การอ่านหน่วยมาตรวัดพลังงานไฟฟ้าระบบหนึ่งเฟส ด้วยแลนแบบไร้สาย

นายศักดิ์ชัย ไวยลาภ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2547

ISBN 974-533-361-1

TELE-METER READING

SINGLE PHASE POWER METER

USING WIRELESS LOCAL AREA NETWORK

Mr. Sakchai Wiyalap

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Electrical Engineering

Suranaree University of Technology

Academic Year 2004

ISBN 974-533-361-1

การอ่านหน่วยมาตรวัดพลังงานไฟฟ้าระบบหนึ่งเฟสด้วยแลนแบบไร้สาย

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(รศ. ดร.สราวุฒิ สุจิตจร)
ประธานกรรมการ

(ผศ. ดร.อนันท์ อุ่นศิวิไลย์)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(อาจารย์ ดร.รังสรรค์ ทองทา)
กรรมการ

(รศ. น.ท. ดร.สราวุฒิ สุจิตจร)
รองอธิการบดีฝ่ายวิชาการ

(รศ. น.อ. ดร.วรพจน์ ขำพิศ)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ศักดิ์ชัย ไวยลาภ : การอ่านหน่วยมาตรวัดพลังงานไฟฟ้าระบบหนึ่งเฟสด้วยแลนแบบไร้สาย (TELE-METER READING SINGLE PHASE POWER METER USING WIRELESS LOCAL AREA NETWORK) อาจารย์ที่ปรึกษา :ผู้ช่วยศาสตราจารย์ ดร.อนันท์ อุ่นศิวิไลย์, 200 หน้า. ISBN 974-533-361-1

การใช้งานมาตรวัดพลังงานไฟฟ้าชนิดกิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำ ปัจจุบันมีปริมาณมาก การบันทึกหน่วยพลังงานต้องอาศัยบุคลากรที่ชำนาญในแต่ละพื้นที่ แต่จำนวนบุคลากรที่ทำหน้าที่นี้มีจำกัดจึงต้องใช้เวลาหลายวันในการรวบรวม ทำให้ข้อมูลหน่วยพลังงานในรอบเดือนที่ได้จากมาตรวัดแต่ละเครื่องมีความคลาดเคลื่อน ในวิทยานิพนธ์นี้ กล่าวถึงการพัฒนาระบบเครือข่ายของแลนแบบไร้สาย สำหรับการอ่านหน่วยพลังงานไฟฟ้าจากมาตรวัดพลังงานระบบหนึ่งเฟสโดยอัตโนมัติ โปรแกรมด้านอุปกรณ์รวมช่องสัญญาณใช้เคลไฟ 5 มาตรวัดพลังงานที่ใช้เป็นชนิดกิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำ เชื่อมโยงผ่านไมโครคอนโทรลเลอร์โดยใช้โปรแกรมไดนามิคซี 7.06 ใช้โพรโตคอล TCP/IP เพื่อกำหนดหมายเลข IP แอดเดรสที่เป็นชั้นซีและเป็นแบบเครือข่ายส่วนบุคคลให้กับมาตรวัดพลังงาน ข้อมูลที่ได้รับจัดเก็บไว้ในฐานข้อมูลของโปรแกรมแอคเซส 97 ผ่านระบบปฏิบัติการวินโดว์ ผลการทดสอบระบบการรับส่งข้อมูลหน่วยพลังงาน ที่ระยะทาง 11 เมตร และ 180 เมตร ระบบสามารถติดตามการใช้หน่วยพลังงานและบันทึกข้อมูลการใช้หน่วยพลังงานได้เองอัตโนมัติ

สาขาวิชาวิศวกรรมไฟฟ้า ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2547 ลายมือชื่ออาจารย์ที่ปรึกษา_____

SAKCHAI WIYALAP : TELE-METER READING SINGLE PHASE POWER METER USING WIRELESS LOCAL AREA NETWORK. THESIS ADVISOR : ASST. PROF. ANANT OONSIVILAI, Ph.D. 200 PP. ISBN 974-533-361-1

AUTOMATIC METER READING/TELE-METER READING

At present, many induction kilowatt-hour power meters are used. The skillful persons, they will be to storage energy units in local area, but they are limited that using more date. The data of energy units from each meter has incorrect on monthly basis. Reported by this thesis, describes the development of wireless LAN for automatic tele-meter reading single phase power meter. The program of concentrator uses Delphi 5. The meter, which is induction kilowatt-hour meter type, is interfaced into microcontroller using Dynamic C 7.06 program. The TCP/IP protocol defined as IP address of power meter is C class and private network. The received data will be stored at the Access 97 database via the window operating system. The test results indicated that the system could be received and transmitted data at distance 11 and 180 meters, it could monitor energy units and automatic stored data of energy units.

School of Electrical Engineering

Academic Year 2004

Student's Signature Sakchai W.

Advisor's Signature Anant O.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้พัฒนาขอกราบขอบพระคุณ บุคคล และกลุ่มบุคคลต่างๆ ที่ให้คำปรึกษา แนะนำ ช่วยเหลือ ทั้งในด้านวิชาการและการดำเนินงานพัฒนา ได้แก่

ผู้ช่วยศาสตราจารย์ ดร.อนันท์ อุ่นศิวิไลย์ อาจารย์ที่ปรึกษาวิทยานิพนธ์

รองศาสตราจารย์ ดร.สราวุฒิ สุจิตจร หัวหน้าสาขาวิชาวิศวกรรมไฟฟ้า อาจารย์ ดร.รังสรรค์ ทองทา อาจารย์ประจำสาขาวิชาวิศวกรรมโทรคมนาคม ผู้ช่วยศาสตราจารย์ ดร.กิตติ อัตถกิจมงคล และผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ศรีแก้ว อาจารย์ประจำสาขาวิชาวิศวกรรมไฟฟ้า สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ผู้จัดการ วีระ โภคนิภา และเพื่อนพนักงานการไฟฟ้าส่วนภูมิภาคอำเภอสีคิ้ว ที่ให้โอกาสในการลาศึกษา

ขอขอบคุณ เจ้าหน้าที่ประจำอาคารเครื่องมือ พี่ๆ เพื่อนๆ และน้องๆ บัณฑิตศึกษาทุกท่าน ที่ให้กำลังใจ ให้คำปรึกษามาโดยตลอด

ขอขอบคุณ คุณอัญชุลี รักด่านกลาง และคุณภัทรวรรณ สิทธิกวินกุล ที่ให้กำลังใจ ให้คำแนะนำที่ดีมาโดยตลอด

ขอขอบคุณ สถาบันวิจัยและพัฒนา ที่ให้ทุนสนับสนุนในการพัฒนา

ท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา ที่ให้การเลี้ยงดูอบรมและส่งเสริมการศึกษาเป็นอย่างดี จนทำให้ผู้พัฒนาประสบความสำเร็จในชีวิตมาตลอด

ศักดิ์ชัย ไวยลาภ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเป็นที่ทราบกันดีว่ามาตรวัดที่ใช้วัดหน่วยพลังงานไฟฟ้าเป็น ชนิด กิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำ (induction kilowatt-hour meter) ซึ่งเป็นที่นิยมใช้งานในหลายประเทศ ประเทศไทยมีการติดตั้งใช้งานมาตั้งแต่อดีตจนถึงปัจจุบัน ส่งผลให้มาตรวัดชนิดนี้มีปริมาณที่สูง การพิจารณาเปลี่ยนแปลงมาตรวัดระบบเดิมทั้งระบบให้เป็นระบบใหม่ที่ทันสมัยและมีความสามารถรับส่งหน่วยพลังงานไฟฟ้าได้เองอัตโนมัติจะต้องใช้งบประมาณที่สูงมากเช่นกัน

การจดหน่วยพลังงานไฟฟ้าจากมาตรวัดหน่วยพลังงานไฟฟ้ายังคงอาศัยแรงงานพนักงานของหน่วยงานที่รับผิดชอบ ที่มีความชำนาญในแต่ละพื้นที่เข้าไปจดหน่วยอยู่เป็นประจำทุกเดือนเช่นเดิม ทำให้เกิดต้นทุนที่มีแนวโน้มสูงขึ้นทุกปีตามภาวะความเจริญทางด้านเศรษฐกิจ อีกทั้งเป็นงานในลักษณะเดิม ซึ่งไม่ก่อให้เกิดการพัฒนาทักษะและความก้าวหน้าทางอาชีพแต่อย่างใด จากจำนวนมาตรวัดในแต่ละพื้นที่มีเป็นจำนวนมาก การจดหน่วยของพนักงานที่มีจำนวนจำกัดจึงต้องใช้เวลาหลายวัน ทำให้ข้อมูลหน่วยพลังงานที่ได้จากมาตรวัดแต่ละเครื่องมีระยะห่างของรอบการใช้พลังงานไม่คงที่อยู่เป็นประจำ การนำข้อมูลที่ได้จากการจดหน่วยมาใช้ในการวิเคราะห์ และการจัดการพลังงานไฟฟ้า จึงเป็นข้อมูลที่ยังมีความคลาดเคลื่อน

ปัจจุบันเทคโนโลยีที่ใช้ในการอ่านหน่วยพลังงานไฟฟ้าแบบอัตโนมัติมีด้วยกันหลายแบบ ตามตารางที่ 1.1 วัฒนา สันทัดพร้อม หัวหน้าแผนกพัฒนาการจัดการด้านการใช้ไฟฟ้า กองพัฒนาระบบไฟฟ้า การไฟฟ้าส่วนภูมิภาค (การสื่อสารระหว่างบุคคล, 30 สิงหาคม 2545) กล่าวว่าหน่วยงานได้มีการนำมาตรวัดพลังงานที่ใช้เทคโนโลยีทางด้านดิจิตอลเข้ามาทดลองใช้งาน รวมถึงการเพิ่มอุปกรณ์เสริมให้กับมาตรวัดชนิดจานเหนี่ยวนำ (induction disk) ให้สามารถอ่านหน่วยของพลังงานไฟฟ้าได้เองอัตโนมัติโดยอาศัยการส่งคลื่นพาห์ผ่านสายส่งกำลัง (power line carrier) เข้าหาตัวรวมช่องสัญญาณ (concentrator) ด้วยเทคนิคการเข้าข้อมูลที่แตกต่างกันไป ก่อนจะส่งข้อมูลนั้นผ่านโมเด็มเข้าสู่หน่วยงานย่อย

การนำสายส่งไฟฟ้ากำลังที่เดิมใช้ในการส่งพลังงานไฟฟ้ามาใช้ในการรับส่งข้อมูลหน่วยพลังงานไฟฟ้าร่วมด้วย ทำให้การออกแบบระบบให้มีความเร็วในการรับส่งข้อมูลสูง หรือให้มีความน่าเชื่อถือนั้น ทำได้ยากเพราะสายส่งไฟฟ้ากำลังเป็นตัวกลางที่มีการรบกวนรุนแรง มีการแทรกสอด

อย่างมากเมื่อโหลดมีการเปลี่ยนแปลงทำให้คาดเดาผลของการรับส่งข้อมูลทำได้ลำบาก (Hooijen, 1998) จึงทำให้ระบบนี้มีความเหมาะสมกับการใช้งานในบางพื้นที่ ที่มีการรบกวนน้อยและภาวะของอิมพีแดนซ์ที่เกิดจากการใช้กระแสไฟฟ้าไม่เปลี่ยนแปลงมากนัก

การส่งข้อมูลหน่วยพลังงานโดยอาศัยระบบเครือข่ายของโทรศัพท์ มาตรวัดพลังงานไฟฟ้าที่ติดตั้งในระบบจะต้องมีหมายเลขโทรศัพท์เป็นของตัวเอง ทำให้เกิดความต้องการช่องของสัญญาณที่ใช้ในการติดต่อเกิดขึ้นเป็นจำนวนมาก ซึ่งการอ่านหน่วยพลังงานไฟฟ้าส่วนใหญ่จะเกิดขึ้นเพียงเดือนละหนึ่งครั้ง แต่หน่วยงานที่เป็นเจ้าของมาตรวัดจะต้องสูญเสียค่าใช้จ่ายในการเช่าหมายเลขโทรศัพท์และค่าบำรุงรักษาตลอดทั้งเดือน

ในงานพัฒนาชิ้นนี้จึงได้มุ่งประเด็นไปที่การเพิ่มประสิทธิภาพให้กับมาตรวัดพลังงานไฟฟ้าชนิดจานเหนี่ยวนำที่เป็นแบบหนึ่งเฟสเดิม ให้สามารถอ่านหน่วยของพลังงานไฟฟ้าและบันทึกข้อมูลการใช้พลังงาน ในรอบเดือนได้เองอัตโนมัติโดยอาศัยการส่งผ่านข้อมูลหน่วยพลังงานบนระบบแลนแบบไร้สาย (wireless local area network) เข้าสู่ตัวรวมช่องสัญญาณ เพื่อคำนวณค่าการใช้พลังงานในรอบเดือน (การไฟฟ้าส่วนภูมิภาค, อัตราค่าไฟฟ้า, 2543) โดยอาศัยการทำงานร่วมกันระหว่าง มาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำระบบหนึ่งเฟส อุปกรณ์ในการตรวจจับการหมุนของจานเหนี่ยวนำ อุปกรณ์รับส่งข้อมูลระบบแลนแบบไร้สาย ไมโครคอนโทรลเลอร์และไมโครคอมพิวเตอร์ ซึ่งอุปกรณ์ที่ใช้ในงานพัฒนานี้มีแนวโน้มของราคาถูกลงตามปริมาณความต้องการใช้งาน ที่มีจำนวนเพิ่มขึ้นเรื่อยๆในปัจจุบัน

1.2 วัตถุประสงค์ของการพัฒนา

- เพื่อศึกษาหลักการของระบบสื่อสารที่ใช้ในการรับส่งข้อมูล
- เพื่อศึกษาเทคโนโลยีและการทำงานของระบบแลนและแลนแบบไร้สาย ในการส่งข้อมูล ตามมาตรฐาน IEEE 802
- เพื่อพัฒนาโปรแกรมรับส่งข้อมูลและฐานข้อมูล
- เพื่อพัฒนาระบบการอ่านหน่วยการใช้พลังงานไฟฟ้า สำหรับมาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ ระบบหนึ่งเฟส

1.3 ข้อตกลงเบื้องต้น

- ใช้โพรโตคอล transmission control protocol/internet protocol (TCP/IP) ในการรับส่งข้อมูลหน่วยการใช้พลังงานไฟฟ้า
- ใช้มาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ ระบบหนึ่งเฟส

- ใช้แลนแบบไร้สายในการส่งผ่านข้อมูลหน่วยการใช้พลังงานไฟฟ้า

1.4 ขอบเขตของงานพัฒนา

- ออกแบบและสร้างชุดเชื่อมโยงระหว่างไมโครคอนโทรลเลอร์กับมาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ ระบบหนึ่งเฟส
- ออกแบบและสร้างระบบต้นแบบในการรับส่งข้อมูลหน่วยการใช้พลังงานไฟฟ้า
- ออกแบบและพัฒนาโปรแกรมคอมพิวเตอร์ให้ทำหน้าที่รวบรวมข้อมูล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- เพิ่มประสิทธิภาพในการอ่านหน่วยการใช้พลังงานไฟฟ้า
- นำระบบแลนแบบไร้สาย มาใช้ในการอ่านหน่วยการใช้พลังงานไฟฟ้า
- เพิ่มประสิทธิภาพให้กับมาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ ระบบหนึ่งเฟสที่ใช้งานอยู่เดิม ให้ส่งหน่วยพลังงานได้เองแบบอัตโนมัติ
- ติดตามการใช้พลังงานไฟฟ้าของมาตรวัดพลังงานแต่ละเครื่องได้ตามต้องการ

1.6 การจัดรูปเล่มวิทยานิพนธ์

วิทยานิพนธ์นี้ประกอบด้วย 5 บท และ 4 ภาคผนวก บทที่ 1 เป็นบทนำ กล่าวถึงความสำคัญของปัญหา วัตถุประสงค์ และเป้าหมายของงานวิทยานิพนธ์ รวมทั้งขอบเขตของงาน ส่วนบทอื่นๆ ประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 2 กล่าวถึงหลักการทั่วๆไปของระบบสื่อสารแบบไร้สาย หลักการทำงานของระบบแลน รวมถึงรูปแบบของโพรโตคอลที่เกี่ยวข้อง

บทที่ 3 กล่าวถึงวิธีการและขั้นตอนในการออกแบบระบบ

บทที่ 4 กล่าวถึงผลที่ได้จากการทดสอบระบบ

บทที่ 5 เป็นบทสรุปและข้อเสนอแนะ

ภาคผนวก ก. แสดงรายละเอียดของโปรแกรมเดลไฟ (delphi) ที่ใช้ในการจัดเก็บข้อมูลและแสดงหน่วยของการใช้พลังงานไฟฟ้า

ภาคผนวก ข. แสดงรายละเอียดของโปรแกรมไดนามิค ซี (dynamic c) ที่ใช้ในการจัดรูปแบบของข้อมูล และรับการอินเตอร์รัฟจากมาตรวัดพลังงานไฟฟ้า

ภาคผนวก ค. แสดงรายละเอียดข้อมูลต่างๆ ของไมโครคอนโทรลเลอร์ และไอซี ที่ใช้งาน

ภาคผนวก ง. แสดงบทความที่ได้รับการตีพิมพ์เผยแพร่

ตารางที่ 1.1 แสดงประเภทของเทคโนโลยีการอ่านหน่วยพลังงานแบบอัตโนมัติ และผลการเปรียบเทียบ[1]

เทคโนโลยีการส่งข้อมูล	ข้อได้เปรียบ	ข้อเสียเปรียบ
วิทยุ (radio)	- ค่าใช้จ่ายต่ำ - การติดตั้งง่าย	- การสื่อสารทางเดียว
โทรศัพท์เรียกเข้า (dial-in telephone)	- สะดวกในการใช้งาน - ใช้งานได้ดีในกรณีที่จำนวนผู้ใช้โทรศัพท์น้อย - ประหยัดในกรณีที่อัตราค่าโทรศัพท์ไม่เปลี่ยนแปลงตามเวลาการใช้	- ไม่สามารถเรียกข้อมูลได้ตามต้องการ - ต้องใช้โทรศัพท์ของผู้รับบริการ
โทรศัพท์โทรออก (dial-out telephone)	- เรียกข้อมูลได้ตามต้องการ - ใช้งานได้ดีกับระบบขนาดใหญ่ที่มีหมายเลขโทรศัพท์ของผู้รับบริการจำนวนมาก	- ขึ้นอยู่กับเทคโนโลยีของระบบการสื่อสาร - ต้องใช้โทรศัพท์ของหน่ายงานที่ให้บริการ - อัตราค่าบริการสูง
โทรศัพท์แบบรังผึ้ง (cellular telephone)	- เหมือนระบบโทรศัพท์เรียกเข้า - ใช้งานได้ดีในพื้นที่ห่างไกล	- มีต้นทุนในการสร้างสถานีย่อย - ไม่มีการทดลองใช้งานจริง
ระบบวิทยุบีบอัดข้อมูล (packet radio system)	- การสื่อสารสองทาง - เรียกข้อมูลได้ตามต้องการ	- ระบบมีความซับซ้อน - มีค่าใช้จ่ายในการสร้างโครงสร้างพื้นฐาน
ระบบรังผึ้ง 900 MHz (900 MHz cellular system)	- เรียกข้อมูลได้ตามต้องการ - ครอบคลุมทุกพื้นที่	- อุปกรณ์มีราคาแพง - ค่าใช้จ่ายในการติดตั้งสูง

ตารางที่ 1.1 (ต่อ)

เทคโนโลยีการส่งข้อมูล	ข้อได้เปรียบ	ข้อเสียเปรียบ
คลื่นพาห์ผ่านสายส่งกำลัง	- อิสระจากระบบสื่อสารพื้นฐาน - ติดตั้งเครื่องวัดพลังงานเพิ่มได้ทุกเวลา	- อุปกรณ์ที่ติดตั้งที่หม้อแปลงมีราคาแพง - อัตราการรับส่งข้อมูลต่ำ

[1] หมายเหตุ “Design and cost analysis of an automatic meter reading system for Electricite’ du Liban”,Ghajar,Khalife and Richani,Utilities Policy 9,2000,193-205

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 บทนำ

การรับส่งข้อมูลในระบบการสื่อสารแบบไร้สาย เทคนิคการมอดูเลตมีหลายวิธีด้วยกัน จุดเด่นของแต่ละวิธีขึ้นอยู่กับกาลเวลา และปริมาณข้อมูลที่ต้องการส่ง รวมถึงจำนวนของอุปกรณ์สื่อสารที่ต้องการเข้าใช้ตัวกลางในการส่งร่วมกัน

การคิดค้นพัฒนารูปแบบของโพรโตคอลขึ้นมา ให้เหมาะสมกับวิธีการมอดูเลต และจำนวนปริมาณข้อมูลที่ต้องการรับส่ง ทำให้การขนถ่ายข้อมูลเกิดความปลอดภัย และมีความน่าเชื่อถือของข้อมูลมากขึ้น ซึ่งในระบบแลนแบบไร้สายก็เช่นเดียวกัน

2.2 ระบบการรับส่งข้อมูลในระบบสื่อสาร

ระบบแถบฐาน (base band) จะรับส่งสัญญาณดิจิตอลที่ไม่มีการมอดูเลต ซึ่งต้องใช้ความกว้างแถบที่กว้าง ทำให้ไม่สามารถแบ่งช่องสัญญาณได้ อัตราเร็วในการส่งโดยปกติประมาณ 1 ถึง 100 Mbps การเพิ่มพื้นที่ของการรับส่งข้อมูล หรือการส่งข้อมูลข้ามพื้นที่จะต้องอาศัยการทวนสัญญาณ

การพิจารณาขีดความสามารถของระบบ ใช้อัตราส่วนระหว่างจำนวนข้อมูลที่ผิดพลาดเทียบกับจำนวนข้อมูลหลัก (bit error rate) ถ้ามีค่าต่ำ แสดงว่าการรับส่งข้อมูลมีประสิทธิภาพสูง

ระบบแถบกว้าง (broad band) จะรับส่งสัญญาณที่มีการมอดูเลต การแบ่งความถี่ออกเป็นหลายความถี่แล้วส่งเข้าไปในระบบทำให้มีช่องในการรับส่งสัญญาณเพิ่มขึ้น

การพิจารณาขีดความสามารถของระบบ ใช้อัตราส่วนของสัญญาณหลักเทียบกับสัญญาณรบกวน (signal/noise) ถ้ามีค่ามาก แสดงว่าระบบการรับส่งสัญญาณมีประสิทธิภาพดีตามไปด้วย

การสื่อสารหรือการรับส่งข้อมูลในปัจจุบันจะอาศัยการทำงานของทั้งสองระบบ คือ ระบบแถบฐานนิยมใช้รับส่งข้อมูลภายในวงจรหรือภายในอุปกรณ์เอง ส่วนระบบแถบกว้างนิยมใช้ในการรับส่งข้อมูลระหว่างอุปกรณ์ที่อยู่ภายนอกที่มีระยะทางที่ไกลขึ้นซึ่งการรับส่งข้อมูลในระบบแลนแบบไร้สายก็เช่นเดียวกัน (ประสิทธิ์ ประพิณมงคลการ, 2536)

2.3 ระบบสื่อสารแบบไร้สาย

ระบบสื่อสารแบบไร้สาย (ลัญฉกร วุฒิสิทธิกุลกิจ, 2540) ส่วนใหญ่จะอาศัยระบบแถบกว้างในการรับส่งข้อมูล ซึ่งเป็นสัญญาณที่มีการมอดูเลตแล้ว ทำให้มีปัจจัยที่จะต้องพิจารณาหลายปัจจัยดังต่อไปนี้

2.3.1 การรับส่งข้อมูลแบบแบ่งเซลล์

การจัดแบ่งเซลล์ในการรับส่งข้อมูลจุดประสงค์เพื่อนำความกว้างของแถบความถี่ที่มีอย่างจำกัดกลับมาใช้งานอีก เพื่อเพิ่มปริมาณอุปกรณ์รับส่งข้อมูล และเพิ่มพื้นที่การให้บริการ ซึ่งจะต้องพิจารณาถึงจำนวนเซลล์และการจัดแบ่งความถี่ ที่จะทำให้อัตราส่วน co-channel interference (C/I) มีค่ามากที่สุด อีกอัตราส่วนหนึ่งที่ใช้บ่งบอกถึงผลกระทบของสัญญาณรบกวน คือ อัตราส่วน co-channel reuse (D/R)

กรณีที่ใช้แบบจำลองของเซลล์เป็นรูปหกเหลี่ยมซึ่งใกล้เคียงกับรูปวงกลม ดังแสดงในรูปที่ 2.1 ซึ่งจะมีประสิทธิภาพมากที่สุด สามารถใช้วิธีทางเรขาคณิตหาความสัมพันธ์ได้ ดังสมการ

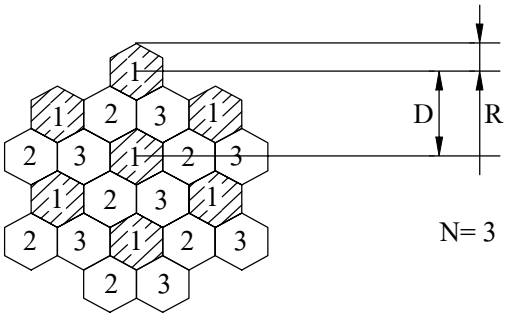
$$D = R\sqrt{3N} \tag{2-1}$$

หรือ $$D/R = \sqrt{3N} \tag{2-2}$$

เมื่อ D คือ ระยะระหว่างเซลล์ 2 เซลล์ ที่ใช้ความถี่เดียวกัน

N คือ จำนวนเซลล์ในหนึ่งกลุ่มเซลล์

R คือ รัศมีของเซลล์ที่ครอบคลุม



รูปที่ 2.1 ตัวอย่างการจัดกลุ่มเซลล์ของระบบเครือข่าย

หากเสาอากาศเป็นแบบกระจายคลื่นรอบทิศทาง (omni-directional antenna) เซลล์ทั้ง 6 เซลล์รอบข้างที่ใช้ความถี่เดียวกันจะส่งผลกระทบต่อการสื่อสารข้อมูลของเซลล์ตรงกลาง

$$\text{กำลังของสัญญาณที่ต้องการ} = K\frac{P}{R^{N}} \tag{2-3}$$

$$\text{กำลังของสัญญาณที่ไม่ต้องการจากเซลล์อื่น} = K\frac{6P}{D^{N}} \tag{2-4}$$

เมื่อ D คือ ระยะระหว่างเซลล์ 2 เซลล์ ที่ใช้ความถี่เดียวกัน

K คือ ค่าคงที่

N คือ ค่าคงที่ที่ขึ้นอยู่กับสภาพพื้นที่ ปกติจะเท่ากับ 4

P คือ กำลังของสัญญาณที่ส่งออกจากเซลล์

R คือ รัศมีของเซลล์ที่ครอบคลุม

$$C/I = \frac{\text{กำลังของสัญญาณที่ต้องการ}}{\text{กำลังของสัญญาณที่ไม่ต้องการจากเซลล์อื่น}} \tag{2-5}$$

$$= \frac{D^{4}}{6R^{4}} \tag{2-6}$$

จากค่า $D/R = \sqrt{3N}$ (2-7)

ดังนั้น $C/I = 1.5N^{2}$ (2-8)

จะเห็นได้ว่าค่า C/I จะดีขึ้นหากจำนวนเซลล์ในหนึ่งกลุ่มมีจำนวนมากขึ้น ในกรณีที่ N = 3 ตามรูปที่ 2.1 ซึ่งเป็นตัวอย่างของการนำจุดเข้าถึง (access point) ของระบบแลนแบบไร้สายมาจัดเป็นกลุ่มเซลล์ จะได้ค่า C/I เป็น 13.5 หรือ 11.3 dB

2.3.2 คุณสมบัติและผลกระทบของการส่งผ่านสัญญาณ

การสะท้อน (reflection) เกิดขึ้นเมื่อคลื่นเคลื่อนที่จากสื่อตัวกลางหนึ่งไปสู่อีกตัวกลางหนึ่ง โดยตัวกลางที่สองต้องมีพื้นที่ใหญ่กว่า เมื่อเทียบกับความยาวคลื่น ผลที่เกิดคือ คลื่นบางส่วนจะมีการสะท้อนออกและส่วนที่เหลือจะเคลื่อนที่เข้าสู่ตัวกลางที่สอง

การเลี้ยวเบน (diffraction) เกิดจากวัตถุที่กีดขวางมีขอบ ทำให้การรับส่งคลื่นสัญญาณไม่เป็นเส้นตรง เกิดการเลี้ยวเบนขึ้น

การแตกกระจาย (scattering) เกิดขึ้นเมื่อคลื่นเคลื่อนที่ชนวัตถุที่มีขนาดเล็ก เมื่อเทียบกับความยาวคลื่น เช่น เสาหรือพื้นผิวที่ขรุขระ

จากคุณสมบัติข้างต้นทำให้การส่งสัญญาณข้อมูลให้กับอุปกรณ์รับสัญญาณที่อยู่ใกล้กับวัตถุ หรือสิ่งก่อสร้างรอบๆ สัญญาณข้อมูลที่ได้รับจะเป็นสัญญาณที่แพร่มาจากหลายทิศทางและต่างเวลากัน ทำให้เกิดการแปรเปลี่ยนของสัญญาณอย่างมาก

2.3.3 การสูญเสียกำลังของสัญญาณในการส่งสัญญาณข้อมูล

การสูญเสียในอวกาศว่าง (free space path loss) เป็นการสูญเสียที่เกิดจากการแพร่กระจายของคลื่นด้วยมุมกว้างในบริเวณว่างเปล่าทำให้ความเข้มของกำลังคลื่นสัญญาณอ่อนตัวลงตามระยะทาง ซึ่งแสดงความสัมพันธ์ได้ ดังนี้

$$\frac{P_R}{P_T} = G_T G_R \left[\frac{\lambda}{4\pi d} \right]^2 \qquad (2\text{-}9)$$

d คือ ระยะระหว่างภาครับและภาคส่ง

G_R คือ อัตราขยายของสายอากาศด้านรับ

G_T คือ อัตราขยายของสายอากาศด้านส่ง

P_R คือ กำลังของสัญญาณที่รับได้

P_T คือ กำลังของสัญญาณที่ส่งออก

λ คือ ความยาวคลื่น (c/f เมื่อ c เป็นความเร็วแสง $3x10^8$ เมตร/วินาที , f เป็นความถี่ของสัญญาณ)

หรือ กำลังสูญเสีย

$$dB = 10 \log (P_T) - 10 \log (P_R)$$

$$= 32.44 + 20 \log (f) + 20 \log (d) - 10 \log (G_R) - 10 \log (G_T) \quad (2\text{-}10)$$

การสูญเสียกำลังจากพื้นผิว (plane path loss) ผลกระทบของพื้นผิวต่อการส่งสัญญาณข้อมูลจะมีผลทำให้เฟสของสัญญาณเปลี่ยนไป หากพิจารณาสัญญาณที่รับได้จะประกอบด้วย 2 ส่วน คือ สัญญาณที่ได้จากการส่งโดยตรง และสัญญาณที่ได้จากการสะท้อนของพื้นผิว ซึ่งสามารถแสดงความสัมพันธ์ได้ ดังนี้

$$\frac{P_R}{P_T} = G_T G_R \left[\frac{h_T h_R}{d^2} \right]^2 \quad (2\text{-}11)$$

d คือ ระยะระหว่างภาครับและภาคส่ง

G_R คือ อัตราขยายของสายอากาศด้านรับ

G_T คือ อัตราขยายของสายอากาศด้านส่ง

h_T คือ ความสูงของเสาอากาศทางด้านส่งเทียบกับพื้น

h_R คือ ความสูงของเสาอากาศทางด้านรับเทียบกับพื้น

หรือ กำลังสูญเสีย

$$dB = 10 \log (P_T) - 10 \log (P_R)$$

$$= 40 \log (d) - 20 \log (h_T) - 20 \log (h_R) - 10 \log (G_T) - 10 \log (G_R) \quad (2.12)$$

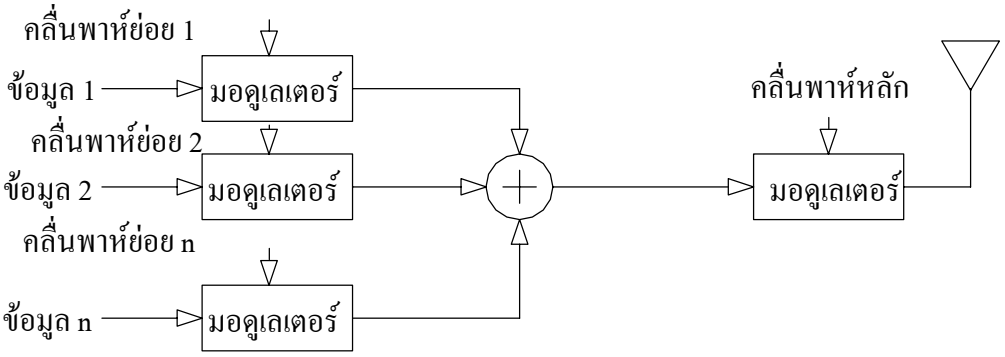
ในทางปฏิบัติระบบจะมีความซับซ้อนมาก เพราะมีองค์ประกอบต่างๆ มากมายในสภาพแวดล้อมจริง การจะหาค่าการสูญเสียกำลังได้จะต้องอาศัยการจำลองแบบจำลองจากสภาพการใช้งานจริง

2.4 การเข้าถึงหลายทางในการสื่อสาร

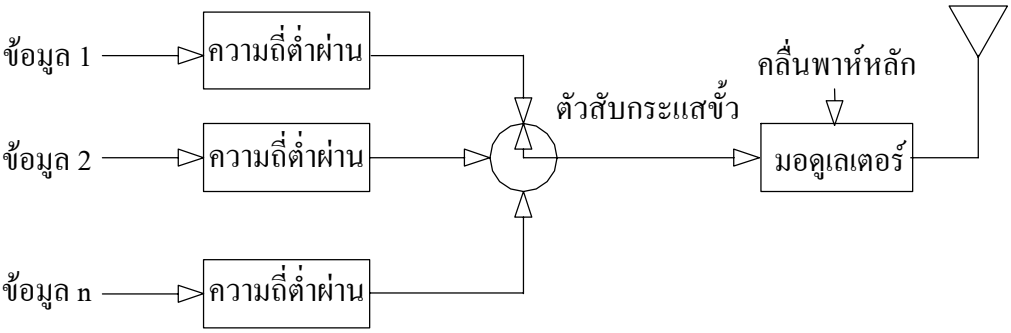
การเข้าถึงหลายทางในมุมมองของระบบมอดูเลตแบ่งออกได้เป็น 3 ชนิด คือ

2.4.1 การเข้าถึงหลายทางแบบแบ่งความถี่ (frequency division multiple access)

เป็นการแบ่งความกว้างแถบของความถี่เป็นช่องๆ ไม่เหลื่อมล้ำกัน ให้กับสัญญาณข้อมูล โดยแต่ละช่องจะมีคลื่นพาห์ย่อย มอดูเลตกับสัญญาณข้อมูลนั้น สัญญาณที่มอดูเลตแล้วจะรวมกันเป็นสัญญาณมัลติเพลกซ์ (composite multiplexed signal) สัญญาณที่ได้อาจจะส่งโดยตรง หรือนำไปมอดูเลตกับคลื่นพาห์หลักก่อนส่ง เทคนิคนี้ใช้ในการเข้าถึงได้หลายทางแบบแอนะลอกโดยจะเริ่มครอบครองช่องของสัญญาณช่องใดช่องหนึ่งไปตลอดเวลาจนกว่าจะสิ้นสุดการรับส่งข้อมูล ดังแสดงในรูปที่ 2.2 (ไพศาล สงวนหมู่ และ ยืน ภู่วรวรรณ, 2536)



รูปที่ 2.2 การจัดสรรช่องสัญญาณโดยใช้เทคนิคการเข้าถึงหลายทางแบบแบ่งความถี่



รูปที่ 2.3 การจัดสรรช่องสัญญาณโดยใช้เทคนิคการเข้าถึงหลายทางแบบแบ่งเวลา

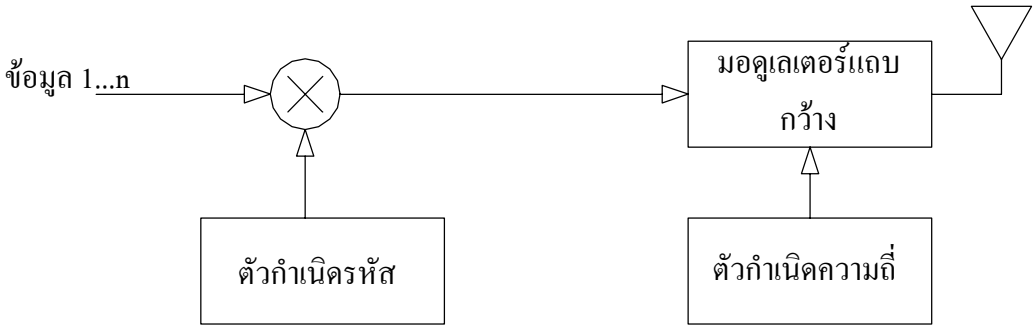
2.4.2 การเข้าถึงหลายทางแบบแบ่งเวลา (time division multiple access)

สัญญาณข้อมูลของแต่ละช่องจะถูกจัดกลุ่ม โดยแบ่งตามช่วงเวลาที่สม่ำเสมอ สัญญาณข้อมูลจากช่องอื่นก็เช่นเดียวกัน ณ เวลาหนึ่งเป็นสัญญาณข้อมูลจากช่องหนึ่งและจะตามด้วยสัญญาณข้อมูลช่องถัดไปจนกว่าจะได้สัญญาณข้อมูลครบทุกช่อง สัญญาณที่ได้ จะผ่านการมอดูเลตและมีลักษณะเป็นสัญญาณข้อมูลแบบอนุกรมส่งด้วยความเร็วสูง เทคนิคนี้ใช้ในการเข้าถึงหลายทางแบบดิจิตอล ดังแสดงในรูปที่ 2.3

2.4.3 การเข้าถึงหลายทางแบบแบ่งรหัส (code division multiple access)

