปนี้

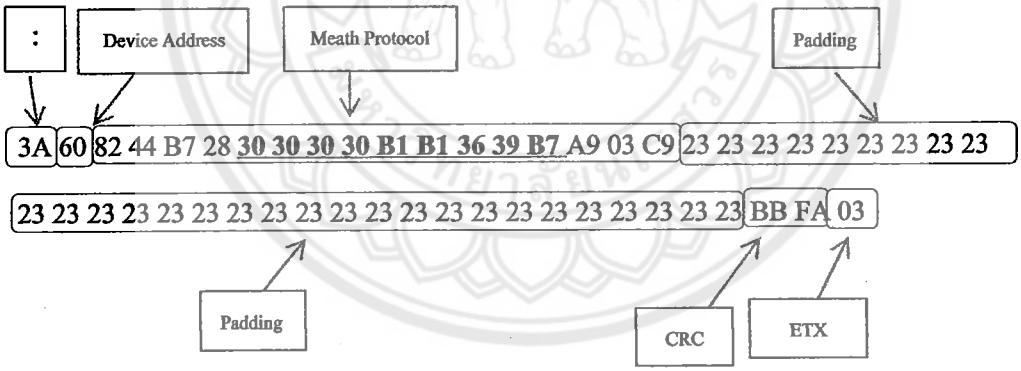
Send : Reading message จะเป็นรูปแบบถอดรหัสการส่งโปรโตคอลไปให้มิเตอร์เพื่อให้มิเตอร์ส่งข้อมูลกลับมาดังแสดงด้านล่าง





รูปที่ 2.23 โครงสร้าง รหัสโปรโตคอลมิเตอร์ส่วนของมิเตอร์ส่งข้อมูลพลังงานไฟฟ้าตอบกลับมา

Receive : Data message เป็นรูปแบบการถอดรหัสโปรโตคอลที่มิเตอร์ส่งข้อมูลกับมาดังแสดงด้านล่าง

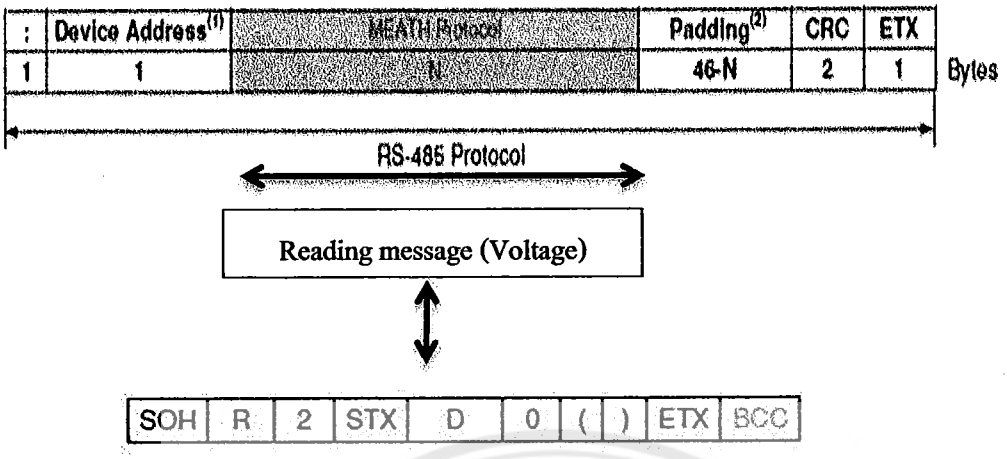


หลักการอ่านค่าพลังงานไฟฟ้าที่มิเตอร์ส่งข้อมูลกลับมาเป็นข้อมูลดิบ 51 ไบต์ ไบต์ที่7-15 ตัวเลขข้างหลังจะเป็นเลขที่บอกค่าพลังงานหน่วยเป็น Wh

Reading message : Energy = 000011697 Wh

=11.697 kWh

➢ 3. โปรโตคอลแสดงค่าแรงดันไฟฟ้า (V)



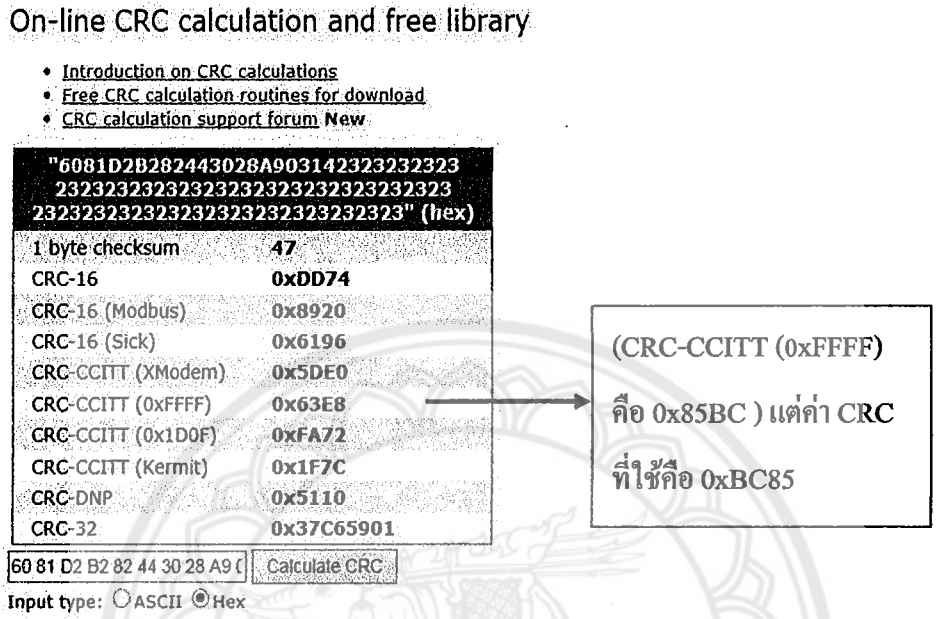
รูปที่ 2.24 โครงสร้างรหัสโปรโตคอลส่วนของการส่งข้อมูลแรงดันไฟฟ้าไปถามมิเตอร์

การถอดรหัสโปรโตคอลมิเตอร์ส่วน MEATH Protocol (Reading message (Voltage) จะเป็นการถอดรหัสตัวอักษรต่างๆเพื่อคำนวณหาค่า BCC Data ตารางที่ 2.3

ตารางที่ 2.3 ตารางถอดรหัสโปรโตคอลส่วนของ Reading message (Voltage)

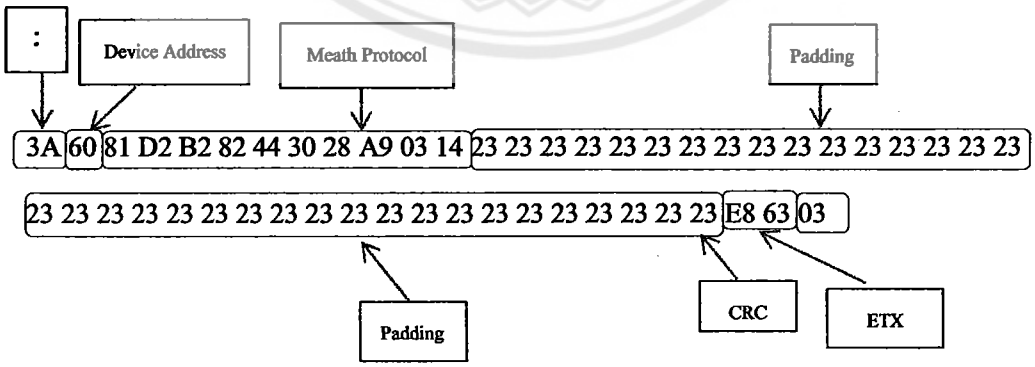
0	1	2	3	4	5	6	P	P =	parity
1	0	0	0	0	0	0	1	SOH	0x81
0	1	0	0	1	0	1	1	R	0xD2
0	1	0	0	1	1	0	1	2	0xB2
0	1	0	0	0	0	0	1	STX	0x82
0	0	1	0	0	0	1	0	D	0x44
0	0	0	0	1	1	0	0	0	0xB7
0	0	0	1	0	1	0	0	(	0x28
1	0	0	1	0	1	0	1	)	0xA9
1	1	0	0	0	0	0	0	ETX	0x03
0	0	1	0	1	0	0	0	BCC	0x14

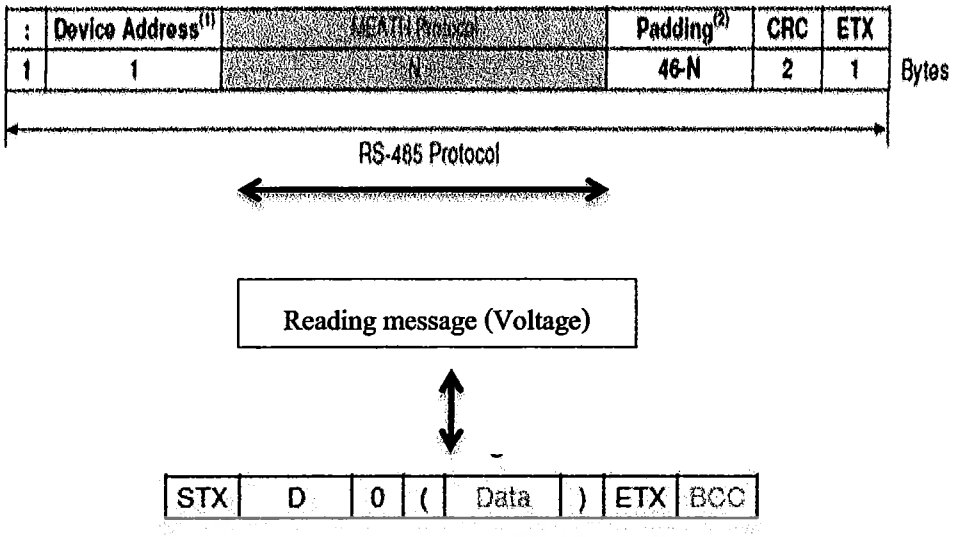
เมื่อคำนวณการถอดรหัสเพื่อหาค่า BCC ได้แล้ว จะมาคำนวณค่า CRC ดังรูปที่ 2.25 เลือก Input type : Hex แล้วนำไบต์ที่ 2-48 มาใส่ในช่อง Calculate CRC จะปรากฏค่าCRC แบบกลับไบต์ที่บรรทัด (CRC-CCITT (0xFFFF) คือ 0x85BC) แต่ค่า CRC ที่ใช้คือ 0xBC85



รูปที่ 2.25 โปรแกรมคำนวณ CRCของการส่งโปรโตคอลแรงดันไฟฟ้า จากเว็บไซต์ (สามารถดูข้อมูลเพิ่มเติมได้ที่ [10]

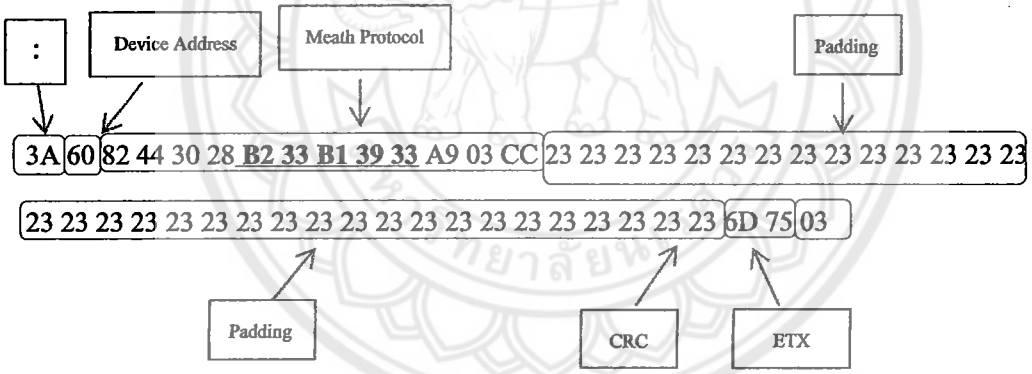
Send : Reading message จะเป็นรูปแบบถอดรหัสการส่งโปรโตคอลไปให้มิเตอร์ เพื่อให้มิเตอร์ส่งข้อมูลกลับมาดังแสดงด้านล่าง





รูปที่ 2.26 โครงสร้าง รหัสโปรโตคอลมิเตอร์ส่วนของมิเตอร์ส่งข้อมูลแรงดันไฟฟ้าตอบกลับมา

Receive : Data message เป็นรูปแบบการถอดรหัสโปรโตคอลที่มิเตอร์ส่งข้อมูลกับมาดังแสดงด้านล่าง

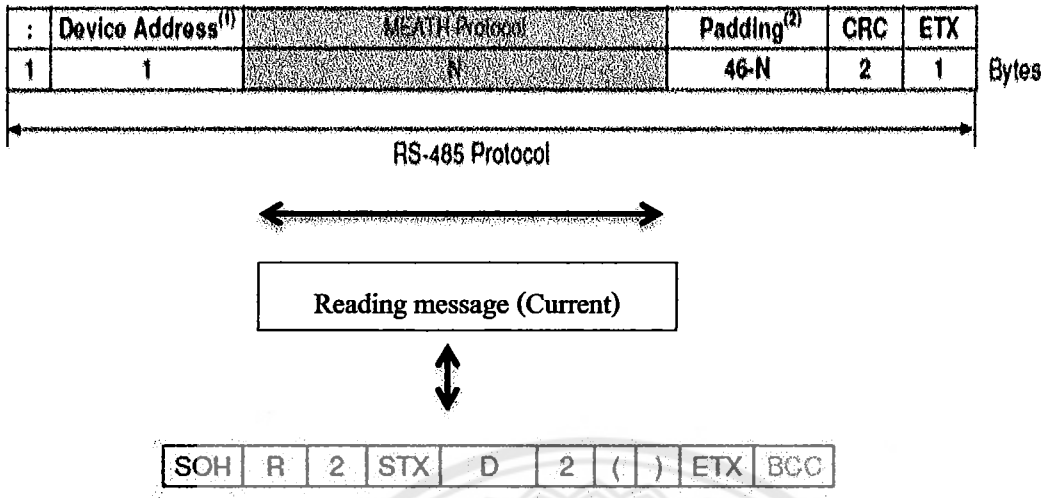


หลักการอ่านค่าแรงดันไฟฟ้าที่มิเตอร์ส่งข้อมูลกลับมาเป็นข้อมูลดิบ 51 ไบต์ ไบต์ที่7-11 ตัวเลขข้างหลังจะเป็นเลขที่บอกค่าแรงดันไฟฟ้าหน่วยเป็น V

Reading message : Voltage = 23193 x 10mV

=231.93 V

➢ 4. โปรโตคอลแสดงค่ากระแสไฟฟ้า (A)



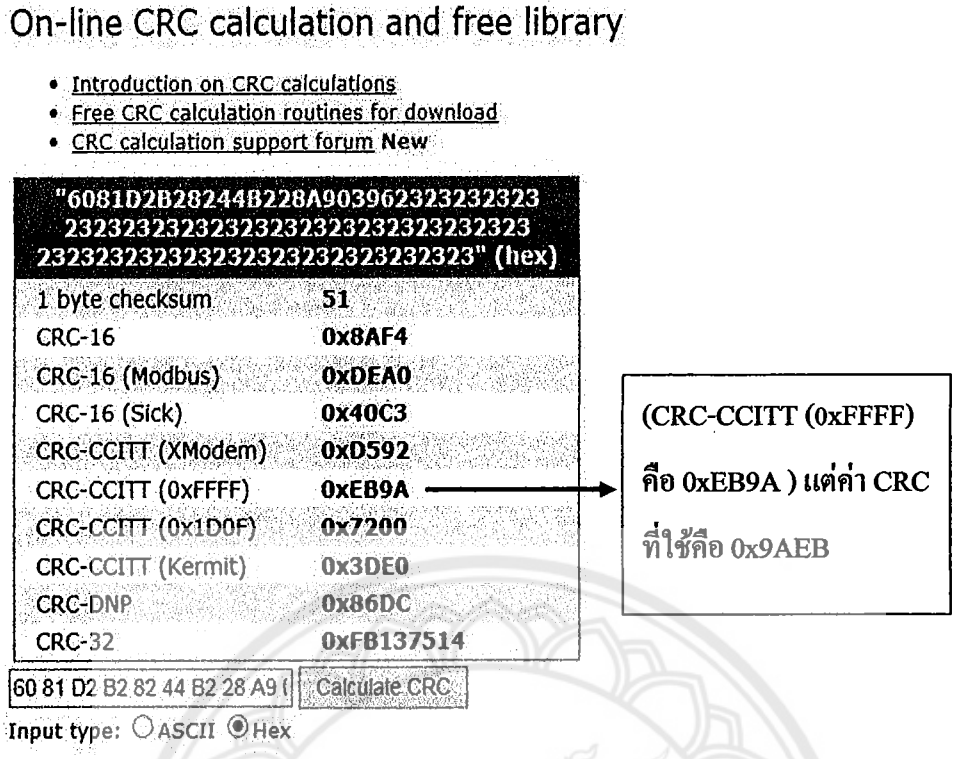
รูปที่ 2.27 โครงสร้างรหัสโปรโตคอลส่วนของการส่งข้อมูลกระแสไฟฟ้าไปถามมิเตอร์

การถอดรหัสโปรโตคอลมิเตอร์ส่วน MEATH Protocol (Reading message (Voltage)) ดังตารางที่ 2.4 นี้จะเป็นการถอดรหัสตัวอักษรต่างๆเพื่อคำนวณหาค่า BCC Data

ตารางที่ 2.4 ตารางถอดรหัสโปรโตคอลส่วนของ Reading message (Current)

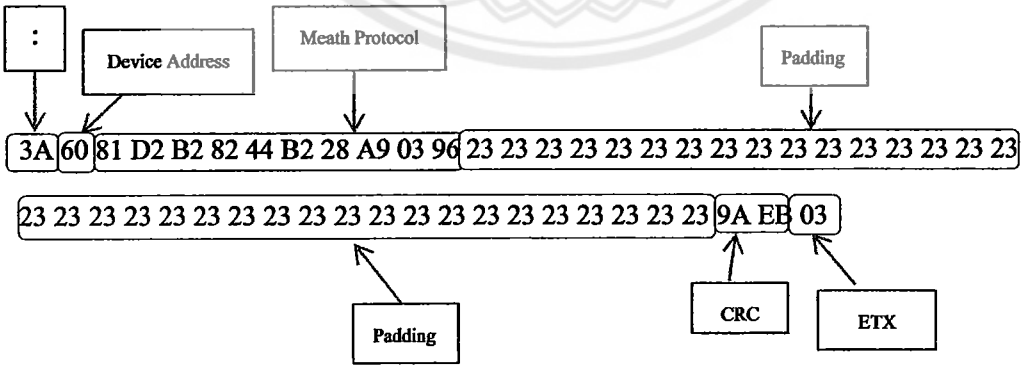
0	1	2	3	4	5	6	P	P =	parity
1	0	0	0	0	0	0	1	SOH	0x81
0	1	0	0	1	0	1	1	R	0xD2
0	1	0	0	1	1	0	1	2	0xB2
0	1	0	0	0	0	0	1	STX	0x82
0	0	1	0	0	0	1	0	D	0x44
0	1	0	0	1	1	0	1	2	0xB7
0	0	0	1	0	1	0	0	(	0x28
1	0	0	1	0	1	0	1	)	0xA9
1	1	0	0	0	0	0	0	ETX	0x03
0	1	1	0	1	0	0	1	BCC	0x96

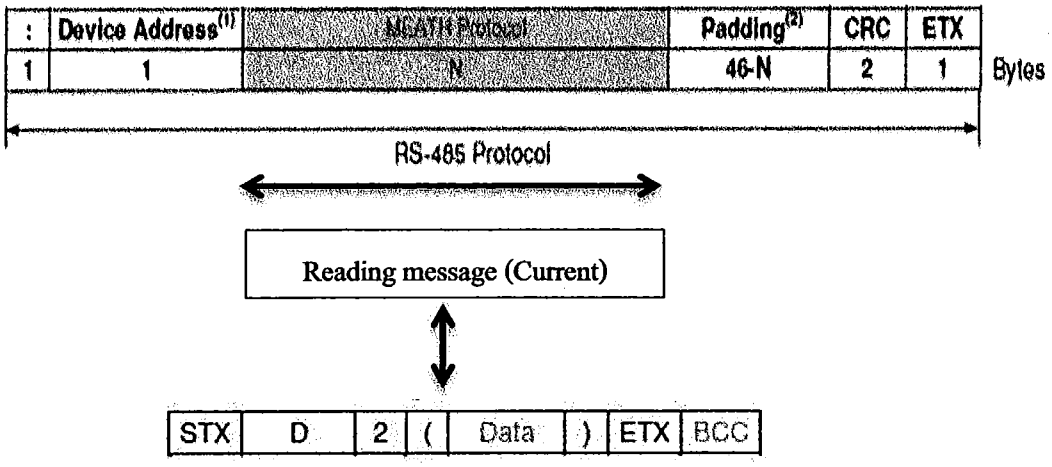
เมื่อคำนวณการถอดรหัสเพื่อหาค่า BCC ได้แล้ว จะมาคำนวณค่า CRC ดังรูปที่ 2.28 เลือก Input type : Hex แล้วนำไบต์ที่ 2-48 มาใส่ในช่อง Calculate CRC จะปรากฏค่าCRC แบบกลับไบต์ที่บรรทัด (CRC-CCITT (0xFFFF) คือ 0xEB9A) แต่ค่า CRC ที่ใช้คือ 0x9AEB



รูปที่ 2.28 โปรแกรมคำนวณ CRCของการส่งโปรโตคอลกระแสไฟฟ้า จากเว็บไซต์ (สามารถดูข้อมูลเพิ่มเติมได้ที่ [10])

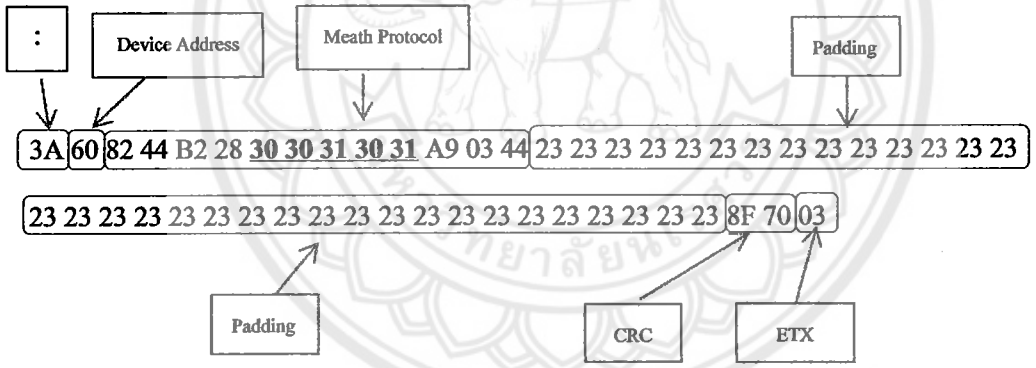
Send : Reading message จะเป็นรูปแบบถอดรหัสการส่งโปรโตคอลไปให้มิเตอร์ เพื่อให้มิเตอร์ส่งข้อมูลกลับมาดังแสดงด้านล่าง





รูปที่ 2.29 โครงสร้าง รหัสโปรโตคอลมิเตอร์ส่วนของมิเตอร์ส่งข้อมูลกระแสไฟฟ้าตอบกลับมา

Receive : Data message เป็นรูปแบบการถอดรหัสโปรโตคอลที่มิเตอร์ส่งข้อมูลกับมาดังแสดงด้านล่าง



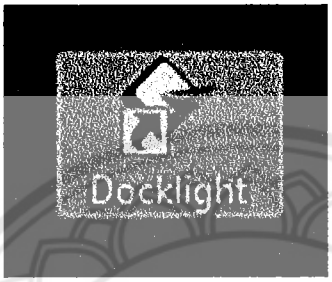
หลักการอ่านค่ากระแสไฟฟ้าที่มิเตอร์ส่งข้อมูลกลับมาเป็นข้อมูลดิบ 51 ไบต์ ไบต์ที่7-11 ตัวเลขข้างหลังจะเป็นเลขที่บอกค่ากระแสไฟฟ้าหน่วยเป็น A

Reading message : Current = 00101 x 10mA

=001.01 A

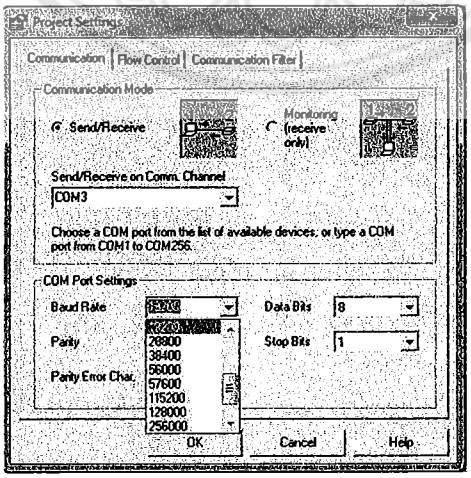
2.5 โปรแกรมที่ใช้ทดสอบค่าโปรโตคอลของมิเตอร์พลังงานไฟฟ้าแบบดิจิตอล

Docklight เป็นเครื่องมือการทดสอบแบบอัตโนมัติสำหรับโปรโตคอลการสื่อสารแบบอนุกรมผ่าน COM (RS485) Docklight เป็นฟังก์ชัน การส่งข้อมูลที่กำหนดไว้ล่วงหน้าลำดับการตรวจหาลำดับเฉพาะภายในกระแสข้อมูลที่เข้ามา และฝังไว้ในรหัสการทดสอบ Docklight ให้ทั้งความยืดหยุ่นและความเรียบง่าย (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [4])



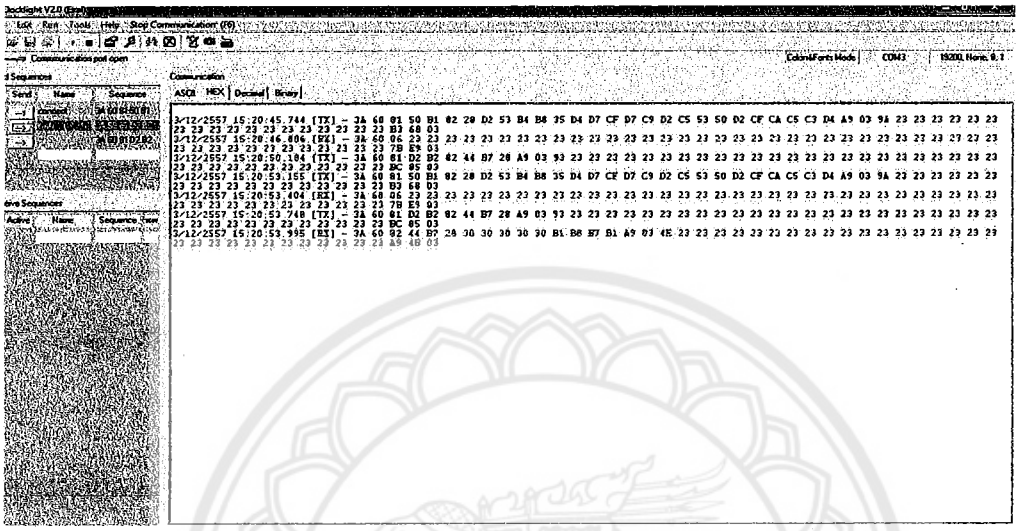
รูปที่ 2.30 โปรแกรม Docklight เป็นโปรแกรมที่ใช้ทดสอบโปรโตคอล

สคริปต์การกำหนดค่าอุปกรณ์อัตโนมัติ ทดสอบโปรโตคอลที่มีการคำนวณค่าตรวจสอบอัตโนมัติ Docklight สคริปต์เริ่มต้น เริ่มต้นโดยอัตโนมัติพอร์ต COM เข้าสู่ระบบงานที่เริ่มต้นคอมพิวเตอร์ Docklight จะทำการอ่านข้อมูลที่มิเตอร์ส่งผ่าน RS-485 ส่งมาให้คอมพิวเตอร์ จะแสดงผลออกมาเป็นตัวเลข 51 ไบต์



รูปที่ 2.31 การตั้งค่า Baud Rate

เมื่อเปิดโปรแกรม Docklight ขึ้นมาจะต้องตั้งค่า Baud Rat จากรูปที่ 2.31 แสดงการทำงานของโปรแกรม Docklight ส่วนของการตั้งค่า Baud Rat rate คือการสื่อสารหรือการรับส่งข้อมูลในระบบดิจิตอล การสื่อสารกับมิเตอร์ไฟฟ้าแบบดิจิตอลมิซูบิชิ SX-1 จะใช้ Baud rate ที่19200 เพื่อตั้งค่าให้คอมพิวเตอร์อ่านค่าจากมิเตอร์ได้



รูปที่ 2.32 หน้าต่างแสดงค่าข้อมูลโปรโตคอล

จากรูปที่ 2.32 เป็นหน้าต่างแสดงค่าข้อมูลของโปรโตคอลจากมิเตอร์เมื่อตั้งค่า Baud Rat แล้วเราจะใส่ข้อมูลโปรโตคอลของมิเตอร์เข้าไปเพื่อให้ โปรแกรม Docklight เป็นตัวกลางในการให้มิเตอร์ส่งข้อมูลกับมาเป็นข้อมูลดิบ 51 ไบต์

2.6 ไพธอน

ไพธอนเป็นภาษาคอมพิวเตอร์ระดับสูงมากที่ทำงานแบบแปลทีละคำสั่งโดยมีจุดเด่นจากการมีชนิดข้อมูลระดับสูง เช่น list และ dictionary เป็นต้นการใช้ย่อหน้าในการจำแนกระดับและขอบเขตของการประมวลผล และการที่ไม่จำเป็นต้องประกาศชนิดของตัวแปรซอฟต์แวร์ ทำหน้าที่เป็นตัวแปลภาษา ไพธอน ที่ใช้อย่างแพร่หลายในปัจจุบันได้แก่ Classic Python หรือ Python ซึ่งพัฒนาขึ้นด้วยภาษาซี

การพัฒนาซอฟต์แวร์ด้วยภาษาไพธอนจำเป็นต้องใช้โปรแกรมสำหรับแก้ไขไฟล์ตัวอักษรเพื่อเขียนสคริปต์ในการประมวลผลลงในไฟล์จากนั้นจึงใช้โปรแกรมตัวแปลภาษากล่าวคือไพธอนในลินุกส์หรือpython.exe ในวินโดว์เพื่อประมวลผลแต่ละคำสั่งในไฟล์นั้นการเรียกใช้โปรแกรมตัวแปลภาษาโดยไม่ระบุไฟล์สคริปต์จะทำให้เข้าสู่สถานการณ์แปลแบบตอบโต้ซึ่งโปรแกรมจะแสดงเครื่องหมายรอการสั่งการจากผู้ใช้เมื่อผู้ใช้ป้อนคำสั่งโปรแกรมก็จะประมวลผลเฉพาะคำสั่งนั้นและแสดงผลออกมาทางหน้าจอ

ข้อดีของภาษาไพธอนที่แตกต่างจากภาษาอื่น

1. ง่ายต่อการเรียนรู้ โดยภาษา Python มีโครงสร้างของภาษาไม่ซับซ้อนเข้าใจง่าย ซึ่งโครงสร้าง ภาษา Python จะคล้ายกับภาษา C มาก เพราะภาษา Python สร้างขึ้นมาโดยใช้ภาษา C ทำให้ผู้ที่คุ้นเคยภาษา C อยู่แล้วใช้งานภาษา Python ได้ไม่ยาก นอกจากนี้โดยตัวภาษาเองมีความยืดหยุ่นสูงทำให้การจัดการกับงานด้านข้อความและTextFileได้เป็นอย่างดี
2. ใช้ได้หลายแพลตฟอร์ม ในช่วงแรกภาษา Python ถูกออกแบบใช้งานกับระบบ Unix อยู่ก็จริง แต่ในปัจจุบันได้มีการพัฒนาตัวแปลภาษาPythonให้สามารถใช้กับระบบปฏิบัติการอื่นๆอาทิเช่นLinux,Windows 95/98/ME,WindowsNT,Windows2000,OS/2
3. ภาษา Python ถูกสร้างขึ้นโดยได้รวบรวมเอาส่วนดีของภาษาต่างๆ เข้ามาไว้ด้วยกัน อาทิเช่น ภาษา C, C++,Java,Perl
4. ภาษา Python เป็นภาษาประเภท Server side Script คือการทำงานของภาษา Python จะทำงานด้านฝั่ง Serverแล้วส่งผลลัพธ์กลับมายังClientทำให้มีความปลอดภัยสูง
5. ใช้พัฒนา Web Service โดยที่ภาษา Python สามารถนำมาพัฒนาเว็บเซอร์วิส รวมทั้งใช้บริหารการสร้างเว็บไซต์สำเร็จรูปที่เรียกว่า Content Management Framework (CMF) ตัวอย่าง CMF ที่มีชื่อเสียงมากและเบื้องหลังทำงานด้วย python คือ Plone (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [5])

สาเหตุที่เลือกใช้ภาษาไพธอนใช้เวลาในการเขียนและคอมไพล์ไม่มาก ทำให้เหมาะกับงานด้านการดูแลระบบ (System administration) เป็นอย่างยิ่ง ภาษาไพทอนจะคล้ายกับภาษา C มาก โดยจะมีโครงสร้างที่ไม่ซับซ้อน มีไวยากรณ์ที่ได้กำจัดการใช้สัญลักษณ์ที่ใช้ในการแบ่งบล็อกของโปรแกรมและใช้การย่อหน้าแทน ทำให้สามารถอ่านโปรแกรมที่เขียนได้ง่าย

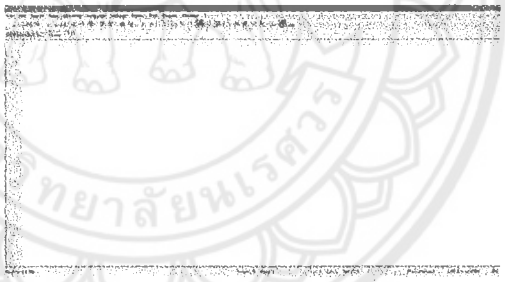
บทที่ 3
วิธีการดำเนินงาน

บทนี้จะกล่าวถึงโปรแกรมที่ใช้ในการเขียนภาษาในการเขียนและรายละเอียดของโครงสร้างโปรแกรมในส่วนต่างๆ เช่น ส่วนของการรับส่งข้อมูลของมิเตอร์ ส่วนการแสดงพลังงานไฟฟ้า แรงดันไฟฟ้า กระแสะไฟฟ้าและการบันทึกผลลงในไฟล์

3.1 โครงสร้างโปรแกรมที่พัฒนา

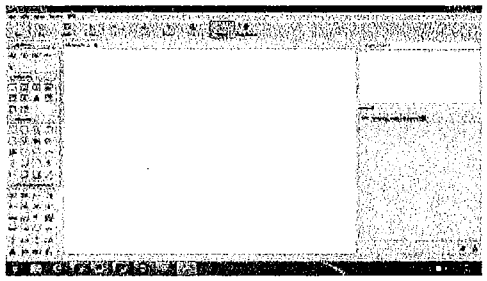
โปรแกรมที่ใช้คือโปรแกรม Notepad ++ ดังรูปที่ 3.1 โดยใช้ภาษาไพธอนในการพัฒนาเนื่องจาก+โปรแกรมในคอมพิวเตอร์กับมิเตอร์ไฟฟ้าแบบดิจิตอล แล้วใช้โปรแกรม Glade Interface Designer Gtk + 2 (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [7])

ดังรูปที่ 3.2 เป็นโปรแกรมที่ใช้งานร่วมกับ ไพธอน โดย Glade Designer นั้นจะถูกแยกออกเป็นสองส่วนชัดเจน ในส่วนของการออกแบบหน้าต่างแสดงผล และ ส่วนของโค้ด ซึ่งสามารถลิ้งโค้ด มาเขียนบนโปรแกรม Notepad ++ ได้เลย (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [6])



(ก) (ข)

รูปที่ 3.1 โปรแกรมNotpad++โปรแกรมที่ใช้ในการเขียนโปรแกรม [6]



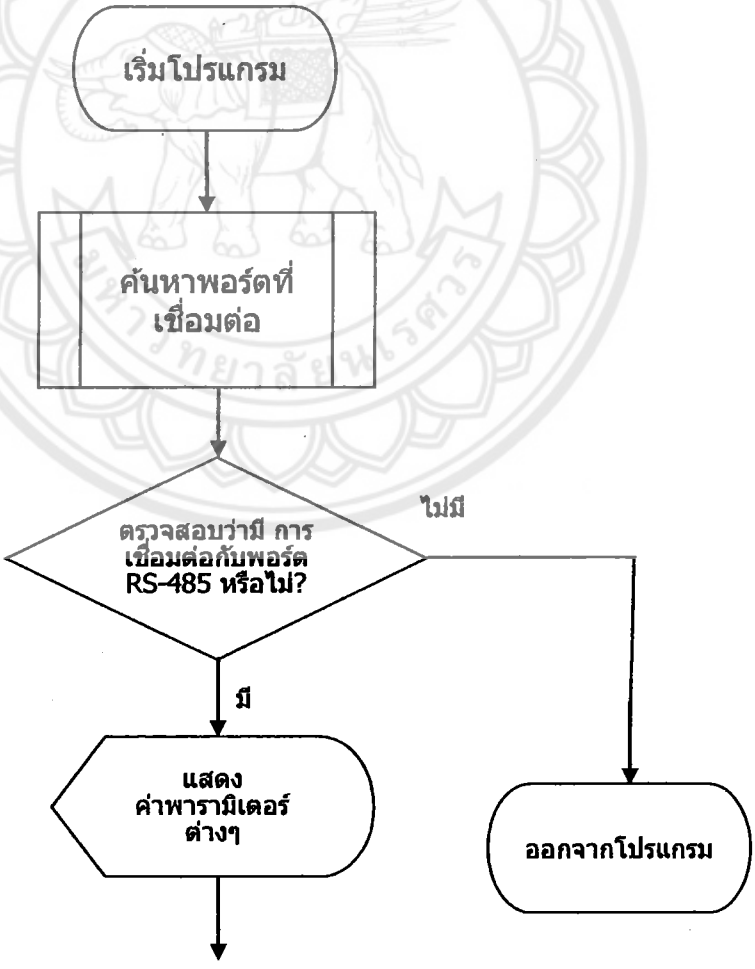
(ก) (ข)

รูปที่ 3.2 โปรแกรมที่ใช้ในการเขียนหน้าต่างแสดงผล(สามารถดูรายละเอียดเพิ่มเติมได้ที่ [7]

3.2 โครงสร้างของโปรแกรมทั้งหมด

การทำงานของโปรแกรมเมื่อเปิดโปรแกรมขึ้นมา ขั้นตอนแรกจะค้นหาพอร์ตที่เชื่อมต่อกับคอมพิวเตอร์ ว่ามีพอร์ตใดมีการเชื่อมหรือทำงานอยู่หรือไม่ ถ้าโปรแกรมค้นหาพอร์ตไม่เจอจะออกจากโปรแกรมทันที แต่เมื่อโปรแกรมพบพอร์ต ที่เชื่อมต่อกับ RS-485 โปรแกรมจะแสดงที่หน้าจอแสดงผลค่าพลังงานไฟฟ้า แรงดันไฟฟ้า สามารถเปลี่ยนแปลงการตั้งค่าพารามิเตอร์ใหม่อีกครั้งได้ โดยกด ปุ่ม Setting เมื่อตั้งค่าเสร็จแล้ว กดปุ่ม save โปรแกรมจะเริ่มทำงาน โดยเรียกฟังก์ชันการแสดงค่า เวลา วัน เดือน ปี ตามเวลาปัจจุบัน คือตามเวลาในเครื่องคอมพิวเตอร์ ต่อมาเรียกฟังก์ชันค่าพารามิเตอร์ต่างๆที่จะให้แสดงค่าบนหน้าจอคือ พลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า เก็บไว ต่อไปเมื่อเรียกค่าที่เก็บไวแล้วจะแสดงค่าที่หน้าจอแสดงผล แล้วบันทึกไวใน ไฟล์ log.txt เมื่อกดปุ่ม Exit จะออกจากโปรแกรมทันที แต่ถ้าไม่กดปุ่ม Exit โปรแกรมจะทำงานวนลูปจากขั้นตอนการแสดงผลและบันทึกค่าไปเรื่อยๆ

ดังรูปที่ 3.3 ถึงรูปที่ 3.4

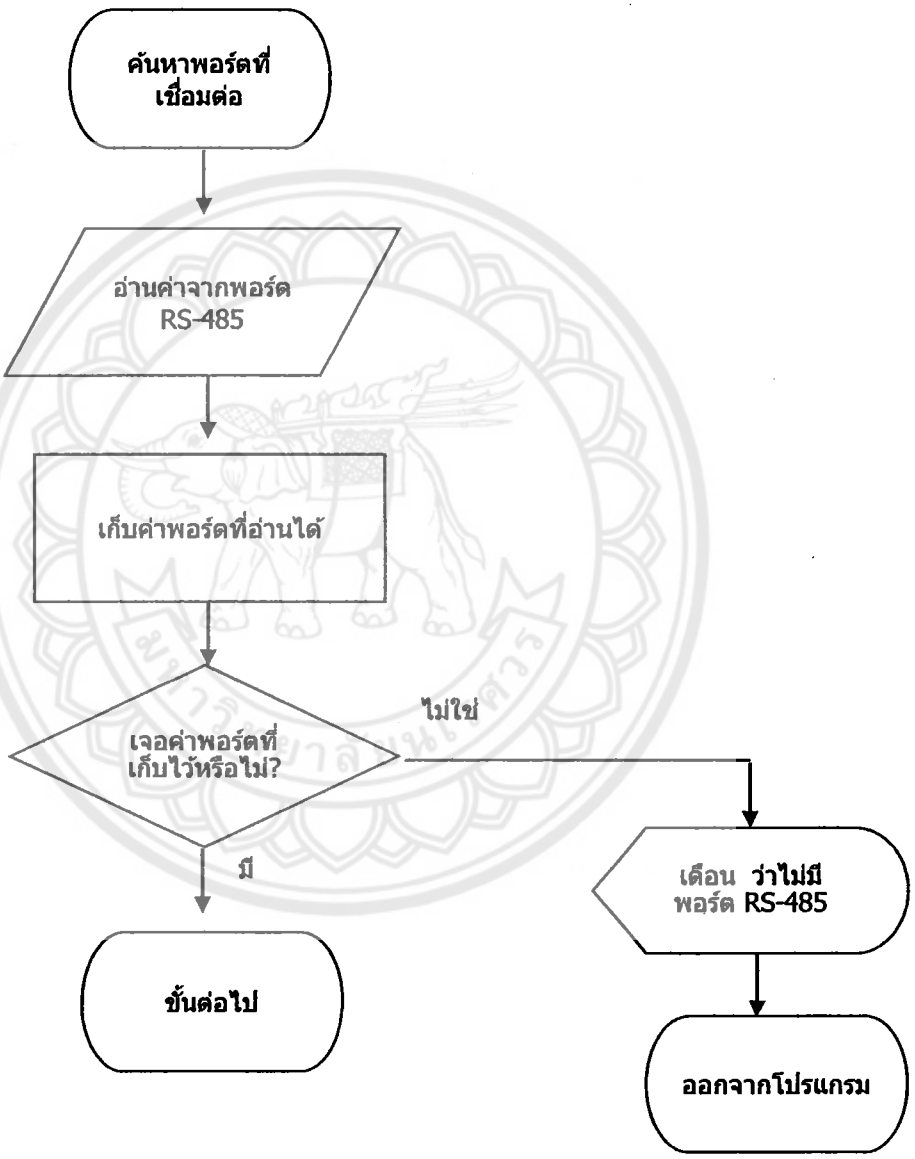


รูปที่ 3.3 โครงสร้างการทำงานของโปรแกรมภาพรวมทั้งหมด

รูปที่ 3.4 (ต่อ)โครงสร้างการทำงานของโปรแกรมภาพรวมทั้งหมด

3.2.1 โครงสร้างการทำงานส่วนของพอร์ตสื่อสารระหว่างคอมพิวเตอร์กับมิเตอร์

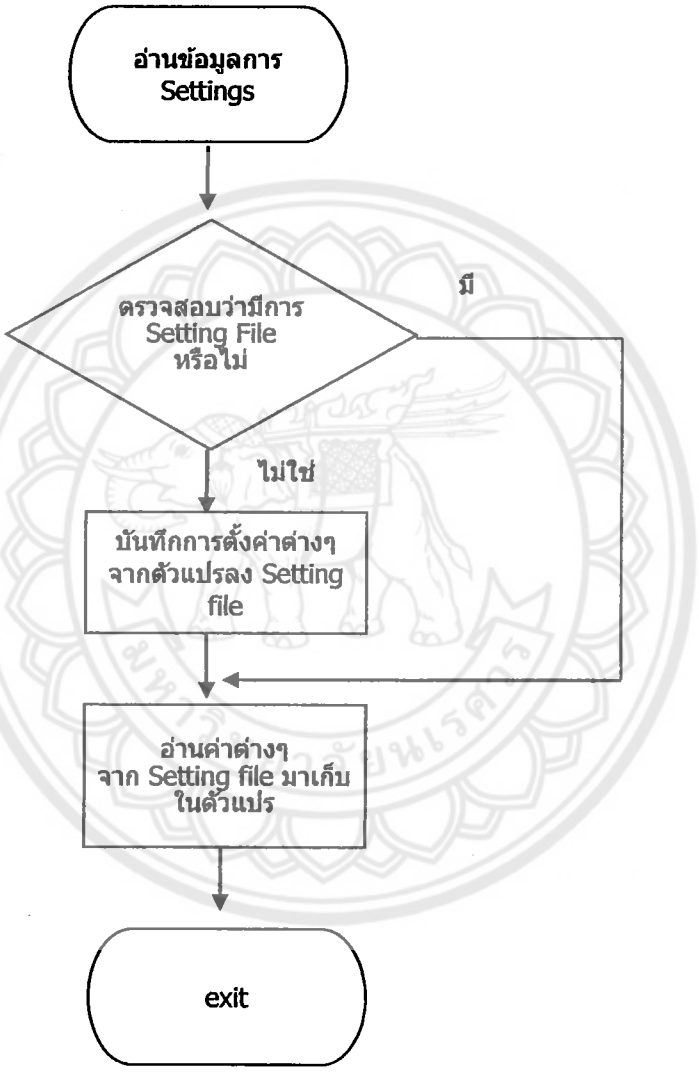
เมื่อทำการเชื่อมต่อ คอมพิวเตอร์ กับมิเตอร์ ผ่าน RS-485 โปรแกรมจะอ่านค่าจากพอร์ต USB ที่เชื่อมRS-485 เพื่อเก็บชื่อพอร์ตไว้ จากนั้นโปรแกรมจะถามว่า มีชื่อพอร์ตเก็บไว้ในรายการที่เลือกหรือไม่ ถ้าไม่มีจะเตือนว่า ไม่มีพอร์ต RS-485 เชื่อมต่ออยู่แล้วจะออกจากโปรแกรม แต่ถ้ามีโปรแกรมจะทำงานขั้นต่อไป ดังรูปที่ 3.5



รูปที่ 3.5 โครงสร้างการทำงานส่วนของพอร์ตสื่อสารระหว่างคอมพิวเตอร์กับมิเตอร์

3.2.2 โครงสร้างการทำงานของโปรแกรมส่วนของฟังก์ชันการตั้งค่าเวลาและบันทึกข้อมูล

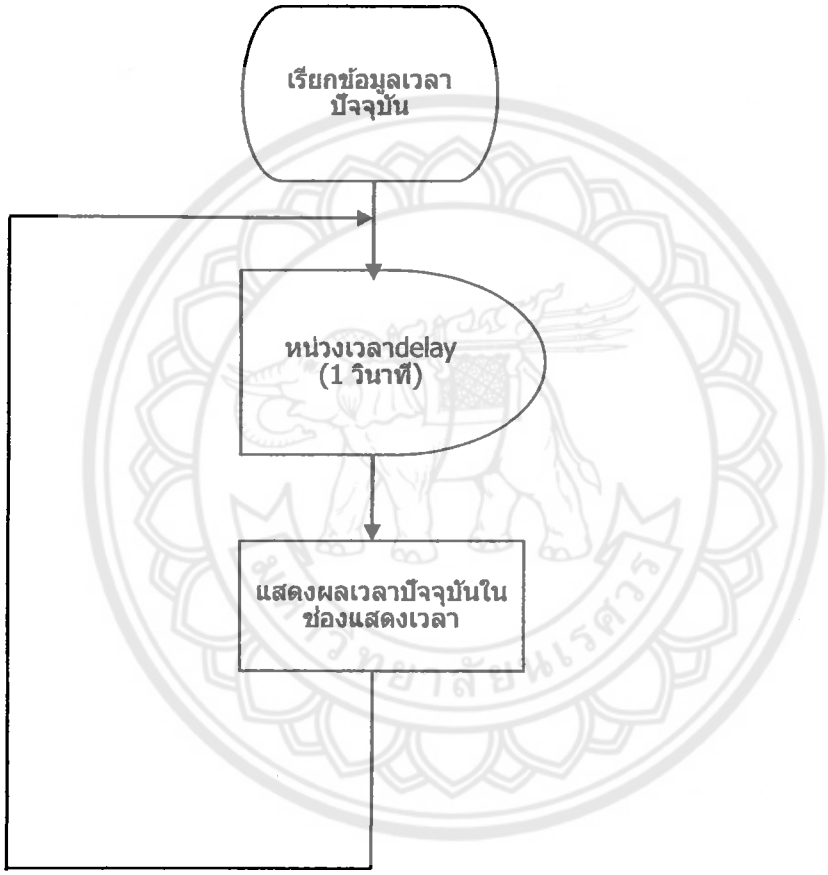
เมื่อโปรแกรมค้นเจอพอร์ต RS-485 ที่เป็นตัวรับส่งข้อมูลพลังงานไฟฟ้าจากมิเตอร์ได้แล้วโปรแกรมจะทำการตั้งค่าและบันทึกข้อมูลลงในตัวแปร Setting file แล้วโปรแกรมจะอ่านค่าของพารามิเตอร์ต่างๆ จาก Setting file ดังรูปที่ 3.6