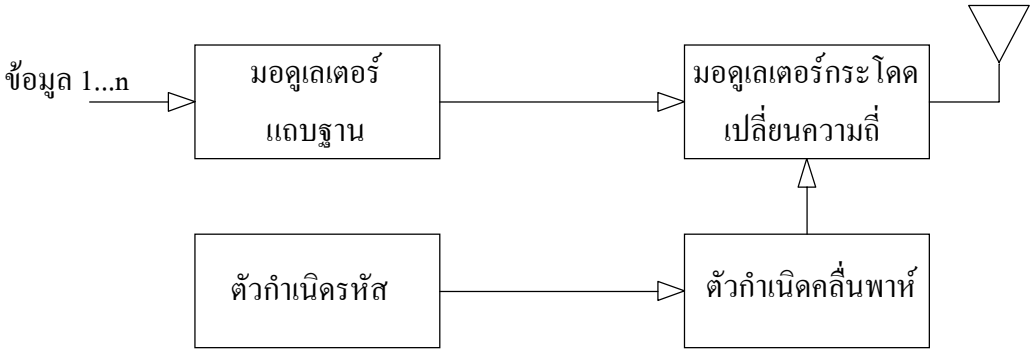
เป็นวิธีกระจายข้อมูลไปบนความถี่ที่มีความกว้างแถบกว้าง (Artecch, R., P., 1996) ทำให้สามารถรองรับข้อมูลได้หลายช่อง สามารถแบ่งได้เป็น 2 วิธีหลัก คือ

การเข้าถึงหลายทางแบบแบ่งรหัสลำดับโดยตรง (direct sequence code division multiple access) ซึ่งใช้ในระบบแลนแบบไร้สายที่ใช้ในการพัฒนานี้ สัญญาณข้อมูลแต่ละช่องจะถูกคูณด้วยรหัสที่มีความแตกต่างกัน ได้สัญญาณที่มีความยาวเพิ่มขึ้น นำไปมอดูเลตกับคลื่นพาห์แล้วส่งออก ทำให้ความกว้างแถบของสัญญาณข้อมูลมีความกว้างมากขึ้น ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 การจัดสรรช่องสัญญาณโดยใช้เทคนิคการเข้าถึงหลายทางแบบแบ่งรหัสลำดับโดยตรง

การเข้าถึงหลายทางแบบแบ่งรหัสกระโดดเปลี่ยนความถี่ (frequency hopping code division multiple access) ความกว้างแถบของความถี่หลักจะถูกหารด้วยจำนวนของช่องสัญญาณที่กำหนด ได้ความกว้างแถบของช่องสัญญาณสำหรับข้อมูลที่จะใช้ในระหว่างการส่ง ความถี่ของคลื่นพาห์จะเปลี่ยนแปลงตามจำนวนของช่องสัญญาณ โดยถูกกำหนดลำดับการเปลี่ยนแปลง ด้วยรหัสของผู้ใช้ ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 การจัดสรรช่องสัญญาณโดยใช้เทคนิคการเข้าถึงหลายทางแบบแบ่งรหัสกระโดดเปลี่ยนความถี่

2.5 การทำงานและโพรโตคอลที่ใช้ในระบบแลน

กล่าวถึงรายละเอียดของระบบแลนแบบเดิมที่ยังมีใช้งานอยู่ในปัจจุบันซึ่งเป็นพื้นฐานสำหรับอ้างอิงที่ระบบแลนแบบไร้สายยังคงต้องมี เพื่อให้สามารถใช้งานร่วมกับระบบแลนเดิมได้ ในช่วงของรอยต่อทางด้านเทคโนโลยี ระบบแลนเป็นการเชื่อมโยงเครื่องคอมพิวเตอร์เข้าด้วยกันเป็นข่ายงานภายในขอบเขตพื้นที่ที่จำกัด เพื่อใช้ทรัพยากรต่างๆ ร่วมกัน

คุณสมบัติทั่วไปของการสื่อสารระบบแลน

- อัตราการส่งข้อมูลได้สูง 0.1-100 Mbps
- ระยะทางการส่ง 0.1-25 km
- อัตราการผิดพลาดของข้อมูลต่ำ 10^{-8} - 10^{-11} (ไพศาล สงวนหมู่ และ ยืน ภู่วรวรรณ, 2536)

2.5.1 อุปกรณ์ที่ใช้ในระบบแลน

รีพีตเตอร์ (repeater) เป็นอุปกรณ์ทวนสัญญาณข้อมูลดิจิตอล เพื่อป้องกันการขาดหายไปของสัญญาณ เมื่อทำการส่งข้อมูลในระยะทางไกลๆ การใช้รีพีตเตอร์จึงช่วยเพิ่มพื้นที่การให้บริการของเครือข่าย

ฮับ (hub) ทำหน้าที่เป็นมัลติเพลกซ์เซอร์ นิยมใช้ในระบบแลน ที่มีผู้ใช้ในปริมาณไม่มากนัก

บริดจ์ (bridge) ออกแบบมาเพื่อใช้ติดต่อระหว่างระบบแลน 2 เครือข่ายที่เป็นประเภทเดียวกัน และใช้โพรโตคอลเหมือนกัน

เกตเวย์ (gateway) ทำหน้าที่หลักช่วยทำให้ระบบแลน 2 เครือข่าย หรือมากกว่า ที่มีลักษณะการเชื่อมต่อ และโพรโตคอลไม่เหมือนกัน ให้สามารถติดต่อกันได้เสมือนเป็นเครือข่ายเดียวกัน เช่น การรับส่งข้อมูลกันระหว่างระบบแลน 2 เครือข่าย หรือระบบแลนกับเครื่องคอมพิวเตอร์เมนเฟรม

เราเตอร์ (router) ทำหน้าที่จัดเส้นทางของข้อมูลในเครือข่าย ซึ่งสามารถทำการเชื่อมต่อเครือข่ายได้มากกว่า 2 เครือข่ายทั้งที่มีลักษณะเหมือนกัน หรือแตกต่างกันก็ได้ (ฉัตรชัย สุมามาลย์, 2542)

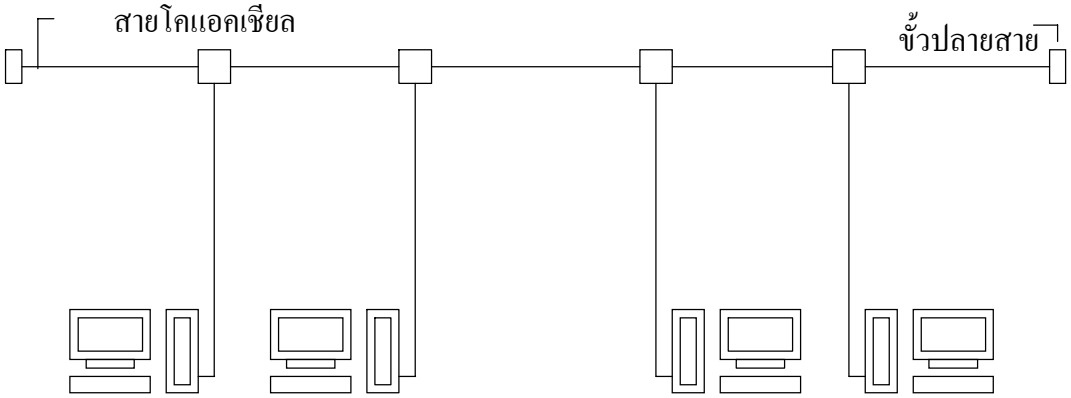
2.5.2 โทโพโลยีของระบบเครือข่าย

โทโพโลยีรูปดาว (star topology) ดังรูปที่ 2.6 การติดต่อระหว่างคอมพิวเตอร์จะถูกควบคุมด้วยคอมพิวเตอร์หลัก (host) การต่อเพิ่มคอมพิวเตอร์ใหม่ในระบบสามารถทำได้ง่าย โดยการต่อสายเข้าคอมพิวเตอร์หลักหรือฮับได้โดยตรง ผู้ดูแลระบบสามารถตรวจสอบ และกำหนดสถานะการทำงานของคอมพิวเตอร์ต่างๆ ได้ง่าย

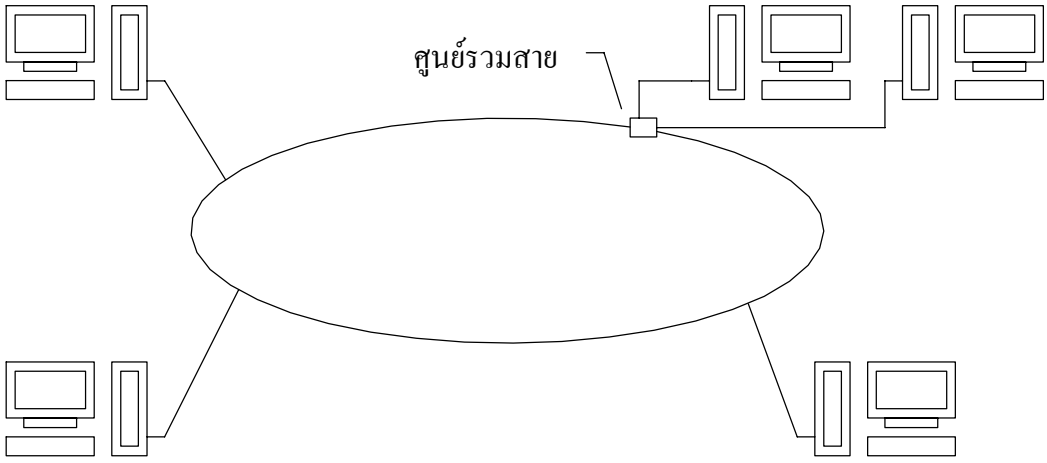
รูปที่ 2.6 โทโพโลยีรูปดาว

โทโพโลยีแบบบัส (bus topology) ดังรูปที่ 2.7 ประกอบด้วยทางเดินหลักของข้อมูลแล้วแตกสาขาไปหาคอมพิวเตอร์ย่อย ทำให้ประหยัดการใช้สายสัญญาณ แต่ผู้ดูแลระบบจะทำการตรวจสอบได้ยาก ความปลอดภัยของข้อมูลต่ำเนื่องจากข้อมูลทั้งหมดจะถูกส่งผ่านทางเดินหลักของข้อมูล

โทโพโลยีแบบวงแหวน (ring topology) ดังรูปที่ 2.8 ประกอบด้วยคอมพิวเตอร์หลายๆ ตัวเชื่อมต่อกันเป็นรูปวงแหวน ข้อมูลจะถูกส่งจากคอมพิวเตอร์แต่ละตัวไปในทิศทางเดียวกัน ผู้ดูแลระบบสามารถเลือกคอมพิวเตอร์ใดก็ได้เป็นจุดตรวจสอบ การต่อเพิ่มจะต้องปิดระบบ หรือใช้อุปกรณ์ศูนย์รวมสาย (wire centers) มาติดตั้งทำให้สะดวกขึ้น (ไพศาล สงวนหมู่ และ ยืน ภู่วรวรรณ, 2536)



รูปที่ 2.7 โทโพโลยีแบบบัส

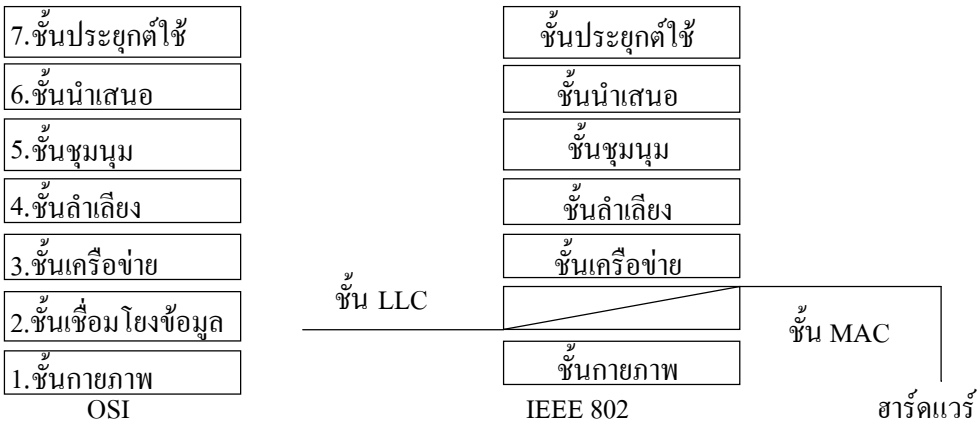


รูปที่ 2.8 โทโพโลยีแบบวงแหวน

2.5.3 มาตรฐานของระบบเครือข่าย

องค์การระหว่างประเทศเพื่อการทำเป็นมาตรฐาน (international standards of organization) ได้กำหนดมาตรฐานการสื่อสารคอมพิวเตอร์ระบบเปิด (open system interconnection :OSI) ที่ทำให้การเชื่อมต่อระบบเครือข่ายสามารถสื่อสารกันได้ในรูปแบบเดียวกัน (อภิชาต อัศวาดิศยางกูร, 2539) ประกอบด้วยชั้นต่างๆ 7 ชั้น ดังรูปที่ 2.9 ซึ่งแต่ละชั้นมีรายละเอียดและหน้าที่การทำงานดังนี้

ชั้นที่ 1 ชั้นกายภาพ (physical layer) เป็นชั้นล่างสุดของการติดต่อสื่อสาร ทำหน้าที่รับส่งข้อมูลจริงๆ ระหว่างคอมพิวเตอร์เครื่องหนึ่งกับอีกเครื่องหนึ่ง เป็นข้อกำหนดเกี่ยวกับฮาร์ดแวร์ที่ใช้ในระบบ



รูปที่ 2.9 เปรียบเทียบมาตรฐาน OSI กับมาตรฐาน IEEE 802

ชั้นที่ 2 ชั้นเชื่อมโยงข้อมูล (data link layer) ทำหน้าที่จัดการกับข้อมูลให้เป็นเฟรม โดยการเพิ่มเขต (field) ตำแหน่ง เขตตรวจสอบความผิดพลาดของข้อมูล การร้องขอ หรือยกเลิกการติดต่อเพื่อป้องกันไม่ให้การส่งข้อมูลเร็วเกินความสามารถของระบบ

ชั้นที่ 3 ชั้นเครือข่าย (network layer) กำหนดเส้นทางการเดินของข้อมูลระหว่างต้นทางปลายทาง เลือกเส้นทางที่สั้น ใช้เวลาน้อย ถ้าเส้นทางใดมีจำนวนข้อมูลมากเกินไป ก็จะเปลี่ยนเส้นทางให้ใหม่ รวมถึงการเป็นตัวเชื่อมระหว่างเครือข่ายต่างชนิดกัน

ชั้นที่ 4 ชั้นลำเลียง (transport layer) ทำการรวมกลุ่มของข้อมูล หรือแยกกลุ่มของข้อมูลออก เพื่อให้เข้ากับข้อกำหนดของชั้นที่ 3 และชั้นที่ 5

ชั้นที่ 5 ชั้นชุมนุม (session layer) ทำหน้าที่เชื่อมโยงระหว่างผู้ใช้งานกับเครื่องคอมพิวเตอร์

ชั้นที่ 6 ชั้นนำเสนอ (presentation layer) กำหนดรูปแบบของข้อมูล เพื่อให้ผู้ส่งและผู้รับเข้าใจความหมายของข้อมูลตรงกับการเข้ารหัส โครงสร้างข้อมูล การบีบอัดข้อมูล ให้อยู่ในรูปแบบมาตรฐานที่สามารถใช้ได้กับเครื่องคอมพิวเตอร์ทุกเครื่อง

ชั้นที่ 7 ชั้นการประยุกต์ใช้ (application layer) เป็นโปรแกรมประยุกต์ที่ใช้งานในระบบเครือข่าย

การพัฒนาระบบตามชั้นของมาตรฐานการสื่อสารคอมพิวเตอร์ระบบเปิด ทำให้การเพิ่มชั้นย่อย การแก้ไขชั้น ไม่จำเป็นต้องไปแก้ไขชั้นอื่นๆตาม สถาบัน IEEE จึงได้พัฒนามาตรฐานขึ้นมาโดยใช้ชั้นต่างๆ ตามมาตรฐานดังกล่าว เป็นตัวกำหนด ส่วนในชั้นเชื่อมโยงข้อมูล IEEE ได้มีชั้นย่อยเพิ่มขึ้นอีก 2 ส่วนคือ ชั้น logical link control (LLC) ทำหน้าที่จัดหรือแยกข้อมูลออกจากเฟรม พร้อมทั้งกำหนดหรือตรวจสอบเขตรหัสวนเหลือเฟือ (cyclic redundancy check : CRC) ขึ้นอยู่กับว่าเป็นการส่งหรือการรับข้อมูล และชั้น media access control (MAC) เป็นโพรโตคอลที่สนับสนุน

การทำงานในชั้นกายภาพ ให้ใช้ตัวกลางร่วมกันแต่แบ่งเวลาในการใช้ ด้วยวิธีการเข้าถึงโดยสุ่ม (random access) ทำให้อุปกรณ์ในการรับส่งข้อมูลทุกตัวมีโอกาสที่จะใช้ตัวกลางเวลาใดก็ได้

ระบบแลนตามมาตรฐาน IEEE 802.3 ได้มีการกำหนดรูปแบบเฟรมของข้อมูลที่ใช้ในการรับส่งข้อมูลดังแสดงในรูปที่ 2.10 ซึ่งประกอบด้วยส่วนต่างๆหลายส่วนด้วยกันดังนี้

ส่วนนำ	เริ่มต้น	แอดเดรสปลายทาง	แอดเดรสต้นทาง	ความยาว	สารนิเทศ	CRC
7 ไบต์	1 ไบต์	6 ไบต์	6 ไบต์	2 ไบต์	46 - 1500 ไบต์	4 ไบต์
เฟรม						

รูปที่ 2.10 รูปแบบของเฟรมข้อมูลตาม IEEE 802.3

ส่วนนำ (preamble) ประกอบด้วยข้อมูล 0 และ 1 สลับกันไปเรื่อยๆ เพื่อใช้อ้างอิงในการสร้างสัญญาณนาฬิกาที่จะซิงโครไนซ์

เริ่มต้น (start) เตือนให้เตรียมรับข้อมูลที่จะส่งมา โดยบิตสุดท้ายเป็น 11 ซึ่งส่วนใหญ่ข้อมูลในไบต์นี้จะเป็น 10101011

แอดเดรสปลายทาง (destination address) ตำแหน่งรับด้านปลายทาง

แอดเดรสต้นทาง (source address) ตำแหน่งส่งด้านต้นทาง หากจัดให้อยู่ในรูปของเลขฐานสิบหกจะได้จำนวน 12 หลัก 6 หลักแรกเป็นหมายเลขของผู้ผลิต 6 หลักสุดท้ายเป็นหมายเลขผลิตภัณฑ์ของผู้ผลิต

ความยาว (length) ระบุจำนวนของข้อมูลที่อยู่ในเขตของข้อมูล

สารนิเทศ (information) เป็นข้อมูลที่ต้องการส่ง มีค่าน้อยที่สุด 46 ไบต์ และสูงสุด 1500 ไบต์

ตรวจสอบรหัสวนเหลือเฟือ ใช้ในการตรวจสอบข้อมูลที่ได้รับ

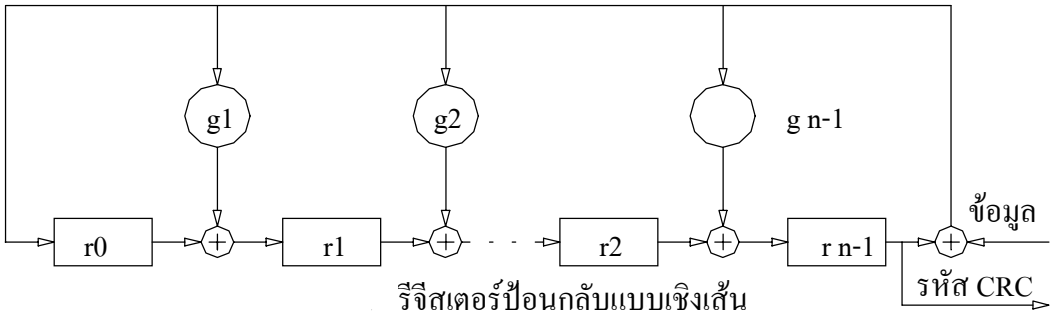
รหัสวนเหลือเฟือ (Scott, 1999) ใช้ในการตรวจสอบความผิดพลาดของการส่งข้อมูลที่เป็นกลุ่มข้อมูลมีขนาดยาวมากสามารถเข้ารหัสได้โดยการนำข้อมูลมาจัดให้อยู่ในรูปสมการโพลิโนเมียล

แทนด้วย M(x) คูณด้วย x^C เมื่อ c คือความยาวของรหัสตรวจสอบ รหัสตรวจสอบที่ใช้ในระบบเครือข่ายแลน แทนด้วย$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

หาร $x^CM(x)$ ด้วย G(x) เศษของการหารเป็นรหัสวนเหลือเฟือ แทนด้วย R(x) ให้นำไปต่อท้าย $x^CM(x)$ ซึ่งเป็นข้อมูลที่เลื่อนไปจำนวน c บิต จะได้สมการโพลิโนเมียลของบิตข้อมูลและรหัสวนเหลือเฟือที่ส่งออกไป แทนด้วย

$$T(x) = x^CM(x)+R(x) \qquad (2\text{-}13)$$

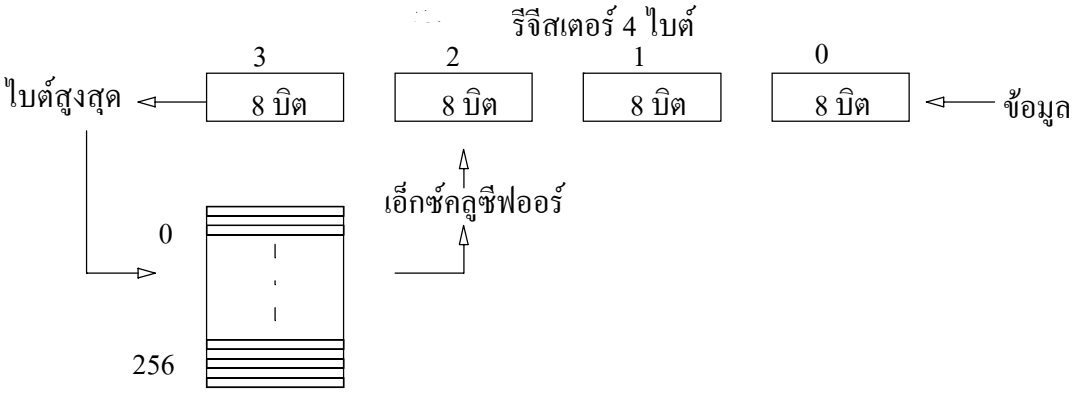
การเข้ารหัสวนเหลือเฟือทำได้ทั้งซอฟต์แวร์และฮาร์ดแวร์ แต่จะนิยมใช้ฮาร์ดแวร์มากกว่า เพราะมีความเร็วในการคำนวณสูงกว่า โดยอาศัยการเลื่อนบิตข้อมูลด้วยรีจิสเตอร์ และการบวกแบบมอดูโลทู (modulo-2) ด้วยเอ็กซ์คลูซีฟออร์ ดังแสดงในรูปที่ 2.11 ซึ่งเป็นการสร้างรหัสโดยอาศัยการทำงานแบบบิต



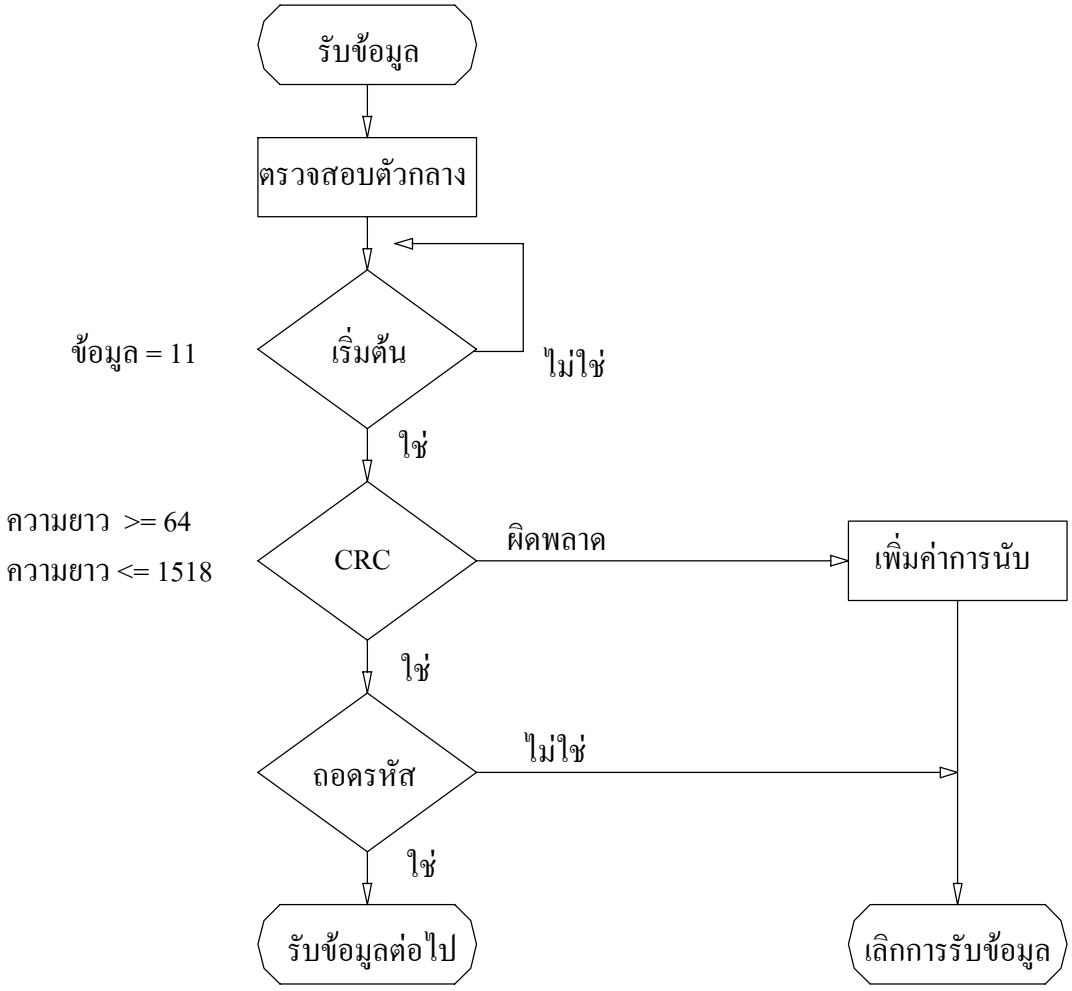
รูปที่ 2.11 การสร้างรหัสวนเหลือเฟือด้วยฮาร์ดแวร์แบบบิต

ปัจจุบันมีการพัฒนาใช้การเลื่อนข้อมูลเป็นไบต์ตามรูปที่ 2.12 โดยอาศัยตารางเข้ามาช่วยในการเก็บข้อมูลทำให้มีความใกล้เคียงกับวิธีการหารทางคณิตศาสตร์มากขึ้น ทำให้การเข้ารหัสวนเหลือเฟือของข้อมูลจำนวนมาก ทำงานได้เร็วขึ้น

carrier sense multiple access with collision detection (CSMA/CD) (Fairhurst, online, 2002) เป็นโพรโตคอลที่ใช้ควบคุมการเข้าใช้ตัวกลางในเครือข่ายร่วมกัน ในขบวนการรับเฟรมจะรับเป็นรหัสแมนเชสเตอร์ (manchester) ซึ่งจะกล่าวในรายละเอียดต่อไปในหัวข้อ 2.5.4 นำมาถอดรหัสให้เป็นเฟรมของข้อมูล ตรวจสอบความผิดพลาดของเฟรมข้อมูล และตำแหน่งปลายทาง ถ้าเฟรมข้อมูลและตำแหน่งมีความถูกต้อง จึงจะรับข้อมูลเข้ามาตามอัลกอลิทึมในรูปที่ 2.13

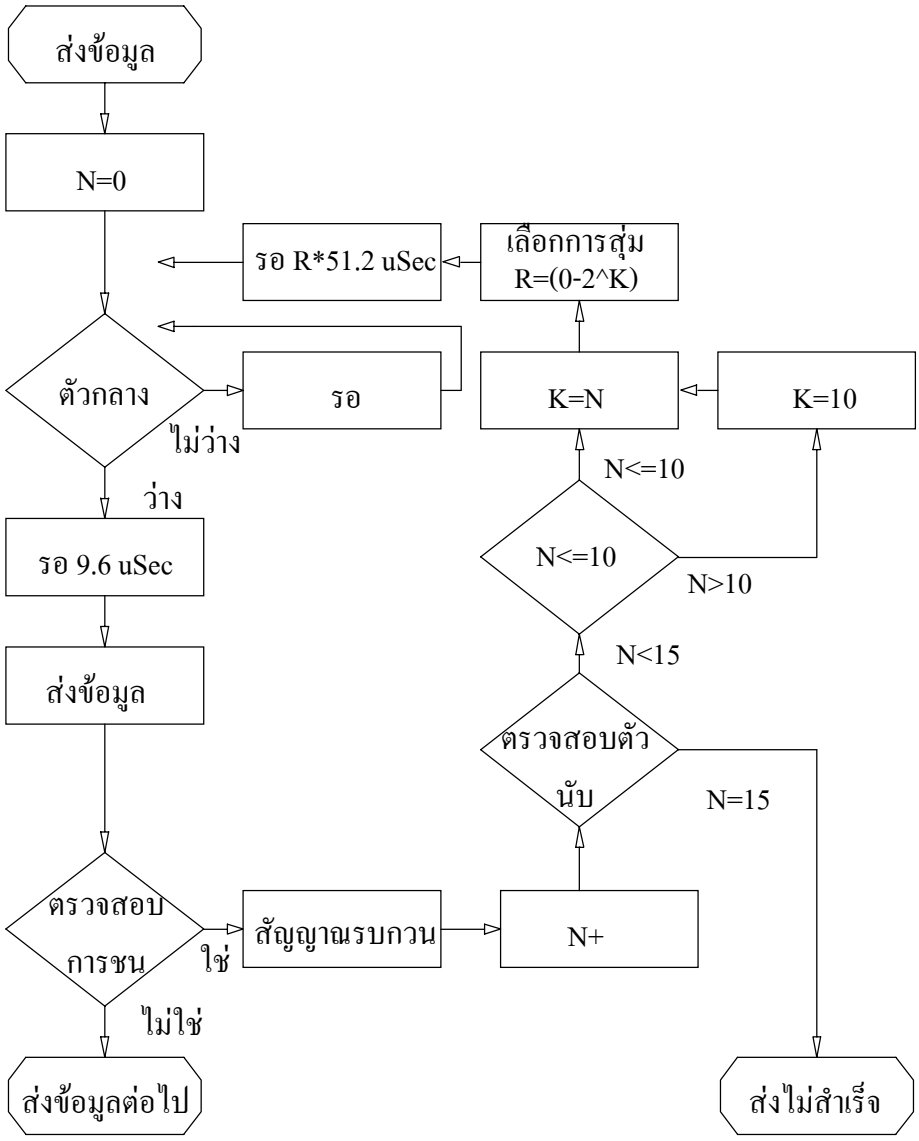


รูปที่ 2.12 การสร้างรหัสวนเหลือเฟือด้วยฮาร์ดแวร์แบบไบต์



รูปที่ 2.13 อัลกอลิทึมในการรับข้อมูลของ CSMA/CD

ส่วนอัลกอลิทึมในรูปที่ 2.14 เป็นขบวนการส่งข้อมูล จะมีการตรวจสอบเครือข่ายก่อน ถ้าเครือข่ายว่างจะหน่วงเวลาเพื่อให้ตำแหน่งที่จะรับข้อมูลเตรียมพร้อมและป้องกันการชนของข้อมูล หลังจากส่งข้อมูลแล้ว จะตรวจสอบการชนกันของข้อมูลตลอด ถ้าหากมีการชนของข้อมูลเกิดขึ้น ตำแหน่งที่ส่งข้อมูลจะส่งสัญญาณรบกวน เพื่อให้ด้านรับทราบว่าเกิดการชนกันของข้อมูลในเครือข่าย และตำแหน่งที่ส่งข้อมูลจะหน่วงเวลาการส่งข้อมูลโดยวิธีการสุ่ม กลไกในการตรวจสอบการชน ทำโดยการเปรียบเทียบข้อมูลที่ส่งออกไปกับข้อมูลที่รับเข้ามา ต้องเป็นข้อมูลที่เหมือนกันจึงจะสรุปได้ว่าไม่เกิดการชนกันของข้อมูลในระบบเครือข่าย



รูปที่ 2.14 อัลกอลิทึมในการส่งข้อมูลของ CSMA/CD

2.5.4 การเข้ารหัสแบบแมนเชสเตอร์

ข้อมูลที่ส่งเข้าในระบบเครือข่ายแลนต้องอยู่ในรูปแบบของรหัสแบบแมนเชสเตอร์ เพื่อหลีกเลี่ยงการส่งส่วนประกอบกระแสตรง (dc component) เข้าไปในตัวกลางตามรูปที่ 2.15 ซึ่งเป็นการสิ้นเปลืองพลังงานและทางด้านรับข้อมูลสามารถทราบความผิดพลาดของข้อมูลได้ในระดับหนึ่ง การเข้ารหัสสามารถทำได้ง่ายโดยการใช้เอ็กซ์คลูซีฟออร์ ขาสัญญาณอินพุตแรกจะเป็นข้อมูล ส่วนขาสัญญาณสุดท้ายจะเป็นสัญญาณนาฬิกา

รูปที่ 2.15 การเข้ารหัสแบบแมนเชสเตอร์

2.6 ระบบแลนแบบไร้สาย

การส่งข้อมูลโดยอาศัยตัวกลางในการส่งที่เป็นสัญญาณความถี่ ความผิดพลาดของข้อมูลและการถูกจารกรรมข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา ยิ่งมีผู้ใช้มากขึ้นความผิดพลาดและโอกาสการถูกจารกรรมก็มีมากขึ้นตามไปด้วยเช่นกัน การนำเอาโพรโตคอล MAC เข้ามาใช้งาน เพื่อให้การรับส่งข้อมูลมีความผิดพลาดต่ำ การนำตำแหน่งแอดเดรสของ MAC เข้ามาช่วยในการกลั่นกรอง รวมถึงการนำ wired equivalent privacy (WEP) เข้ามาช่วยในการรักษาความปลอดภัยของข้อมูล ทำให้สามารถเพิ่มความน่าเชื่อถือขึ้นได้

2.6.1 มาตรฐานของระบบแลนแบบไร้สาย

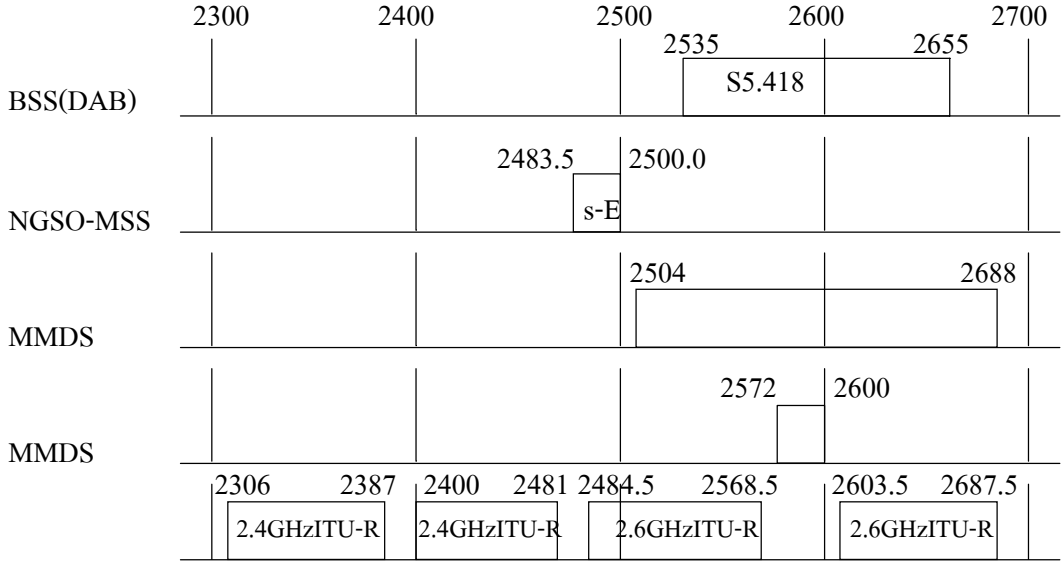
IEEE 802.11 เป็นมาตรฐาน ข้อกำหนดพื้นฐานของการพัฒนาระบบแลนแบบไร้สาย ซึ่งประกอบด้วยอนุกรมของมาตรฐานย่อย 3 มาตรฐานด้วยกัน คือ

IEEE 802.11a เป็นมาตรฐานที่ถูกเสนอขึ้นมาก่อนโดยใช้เทคนิคการรับส่งข้อมูลแบบการเข้าถึงหลายทางแบบแบ่งความถี่ตั้งฉาก ที่แถบความถี่ 5 GHz มีอัตราการส่งข้อมูลสูงสุด 54 Mbps

IEEE 802.11b เป็นมาตรฐานที่ถูกนำเสนอทีหลังแต่กำหนดเสร็จสิ้น และถูกนำมาใช้งานก่อนมาตรฐาน IEEE 802.11a ใช้เทคนิคการรับส่งข้อมูลแบบการเข้าถึงหลายทางแบบแบ่งรหัสลำดับโดยตรง ที่แถบความถี่ 2.4 GHz มีอัตราการส่งข้อมูลสูงสุด 11 Mbps ซึ่งเป็นที่นิยมใช้งานในขณะนี้

IEEE 802.11g เป็นมาตรฐานที่รวมเอาจุดเด่นของมาตรฐานที่กำหนดมาข้างต้นเข้าด้วยกัน โดยใช้เทคนิคการรับส่งข้อมูลแบบการเข้าถึงหลายทางแบบแบ่งความถี่ตั้งฉาก ที่แถบความถี่ 2.4 GHz มีอัตราการส่งข้อมูลสูงสุด 54 Mbps