รูปที่ 3.6 โครงสร้างการทำงานของโปรแกรมส่วนของฟังก์ชันการตั้งค่าเวลาและบันทึกข้อมูล

3.2.3 โครงสร้างการทำงานของโปรแกรมส่วนของฟังก์ชันแสดง วัน เดือน ปี และ เวลา

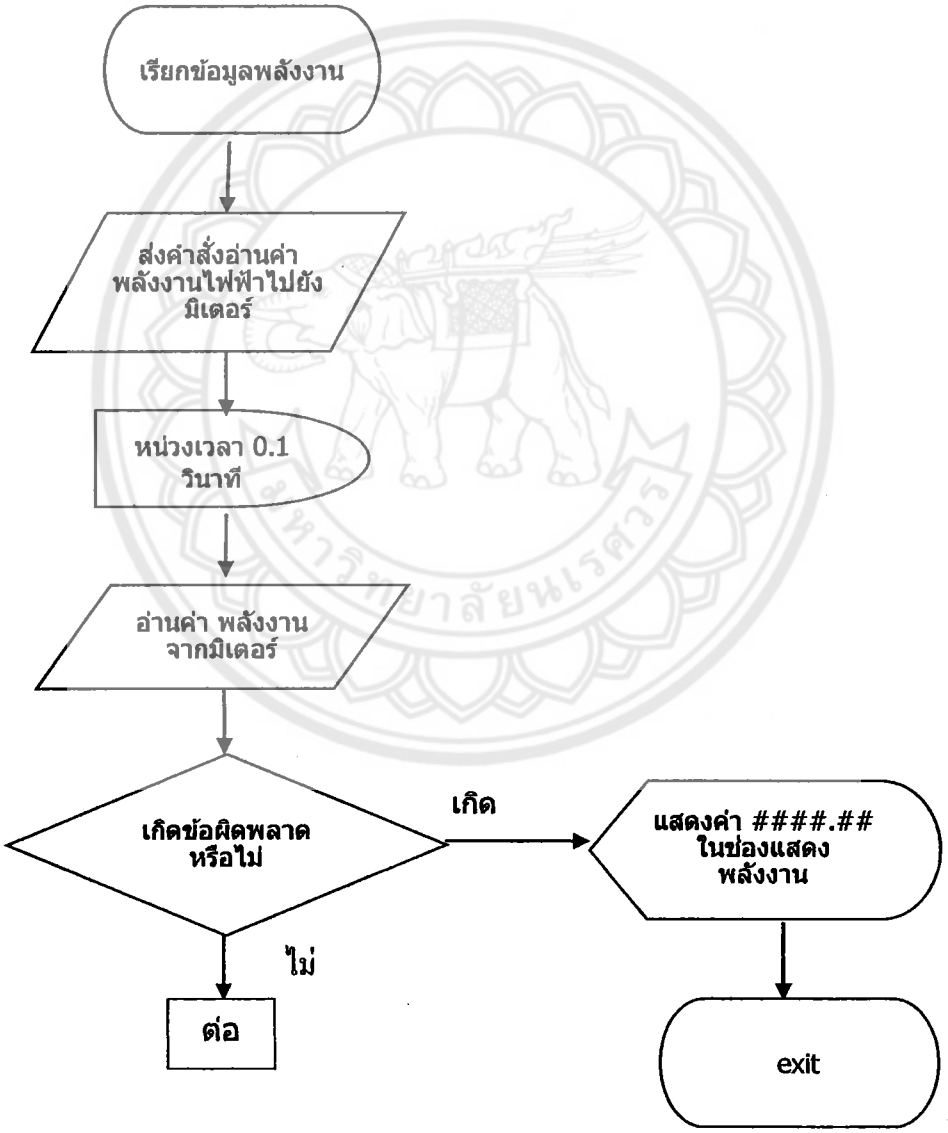
โปรแกรมจะเรียกฟังก์ชันการแสดงค่า เวลา วัน เดือน ปี ตามเวลาปัจจุบัน คือตามเวลาตามคอมพิวเตอร์ โดยมีการหน่วงเวลา 1 วินาที แล้วจะแสดงค่า เวลา วัน เดือน ปี ที่หน้าจอแสดงผล โปรแกรมจะทำงานวนลูปไปเรื่อยๆ ดังรูปที่ 3.7



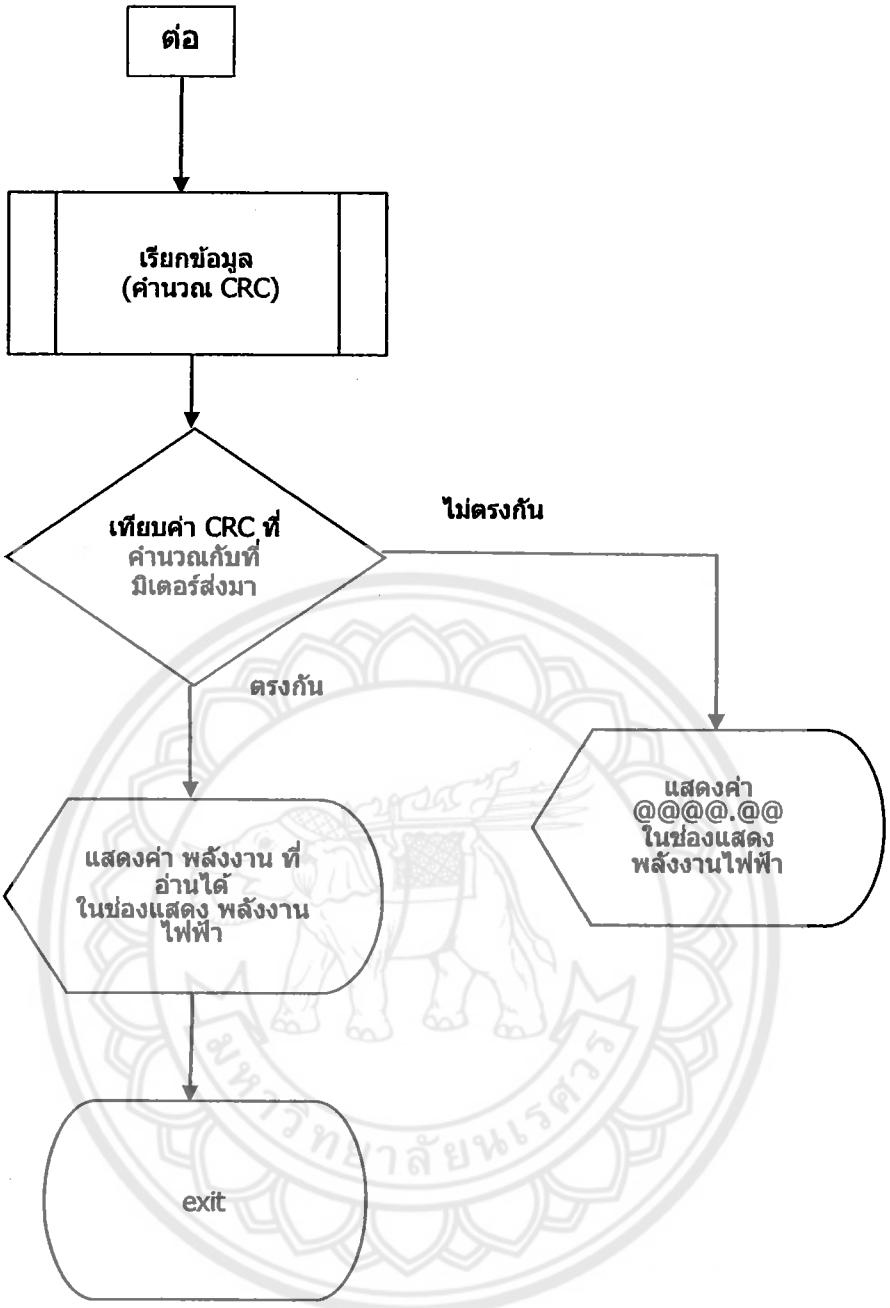
รูปที่ 3.7 โครงสร้างการทำงานของโปรแกรมส่วนของฟังก์ชันแสดง วัน เดือน ปี และ เวลา

3.2.4 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าพลังงานไฟฟ้า(Wh)

โปรแกรมจะเรียกฟังก์ชันค่าพลังงานไฟฟ้า เพื่อส่งคำสั่งอ่านค่าพลังงานไฟฟ้าไปยังมิเตอร์ และให้มิเตอร์ตอบกลับค่าพลังงานไฟฟ้ามา โดยหน่วงเวลา 1 วินาที ถ้าเกิดข้อผิดพลาดจากการอ่านค่าพลังงานไฟฟ้า หน้าจอจะแสดงค่า ####.## แล้วออกจากโปรแกรม แต่ถ้าไม่เกิดข้อผิดพลาดใดๆ โปรแกรมจะเรียกฟังก์ชัน การคำนวณค่า CRC เพื่อเทียบค่า CRC ที่โปรกรมคำนวณได้กับค่าที่มิเตอร์ส่งค่ากลับมาว่าตรงกันหรือไม่ถ้าไม่ตรง หน้าจอจะแสดงค่า @@@@.@@ แล้วออกจากโปรแกรม ถ้าตรง โปรแกรมจะแสดงค่าพลังงานไฟฟ้าบนหน้าจอแสดงผล เป็นหน่วย W/h ดังรูปที่3.8 ถึงรูปที่ 3.9



รูปที่ 3.8 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าพลังงานไฟฟ้า

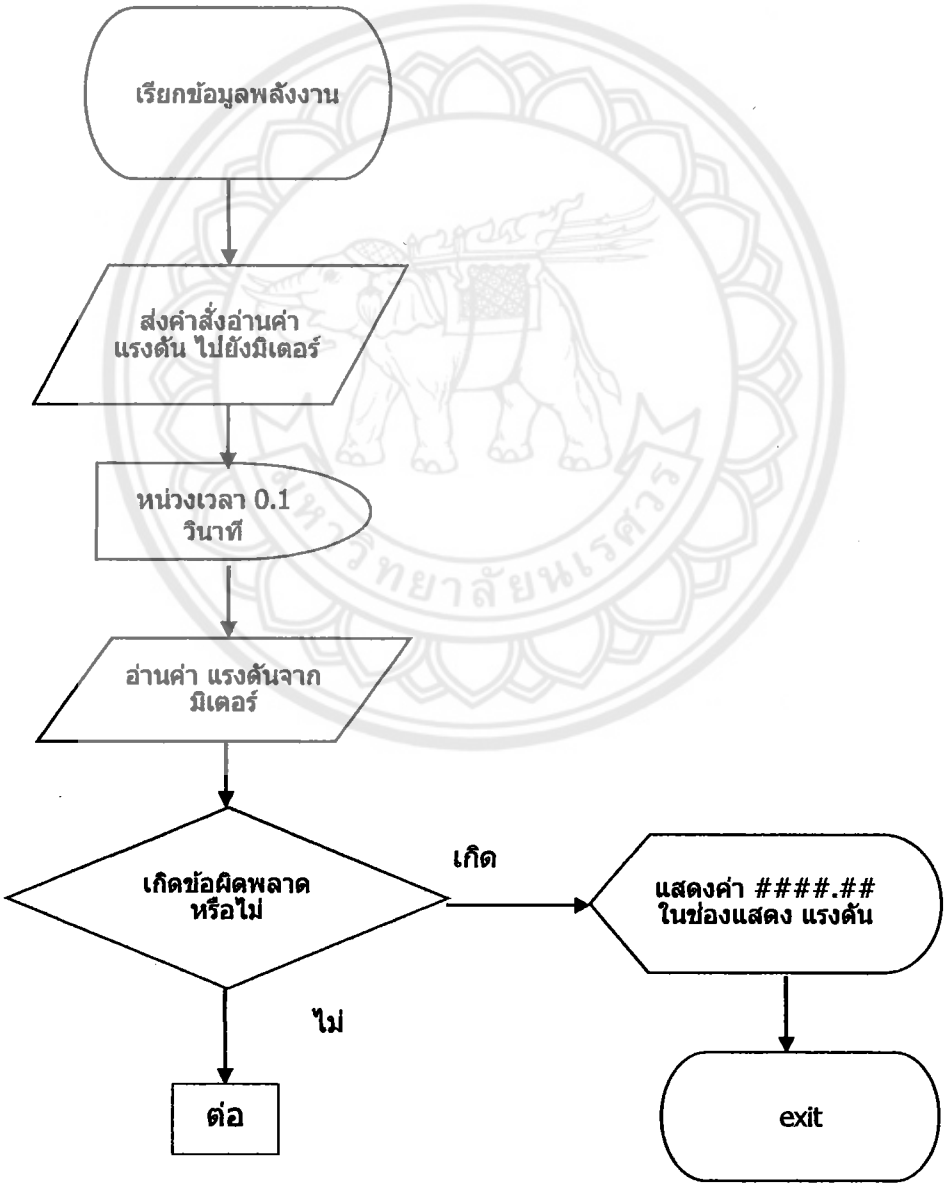


รูปที่ 3.9 (ต่อ) โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าพลังงานไฟฟ้า

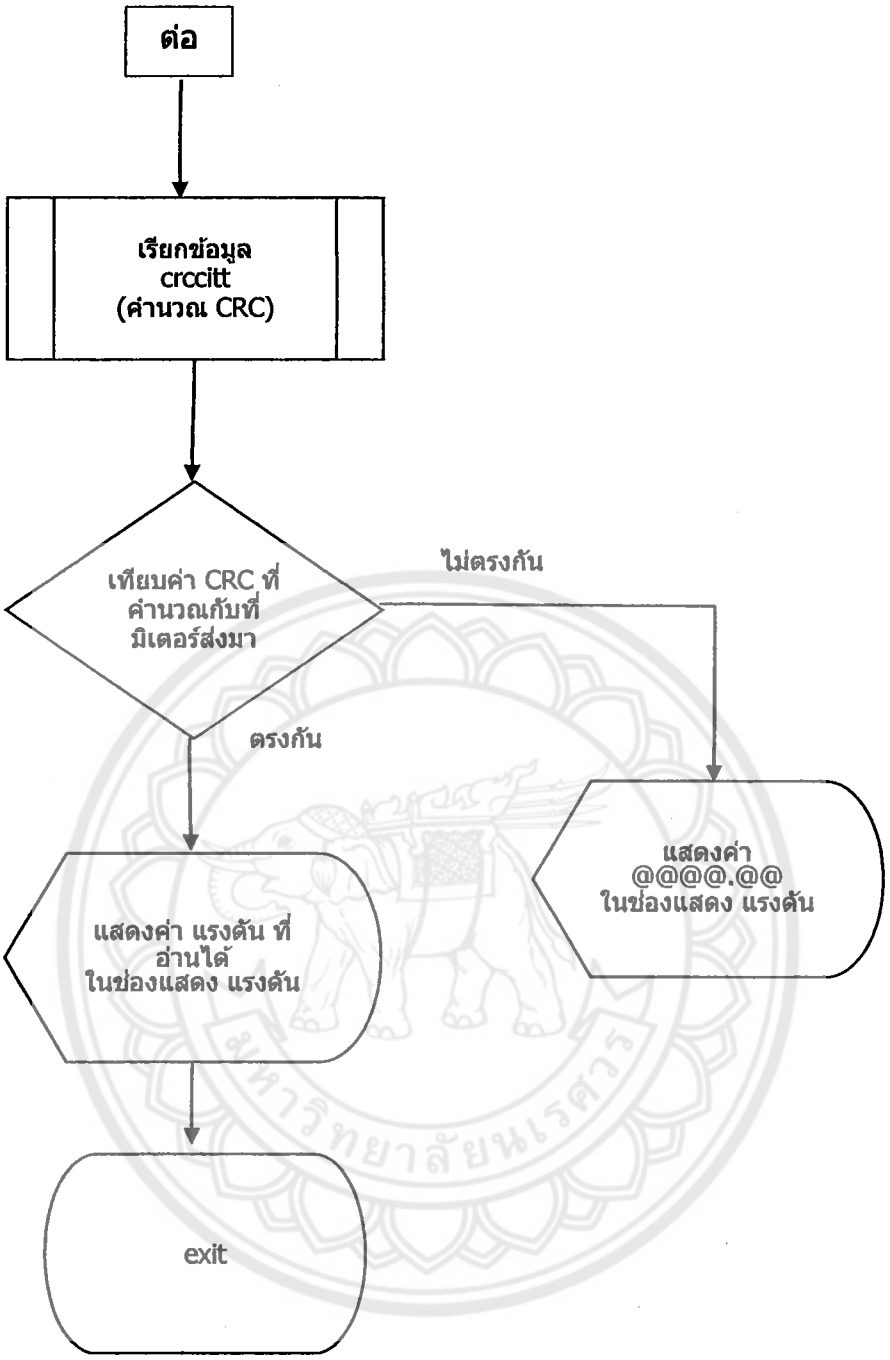
**การคำนวณ CRC (สามารถดูรายละเอียดเพิ่มเติมได้ที่ [8])

3.2.5 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าแรงดันไฟฟ้า

โปรแกรมจะเรียกฟังก์ชันค่าแรงไฟฟ้า เพื่อส่งคำสั่งอ่านค่าพลังงานไฟฟ้าไปยังมิเตอร์ และให้มิเตอร์ตอบกลับค่าแรงดันไฟฟ้ามา โดยหน่วงเวลา 1 วินาที ถ้าเกิดข้อผิดพลาดจากการอ่านค่าแรงดันไฟฟ้า หน้าจอจะแสดงค่า ####.## แล้วออกจากโปรแกรม แต่ถ้าไม่เกิดข้อผิดพลาดใดๆ โปรแกรมจะเรียกฟังก์ชัน การคำนวณค่า CRC เพื่อเทียบค่า CRC ที่โปรกรมคำนวณได้กับค่าที่มิเตอร์ส่งค่ากลับมาว่าตรงกันหรือไม่ถ้าไม่ตรง หน้าจอจะแสดงค่า @@@@.@@ แล้วออกจากโปรแกรม ถ้าตรง โปรแกรมจะแสดงค่าแรงดันไฟฟ้าบนหน้าจอแสดงผล เป็นหน่วย V ดังรูปที่3.10 ถึงรูปที่ 3.11



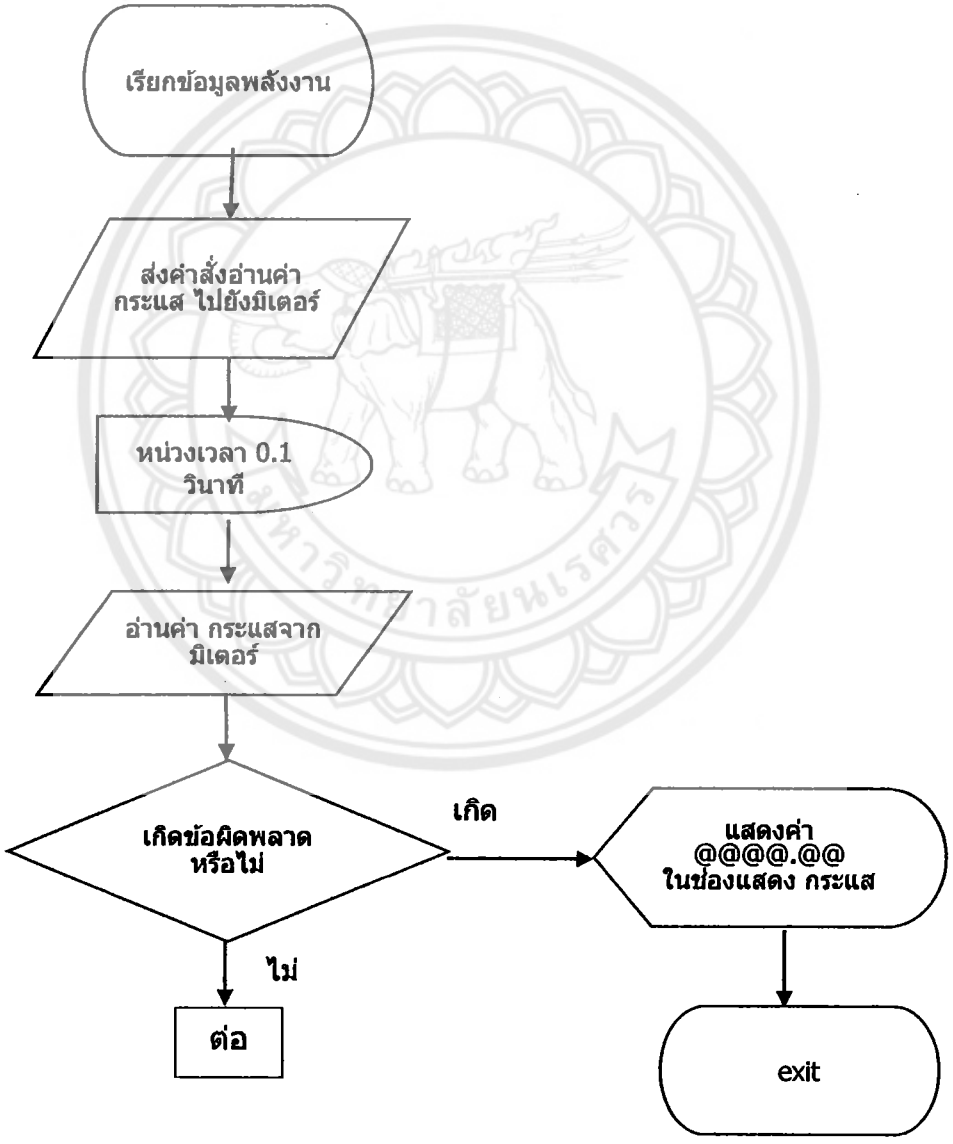
รูปที่3.10 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าแรงดันไฟฟ้า



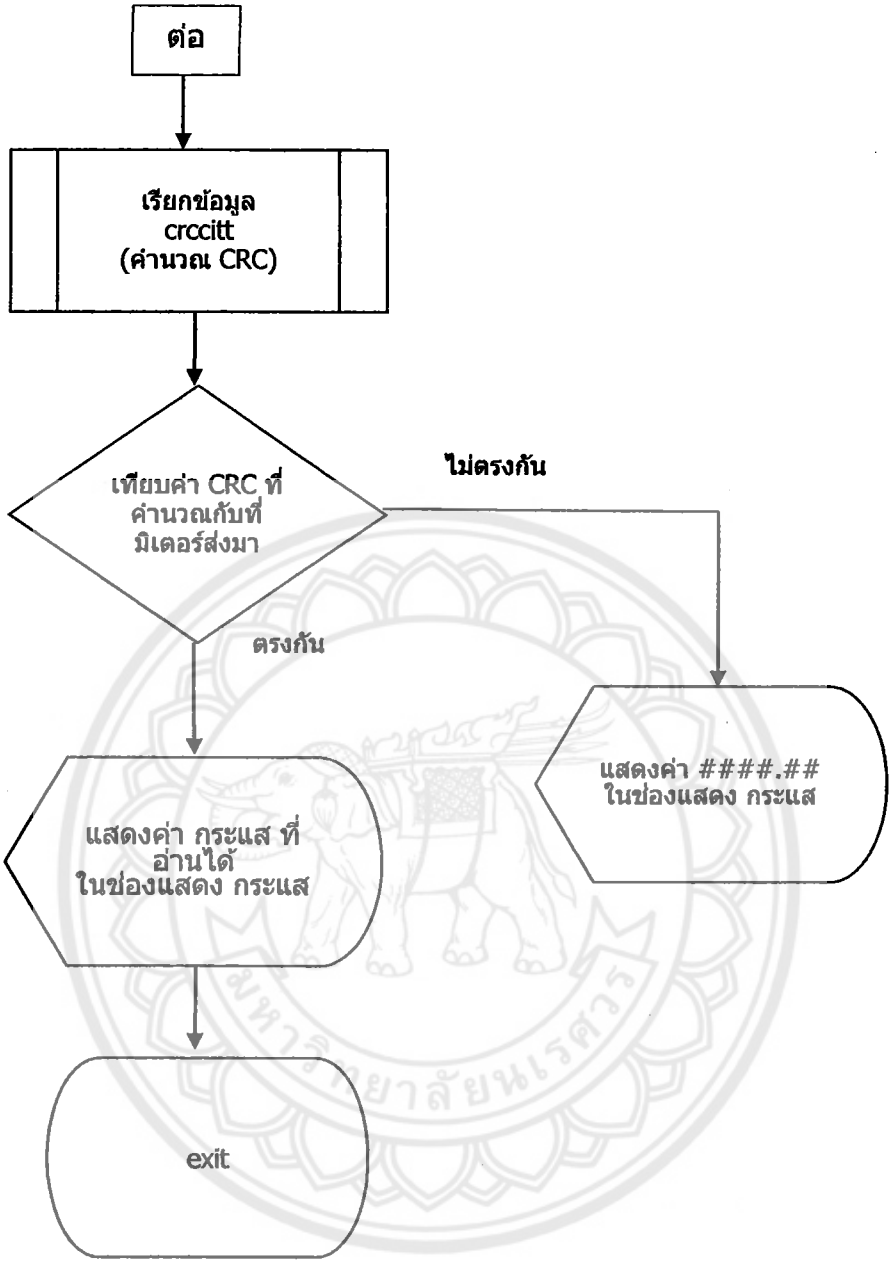
รูปที่3.11 (ต่อ) โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่าแรงดันไฟฟ้า

3.2.6 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่ากระแสไฟฟ้า

โปรแกรมจะเรียกฟังก์ชันค่ากระแสไฟฟ้า เพื่อส่งคำสั่งอ่านค่าพลังงานไฟฟ้าไปยังมิเตอร์ และให้มิเตอร์ตอบกลับค่ากระแสไฟฟ้ามา โดยหน่วงเวลา 1 วินาที ถ้าเกิดข้อผิดพลาดจากการอ่านค่ากระแสไฟฟ้า หน้าจอจะแสดงค่า ####.## แล้วออกจากโปรแกรม แต่ถ้าไม่เกิดข้อผิดพลาดใดๆ โปรแกรมจะเรียกฟังก์ชัน การคำนวณค่า CRC เพื่อเทียบค่า CRC ที่โปรกรมคำนวณได้กับค่าที่มิเตอร์ส่งค่ากลับมาว่าตรงกันหรือไม่ถ้าไม่ตรง หน้าจอจะแสดงค่า @@@@.@@ แล้วออกจากโปรแกรม ถ้าตรง โปรแกรมจะแสดงค่ากระแสไฟฟ้าบนหน้าจอแสดงผล เป็นหน่วย A ดังรูปที่3.12 ถึงรูป 3.13



รูปที่ 3.12 โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่ากระแสไฟฟ้า

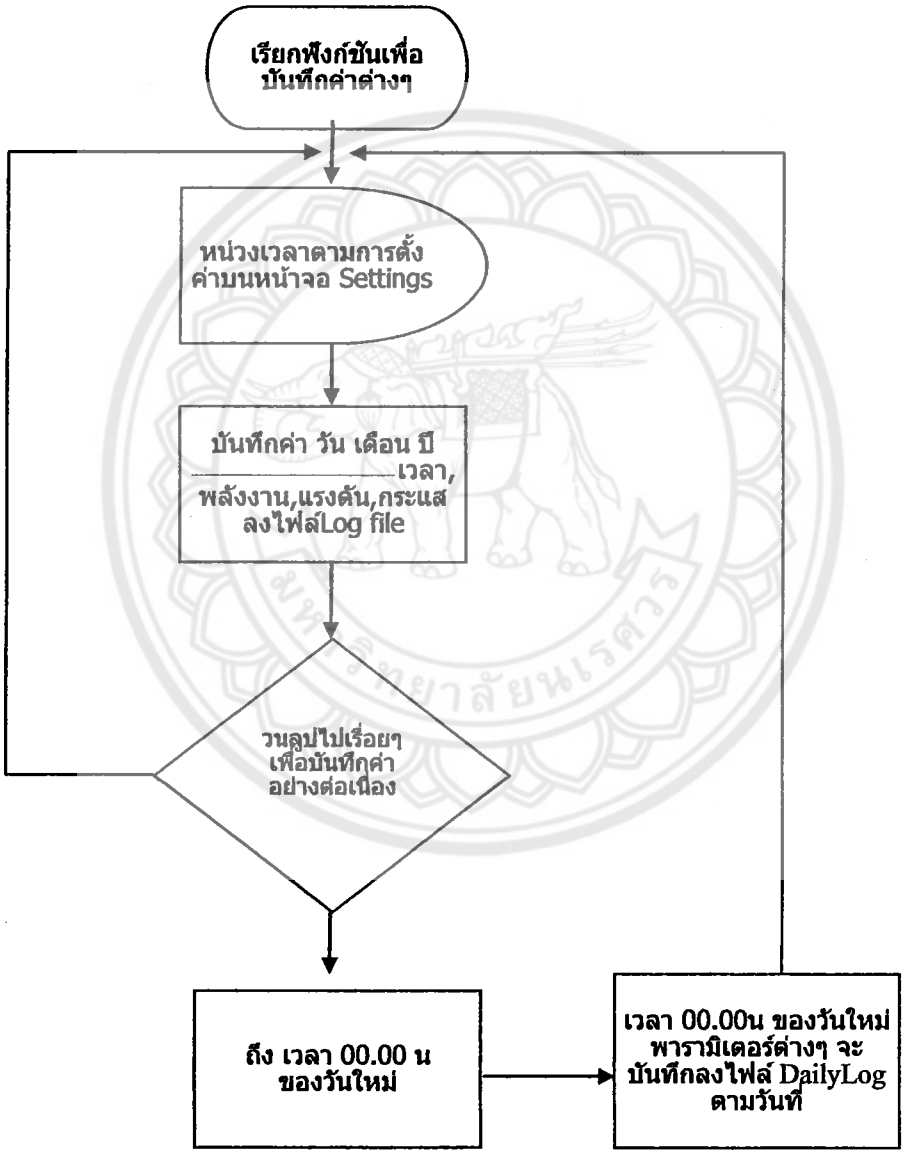


รูปที่ 3.13 (ต่อ)โครงสร้างการทำงานของโปรแกรมส่วนของการแสดงค่ากระแสไฟฟ้า

3.2.7 โครงสร้างการทำงานของโปรแกรมส่วนของการบันทึกค่า วัน เดือน ปี เวลา พลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า

โปรแกรมจะเรียกฟังก์ชันเพื่อบันทึกค่าต่างๆ โดยโปรแกรมจะหน่วงเวลาตามการตั้งค่าบนหน้าจอ Settings แล้วจะบันทึกข้อมูลค่า วัน เดือน ปี เวลา พลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า ลงบน Log file เมื่อถึงเวลา 00.00น ของวันใหม่ พารามิเตอร์ต่างๆจะถูกบันทึกลงไฟล์วันต่อไป สามารถดูการบันทึกค่าต่างๆ เป็นรายวันได้ที่ ไฟล์ DailyLog และสามารถดูค่าที่บันทึกทุกวันได้ที่ไฟล์ log.txt อีกด้วย

ดังรูป 3.14



รูปที่ 3.14 โครงสร้างการทำงานของโปรแกรมส่วนของการบันทึกค่า วัน เดือน ปี เวลา พลังงานไฟฟ้า แรงดันไฟฟ้า กระแสไฟฟ้า

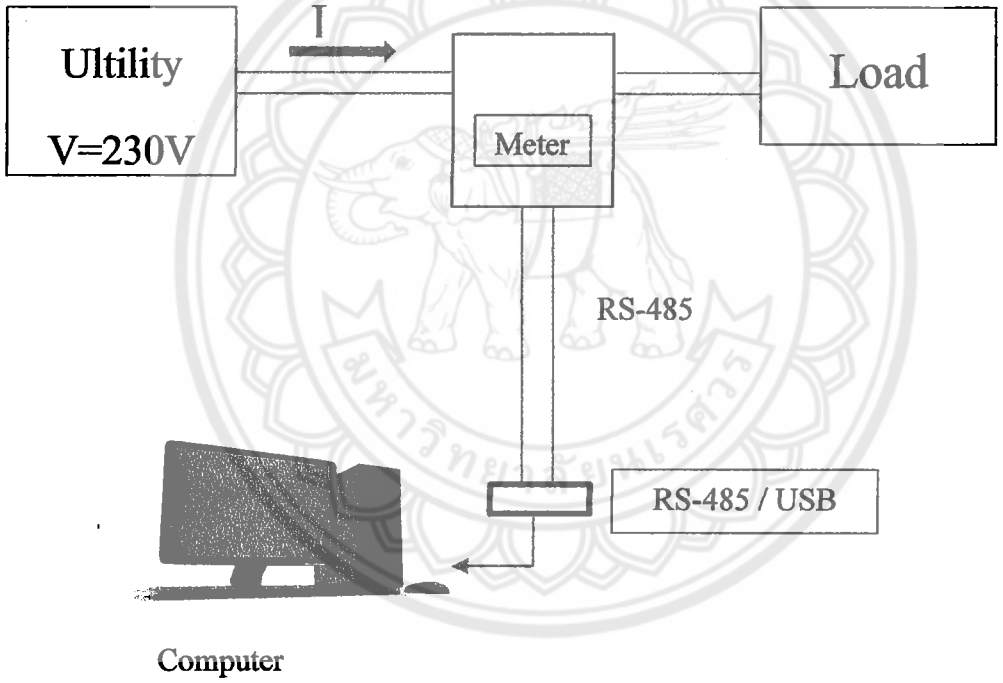
*** ต้องตั้งค่าเวลาในเครื่องคอมพิวเตอร์ให้เป็น 00.00-23.59 น.

3.3 การแปลงข้อมูลจากมิเตอร์ไฟฟ้าดิจิตอล

การที่ข้อมูลพลังงานจากมิเตอร์พลังงานไฟฟ้าแบบดิจิตอลจะแสดงออกทางหน้าต่างแสดงผลได้ต้องมีการถอดรหัสข้อมูลจากเครื่องมิเตอร์พลังงานไฟฟ้าแบบดิจิตอล คือตัวเลขโปรโตคอลที่ติดมากับตัวเครื่องที่จะส่งข้อมูลค่าพลังงานมายังหน้าจอแสดงผล โดยผ่านการแปลงข้อมูลจาก RS-485 ระหว่างเครื่องคอมพิวเตอร์กับมิเตอร์พลังงานไฟฟ้าแบบดิจิตอล

3.4 การทำงานของโปรแกรมในแต่ละส่วน

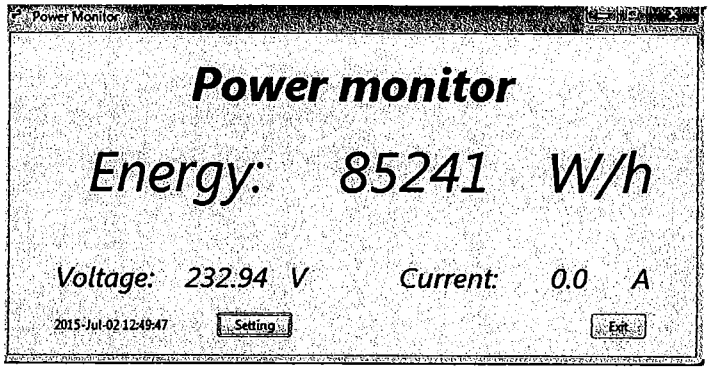
3.4.1 การทำงานของโปรแกรมและการใช้งานโปรแกรม



รูปที่ 3.15 การต่อชุดทดสอบเพื่อให้ได้ค่าดังรูปที่ 3.16

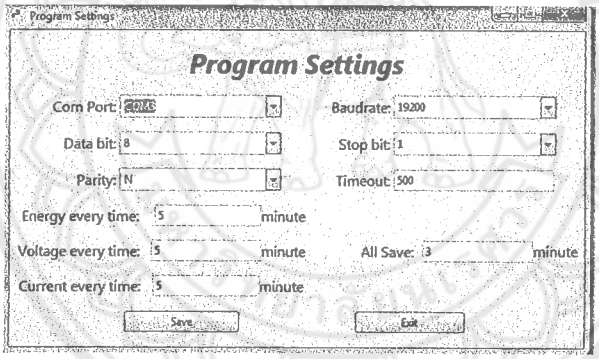
ต่อโหลดเข้ากับมิเตอร์โดยโหลดที่ใช้เราจะทดลองโดยหลอดไฟ เพราะกินกระแสสูงสามารถแสดงค่ากระแสไฟฟ้าได้อย่างชัดเจนและสามารถทดลองได้เป็นเวลานาน เมื่อต่อโหลดเข้ากับมิเตอร์แล้วต่อมิเตอร์อีกข้างหนึ่งเข้ากับ RS-485 เพื่อเป็นตัวเชื่อมต่อกับคอมพิวเตอร์ จากนั้นเริ่มเปิดเครื่องคอมพิวเตอร์ โปรแกรมจะเริ่มบันทึกค่าโดยอัตโนมัติดังรูปที่ 3.16 โปรแกรมจะแสดงการบันทึกข้อมูลค่าพลังงานไฟฟ้าจากมิเตอร์ไฟฟ้าแบบดิจิตอลเมื่อค่าพลังงานเริ่มบันทึกค่าแล้วสามารถกดปุ่ม Setting เพื่อกลับไปตั้งค่าปรับเปลี่ยนเวลาในการบันทึกค่าใหม่อย่างเช่น เปลี่ยนจาก

การบันทึกค่าทุกๆ 3 นาที ให้เป็นบันทึกค่าทุกๆ 5 นาที ได้ดัง รูปที่ 3.17



รูปที่ 3.16 หน้าต่างแสดงผลของค่าพลังงานจากมิเตอร์ไฟฟ้าแบบดิจิตอล

เมื่อต้องการปรับเปลี่ยนเวลาในการบันทึกค่าใหม่หน้าต่างแสดงข้อมูลพลังงานจากมิเตอร์ จะกลับมาเป็นดังรูป 3.17 เมื่อตั้งค่าเวลาเสร็จแล้วกด save โปรแกรมจะกลับไป หน้าต่างแสดงผลของค่าพลังงานจากมิเตอร์ไฟฟ้าแบบดิจิตอล ดังรูปที่ 3.16 เหมือนเดิม



รูปที่ 3.17 หน้าต่างการตั้งค่าเวลาในการบันทึกค่าของพลังงาน

จากรูปที่ 3.17 เป็นหน้าต่างแสดงผลนี้จะแสดงค่าต่างๆที่เชื่อมระหว่าง RS-485 กับคอมพิวเตอร์

Com Port คือช่องที่ RS-485 ต่อเข้ากับคอมพิวเตอร์ว่างต่อกับคอมที่ com port ที่เท่าไรในการทดลองนี้จะต่อเข้ากับcom port ที่ 3

Baud rate คือการสื่อสารหรือการรับส่งข้อมูลในระบบดิจิตอลการสื่อสารกับมิเตอร์ไฟฟ้าแบบดิจิตอลมิซูบิชิ SX-1 จะใช้ Baud rate ที่19200

Data bit คือข้อมูลที่เป็นองค์ประกอบสำคัญอย่างหนึ่งในระบบคอมพิวเตอร์ เป็นสิ่งที่ต้องป้อนเข้าไปในคอมพิวเตอร์ ข้อมูลที่สามารถนำมาใช้กับคอมพิวเตอร์ได้ มี 5 ประเภท คือ ข้อมูลตัวเลข (Numeric Data) ข้อมูลตัวอักษร (Text Data) ข้อมูลเสียง (Audio Data) ข้อมูลภาพ (Images

Data) และข้อมูลภาพเคลื่อนไหว (Video Data) ในการนำข้อมูลไปใช้จะมีระดับโครงสร้างของข้อมูล ได้แก่ ตัวเลข หรือ ตัวอักษร หรือ สัญลักษณ์พิเศษ 1 ตัว เช่น 0, 1, ..., 9, A, B, ..., Z และเครื่องหมายต่างๆ ซึ่ง 1 ไบต์จะเท่ากับ 8 บิต หรือ ตัวอักขระ 1 ตัว เป็นต้น

Stop bit คือบิตหยุดสัญญาณสั่งบอกว่าให้หยุดเพราะการถ่ายโอนข้อมูลสำเร็จเรียบร้อยแล้ว

Parity คือบิตข้อมูลที่ใช้ในการตรวจสอบความผิดพลาดของการรับสัญญาณที่ปลายทางหรือสายที่ควบคุมการโต้ตอบ เพื่อควบคุมจังหวะของการรับ-ส่งข้อมูลแต่ละชุดในการคลองนี้ N คือ Non ไม่มีภาวะบิตคู่หรือภาวะคี่

Timeout คือ เป็นเวลาเริ่มต้นในการอ่านค่าจากพอร์ต หน่อยเป็น ms

Energy every time คือช่องที่สามารถระบุเวลาได้ว่าจะให้ค่าพลังงานแสดงบนหน้าต่างแสดงผลทุกๆกี่วินาที

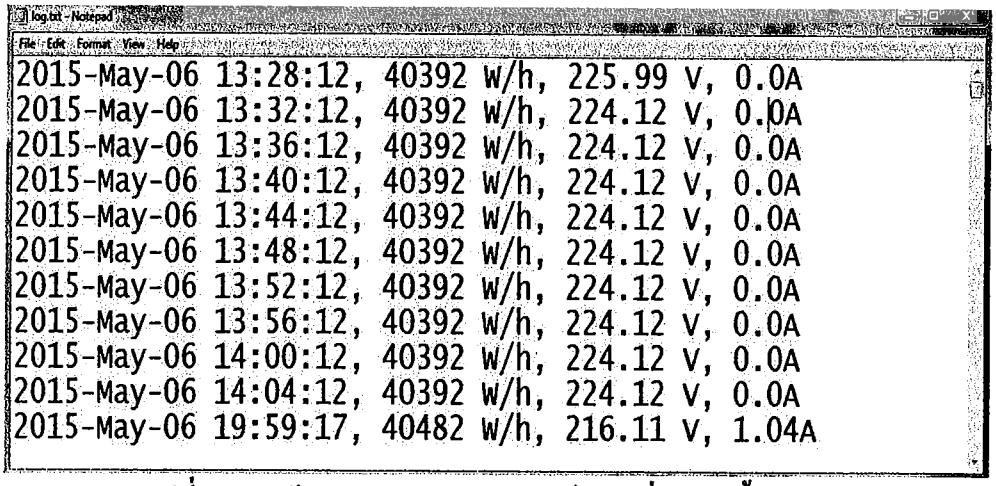
Voltage every time คือช่องที่สามารถระบุเวลาได้ว่าจะให้ค่าแรงดันไฟฟ้าแสดงบนหน้าต่างแสดงผลทุกๆกี่วินาที

Current every time คือช่องที่สามารถระบุเวลาได้ว่าจะให้ค่ากระแสไฟฟ้าแสดงบนหน้าต่างแสดงผลทุกๆกี่วินาที

All save คือช่องที่สามารถระบุเวลาได้ว่าจะให้ค่าของข้อมูลทางไฟฟ้าทั้งหมดแสดงหน้าต่างแสดงผลเป็นเวลาเท่าไรหน่วยเป็นนาที

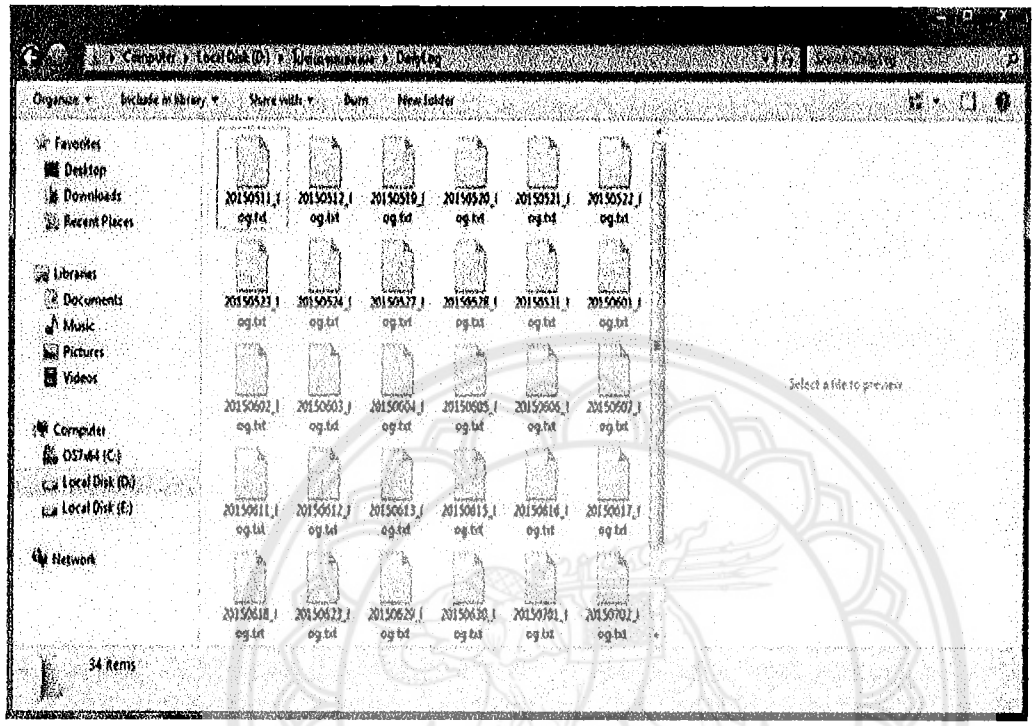
3.4.2 การแสดงการจัดเก็บข้อมูลจากมิเตอร์วัดพลังงานไฟฟ้าชนิดดิจิตอล

เมื่อโปรแกรมอ่านค่าจากมิเตอร์ข้อมูลพลังงานไฟฟ้าจะถูกเก็บในไฟล์ log.txt แสดงค่าดังรูป จากรูปที่ 3.18 เมื่อเปิดโปรแกรมให้โปรแกรมเริ่มบันทึกค่าข้อมูลพลังงานจากมิเตอร์จะมาจัดเก็บที่ โฟล์เดอร์ log.txt เพื่อเรียกดูย้อนหลัง



รูปที่ 3.18 หน้าต่างแสดงค่าพารามิเตอร์ต่างๆที่บันทึกทั้งหมด

เราสามารถเก็บไฟล์ข้อมูลที่บันทึกจากมิเตอร์แบ่งเป็นวันละไฟล์ เพื่อความสะดวกในการค้นหาวันที่ย้อนหลังเมื่อต้องการดูสถิติการใช้ค่าพลังงานไฟฟ้าได้อีกด้วยดังรูปที่ 3.19



รูปที่ 3.19 หน้าต่างแสดงค่าไฟล์ที่โปรแกรมบันทึกทั้งหมด

บทที่ 4
การทดสอบและผลการทดสอบ

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

4.1 การทดสอบการแสดงผลทางหน้าจอแสดงผล

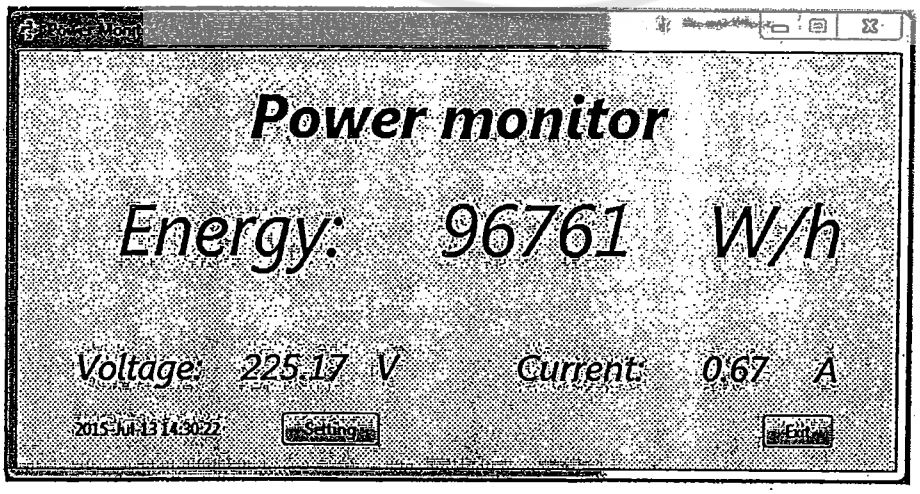
การทดสอบการแสดงผล จะทำโดยการต่อชุดทดสอบ เข้ากับมิเตอร์ และ ต่อสายสัญญาณเข้ากับโปรแกรม เพื่อทดสอบว่า โปรแกรมแสดงผลได้ตรงตามหน้าจอของตัวมิเตอร์หรือไม่ มีความคลาดเคลื่อนมากน้อยเพียงใด โดยที่ ค่าการแสดงผล จะนำมาแสดง 3ค่าหลักๆ คือ

1. ค่าพลังงานไฟฟ้า*(E)*
2. ค่าแรงดันไฟฟ้า*(V)*
3. ค่ากระแสไฟฟ้า (A)

ผลการทดสอบ การแสดงผลของหน้าจอมิเตอร์ เทียบกับ การแสดงผลของโปรแกรม

รายละเอียดชุดทดสอบ

- แรงดันจากการไฟฟ้า 220-230 โวตล์
- หลอดไฟ ขนาด 100 วัตต์ 3 หลอด



รูปที่ 4.1หน้าต่างแสดงผลของโปรแกรม

รูปที่ 4.2หน้าจอแสดงผลของมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล

จากการทดสอบการแสดงผลของโปรแกรม จากรูปที่ 4.1 และ 4.2 พบว่า โปรแกรมสามารถที่จะแสดงผลออกมาทางหน้าจอได้อย่างชัดเจน และมีค่าพารามิเตอร์ต่างๆ ถูกต้องตามหน้าจอของมิเตอร์ และมีความละเอียดมากกว่าค่าที่หน้าจอมิเตอร์ โปรแกรมสามารถวัดค่าแรงดันไฟฟ้าและกระแสไฟฟ้าได้อีกด้วยไม่มีความคลาดเคลื่อนในการแสดงผล และโปรแกรม สามารถแสดงผลแบบติดตามได้ หากมีการเปลี่ยนแปลงของค่าพลังงานทันทีทันใด