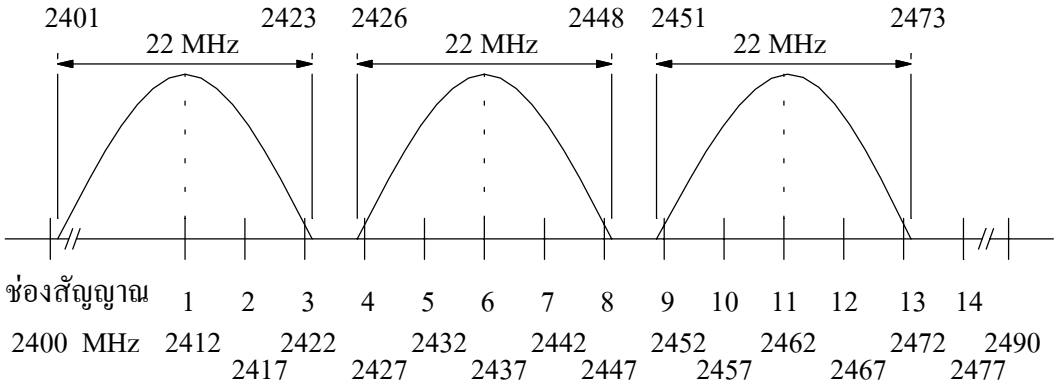
มาตรฐานของระบบแลนแบบไร้สาย ใช้แถบความถี่ของย่านอุตสาหกรรม วิทยาศาสตร์ และทางการแพทย์ มีแถบความถี่ที่ 2.4 GHz แต่ความกว้างแถบที่ใช้งานในแต่ละประเทศไม่เท่ากัน ทำให้แลนแบบไร้สายที่นำมาใช้งานถูกจำกัดจำนวนช่องที่ไม่เท่ากันด้วย ในบางประเทศไม่ต้องขออนุญาตในการนำแถบความถี่ดังกล่าวมาใช้งาน สำหรับประเทศไทยบริษัทผู้นำเข้าอุปกรณ์เป็นผู้ขออนุญาตการใช้ความถี่ และใช้งานอยู่ในย่าน 2.4 GHz ถึง 2.481 GHz (Post and Telegraph Department., online, 2002) ตามรูปที่ 2.16 การนำแถบความถี่นี้มาใช้งานโอกาสที่จะได้รับสัญญาณรบกวนมีสูง นอกจากนี้ยังมีส่วนของเฟดดิงหลายวิถี (multi part fading) และการส่งในระยะไกลจะมีการสูญเสียกำลัง ซึ่งล้วนแต่เป็นปัจจัยที่ทำให้เกิดความผิดพลาดของข้อมูล มาตรฐาน IEEE 802.11 จึงกำหนดให้มีการลดอัตราการส่งข้อมูลลงโดยอัตโนมัติเมื่อพบว่ามีความผิดพลาดของข้อมูลในการรับส่งเกิดขึ้น ต่ำสุดจะอยู่ที่ 1 Mbps



รูปที่ 2.16 การจัดสรรความถี่ย่าน 2.3 – 2.7 GHz สำหรับประเทศไทย

ระบบแลนแบบไร้สาย ได้แบ่งความกว้างแถบออกเป็นช่องความถี่ย่อยๆ 14 ช่อง ห่างกันช่องละ 5 MHz แต่ละช่องได้รับการออกแบบให้ใช้ความกว้างแถบเท่ากับ 22 MHz ซึ่งทำให้เกิดการเหลื่อมล้ำกัน ดังรูปที่ 2.17

การนำไปใช้งานจะต้องเลือกช่องความถี่ที่ไม่ให้เกิดการรบกวนกัน โดยแต่ละช่องจะต้องมีความถี่กลางห่างกันไม่น้อยกว่า 25 MHz (ห่างกัน 5 ช่อง)



รูปที่ 2.17 การจัดสรรและการเลือกใช้ช่องสัญญาณระบบแลนแบบไร้สาย

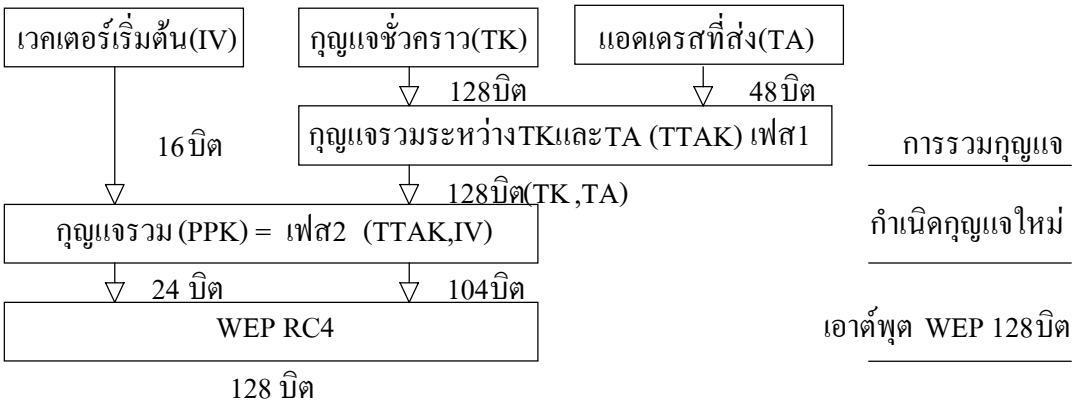
2.6.2 ความปลอดภัยในระบบแลนแบบไร้สาย

โดยทั่วไปในเครือข่ายมาตรฐานของ IEEE 802.11 จะใช้ระบบควบคุมผู้ที่จะเข้ามาในระบบเครือข่าย ซึ่งมีอยู่ 3 วิธีด้วยกันคือ

service set identifier (SSID) เป็นการจัดกลุ่มการใช้เครือข่ายให้กับจุดเข้าถึงแต่ละตัวหรือแต่ละกลุ่มจุดเข้าถึง โดยอาศัย SSID เฉพาะตัวที่แตกต่างกันไป เครื่องลูกข่ายที่จะเข้าใช้งานจะต้องกำหนดว่าจะใช้เครือข่ายของจุดเข้าถึงตัวใดหรือกลุ่มใด โดยทั่วไปจะกำหนดให้เครื่องลูกข่ายสามารถใช้ได้กับหลายๆ จุดเข้าถึง เพื่อให้ใช้งานในตำแหน่งต่างๆ ได้

การกรองตำแหน่งแอดเดรส MAC (MAC address filtering) ที่การ์ดระบบแลนแบบไร้สายของเครื่องลูกข่ายจะมีตำแหน่งแอดเดรส MAC เฉพาะตัวที่แตกต่างกันไป การที่จุดเข้าถึงแต่ละตัวจะยอมให้เครื่องเข้าใช้เครือข่ายได้ก็ต่อเมื่อจุดเข้าถึงนั้นมีตำแหน่งแอดเดรส MAC ของเครื่องลูกข่ายเก็บไว้ที่ตัวของจุดเข้าถึงด้วย

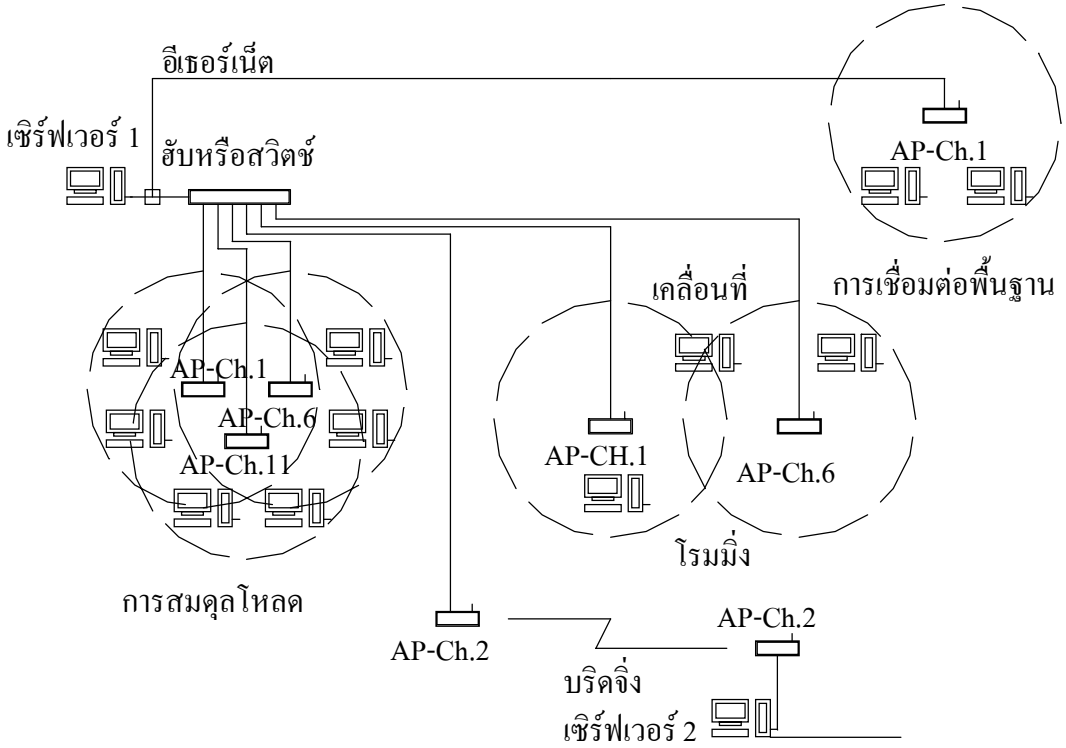
WEP เป็นการเข้ารหัสของการสื่อสารระหว่างเครื่องลูกข่ายกับตัวจุดเข้าถึง โดยอาศัยอัลกอลิทึมการเข้ารหัสแบบ ron's code pseudo random number generator (RC4PRNG) และเข้ารหัสแบบต่อเนื่อง ภายใต้ระบบเครือข่ายเดียวกันที่ใช้ WEP เครื่องลูกข่ายและจุดเข้าถึงทุกตัวจะต้องใช้กุญแจสำหรับเข้ารหัสหรือถอดรหัสตัวเดียวกัน ซึ่งมีกระบวนการในการเข้ารหัส ดังรูปที่ 2.18



รูปที่ 2.18 การเข้ารหัสความปลอดภัยแบบ WEP ที่ใช้ในระบบแลนแบบไร้สาย

2.6.3 สถาปัตยกรรมของระบบเครือข่ายแลนแบบไร้สาย

เป็นสถาปัตยกรรมที่มีความใกล้เคียงกับสถาปัตกรรมของระบบโทรศัพท์เคลื่อนที่ตามแสดงในรูปที่ 2.19 ซึ่งมีลักษณะในการรับส่งข้อมูลดังนี้



รูปที่ 2.19 สถาปัตยกรรมการเชื่อมต่อระบบเครือข่ายแลนแบบไร้สาย

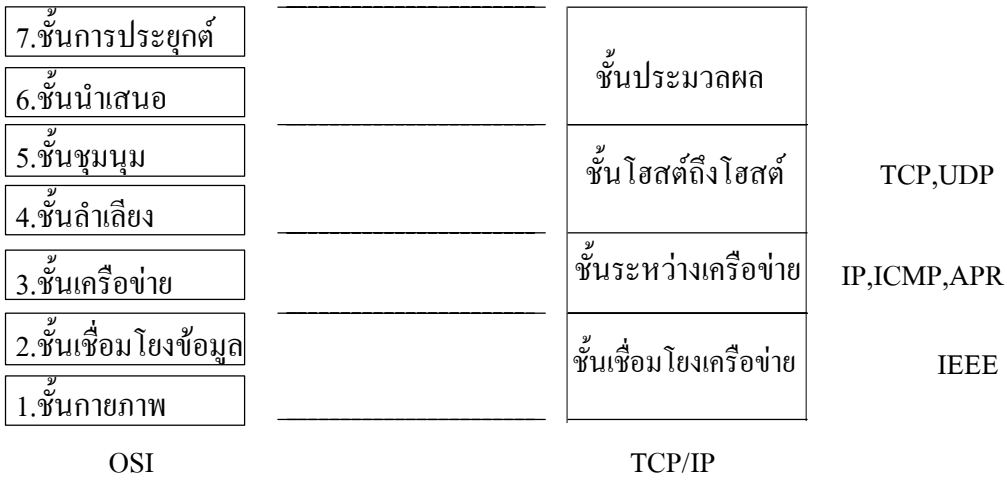
การสมดุลโหลด (load balancing) คือ ลักษณะการจัดวางจุดเข้าถึง ให้มีรัศมีของความถี่ในแต่ละช่องของจุดเข้าถึงแต่ละเครื่องซ้อนทับกัน เพื่อเพิ่มความเร็วในการรับส่งข้อมูลให้กับผู้ใช้งาน หรือเพิ่มจำนวนผู้ใช้งานในพื้นที่เดิมที่จุดเข้าถึงเดิมไม่สามารถรองรับปริมาณข้อมูลที่เพิ่มขึ้นตามจำนวนของผู้ใช้งานได้

โรมมิ่ง (roaming) คือ ลักษณะการจัดวางจุดเข้าถึงให้มีรัศมีของความถี่ในแต่ละช่องที่จุดเข้าถึงแต่ละเครื่องเหลื่อมล้ำกัน ทำให้ผู้ใช้งานในเครือข่ายสามารถเคลื่อนที่ข้ามรัศมีของจุดเข้าถึงได้อย่างอิสระ

บริดจิ่ง (bridging) คือ การติดตั้งจุดเข้าถึงสองเครื่อง เพื่อเชื่อมโยงเครือข่ายที่มีระบบการทำงานเหมือนกันสองเครือข่ายเข้าด้วยกัน

2.7 การทำงานของโพรโตคอล TCP/IP

การรับส่งข้อมูลด้วยระบบแลนแบบไร้สายโดยอาศัยโปรแกรมประยุกต์ (สุวัฒน์ ปุณณชัยยะ, สุพจน์ ปุณณชัยยะ และ ต้น ตัณฑ์สิทธิวงศ์, 2543) ซึ่งเปรียบได้กับการเรียกใช้ชั้นที่ 7, 6 และ 5 ของมาตรฐานการสื่อสารคอมพิวเตอร์ระบบเปิด หลังจากนั้นข้อมูลจะถูกเปลี่ยนให้เป็นเฟรมของ TCP ในชั้นที่ 4 และเฟรมของ IP ในชั้นที่ 3 ก่อนเปลี่ยนรูปแบบของเฟรมให้เป็นไปตามมาตรฐานของ IEEE ซึ่งทำงานในชั้นที่ 2 และ 1 ตามลำดับ ดังแสดงในรูปที่ 2.20 TCP/IP เป็นกลุ่มของโพรโตคอลหลายตัวที่ประกอบกันเป็นชุดในการใช้งาน ซึ่งมีโพรโตคอลหลักเป็น TCP และ IP และมีรายละเอียดการทำงานดังนี้



รูปที่ 2.20 โพรโตคอล TCP/IP เปรียบเทียบกับแบบจำลอง OSI

ชั้นประมวลผล (process layer) เป็นชั้นที่อยู่บนสุดทำงาน 2 หน้าที่เทียบได้กับชั้นการประยุกต์ใช้และชั้นนำเสนอ การทำงานของการประยุกต์ใช้ต่างๆ จะอยู่ที่ชั้นนี้ และมีการติดต่อกันด้วยโพรโตคอลเฉพาะแล้วแต่การประยุกต์ใช้งาน โพรโตคอล TCP/IP สามารถรองรับให้โพรโตคอลอื่นทำงานได้หลายกระบวนการและหลายโพรโตคอลได้พร้อมกัน ทำให้สามารถเปิดโปรแกรมใช้งานได้หลายๆ โปรแกรมพร้อมกัน

ชั้นโฮสต์ถึงโฮสต์ (host-to-host layer) ชั้นนี้จะสร้างการเชื่อมต่อระหว่างการประยุกต์ใช้โดยจุดที่เชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่าพอร์ต ในแต่ละการประยุกต์ใช้จะเชื่อมผ่านพอร์ตที่มีหมายเลขต่างๆ ไม่ซ้ำกัน ทำให้การรับส่งข้อมูลในแต่ละโพรโตคอลทำได้ถูกต้อง ในชั้นนี้ จะมีโพรโตคอลทำงานอยู่สองตัวที่แตกต่างกัน คือ โพรโตคอล TCP และโพรโตคอล user datagram protocol (UDP)

โพรโตคอล TCP (Parker, 1994) การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่ง แต่จะแบ่งข้อมูลออกเป็นส่วนย่อยๆ แล้วจึงส่งไปอย่างต่อเนื่อง ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูลใหม่ ให้ต่อเนื่องกัน ในระหว่างการรับส่งข้อมูล จะมีขบวนการทวนสอบข้อมูลเพื่อให้ข้อมูลมีความถูกต้องโดยการส่งสัญญาณการตอบรับ (acknowledgement signal) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น ซึ่งมีการกำหนดรูปแบบเฟรมของข้อมูลดังแสดงในรูปที่ 2.21 ประกอบด้วยส่วนต่างๆ ดังนี้

รูปที่ 2.21 เฟรมข้อมูลของโพรโตคอล TCP

พอร์ตต้นทาง (source port) แสดงถึงหมายเลขช่องทางที่การประยุกต์ใช้ต้นทางใช้ในการเชื่อมต่อ

พอร์ตปลายทาง (destination port)	แสดงถึงหมายเลขช่องทางปลายทางที่จะนำข้อมูลออก
หมายเลขลำดับ (sequence number)	แสดงตำแหน่งของข้อมูลที่ถูกแบ่งออกเป็นส่วนย่อยๆ ว่าอยู่ตำแหน่งใด เพื่อให้ทางด้านรับสามารถนำข้อมูลนั้นมาประกอบเข้าด้วยกันได้อย่างสมบูรณ์
การตอบรับ	ใช้ส่งสัญญาณในการทวนสอบ
ข้อมูลออฟเซต (data offset)	ใช้แสดงจุดเริ่มต้นของข้อมูล
สำรอง (reserve)	ถูกสำรองไว้เพื่อใช้ในอนาคต ปกติจะมีค่าเป็น ศูนย์
รหัส	เป็นรหัสที่ใช้ควบคุมการติดต่อ เช่น การเข้าจังหวะในการส่ง การเพิ่มเงื่อนไขในการแก้ไขข้อมูล
วินโดว์ (window)	แสดงจำนวนบล็อกข้อมูลย่อยทั้งหมดที่ทางด้านรับข้อมูลจะต้องรับ
ผลรวมการตรวจสอบ (checksum)	ตรวจสอบความถูกต้องของข้อมูลเฉพาะในส่วนหัวของ TCP
ตัวชี้ด่วน (urgent pointer)	เป็นส่วนสนับสนุนการแสดงตำแหน่งของข้อมูลที่ถูกแบ่งเป็นส่วนย่อยๆ ซึ่ง TCP ไม่ได้กำหนดให้ แต่ถูกกำหนดมาจากส่วนของการประยุกต์ใช้
สิ่งเผื่อเลือก (options)	ใช้เก็บข้อมูลปลีกย่อย มีขนาดไม่แน่นอน ซึ่งส่วนใหญ่ไม่มีการนำมาใช้งาน
การเพิ่มให้เต็ม (padding)	เป็นส่วนเติมเต็มให้กับส่วนหัวของ IP เนื่องจากความไม่แน่นอนของข้อมูลในส่วนของสิ่งเผื่อเลือก

โพรโตคอล UDP เป็นการรับส่งข้อมูลแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน ทำให้ด้านส่งข้อมูลไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโพรโตคอล TCP และไม่มีการตรวจสอบความถูกต้องของข้อมูลด้วย เนื่องจากโพรโตคอล UDP ไม่มีสัญญาณการตอบรับ ทำให้การใช้โพรโตคอล UDP ในการส่งผ่านข้อมูลอาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง

ชั้นระหว่างเครือข่าย (internetwork layer) มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโพรโตคอลที่เป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่าย คือ โพรโตคอล IP ที่ร่วมทำงานร่วมกับอีกสองโพรโตคอล คือ โพรโตคอล internet control message protocol (ICMP) และโพรโตคอล address resolution protocol (ARP)

โพรโตคอล IP (Parker, 1994) ทำหน้าที่ให้บริการส่งผ่านข้อมูลข้ามไปยังเครือข่ายใดๆ ได้อย่างถูกต้อง โดยทำงานร่วมกับอุปกรณ์เราเตอร์ ในโพรโตคอล IP จะมีข้อมูลของหมายเลข IP ปลายทางที่จะส่งข้อมูลไปและเมื่อถึงเครือข่ายปลายทางแล้ว จะมีกลไกแปลงหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์ที่ถูกต้อง มีการกำหนดรูปแบบเฟรมของข้อมูลดังแสดงในรูปที่ 2.22 ซึ่งประกอบด้วยส่วนต่างๆ ดังนี้

รูปที่ 2.22 เฟรมข้อมูลของโพรโตคอล IP

เวอร์ชัน (version)	ปัจจุบันถูกกำหนดให้มีค่าเป็น 4 (IPv4) ในอนาคตหากมีการกำหนดมาตรฐานในระดับที่สูงขึ้น ค่าที่กำหนดก็จะเปลี่ยนแปลงตาม
ความยาว	กำหนดความยาวของส่วนหัวที่ใช้ควบคุม IP
ประเภทการบริการ (type of service)	เพื่อบอกให้ทราบว่าจะดำเนินการกับข้อมูลแบบประวิงต่ำ (low delay) หรือแบบอัตราปริมาณงาน (throughput)
ความยาวรวม	แสดงค่าความยาวทั้งหมดของเฟรมเป็นจำนวนไบต์ ซึ่งมีขนาดเล็กที่สุด 576 ไบต์ และขนาดใหญ่ที่สุด 65,535 ไบต์
ตัวบอกลักษณะ (identification)	ใช้บอกให้ทราบว่าข้อมูลที่ถูกแยกย่อยเป็นส่วนๆ นั้นมาจากที่ใด

แฟล็กและแฟล็กออฟเซต (flags and flags offset)	เป็นส่วนที่ใช้ระบุการแยกและการรวมข้อมูล เพื่อให้ข้อมูลที่ถูกแยกเป็นส่วนย่อยสามารถกลับมารวมกันใหม่ได้อย่างถูกต้อง
เวลา	แสดงเวลามากที่สุด ที่ข้อมูลสามารถส่งผ่านเครือข่ายไปถึงปลายทาง ถ้าข้อมูลส่งผ่านเราเตอร์ เวลาจะถูกลดลงครั้งละหนึ่ง ทำให้สามารถนับจำนวนเครือข่ายได้ด้วย ปกติจะกำหนดค่าเป็น 32 วินาที
โพรโตคอล	ใช้บ่งบอกถึงโพรโตคอลที่ทำงานในระดับบนขึ้นไปหนึ่งระดับว่าเป็นโพรโตคอลชนิดใด
ผลรวมการตรวจสอบส่วนหัว	ใช้ตรวจสอบความถูกต้องของข้อมูลเฉพาะในส่วนหัวที่ใช้ควบคุม IP ถ้าพบความผิดพลาดจะไม่นำข้อมูลดังกล่าวมาใช้งาน
แอดเดรส IP ต้นกำเนิด	แสดงแอดเดรส IP ต้นทางที่ส่งข้อมูล
แอดเดรส IP ปลายทาง	แสดงแอดเดรส IP ปลายทางที่รับข้อมูล
สิ่งเผื่อเลือก	ใช้เก็บข้อมูลปลีกย่อยมีขนาดไม่แน่นอน ซึ่งส่วนใหญ่ไม่มีการนำมาใช้งาน
การเพิ่มให้เต็ม	เป็นส่วนเติมเต็ม เนื่องจากความไม่แน่นอนของข้อมูลในส่วนของสิ่งเผื่อเลือก

โพรโตคอล ICMP ทำหน้าที่แจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูล ซึ่งข้อความต่างๆ ที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของเฟรม IP ดังนั้นโพรโตคอล ICMP จึงเป็นเครื่องมืออย่างหนึ่งในการช่วยทดสอบเครือข่าย เช่น คำสั่ง ping ที่มีการเรียกใช้งานโพรโตคอล ICMP และแจ้งเป็นข้อความให้ทราบ

โพรโตคอล ARP เป็นตารางความจำที่มีรายการความสัมพันธ์ระหว่างหมายเลข IP และหมายเลขของฮาร์ดแวร์ ปกติจะถูกเรียกใช้งานโดยโพรโตคอล IP เพื่อช่วยแปลงหมายเลข IP ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง เช่น แผงวงจรแลนซึ่งจะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใคร

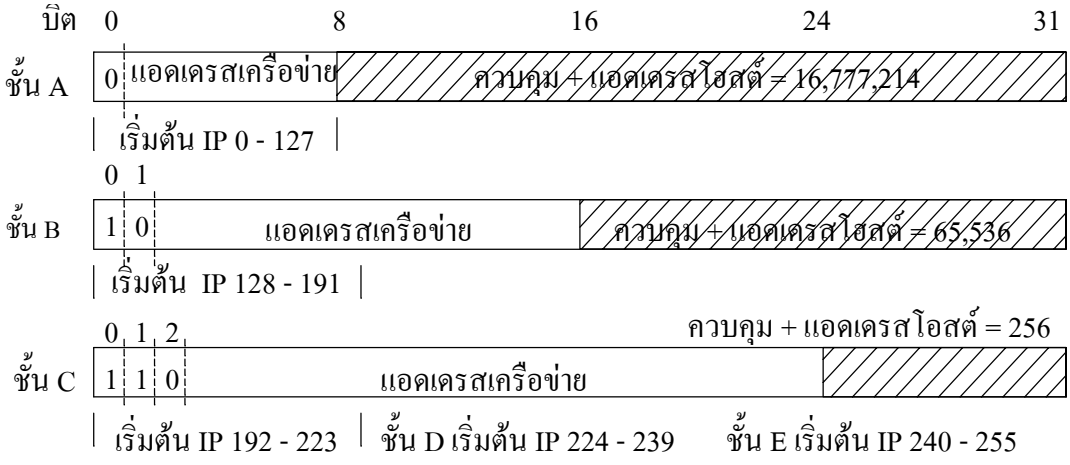
ชั้นเชื่อมโยงเครือข่าย (network interface layer) เป็นการทำงานที่ชั้นล่างสุด ซึ่งเป็นการแปลงเฟรมข้อมูลให้อยู่ในรูปที่เหมาะสม และเปลี่ยนเป็นสัญญาณไฟฟ้าก่อนส่งต่อไปยังเครือข่าย

หมายเลขแอดเดรส IP ถูกกำหนดขึ้นมาให้เป็นหมายเลขอ้างอิงประจำตัวของอุปกรณ์ต่างๆ ที่เชื่อมต่ออยู่ในเครือข่ายซึ่งจะต้องไม่ซ้ำกัน หมายเลขแอดเดรส IP นี้จะไม่ผูกติดกับฮาร์ดแวร์แต่อย่างใด จึงสามารถกำหนดใหม่ หรือแก้ไขเปลี่ยนแปลงได้ เมื่อมีการเปลี่ยนแปลงฮาร์ดแวร์ ต่างจากหมายเลขแอดเดรส MAC ซึ่งเป็นหมายเลขประจำตัวของอุปกรณ์ที่ต่ออยู่ในเครือข่าย ถูกกำหนดจากบริษัทผู้ผลิตอุปกรณ์ตั้งแต่เริ่มผลิต

เพื่อให้การกำหนดแจกจ่ายหมายเลขแอดเดรส IP ให้กับเครือข่ายต่างๆ ได้อย่างเหมาะสม และใช้งานให้เกิดประโยชน์สูงสุด จึงมีหน่วยงานกลาง internet network information center (InterNIC) ทำหน้าที่กำหนด และแจกจ่ายให้แต่ละองค์กรได้ใช้งาน มีมาตรฐานการแบ่งระดับชั้นของเครือข่ายที่ชัดเจนดังรูปที่ 2.23 ซึ่งแบ่งได้เป็น 5 ระดับ คือ

ชั้นเอ บิตแรกของไบต์แรกสุดจะเป็น 0 เสมอ หมายเลขแอดเดรส IP เริ่มตั้งแต่ 0 ถึง 127 จะกำหนดให้กับเครือข่ายที่มีขนาดใหญ่ หนึ่งเครือข่ายสามารถมีเครื่องลูกข่ายได้มากกว่า 16 ล้านเครื่อง หมายเลขแอดเดรส IP จะเป็นลักษณะ net.host.host.host

ชั้นบี บิตแรกของไบต์แรกสุดจะเป็น 1 และ 0 เสมอ หมายเลขแอดเดรส IP เริ่มตั้งแต่ 128 ถึง 191 จะกำหนดให้กับเครือข่ายที่มีขนาดใหญ่เช่นกัน แต่เล็กกว่าชั้นเอ ในแต่ละเครือข่ายสามารถมีเครื่องลูกข่ายได้ 2^{16} ลบด้วยจำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่าย ได้ 64,516 เครื่อง หมายเลขแอดเดรส IP จะเป็นลักษณะ net.net.host.host



รูปที่ 2.23 การแบ่งชั้นของเครือข่ายหมายเลขแอดเดรส IP

ชั้นซี บิตแรกของไบต์แรกสุดจะเป็น 1, 1 และ 0 เสมอ หมายเลขแอดเดรส IP เริ่มตั้งแต่ 192 ถึง 223 จะกำหนดให้กับเครือข่ายที่เป็นองค์กรทั่วไป ซึ่งส่วนใหญ่จะเป็นองค์กรขนาดกลางถึงเล็ก

ในแต่ละเครือข่ายมีเครื่องลูกข่ายไม่เกิน 2^8 ลบด้วยจำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่ายได้ 254 เครื่อง หมายเลขแอดเดรส IP จะเป็นลักษณะ net.net.net.host

จำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่ายมีสองแอดเดรส คือ 0 กำหนดให้เป็นหมายเลขอ้างอิงของเครือข่าย และ 255 กำหนดให้เป็นหมายเลขแพร่สัญญาณเครือข่าย (network broadcast)

ชั้นดี เป็นหมายเลขแอดเดรส IP สำรองไว้สำหรับส่งข้อมูล ไม่แจกจ่ายให้ใช้งานทั่วไป

ชั้นอี เป็นหมายเลขแอดเดรส IP พิเศษที่ใช้สำหรับงานทดสอบและพัฒนา ไม่มีการกำหนดให้ใช้งานทั่วไป

2.7.1 เครือข่ายย่อย (Subnet)

ปัจจุบันการกำหนดหมายเลขแอดเดรส IP ของเครือข่ายประเภท ชั้นเอ และชั้นบี นั้นไม่มีการกำหนดให้แล้ว เนื่องจากแทบไม่มีเครือข่ายใดที่มีความจำเป็นต้องใช้หมายเลขแอดเดรส IP มากขนาดนั้น คงเหลือแต่ชั้นซีเท่านั้น หากว่าแต่ละเครือข่ายของชั้นซีเชื่อมต่อเครื่องลูกข่ายไม่กี่เครื่อง หมายเลขแอดเดรส IP ที่เหลือก็จะเสียเปล่า และจะแบ่งให้เครือข่ายอื่นนำไปใช้งานไม่ได้ เนื่องจากอุปกรณ์บางประเภทจะพิจารณาที่หมายเลขแอดเดรส IP 24 บิตซ้ายมือคือดูที่หมายเลขแอดเดรสประจำเครือข่าย การทำเครือข่ายย่อยจะช่วยให้การกำหนดหมายเลขแอดเดรส IP ถูกแบ่งออกเป็นเครือข่ายย่อยๆ เพื่อให้ใช้หมายเลขแอดเดรส IP ได้อย่างมีประสิทธิภาพ ตามตารางที่ 2.1

การแบ่งเครือข่ายย่อยนี้เราเตอร์จะต้องทราบข้อมูลว่าการแบ่งนั้นใช้ข้อมูลกี่บิตเป็นเครือข่ายย่อยและใช้ข้อมูลกี่บิตเป็นแอดเดรสคอมพิวเตอร์แม่ จึงจะส่งข้อมูลให้เครือข่ายได้อย่างถูกต้อง ข้อมูลการแบ่งเครือข่ายย่อยที่แจ้งให้เราเตอร์ทราบนี้ เรียกว่าซับเน็ตมาซค (subnet mask)

ตารางที่ 2.1 การแบ่งเครือข่ายย่อยทั้งหมดของชั้นซี

จำนวนเครือข่ายย่อย	จำนวนเครื่องลูกข่าย	เลขฐานสิบของซับเน็ตมาซค	จำนวนบิตที่ใช้กำหนดเครือข่ายย่อย
2	62	255.255.255.192	2
6	30	255.255.255.224	3
14	14	255.255.255.240	4
30	6	255.255.255.248	5
62	2	255.255.255.252	6

จากความนิยมของการใช้งานโพรโตคอล TCP/IP มีเป็นจำนวนมาก หน่วยงาน InterNIC จึงมีการกำหนดให้หมายเลขประจำเครือข่ายบางช่วงเป็นหมายเลขพิเศษสำรองเอาไว้

ใช้งานในวัตถุประสงค์แบบเครือข่ายส่วนบุคคล (private network) ที่ไม่สามารถติดต่อกับเครือข่ายภายนอกได้ (non-routable network) โดยมีหมายเลขเครือข่ายดังกล่าวแสดงในตารางที่ 2.2

ตารางที่ 2.2 การแบ่งเครือข่ายแบบเครือข่ายส่วนบุคคล

ประเภทชั้นของเครือข่าย	จำนวนเครือข่ายที่กำหนดได้	หมายเลขเครือข่าย
A	1	10.0.0.0
B	16	172.16.0.0 – 173.31.0.0
C	256	192.168.0.0 – 192.168.255.0

2.7.2 แอดเดรสวนกลับ (loop back address)

เป็นแอดเดรสที่กำหนดขึ้นเพื่อใช้ในงานที่ต้องการให้ขบวนการทำงานหนึ่งติดต่อกับอีกขบวนการหนึ่งภายในเครื่องเดียวกัน หากมีการส่งข้อมูลผ่านไปที่หมายเลขแอดเดรส IP เหล่านี้จะไม่มีการส่งข้อมูลออกนอกเครือข่าย แต่จะย้อนกลับมาที่หมายเลขแอดเดรส IP ต้นทางเช่นเดิม เช่น หมายเลขแอดเดรส IP 127.0.0.0

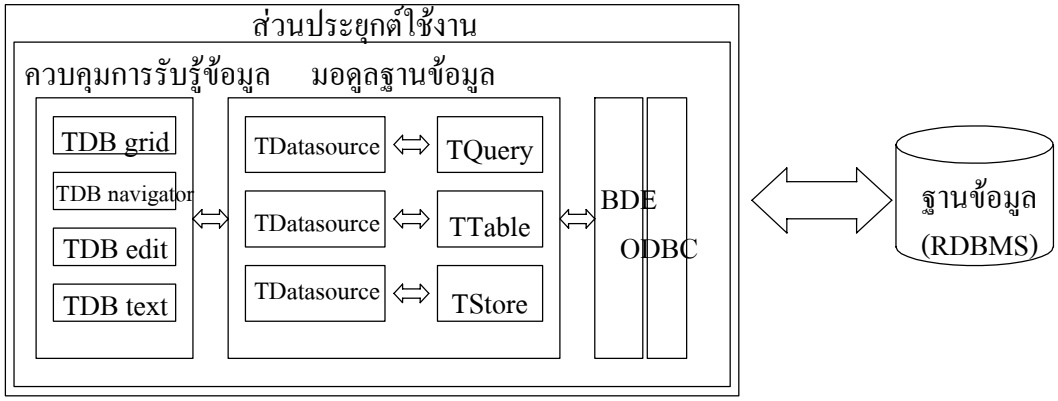
2.8 ชนิดของฐานข้อมูลและการเข้าถึงข้อมูล

ฐานข้อมูลและการเข้าถึงข้อมูล (ประพันธ์ อัศวภาณุวัฒน์, 2543) แบ่งเป็นหลายมอดูลด้วยกัน ดังนี้

มอดูลฐานข้อมูลแบบชั้นเดียว (single-tiered database module) เป็นการเข้าถึงแฟ้มข้อมูลแบบเก่า เหมาะสำหรับการใช้งานประเภทผู้ใช้เดี่ยว การประยุกต์ใช้งานจะประกอบไปด้วย ส่วนควบคุมการรับรู้ข้อมูล (data aware controls) เป็นส่วนของการนำเสนอข้อมูลให้กับผู้ใช้ โดยติดต่อกับมอดูลฐานข้อมูล เพื่อเชื่อมโยง borland database engine (BDE) ของโปรแกรมเดลไฟ ซึ่งทำหน้าที่เป็นล่าม ระหว่างระบบกับแฟ้มข้อมูลจริง ทำให้การใช้งานสามารถเข้าถึงข้อมูลได้ โดยผู้พัฒนาระบบไม่จำเป็นต้องเขียนโปรแกรมติดต่อกับฐานข้อมูลชนิดต่างๆ เอง

มอดูลฐานข้อมูลแบบไคลเอ็นต์/เซิร์ฟเวอร์ (client/server database module) ตามรูปที่ 2.24 เป็นระบบที่เน้นการเข้าถึงข้อมูลแบบผู้ใช้งานหลายคน บนระบบข้อมูลแบบสัมพันธ์ relation database management system (RDBMS) ผู้ใช้งานสามารถใช้งานพร้อมๆกันได้ ผ่านทางการเชื่อมโยง BDE ของโปรแกรมเดลไฟ และการเชื่อมโยง open data base connectivity (ODBC) ของวินโดวส์ เมื่อมีการอ่านหรือบันทึกข้อมูล คำสั่งจะถูกนำไปประมวลผลที่เซิร์ฟเวอร์ ด้วยคำสั่ง

structured query language (SQL) ทำให้เซิร์ฟเวอร์สามารถรองรับการทำงานแบบผู้ใช้งานหลายคน และจัดเก็บข้อมูลที่ซ้ำซ้อนได้ ซึ่งนำมาใช้พัฒนาในงานวิทยานิพนธ์นี้