4.2 การทดสอบการการจัดเก็บข้อมูล

ในการทดสอบการบันทึกค่านั้น จะเป็นการทดสอบ เก็บค่าข้อมูล ที่เวลาต่างกัน เพื่อทดสอบความแม่นยำในการเก็บค่า ตามความห่างของเวลาที่ตั้งไว้ และ จะตั้งค่าโปรแกรม ให้สร้างไฟล์เก็บข้อมูลขึ้นมาใหม่ เมื่อเก็บค่าครบตามเวลาที่ตั้ง

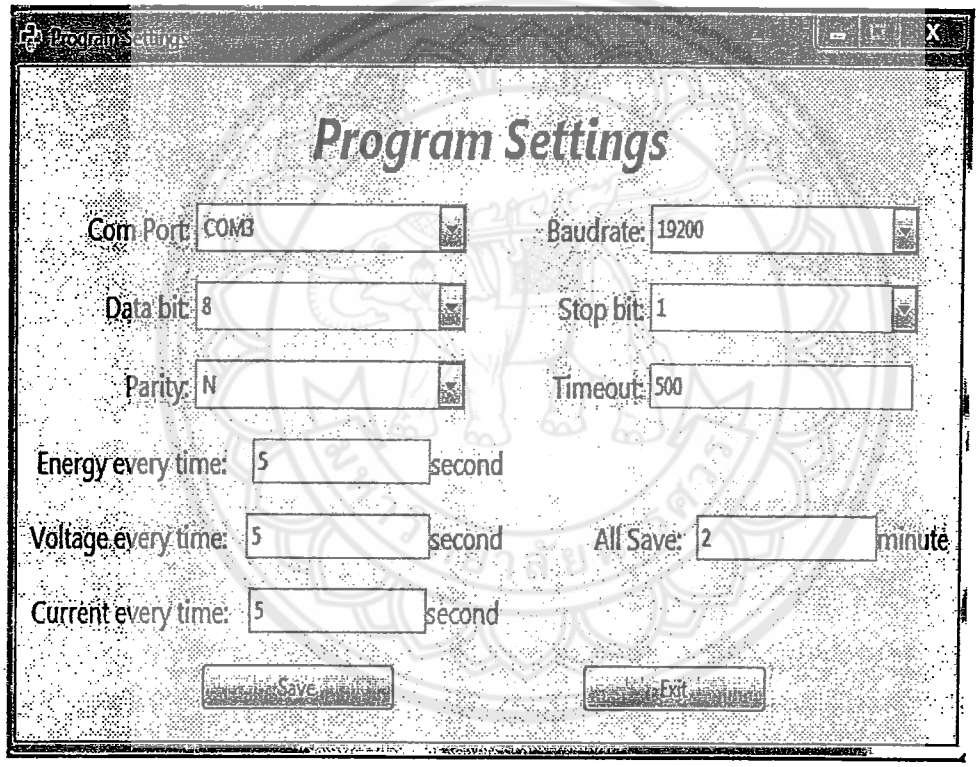
1.เก็บค่าพารามิเตอร์ไฟฟ้าต่างๆทุกๆ 5วินาที แรงดันไฟฟ้าทุกๆ 5 วินาที กระแสไฟฟ้าทุกๆ 5 วินาที เมื่อเก็บครบ 2นาทีและบันทึกค่าทุกๆ 2 นาที

2.เก็บค่าพารามิเตอร์ไฟฟ้าต่างๆทุกๆ 10วินาที แรงดันไฟฟ้าทุกๆ 5 วินาที กระแสไฟฟ้าทุกๆ 10 วินาที เมื่อเก็บครบ 3นาทีและบันทึกค่าทุกๆ 3 นาที

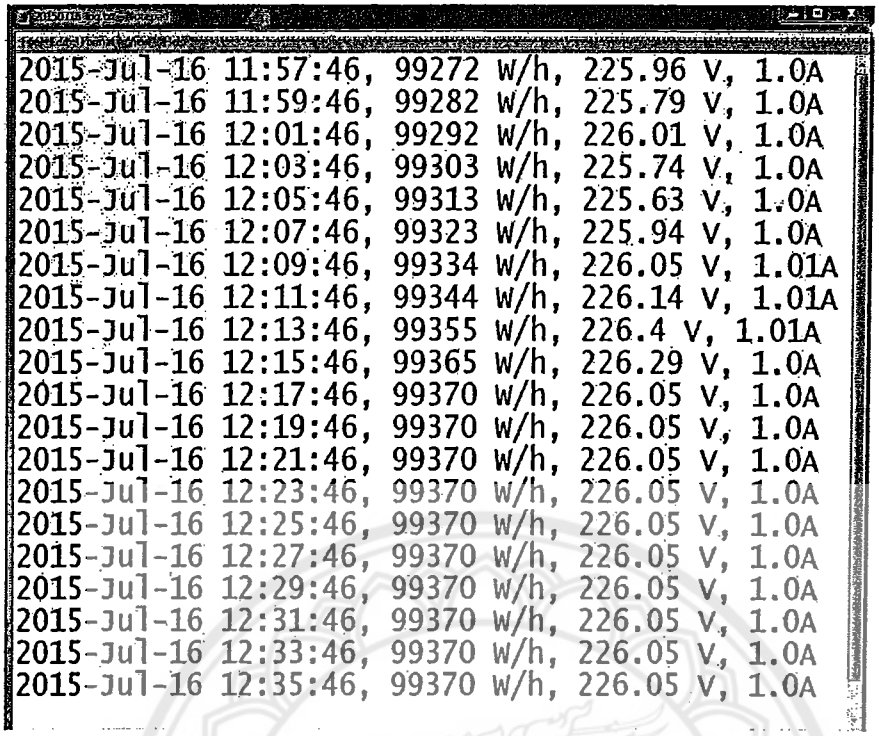
3. เก็บค่าพารามิเตอร์ไฟฟ้าต่างๆทุกๆ 10 วินาที แรงดันไฟฟ้าทุกๆ 5 วินาที กระแสไฟฟ้าทุกๆ 10 วินาที เมื่อเก็บครบ 3 นาทีและบันทึกค่าทุกๆ 3 นาที เป็นเวลา 3 วัน

จะให้โปรแกรมขึ้นไฟล์บันทึกค่าใหม่ ที่ต้องขึ้นไฟล์ใหม่นั้น ก็เพราะจะเป็นการสะดวกต่อดูบันทึกข้อมูลย้อนหลัง และจำนวนข้อมูลจะได้ไม่มีมากจนเกินไป

กรณีที่ 1 บันทึกค่าพารามิเตอร์ต่างๆ ทุกๆ 2 นาทีได้ผลดังรูป

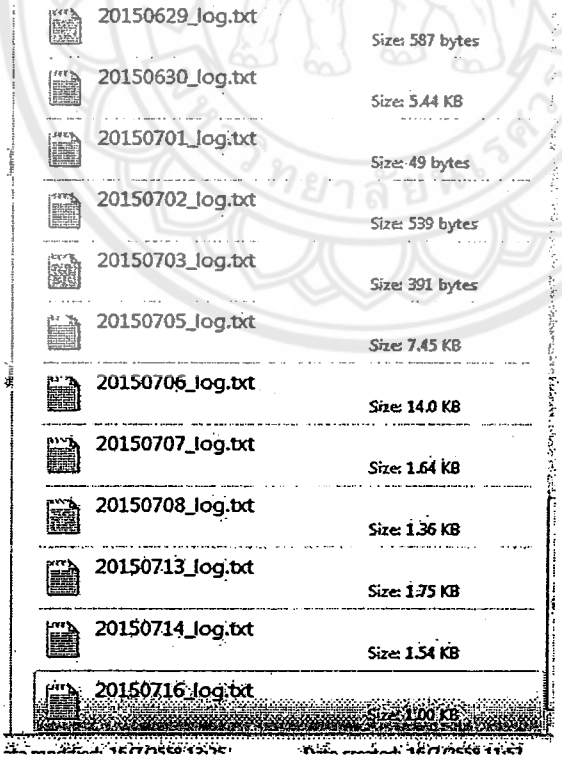


รูปที่ 4.3 การตั้งค่าการบันทึกค่าของโปรแกรม เมื่อบันทึกค่าครบ 2 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



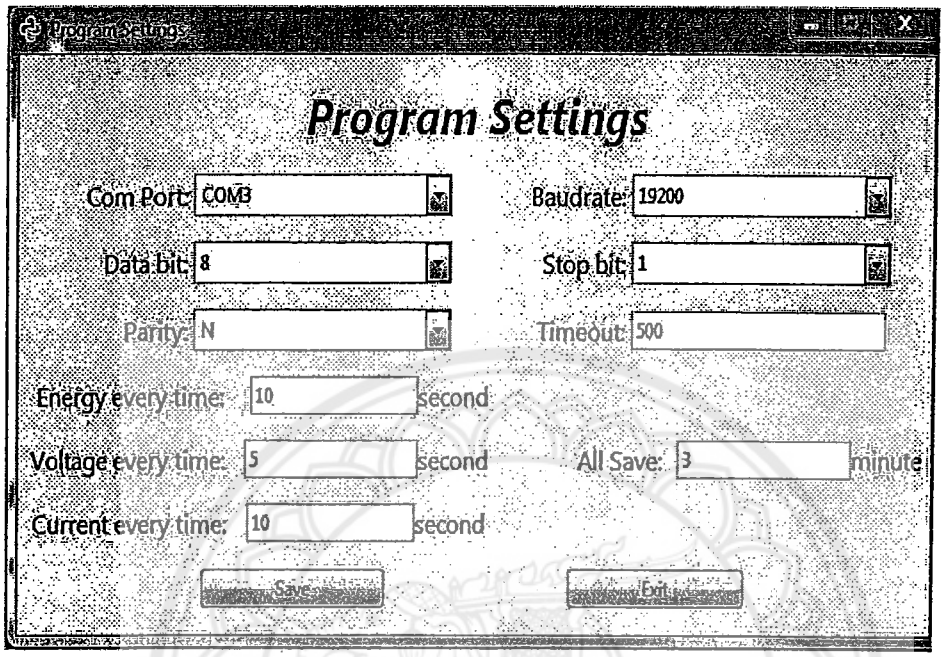
```
2015-Jul-16 11:57:46, 99272 W/h, 225.96 V, 1.0A
2015-Jul-16 11:59:46, 99282 W/h, 225.79 V, 1.0A
2015-Jul-16 12:01:46, 99292 W/h, 226.01 V, 1.0A
2015-Jul-16 12:03:46, 99303 W/h, 225.74 V, 1.0A
2015-Jul-16 12:05:46, 99313 W/h, 225.63 V, 1.0A
2015-Jul-16 12:07:46, 99323 W/h, 225.94 V, 1.0A
2015-Jul-16 12:09:46, 99334 W/h, 226.05 V, 1.01A
2015-Jul-16 12:11:46, 99344 W/h, 226.14 V, 1.01A
2015-Jul-16 12:13:46, 99355 W/h, 226.4 V, 1.01A
2015-Jul-16 12:15:46, 99365 W/h, 226.29 V, 1.0A
2015-Jul-16 12:17:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:19:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:21:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:23:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:25:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:27:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:29:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:31:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:33:46, 99370 W/h, 226.05 V, 1.0A
2015-Jul-16 12:35:46, 99370 W/h, 226.05 V, 1.0A
```

รูปที่ 4.4เป็นผลการบันทึกข้อมูล ของค่าพารามิเตอร์ต่างๆ ทุกๆ 2 นาที

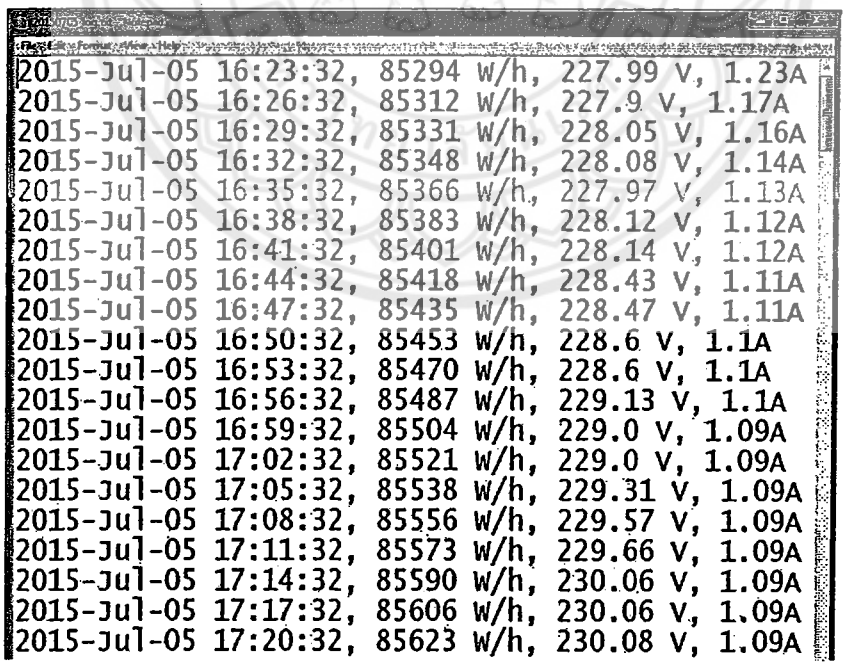


รูปที่ 4.5เมื่อครบ 1 วัน หรือเวลา 00.00น การบันทึกค่าพารามิเตอร์จะขึ้นไฟล์ใหม่เป็นวันที่ถัดไป

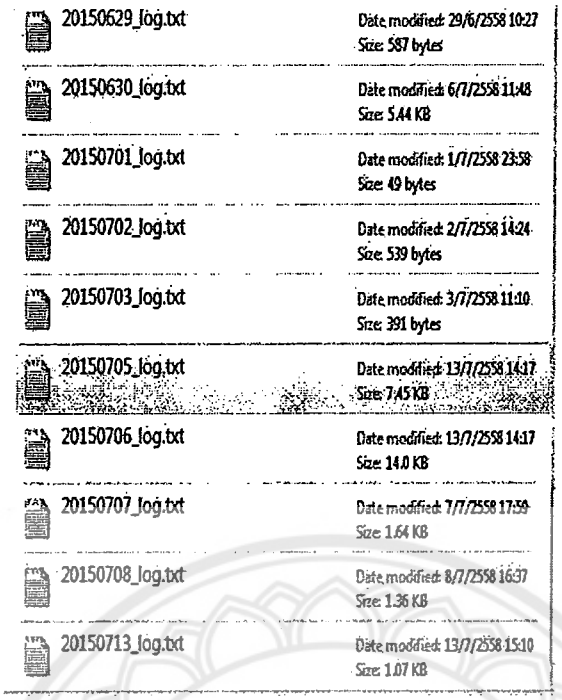
กรณีที่ 2 บันทึกค่าพารามิเตอร์ต่างๆ ทุกๆ 3 นาทีได้ผลดังรูป



รูปที่ 4.6 การตั้งค่าการบันทึกค่าของโปรแกรม เมื่อบันทึกค่าครบ 3นาทีให้ขึ้นที่เก็บค่าอันใหม่

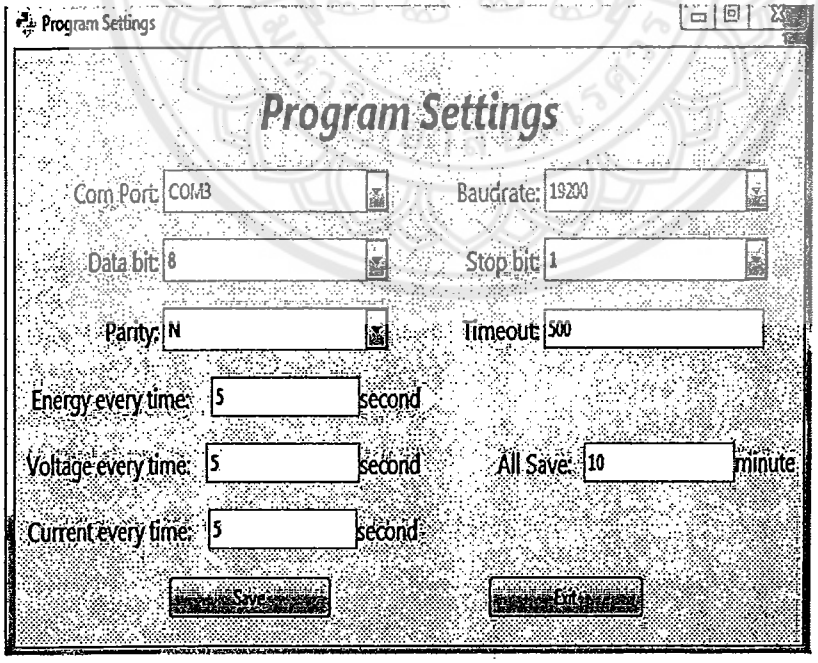


รูปที่ 4.7 เป็นผลการบันทึกข้อมูล ของค่าพารามิเตอร์ต่างๆ ทุกๆ 3 นาที

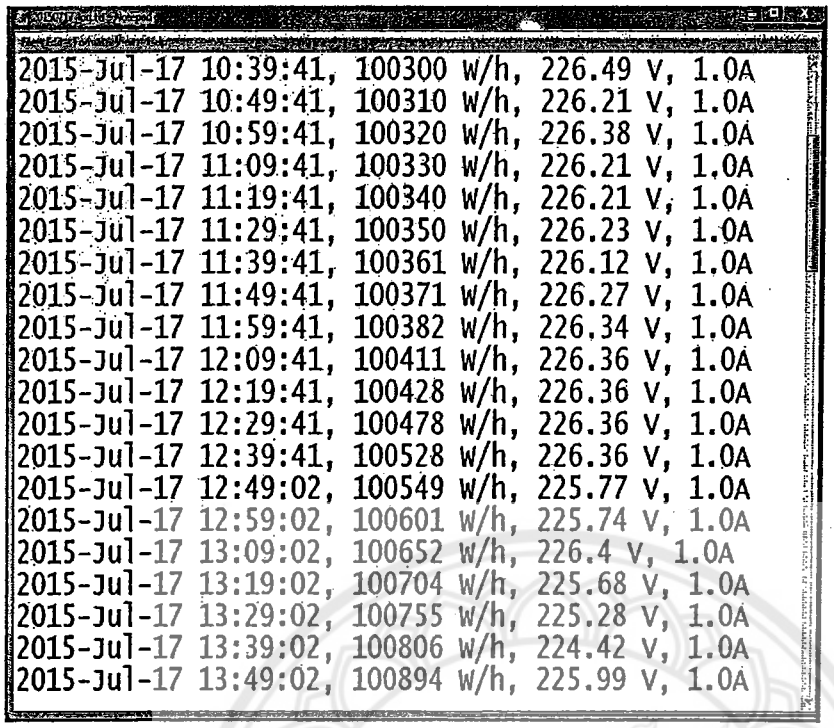


รูปที่ 4.8เมื่อครบ 1 วัน หรือเวลา 00.00น การบันทึกค่าพารามิเตอร์ จะขึ้นไฟล์ใหม่เป็นวันที่ถัดไป

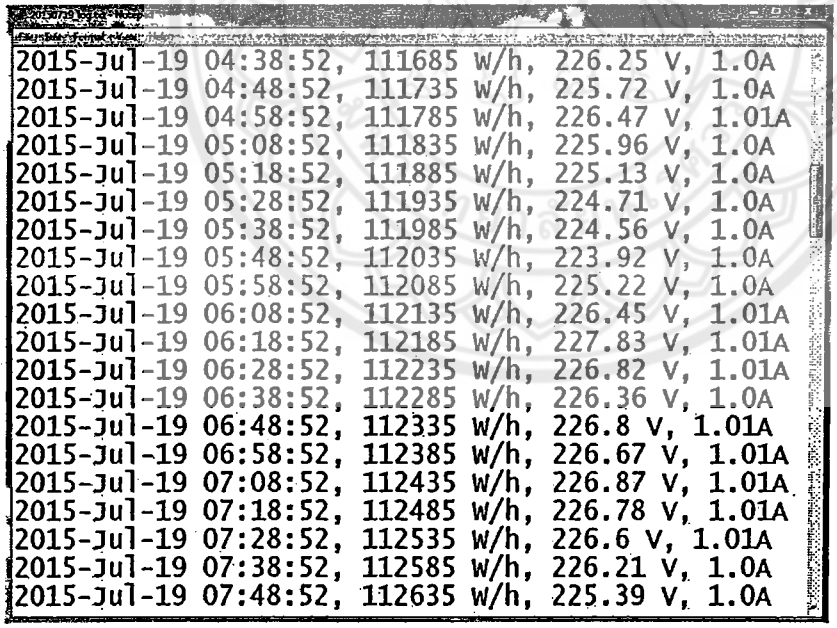
กรณีที่ 3 บันทึกค่าพารามิเตอร์ต่างๆ ทุกๆ 10 นาทีเป็นเวลา 3 วันได้ผลดังรูป



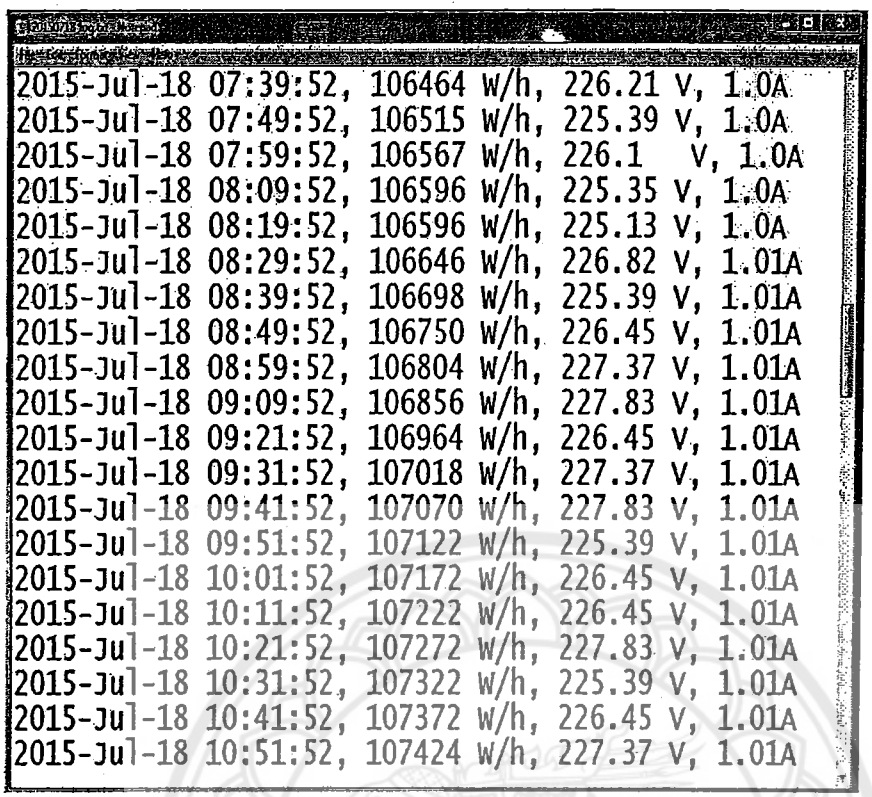
รูปที่ 4.9 การตั้งค่าการบันทึกค่าของโปรแกรม เมื่อบันทึกค่าครบ 10 นาทีให้ขึ้นที่เก็บไฟล์อันใหม่



รูปที่ 4.10 (ก)เป็นผลการบันทึกข้อมูล ของค่าพารามิเตอร์ต่างๆ ทุกๆ 10 นาที เป็นเวลา 3 วัน

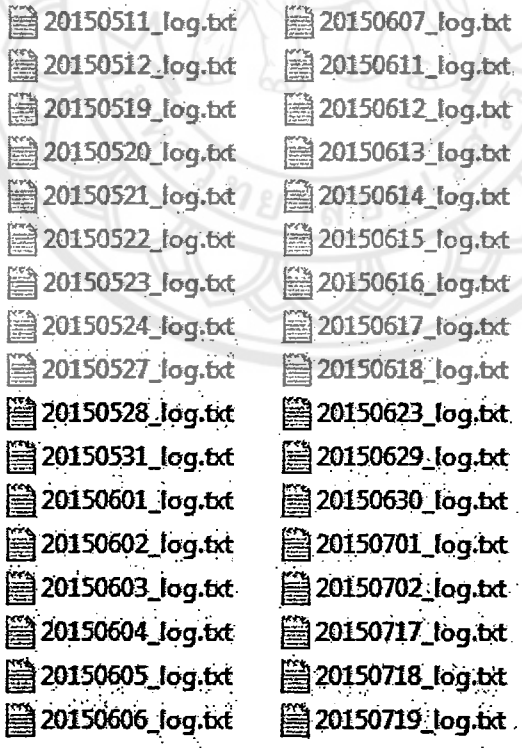


รูปที่ 4.11 (ข)เป็นผลการบันทึกข้อมูล ของค่าพารามิเตอร์ต่างๆ ทุกๆ 10 นาที เป็นเวลา 3 วัน



```
2015-Jul-18 07:39:52, 106464 W/h, 226.21 V, 1.0A
2015-Jul-18 07:49:52, 106515 W/h, 225.39 V, 1.0A
2015-Jul-18 07:59:52, 106567 W/h, 226.1  V, 1.0A
2015-Jul-18 08:09:52, 106596 W/h, 225.35 V, 1.0A
2015-Jul-18 08:19:52, 106596 W/h, 225.13 V, 1.0A
2015-Jul-18 08:29:52, 106646 W/h, 226.82 V, 1.01A
2015-Jul-18 08:39:52, 106698 W/h, 225.39 V, 1.01A
2015-Jul-18 08:49:52, 106750 W/h, 226.45 V, 1.01A
2015-Jul-18 08:59:52, 106804 W/h, 227.37 V, 1.01A
2015-Jul-18 09:09:52, 106856 W/h, 227.83 V, 1.01A
2015-Jul-18 09:21:52, 106964 W/h, 226.45 V, 1.01A
2015-Jul-18 09:31:52, 107018 W/h, 227.37 V, 1.01A
2015-Jul-18 09:41:52, 107070 W/h, 227.83 V, 1.01A
2015-Jul-18 09:51:52, 107122 W/h, 225.39 V, 1.01A
2015-Jul-18 10:01:52, 107172 W/h, 226.45 V, 1.01A
2015-Jul-18 10:11:52, 107222 W/h, 226.45 V, 1.01A
2015-Jul-18 10:21:52, 107272 W/h, 227.83 V, 1.01A
2015-Jul-18 10:31:52, 107322 W/h, 225.39 V, 1.01A
2015-Jul-18 10:41:52, 107372 W/h, 226.45 V, 1.01A
2015-Jul-18 10:51:52, 107424 W/h, 227.37 V, 1.01A
```

รูปที่ 4.12(ค)เป็นผลการบันทึกข้อมูล ของค่าพารามิเตอร์ต่างๆ ทุกๆ 10 นาที เป็นเวลา 3 วัน



20150511_log.txt
20150512_log.txt
20150519_log.txt
20150520_log.txt
20150521_log.txt
20150522_log.txt
20150523_log.txt
20150524_log.txt
20150527_log.txt
20150528_log.txt
20150531_log.txt
20150601_log.txt
20150602_log.txt
20150603_log.txt
20150604_log.txt
20150605_log.txt
20150606_log.txt
20150607_log.txt
20150611_log.txt
20150612_log.txt
20150613_log.txt
20150614_log.txt
20150615_log.txt
20150616_log.txt
20150617_log.txt
20150618_log.txt
20150623_log.txt
20150629_log.txt
20150630_log.txt
20150701_log.txt
20150702_log.txt
20150717_log.txt
20150718_log.txt
20150719_log.txt

รูปที่ 4.13เมื่อครบ 1 วัน หรือเวลา 00.00น การบันทึกค่าพารามิเตอร์ จะขึ้นไฟล์ใหม่เป็นวันถัดไป

ผลการทดสอบ จากรูปที่ 4.3 ถึงรูปที่ 4.5 เป็นการทดสอบ การบันทึกค่าของข้อมูล เมื่อตั้งค่าการแสดงผลของพลังงานไฟฟ้าทุกๆ 10 วินาที ตั้งค่าการแสดงผลของแรงดันไฟฟ้าทุกๆ 5 วินาที และตั้งค่าการแสดงผลของกระแสไฟฟ้าทุกๆ 10 วินาที ให้พารามิเตอร์ทุกตัว บันทึกผลทุกๆ 2 นาที โปรแกรมสามารถบันทึกตามเวลาที่ตั้งไว้ได้อย่างถูกต้อง และ ไม่มีความคลาดเคลื่อนเกิดขึ้น

ผลการทดสอบ จากรูปที่ 4.6 ถึงรูป 4.8 เป็นการทดสอบ การบันทึกค่าของข้อมูล เมื่อตั้งค่าการแสดงผลของพลังงานไฟฟ้าทุกๆ 5 วินาที ตั้งค่าการแสดงผลของแรงดันไฟฟ้าทุกๆ 5 วินาที และตั้งค่าการแสดงผลของกระแสไฟฟ้าทุกๆ 5 วินาที ให้พารามิเตอร์ทุกตัว บันทึกผลทุกๆ 3 นาที โปรแกรมสามารถบันทึกตามเวลาที่ตั้งไว้ได้อย่างถูกต้อง และ ไม่มีความคลาดเคลื่อนเกิดขึ้น

ผลการทดสอบ จากรูปที่ 4.9 ถึงรูป 4.13 เป็นการทดสอบ การบันทึกค่าของข้อมูล เมื่อตั้งค่าการแสดงผลของพลังงานไฟฟ้าทุกๆ 5 วินาที ตั้งค่าการแสดงผลของแรงดันไฟฟ้าทุกๆ 5 วินาที และตั้งค่าการแสดงผลของกระแสไฟฟ้าทุกๆ 5 วินาทีให้บันทึกค่าทุกๆ 10 นาที เป็นเวลา 3 วัน พบว่า แม้จะมีการตั้งระยะเวลาในการบันทึกที่นานขึ้นเป็นระดับ ชั่วโมง โปรแกรมก็ยังสามารถที่จะบันทึกค่าได้อย่างถูกต้อง และมีความคลาดเคลื่อนของเวลาในการบันทึกในระดับ วินาที และนาที และในการทดลองนี้เป็นการทดลองที่นานถึง 3 วัน จะเห็นได้ว่าโปรแกรมยังสามารถทำงานต่อเนื่องได้ แม้จะมีความคลาดเคลื่อนในส่วนของเวลาเกิดขึ้น แต่เนื่องจากระยะเวลาที่จำกัด จึงทำการทดลองสูงสุดไว้ที่ 3 วัน เนื่องจากปัญหาความร้อนที่อาจทำให้เกิดอันตรายต่อคอมพิวเตอร์ และห้องทดลอง

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลองวัดค่าพลังงาน แรงดันไฟฟ้า กระแสไฟฟ้า ของภาระที่ต่อเข้ากับระบบ หลอดไฟ โดยต่อผ่านมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล MITSUBISHI รุ่น S-X1และทำการบันทึกค่าและแสดงผลผ่านโปรแกรมที่ได้พัฒนาขึ้นมา เมื่อพิจารณาผลการทดลองพบว่า

ค่าพลังงานไฟฟ้าจากผลการทดลอง โดยขึ้นอยู่กับภาระต่างๆที่นำมาเชื่อมต่อเข้ากับระบบ ซึ่งค่าพลังงานไฟฟ้าที่โปรแกรมวัดได้นั้น มีค่าตรงกับหน้าจอของตัวมิเตอร์ แม้มีการเปลี่ยนแปลงของภาระแบบกะทันหัน ค่าพลังงานไฟฟ้าจะเปลี่ยนแปลงช้าเร็วขึ้นอยู่กับภาระโหลดและการตั้งค่าเวลาให้ค่าพลังงานไฟฟ้าแสดงผล การแสดงผลของโปรแกรมยังสามารถแสดงผลแบบติดตามได้อย่างต่อเนื่อง หากมีการตั้งค่าการบันทึกผลแบบละเอียดในระดับวินาที ก็จะสามารถบันทึกผลแบบติดตามได้

ค่าแรงดันไฟฟ้า ที่วัดได้จากการทดลอง จะมีปริมาณไม่เท่ากันในทุกการทดลอง เนื่องจากมีการเปลี่ยนค่าของภาระที่นำมาต่อให้มากขึ้นหรือลดลง และ ภาระชนิดหลอดไฟมีการขาดของไส้หลอดเกิดขึ้น แต่ก็ไม่เกิดปัญหากับการแสดงผล และบันทึกผลของโปรแกรม และในแรงดันไฟฟ้า มีการเปลี่ยนแปลงอยู่ตลอดเวลา

ค่ากระแสไฟฟ้าจากผลการทดลองจะขึ้นอยู่กับภาระโหลดถ้าภาระโหลดกินกระแสมากๆ ค่ากระแสไฟฟ้าจะแสดงบนหน้าจออย่างเห็นได้ชัด ส่วนใหญ่ค่าที่แสดงจะเป็นค่าเดียวกันเพราะว่าเมื่อต่อภาระโหลดหนึ่งครั้งโปรแกรมกว่าจะบึนทึกค่าอัตโนมัติไปเรื่อยๆ จนก็เราจะเพิ่มภาระโหลดหรือลดภาระโหลด

5.2 ประเมินผล

จากการดำเนินงานเทียบกับวัตถุประสงค์ได้ผลดังนี้

1. สามารถพัฒนาโปรแกรมสำหรับอ่านค่าและบันทึกผลสำหรับมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล รุ่น MITSUBISHI รุ่น S-X1 ได้ โดยมีการแสดงค่า พลังงานไฟฟ้าจริง แรงดันไฟฟ้า และกระแสไฟฟ้า
2. มีความรู้ความเข้าใจเกี่ยวกับระบบสื่อสาร การสร้างโปรโตคอล การถอดรหัสโปรโตคอลออกมา เพื่อทำการติดต่อสื่อสาร กับติมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอลได้
3. สามารถใช้องค์ความรู้ที่ได้จากในห้องเรียน ในการฝึกงาน มาใช้งานได้จริง ในการเชื่อมต่อระบบสื่อสารเข้าด้วยกัน

5.3 ปัญหา ข้อเสนอแนะและแนวทางแก้ไข

1. ปัญหาจากการต่อระบบเข้ากับภาระโหลด โหลดกินกระแสน้อยเกินไป ทำให้ในตอนแรกไม่สามารถวัดค่ากระแสไฟฟ้าได้ เพราะการทดลองแรกเราใช้คอมพิวเตอร์กับพัดลมเป็นภาระโหลดซึ่งกินกระแสน้อยมากซึ่งโปรแกรม ตั้งค่าหน่วยของการวัดค่ากระแส ไว้สองทศนิยมค่ากระแสไฟฟ้าจึงไม่แสดงที่หน้าจอแสดงผล ต้องทำการการเปลี่ยนขนาดภาระโหลดเป็นหลอดไฟ ให้มีขนาดการกินกระแสที่เยอะ ค่ากระแสไฟฟ้าจึงสามารถแสดงที่หน้าจอแสดงผลได้

2. ปัญหาจากการเชื่อมต่อระบบสื่อสารที่ผิดพลาด ทำให้อุปกรณ์บางส่วนได้รับความเสียหาย จึงต้องทำการซ่อมแซม ทำให้ชุดทดสอบที่ออกมา ไม่มีเรียบร้อยเท่าที่ควรจะเป็น และมีอันตรายจากกระแสไฟฟ้าอยู่มาก โดยเฉพาะจุดเชื่อมต่อ จึงต้องทำการซื้ออุปกรณ์ฉนวนมาใส่ชุดเชื่อมต่อเพื่อความปลอดภัย

3. สายสัญญาณที่ใช้ในการเชื่อมต่อระบบสื่อสาร มีความเปราะบางมาก ในการเชื่อมต่อ หรือเคลื่อนย้ายอุปกรณ์ทุกครั้ง ต้องระมัดระวังเป็นพิเศษ

5.4 แนวทางในการพัฒนาต่อไป

ผลที่ได้จากการทดลองในโครงการนี้ สามารถนำไปประยุกต์ใช้ในการศึกษาและเป็นแนวทางในการสร้างโปรโตคอล เพื่อ อ่านค่าจากมิเตอร์วัดพลังงานไฟฟ้าชนิดอื่นๆ และสามารถประยุกต์จากการวัดผลแบบ 1 เฟส ไปเป็น 3 เฟส ได้ เพื่อใช้ในทางอุตสาหกรรมต่อไป อีกทั้งยังสามารถพัฒนาต่อเพื่อไปใช้กับคอมพิวเตอร์ขนาดเล็กเช่น Raspberry pi เพื่อใช้เป็น Data logger นำไปติดตั้งกับอุปกรณ์ไฟฟ้าขนาดใหญ่ ที่ต้องการวัดค่าและเก็บข้อมูลได้ สามารถนำไปประยุกต์ทำเป็นแอปพลิเคชัน นำไปใช้ใน สมาร์ทโฟน เพื่อที่จะดูการเปลี่ยนแปลง หรือทำการแจ้งเตือน หากเกิดปัญหาจากค่าพลังงานไฟฟ้า โดยผู้ใช้สามารถควบคุมจากระยะไกลได้รูปแบบของภาษา ไพธอนเป็นภาษาที่ค่อนข้างง่ายต่อการเรียนรู้ โครงสร้างของภาษาไม่ซับซ้อนเข้าใจง่าย ซึ่งโครงสร้างภาษา ไพธอนจะคล้ายกับภาษา C มาก เพราะภาษาไพธอนสร้างขึ้นมาโดยใช้ภาษา C ทำให้ผู้ที่คุ้นเคยภาษา C อยู่แล้วใช้งานภาษา ไพธอนได้ไม่ยาก ตัวภาษาเองมีความยืดหยุ่นสูงทำให้การจัดการกับงานด้านข้อความและTextFileได้เป็นอย่างดี

เอกสารอ้างอิง

[1] www.thai-interelectric.com/(2558)โดยบริษัทMITSUBISHIELECTRICAUTOMATION (THAILAND) CO., LTD.เรื่องมิเตอร์วัดพลังงานไฟฟ้าแบบดิจิตอล MITSUBISHI รุ่น S-X 1[ออนไลน์] เข้าถึงได้จาก http://www.thai-interelectric.com/

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

[3] เอกสารการเชื่อมต่อโปรโตคอล โดยบริษัทMITSUBISHIELECTRICAUTOMATION (THAILAND) CO., LTD. เรื่องคู่มือการให้งานโปรโตคอล) Single-phase Electronic Meter SX1-A31N AMR Protocol Specifications ข้อมูลเพิ่มเติมอยู่ภาคผนวก

[4] http://th.softoware.net/ (2558)โดยบริษัท softoware.netเรื่องการใช้โปรแกรมDocklight [ออนไลน์] เข้าถึงได้จากhttp://th.softoware.net/apps/download-docklight-scripting-for-windows.html

[5] http://python.cmsthailand.com/basic_python.html (2558) โดยบริษัทpython.cmsthailand เรื่องเกี่ยวกับภาษาไพธอน[ออนไลน์]เข้าถึงได้จาก http://python.cmsthailand.com/basic_python.html

[6] http://software.thaiware.com/ (2558)โดยบริษัทsoftware.thaiware เรื่องเกี่ยวกับการใช้โปรรแกรม[ออนไลน์] เข้าถึงได้จากhttp://software.thaiware.com/10060-Notepad-Plus-Plus.html

[7] http://www.pythai.org/ (2558) โดยบริษัท pythai.org เรื่องเกี่ยวกับโปรแกรมGlade InterfaceDesignerGtk+2[ออนไลน์]เข้าถึงได้จาก http://www.pythai.org/doc/chapter1_intro.pdf

[8] http://home.npru.ac.th/โดยบริษัทhome.npru.ac.th เรื่องเกี่ยวกับการคำนวณ ค่า CRC [ออนไลน์]เข้าถึงไhttp://home.npru.ac.th/supakit/Slide_7122702/Error%20Detection.pdf

[9] www.asciitable.com/ โดยบริษัทasciitableเรื่องเกี่ยวกับการถอดรหัสโปรโตคอล [ออนไลน์] เข้าถึงได้จาก http://www.asciitable.com/

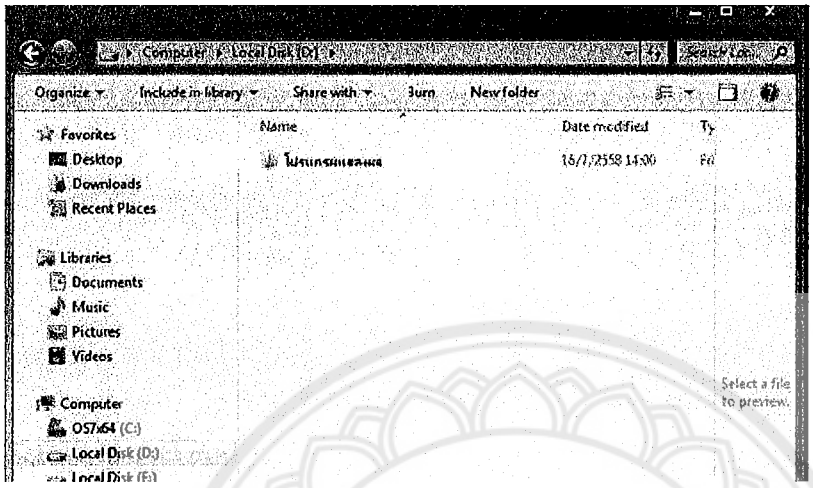
[10] www.lammertbies.nl โดยบริษัท lammertbies.nlเรื่องการคำวาน CRC [ออนไลน์] เข้าถึงได้จากhttp://www.lammertbies.nl/comm/info/crc-calculation.html

ภาคผนวก ก.

วิธีใช้งานโปรแกรมเบื้องต้น

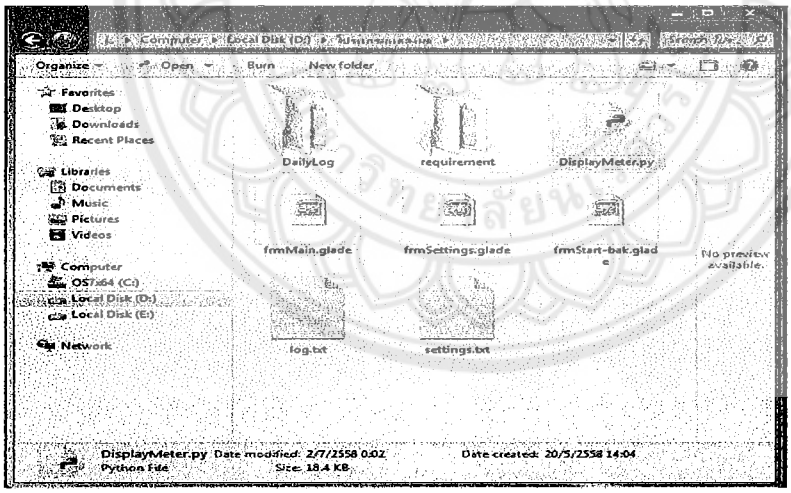
การใช้งานโปรแกรมเบื้องต้น

1. เปิดโปรแกรมแสดงผล เป็นไฟล์ที่บันทึกโปรแกรม



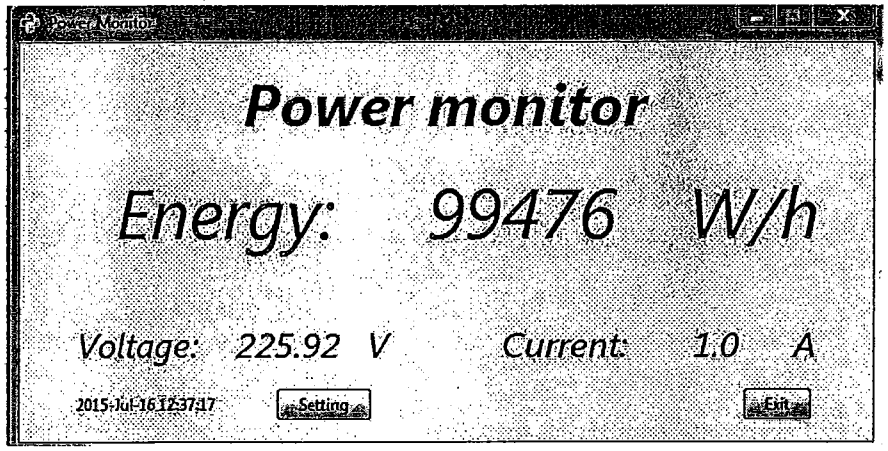
รูปที่ ก.1เปิดโปรแกรมแสดงผล เป็นไฟล์ที่บันทึกโปรแกรม

2. จะเห็นว่าในโฟลเดอร์ จะมีไฟล์ทั้งหมด 8 ไฟล์ ดับเบิ้ลคลิ๊กที่ไฟล์ DisplayMeter.py

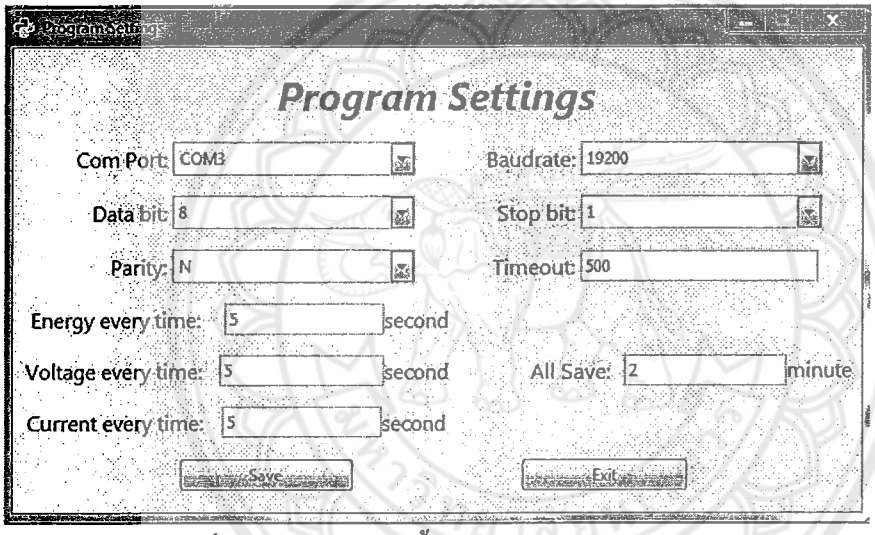


รูปที่ ก.2 เปิดโปรแกรมแสดงผล เป็นไฟล์ที่บันทึกโปรแกรม

3. เมื่อดับเบิ้ลคลิกที่ไฟล์ DisplayMeter.py จะปรากฏหน้าต่างแสดงผลดังรูปที่ ก.3 ซึ่งหน้าต่างแสดงผลนี้สามารถตั้งค่าเวลาในการบันทึกผลได้โดยกดปุ่ม Setting เพื่อตั้งเวลาใหม่ สามารถตั้งค่าเวลาให้พารามิเตอร์ต่างๆแสดงผลได้ อย่างเช่น ให้พลังงานไฟฟ้าแสดงค่าทุกๆ 5 วินาที แรงดันไฟฟ้าแสดงค่าทุกๆ 5 วินาที กระแสไฟฟ้าแสดงค่าทุกๆ 5 วินาที และให้บันทึกค่าพารามิเตอร์ต่างๆทุกๆ 2 นาที ดังรูปที่ ก.4

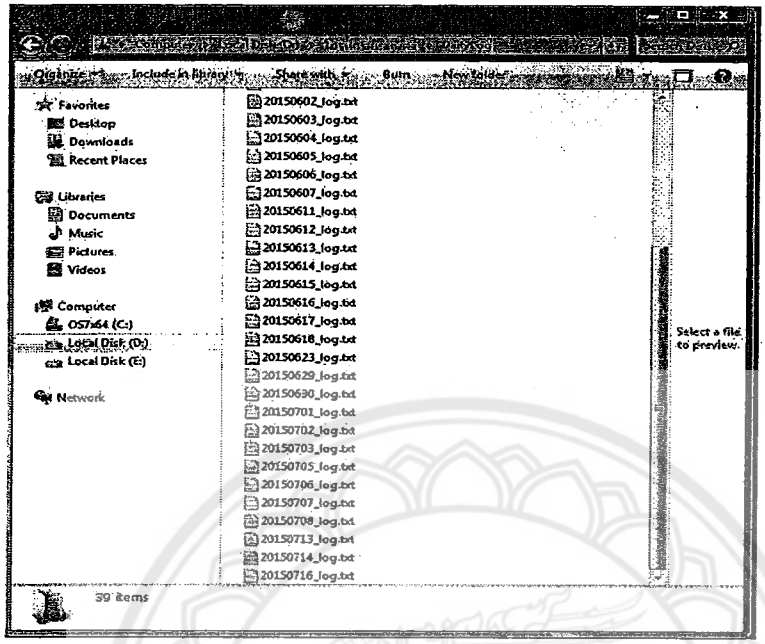


รูปที่ ก.3 หน้าต่างแสดงค่าพารามิเตอร์ต่างๆ

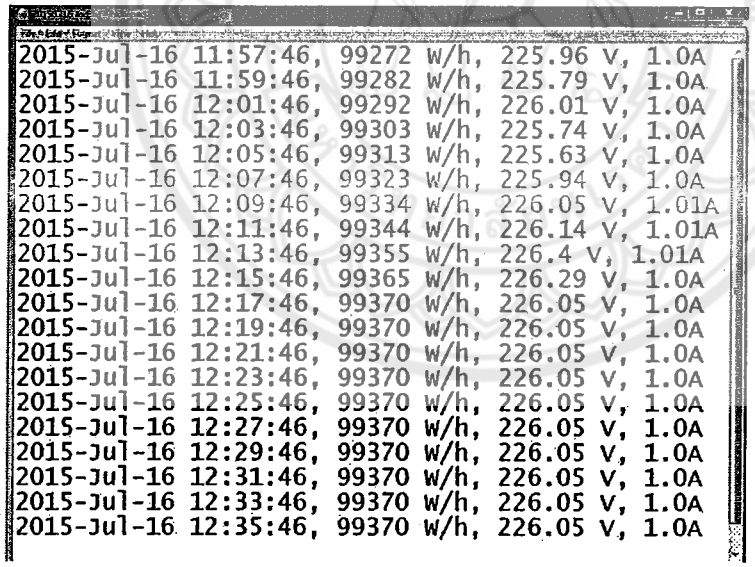


รูปที่ ก.4 หน้าต่างการตั้งค่าเวลาในการบันทึกค่าต่างๆ

4. สามารถเปิดไฟล์ที่โปรแกรมบันทึกค่าได้ที่ไฟล์ DailyLog ดังรูปที่ ก.2 จะพบไฟล์แต่ละวันของการบันทึกข้อมูลทั้งหมดดังรูปที่ ก.5 เมื่อเปิดไฟล์แต่ละวันขึ้นมาจะแสดงดังรูปที่ ก.6