รูปที่ 2.24 มอดูลฐานข้อมูลแบบไคลเอ็นต์/เซิร์ฟเวอร์

มอดูลฐานข้อมูลแบบหลายชั้น (multi-tiered database module) เป็นแนวความคิดในการแบ่งการทำงานของระบบฐานข้อมูลออกเป็นหน่วยย่อย ด้วยการแยกส่วนการติดต่อข้อมูลออกจากส่วนการใช้งานของไคลเอ็นต์ ไปรวมไว้กับส่วนการใช้งานของเซิร์ฟเวอร์ ให้ทำหน้าที่เป็นศูนย์การรวบรวมกิจกรรมการติดต่อของข้อมูล เพื่อลดบทบาทการทำงานของส่วนการใช้งานของไคลเอ็นต์ลง โปรแกรมที่ใช้งานโดยรวมมีขนาดเล็กลงทำให้ง่ายต่อการแจกจ่ายและลดความซ้ำซ้อนในการติดต่อข้อมูลเพราะรวบรวมไว้ในจุดเดียวกัน เป็นการเพิ่มประสิทธิภาพให้กับระบบ

2.9 โพรโตคอล I^2C (I^2C bus protocal)

เป็นโพรโตคอลที่ใช้ในการขนถ่ายข้อมูลแบบสองทิศทาง ซึ่งใช้งานกับไอซีติดตามการทำงานของไมโครคอนโทรลเลอร์ โดยอาศัยสายสัญญาณเพียงสองเส้น คือ สายสัญญาณนาฬิกา (serial clock) และสายสัญญาณข้อมูล (serial data) ในหนึ่งคาบของสัญญาณนาฬิกาจะส่งข้อมูลได้เพียงหนึ่งบิต อุปกรณ์ที่จะเข้าใช้สายสัญญาณเพื่อติดต่อกับอุปกรณ์อื่นจะทำหน้าที่ผลิตสัญญาณนาฬิกา ควบคุมการใช้สายสัญญาณและสร้างสภาวะต่างๆ ให้เหมาะสมกับการใช้งานของโพรโตคอล ดังแสดงในรูปที่ 2.25

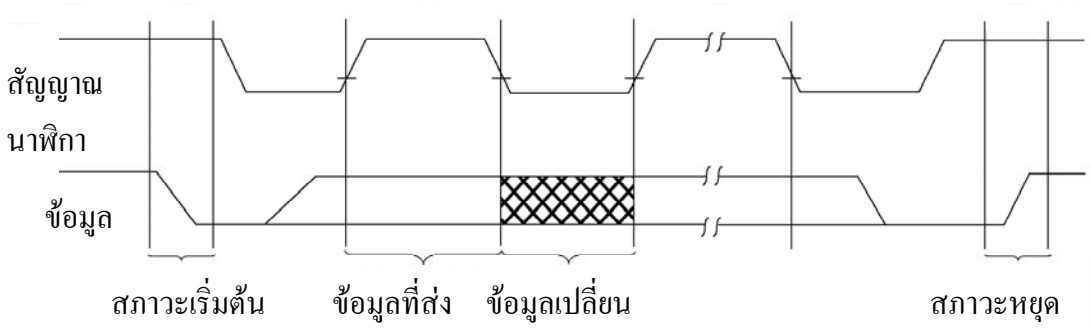
ข้อกำหนดของโพรโตคอล

สภาวะที่ไม่มีการส่งข้อมูล สายสัญญาณทั้งสองเส้นจะมีระดับสัญญาณเป็นหนึ่ง

สภาวะเริ่มต้นของข้อมูล สายสัญญาณนาฬิกามีระดับสัญญาณเป็นหนึ่ง สายสัญญาณข้อมูลเปลี่ยนแปลงสัญญาณจากระดับหนึ่งเป็นระดับศูนย์

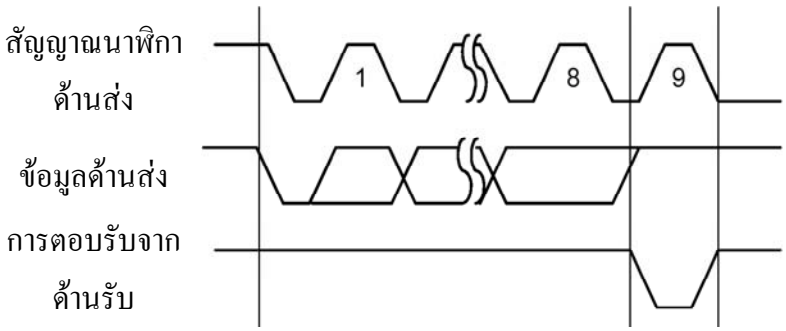
สภาวะหยุดของข้อมูล สายสัญญาณนาฬิกามีระดับสัญญาณเป็นหนึ่ง สายสัญญาณข้อมูลเปลี่ยนแปลงสัญญาณจากระดับศูนย์เป็นระดับหนึ่ง

สภาวะที่มีการส่งข้อมูล แบ่งเป็นสองส่วนคือส่วนที่หนึ่งสายสัญญาณนาฬิกามีระดับสัญญาณเป็นหนึ่งสายสัญญาณข้อมูลต้องไม่มีการเปลี่ยนแปลงระดับสัญญาณของข้อมูล ส่วนที่สองสายสัญญาณนาฬิกามีระดับสัญญาณเป็นศูนย์สายสัญญาณข้อมูลมีการเปลี่ยนแปลงระดับสัญญาณของข้อมูลเพื่อเตรียมส่งในรอบต่อไปที่สายสัญญาณนาฬิกามีระดับสัญญาณเป็นหนึ่ง



รูปที่ 2.25 สภาวะต่างๆ ตามข้อกำหนดของโพรโตคอล

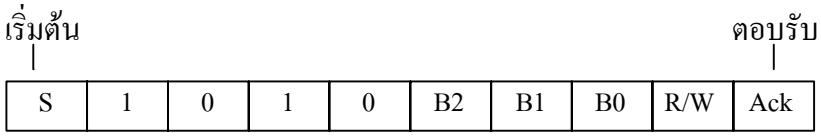
สภาวะยอมรับข้อมูล (acknowledge) ขณะที่สายสัญญาณนาฬิกามีระดับสัญญาณเป็นหนึ่งข้อมูลที่มีระดับเป็นศูนย์จะถูกส่งมาจากอุปกรณ์ที่รับข้อมูล เมื่อมีการรับข้อมูลครบหนึ่งไบต์ ดังแสดงในรูปที่ 2.26



รูปที่ 2.26 การส่งสภาวะยอมรับข้อมูล

เมื่อมีสภาวะเริ่มต้นเกิดขึ้นจะต้องส่งข้อมูลจำนวนหนึ่งไบต์เพื่อทำการควบคุมการทำงานของอุปกรณ์ให้พร้อมทำการส่งหรือรับข้อมูล ซึ่งประกอบด้วย รหัสควบคุม หมายเลขแอดเดรสของข้อมูล และคำสั่งสำหรับรับหรือส่งข้อมูล แล้วจึงตามหลังด้วยไบต์ของข้อมูล ดังแสดงในรูปที่ 2.27

การใช้งาน	รหัสควบคุม	หมายเลขแอดเดรส	อ่าน/เขียน
อ่าน	1010	แอดเดรส	1
เขียน	1010	แอดเดรส	0



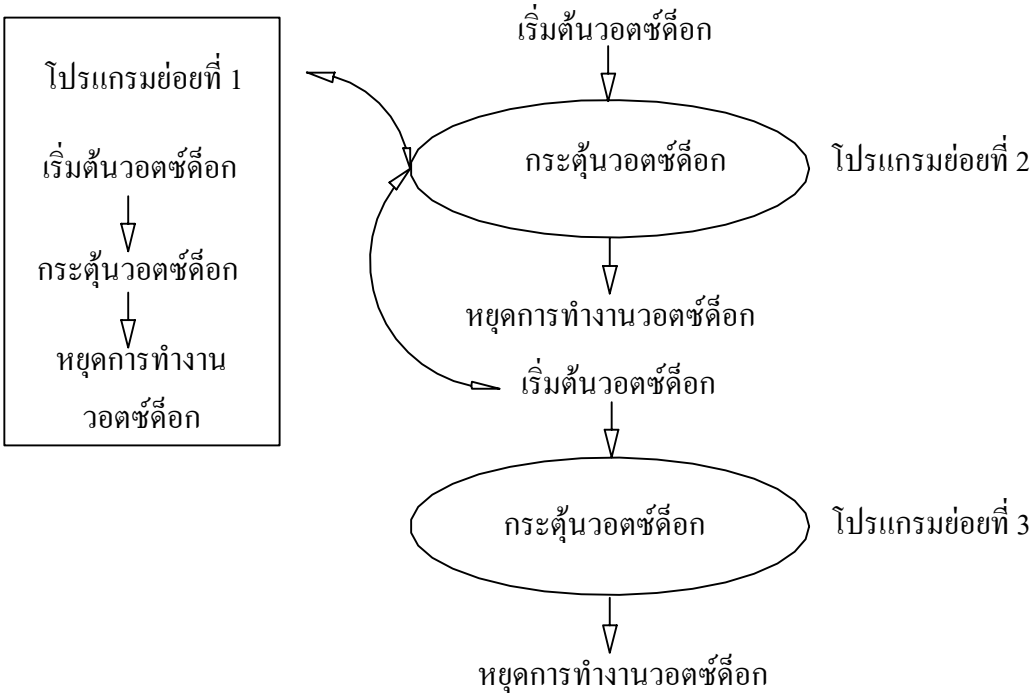
รูปที่ 2.27 รายละเอียดของไบต์ที่ใช้ควบคุม

2.10 ระบบวอตซ์ด็อก (watchdog)

เป็นการติดตามการทำงานของไมโครคอนโทรลเลอร์ โดยการอินเตอร์รัฟให้ไมโครคอนโทรลเลอร์เก็บค่าสำหรับการนับไว้หนึ่งค่า หลังจากนั้นระบบจะทำลดค่าดังกล่าวไปเรื่อยๆ จนกระทั่งเป็นศูนย์ ไมโครคอนโทรลเลอร์จึงจะรีเซตระบบทั้งหมดแล้วเริ่มต้นการทำงานของโปรแกรมใหม่ การใช้งานระบบวอตซ์ด็อกนั้นเมื่อทำการอินเตอร์รัฟให้เก็บค่าสำหรับการนับแล้ว เมื่อโปรแกรมทำงานไประยะหนึ่งจะต้องทำการกระตุ้นระบบวอทซ์ด็อกโดยการอินเตอร์ให้ค่าการนับกลับไปเป็นค่าเดิมที่เคยเก็บไว้ เพื่อป้องกันไม่ให้ระบบทำการลดค่าดังกล่าวจนกระทั่งเป็นศูนย์

ประโยชน์ของการนำระบบวอทซ์ด็อกมาใช้งานเพื่อให้ไมโครคอนโทรลเลอร์สามารถรีเซตระบบการทำงานได้เองหากไมโครคอนโทรลเลอร์ทำงานค้างอยู่ที่เดิม

ไมโครคอนโทรลเลอร์แรบบิท 2000 และโปรแกรมไดนามิคซี สามารถเก็บค่าสำหรับการนับได้ตั้งแต่ 1 ถึง 255 ค่า ซึ่งในการลดค่าแต่ละครั้งนั้นใช้เวลา 62.5 มิลลิวินาที ดังนั้นการตั้งค่าในการนับแต่ละครั้งจะต้องให้มีความเหมาะสมกับค่าเวลาที่ใช้ในการทำงานของโปรแกรม คือจะต้องตั้งค่าการนับสำหรับระบบวอทซ์ด็อกเมื่อคำนวณเป็นเวลาแล้วอย่างน้อยที่สุดจะต้องมากกว่าเวลาที่โปรแกรมนั้นทำงาน เพื่อให้โปรแกรมสามารถกระตุ้นระบบวอทซ์ด็อกได้ทัน ก่อนที่ระบบจะรีเซตตัวเอง ดังรูปที่ 2.28 ซึ่งเป็นขั้นตอนในการทำงานของระบบนี้



รูปที่ 2.28 ตัวอย่างขั้นตอนในการทำงานของระบบวอตซ์ด็อก

2.11 บทสรุป

การส่งข้อมูลด้วยระบบแลนแบบไร้สายตามมาตรฐาน IEEE 802.11b ใช้เทคนิคในการรับส่งข้อมูลแบบการเข้าถึงหลายทางแบบแบ่งรหัส มีแถบความถี่ที่ใช้งานระหว่าง 2.4 GHz ถึง 2.481 GHz ซึ่งมีสถาปัตยกรรมของระบบเครือข่ายคล้ายระบบโทรศัพท์เคลื่อนที่ มีระบบรักษาความปลอดภัยของข้อมูลและอาศัยโพรโตคอลทำงานร่วมกันในการขนถ่ายข้อมูลและตรวจสอบข้อมูลหลายโพรโตคอลด้วยกัน

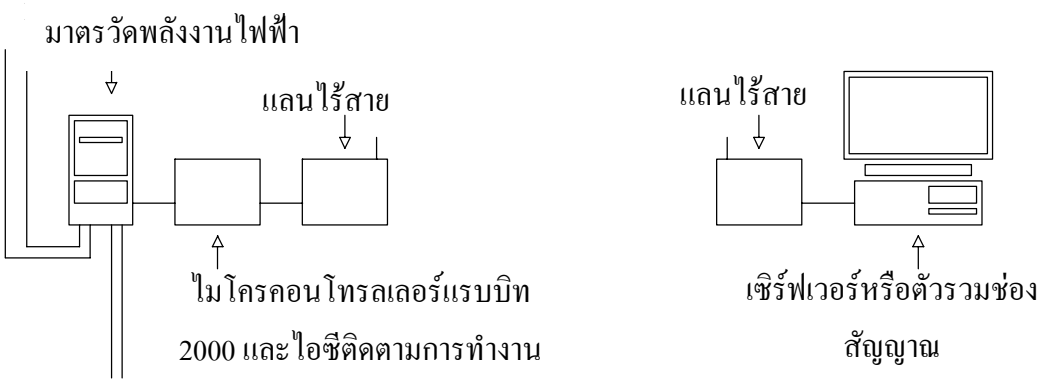
TCP/IP เป็นกลุ่มของโพรโตคอลที่ประกอบกันเป็นชุดในการใช้งาน การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่ง แต่จะแบ่งข้อมูลออกเป็นส่วนย่อยๆ แล้วส่งไปอย่างต่อเนื่อง มีขบวนการทวนสอบข้อมูลเพื่อให้ข้อมูลมีความถูกต้องและส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางเกิดความผิดพลาดขึ้น

หน่วยงาน InterNIC ทำหน้าที่กำหนดและแจกจ่ายหมายเลขแอดเดรส IP ให้หน่วยงานต่างๆ ใช้งานในเครือข่าย แต่เนื่องจากหมายเลขแอดเดรส IP มีอยู่อย่างจำกัด และมีบางหมายเลขไม่ได้ใช้งาน แต่ไม่สามารถแจกจ่ายให้กับหน่วยงานอื่นได้ จึงมีการกำหนดซับเน็ตมาซคขึ้น เพื่อแบ่งเครือข่ายให้เป็นเครือข่ายย่อยๆ ทำให้การใช้หมายเลขแอดเดรส IP มีประสิทธิภาพมากขึ้น

บทที่ 3
วิธีดำเนินการพัฒนาด้านวิศวกรรม

3.1 บทนำ

การพัฒนาระบบการอ่านหน่วยมาตรวัดพลังงานไฟฟ้าระบบหนึ่งเฟสด้วยแลนแบบไร้สาย ได้ดำเนินการติดตั้งไมโครคอนโทรลเลอร์เพิ่มเติมที่มาตรวัดพลังงานไฟฟ้าเดิม และพัฒนาโปรแกรม เพื่อนับจำนวนรอบการหมุนของมาตรวัด จัดรูปแบบของข้อมูล และส่งข้อมูลหน่วยการใช้พลังงานไฟฟ้า หรือรอรับคำสั่งต่างๆจากอุปกรณ์รวมช่องสัญญาณ ซึ่งในงานพัฒนานี้อุปกรณ์รวมช่องสัญญาณเป็นเครื่องคอมพิวเตอร์ที่มีการพัฒนาโปรแกรมให้ทำการแยกข้อมูล เพื่อนำไปเก็บเป็นฐานข้อมูลและแสดงผลให้ทราบถึงสถานะการทำงานของระบบในภาพรวม ระบบการอ่านหน่วยมาตรวัดพลังงานนี้ได้อาศัยระบบแลนแบบไร้สายเป็นอุปกรณ์หลักในการทำหน้าที่รับส่งข้อมูล โดยการกำหนดหมายเลขแอดเดรส IP ของเครือข่ายให้เป็นแบบเครือข่ายส่วนบุคคล ดังแสดงระบบในภาพรวมในรูปที่ 3.1



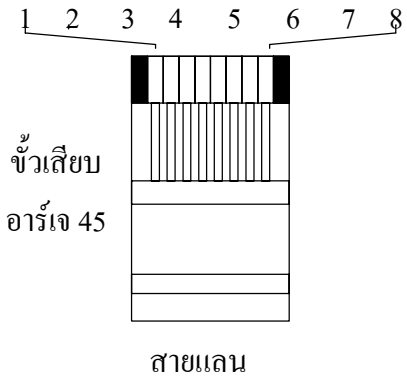
รูปที่ 3.1 ระบบการอ่านหน่วยมาตรวัดพลังงานไฟฟ้าระบบหนึ่งเฟสด้วยแลนแบบไร้สาย

3.2 วิธีการพัฒนา

กล่าวถึงการเชื่อมต่ออุปกรณ์ของระบบ การปรับตั้งค่าต่างๆของอุปกรณ์ การจัดรูปแบบของข้อมูล รวมถึงการพัฒนาโปรแกรมเพื่อควบคุมและการเชื่อมต่อฐานข้อมูล เพื่อให้ระบบทำงานตามวัตถุประสงค์ของงานพัฒนา

3.2.1 เชื่อมต่อระบบแลนแบบไร้สายเข้ากับคอมพิวเตอร์และ ไมโครคอนโทรลเลอร์

มาตรวัดพลังงานไฟฟ้าได้อาศัยไมโครคอนโทรลเลอร์แรบบิท 2000 เป็นศูนย์กลางการประมวลผล ซึ่งมีหน้าที่ในการรับหน่วยพลังงานจากมาตรวัดเข้ามาจัดรูปแบบของข้อมูลดังแสดงรายละเอียดในหัวข้อต่อไป ก่อนทำการส่งข้อมูลให้ระบบแลนแบบไร้สาย ทำการส่งข้อมูลให้กับคอมพิวเตอร์ที่มีหน้าที่เป็นตัวรวมช่องสัญญาณ โดยการเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับแลนแบบไร้สาย และคอมพิวเตอร์กับแลนแบบไร้สายจะอาศัยสายแลนที่เข้าขั้วเสียบแลนแบบอาร์เจ 45 ดังแสดงตำแหน่งขาของขั้วเสียบตามรูปที่ 3.2 และต่อสายแลนแบบสลับสายข้าม (cross over) หรือแบบจุดต่อจุด ซึ่งมีรายละเอียดรหัสสีของสายแลนสำหรับเข้าขั้วเสียบตามตารางที่ 3.1



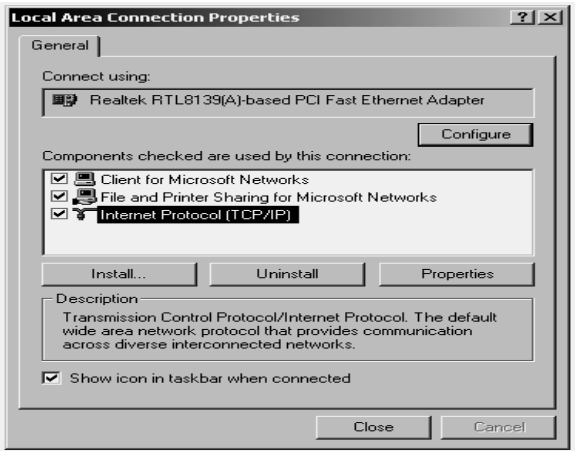
รูปที่ 3.2 ตำแหน่งขาขั้วเสียบแลนแบบอาร์เจ 45 ที่เขี้ยวล็อกอยู่ด้านล่าง

ตารางที่ 3.1 รหัสสีของสายแลนและตำแหน่งหน้าที่ของขั้วเสียบแลนแบบอาร์เจ 45 สำหรับแลนระบบ 10/100 ฐาน T

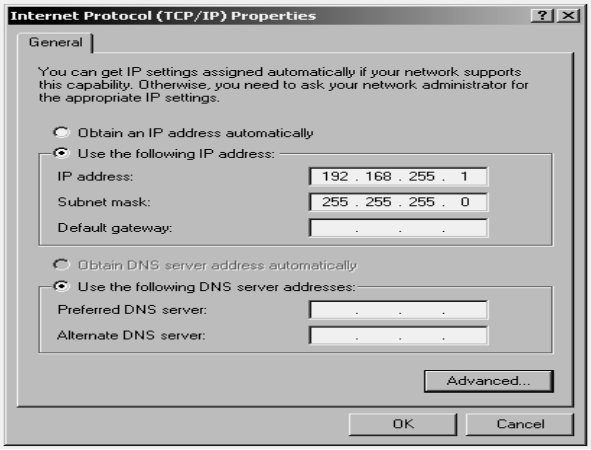
ตำแหน่ง	หน้าที่	รหัสสีสายด้านส่ง	รหัสสีสายด้านรับ (จุดต่อจุด)	รหัสสีสายด้านรับ (โทโพโลยีรูปดาว)
1	ส่งสัญญาณ +Tx	ขาวส้ม	ขาวเขียว	ขาวส้ม
2	ส่งสัญญาณ –Tx	ส้ม	เขียว	ส้ม
3	รับสัญญาณ +Rx	ขาวเขียว	ขาวส้ม	ขาวเขียว
4	ยังไม่ใช้งาน	น้ำเงิน	น้ำเงิน	น้ำเงิน
5	ยังไม่ใช้งาน	ขาวน้ำเงิน	ขาวน้ำเงิน	ขาวน้ำเงิน
6	รับสัญญาณ -Rx	เขียว	ส้ม	เขียว
7	ยังไม่ใช้งาน	ขาวน้ำตาล	ขาวน้ำตาล	ขาวน้ำตาล
8	ยังไม่ใช้งาน	น้ำตาล	น้ำตาล	น้ำตาล

โพรโตคอล TCP/IP ในงานพัฒนานี้มีขั้นตอนในการติดตั้งดังแสดงในรูปที่ 3.3 ซึ่งได้มีการตั้งค่าหมายเลขแอดเดรส IP ของระบบทั้งด้านมาตรวัดพลังงานและด้านอุปกรณ์รวมช่องสัญญาณให้เป็นชั้นซีและเป็นแบบเครือข่ายส่วนบุคคลที่ไม่สามารถติดต่อจากระบบเครือข่ายภายนอกได้ มีหมายเลขเครือข่ายอยู่ระหว่าง 192.168.0.0 ถึง 192.168.0.255 การตั้งหมายเลขแอดเดรส IP ในเครือข่ายเดียวกันจะต้องมีค่าไม่ซ้ำกัน นอกจากนี้ยังกำหนดหมายเลขเครือข่ายย่อยให้ซับเน็ตมาซคมีค่าเป็น 255.255.255.0 เพื่อรองรับจำนวนมาตรวัดให้ได้มากที่สุด 254 เครื่อง

รูปที่ 3.3 ขั้นตอนการติดตั้งโพรโตคอล TCP/IP ของเครือข่าย



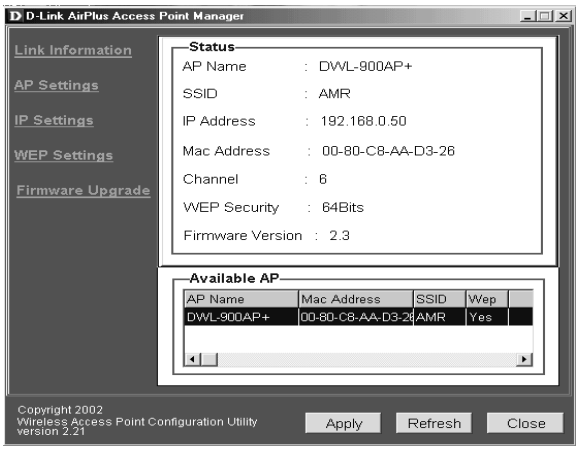
เลือกคลิกซ้ายที่โพรโตคอล TCP/IP แล้วเลือกกดที่ปุ่ม properties



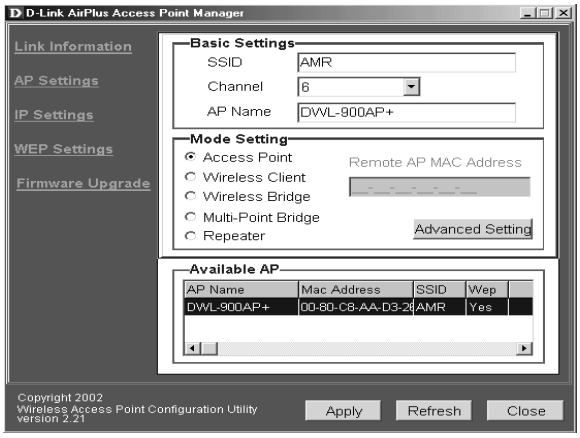
ป้อน IP address ของเครื่องคอมพิวเตอร์ที่เป็นตัวรวมช่องสัญญาณ ซึ่งกำหนดเป็น 192.168.0.1 และ subnet mask เป็น 255.255.255.0 แล้วคลิกซ้ายที่ OK

รูปที่ 3.3 (ต่อ) ขั้นตอนการติดตั้งโพรโตคอล TCP/IP ของเครือข่าย

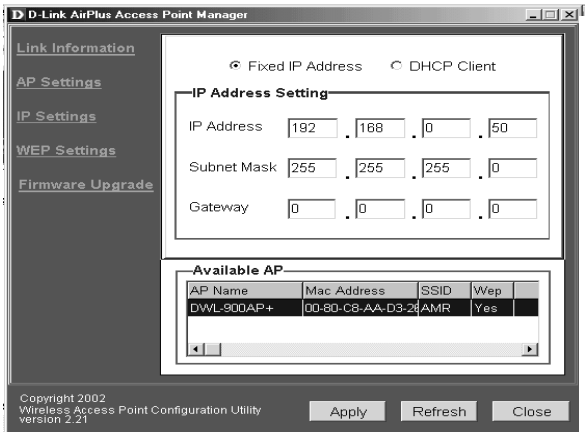
การปรับตั้งค่าของอุปกรณ์รับส่งข้อมูลบนระบบแลนแบบไร้สายทั้งด้านมาตรวัดพลังงานและด้านตัวรวมช่องสัญญาณซึ่งประกอบด้วย ชื่อของ SSID ช่องสัญญาณที่ใช้งาน ชื่อของจุดเข้าถึง หมายเลขแอดเดรส IP หมายเลขซับเน็ตมาซค และรหัสของกุญแจความปลอดภัย WEP โดยได้อาศัยโปรแกรมการจัดการแอร์พลัส เอพี (airplus ap) ของบริษัทดีลิงค์ (d-link) ซึ่งมีรายละเอียดในการปรับตั้งค่าการทำงานแสดงในรูปที่ 3.4 เมื่อดำเนินการปรับตั้งค่าต่างๆแล้วเสร็จ อุปกรณ์จะทำการเก็บค่าเหล่านั้นไว้



หน้าจอแสดงข้อมูลต่างๆ ที่ได้รับการปรับตั้งค่าแล้ว และจำนวนจุดเข้าถึงที่สามารถเข้าใช้งานได้

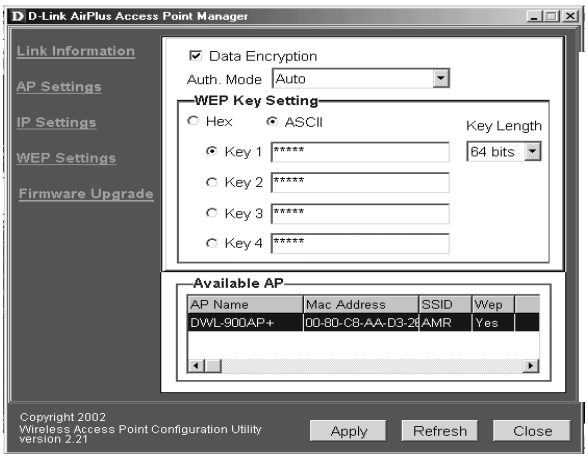


ปรับตั้งค่าของจุดเข้าถึงที่กำหนดชื่อให้ SSID เป็น AMR ซึ่งเป็นส่วนหนึ่งของการรักษาความปลอดภัย ช่องสัญญาณที่ใช้งานเป็นช่อง 6 และจุดเข้าถึงได้กำหนดชื่อให้เป็น DWL-900AP+

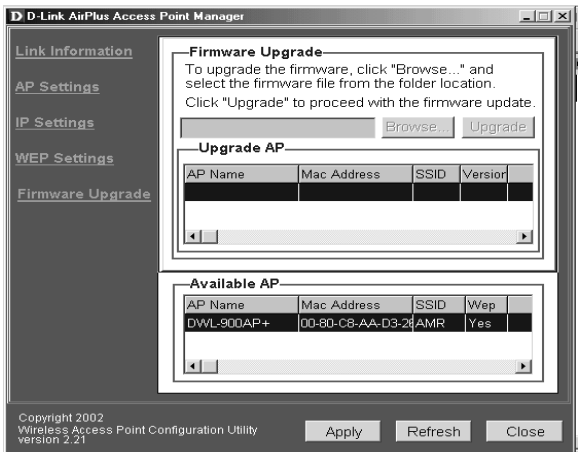


ปรับตั้งค่าของโพรโตคอล TCP/IP กำหนดหมายเลขแอดเดรส IP เป็น 192.168.0.50 ซึ่งเป็นหมายเลขเครือข่ายส่วนบุคคล กำหนดซับเน็ต-มาซคเป็น 255.255.255.0 ทำให้รองรับมาตรวัดได้ถึง 254 เครื่อง

รูปที่ 3.4 การปรับตั้งค่าการทำงานให้กับระบบแลนแบบไร้สาย



ปรับตั้งค่าการเข้ารหัสความปลอดภัย ที่กำหนดรหัสของกุญแจเป็น PEA

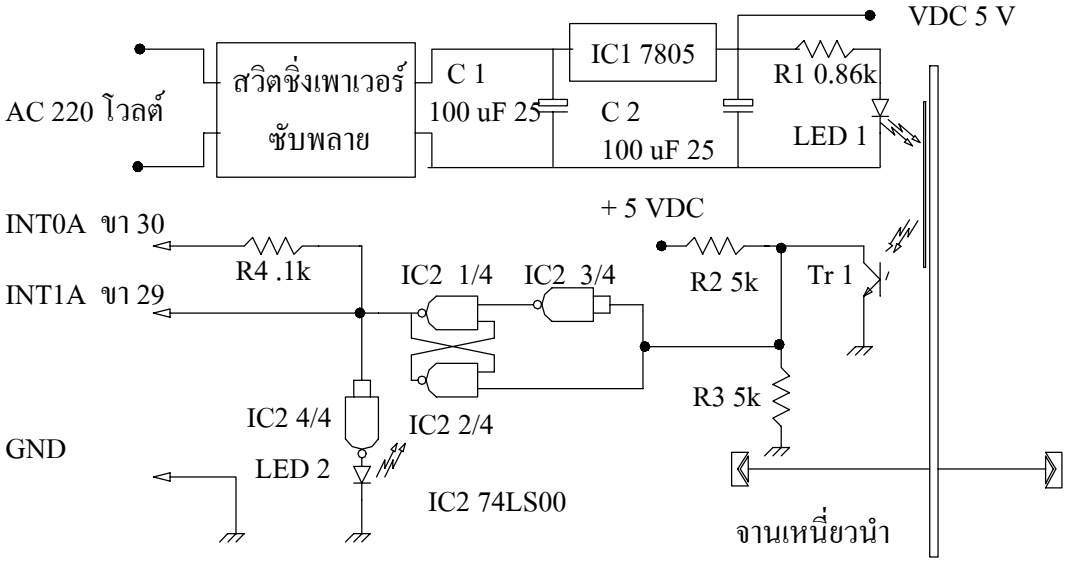


การปรับปรุงโปรแกรมสำหรับอุปกรณ์รับส่งข้อมูลระบบแลนไร้สายจากบริษัทผู้ผลิต

รูปที่ 3.4 (ต่อ)การปรับตั้งค่าการทำงานให้กับระบบแลนแบบไร้สาย

3.2.2 เชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับมาตรวัดหน่วยพลังงานไฟฟ้าแบบกิโลวัตต์-ชั่วโมงชนิดจานเหนี่ยวนำ ระบบหนึ่งเฟส

ไมโครคอนโทรลเลอร์แรบบิท 2000 ที่ติดตั้งกับมาตรวัดพลังงานจะต้องได้รับสัญญาณการอินเตอร์รัฟเมื่อจานเหนี่ยวนำของมาตรวัดพลังงานหมุนครบหนึ่งรอบจากจงจรตรวจจับการหมุนที่แสดงในรูปที่ 3.5 เพื่อนำไปนับจำนวนรอบการหมุนจนกว่าจะได้จำนวนรอบการหมุนตามที่มาตรวัดพลังงานกำหนดให้เป็นหนึ่งหน่วยพลังงาน ซึ่งมาตรวัดพลังงานไฟฟ้าชนิดจานเหนี่ยวนำหมุนส่วนใหญ่และที่ใช้ในงานพัฒนานี้มีรอบการหมุน 1200 รอบ



รูปที่ 3.5 วงจรส่วนที่ตรวจจับการหมุนของมาตรวัดพลังงาน

การออกแบบจงจรตรวจจับการหมุนได้อาศัยแรงดัน 220 โวลต์ ภายในมาตรวัดเป็นแหล่งจ่ายพลังงานผ่านทางสวิตชิ่งเพาเวอร์ซับพลายให้ไดโอดเปล่งแสง มีการติดตั้งแถบทึบแสงขนาดเล็กไว้ที่จานเหนี่ยวนำของมาตรวัด เพื่อให้โฟโต้ทรานซีสเตอร์ ซึ่งเป็นอุปกรณ์สำหรับรับแสงสร้างสัญญาณที่มีขอบขาขึ้นและขอบขาลง ตามจังหวะที่แถบทึบแสงหมุน

นอกจากนี้ยังมีการจัดรูปแบบของสัญญาณให้ดีขึ้นด้วยการใช้เกตแนนด์มาต่อเป็นลักษณะของดีฟลิปฟลอป ซึ่งสัญญาณเอาต์พุตจะแปรตามสัญญาณอินพุต และใช้เกตแนนด์ส่วนที่เหลือขับไดโอดเปล่งแสงเพื่อแสดงสภาวะของการเกิดสัญญาณ

การรับส่งข้อมูลด้านมาตรวัดได้พัฒนาโปรแกรมโดยอาศัยโปรแกรมไดนามิค ซี ให้เลือกกรณีการทำงานโดยอาศัยตัวแปรรหัสส่วนการทำงานเป็น 3 แบบ คือ ส่วนการปรับตั้งเวลาของไมโครคอนโทรลเลอร์ ส่วนติดตามการใช้ไฟฟ้า และส่วนการส่งหน่วยพลังงานรอบเดือน ดังแสดงขั้นตอนในการรับส่งข้อมูลตามตัวแปรรหัสส่วนการทำงานในรูปที่ 3.7 และขั้นตอนในการทำงานของโปรแกรมในรูปที่ 3.8 โดยในแต่ละส่วนการทำงานจะมีการจัดรูปแบบข้อมูลที่แตกต่างกัน ดังรูปที่ 3.6 และมีรายละเอียดรูปแบบของข้อมูลดังนี้

รูปแบบข้อมูลเมื่อรหัสส่วนการทำงานเป็นหนึ่ง

หมายเลข IP 192.168.0.xxx	รหัสส่วนการทำงาน = 1	เวลา ชั่วโมง-นาที-วินาที	วันเดือนปี วัน-เดือน-ปี

รูปแบบข้อมูลเมื่อรหัสส่วนการทำงานเป็นสอง

หมายเลข IP 192.168.0.xxx	รหัสส่วนการทำงาน = 2	พลังงาน = 1	เวลา ชั่วโมง-นาที-วินาที	วันเดือนปี วัน-เดือน-ปี

รูปแบบข้อมูลเมื่อรหัสส่วนการทำงานเป็นสาม

หมายเลข IP 192.168.0.xxx	รหัสส่วนการทำงาน = 3	พลังงาน ก.	พลังงาน ข.	เวลา ชั่วโมง-นาที-วินาที	วันเดือนปี วัน-เดือน-ปี

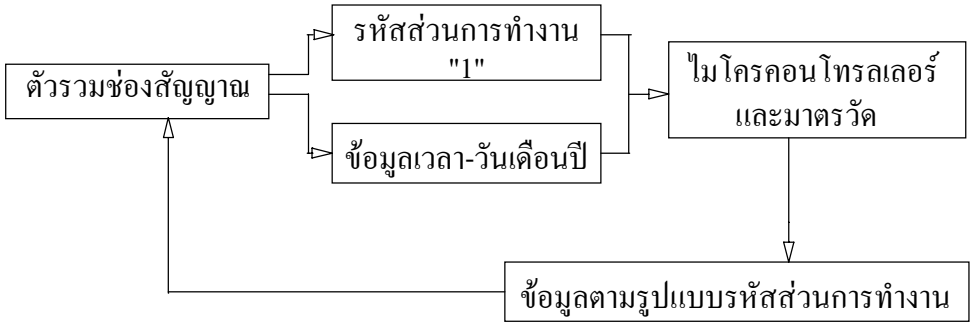
รูปที่ 3.6 การจัดรูปแบบข้อมูลซึ่งแตกต่างกันตามรหัสส่วนการทำงาน

หมายเลข IP แอดเดรส	เพื่อแจ้งให้ด้านตัวรวมช่องสัญญาณทราบว่า ข้อมูลที่รับได้ส่งมาจากมาตรวัดใด
รหัสส่วนการทำงาน	เพื่อให้โปรแกรมด้านตัวรวมช่องสัญญาณทราบว่าจะดำเนินการกับข้อมูลส่วนที่เหลือต่ออย่างไร
ข้อมูลพลังงาน=1	เพื่อติดตามการใช้ไฟฟ้า
ข้อมูลพลังงาน ก.	คือข้อมูลพลังงานรวมในช่วงเวลา 09.00 – 21.59 น. ที่ส่งให้ตัวรวมช่องสัญญาณ
ข้อมูลพลังงาน ข.	คือข้อมูลพลังงานรวมในช่วงเวลา 22.00 – 08.59 น. ที่ส่งให้ตัวรวมช่องสัญญาณ
ข้อมูลเวลา	คือเวลาที่มาตรวัดทำการส่งข้อมูลในขณะนั้น
ข้อมูลวันเดือนปี	คือวันเดือนปีที่มาตรวัดทำการส่งข้อมูลในขณะนั้น