รูปที่ ก.5 ไฟล์ที่บันทึกข้อมูลทั้งหมดในแต่ละวัน



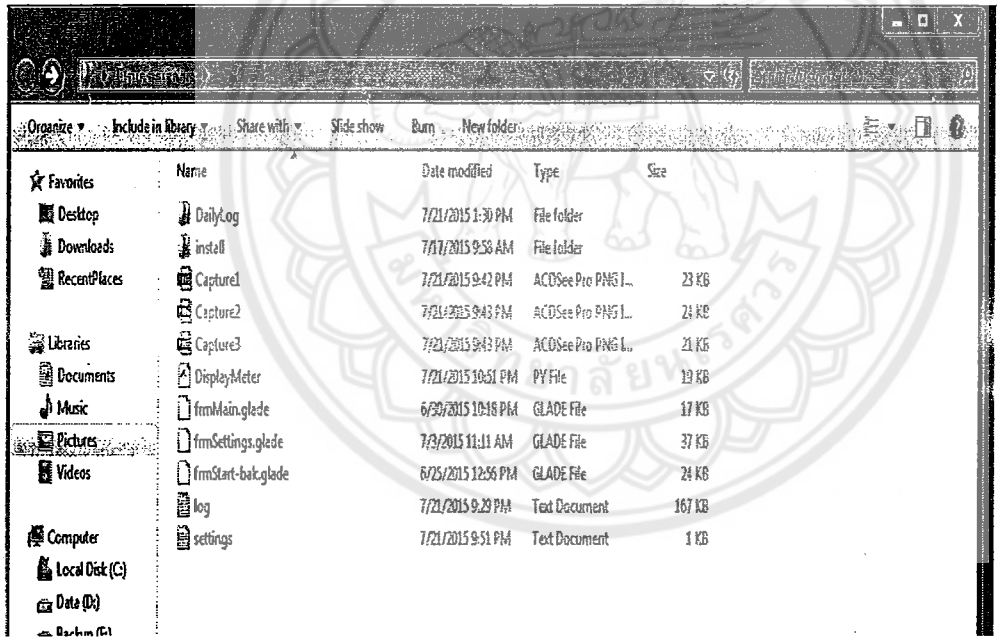
รูปที่ ก.6 ไฟล์ที่แสดงข้อมูลในการบันทึกค่าใน1วัน

การติดตั้งโปรแกรม

1. ลงโฟล์เดอร์ โปรแกรมแสดงผลดังรูปที่ ก.7

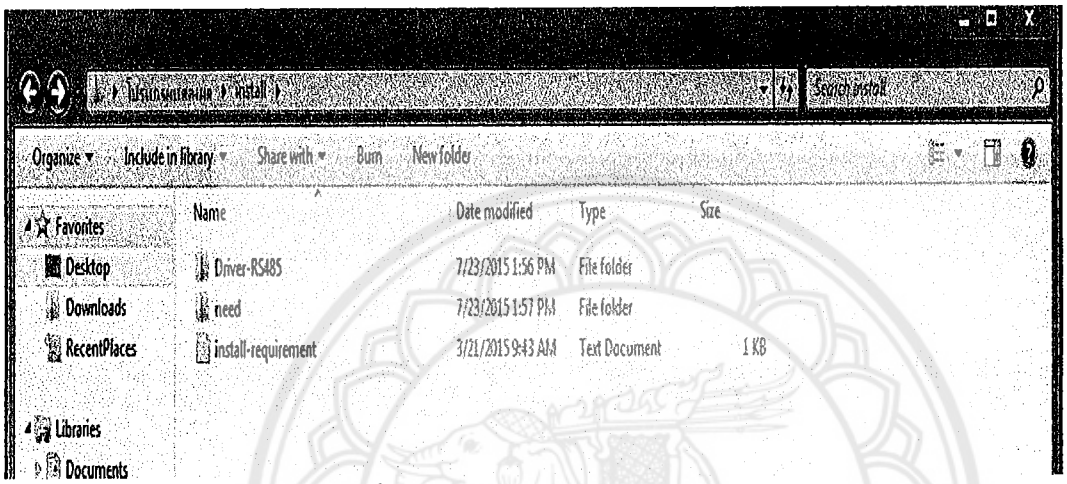
รูปที่ ก.7 โฟล์เดอร์โปรแกรมแสดงผล

2. เปิดโฟล์เดอร์ในโฟล์เดอร์จะมี 2 โฟล์เดอร์ 9 ไฟล์

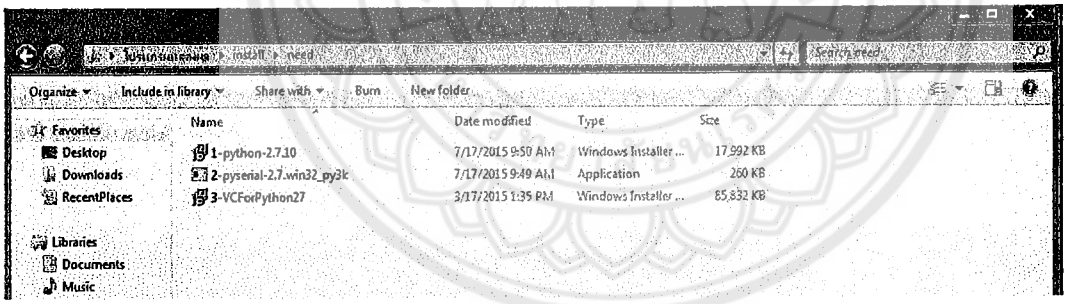


รูปที่ ก.8ข้อมูลในโฟล์เดอร์โปรแกรมแสดงผล

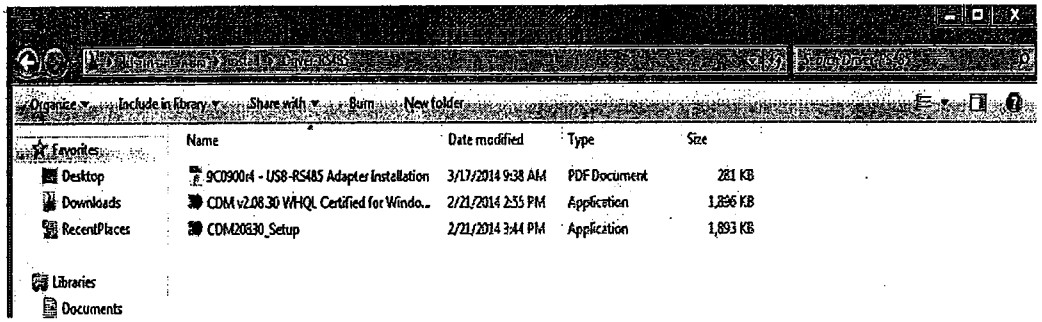
3. เปิดที่โฟล์เดอร์ install ดังรูปที่ ก.9 จะพบ โฟล์เดอร์need โฟล์เดอร์ Driver-RS485 และไฟล์install-requirement โฟล์เดอร์need เป็นโฟล์เดอร์ที่ติดตั้งภาษาไพธอนดังรูปที่ ก.10 ติดตั้งโดยดับเบิ้ลคลิกทุกไฟล์ก็จะสามารถติดตั้งภาษาพธอนได้แล้ว ต่อไปจะเป็นการติดตั้งไดร์เวอร์ RS-485 โดยเปิดโฟล์เดอร์ Driver-RS485 จะพบไฟล์3 ไฟล์ดังรูปที่ ก.11 ดับเบิ้ลคลิกเพื่อติดตั้งไดร์เวอร์ได้เลย



รูปที่ ก.9 ข้อมูลในโฟล์เดอร์ install



รูปที่ ก.10 ข้อมูลการติดตังภาษาไพธอน



รูปที่ ก.10ข้อมูลการติดตั้งไดร์เวอร์ RS485

ภาคผนวก ข.

รายละเอียดโค้ดคำสั่งในโปรแกรม

```
# ตัวแปรต่างๆในแบบ Array
serial_comport_list = {}
serial_baudrate_list = [
'2400',
'4800',
'9600',
'19200',
'38400',
'115200'
]
serial_databit_list = [
serial.FIVEBITS,
serial.SIXBITS,
serial.SEVENBITS,
serial.EIGHTBITS
]
serial_stopbit_list = [
serial.STOPBITS_ONE,
serial.STOPBITS_ONE_POINT_FIVE,
serial.STOPBITS_TWO
]
serial_parity_list = [
serial.PARITY_NONE,
serial.PARITY_EVEN,
serial.PARITY_ODD
]
log_save_schedule = [
'19:30:00',
'19:31:00',
'19:32:00'
]
# พาธสำหรับเก็บค่า Settings
```

```
settingFile = os.path.dirname(__file__) + "\settings.txt"

# พาธและชื่อไฟล์สำหรับบันทึกค่า log
logPath = os.path.dirname(__file__)
logFile = "log.txt"

# กำหนดค่าเริ่มต้นสำหรับพารามิเตอร์ต่างๆของ Comport และ เวลาในการแสดงผล
serial_comport_index = 0
serial_baudrate_index = 2
serial_parity_index = 0
serial_databit_index = 3
serial_stopbit_index = 0
serial_timeout = 1000
save_period = 5
energy_display_period = 5
voltage_display_period = 5
current_display_period = 5

# Label Object เจ็คสำหรับแสดงผลค่าต่างๆ (แสดงผลเป็นตัวอักษร)
None คือค่าเริ่มต้น
lblEnergy = None
lblVoltage = None
lblCurrent = None
lblDatetime = None
```

```
# คำสั่งสำหรับอ่านหรือเขียนข้อมูลจากเพาเวอร์มิเตอร์
cmdConnect =
"\x3A\x60\x81\x50\xB1\x82\x28\xD2\x53\xB4\xB8\x35\xD4\xD7\xCF\xD7\xC9\xD2\xC5\x53\x
50\xD2\xCF\xCA\xC5\xC3\xD4\xA9\x03\x9A\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x
23\x23\x23\x23\x23\x23\x23\xB3\x68\x03"
cmdDisconnect =
"\x3A\x60\x81\x50\xB1\x82\x28\xD2\x53\xB4\xB8\x35\xD4\xD7\xCF\xD7\xC9\xD2\xC5\x53\x
50\xD2\xCF\xCA\xC5\xC3\xD4\xA9\x03\x9A\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x
23\x23\x23\x23\x23\x23\x23\xB3\x68\x03"
cmdEnergy =
"\x3A\x60\x81\xD2\xB2\x82\x44\xB7\x28\xA9\x03\x93\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\xBC\x85\x03"
cmdVoltage =
"\x3A\x60\x81\xD2\xB2\x82\x44\x30\x28\xA9\x03\x14\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\xE8\x63\x03"
cmdCurrent =
"\x3A\x60\x81\xD2\xB2\x82\x44\xB2\x28\xA9\x03\x96\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\x23\
x23\x23\x23\x23\x23\x9A\xEB\x03"

# ตัวแปรสำหรับเก็บค่าที่อ่านได้จากมิเตอร์
read_energy = "####.##"
read_voltage = "####.##"
read_current = "####.##"
```

```
# ฟังก์ชันสำหรับการคำนวณ crccrit เอาไว้เช็คความถูกต้องของข้อมูลที่อ่านหรือเขียนจากมิเตอร์
defcrccitt(hex_string):
byte_seq = binascii.unhexlify(hex_string)
crc = crc16.crc16xmodem(byte_seq, 0xFFFF)
return '{:04x}'.format(crc&0xFFFF)

# ฟังก์ชันสำหรับการแสดงผลเวลา
defshowDatetime( dummy, delay ):
while True:
time.sleep(delay)
lblDatetime.set_text(time.strftime('%Y-%b-%d %H:%M:%S'))

# ฟังก์ชันสำหรับแสดงผลค่าที่อ่านได้ทั้งหมด
defshowReadingValue( dummy, delay ):
while not meterConnect():
pass
while True:
time.sleep(delay/3)
getEnergy()
time.sleep(delay/3)
getVoltage()
time.sleep(delay/3)
getCurrent()

# เริ่มติดต่อกับมิเตอร์
defmeterConnect():
global comport
try:

# ส่งคำสั่งในการติดต่อกับมิเตอร์ไปยังมิเตอร์
comport.write(cmdConnect)
```

```
# รอ 0.1 วินาที
time.sleep(0.1)

# อ่านค่าที่ส่งกลับมาจากมิเตอร์
ret = comport.read(51)

# เช็ค crcว่าตรงกับที่เครื่องส่งมาหรือไม่
crc = crccitt(ret[1:-3].encode('hex')).decode('hex')
if ((ret[2:3] == '\x06') and (ret[-3:-1] == crc[-1:] + crc[:1])):

return True # ถ้าตรงกันให้คืนค่า true
else:
return False # ถ้าไม่ตรงกันให้คืนค่า false

# ถ้าเกิดข้อผิดพลาดอย่างอื่นให้คืนค่า false
exceptValueError:
return False
exceptIOError:
return False
```

```
# เลิกเชื่อมต่อกับมิเตอร์
defmeterDisconnect():
global comport
try:
comport.write(cmdDisconnect)
time.sleep(0.1)
ret = comport.read(51)
crc = crccitt(ret[1:-3].encode('hex')).decode('hex')
if (ret[-3:-1] == crc[-1:] + crc[:1]):
return True
else:
return False
exceptValueError:
lblEnergy.set_text("@@@@.@@");
exceptIOError:
lblEnergy.set_text("@@@@.@@");

# อ่านค่าพลังงานไฟฟ้าจากมิเตอร์
defgetEnergy():
global comport
globalread_energy
try:

# ส่งคำสั่งในการติดต่อกับมิเตอร์ไปยังมิเตอร์
comport.write(cmdEnergy)

# รอ 0.1 วินาที
time.sleep(0.1)

# อ่านค่าที่ส่งกลับมาจากมิเตอร์
ret = comport.read(51)
```

```
# เช็ค crcว่าตรงกับที่เครื่องส่งมาหรือไม่
crc = crccitt(ret[1:-3].encode('hex')).decode('hex')
if (ret[-3:-1] == crc[-1:] + crc[:1]):
strEnergy = ''
forch in ret[6:15]:
if (ch> '\x80'):
ch = chr(int(ch.encode('hex'), 16) - 0x80)
strEnergy += str(ch)
#print strEnergy
#lblEnergy.set_text(str(int(strEnergy)));

# ถ้าค่า crcตรงกันให้คืนค่าที่อ่านได้กลับไป
read_energy = "%02d" % (int(strEnergy))
#return True
else:

# ถ้าค่า crcไม่ตรงกันให้คืนค่า @@@@.@@กลับไป
read_energy = "@@@@.@@"
#return False

# ถ้าเกิดข้อผิดพลาดอื่นๆให้คืนค่า ####.##
exceptValueError:
read_energy = "####.##"
exceptIOError:
read_energy = "####.##"

# อ่านค่าแรงดันไฟฟ้าจากมิเตอร์
defgetVoltage():
global comport
globalread_voltage
try:
while not(comport.readable() and comport.writable()):
```

```
time.sleep(0.01)
comport.write(cmdVoltage)
time.sleep(0.1)
ret = comport.read(51)
crc = crccitt(ret[1:-3].encode('hex')).decode('hex')
if (ret[-3:-1] == crc[-1:] + crc[:1]):
strVoltage = ''
forch in ret[6:11]:
if (ch> '\x80'):
ch = chr(int(ch.encode('hex'), 16) - 0x80)
strVoltage += str(ch)
#print strVoltage
read_voltage = str(int(strVoltage) * 0.01)
else:
read_voltage = "####.##"
exceptValueError:
read_voltage = "####.##"
exceptIOError:
read_voltage = "####.##"
```

 # อ่านค่ากระแสจากมิเตอร์

```
defgetCurrent():
global comport
globalread_current
try:
while not(comport.readable() and comport.writable()):
time.sleep(0.01)
comport.write(cmdCurrent)
time.sleep(0.1)
ret = comport.read(51)
crc = crccitt(ret[1:-3].encode('hex')).decode('hex')
if (ret[-3:-1] == crc[-1:] + crc[:1]):
```

```
strCurrent = ''
forch in ret[6:11]:
if (ch> '\x80'):
ch = chr(int(ch.encode('hex'), 16) - 0x80)
strCurrent += str(ch)
#print strCurrent
# current is read value * 0.01
read_current = str(int(strCurrent) * 0.01)
else:
read_current = "####.##"
exceptValueError:
read_current = "####.##"
exceptIOError:
read_current = "####.##"

# แสดงค่าพลังงานไฟฟ้าบนหน้าจอแสดงผล
defshowEnergy( dummy, delay ):
globalread_energy
globalenergy_display_period
while True:
counter = 0

# หากจำนวนวินาทียังไม่ถึงกำหนดให้วนลูปนับไปเรื่อยๆ
while (counter <energy_display_period):
t1 = time.time()
#print str(save_period)
delayTime = 1 - (t1 - int(t1)) + 0.1

# รอประมาณ 1 วินาที
time.sleep(delayTime)
counter += 1
```

```
# หากจำนวนวินาทีถึงกำหนดแล้วให้แสดงค่าพลังงานไฟฟ้า
lblEnergy.set_text(read_energy);

# แสดงค่าแรงดันบนหน้าจอ
defshowVoltage( dummy, delay ):
globalread_voltage
globalvoltage_display_period
while True:
counter = 0

# หากจำนวนวินาทียังไม่ถึงกำหนดให้วนลูปนับไปเรื่อยๆ

while (counter <voltage_display_period):

t1 = time.time()

#print str(save_period)
delayTime = 1 - (t1 - int(t1)) + 0.1
time.sleep(delayTime)
counter += 1

# หากจำนวนวินาทีถึงกำหนดแล้วให้แสดงค่าแรงดัน
lblVoltage.set_text(read_voltage);

# แสดงค่ากระแสบนหน้าจอ
defshowCurrent( dummy, delay ):
globalread_current
globalcurrent_display_period
while True:
counter = 0
# หากจำนวนวินาทียังไม่ถึงกำหนดให้วนลูปนับไปเรื่อยๆ
while (counter <current_display_period):
```

```
t1 = time.time()
#print str(save_period)
delayTime = 1 - (t1 - int(t1)) + 0.1
time.sleep(delayTime)
counter += 1
```

 # หากจำนวนวินาทีถึงกำหนดแล้วให้แสดงค่ากระแส

```
blCurrent.set_text(read_current);
```

 # บันทึกค่าที่แสดงบนหน้าจอลง log file

```
deflogSave( dummy, delay ):
globalsave_period
while True:
counter = 0
```

 # หากจำนวนวินาทียังไม่ถึงกำหนดให้วนลูปนับไปเรื่อยๆ

```
while (counter <save_period * 60):
t1 = time.time()
#print str(save_period)
delayTime = 1 - (t1 - int(t1)) + 0.1
time.sleep(delayTime)
counter += 1
#print(time.time(), "Log Save")
logVal = time.strftime('%Y-%b-%d %H:%M:%S') + ', ' + lblEnergy.get_text() + ' W/h, ' +
lblVoltage.get_text() + ' V, ' + lblCurrent.get_text() + 'A\n'
f = open(logPath + '\\' + logFile, "a")
f.write(logVal)
f.close()
if not os.path.exists(logPath + '\\DailyLog'):
os.makedirs(logPath + '\\DailyLog')
```

```
f = open(logPath + '\\DailyLog\\' + time.strftime('%Y%m%d') + "_" + logFile, "a")
f.write(logVal)
f.close()
```

 # ดึงชื่อพอร์ตทั้งหมดมาเก็บในรายการพอร์ต (serial_comport_list)

```
deflistComport():
i = 0
for s in serial.tools.list_ports_windows.comports():
serial_comport_list[i] = s[0]
i += 1
```

 # ถ้า I = 0 หมายถึงไม่พบซีเรียลพอร์ตต่ออยู่กับคอมพิวเตอร์ให้โชว์ข้อความ Error

```
if i == 0:
md = gtk.MessageDialog(None,
gtk.DIALOG_MODAL,
gtk.MESSAGE_ERROR,
gtk.BUTTONS_OK,
"No comport available!")
md.run()
md.destroy()
sys.exit()
return False
return True
```

 # บันทึกค่าการตั้งค่าต่างๆลงไฟล์log file

```
defwriteSettings():
f = open(settingFile, "w")
f.write(str(serial_comport_index) + '\n')
f.write(str(serial_baudrate_index) + '\n')
f.write(str(serial_databit_index) + '\n')
```

```
f.write(str(serial_stopbit_index) + '\n')
f.write(str(serial_parity_index) + '\n')
f.write(str(serial_timeout) + '\n')
f.write(str(save_period) + '\n');
f.write(str(energy_display_period) + '\n');
f.write(str(voltage_display_period) + '\n');
f.write(str(current_display_period) + '\n');
f.close()
return
```

 # อ่านค่าการตั้งค่าต่างๆมาเก็บในตัวแปร

```
defreadSettings():
globalserial_comport_index
globalserial_baudrate_index
globalserial_parity_index
globalserial_databit_index
globalserial_stopbit_index
globalserial_timeout
globalsave_period
globalenergy_display_period
globalvoltage_display_period
globalcurrent_display_period
```

 # ถ้าไม่เจอไฟล์การตั้งค่าให้เขียนค่าเริ่มต้นเข้าไปในไฟล์

```
if not os.path.isfile(settingFile):
writeSettings()
```

 # ทำการเปิดไฟล์การตั้งค่า (settings.txt)

```
f = open(settingFile)
try:
```

```
# อ่านค่า serial port name
serial_comport_index = int(f.readline().replace('\n', ''))
ifserial_comport_index<0:
serial_comport_index = 0


# อ่านค่า baudrate

serial_baudrate_index = int(f.readline().replace('\n', ''))
ifserial_baudrate_index<0:
serial_baudrate_index = 0

# อ่านค่า data bit length
serial_databit_index = int(f.readline().replace('\n', ''))
ifserial_databit_index<0:
serial_databit_index = 0

# อ่านค่า stop bit length
serial_stopbit_index = int(f.readline().replace('\n', ''))
ifserial_stopbit_index<0:
serial_stopbit_index = 0

# อ่านค่า parity
serial_parity_index = int(f.readline().replace('\n', ''))
ifserial_parity_index<0:
serial_parity_index = 0

# อ่านค่า timeout
strTemp = f.readline().replace('\n', '')
ifstrTemp == 'None':
serial_timeout = 10000
else:
```

```
serial_timeout = int(strTemp)

# อ่านค่า save_period
strTemp = f.readline().replace('\n', '')
ifstrTemp == 'None' or strTemp == '':
save_period = 5
else:
save_period = int(strTemp)

# อ่านค่า energy_display_period
strTemp = f.readline().replace('\n', '')
ifstrTemp == 'None' or strTemp == '':
energy_display_period = 5
else:
energy_display_period = int(strTemp)

# อ่านค่า voltage_display_period
strTemp = f.readline().replace('\n', '')
ifstrTemp == 'None' or strTemp == '':
voltage_display_period = 5
else:
voltage_display_period = int(strTemp)

# อ่านค่า current_display_period
strTemp = f.readline().replace('\n', '')
ifstrTemp == 'None' or strTemp == '':
current_display_period = 5
else:
current_display_period = int(strTemp)
finally:
f.close()
```

```
# ปิดไฟล์
f.close()
return

# เช็คว่าสามารถใช้ serial port ได้หรือไม่
defcheckComport():
global comport
try:
comport = serial.Serial(serial_comport_list[serial_comport_index],
serial_baudrate_list[serial_baudrate_index],
serial_databit_list[serial_databit_index],
serial_parity_list[serial_parity_index],
serial_stopbit_list[serial_stopbit_index],
serial_timeout)
#comport.close()
return True

# ถ้าใช้ไม่ได้ให้แจ้ง error
exceptValueError:
md = gtk.MessageDialog(None,
gtk.DIALOG_MODAL,
gtk.MESSAGE_ERROR,
gtk.BUTTONS_OK,
"Unknown error! please check.")
md.run()
md.destroy()
return False
exceptIOError:
md = gtk.MessageDialog(None,
gtk.DIALOG_MODAL,
gtk.MESSAGE_ERROR,
gtk.BUTTONS_OK,
```

```
"Can not open comport '" + serial_comport_list[serial_comport_index] + "' !")
md.run()
md.destroy()
return False
```