ได้มีการติดตั้งไอซีช่วยติดตามการทำงานของไมโครคอนโทรลเลอร์เบอร์ 24LC16 ที่มีการติดต่อกับไมโครคอนโทรลเลอร์โดยใช้โพรโตคอล I^2C ดังได้กล่าวในรายละเอียดของโพรโตคอลแล้วในตอนท้ายของบทที่ 2 เพื่อบันทึกค่าสภาวะการทำงานและข้อมูลหน่วยพลังงาน เป็นการป้องกันการสูญหายของข้อมูลถ้าเกิดเหตุการณ์การทำงานของไมโครคอนโทรลเลอร์ล้มเหลว

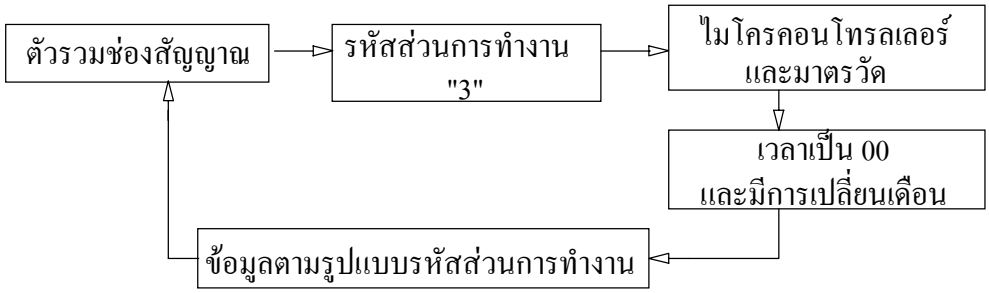
ส่วนของการปรับตั้งเวลาไมโครคอนโทรลเลอร์จะทำงานเมื่อได้รับรหัสส่วนการทำงานเป็นหนึ่ง ซึ่งจะมีข้อมูลเวลาและวันเดือนปี ส่งออกมาจากอุปกรณ์รวมช่องสัญญาณ ดังนั้น เวลาที่มาตรวัดพลังงานไฟฟ้าจะเป็นค่าเดียวกันกับอุปกรณ์รวมช่องสัญญาณเมื่อมีการปรับตั้งเวลาแล้ว

รหัสส่วนการทำงานเป็น "หนึ่ง" ปรับตั้งเวลาไมโครคอนโทรลเลอร์



รหัสส่วนการทำงานเป็น "สอง" ติดตามการใช้ไฟฟ้า

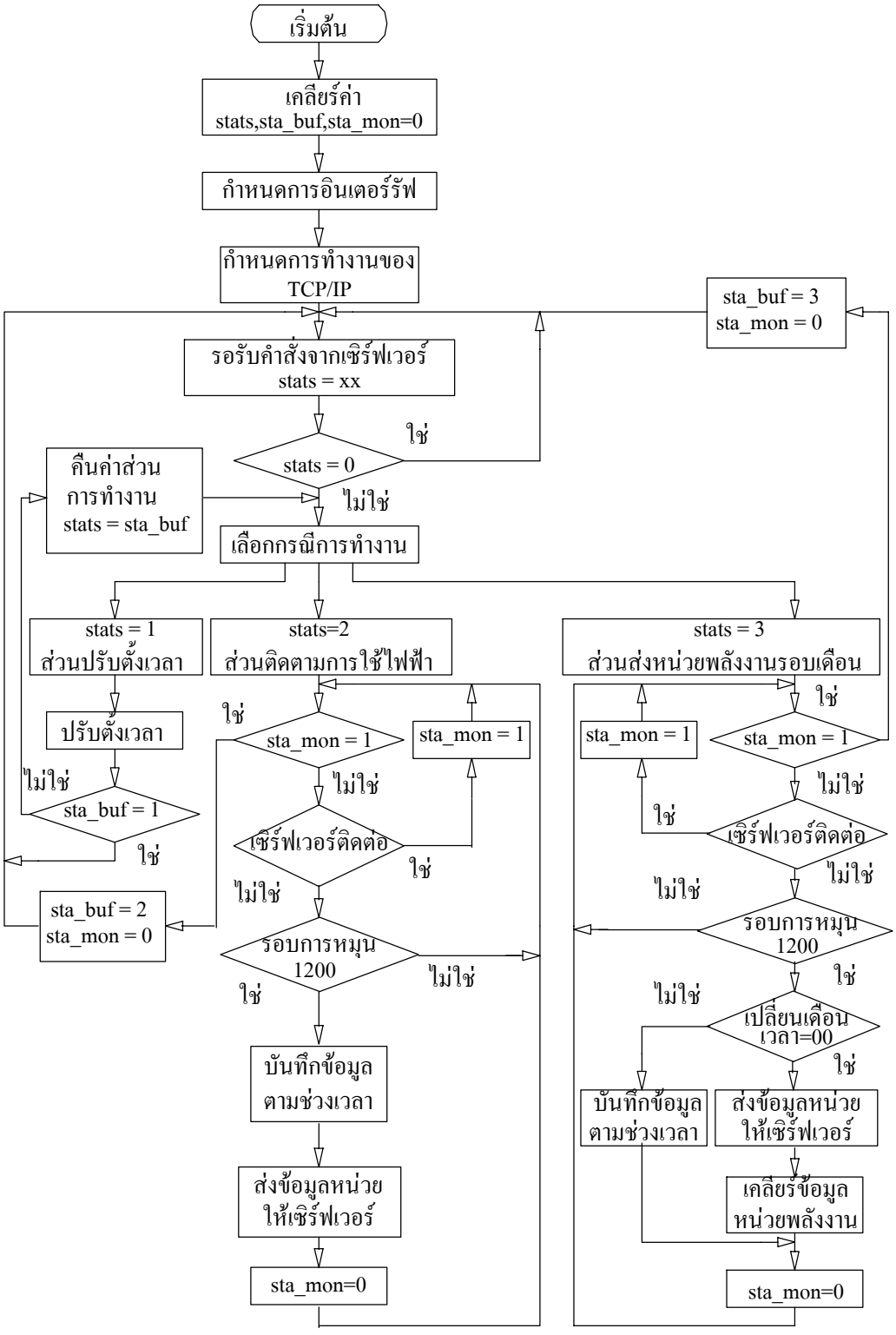
รหัสส่วนการทำงานเป็น "สาม" ส่งหน่วยพลังงานในรอบเดือน



รูปที่ 3.7 ขั้นตอนการรับส่งข้อมูลตามรหัสส่วนการทำงาน

ส่วนของการติดตามการใช้ไฟฟ้าจะทำงานเมื่อได้รับรหัสส่วนการทำงานเป็นสองซึ่งไมโครคอนโทรลเลอร์จะนับรอบการหมุนของมาตรวัดจนครบจำนวนรอบที่ตั้งไว้จึงจะทำการบันทึกข้อมูลและส่งข้อมูลหน่วยพลังงานให้อุปกรณ์รวมช่องสัญญาณหนึ่งครั้ง

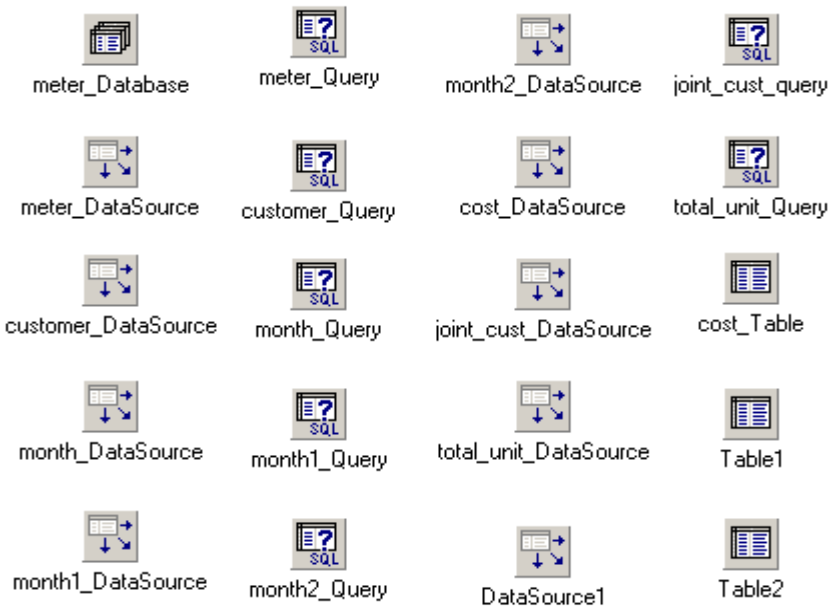
ส่วนของการส่งหน่วยพลังงานในรอบเดือนจะทำงานเมื่อได้รับรหัสส่วนการทำงานเป็นสาม ซึ่งไมโครคอนโทรลเลอร์จะบันทึกข้อมูลตามช่วงเวลาไปเรื่อยๆ เมื่อมีการใช้พลังงานครบหนึ่งหน่วย และจะส่งข้อมูลพลังงานรวมให้อุปกรณ์รวมช่องสัญญาณเมื่อมีการเริ่มต้นเดือนใหม่



รูปที่ 3.8 ขั้นตอนการทำงานของโปรแกรมรับส่งข้อมูลสำหรับไมโครคอนโทรลเลอร์

3.2.3 การเชื่อมต่อฐานข้อมูลและเขียนโปรแกรมการใช้งาน

ฐานข้อมูลที่ใช้ในงานพัฒนามีลักษณะการเข้าถึงเป็นฐานข้อมูลแบบไคลเอ็นต์/เซิร์ฟเวอร์ ด้วยการสร้างมอดูลของฐานข้อมูลในโปรแกรมเดลไฟ ดังแสดงในรูปที่ 3.9 ซึ่งเชื่อมต่อกับฐานข้อมูลชื่อ meter_db ภายในโปรแกรมฐานข้อมูลแอคเซส ผ่านการเชื่อมโยงของ ODBC ในระบบปฏิบัติการวินโดวส์ ซึ่งแสดงขั้นตอนในการเชื่อมโยงในรูปที่ 3.10 โดยอาศัยภาษา SQL ที่อยู่ภายในโปรแกรมเดลไฟ เป็นเครื่องมือช่วยสำหรับเข้าถึงข้อมูลตั้งแต่หนึ่งตารางขึ้นไป



รูปที่ 3.9 มอดูลข้อมูลของโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ

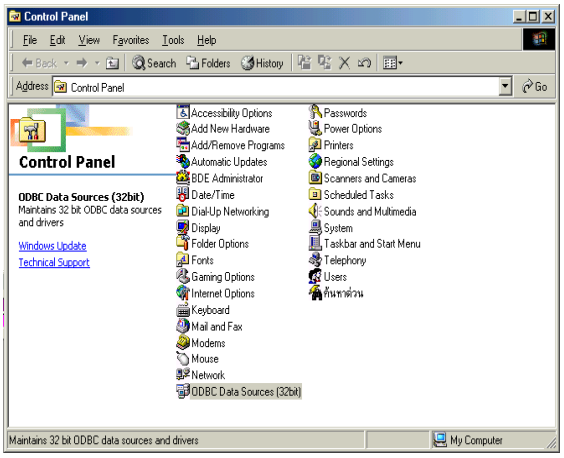
จุดมุ่งหมายของการสร้างมอดูลข้อมูลเพื่อให้หน้าจอต่างๆ ในโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ ที่มีความจำเป็นต้องใช้ตารางเดียวกันที่อยู่ในฐานข้อมูลเดียวกัน สามารถเข้าถึงรายละเอียดของข้อมูลในตารางนั้นได้


ส่วนประกอบฐานข้อมูล (data base component) ใช้ในการอ้างอิงถึงแหล่งของฐานข้อมูลซึ่งประกอบไปด้วยตารางตั้งแต่หนึ่งตารางขึ้นไป ผ่านการเชื่อมโยงจาก ODBC

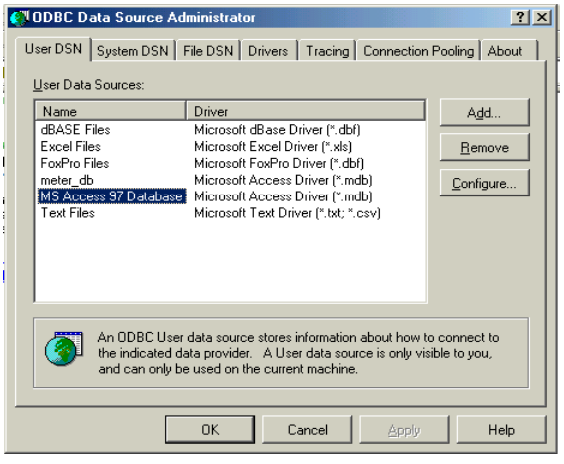

ส่วนประกอบแหล่งข้อมูล (data source component) เป็นเส้นทางในการส่งถ่ายข้อมูลระหว่างกลุ่มของข้อมูล (data set) กับส่วนประกอบในการรับรู้ข้อมูล (data aware component)

ส่วนประกอบคิวรี (query component) ใช้สำหรับการเลือกรายการบันทึก (record) และขอบเขตข้อมูล (field) จากตารางตั้งแต่หนึ่งตาราง หรือหลายตารางก็ได้ ซึ่งอยู่ภายในส่วนประกอบฐานข้อมูล

ส่วนประกอบตาราง (table component) ใช้สำหรับการเลือกกลุ่มของรายการบันทึกที่มีโครงสร้างของขอบเขตข้อมูลเหมือนกัน ซึ่งอยู่ภายในส่วนประกอบฐานข้อมูล

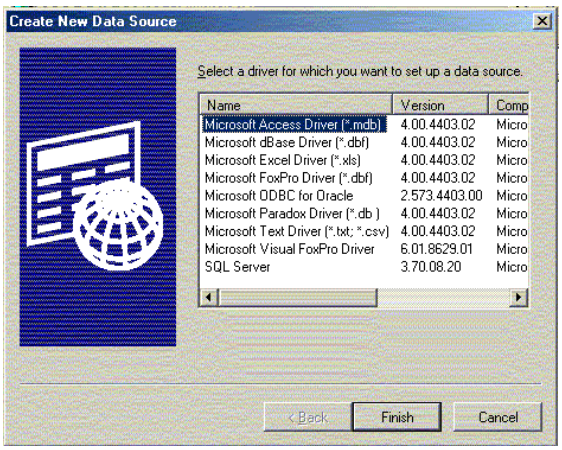


เปิดหน้าจอ control panel เลือกไอคอน ODBC data source (32 bit) เพื่อเข้าสู่หน้าจอ ODBC data source administrator

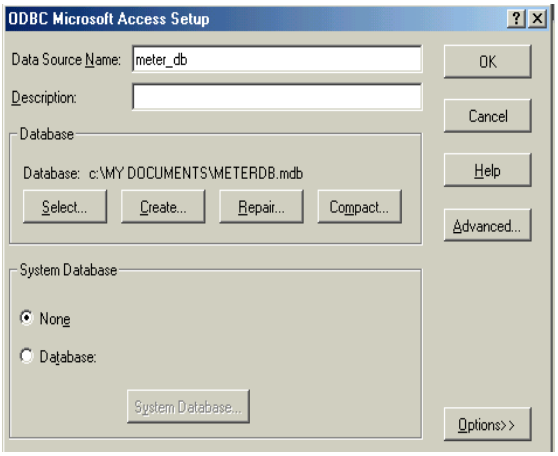


เลือกประเภทของแหล่งฐานข้อมูลที่ใช้ กดปุ่ม add เพื่อเข้าสู่หน้าจอ create new data source

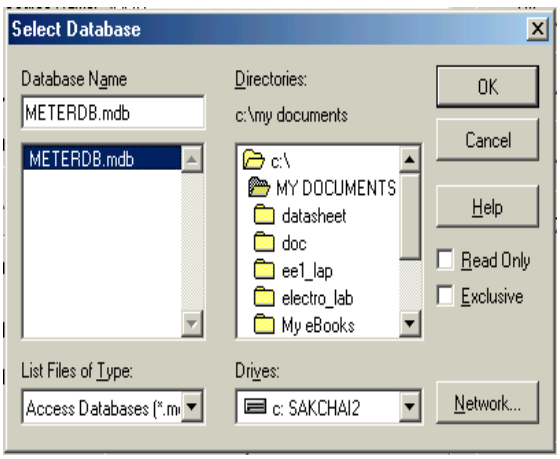
รูปที่ 3.10 ขั้นตอนการเชื่อมต่อฐานข้อมูลด้วย ODBC



เลือกประเภทของไดเวอร์ กดปุ่ม finish เพื่อเข้าสู่หน้าจอ ODBC microsoft access setup



กำหนดชื่อ data source name เป็น meter_db เพื่อใช้อ้างถึงฐานข้อมูลที่สร้าง กดปุ่ม select เพื่อเลือกตำแหน่งของแฟ้มข้อมูลที่เป็นฐานข้อมูล



เลือกแฟ้มฐานข้อมูล ซึ่งในงานพัฒนานี้เป็น c:\mydocuments\meter_db

รูปที่ 3.10 (ต่อ) ขั้นตอนการเชื่อมต่อฐานข้อมูลด้วย ODBC

ได้มีการออกแบบสร้างตารางฐานข้อมูลที่อยู่ภายในโปรแกรมแอคเซสเพื่อรอรับคำสั่งในการปฏิบัติการกับข้อมูลจากโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติหลายตารางด้วยกัน ซึ่งในแต่ละตารางได้อาศัยขอบเขตข้อมูลชื่อ cust_no เป็นกุญแจหลัก (primary key) ในการอ้างอิง

ตาราง customer ใช้ในการบันทึกประวัติของผู้ใช้ไฟฟ้า ประกอบด้วย ชื่อ นามสกุล ที่อยู่ที่หนึ่ง ที่อยู่ที่สอง และหมายเลขโทรศัพท์

ตาราง meter ใช้ในการบันทึกประวัติของมาตรวัดพลังงาน ประกอบด้วย หมายเลขแอดเดรส IP หมายเลขของการไฟฟ้า โรงงานที่ผลิต หมายเลขที่ผลิต วันแรกที่ติดตั้งใช้งาน อัตราการทนกระแสไฟฟ้า รอบการหมุน และหน่วยพลังงานที่คงค้างในมาตรวัด

ตาราง energy_cost ใช้ในการบันทึกอัตราค่าพลังงานไฟฟ้าตามอัตราของการไฟฟ้าส่วนภูมิภาค

ตาราง month_unit1 ใช้บันทึกหน่วยของพลังงานในช่วงเวลา 09.00 – 21.59 น. ของทุกวันในรอบหนึ่งเดือน เป็นระยะเวลาหนึ่งปี

ตาราง month_unit2 ใช้บันทึกหน่วยของพลังงานในช่วงเวลา 22.00 – 08.59 น. ของทุกวันในรอบหนึ่งเดือน เป็นระยะเวลาหนึ่งปี

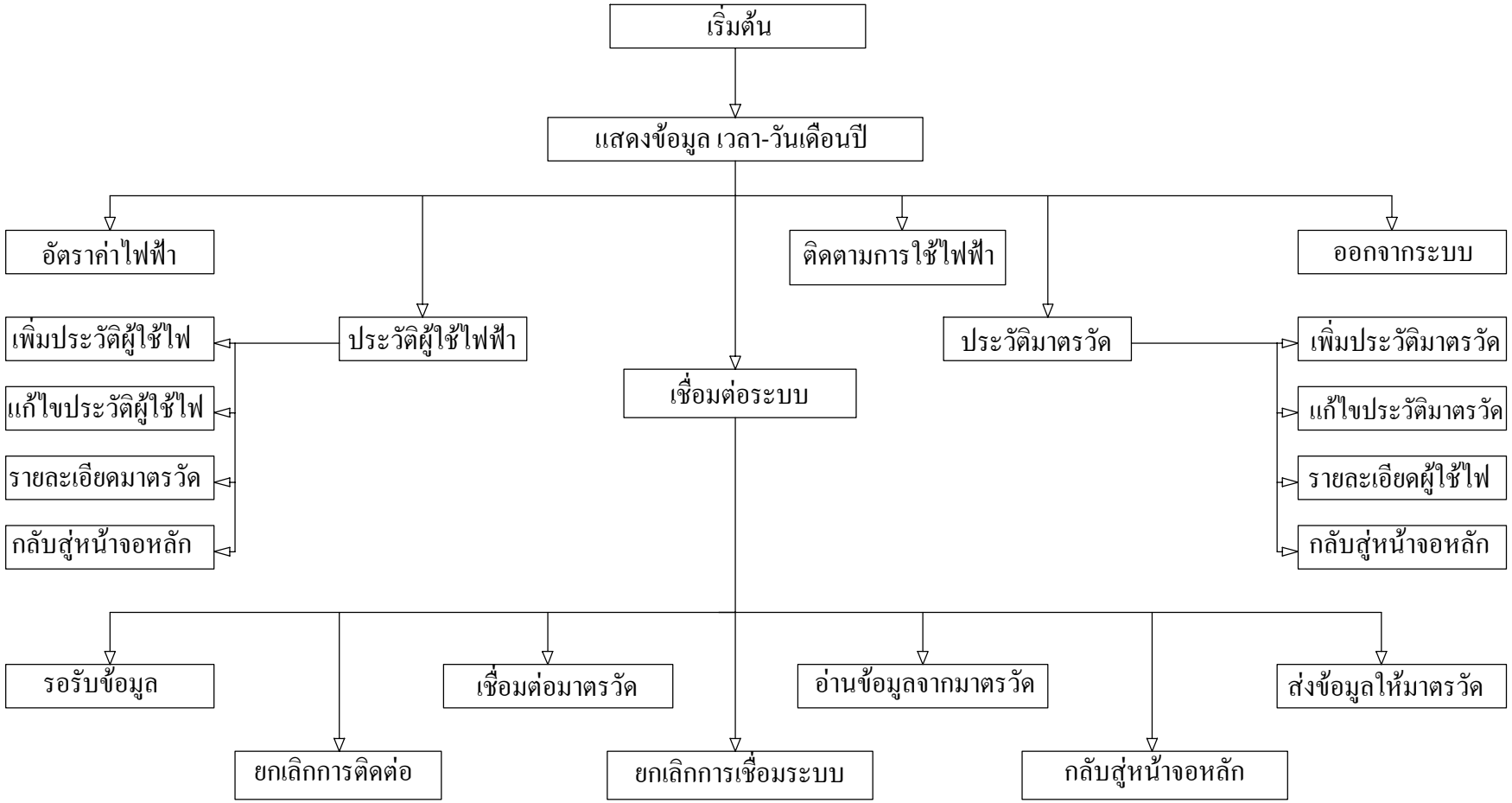
ตาราง month_total ใช้บันทึกผลรวมของหน่วยพลังงานที่ได้จากตาราง month_unit1 และ month_unit2 ในรอบหนึ่งเดือน เป็นระยะเวลาหนึ่งปี

ตาราง profile1 ใช้บันทึกหน่วยของพลังงานในทุกครั้งที่มีการรับข้อมูลจากมาตรวัด

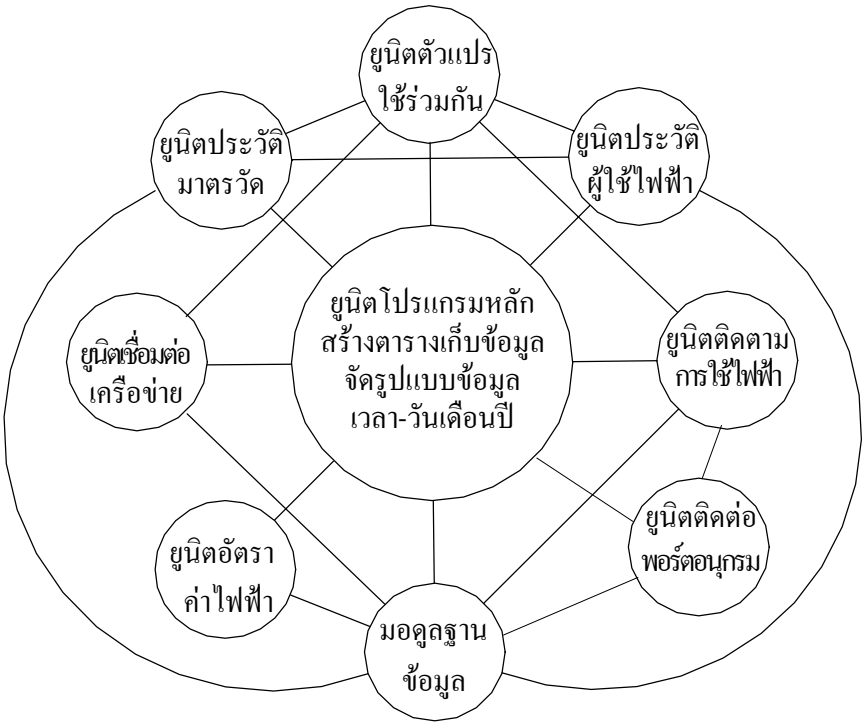
ในกรณีที่มีการติดตามการใช้พลังงานไฟฟ้า จะมีการสร้างตารางขึ้นใหม่ เพื่อเก็บข้อมูลหน่วยพลังงานของทุก 15 นาที ในรอบหนึ่งวัน โดยตารางที่สร้างขึ้นจะมีชื่อตามขอบเขตข้อมูลหมายเลขผู้ใช้ไฟฟ้า (cust_no) ที่อ้างอิงถึง

สำหรับโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติได้พัฒนาขึ้นมาจากโปรแกรมเดลไฟ ซึ่งออกแบบโครงสร้างของโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติดังแสดงในรูปที่ 3.11 ภายในโปรแกรมได้แยกออกเป็นส่วนย่อยๆจำนวนทั้งหมด 8 ยูนิต และ 1 มอดูล เพื่อให้ง่ายต่อการพัฒนา ดังแสดงในรูปที่ 3.12 ซึ่งแต่ละส่วนมีขั้นตอนและแอปพลิเคชั่นให้ใช้งาน ดังนี้

ยูนิตโปรแกรมหลัก ใช้ในการสร้างตารางเก็บข้อมูล จัดรูปแบบของข้อมูลเพื่อส่งต่อไปเก็บในฐานข้อมูลที่อยู่ภายในโปรแกรมแอคเซส รวมถึงการเข้าถึงข้อมูลเพื่อส่งผ่านต่อให้ยูนิตย่อยอื่นๆ ต่อไป



รูปที่ 3.11 โครงสร้างโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ



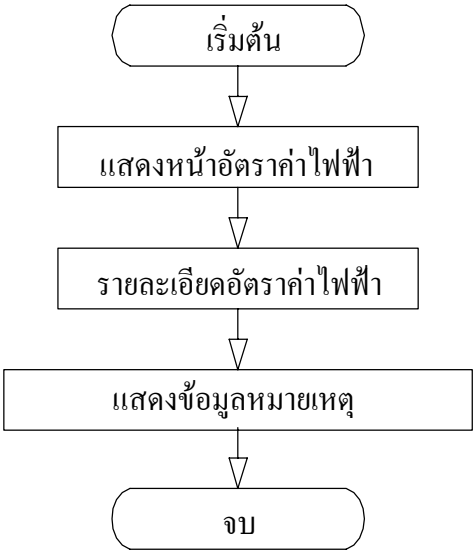
รูปที่ 3.12 ความสัมพันธ์ของยูนิตย่อยที่ทำงานร่วมกับยูนิตโปรแกรมหลัก

ยูนิตอัตราค่าไฟฟ้า แสดงข้อมูลอัตราค่าพลังงานไฟฟ้า จากตาราง energy_cost และข้อมูลหมายเหตุของอัตราค่าพลังงานไฟฟ้าจากแฟ้ม c:\my documents\comment_rate.txt มีขั้นตอนในการทำงาน ดังรูปที่ 3.13

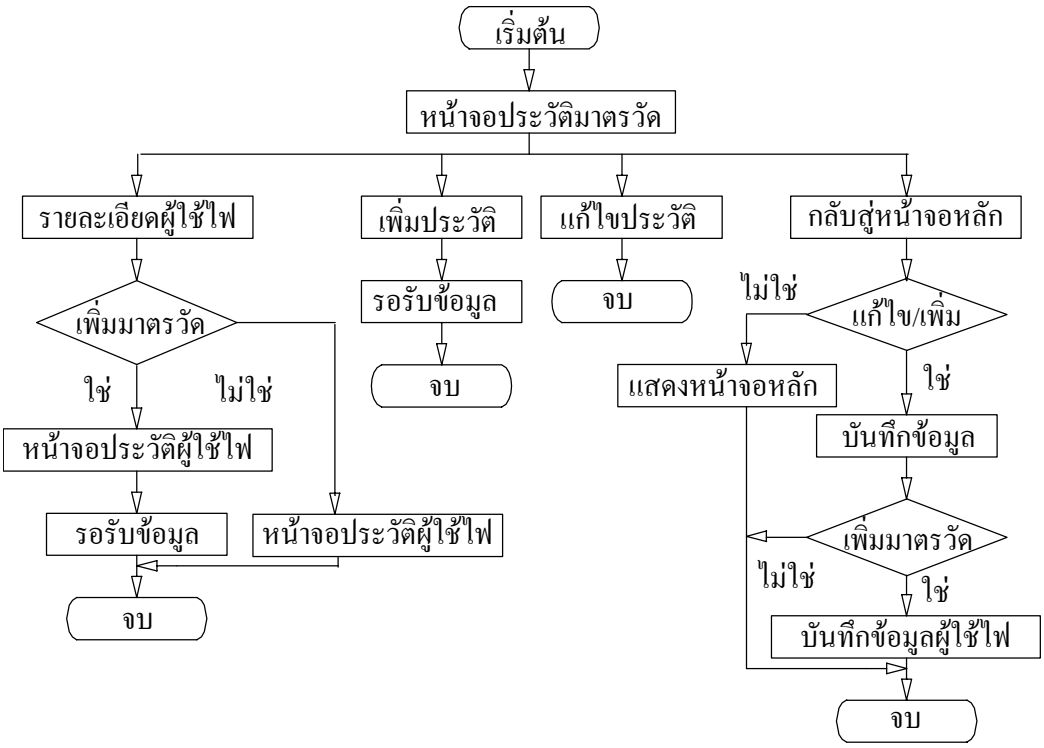
ยูนิตประวัติมาตรวัด แสดงข้อมูลประวัติมาตรวัด การเพิ่มการแก้ไขประวัติของมาตรวัดจากตาราง meter และแสดงรายละเอียดของผู้ใช้ไฟฟ้าที่มีลำดับของกุญแจหลักตรงกับกุญแจหลักของข้อมูลประวัติมาตรวัด มีขั้นตอนในการทำงานดังแสดงในรูปที่ 3.14

ยูนิตประวัติผู้ใช้ไฟฟ้า แสดงข้อมูลประวัติผู้ใช้ไฟฟ้า การเพิ่มการแก้ไขประวัติของผู้ใช้ไฟฟ้าจากตาราง customer และแสดงรายละเอียดของมาตรวัดที่มีลำดับของกุญแจหลักตรงกับกุญแจหลักของข้อมูลประวัติผู้ใช้ไฟฟ้า มีขั้นตอนในการทำงาน ดังรูปที่ 3.15

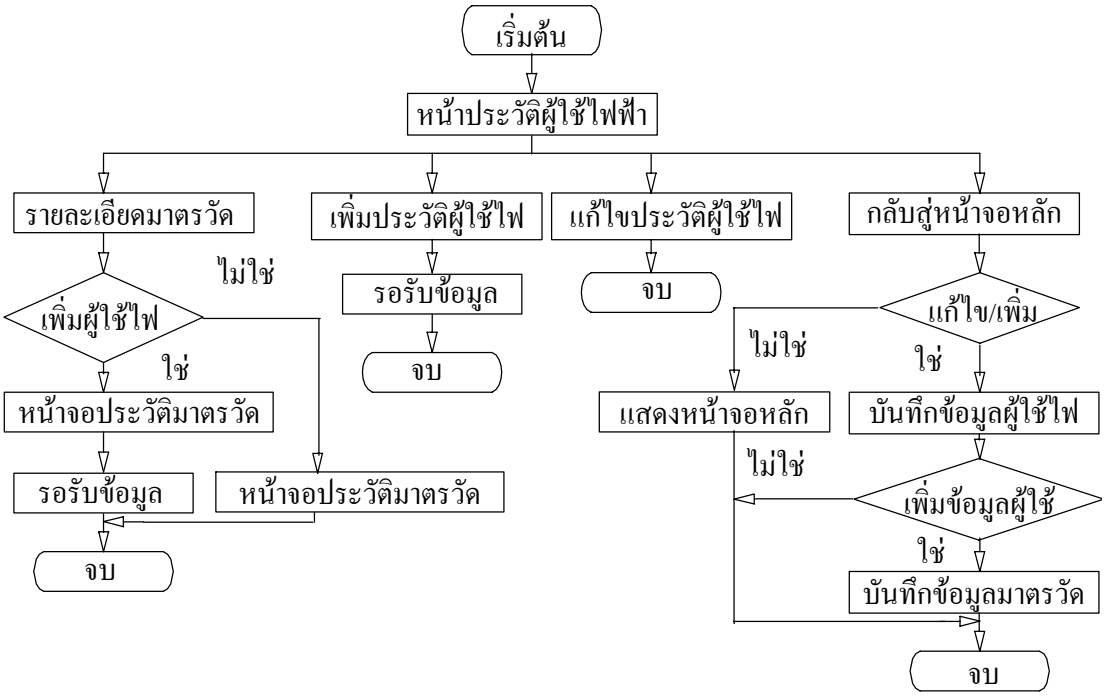
ยูนิตติดตามการใช้ไฟฟ้า แสดงข้อมูลหน่วยพลังงานไฟฟ้าที่ติดตามในรอบวันจากตารางที่สร้างขึ้นใหม่อัตโนมัติ ชื่อ t_cust_h+ “หมายเลขผู้ใช้ไฟฟ้า” และคำนวณอัตราค่าพลังงานไฟฟ้าโดยอาศัยข้อมูลจากตาราง month_unit1, month_unit2 และ month_total มีขั้นตอนในการทำงานดังแสดงในรูปที่ 3.16



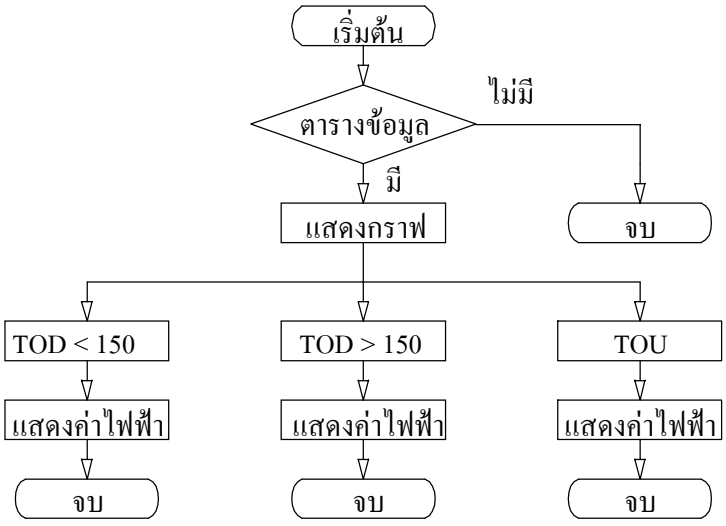
รูปที่ 3.13 ขั้นตอนการทำงานของยูนิตอัตราค่าไฟฟ้า



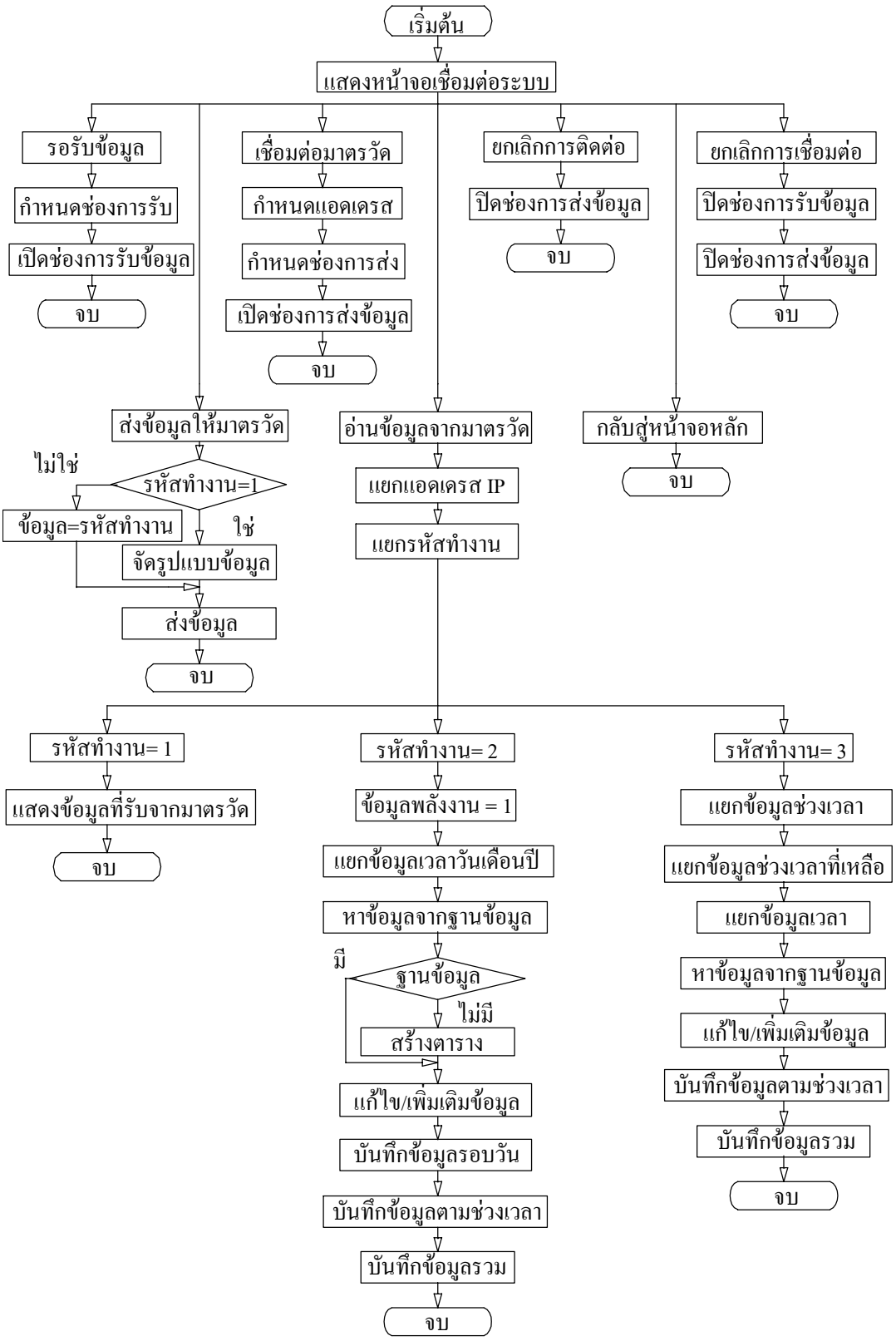
รูปที่ 3.14 ขั้นตอนการทำงานของยูนิตประวัติมาตรวัด