 # กำหนดให้ใช้ frmMain.gladeในการแสดงผล

```
self.gladefile = "frmMain.glade"
self.builder = gtk.Builder()
self.builder.add_from_file(self.gladefile)
self.builder.connect_signals(self)
self.window = self.builder.get_object("frmMain")
```

 # กำหนดว่าจะใช้ส่วนไหนของฟอร์มในการแสดงผลหรืออ่านค่า(ส่วนเชื่อมต่อกับGlade Interface Designer Gtk+2)

```
vbox1 = self.window.get_child()
hbox1 = vbox1.get_children()[3]
hbox2 = vbox1.get_children()[5]
hbox3 = vbox1.get_children()[7]
```

 # กำหนดส่วนที่จะแสดงผลค่าต่างๆ(ส่วนเชื่อมต่อกับGlade Interface Designer Gtk+2)

```
lblEnergy = hbox1.get_children()[1]
lblVoltage = hbox2.get_children()[1]
lblCurrent = hbox2.get_children()[4]
lblDatetime = hbox3.get_children()[0]
try:
```

 # เรียกใช้ฟังก์ชันต่างๆ

```
thread.start_new_thread(showDatetime, ( "string", 1))
thread.start_new_thread(showReadingValue, ("string", 3))
thread.start_new_thread(showEnergy, ("string", 1))
```

```
thread.start_new_thread(showVoltage, ("string", 1))
thread.start_new_thread(showCurrent, ("string", 1))
thread.start_new_thread(logSave, ( "string", 1))
except:

# หากไม่สามารถเรียกใช้ฟังก์ชันต่างๆได้ให้ขึ้นข้อความ error แล้วปิดโปรแกรม
print("Error: unable to start thread")
md = gtk.MessageDialog(None,
gtk.DIALOG_MODAL,
gtk.MESSAGE_ERROR,
gtk.BUTTONS_OK,
"System Error: unable to start thread!")
md.run()
md.destroy()
gtk.main_quit()
self.window.show()
defon_frmMain_destroy(self, object, data=None):
gtk.main_quit()

# ถ้ากดปุ่ม Settings ให้โชว์หน้าต่างการตั้งค่า frmSettings

defon_btnSettings_clicked(self, hbox3, data=None):
frmsettings = frmSettings()
frmsettings.run()

# ถ้ากดปุ่ม Exit ให้ออกจากโปรแกรม
defon_btnExit_clicked(self, hbox3, data=None):
gtk.main_quit()

# กำหนดให้ frmSettings.gladeเป็นฟอร์มสำหรับแสดงผลหน้าการตั้งค่า
self.gladefile = "frmSettings.glade"
self.builder = gtk.Builder()
```

```
# Add glade file for Settings page
self.builder.add_from_file(self.gladefile)
# Load window object from glade file
self.builder.connect_signals(self)
self.window = self.builder.get_object("frmSettings")

# กำหนดว่าจะใช้ส่วนไหนของฟอร์มในการแสดงผลหรืออ่านค่า(ส่วนเชื่อมต่อกับGlade Interface
Designer Gtk+2)
vbox1 = self.window.get_child()
hbox1 = vbox1.get_children()[3]
hbox2 = vbox1.get_children()[5]
hbox3 = vbox1.get_children()[7]
hbox6 = vbox1.get_children()[9]
hbox5 = vbox1.get_children()[11]
hbox7 = vbox1.get_children()[13]

# กำหนดส่วนที่จะแสดงผลค่าต่างๆ
mdlComport = gtk.ListStore(str)
mdlBaudrate = gtk.ListStore(str)
mdlDatabit = gtk.ListStore(str)
mdlStopbit = gtk.ListStore(str)
mdlParity = gtk.ListStore(str)
# Commport list
cmbCommport = hbox1.get_children()[1]
cmbCommport.set_model(model=mdlComport)
for i in range(len(serial_comport_list)):
cmbCommport.append_text(serial_comport_list[i])
cmbCommport.set_active(serial_comport_index)
# Baudrate list
cmbBaudrate = hbox1.get_children()[4]
cmbBaudrate.set_model(model=mdlBaudrate)
for i in range(len(serial_baudrate_list)):
```

```
cmbBaudrate.append_text(str(serial_baudrate_list[i]))
cmbBaudrate.set_active(serial_baudrate_index)
# Data bit list
cmbDatabit = hbox2.get_children()[1]
cmbDatabit.set_model(model=mdlDatabit)
for i in range(len(serial_databit_list)):
cmbDatabit.append_text(str(serial_databit_list[i]))
cmbDatabit.set_active(serial_databit_index)
# Stop bit list
cmbStopbit = hbox2.get_children()[4]
cmbStopbit.set_model(model=mdlStopbit)
for i in range(len(serial_stopbit_list)):
cmbStopbit.append_text(str(serial_stopbit_list[i]))
cmbStopbit.set_active(serial_stopbit_index)
# Parity combobox list

cmbParity = hbox3.get_children()[1]
cmbParity.set_model(model=mdlParity)
for i in range(len(serial_parity_list)):
cmbParity.append_text(str(serial_parity_list[i]))
cmbParity.set_active(serial_parity_index)
# Serial time out
txtTimeout = hbox3.get_children()[4]
txtTimeout.set_text(str(serial_timeout))
# Energy save period
txtEnergyDisplayPeriod = hbox6.get_children()[1]
txtEnergyDisplayPeriod.set_text(str(energy_display_period))
# Voltage save period
txtVoltageDisplayPeriod = hbox5.get_children()[1]
txtVoltageDisplayPeriod.set_text(str(voltage_display_period))
# Save period
txtSavePeriod = hbox5.get_children()[4]
```

```
txtSavePeriod.set_text(str(save_period))
# Current save period
txtCurrentDisplayPeriod = hbox7.get_children()[1]
txtCurrentDisplayPeriod.set_text(str(current_display_period))
self.window.show()
defon_frmSettings_destroy(self, object, data=None):
gtk.main_quit()

# ถ้ากดปุ่ม Saveให้ทำการบันทึกค่าการตั้งค่าลงในไฟล์ settings.txt
defon_btnSave_clicked(self, hbox4, data=None):
globalserial_comport_index
globalserial_baudrate_index
globalserial_databit_index
globalserial_stopbit_index
globalserial_parity_index
globalserial_timeout
globalsave_period
globalenergy_display_period
globalvoltage_display_period
globalcurrent_display_period

# ดึงค่าการตั้งค่าจากฟอร์มมาเก็บในตัวแปรต่างๆ
serial_comport_index = cmbCommport.get_active()
serial_baudrate_index = cmbBaudrate.get_active()
serial_databit_index = cmbDatabit.get_active()
serial_stopbit_index = cmbStopbit.get_active()
serial_parity_index = cmbParity.get_active()
serial_timeout = int(txtTimeout.get_text())
save_period = int(txtSavePeriod.get_text())
energy_display_period = int(txtEnergyDisplayPeriod.get_text())
voltage_display_period = int(txtVoltageDisplayPeriod.get_text())
```

```
current_display_period = int(txtCurrentDisplayPeriod.get_text())

# บันทึกค่าต่างๆลงในไฟล์ settings.txt

writeSettings()

# ปิดหน้าจอ frmSettings
self.window.hide()
#frmmain = frmMain()
#frmmain.run()

# ถ้ากดปุ่ม Exit ให้ออกจากโปรแกรม
defon_btnExit_clicked(self, hbox4, data=None):
readSettings()
self.window.hide()
#frmmain = frmMain()
#frmmain.run()

#เริ่มต้นโปรแกรม
# เช็คว่ามี serial port ต่ออยู่กับคอมพิวเตอร์หรือไม่
if not listComport():
gtk.main_quit()

# อ่านค่าการตั้งค่าของโปรแกรมจากไฟล์ settings.txt
readSettings()
checkComport()

# เช็คว่าสามารถใช้ serial port ได้หรือไม่
#if not checkComport():
# gtk.main_quit()
# เริ่มแสดงผลหน้าจอ frmMain
if __name__ == "__main__":
```

```
main = frmMain()
gtk.main()
```

ภาคผนวก ค.

รายละเอียดการถอดรหัสโปรโตคอล

ภาคผนวก ค.

รายละเอียดการถอดรหัสโปรโตคอล

Single-phase Electronic Meter

SX1-A31N

AMR Protocol Specifications

SPEC. NO. : **MED-A0028A**

Date :	February 8, 2012
Changed :	Rev.A, November 1st, 2012

Drawn :	K. Nuttapol
Checked :	
Approved :	

MITSUBISHI ELECTRIC AUTOMATION (THAILAND) CO., LTD

SX1 PROTOCOL SPECIFICATION

The communications messages are consist of two protocol. First is RS-485 protocol, its cover whole of message and the second is inner protocol we named MEATH protocol. Protocol layers are show in figure 1.1-1 and specifications of each are describe in following sections.

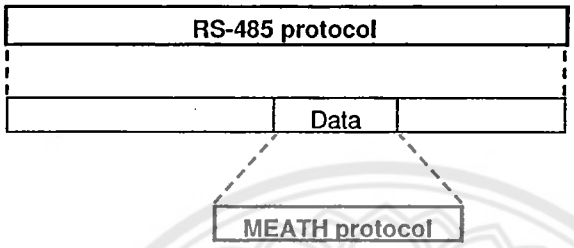


Figure 1.1-1 Protocol stack layer

1. RS-485 Protocol

1.1 Specifications

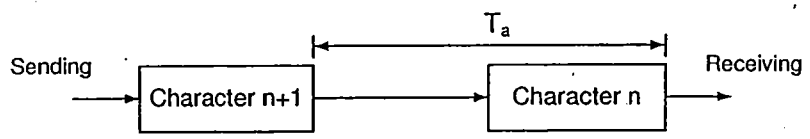
Table 1.1-1 RS-485 communication and RS-485 protocol specifications

Type of transmission	Asynchronous serial bit, Half duplex
Physical interface	RS – 485 (2wires) line driver.
Transmission speed	19,200 bps.
Protocol standard	RS-485 Protocol
Technique	Single Master / Multi Slaves
Packet size	51 bytes (Fixed)
Error detection	**CCITT CRC16 (cyclical redundancy check)** Polynomial = 0x1021 Non-reflect algorithm (MSB first) Initial value – 0xFFFF **Note :** Testing value = "123456789" CRC value = 0x29B1 See annex A for more details
Data format	**Binary** LSB first ST | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | SP ST : Start bit (1bit / logic 0) SP : Stop bit (1bit / logic 1) B0 – B7 : Binary data (8bits)

1.2 Communication Timing

1.2.1 Time between each character

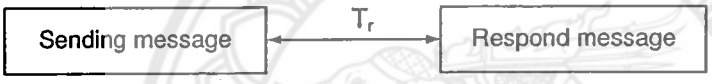
Every byte within a packet must precede the previous byte by take the time less than 1,500 ms. If time is more than, all of data will be flush out from the receive buffer.



$T_a < 1{,}500ms.$

1.2.2 Time between each packet (Response time)

The response time from a master request to a slave response or in case of sent data in multiple packet. Any new packet of data will be wait $200 \le t_r \le 1{,}500ms$ before send to receiver.



$200ms. \le T_r \le 1{,}500ms.$

S/W target for response waiting time = 210 ms.

S/W target for response timeout = 1,500 ms.

S/W target for inter character timeout = 1,400 ms.

1.3 Protocol Structure

RS-485 message structure for every message type (Send message, Receive message, and Receive connect message) are the same, this show in figure 1.3-1.

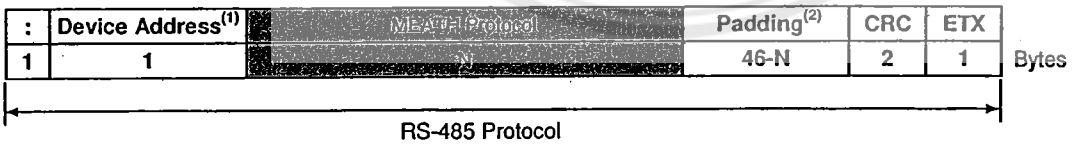


Figure 1.3-1 RS-485 message structure

(1)Device address (RS-485 address)

Data size : 1 byte
Data format : Binary
Default device address : See annex B

(2)Padding

Data size : 46-N byte
Data format : Fill in with # ASCII

Remark Special character ASCII code in communication message

: = 0x3A

= 0x23

ETX = 0x03

1.4 Protocol Analyze

For get data from received message, first analyze RS-485 protocol (outer protocol layer) to get MEATH protocol layer (inner protocol layer), and second analyze the MEATH protocol for get the target data. MEATH protocol will describe on section 2.

1.5 Error Detection

When CRC is incorrect, the meter will reject the command and will not reply to the master.

2. MEATH Protocol Specifications

2.1 Specifications

Table 2.1-1 MEATH protocol specifications

Protocol standard	MEATH Protocol (Custom)
Error detection	**Block Check Character (BCC)** See annex A for more detail
Data format	**ASCII** LSB first ST | B0 | B1 | B2 | B3 | B4 | B5 | B6 | P | SP ST : Start bit (1 bit/ logic 0) SP : Stop bit (1 bit/ logic 1) P : Parity bit (1 bit/ even) B0 – B6: Character data (7 bit)

2.2 Communication Sequence

As figure 2.2-1 sequence of message is starts from connect message to establish the communication. Then will read target data (such as kWh energy, line voltage and current) and when all target data reading finished program will send disconnect command to terminate the communication.

Sequence of message

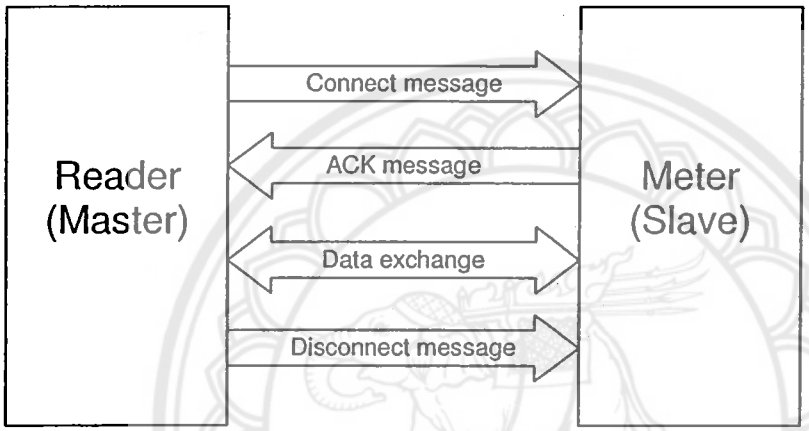


Figure 2.2-1 Message sequence

Sequence of data exchange

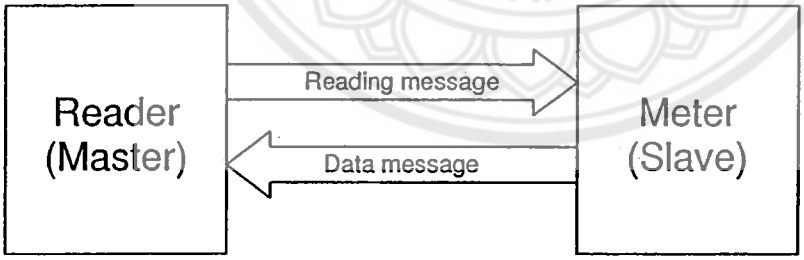


Figure 2.2-2 Data exchange sequence

The sequence of data exchange will start from the reader sends reading command message to the meter. After meter receives the command message it will send the data message to the reader. For reading command meter will send data message back one by one for each command.

2.3 Protocol Structure

MEATH protocol will start after "Device address" ("Device address" is on RS-485 protocol layer) and finish at "BCC", this show in figure 2.3-1.

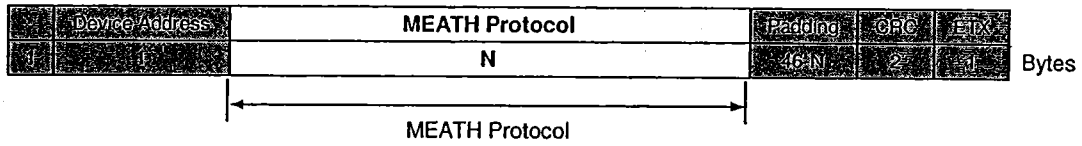


Figure 2.3-1 MEATH protocol structure

2.4 MEATH protocol message

2.4.1 Connect / Disconnect Message

2.4.1.1 Connect Message

Send : Connect message

SOH	P	1	STX	(	Data	)	ETX	BCC

Data size : 20 bytes

Data (Binary): D2 53 B4 B8 35 D4 D7 CF D7 C9 D2 C5 53 50 D2 CF CA C5 C3 D4

Receive : ACK message

ACK

Data size : 1 byte

Data (Binary) : 06

2.4.1.2 Disconnect message

Send : Disconnect message

SOH	B	0	ETX	BCC

2.4.2 Meter Information

2.4.2.1 Meter ID* reading

Send : Reading message

SOH	R	2	STX	0	0	(	)	ETX	BCC

Receive : Data message

STX	0	0	(	Data	)	ETX	BCC

Data size : 7 bytes

Data format : ASCII

Unit : -

*See annex B

2.4.3 Energy Value

2.4.3.1 kWh energy reading

Send : Reading message

SOH	R	2	STX	D	7	(	)	ETX	BCC

Receive : Data message

STX	D	7	(	Data	)	ETX	BCC

Data size : 9 bytes

Data format : ASCII

Unit : Wh

2.4.4 PQM Value

2.4.4.1 RMS line voltage reading

Send : Reading message

SOH	R	2	STX	D	0	(	)	ETX	BCC

Receive : Data message

STX	D	0	(	Data	)	ETX	BCC

Data size : 5 bytes

Data format : ASCII

Unit : 10 mV

2.4.4.2 RMS current reading

Send : Reading message

SOH	R	2	STX	D	2	(	)	ETX	BCC

Receive : Data message

STX	D	2	(	Data	)	ETX	BCC

Data size : 5 bytes

Data format : ASCII

Unit : 10 mA

See annex C for examples of data.

Annex A
BCC and CRC calculation

1. Block check character (BCC) calculation

1) Rule for generating the longitudinal parity block check character

1.1) Block check character

1.1.1 The block check character shall be composed of 7 bits plus an even parity bit

1.1.2 Each of the first 7 bits of the block check character shall be the modulo 2 binary sum of every element in the same bit 0 to bit 6 column of the successive character of the transmitted block.

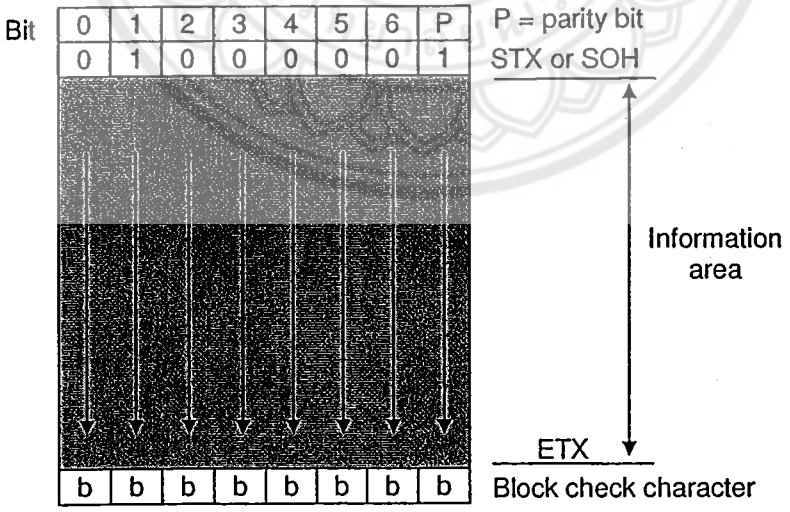
1.1.3 The longitudinal parity of each column of the block, including the block check character, shall be even.

1.2) Summation

1.2.1 The summation to obtain the block check character shall be started by the first appearance of either SOH (Start of Heading) or STX (Start of Text).

1.2.2 The starting character shall not be included in the summation.

1.2.3 If an STX character appears after the summation has been started by SOH, the STX character shall be included in the summation as if it were a text character.



2) Example of block check character calculation

Assume that the read command message has the stream of character like this;

HHU →

SOH	R	2	STX	D	7	(	)	ETX	BCC

Bit	0	1	2	3	4	5	6	P	P = parity bit	
	1	0	0	0	0	0	0	1	SOH	0x81
	0	1	0	0	1	0	1	1	R	0xD2
	0	1	0	0	1	1	0	1	2	0xB2
	0	1	0	0	0	0	0	1	STX	0x82
	0	0	1	0	0	0	1	0	D	0x44
	1	1	1	0	1	1	0	1	7	0xB7
	0	0	0	1	0	1	0	0	(	0x28
	1	0	0	1	0	1	0	1	)	0xA9
	1	0	0	0	0	0	0	0	ETX	0x03
1	1	0	0	1	0	0	1		BCC	0x93

From the figure, the highlight area is the field of character for generating block check character. Each bit of block check character is the longitudinal even parity of each column of the block, including the block check character, shall be even. So from the example the block check character from the calculation will be (93h).

2. Cyclical Redundancy Check (CRC-16) calculation

The Cyclical Redundancy Check (CRC-16) field is two bytes, containing a 16–bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The receiving device recalculates a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, that's means the communication error are occurs. Program will flush message and do not operate any operation and any response message.

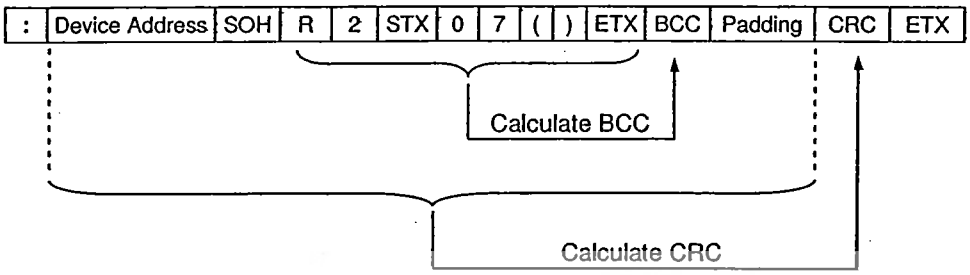
Placing the CRC into the Message

When the 16–bit CRC (two 8–bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte. For example, if the CRC value is 1241 hex (0001 0010 0100 0001) the CRC value will be place into message as follow.

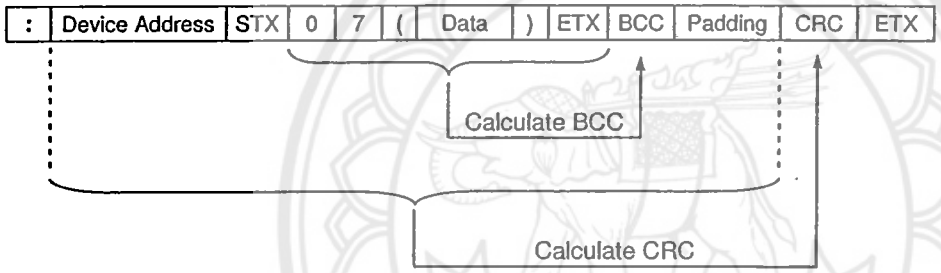
:	Device Address	Data Field	Reserve	CRC_{Lo}	CRC_{Hi}	ETX
				41	12	

BCC and CRC calculation field.

Send



Receive



Annex C
Example Message

1. Example of Connect message

Send: Connect message

3A 23 81 50 B1 82 28 D2 53 B4 B8 35 D4 D7 C1 B7 C9 D2 C5 53 50 D2
CB CA C5 C3 D4 A9 03 0A 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 04 50 03

Receive: ACK message

3A 23 06 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 CC D1 03

2. Example of Meter ID reading

Send : Reading message

3A 23 81 D2 B2 82 30 30 28 A9 03 60 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 96 DC 03

Receive: Data message

3A 23 82 30 30 28 *B7 39 30 30 B2 33 35* A9 03 B8 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 0A F2 03

Reading result: Meter ID is *7900235*

3. Example of kWh energy reading

Send: Reading message

```
3A 23 81 D2 B2 82 44 B7 28 A9 03 93 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 0B BD 03
```

Receive: Data message

```
3A 23 82 44 B7 28 30 30 30 30 32 39 33 34 39 A9 03 44 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 A7 CC 03
```

Reading result: Energy = *000029349* Wh

= **29.349 kWh**

4. Example of RMS line voltage reading

Send: Reading message

```
3A 23 81 D2 B2 82 44 30 28 A9 03 14 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 5F 5B 03
```

Receive: Data message

```
3A 23 82 44 30 28 B2 B1 B8 B2 B2 A9 03 4D 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 B7 19 03
```

Reading result: RMS line voltage = *21822* × 10mV

= **218.22 V**

5. Example of RMS current reading

Send: Reading message

3A 23 81 D2 B2 82 44 B2 28 A9 03 96 23 23 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 2D D3 03

Receive: Data message

3A 23 82 44 B2 28 ***30 30 30 B8 33*** A9 03 CF 23 23 23 23 23 23 23 23
23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
23 23 23 23 B3 1B 03

Reading result: RMS current = *00083* × 10mA
= 0.83 A