รูปที่ 3.15 ขั้นตอนการทำงานของยูนิตประวัติผู้ใช้ไฟฟ้า



รูปที่ 3.16 ขั้นตอนการทำงานของยูนิตติดตามการใช้ไฟฟ้า

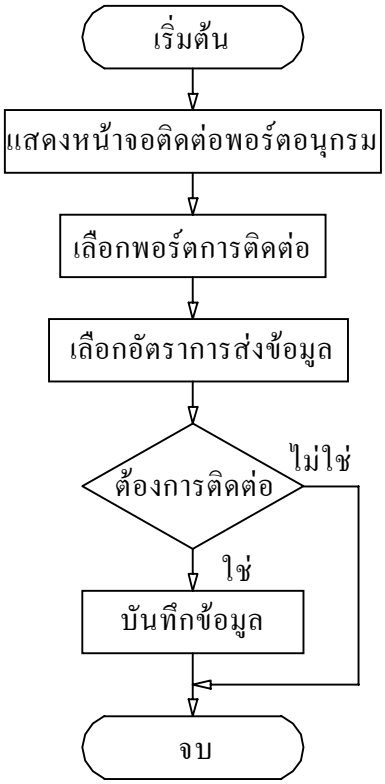


รูปที่ 3.17 ขั้นตอนการทำงานของยูนิตเชื่อมต่อเครือข่าย

ยูนิตเชื่อมต่อเครือข่าย ทำหน้าที่ติดต่อกับมาตรวัดและให้มาตรวัดทำหน้าที่ตามที่ผู้ใช้งานโปรแกรมกำหนด คือ ปรับตั้งเวลามาตรวัด หรือติดตามการใช้ไฟฟ้า หรือส่งหน่วยพลังงานในรอบเดือน รวมถึงการรอรับข้อมูลหรือการอ่านข้อมูลจากมาตรวัด เพื่อนำไปจัดเก็บในฐานข้อมูลต่อไป มีขั้นตอนในการทำงานดังแสดงในรูปที่ 3.17

ยูนิตติดต่อพอร์ตอนุกรมทำหน้าที่ติดต่อกับมาตรวัด เพื่อให้มาตรวัดส่งข้อมูลที่มาตรวัดบันทึกไว้เมื่อทำการส่งข้อมูลผ่านระบบแลนแบบไร้สายในแต่ละครั้ง ซึ่งข้อมูลที่ได้จะใช้ประโยชน์สำหรับเปรียบเทียบการทำงานของระบบ มีขั้นตอนในการทำงานดังแสดงในรูปที่ 3.18

ยูนิตมอดูลฐานข้อมูล ใช้เป็นศูนย์กลางในการจัดการกับข้อมูล ดังได้กล่าวในรายละเอียดแล้วในตอนต้นของหัวข้อ 3.2.3



รูปที่ 3.18 ขั้นตอนการทำงานของยูนิตติดต่อพอร์ตอนุกรม

3.2.4 การหาค่าความต้องการการใช้กำลังงานไฟฟ้า

ค่าความต้องการการใช้กำลังงานไฟฟ้า เป็นค่ากิโลวัตต์รวมที่เวลาทุก 15 นาที แต่มาตรวัดหน่วยพลังงานไฟฟ้าในการพัฒนานี้เป็นแบบจานหมุนมีหน่วยเป็น กิโลวัตต์-ชั่วโมง ซึ่งเป็นหน่วยของการใช้พลังงานไฟฟ้า จึงต้องนำหน่วยของพลังงานไฟฟ้าดังกล่าวมาปรับเวลาให้อยู่ในรูปของ

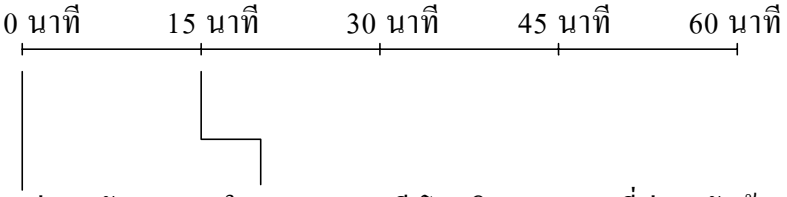
กิโลวัตต์-นาที และเปรียบเทียบหน่วยดังกล่าวกับช่วงเวลา 15 นาที ดังสมการที่ 3.1 และ 3.2 ทำให้สรุปได้ว่า ค่าความต้องการการใช้กำลังงานไฟฟ้าเกิดจากการรวมหน่วยของพลังงานไฟฟ้าในทุกๆ 15 นาที แล้วนำค่าที่ได้มาคูณด้วย 4 อีกครั้ง ดังสมการที่ 3.3 จึงได้มีการออกแบบโปรแกรมให้รวมหน่วยของพลังงานไฟฟ้าในทุกๆ 15 นาที โดยใช้ข้อมูลเวลาที่จัดส่งมาพร้อมกับข้อมูลมาเป็นฐานเวลาในการพิจารณาด้วย ดังรูปที่ 3.19

$$\text{ความต้องการกำลังงานไฟฟ้า} = \frac{(P_1t_1 / 60) + (P_2t_2 / 60) + \ldots + (P_nt_n / 60)}{(15 / 60)} \quad (3.1)$$

$$\text{ความต้องการกำลังงานไฟฟ้า} = \frac{P_1T_1 + P_2T_2 + \ldots + P_nT_n}{(1/4)} \quad (3.2)$$

$$\text{ความต้องการกำลังงานไฟฟ้า} = 4 \times (E_1 + E_2 + \ldots + E_n) \quad (3.3)$$

เมื่อ P คือ กำลังงานไฟฟ้า (กิโลวัตต์)
T คือ เวลา (ชั่วโมง)
t คือ เวลา (นาที)
E คือ พลังงานไฟฟ้า (กิโลวัตต์ – ชั่วโมง)

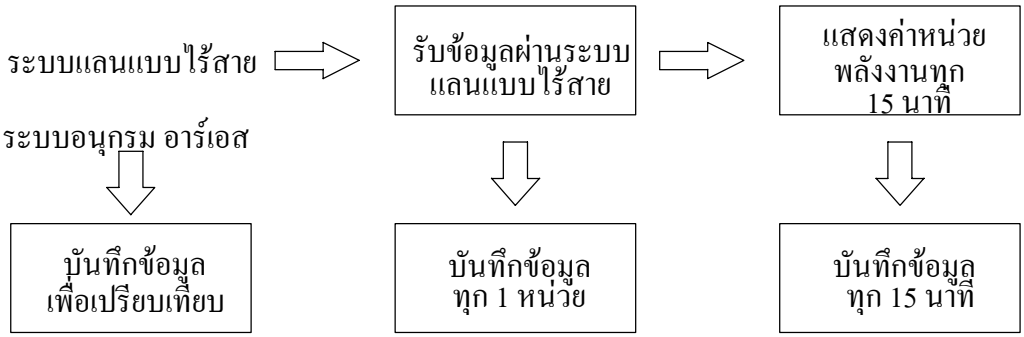


รูปที่ 3.19 ขั้นตอนการหาค่าความต้องการการใช้กำลังงานไฟฟ้า

เพื่อให้ค่าความต้องการการใช้กำลังงานไฟฟ้ามีความละเอียดขึ้น จึงกำหนดให้ทำการนับจำนวนรอบการหมุนที่ไมโครคอนโทรลเลอร์เป็น 12 รอบ ซึ่งมาตรวัดพลังงานไฟฟ้าหมุน 1200 รอบได้จำนวนหน่วย 1 หน่วยกิโลวัตต์-ชั่วโมง ไมโครคอนโทรลเลอร์นับรอบ 12 รอบ จะได้จำนวนหน่วยเป็น 0.01 หน่วยกิโลวัตต์-ชั่วโมง

3.2.5 การตรวจสอบการทำงานของระบบ

การทำงานของระบบที่สภาวะของการทำงานปกติได้อาศัยการรับส่งข้อมูลผ่านทางระบบเครือข่ายแลนแบบไร้สายซึ่งได้มีการบันทึกข้อมูลทุกครั้งที่มีการรับส่งข้อมูลหน่วยพลังงานไฟฟ้า ข้อมูลอีกส่วนหนึ่งถูกนำไปรวมกันเพื่อหาค่าความต้องการในการใช้กำลังงานไฟฟ้า ก่อนที่จะทำการบันทึกข้อมูลอีกครั้ง ดังรูปที่ 3.20 ในการเปรียบเทียบข้อมูล ได้อาศัยไมโครคอนโทรลเลอร์ให้ทำการเก็บข้อมูลทุกครั้งที่มีการส่งข้อมูลหน่วยพลังงานและจะทำการเรียกดูข้อมูลเพื่อเปรียบเทียบผ่านทางพอร์ตอนุกรม อาร์เอส 232



รูปที่ 3.20 ขั้นตอนการเก็บข้อมูลเพื่อตรวจสอบการทำงานของระบบ

รูปที่ 3.21 ระบบต้นแบบของงานพัฒนา

ในการประกอบระบบต้นแบบได้ประกอบอุปกรณ์รับส่งของระบบแลนแบบไร้สาย ไมโครคอนโทรลเลอร์และมาตรวัดพลังงานไฟฟ้า ไว้ในกล่องให้เป็นชุดเดียวกัน เพื่อให้ง่ายต่อการติดตั้งในพื้นที่ทดสอบและป้องกันไม่ให้ระบบต้นแบบเกิดการชำรุดเสียหาย ดังแสดงในรูปที่ 3.21

3.3 เครื่องมือที่ใช้ในการพัฒนา

- โปรแกรมคอมพิวเตอร์เดลไฟ 5
- โปรแกรมคอมพิวเตอร์แอคเซส 97
- โปรแกรมคอมพิวเตอร์ไดนามิค ซี 7.06
- อุปกรณ์รับส่งข้อมูลบนแลนแบบไร้สาย
- ไมโครคอนโทรลเลอร์แรบบิท 2000
- มาตรวัดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ ระบบหนึ่งเฟส
- เครื่องคอมพิวเตอร์เพนเทียม 1.1 GHz

3.4 การเก็บรวบรวมข้อมูล

เก็บข้อมูลหน่วยพลังงานที่มาตรวัดกิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำ ระบบหนึ่งเฟส และข้อมูลที่รับได้ของอุปกรณ์รวมช่องสัญญาณ

3.5 การวิเคราะห์ข้อมูล

ใช้การเปรียบเทียบความถูกต้องของข้อมูลหน่วยพลังงานที่มาตรวัดกิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำ ระบบหนึ่งเฟสที่ติดตั้งในระบบ กับหน่วยพลังงานที่ได้ของอุปกรณ์รวมช่องสัญญาณ

3.6 บทสรุป

การรับส่งข้อมูลโดยอาศัยระบบแลนแบบไร้สายจะต้องมีการปรับตั้งค่า SSID ช่องของการสื่อสารและรหัสกุญแจ WEP ให้ตรงกัน หมายเลขแอดเดรส IP ที่กำหนดจะต้องไม่ซ้ำกันจึงจะสามารถติดต่อกันได้ ในส่วนของโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติที่พัฒนาขึ้นจะมีการจัดเก็บฐานข้อมูลต่างๆของระบบ เช่น ข้อมูลประวัติของมาตรวัด ข้อมูลประวัติของผู้ใช้ไฟฟ้า ข้อมูลหน่วยการใช้พลังงานไฟฟ้าในรอบวันและในรอบเดือน ด้านมาตรวัดจะทำหน้าที่เก็บรวบรวมหน่วยพลังงานไฟฟ้าเพื่อจัดส่งให้อุปกรณ์รวมช่องสัญญาณตามเงื่อนไขที่กำหนดมาให้จากอุปกรณ์รวมช่องสัญญาณ ซึ่งจะทำงานสัมพันธ์กันโดยมีหน้าที่หลักในการทำงานของโปรแกรม คือ การปรับเวลาให้กับมาตรวัด การติดตามการใช้ไฟฟ้า และการส่งหน่วยพลังงานไฟฟ้าในรอบเดือน

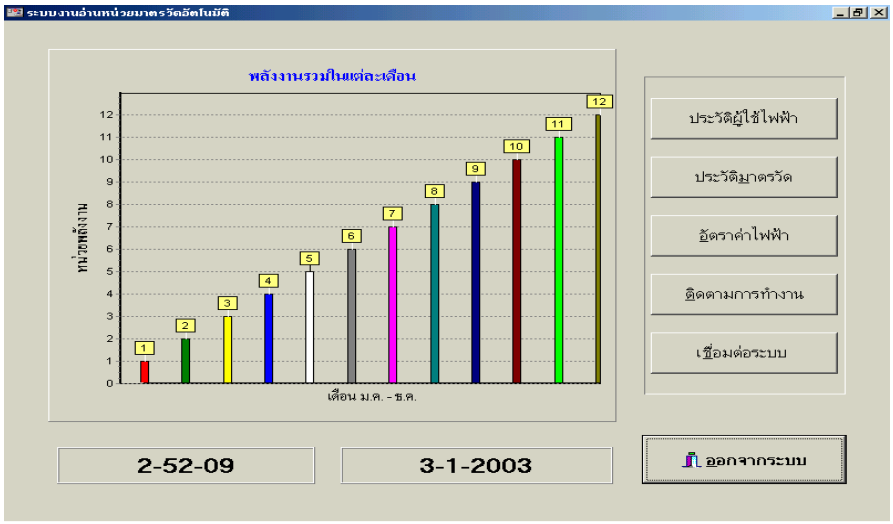
บทที่ 4
การทดสอบและอภิปรายผล

4.1 บทนำ

กล่าวถึงการใช้งานของโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ ที่ใช้ในการควบคุมการทำงานของมาตรวัดพลังงานไฟฟ้าและการแสดงผลข้อมูล ซึ่งได้มีการติดตั้งเพื่อทดสอบระบบเป็นจำนวน 2 แห่ง ที่มีระยะทางและสภาพแวดล้อมแตกต่างกัน เป็นระยะเวลา 24 ชั่วโมง 7 วัน และ 1 เดือน นำผลที่ได้มาหาความผิดพลาดของระบบ

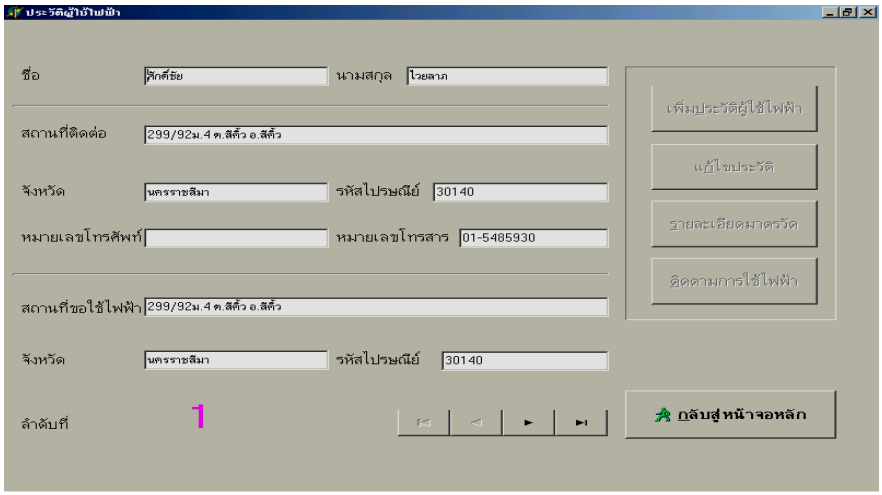
4.2 การใช้งานโปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ

หน้าจอหลักของโปรแกรม ดังแสดงในรูปที่ 4.1 ประกอบด้วยกราฟการใช้พลังงานไฟฟ้ารวมของมาตรวัดในรอบเดือนที่เก็บในฐานข้อมูลของอุปกรณ์รวมช่องสัญญาณ มีการแสดงเวลาวันเดือนปี และตัวเลือกยูนิตย่อย 5 ยูนิต คือ ยูนิตประวัติผู้ใช้ไฟฟ้า ยูนิตประวัติมาตรวัด ยูนิตอัตราค่าไฟฟ้า ยูนิตติดตามการทำงานของมาตรวัด และยูนิตเชื่อมต่อระบบ ซึ่งจะกล่าวในรายละเอียดต่อไป นอกจากนี้ยังมีปุ่มสำหรับการออกจากระบบงานอ่านหน่วยมาตรวัดอัตโนมัติด้วย



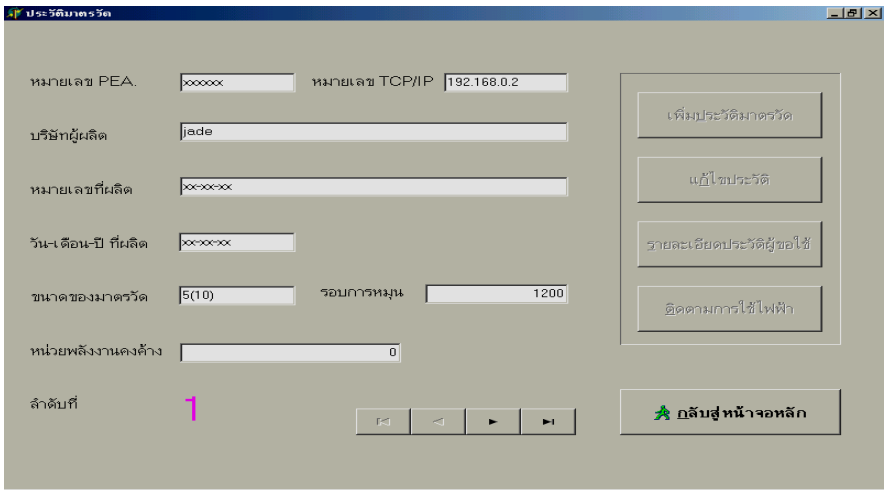
รูปที่ 4.1 หน้าจอระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ

ยูนิตประวัติผู้ใช้ไฟฟ้ามีหน้าจอดังแสดงในรูปที่ 4.2 ประกอบด้วยรายละเอียดประวัติผู้ใช้ไฟฟ้าที่อยู่ภายในฐานข้อมูล ซึ่งสามารถเพิ่มเติมหรือแก้ไขประวัติผู้ใช้ไฟฟ้าได้ มีปุ่มสำหรับการเชื่อมต่อยูนิตประวัติของมาตรวัดในกรณีที่มีการเพิ่มจำนวนประวัติผู้ใช้ไฟฟ้า ปุ่มสำหรับเชื่อมต่อยูนิตติดตามการทำงาน และปุ่มการกลับเข้าสู่หน้าจอหลักของโปรแกรม



รูปที่ 4.2 หน้าจอประวัติผู้ใช้ไฟฟ้า

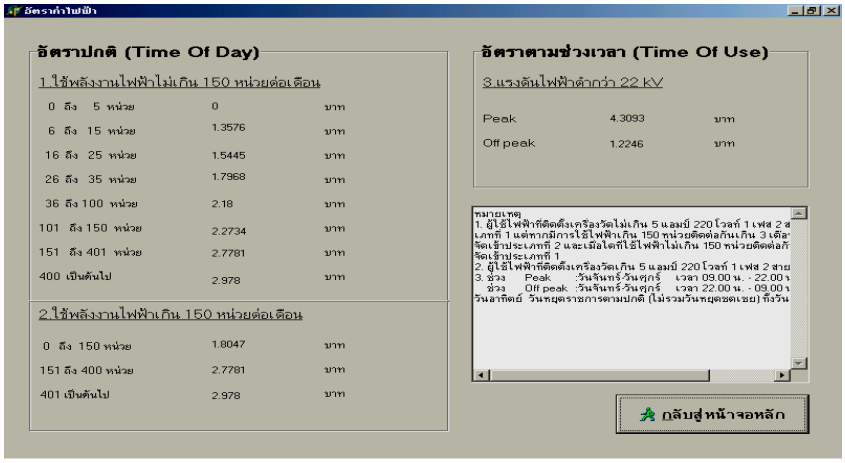
ยูนิตประวัติมาตรวัดมีหน้าจอแสดงดังรูปที่ 4.3 ประกอบด้วยรายละเอียดประวัติของมาตรวัด



รูปที่ 4.3 หน้าจอประวัติมาตรวัด

พลังงานไฟฟ้าที่อยู่ภายในฐานข้อมูล ซึ่งสามารถเพิ่มเติมหรือแก้ไขประวัติของมาตรวัดได้ มีปุ่มสำหรับการเชื่อมต่อยูนิตประวัติของผู้ใช้ไฟในกรณีที่มีการเพิ่มประวัติของมาตรวัด ปุ่มสำหรับเชื่อมต่อยูนิตติดตามการทำงาน และปุ่มการกลับเข้าสู่หน้าจอหลักของโปรแกรม

ยูนิตอัตราค่าไฟฟ้ามีหน้าจอดังรูปที่ 4.4 เป็นรายละเอียดของอัตราค่าไฟฟ้าที่อธิบายตามหลักเกณฑ์ของการไฟฟ้าส่วนภูมิภาค ในงานพัฒนานี้ สามารถแบ่งได้เป็น 3 อัตรา คือ อัตราปกติ (time of day : TOD) กรณีหน่วยพลังงานไฟฟ้าไม่เกิน 150 หน่วย อัตราปกติ กรณีหน่วยพลังงานไฟฟ้าเกิน 150 หน่วย และอัตราตามช่วงเวลา (time of use : TOU)

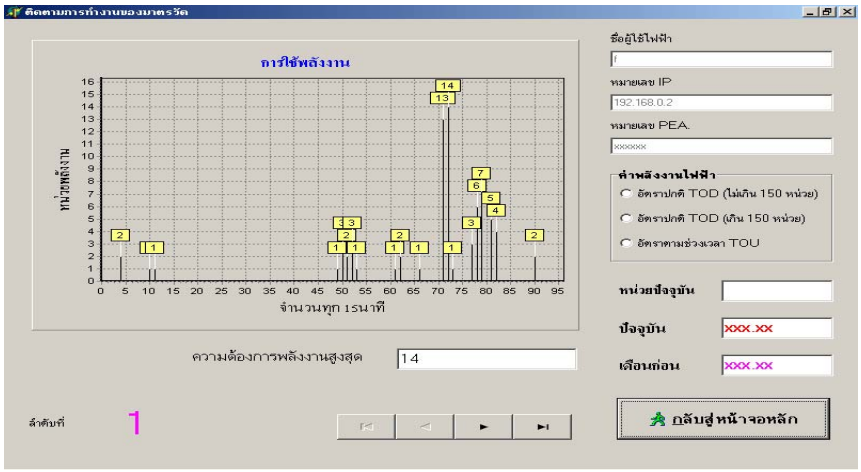


รูปที่ 4.4 หน้าจออัตราค่าไฟฟ้า

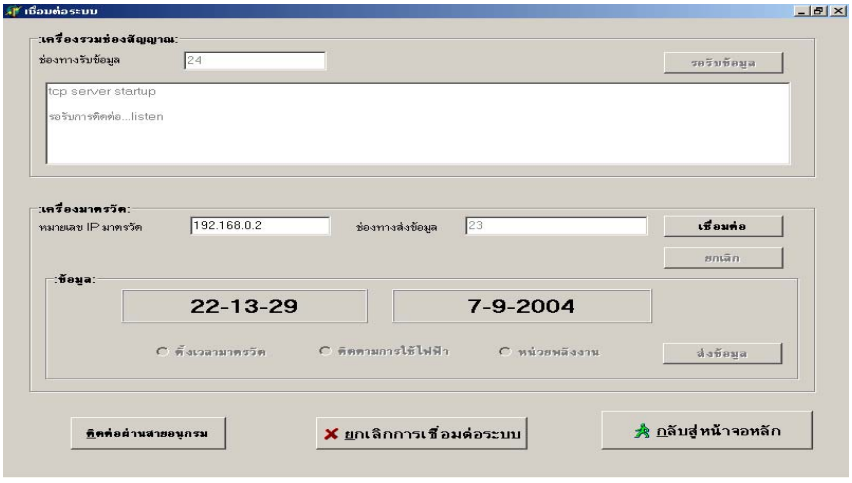
ยูนิตติดตามการทำงานของมาตรวัดมีหน้าจอดังแสดงในรูปที่ 4.5 ประกอบด้วยกราฟการใช้พลังงานของมาตรวัดที่ต้องการติดตามการทำงาน ซึ่งจะเก็บและแสดงข้อมูลในทุก 15 นาทีเป็นเวลา 24 ชั่วโมง พร้อมทั้งค่าความต้องการการใช้พลังงานสูงสุด ด้านขวาของจอภาพแสดงรายละเอียดบางส่วนของประวัติผู้ใช้ไฟฟ้าและประวัติมาตรวัด ส่วนการคำนวณค่าพลังงานไฟฟ้า มีการคำนวณให้เลือกทั้ง 3 อัตรา แสดงค่าพลังงานไฟฟ้า ณ ภาวะปัจจุบันและย้อนหลังหนึ่งเดือน

ยูนิตเชื่อมต่อระบบมีหน้าจอแสดงดังรูปที่ 4.6 ประกอบด้วยส่วนของการรอรับข้อมูล ซึ่งกำหนดช่องทางในการรับข้อมูลทางด้านซอฟต์แวร์ เป็นช่องที่ 24 ในการรับข้อมูล และแสดงข้อมูลต่างๆ ที่รับได้จากมาตรวัดก่อนนำเข้าไปจัดเก็บเป็นฐานข้อมูล

ส่วนของการเชื่อมต่อ กำหนดช่องในการเชื่อมต่อและส่งข้อมูลทางด้านซอฟต์แวร์ เป็นช่องที่ 23 ในการส่งข้อมูล ก่อนดำเนินการเชื่อมต่อจะต้องกำหนดหมายเลขแอดเดรส IP ของมาตรวัดปลายทางที่ต้องการติดต่อ หากเกิดความผิดพลาดจากการป้อนข้อมูล สามารถยกเลิกการเชื่อมต่อนั้นโดยอาศัยเมนูย่อยยกเลิก



รูปที่ 4.5 หน้าจอติดตามการทำงานของมาตรวัด

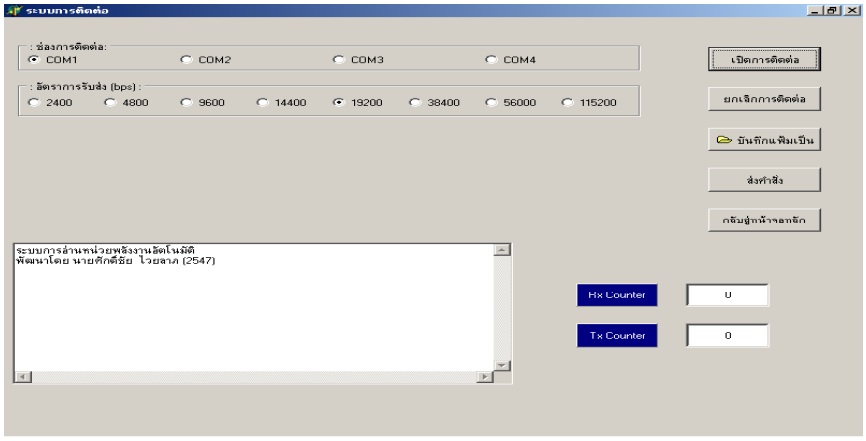


รูปที่ 4.6 หน้าจอเชื่อมต่อระบบ

ส่วนของการส่งข้อมูลที่เป็นลักษณะของคำสั่ง อุปกรณ์รวมช่องสัญญาณจะส่งรหัสส่วนการทำงานให้กับมาตรวัดเพื่อให้มาตรวัดนั้นทำงานตามที่ผู้ใช้งานต้องการ ซึ่งประกอบด้วย 3 คำสั่ง คือ การตั้งเวลามาตรวัด การติดตามการใช้ไฟฟ้า และการส่งหน่วยพลังงาน

นอกจากนี้ยังมีเมนูย่อยที่ใช้สำหรับการติดต่อพอร์ตอนุกรม อาร์เอส 232 เมนูย่อยสำหรับยกเลิกการเชื่อมต่อของระบบทั้งหมดซึ่งโปรแกรมจะทำการปิดช่องทางในการรับและการส่งข้อมูลทั้งหมด และเมนูย่อยสำหรับการกลับเข้าสู่หน้าจอหลักของโปรแกรม

ยูนิตติดต่อผ่านพอร์ตอนุกรมมีหน้าจอแสดงดังรูปที่ 4.7 ประกอบด้วยส่วนของการเลือกช่องการติดต่อ การเลือกอัตราการรับส่งข้อมูล ซึ่งค่าในการเลือกดังกล่าวจะต้องมีความถูกต้องกับค่าที่กำหนดไว้ที่มาตรวัดพลังงานไฟฟ้าด้วย ซึ่งในการพัฒนาครั้งนี้กำหนดให้ติดต่อผ่านทางช่องการติดต่อที่ 1 อัตราการรับส่งที่ 19,200 บิตต่อวินาที เมื่อเปิดการติดต่อกับมาตรวัดสำเร็จจะบันทึกข้อมูลที่ได้ลงในแฟ้มข้อมูล เพื่อประโยชน์สำหรับการเปรียบเทียบข้อมูลต่อไป



รูปที่ 4.7 หน้าจอติดต่อผ่านพอร์ตอนุกรม

4.3 การทดสอบและผลการทดสอบ

สถานที่ทดสอบที่หนึ่ง บ้านเลขที่ 299/92 หมู่ 9 ต.สีคิ้ว อ.สีคิ้ว จ.นครราชสีมา มีระยะทางในการทดสอบโดยประมาณ 11 เมตร ดังแสดงในรูปที่ 4.8 และ 4.9 มีการเก็บข้อมูลหน่วยพลังงานที่มาตรวัดของการไฟฟ้าส่วนภูมิภาค ที่มาตรวัดของระบบต้นแบบ และข้อมูลหน่วยพลังงานที่รับได้

ของอุปกรณ์รวมช่องสัญญาณ เป็นระยะเวลา 24 ชั่วโมงผลที่ได้ตามแสดงในตารางที่ 4.1 และระยะเวลา 7 วัน ผลดังแสดงในตารางที่ 4.2 นำผลที่ได้จากตารางทั้งสองมาเขียนกราฟเพื่อให้การเปรียบเทียบข้อมูลมีความชัดเจนมากขึ้น ดังรูปที่ 4.12 และ 4.13 ตามลำดับ

สถานที่ทดสอบที่สอง บ้านพักพนักงาน ภายในบริเวณการไฟฟ้าส่วนภูมิภาค ต.สีคิ้ว อ.สีคิ้ว จ.นครราชสีมา มีระยะทางในการทดสอบโดยประมาณ 180 เมตร ดังแสดงในรูปที่ 4.10 และ 4.11 ได้ดำเนินการเก็บข้อมูลและเขียนกราฟในลักษณะเดียวกันกับสถานที่ทดสอบที่หนึ่ง ดังแสดงผลในตารางที่ 4.3 และตารางที่ 4.4 กราฟเปรียบเทียบแสดงในรูปที่ 4.14 และรูปที่ 4.15 ตามลำดับ

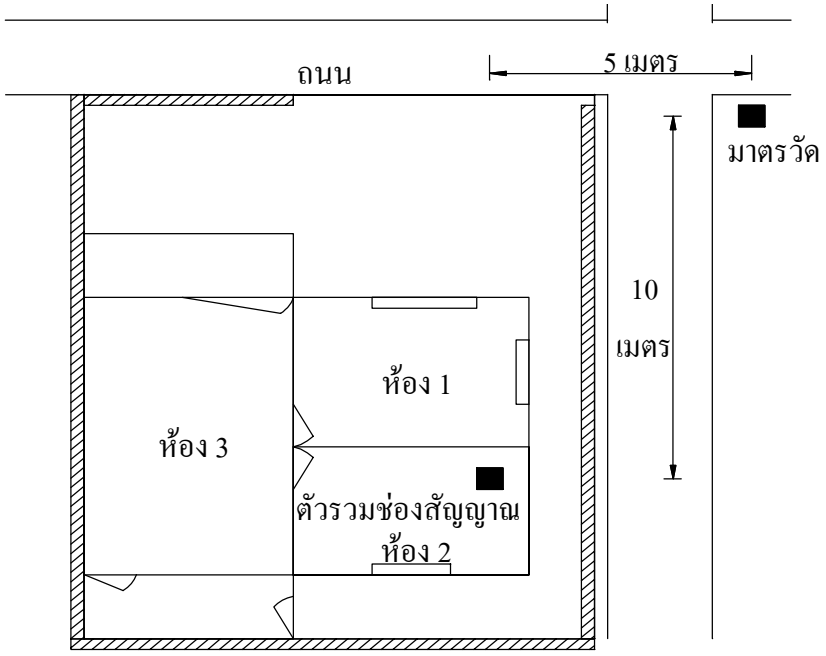
ผลการทดสอบจุดติดตั้งที่หนึ่งและจุดที่สองเป็นระยะเวลา 1 เดือน จำนวนหน่วยของมาตรวัดระบบต้นแบบกับข้อมูลหน่วยที่ได้จากตัวรวมช่องสัญญาณมีค่าเท่ากันคือ จุดติดตั้งที่หนึ่งมีจำนวน 287 หน่วย และจุดติดตั้งที่สองมีจำนวน 43 หน่วย

4.4 วิเคราะห์และอภิปรายผล

ในการวิเคราะห์และอภิปรายผลการทดสอบนั้นอาศัยข้อมูลหน่วยพลังงานที่เก็บจากมาตรวัดกิโลวัตต์ – ชั่วโมง แบบเหนี่ยวนำ ระบบหนึ่งเฟส ของการไฟฟ้าส่วนภูมิภาคเปรียบเทียบความถูกต้องของข้อมูลกับระบบต้นแบบที่ใช้ในงานพัฒนา เพื่อคำนวณหาเปอร์เซ็นต์ความผิดพลาดของระบบ

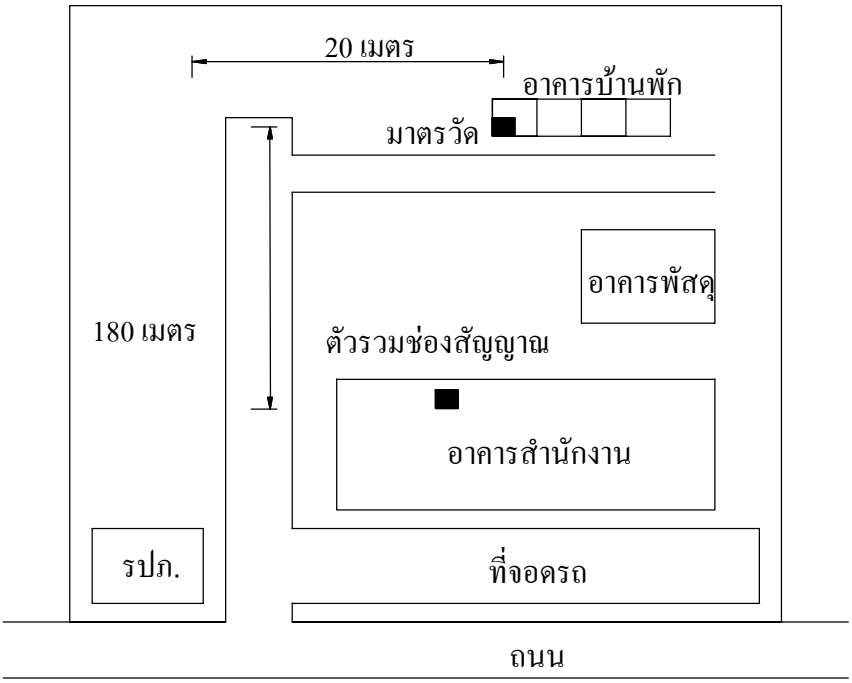
การทดสอบส่งหน่วยพลังงานในรอบเดือนได้ทำการตั้งค่ารอบการหมุนที่ไมโครคอนโทรลเลอร์เป็น 1200 รอบ เท่ากับจำนวนรอบที่ป้ายแสดงรายละเอียดของมาตรวัดกำหนด แต่ในการติดตามการใช้พลังงานในรอบวันนั้นได้มีการปรับตั้งค่ารอบการหมุนที่ไมโครคอนโทรลเลอร์ใหม่ จากจำนวนรอบการหมุนของมาตรวัด ซึ่งต้องหมุนถึง 1200 รอบจึงจะได้จำนวนหน่วยการใช้พลังงานไฟฟ้าครบหนึ่งหน่วยกิโลวัตต์-ชั่วโมง ทำให้การติดตามผลของระบบต้นแบบทำได้ยาก หรือทำไม่ได้ในบางชั่วโมงที่ติดตามผล เนื่องจากปริมาณการใช้พลังงานน้อยมากในบางช่วงเวลา จึงทำการปรับตั้งค่าการนับจำนวนรอบการหมุนที่ไมโครคอนโทรลเลอร์เป็น 12 รอบ ให้ทำการส่งข้อมูลหนึ่งครั้งเพื่อให้การเก็บข้อมูลของระบบต้นแบบทำได้ละเอียดขึ้น เป็นผลทำให้ข้อมูลที่ได้จากอุปกรณ์รวมช่องสัญญาณต้องหารด้วย 100 อีกครั้ง

มาตรวัดพลังงานไฟฟ้าหมุน 1200 รอบได้จำนวนหน่วย 1 หน่วยกิโลวัตต์-ชั่วโมง ไมโครคอนโทรลเลอร์นับรอบได้ 12 รอบ ได้จำนวนหน่วย 0.01 หน่วยกิโลวัตต์-ชั่วโมง



รูปที่ 4.8 จุดติดตั้งทดสอบระบบจุดที่หนึ่ง

รูปที่ 4.9 การติดตั้งระบบจุดที่หนึ่ง



รูปที่ 4.10 จุดติดตั้งทดสอบระบบจุดที่สอง

รูปที่ 4.11 การติดตั้งระบบจุดที่สอง

ตารางที่ 4.1 ผลการทดสอบจุดติดตั้งที่หนึ่ง ระยะเวลา 24 ชั่วโมง

เวลา		00.00 น.	01.00 น.	02.00 น.	03.00 น.	04.00 น.	05.00 น.	06.00 น.	07.00 น.	08.00 น.	09.00 น.	10.00 น.	11.00 น.
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	8304.17	8304.65	8305.05	8305.47	8305.87	8306.35	8306.90	8307.35	8307.55	8307.80	8308.25	8308.70
	จำนวนหน่วย	0.97	0.48	0.40	0.42	0.40	0.48	0.55	0.45	0.20	0.25	0.45	0.45
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0009.12	0009.60	0010.00	0010.42	0010.82	0011.30	0011.85	0012.30	0012.50	0012.75	0013.20	0013.65
	จำนวนหน่วย	0.97	0.48	0.40	0.42	0.40	0.48	0.55	0.45	0.20	0.25	0.45	0.45
ข้อมูลตัวรวมช่องสัญญาณ		0.97	0.48	0.40	0.42	0.40	0.47	0.55	0.45	0.20	0.25	0.45	0.45
เวลา		12.00 น.	13.00 น.	14.00 น.	15.00 น.	16.00 น.	17.00 น.	18.00 น.	19.00 น.	20.00 น.	21.00 น.	22.00 น.	23.00 น.
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	8308.95	8309.25	8309.50	8309.75	8309.90	8310.23	8298.40	8298.75	8299.30	8300.80	8302.00	8303.20
	จำนวนหน่วย	0.25	0.30	0.25	0.25	0.15	0.33	0.42	0.35	0.55	1.50	1.20	1.20
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0013.90	0014.20	0014.45	0014.70	0014.85	0015.18	0003.35	0003.70	0004.25	0005.75	0006.95	0008.15
	จำนวนหน่วย	0.25	0.30	0.25	0.25	0.15	0.33	0.42	0.35	0.55	1.50	1.20	1.20
ข้อมูลตัวรวมช่องสัญญาณ		0.25	0.30	0.25	0.25	0.15	0.33	0.42	0.33	0.54	1.50	1.20	1.20

ตารางที่ 4.2 ผลการทดสอบจุดติดตั้งที่หนึ่ง ระยะเวลา 7 วัน

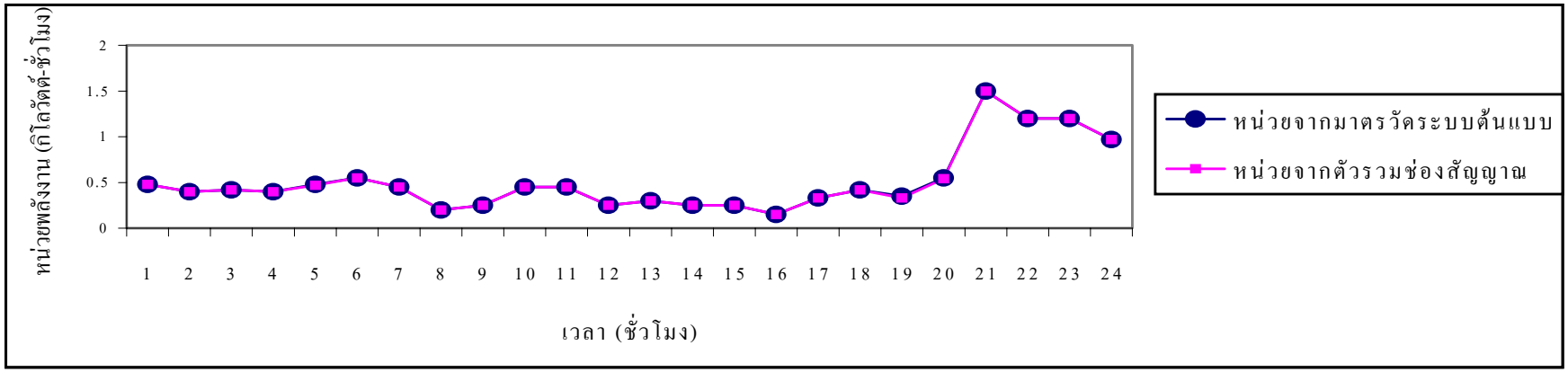
วันที่			1	2	3	4	5	6	7
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	8298.40	8310.65	8319.50	8331.35	8340.85	8351.7	8362.7	8370.65
	จำนวนหน่วย		12.25	8.85	11.85	9.50	10.75	11.10	7.95
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0003.35	0015.60	0024.45	0036.30	0045.80	0056.55	0067.65	0075.60
	จำนวนหน่วย		12.25	8.85	11.85	9.50	10.75	11.10	7.95
ข้อมูลตัวรวมช่องสัญญาณ			12.21	8.85	11.84	9.48	10.71	11.09	7.95

ตารางที่ 4.3 ผลการทดสอบจุดติดตั้งที่สอง ระยะเวลา 24 ชั่วโมง

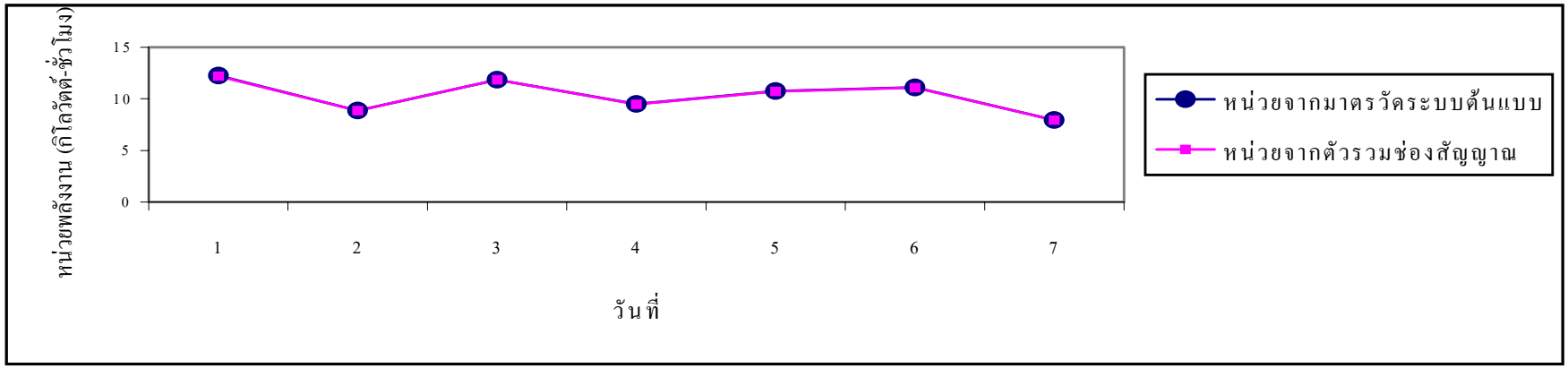
เวลา		00.00 น.	01.00 น.	02.00 น.	03.00 น.	04.00 น.	05.00 น.	06.00 น.	07.00 น.	08.00 น.	09.00 น.	10.00 น.	11.00 น.
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	7350.27	7350.33	7350.37	7350.42	7350.47	7350.53	7350.59	7350.66	7350.72	7350.80	7350.86	7350.91
	จำนวนหน่วย	0.05	0.06	0.07	0.05	0.05	0.06	0.06	0.07	0.06	0.08	0.06	0.05
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0086.47	0086.53	0086.57	0086.62	0086.67	0086.73	0086.79	0086.86	0086.92	0087.00	0087.06	0087.11
	จำนวนหน่วย	0.05	0.06	0.04	0.05	0.05	0.06	0.06	0.07	0.06	0.08	0.06	0.05
ข้อมูลตัวรวมช่องสัญญาณ		0.05	0.06	0.04	0.05	0.05	0.06	0.06	0.07	0.06	0.08	0.06	0.05
เวลา		**12.00 น.**	**13.00 น.**	**14.00 น.**	**15.00 น.**	**16.00 น.**	**17.00 น.**	**18.00 น.**	**19.00 น.**	**20.00 น.**	**21.00 น.**	**22.00 น.**	**23.00 น.**
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	7350.97	7351.03	7351.11	7351.15	7351.20	7356.26	7349.86	7349.92	7350.00	7350.06	7350.16	7350.22
	จำนวนหน่วย	0.06	0.06	0.08	0.04	0.05	0.06	0.10	0.06	0.08	0.06	0.10	0.06
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0087.17	0087.23	0087.31	0087.35	0087.40	0087.46	0086.06	0086.12	0086.20	0086.26	0086.36	0086.42
	จำนวนหน่วย	0.06	0.06	0.08	0.04	0.05	0.06	0.10	0.06	0.08	0.06	0.10	0.06
ข้อมูลตัวรวมช่องสัญญาณ		0.06	0.06	0.08	0.04	0.05	0.06	0.09	0.06	0.08	0.06	0.10	0.06

ตารางที่ 4.4 ผลการทดสอบจุดติดตั้งที่สอง ระยะเวลา 7 วัน

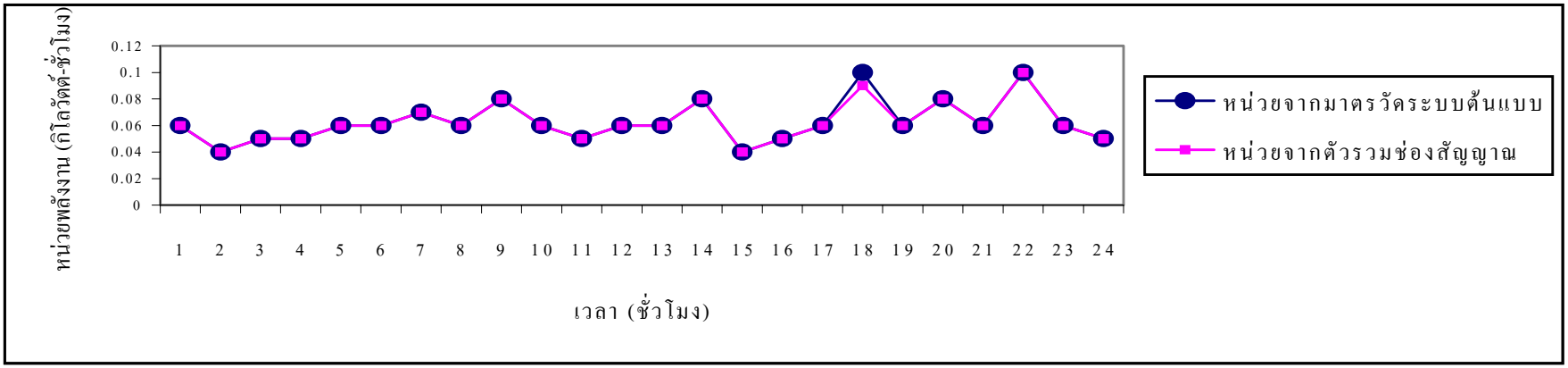
วันที่			1	2	3	4	5	6	7
มาตรวัดการไฟฟ้า	หมายเลขหน่วย	7349.86	7351.36	7352.78	7354.36	7355.89	7358.53	7360.14	7361.63
	จำนวนหน่วย		1.50	1.42	1.58	1.53	2.64	1.61	1.49
มาตรวัดระบบต้นแบบ	หมายเลขหน่วย	0086.12	0087.56	0088.98	0090.56	0092.09	0094.73	0096.34	0097.83
	จำนวนหน่วย		1.50	1.42	1.58	1.53	2.64	1.61	1.49
ข้อมูลตัวรวมช่องสัญญาณ			1.49	1.42	1.58	1.51	2.60	1.60	1.49



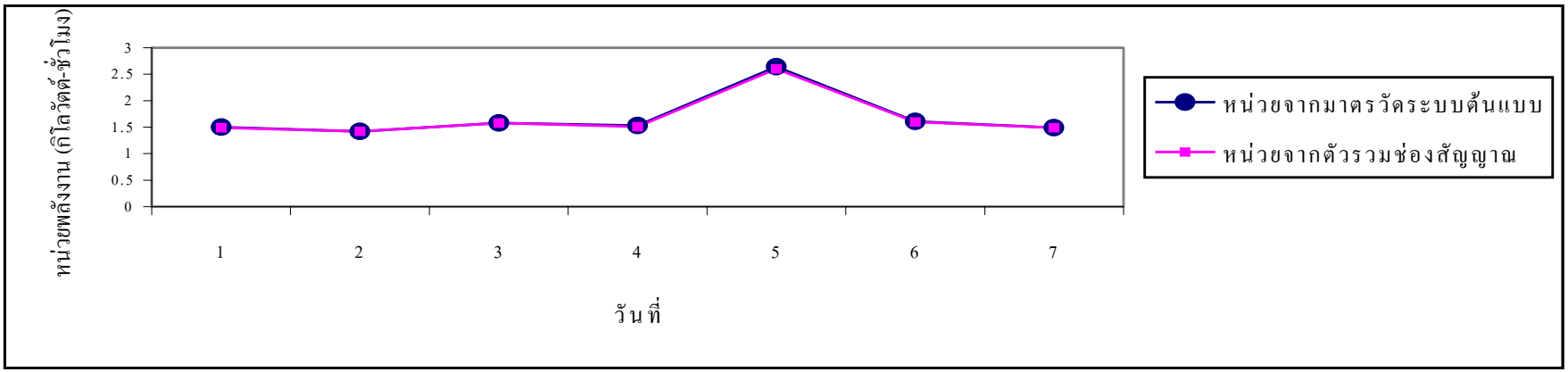
รูปที่ 4.12 กราฟเปรียบเทียบข้อมูลจุดที่หนึ่ง เป็นเวลา 24 ชั่วโมง



รูปที่ 4.13 กราฟเปรียบเทียบข้อมูลจุดที่หนึ่ง เป็นเวลา 7 วัน



รูปที่ 4.14 กราฟเปรียบเทียบข้อมูลจุดที่หนึ่ง เป็นเวลา 24 ชั่วโมง



รูปที่ 4.15 กราฟเปรียบเทียบข้อมูลจุดที่หนึ่ง เป็นเวลา 7 วัน

4.4.1 ความผิดพลาดของหน่วยพลังงานระหว่าง มาตรวัดของการไฟฟ้ากับมาตรวัดของระบบต้นแบบ

$$\%\text{ความผิดพลาด} = \frac{\text{หน่วยรวมมาตรวัดการไฟฟ้า} - \text{หน่วยรวมมาตรวัดระบบต้นแบบ}}{\text{หน่วยรวมมาตรวัดการไฟฟ้า}} \text{ X } 100 \quad (4.1)$$

จากการเก็บผลการทดสอบของจุดติดตั้งทั้งสองแห่งเป็นระยะเวลา 24 ชั่วโมง 7 วัน และ 1 เดือน จำนวนหน่วยของมาตรวัดการไฟฟ้าส่วนภูมิภาคกับจำนวนหน่วยมาตรวัดระบบต้นแบบมีข้อมูลที่ตรงกัน ซึ่งสามารถสรุปได้ว่ามาตรวัดที่ใช้ในระบบต้นแบบมีความถูกต้องของข้อมูลเช่นเดียวกับมาตรวัดของการไฟฟ้า เปอร์เซ็นต์ความผิดพลาดของข้อมูลเป็นศูนย์

4.4.2 ความผิดพลาดของการรับส่งหน่วยพลังงานผ่านระบบเครือข่ายแลนแบบไร้สายระหว่างมาตรวัดของระบบต้นแบบ กับอุปกรณ์รวมช่องสัญญาณ

การพัฒนาในครั้งนี้ได้บันทึกข้อมูลหน่วยพลังงานไฟฟ้าที่ใช้จริงจากมาตรวัดและหน่วยพลังงานไฟฟ้าที่ได้จากระบบต้นแบบลงในตารางที่ 4.1, 4.2, 4.3, และ 4.4 แล้วนำผลรวมของข้อมูลที่ได้มาหาเปอร์เซ็นต์ความผิดพลาด โดยใช้สมการที่ 4.2 ซึ่งมีรายละเอียดในการคำนวณ ดังนี้

$$\%\text{ความผิดพลาด} = \frac{\text{หน่วยรวมของมาตรวัด} - \text{หน่วยรวมอุปกรณ์รวมช่องสัญญาณ}}{\text{หน่วยรวมมาตรวัด}} \text{ X } 100 \quad (4.2)$$

สถานที่ทดสอบที่หนึ่ง

ความผิดพลาดเมื่อทำการทดสอบ 24 ชั่วโมง

$$\%\text{ความผิดพลาด} = \frac{12.25 - 12.21}{12.25} \text{X } 100 = 0.33$$

ความผิดพลาดเมื่อทดสอบ 7 วัน

$$\%\text{ความผิดพลาด} = \frac{72.25 - 72.13}{72.25} \text{X } 100 = 0.17$$

สถานที่ทดสอบที่สอง

ความผิดพลาดเมื่อทำการทดสอบ 24 ชั่วโมง

$$\%\text{ความผิดพลาด} = \frac{1.50 - 1.49}{1.50} \text{ X } 100 = 0.67$$

ความผิดพลาดเมื่อทดสอบ 7 วัน

$$\%\text{ความผิดพลาด} = \frac{11.77 - 11.69}{11.77} \text{X } 100 = 0.68$$

นอกจากนี้ยังสามารถนำข้อมูลหน่วยพลังงานที่จัดเก็บในหน่วยความจำของไมโครคอนโทรลเลอร์ มาเปรียบเทียบกับข้อมูลที่บันทึกไว้ในอุปกรณ์รวมช่องสัญญาณ ได้ว่าระบบรับส่งข้อมูลเกิดความผิดพลาดขึ้นหรือไม่ ดังแสดงในรูปที่ 4.16 ซึ่งด้านซ้ายเป็นข้อมูลที่ได้จากอุปกรณ์รวมช่องสัญญาณที่รับส่งข้อมูลผ่านระบบแลนแบบไร้สาย ส่วนด้านขวาเป็นข้อมูลที่ได้จากหน่วยความจำของไมโครคอนโทรลเลอร์ที่รับได้ผ่านทางพอร์ตอนุกรม อาร์เอส 232

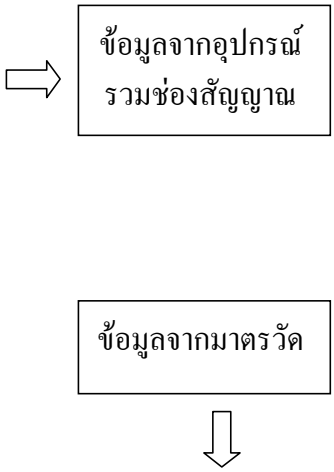
ความผิดผลาดที่เกิดขึ้นในระบบเนื่องมาจากการส่งข้อมูลให้อุปกรณ์รวมช่องสัญญาณไม่สำเร็จ เพราะมีการกำหนดเวลาหน่วงซึ่งเกิดจากการทำงานของโพรโตคอล CSMA/CD หากเวลาที่ใช้ในการเชื่อมต่อระบบนานเกินไป โพรโตคอลจะหยุดการติดต่อเพื่อลดความแออัดของช่องสัญญาณ

การแก้ไขสามารถทำได้โดยการเพิ่มจำนวนการนับรอบของไมโครคอนโทรลเลอร์ให้เป็น 1200 รอบ ซึ่งจะทำให้ความถี่ในการติดต่อกับอุปกรณ์รวมช่องสัญญาณน้อยลง สามารถเพิ่มวงจรวนซ้ำหรือจำนวนรอบที่ใช้ในการติดต่อได้มากขึ้น

ผลการทดสอบและเปรียบเทียบผลที่ได้รับเมื่อกำหนดให้ส่งหน่วยพลังงานในรอบเดือนระหว่างหน่วยพลังงานที่มาตรวัดของระบบต้นแบบกับหน่วยพลังงานที่ตัวรวมช่องสัญญาณปรากฏว่าข้อมูลหน่วยพลังงานมีความถูกต้อง เปอร์เซ็นต์ความผิดพลาดเป็นศูนย์

การเปลี่ยนแปลงเสาสัญญาณของอุปกรณ์แลนแบบไร้สายให้มีค่าอัตราการขยายเพิ่มขึ้นจะทำให้มีสัญญาณที่ดีขึ้น เป็นการเพิ่มระยะทางในการติดต่อและลดความผิดผลาดที่เกิดจากการรับส่งข้อมูลไม่สำเร็จลงได้

time_hour	time_min	time_sec	energyuni
12	5	11	1
12	5	57	1
12	6	42	1
12	7	27	1
12	8	12	1
12	8	58	1
12	9	44	1
12	10	29	1
12	11	15	1
12	12	1	1
12	12	46	1
12	14	18	1
12	15	3	1
12	15	49	1
12	16	34	1
12	17	20	1
12	18	5	1
12	18	50	1
12	19	35	1
12	20	19	1
12	21	4	1
12	21	48	1
12	24	24	1



รูปที่ 4.16 การตรวจสอบการทำงานของระบบรับส่งข้อมูล

4.5 บทสรุป

จากการทดสอบการทำงานของอุปกรณ์และโปรแกรมที่พัฒนาขึ้นในระบบต้นแบบเพื่อใช้ในการรับส่งข้อมูลหน่วยพลังงานไฟฟ้า ระบบสามารถติดตามการใช้พลังงานไฟฟ้าในแต่ละชั่วโมงในรอบวันได้มากกว่า 99 เปอร์เซ็นต์ สามารถส่งหน่วยพลังงานและบันทึกข้อมูลการใช้หน่วยพลังงานในรอบเดือนได้เองอัตโนมัติและมีความถูกต้อง ทำให้นำหน่วยพลังงานนั้นมาคำนวณค่าไฟฟ้าได้ตามอัตราที่การไฟฟ้าส่วนภูมิภาคกำหนด

บทที่ 5
สรุปและเสนอแนะ

5.1 สรุปผลการพัฒนา

หลักการสื่อสารที่ใช้ในการรับส่งข้อมูลหน่วยพลังงานไฟฟ้าในงานพัฒนานี้อาศัยเทคโนโลยีของระบบแลนแบบไร้สายที่มีการมอดูเลตแบบแบ่งรหัสลำดับโดยตรง ซึ่งเป็นการเข้าใช้ช่องสัญญาณแบบหลายทางอีกรูปแบบหนึ่ง สัญญาณข้อมูลแต่ละช่องจะถูกคูณด้วยรหัสที่มีความแตกต่างกันแล้วนำไปมอดูเลตกับคลื่นพาห์ที่มีแถบความถี่อยู่ในย่าน 2.4 GHz นอกจากนี้ยังมีระบบการควบคุมผู้เข้าใช้เครือข่ายอีกระดับหนึ่งเพื่อความปลอดภัยของข้อมูล รัศมีของพื้นที่การรับส่งข้อมูลจะแตกต่างกันออกไปขึ้นอยู่กับสภาพแวดล้อมของตัวกลางที่มีปัจจัยต่อการลดทอนสัญญาณ หลากมีความผิดพลาดของข้อมูลเกิดขึ้น ระบบแลนไร้สายจะทำการส่งข้อมูลซ้ำอีกในอัตราการส่งที่ต่ำลงมาอีก

การพัฒนานี้ได้อาศัยโพรโตคอล TCP/IP ในการขนถ่ายข้อมูล โดยกำหนดหมายเลข IP แอดเดรสเป็นชั้นซีและเป็นแบบเครือข่ายส่วนบุคคลที่ไม่สามารถติดต่อจากระบบเครือข่ายภายนอกได้ สามารถรองรับจำนวนมาตรวัดพลังงานไฟฟ้าได้สูงสุดถึง 254 เครื่อง โพรโตคอล MAC ที่สนับสนุนการเข้าใช้ตัวกลางร่วมกันแต่แบ่งเวลาในการใช้ด้วยวิธีการเข้าถึงแบบการสุ่มทำให้มาตรวัดพลังงานไฟฟ้าแต่ละเครื่องมีโอกาสที่จะใช้ตัวกลางเวลาใดก็ได้

การติดต่อกับมาตรวัดที่ตัวรวมช่องสัญญาณจะอาศัยโปรแกรมที่พัฒนาด้วยโปรแกรมเดลไฟ มีมอดูลฐานข้อมูลเป็นแบบไคลเอนต์/เซิร์ฟเวอร์ ผ่านการเชื่อมโยง ODBC ในระบบปฏิบัติการวินโดวส์เข้าสู่แหล่งข้อมูลที่จัดเก็บโดยโปรแกรมแอคเซส 97 ทางด้านมาตรวัดพลังงานอาศัยโปรแกรมที่พัฒนาด้วยโปรแกรมไดนามิค ซี 7.06 บนบอร์ดไมโครคอนโทรลเลอร์แรบบิท 2000 ให้ทำการรับส่งข้อมูลและเป็นฐานเวลาให้กับมาตรวัด เพื่อใช้ในการแยกหน่วยการใช้พลังงานไฟฟ้าตามช่วงเวลาที่กำหนด

การทดสอบระบบ โดยการจดหน่วยพลังงานไฟฟ้าที่มาตรวัดตามเวลาและวันเดือนปีที่กำหนดเทียบกับข้อมูลที่ได้รับจากตัวรวมช่องสัญญาณที่เงื่อนไขติดตามการใช้พลังงานในรอบวันเป็นระยะเวลา 7 วันมีความเชื่อถือได้กว่าร้อยเปอร์เซ็นต์ ที่เงื่อนไขส่งหน่วยพลังงานในรอบเดือนเป็นระยะเวลา 1 เดือนสามารถส่งข้อมูลหน่วยพลังงานได้อย่างถูกต้อง

5.1 การประยุกต์ผลการพัฒนา

- ประยุกต์ใช้ระบบต้นแบบ ในระบบงานอ่านหน่วยมาตรวัดพลังงานไฟฟ้าอัตโนมัติในรอบเดือน

- ประยุกต์ใช้ระบบต้นแบบ เป็นระบบติดตามการทำงาน หรือเก็บบันทึกข้อมูลต่างๆ ของเครื่องจักรในระบบอุตสาหกรรม

5.2 ข้อเสนอแนะในการพัฒนาต่อไป

ควรเลือกใช้ไมโครคอนโทรลเลอร์ที่มีส่วนขยายเอาต์พุตเป็นพอร์ตสายรวมเอนกประสงค์ (universal bus) เพื่อให้ส่วนของระบบแลนแบบไร้สายที่ใช้ในการรับส่งข้อมูลมีราคาถูกลง นอกจากนี้ยังทำให้เกิดความสะดวกในการติดตั้ง หรือการใช้งาน เนื่องจากไม่มีการใช้สายแลนและขั้วเสียบมาติดตั้งในระบบ

ควรลดขนาดของอุปกรณ์รวมช่องสัญญาณให้มีขนาดเล็กลงซึ่งในงานพัฒนาชิ้นนี้ใช้ ไมโครคอมพิวเตอร์โดยอาจจะเปลี่ยนเป็นไมโครคอนโทรลเลอร์หรืออุปกรณ์อื่นที่มีฟังก์ชันในการประยุกต์ใช้งานใกล้เคียงกัน

การเพิ่มจำนวนจุดเข้าสัญญาณ การติดตั้งรีพีตเตอร์หรือการเปลี่ยนแปลงเสาสัญญาณให้มีค่าอัตราการขยายสูงขึ้นจะทำให้มีสัญญาณที่ดีขึ้น เป็นการเพิ่มหรือขยายพื้นที่ในการใช้งานและลดความผิดพลาดที่อาจเกิดขึ้นกับการรับส่งข้อมูลลงได้

เอกสารอ้างอิง

การไฟฟ้าส่วนภูมิภาค. (2543). **อัตราค่าไฟฟ้า**. กรุงเทพฯ: การไฟฟ้าส่วนภูมิภาค.

ฉัตรชัย สุมามาลย์. (2542). **การสื่อสารข้อมูลคอมพิวเตอร์และระบบเครือข่าย**. กรุงเทพฯ: IT Book.

ประพนธ์ อัศวภาณุวัฒน์. (2543). **Delphi Episode II เทคนิคและการพัฒนาโปรแกรมด้วยเดลไฟ**. กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

ประสิทธิ์ ประพินมงคลการ. (2536). **หลักการสื่อสาร**. กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

ไพศาล สงวนหมู่ และ ยืน ภู่วรวรรณ. (2536). **การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค** . กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

ลัญฉกร วุฒิสิทธิกุลกิจ. (2540). **หลักการระบบโทรศัพท์เคลื่อนที่**. กรุงเทพฯ: จุฬาลงกรณ์มหาวิทยาลัย

สุวัฒน์ ปุณณชัยยะ, สุพจน์ ปุณณชัยยะ และ ต้น ตัณฑ์สุทธิวงศ์. (2543). **เปิดโลกของ TCP/IP และโพรโตคอลของอินเตอร์เน็ต**. กรุงเทพฯ: โปรวิชั่น.

อภิชาต อัศวาดิศยางกูร. (2539). **ทฤษฎีและการใช้ระบบเครือข่าย**. กรุงเทพฯ: ฟิสิกส์เซ็นเตอร์

Artecch, R.P., (1996). **CDMA for wireless personal communication**. New York: House Publishers.

Fairhurst, G., (2002). **Carrier Sense Multiple Access with Collision Detection (CSMA/CD)**. [On-line]. Available : http://www.erg.abdu.ac.uk./user/gorry/course/lanpages/camacd.html.

Ghajar, R., Khalife, J., and Richani, B., (2000). Design and cost analysis of an automatic meter reading system for Electricite' du Liban. **Utilities Policy**. 9:193-205.

Hooijen, O. G., (1998). A channel Model for the Residential Power circuit Used as a Digital Communication Medium. **IEEE Transactions on Electromagnetic Compatibility**. 40(4): 331-336.

Parker, T., (1994), **Teach yourself TCP/IP in 14 days**. New York: Sams Publishing.

Post and Telegraph Department. (2002). **Spectrum Utilization 2.3-2.7 GHz**.. [On-line]. Available : htth://www.ptd.go.th/frequency/im_75.pdf.

Scott, R., (1999). **Understanding Cyclic Redundancy Check**. [On-line]. Available : http://www.4d.com/ACIDOC/CMU/CMU79909. html.

ภาคผนวก ก.

โปรแกรมระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ

โปรแกรมระบบงานอ่านหน่วยมาตรวัดอัติโนมัติ

(ผู้พัฒนา นายศักดิ์ชัย ไวยลาภ, 2546)

ยูนิตระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ

```
unit amrmain;
interface
uses
     Windows, Messages, SysUtils, Classes,
     Graphics, Controls, Forms, Dialogs,
     ExtCtrls, Gauges, StdCtrls, Buttons,
     TeeProcs, TeEngine, Chart, DBChart,
     Db, DBTables, TeeFunci, Comctrls, Series;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

```
type
     Tfamrmain = class(TForm)
     Panel1: TPanel;
     Panel2: TPanel;
     Bevel1: TBevel;
     Timer1: TTimer;
     BitBtn1: TBitBtn;
     button1: TButton;
     Button2: TButton;
     Button3: TButton;
     Button4: TButton;
     Button5: TButton;
     DBChart1: TDBChart;
     Series1: TBarSeries;
```

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
procedure Timer1Timer(Sender: TObject);
```
โพรซีเดอร์ระบบเวลาที่ยูนิตใช้แสดง

```
procedure FormCreate(Sender: TObject);
```
โพรซีเดอร์เก็บค่าตัวแปรช่วยสำหรับตรวจสอบการสร้างหน้าจอย่อย

```
procedure BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์ออกจากระบบ

```
procedure button1Click(Sender: TObject);
```
โพรซีเดอร์แสดงหน้าจอประวัติผู้ใช้ไฟฟ้า

```
procedure Button2Click(Sender: TObject);
```
โพรซีเดอร์แสดงหน้าจอประวัติมาตรวัด

```
procedure Button3Click(Sender: TObject);        โพรซีเดอร์แสดงหน้าจออัตราค่าไฟฟ้า
procedure Button4Click(Sender: TObject);        โพรซีเดอร์แสดงหน้าจอติดตามการใช้ไฟฟ้า
procedure Button5Click(Sender: TObject);        โพรซีเดอร์แสดงหน้าจอเชื่อมต่อระบบ
private
   { Private declarations }
public
   { Public declarations }
procedure inc_row_total;                        โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานรวม
procedure inc_row_unit1;                        โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานช่วง 09.00 – 21.59 น.
procedure inc_row_unit2;                        โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานช่วง 22.00 – 08.59 น.
procedure create_table_h(cust_no_s:string);     โพรซีเดอร์สร้างตารางเก็บข้อมูลหน่วยพลังงานในรอบวัน
procedure append_meter;                         โพรซีเดอร์เพิ่มประวัติมาตรวัด
procedure edit_meter;                           โพรซีเดอร์แก้ไขประวัติมาตรวัด
procedure append_customer;                      โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า
procedure edit_customer;                        โพรซีเดอร์แก้ไขประวัติผู้ใช้ไฟฟ้า
procedure append_mount;                         โพรซีเดอร์เพิ่มจำนวนข้อมูลเดือน
end;
var
        famrmain: Tfamrmain;                    ตัวแปรของหน้าจอระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ
implementation
uses netamr, meter_module,meter, user, mon, costeng, helpamr,centralvar;
                                                เรียกใช้ยูนิต netamr, meter_module, meter, user, mon, costeng, helpamr และ centralvar
{$R *.DFM}
```

```
procedure tfamrmain.inc_row_total;          โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานรวม
begin
        meter_datamodule.mount_Query.append;
                                            เพิ่มจำนวนข้อมูล
        meter_datamodule.mount_Query.FieldValues['mount_1']:=0;
                                            ข้อมูลที่เพิ่มเป็นศูนย์
        meter_datamodule.mount_Query.post;  บันทึกข้อมูลศูนย์ที่เพิ่ม
        meter_datamodule.mount_Query.active:=false;
                                            ยกเลิกการติดต่อกับข้อมูล
        meter_datamodule.mount_Query.active:=true;
                                            เปิดการติดต่อกับข้อมูล
end;
procedure tfamrmain.inc_row_unit1;          โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานช่วง 09.00 – 21.59 น.
begin
        meter_datamodule.mount1_Query.append;
                                            เพิ่มจำนวนข้อมูล
        meter_datamodule.mount1_Query.FieldValues['mount_1']:=0;
                                            ข้อมูลที่เพิ่มเป็นศูนย์
        meter_datamodule.mount1_Query.post; บันทึกข้อมูลศูนย์ที่เพิ่ม
        meter_datamodule.mount1_Query.active:=false;
                                            ยกเลิกการติดต่อกับข้อมูล
        meter_datamodule.mount1_Query.active:=true;
                                            เปิดการติดต่อกับข้อมูล
end;
procedure tfamrmain.inc_row_unit2;          โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วยพลังงานช่วง 22.00 – 08.59 น.
Begin
         meter_datamodule.mount2_Query.append;
```

```
                                        เพิ่มจำนวนข้อมูล
    meter_datamodule.mount2_Query.FieldValues['mount_1']:=0;
                                        ข้อมูลที่เพิ่มเป็นศูนย์
    meter_datamodule.mount2_Query.post; บันทึกข้อมูลศูนย์ที่เพิ่ม
    meter_datamodule.mount2_Query.active:=false;
                                        ยกเลิกการติดต่อกับข้อมูล
    meter_datamodule.mount2_Query.active:=true;
                                        เปิดการติดต่อกับข้อมูล
end;
procedure tfamrmain.create_table_h(cust_no_s:string);
                                        โพรซีเดอร์สร้างตารางเก็บข้อมูลหน่วยพลังงานใน
                                        รอบวัน
begin
    if not meter_datamodule.table1.exists then
                                        ถ้ายังไม่มีตารางในมอดูลข้อมูลให้สร้างตาราง
    begin
        with meter_datamodule.table1 do
                                        อ้างถึงตารางที่สร้างขึ้นเพื่อกำหนดคุณสมบัติ
        begin
            active:=false;              ยกเลิกการติดต่อกับตารางที่สร้าง
            databasename:='meter_db';
                                        กำหนดชื่อของฐานข้อมูล
            tabletype:=ttparadox;       กำหนดประเภทของข้อมูล
            tablename:='t_cust_h'+cust_no_s;
                                        กำหนดชื่อให้ตารางเป็น “t_cust_h หมายเลขผู้ใช้
                                        ไฟฟ้า”
            with fielddefs do
            begin
                clear;
                with addfielddef do
```

```
                                        เพิ่มขอบเขตข้อมูล
                                begin
                                        name:='time_read';
                                        กำหนดชื่อขอบเขตข้อมูลชื่อ “time_read”
                                        datatype:=ftinteger;
                                        กำหนดประเภทของข้อมูล
                                        required:=true;
                                        ยืนยันสิ่งที่กำหนด
                                end;
                                with addfielddef do
                                begin
                                        name:='energyunit';
                                        กำหนดชื่อขอบเขตข้อมูลชื่อ “energyunit”
                                        datatype:=ftinteger;
                                        กำหนดประเภทของข้อมูล
                                        required:=true;
                                        ยืนยันสิ่งที่กำหนด
                                end;
                        end;
                    CreateTable;            สร้างตาราง
                end;
        end;
end;
procedure tfamrmain.append_meter;           โพรซีเดอร์เพิ่มประวัติมาตรวัด
begin
        with meter_datamodule.meter_query doอ้างถึงตำแหน่งมอดูลข้อมูลของมาตรวัด
        begin
                append;                     เพิ่มประวัติมาตรวัด
        end;
end;
```

```
procedure tfamrmain.append_customer;        โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า
begin
        with meter_datamodule.customer_query do
                                            อ้างถึงตำแหน่งมอดูลข้อมูลของผู้ใช้ไฟฟ้า
        begin
                append;                     เพิ่มประวัติผู้ใช้ไฟฟ้า
        end;
end;
procedure tfamrmain.append_mount;           โพรซีเดอร์เพิ่มจำนวนข้อมูลเดือน
begin
        with meter_datamodule.mount_query doอ้างถึงตำแหน่งมอดูลข้อมูลของเดือน
        begin
                append;                     เพิ่มจำนวนข้อมูลเดือน
        end;
end;
procedure tfamrmain.edit_meter;             โพรซีเดอร์แก้ไขประวัติมาตรวัด
begin
        with meter_datamodule.meter_query doอ้างถึงตำแหน่งมอดูลข้อมูลของมาตรวัด
        begin
                edit;                       แก้ไขข้อมูลมาตรวัด
        end;
end;
procedure tfamrmain.edit_customer;          โพรซีเดอร์แก้ไขประวัติผู้ใช้ไฟฟ้า
begin
        with meter_datamodule.customer_query do
                                            อ้างถึงตำแหน่งมอดูลข้อมูลของผู้ใช้ไฟฟ้า
        begin
                edit;                       แก้ไขข้อมูลผู้ใช้ไฟฟ้า
        end;
end;
```

```
procedure Tfnetamr.Timer1Timer(Sender: TObject);
                                        โพรซีเดอร์ระบบเวลาที่ยูนิตใช้แสดง
var
        sd:pchar;
        p:pchar;                                    ประกาศตัวแปรในการค้นหาข้อมูล
        buff:array[0..3] of char;
        l:integer;
        s:string;
begin
        timeseparator:='-';             จัดรูปแบบข้อมูล “ชั่วโมง-นาที-วินาที”
        panel1.caption:=timetostr(now); แสดงข้อมูลของเวลา
        dateseparator:='-';             จัดรูปแบบข้อมูล “วัน-เดือน-ปี”
        panel2.caption :=datetostr(date); แสดงข้อมูลของวันเดือนปี
        s:=panel2.caption;
        p:=@s[1];                       พอยเตอร์ชี้ที่ตำแหน่งแรกของข้อมูลวันเดือนปี
        sd:=strscan(p,'-');
        sd:=sd+1;
        strcopy(p,sd);
        sd:=strscan(p,'-');                         ค้นหาข้อมูลจาก “วัน-เดือน-ปี” เพื่อ
                                                    เก็บข้อมูลเฉพาะวันและเดือน
        l:=strlen(p)-strlen(sd);
        strlcopy(buff,p,l);
        date_mserv:=buff;
end;
procedure Tfamrmain.FormCreate(Sender: TObject);
                                        โพรซีเดอร์เก็บค่าตัวแปรช่วยสำหรับตรวจสอบ
                                        การสร้างหน้าจอย่อย
begin
        old_time_m:=0;
        eng_buff:=1;                    กำหนดค่าเริ่มต้นของตัวแปรเมื่อเริ่มเปิดโปรแกรม
```

```
        senserform:=false;
        createflag:=false;
        timer1timer(timer1);                    เรียกใช้โพรซีเดอร์ระบบเวลาที่ยูนิตใช้แสดงขณะ
                                                เปิดโปรแกรม
end;
procedure Tfamrmain.BitBtn1Click(Sender: TObject);
                                                โพรซีเดอร์ออกจากระบบ
begin
         if MessageDlg('คุณต้องการออกจากระบบงาน'),
         mtConfirmation, [mbYes, mbNo], 0) = mrYes then
                                                แสดงข้อความ “คุณต้องการออกจากระบบงาน”
                                                เพื่อรอรับการยืนยัน
famrmain.close;                                 ออกจากระบบงานอ่านหน่วยมาตรวัดอัตโนมัติ
end;
procedure Tfamrmain.button1Click(Sender: TObject);
                                                โพรซีเดอร์แสดงหน้าจอประวัติผู้ใช้ไฟฟ้า
begin
        if  (fuser=nil) then                    ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติผู้ใช้ไฟฟ้า
                                                ให้สร้างหน้าจอขึ้นมาใหม่
                application.createform(tfuser,fuser);
        fuser.show;                             แสดงหน้าจอประวัติผู้ใช้ไฟฟ้า
        famrmain.Hide;                          ซ่อนหน้าจอหลักของโปรแกรม
end;
procedure Tfamrmain.Button2Click(Sender: TObject);
                                                โพรซีเดอร์แสดงหน้าจอประวัติมาตรวัด
begin
        if  (fmeter=nil) then                   ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติมาตรวัด
                                                ให้สร้างหน้าจอขึ้นมาใหม่
                application.createform(tfmeter,fmeter);
        fmeter.show;                            แสดงหน้าจอประวัติมาตรวัด
```

```
        famrmain.Hide;                          ซ่อนหน้าจอหลักของโปรแกรม
end;
procedure Tfamrmain.Button3Click(Sender: TObject);
                                                โพรซีเดอร์แสดงหน้าจออัตราค่าไฟฟ้า
begin
        if (fcosteng=nil) then                  ถ้ายังไม่มีการเรียกใช้งานหน้าจออัตราค่าไฟฟ้าให้
                                                สร้างหน้าจอขึ้นมาใหม่
                application.createform(tfcosteng,fcosteng);
        fcosteng.show;                          แสดงหน้าจออัตราค่าไฟฟ้า
        famrmain.Hide;                          ซ่อนหน้าจอหลักของโปรแกรม
end;
procedure Tfamrmain.Button4Click(Sender: TObject);
                                                โพรซีเดอร์แสดงหน้าจอติดตามการใช้ไฟฟ้า
begin
        if (fmon=nil) then                      ถ้ายังไม่มีการเรียกใช้งานหน้าจอติดตามการใช้ไฟ
                                                ฟ้าให้สร้างหน้าจอขึ้นมาใหม่
                application.createform(tfmon,fmon);
        fmon.show;                              แสดงหน้าจอติดตามการใช้ไฟฟ้า
        famrmain.Hide;                          ซ่อนหน้าจอหลักของโปรแกรม
end;
procedure Tfamrmain.Button5Click(Sender: TObject);
                                                โพรซีเดอร์แสดงหน้าจอเชื่อมต่อระบบ
begin
        if (fnetamr=nil) then                   ถ้ายังไม่มีการเรียกใช้งานหน้าจอเชื่อมต่อระบบ
                                                ให้สร้างหน้าจอขึ้นมาใหม่
                application.createform(tfnetamr,fnetamr);
        fnetamr.show;                           แสดงหน้าจอเชื่อมต่อระบบ
        famrmain.Hide;                          ซ่อนหน้าจอหลักของโปรแกรม
end;
end.
```

ยูนิตมอดูลของฐานข้อมูล

```
unit meter_module;
interface
uses
        Windows, Messages, SysUtils,
        Classes, Graphics, Controls,
        Forms, Dialogs, Db, DBTables;
type
        Tmeter_datamodule = class(TDataModule)
        customer_DataSource: TDataSource;
        meter_DataSource: TDataSource;
        customer_Query: TQuery;
        meter_Query: TQuery;
        mount1_DataSource: TDataSource;
        mount1_Query: TQuery;
        meter_Database: TDatabase;
        cost_DataSource: TDataSource;
        cost_Table: TTable;
        total_unit_Query: TQuery;
        total_unit_DataSource: TDataSource;
        mount2_Query: TQuery;
        mounnt2_DataSource: TDataSource;
        mount_Query: TQuery;
        mount_DataSource: TDataSource;
        Table1: TTable;
        DataSource1: TDataSource;
        joint_cust_query: TQuery;
        joint_cust_DataSource: TDataSource;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่มอดูลใช้ในการติดต่อ

```
procedure meter_datamodulecreate(Sender: TObject);
                                                    โพรซีเดอร์เชื่อมต่อมอดูลฐานข้อมูล
procedure meter_datamoduledestroy(Sender: TObject);
                                                    โพรซีเดอร์หยุดการเชื่อมต่อมอดูลฐานข้อมูล
private
   { Private declarations }
public
   { Public declarations }
end;
var
        meter_datamodule: Tmeter_datamodule;ตัวแปรของมอดูลฐานข้อมูล
implementation
{$R *.DFM}
procedure Tmeter_datamodule.meter_datamodulecreate(Sender: TObject);
                                                    โพรซีเดอร์เชื่อมต่อมอดูลฐานข้อมูล
begin
        meter_database.connected:=true;         เชื่อมต่อมอดูลฐานข้อมูล
end;
procedure Tmeter_datamodule.meter_datamoduledestroy(Sender: TObject);
                                                    โพรซีเดอร์หยุดการเชื่อมต่อมอดูลฐานข้อมูล
begin
        meter_database.Connected:=false;        หยุดการเชื่อมต่อฐานข้อมูล
end;
end.
```

ยูนิตอัตราค่าไฟฟ้า

```
unit costeng;
interface
uses
        Windows, Messages, SysUtils, Classes,
        Graphics,Controls, Forms, Dialogs,
        StdCtrls,Buttons, DBCGrids, DBCtrls,
        Mask, ExtCtrls, Db, DBTables;
type
        Tfcosteng = class(TForm)
        BitBtn1 : TBitBtn;
        GroupBox1 : TGroupBox;
        Label1 : TLabel;
        Panel1 : TPanel;
        GroupBox2 : TGroupBox;
        Memo1 : TMemo;
        Memo2 : TMemo;
        DBText1 : TDBText;
        DBText2 : TDBText;
        DBText3 : TDBText;
        DBText4 : TDBText;
        DBText5 : TDBText;
        DBText6 : TDBText;
        DBText7 : TDBText;
        DBText8 : TDBText;
        Memo3 : TMemo;
        Label2  : TLabel;
        Memo4 : TMemo;
        Memo5 : TMemo;
        DBText9 : TDBText;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
        DBText10 : TDBText;
        DBText11 : TDBText;
        Label3: TLabel;
        Memo6 : TMemo;
        Memo7 : TMemo;
        DBText12 : TDBText;
```

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
procedure BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์กลับสู่หน้าจอหลักของโปรแกรม

```
procedure FormCreate(Sender: TObject);
```
โพรซีเดอร์แสดงหมายเหตุอัตราค่าไฟฟ้า

```
private
public
end;
var
        Fcosteng: TFcosteng;
```
ตัวแปรของหน้าจออัตราค่าไฟฟ้า

```
implementation
uses amrmain,meter_module;
```
เรียกใช้ยูนิต amrmain และ ยูนิต meter_module

```
{$R *.DFM}
procedure TFcosteng.BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์กลับสู่หน้าจอหลักของโปรแกรม

```
begin
        famrmain.show;
```
แสดงหน้าจอหลักของโปรแกรม

```
        fcosteng.Hide;
```
ซ่อนหน้าจออัตราค่าไฟฟ้า

```
end;
procedure TFcosteng.FormCreate(Sender: TObject);
```
โพรซีเดอร์แสดงหมายเหตุอัตราค่าไฟฟ้า

```
begin
        memo1.Lines.LoadFromFile('c:\my documents\comment_rate.txt');
```
แสดงข้อมูลของหมายเหตุอัตราค่าไฟฟ้าจากตำแหน่ง “c:\my documents\comment_rate.txt ”

```
end;
end.
```

ยูนิตประวัติผู้ใช้ไฟฟ้า

```
unit user;
interface
uses
        Windows, Messages, SysUtils,
        Classes,Graphics,Controls, Forms,
        Dialogs,StdCtrls, Buttons, Mask,
        ExtCtrls, DBCtrls, Db, DBTables;
type
        TFuser = class(TForm)
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        Label4: TLabel;
        Label5: TLabel;
        Label6: TLabel;
        Label7: TLabel;
        Label8: TLabel;
        Label9: TLabel;
        Label10: TLabel;
        Bevel1: TBevel;
        Bevel2: TBevel;
        Bevel3: TBevel;
        BitBtn1: TBitBtn;
        Button1: TButton;
        Button2: TButton;
        Button3: TButton;
        Button4: TButton;
        DBEdit1: TDBEdit;
        DBEdit2: TDBEdit;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
        DBEdit4: TDBEdit;
        DBEdit5: TDBEdit;
        DBEdit6: TDBEdit;
        DBEdit7: TDBEdit;
        DBEdit8: TDBEdit;
        DBEdit9: TDBEdit;
        DBEdit10: TDBEdit;
        DBNavigator1: TDBNavigator;
        DBText1: TDBText;
        Label11: TLabel;
```

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
procedure BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์กลับสู่หน้าจอหลัก

```
procedure Button1Click(Sender: TObject);
```
โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า

```
procedure Button2Click(Sender: TObject);
```
โพรซีเดอร์แก้ไขประวัติผู้ใช้ไฟฟ้า

```
procedure Button3Click(Sender: TObject);
```
โพรซีเดอร์รายละเอียดประวัติมาตรวัด

```
procedure Button4Click(Sender: TObject);
```
โพรซีเดอร์ติดตามการใช้ไฟฟ้า

```
private
{ Private declarations }
procedure checkedit(var editflag:boolean);
```
โพรซีเดอร์ตรวจสอบการป้อนข้อมูล

```
public
{ Public declarations }
procedure buttonenableuser(buttonenab:boolean);
```
โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอขณะป้อนข้อมูล

```
procedure editenableuser(editenab:boolean);
```
โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล

```
end;
var
        Fuser: TFuser;
```
ตัวแปรของหน้าจอประวัติผู้ใช้ไฟฟ้า

```
implementation
uses amrmain,centralvar,meter, meter_module;
```
เรียกใช้ยูนิต amrmain,centralvar,meter และ meter_module

```
{$R *.DFM}
```

```
procedure tfuser.checkedit(var editflag:boolean);
                                        โพรซีเดอร์ตรวจสอบการป้อนข้อมูล
begin
    if (dbedit1.text ='') or (dbedit2.text ='') or (dbedit3.text ='') or
    (dbedit4.text ='') or (dbedit5.text ='') or (dbedit6.text ='') or
    (dbedit7.text ='') or (dbedit8.text ='') or (dbedit9.text ='') or
    (dbedit10.text='')                  ตรวจสอบการป้อนข้อมูลที่ช่องรอรับข้อมูล ถ้ามี
                                        การป้อนข้อมูลไม่ครบตามเงื่อนไขที่กำหนด ค่า
                                        ของตัวแปร editflag จะเป็นเท็จ
    then
        editflag:=false
    else
        editflag:=true;
end;
procedure tfuser.editenableuser(editenab:boolean);
                                        โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล
begin
    dbedit1.enabled:=editenab;
    dbedit2.enabled:=editenab;
    dbedit3.enabled:=editenab;
    dbedit4.enabled:=editenab;
    dbedit5.enabled:=editenab;          เคลียร์และปรับสภาวะของช่องรับ
    dbedit6.enabled:=editenab;          ข้อมูลให้เป็นไปตามค่าของตัวแปร
    dbedit7.enabled:=editenab;          editenab
    dbedit8.enabled:=editenab;
    dbedit9.enabled:=editenab;
    dbedit10.enabled:=editenab;
end;
```

```
procedure tfuser.buttonenableuser(buttonenab:boolean);
                                        โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอขณะป้อน
                                        ข้อมูล
begin
        button1.enabled:=buttonenab;
        button2.enabled:=buttonenab;        ปรับสภาวะของปุ่มที่หน้าจอให้เป็น
        button3.enabled:=buttonenab;        ไปตามค่าของตัวแปร  buttonenab
        button4.enabled:=buttonenab;
end;
procedure TFuser.BitBtn1Click(Sender: TObject);
                                        โพรซีเดอร์กลับสู่หน้าจอหลัก
var
        editflag:boolean;
        pea_number:string;
begin
        if (senserform=false) then      ถ้าไม่มีการแก้ไขหรือเพิ่มเติมข้อมูลประวัติผู้ใช้
                                        ไฟฟ้าให้กลับสู่หน้าจอหลัก
        begin
                famrmain.show;          แสดงหน้าจอหลักของโปรแกรม
                fuser.Hide;             ซ่อนหน้าจอประวัติผู้ใช้ไฟฟ้า
                buttonenableuser(true); เรียกใช้โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอ
                                        ขณะป้อนข้อมูล
        end
        else                            ถ้ามีการแก้ไขหรือเพิ่มเติมข้อมูลต้องกรอกข้อมูล
                                        ให้ครบตามเงื่อนไข
        begin
                checkedit(editflag);    เรียกใช้โพรซีเดอร์ตรวจสอบการป้อนข้อมูล
                if (editflag=false) then ถ้าข้อมูลไม่ครบถ้วนรอการป้อนข้อมูลให้ครบ
                begin
                        showmessage('กรุณาป้อนข้อมูลให้ครบถ้วน');
```

```
                                แสดงข้อความ " กรุณาป้อนข้อมูลให้ครบถ้วน"
        dbedit1.SetFocus;       เลื่อนตัวชี้ไปที่ช่องการรับข้อมูล "ชื่อ"
        exit;                   หยุดรอรับการป้อนข้อมูลประวัติผู้ใช้ไฟฟ้า
end;
famrmain.show;                  แสดงหน้าจอหลักของโปรแกรม
fuser.Hide;                     ซ่อนหน้าจอประวัติผู้ใช้ไฟฟ้า
senserform:=false;
 buttonenableuser(true);        เรียกใช้โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอ
                                ขณะป้อนข้อมูล
editenableuser(false);          เรียกใช้โพรซีเดอร์ตรวจสอบการป้อนข้อมูล
 meter_datamodule.customer_Query.Post;
                                อ้างถึงมอดูลข้อมูลผู้ใช้ไฟฟ้าและบันทึกข้อมูล
meter_datamodule.customer_Query.active:=false;
                                อ้างถึงมอดูลข้อมูลของผู้ใช้ไฟฟ้าและปิดข้อมูล
meter_datamodule.customer_Query.active:=true;
                                อ้างถึงมอดูลข้อมูลของผู้ใช้ไฟฟ้าและเปิดข้อมูล
if (createflag=true) then
begin
        meter_datamodule.meter_Query.post;
                                อ้างถึงมอดูลข้อมูลของมาตรวัดและบันทึกข้อมูล
        meter_datamodule.meter_Query.active:=false;
                                อ้างถึงมอดูลข้อมูลของมาตรวัดและปิดข้อมูล
        meter_datamodule.meter_Query.active:=true;
                                อ้างถึงมอดูลข้อมูลของมาตรวัดและเปิดข้อมูล
        famrmain.inc_row_total;
                                เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                พลังงานรวม
        famrmain.inc_row_unit1;
                                เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                พลังงานช่วงเวลา 09.00- 21.59 น.
```

```
                famrmain.inc_row_unit2;
                                        เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                        พลังงานช่วงเวลา 22.00 – 08.59 น.
                dbnavigator1.Enabled:=true;
                                        ปุ่มนำทางข้อมูลทำงานตามปกติ
                createflag:=false;
            end;
            showmessage('บันทึกข้อมูลเรียบร้อย');
                                        แสดงข้อความ “บันทึกข้อมูลเรียบร้อย”
        end;
end;
procedure TFuser.Button1Click(Sender: TObject);
                                        โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า
begin
        if (senserform=false) then
        begin
            dbnavigator1.Enabled:=false;  ปุ่มนำทางข้อมูลหยุดทำงาน
            famrmain.append_customer;     เรียกใช้โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า
            editenableuser(true);         เตรียมสภาวะของหน้าจอให้พร้อมรับข้อมูล
            dbedit1.SetFocus;             เลื่อนตัวชี้ไปที่ช่องทางรับข้อมูลแรก
            button1.enabled:=false;
            button2.enabled:=false;       } สภาวะของปุ่มที่หน้าจอหยุดทำงาน
            button4.enabled:=false;
            BitBtn1.enabled:=false;
            senserform:=true;
        end
        else
        senserform:=false;
end;
```

```
procedure TFuser.Button2Click(Sender: TObject);
                                        โพรซีเดอร์แก้ไขประวัติผู้ใช้ไฟฟ้า
begin
    famrmain.edit_customer;             เรียกใช้โพรซีเดอร์แก้ไขประวัติผู้ใช้ไฟฟ้า
    editenableuser(true);               เตรียมสภาวะของหน้าจอให้พร้อมรับข้อมูล
    dbedit1.SetFocus;                   เลื่อนตัวชี้ไปที่ช่องทางรับข้อมูลแรก
    buttonenableuser(false);            สภาวะของปุ่มที่หน้าจอหยุดทำงาน
    senserform:=true;
end;
procedure TFuser.Button3Click(Sender: TObject);
                                        โพรซีเดอร์รายละเอียดประวัติมาตรวัด
var
    editflag:boolean;
begin
    if (senserform=false) then          ถ้าไม่มีการเพิ่มประวัติผู้ใช้ไฟฟ้าให้แสดงหน้าจอ
                                        ประวัติมาตรวัด
    begin
        if  (fmeter=nil) then           ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติมาตรวัด
                                        ให้สร้างหน้าจอขึ้นมาใหม่
            application.createform(tfmeter,fmeter);
        fmeter.show;                    แสดงหน้าจอประวัติมาตรวัด
        fuser.Hide;                     ซ่อนหน้าจอประวัติผู้ใช้ไฟฟ้า
        fmeter.buttonenablemeter(false);
    end
    else                                ถ้ามีการเพิ่มประวัติผู้ใช้ไฟฟ้าให้เพิ่มประวัติมาตร
                                        วัด
    begin
        checkedit(editflag);
        if (editflag=false) then        ตรวจสอบการป้อนข้อมูลประวัติผู้ใช้ไฟฟ้า
        begin
            showmessage('กรุณาป้อนข้อมูลให้ครบถ้วน');
```

```
                                        แสดงข้อความ “กรุณาป้อนข้อมูลให้ครบถ้วน”
            dbedit1.SetFocus;
            exit;                       หยุดรอรับการป้อนข้อมูลประวัติผู้ใช้ไฟฟ้า
        end;
        editenableuser(false);
        button1.enabled:=true;
        button2.enabled:=true;
        button4.enabled:=true;          สภาวะของปุ่มที่หน้าจอทำงานตาม
        bitbtn1.enabled:=true;          ปกติ
        if  (fmeter=nil) then           ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติมาตรวัด
                                        ให้สร้างหน้าจอขึ้นมาใหม่
            application.createform(tfmeter,fmeter);
        fmeter.show;                    แสดงหน้าจอประวัติมาตรวัด
        fuser.Hide;                     ซ่อนหน้าจอประวัติผู้ใช้ไฟฟ้า
        dbnavigator1.Enabled:=true;     ปุ่มนำทางข้อมูลทำงานตามปกติ
        fmeter.dbnavigator1.Enabled:=true;
        famrmain.append_meter;          เรียกใช้โพรซีเดอร์เพิ่มประวัติมาตรวัด
        fmeter.buttonenablemeter(false);
                                        เรียกใช้โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอ
        fmeter.editenablemeter(true);   เรียกใช้โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล
        fmeter.dbedit1.SetFocus;
        senserform:=true;
        createflag:=true;
    end;
end;
procedure TFuser.Button4Click(Sender: TObject);
                                        โพรซีเดอร์ติดตามการใช้ไฟฟ้า
begin
```

```
    if  (fmon=nil) then                          ถ้ายังไม่มีการเรียกใช้งานหน้าจอติดตามการใช้ไฟ
                                                 ฟ้าให้สร้างหน้าจอขึ้นมาใหม่
        application.createform(tfmon,fmon);
    fmon.show;                                   แสดงหน้าจอติดตามการใช้ไฟฟ้า
    fuser.Hide;                                  ซ่อนหน้าจอประวัติผู้ใช้ไฟฟ้า
end;
end.
```

ยูนิตประวัติมาตรวัด

```
unit meter;
interface
uses
        Windows, Messages, SysUtils, Classes,
        Graphics, Controls, Forms, Dialogs,
        ExtCtrls, StdCtrls, Buttons, DBCtrls, Db,
        DBTables, Mask;
type
        TFmeter = class(TForm)
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        Label4: TLabel;
        Label5: TLabel;
        Label6: TLabel;
        Label7: TLabel;
        Label8: TLabel;
        Label9: TLabel;
        Bevel1: TBevel;
        BitBtn1: TBitBtn;
        Button1: TButton;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    DBEdit7: TDBEdit;
    DBEdit8: TDBEdit;
    DBNavigator1: TDBNavigator;
    DBText1: TDBText;
```

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
procedure BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์กลับสู่หน้าจอหลัก

```
procedure Button1Click(Sender: TObject);
```
โพรซีเดอร์เพิ่มประวัติมาตรวัด

```
procedure Button2Click(Sender: TObject);
```
โพรซีเดอร์แก้ไขประวัติมาตรวัด

```
procedure Button3Click(Sender: TObject);
```
โพรซีเดอร์รายละเอียดประวัติผู้ใช้ไฟฟ้า

```
procedure Button4Click(Sender: TObject);
```
โพรซีเดอร์ติดตามการใช้ไฟฟ้า

```
private
    { Private declarations }
procedure checkedit(var editflag:boolean);
```
โพรซีเดอร์ตรวจสอบการป้อนข้อมูล

```
public
    { Public declarations }
procedure buttonenablemeter(buttonenab:boolean);
```
โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอขณะป้อนข้อมูล

```
procedure editenablemeter(editenab:boolean);
```
โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล

```
end;
var
```

```
        Fmeter: TFmeter;
implementation
uses amrmain,user,centralvar, meter_module;     เรียกใช้ยูนิต amrmain, user, centralvar และ
                                                meter_module
{$R *.DFM}
procedure tfmeter.checkedit(var editflag:boolean);
                                                โพรซีเดอร์ตรวจสอบการป้อนข้อมูล
begin
        if (dbedit1.text ='') or (dbedit2.text ='') or (dbedit3.text ='') or
        (dbedit4.text ='') or (dbedit5.text ='') or (dbedit6.text ='') or
        (dbedit7.text ='') or (dbedit8.text ='') or (dbedit9.text ='')
                                                ตรวจสอบการป้อนข้อมูลที่ช่องรับข้อมูลถ้ามีการ
                                                ป้อนข้อมูลไม่ครบตามเงื่อนไขที่กำหนดค่าของตัว
                                                แปร editflag จะเป็นเท็จ
        then
                editflag:=false
        else
                editflag:=true;
        end;
procedure tfmeter.editenablemeter(editenab:boolean);
                                                โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล
begin
        dbedit1.enabled:=editenab;
        dbedit2.enabled:=editenab;
        dbedit3.enabled:=editenab;              เคลียร์และปรับสภาวะของช่องรับ
        dbedit4.enabled:=editenab;              ข้อมูลให้เป็นไปตามค่าของตัวแปร
         dbedit5.enabled:=editenab;             editenab
        dbedit6.enabled:=editenab;
        dbedit7.enabled:=editenab;
end;
```

```
procedure tfmeter.buttonenablemeter(buttonenab:boolean);
                                        โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอขณะป้อน
                                        ข้อมูล
begin
        button1.enabled:=buttonenab;
        button2.enabled:=buttonenab;        } ปรับสภาวะของปุ่มที่หน้าจอให้เป็น
        button3.enabled:=buttonenab;        } ไปตามค่าของตัวแปร buttonenab
        button4.enabled:=buttonenab;
end;
procedure TFmeter.BitBtn1Click(Sender: TObject);
                                        โพรซีเดอร์กลับสู่หน้าจอหลัก
var
        editflag:boolean;
        pea_number:string;
begin
        if (senserform=false) then      ถ้าไม่มีการแก้ไขหรือเพิ่มเติมข้อมูลประวัติมาตร
                                        วัดให้กลับสู่หน้าจอหลัก
        begin
                famrmain.show;          แสดงหน้าจอหลักของโปรแกรม
                fmeter.Hide;            ซ่อนหน้าจอประวัติมาตรวัด
                buttonenablemeter(true);
        end
        else                            ถ้ามีการแก้ไขหรือเพิ่มเติมข้อมูลต้องกรอกข้อมูล
                                        ให้ครบตามเงื่อนไข
        begin
                checkedit(editflag);    เรียกใช้โพรซีเดอร์ตรวจสอบการป้อนข้อมูล
                if (editflag=false) then ถ้าข้อมูลไม่ครบถ้วนรอการป้อนข้อมูลให้ครบ
                begin
                        showmessage('กรุณาป้อนข้อมูลให้ครบถ้วน');
                                        แสดงข้อความ “กรุณาป้อนข้อมูลให้ครบถ้วน”
```

```
                dbedit1.SetFocus;        เลื่อนตัวชี้ไปที่ช่องรับข้อมูลแรก
                exit;
        end;
famrmain.show;                           แสดงหน้าจอหลักของโปรแกรม
fmeter.Hide;                             ซ่อนหน้าจอประวัติมาตรวัด
senserform:=false;
buttonenablemeter(true);                 เรียกใช้โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอ
                                         ขณะป้อนข้อมูล
editenablemeter(false);                  เรียกใช้โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล
meter_datamodule.meter_Query.Post;       อ้างถึงมอดูลข้อมูลของมาตรวัดและบันทึกข้อมูล
meter_datamodule.meter_Query.active:=false;
                                         อ้างถึงมอดูลข้อมูลของมาตรวัดและปิดข้อมูล
meter_datamodule.meter_Query.active:=true;
                                         อ้างถึงมอดูลข้อมูลของมาตรวัดและเปิดข้อมูล
if (createflag=true) then
begin
        meter_datamodule.customer_Query.post;
                                         อ้างถึงมอดูลข้อมูลผู้ใช้ไฟฟ้าและบันทึกข้อมูล
        meter_datamodule.customer_Query.active:=false;
                                         อ้างถึงมอดูลข้อมูลผู้ใช้ไฟฟ้าและปิดข้อมูล
        meter_datamodule.customer_Query.active:=true;
                                         อ้างถึงมอดูลข้อมูลผู้ใช้ไฟฟ้าและเปิดข้อมูล
        famrmain.inc_row_total;
                                         เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                         พลังงานรวม
        famrmain.inc_row_unit1;
                                         เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                         พลังงานช่วงเวลา 09.00- 21.59 น.
        famrmain.inc_row_unit2;
```

```
                                        เรียกใช้โพรซีเดอร์เพิ่มจำนวนรายการบันทึกหน่วย
                                        พลังงานช่วงเวลา 22.00 – 08.59 น.
            dbnavigator1.Enabled:=true;  ปุ่มนำทางข้อมูลทำงานตามปกติ
            createflag:=false;
        end;
        showmessage('บันทึกข้อมูลเรียบร้อย');  แสดงข้อความ “บันทึกข้อมูลเรียบร้อย”
        end;
end;
procedure TFmeter.Button1Click(Sender: TObject);
                                        โพรซีเดอร์เพิ่มประวัติมาตรวัด
begin
        if (senserform=false) then
        begin
            dbnavigator1.Enabled:=false;  ปุ่มนำทางข้อมูลหยุดทำงาน
            famrmain.append_meter;        เรียกใช้โพรซีเดอร์เพิ่มประวัติมาตรวัด
            editenablemeter(true);        เตรียมสภาวะของหน้าจอให้พร้อมรับข้อมูล
            dbedit1.SetFocus;             เลื่อนตัวชี้ไปที่ช่องทางการรับข้อมูลแรก
            button1.enabled:=false;
            button2.enabled:=false;       } สภาวะของปุ่มที่หน้าจอหยุดทำงาน
            button4.enabled:=false;
            BitBtn1.enabled:=false;
            senserform:=true;
        end
        else
        senserform:=false;
end;
procedure TFmeter.Button2Click(Sender: TObject);
                                        โพรซีเดอร์แก้ไขประวัติมาตรวัด
begin
        famrmain.edit_meter;            เรียกใช้โพรซีเดอร์แก้ไขประวัติมาตรวัด
```

```
        editenablemeter(true);              เตรียมสภาวะของหน้าจอให้พร้อมรับข้อมูล
        dbedit1.SetFocus;                   เลื่อนตัวชี้ไปที่ช่องทางรับข้อมูลแรก
        buttonenablemeter(false);           สภาวะของปุ่มที่หน้าจอหยุดทำงาน
        senserform:=true;
end;
procedure TFmeter.Button3Click(Sender: TObject);
                                            รายละเอียดประวัติผู้ใช้ไฟฟ้า
var
        editflag:boolean;
begin
        if (senserform=false) then          ถ้าไม่มีการเพิ่มประวัติมาตรวัดให้แสดงหน้าจอ
                                            ประวัติผู้ใช้ไฟฟ้า
        begin
                if  (fuser=nil) then        ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติผู้ใช้ไฟฟ้า
                                            ให้สร้างหน้าจอขึ้นมาใหม่
                        application.createform(tfuser,fuser);
                fuser.show;                 แสดงหน้าจอประวัติผู้ใช้ไฟฟ้า
                fmeter.Hide;                ซ่อนหน้าจอประวัติมาตรวัด
                fuser.buttonenableuser(false);
        end
        else                                ถ้ามีการเพิ่มประวัติมาตรวัดให้เพิ่มประวัติผู้ใช้ไฟ
        begin
                checkedit(editflag);
                if (editflag=false) then    ตรวจสอบการป้อนข้อมูลประวัติมาตรวัดให้ครบ
                begin
                        showmessage('กรุณาป้อนข้อมูลให้ครบถ้วน');
                                            แสดงข้อความ “กรุณาป้อนข้อมูลให้ครบถ้วน”
                        dbedit1.SetFocus;
                        exit;               หยุดรอรับการป้อนข้อมูลประวัติมาตรวัด
                end;
```

```
                editenablemeter(false);
                button1.enabled:=true;
                button2.enabled:=true;          } สภาวะของปุ่มที่หน้าจอทำงานตาม
                button4.enabled:=true;            ปกติ
                bitbtn1.enabled:=true;
                if (fuser=nil) then             ถ้ายังไม่มีการเรียกใช้งานหน้าจอประวัติผู้ใช้ไฟฟ้า
                                                ให้สร้างหน้าจอขึ้นมาใหม่
                    application.createform(tfuser,fuser);
                fuser.show;                     แสดงหน้าจอประวัติผู้ใช้ไฟฟ้า
                fmeter.Hide;                    ซ่อนหน้าจอประวัติมาตรวัด
                dbnavigator1.Enabled:=true;     ปุ่มนำทางข้อมูลทำงานตามปกติ
                fuser.dbnavigator1.Enabled:=true;
                famrmain.append_customer;       เรียกใช้โพรซีเดอร์เพิ่มประวัติผู้ใช้ไฟฟ้า
                fuser.buttonenableuser(false); เรียกใช้โพรซีเดอร์ปรับสภาวะของปุ่มที่หน้าจอ
                                                ขณะป้อนข้อมูล
                fuser.editenableuser(true);     เรียกใช้โพรซีเดอร์ปรับสภาวะของช่องรับข้อมูล
                senserform:=true;
                createflag:=true;
        end;
end;
procedure TFmeter.Button4Click(Sender: TObject);
                                                โพรซีเดอร์ติดตามการใช้ไฟฟ้า
begin
    if (fmon=nil) then                          ถ้ายังไม่มีการเรียกใช้งานหน้าจอติดตามการใช้ไฟ
                                                ฟ้าให้สร้างหน้าจอขึ้นมาใหม่
        application.createform(tfmon,fmon);
    fmon.show;                                  แสดงหน้าจอติดตามการใช้ไฟฟ้า
    fmeter.Hide;                                ซ่อนหน้าจอประวัติมาตรวัด
end;
end.
```

ยูนิตเชื่อมต่อระบบ

```
unit netamr;
interface
uses
        Windows, Messages, SysUtils, Classes,
        Graphics, Controls, Forms, Dialogs,
        StdCtrls, ScktComp, Buttons, ExtCtrls;
type
        TFnetamr = class(TForm)
        ClientSocket1: TClientSocket;
        ServerSocket1: TServerSocket;
        GroupBox1: TGroupBox;
        GroupBox2: TGroupBox;
        GroupBox3: TGroupBox;
        Label1: TLabel;
        Label2: TLabel;
        Label3: TLabel;
        Button1: TButton;
        Button2: TButton;
        Button3: TButton;
        Edit1: TEdit;
        Edit2: TEdit;
        Edit3: TEdit;
        BitBtn1: TBitBtn;
        BitBtn2: TBitBtn;
        RadioButton1: TRadioButton;
        RadioButton2: TRadioButton;
        memo1: TMemo;
        Panel1: TPanel;
        Panel2: TPanel;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

Timer1: TTimer;
RadioButton3: TRadioButton;
Button4: TButton;
} กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

procedure Button1Click(Sender: TObject); โพรซีเดอร์รอรับข้อมูล

procedure Button2Click(Sender: TObject); โพรซีเดอร์เชื่อมต่อมาตรวัด

procedure serversocket1accept(sender:tobject;socket:tcustomwinsocket);
โพรซีเดอร์แสดงการติดต่อกับตัวรวมช่องสัญญาณ

procedure serversocketclienterror(sender:tobject;socket:tcustomwinsocket;errorevnet:
terroreventvar errorcode :integer); โพรซีเดอร์แสดงการติดต่อกับตัวรวมช่องสัญญาณเกิดความผิดพลาด

procedure serversocket1clientread(sender:tobject;socket:tcustomwinsocket);
โพรซีเดอร์อ่านข้อมูลที่ส่งมาจากมาตรวัด

procedure serversocket1listen(sender:tobject;socket:tcustomwinsocket);
โพรซีเดอร์แสดงการรอรับการติดต่อจากมาตรวัด

procedure clientsocket1connect(sender:tobject;socket:tcustomwinsocket);
โพรซีเดอร์แสดงการติดต่อกับมาตรวัด

procedure clientsocket1error(sender:tobject;socket:tcustomwinsocket;errorevent:terrorevent;var
errorcode:integer); โพรซีเดอร์แสดงการติดต่อกับมาตรวัดเกิดความผิดพลาด

procedure clientsocket1lookup(sender:tobject;socket:tcustomwinsocket);
โพรซีเดอร์มองหามาตรวัดที่ต้องการติดต่อ

procedure BitBtn2Click(Sender: TObject); โพรซีเดอร์กลับสู่หน้าจอหลักของโปรแกรม

procedure Timer1Timer(Sender: TObject); โพรซีเดอร์ระบบเวลาที่ยูนิตใช้แสดง

procedure BitBtn1Click(Sender: TObject); โพรซีเดอร์ยกเลิกการเชื่อมต่อระบบ

procedure Button4Click(Sender: TObject); โพรซีเดอร์ยกเลิกการเชื่อมต่อกับมาตรวัด

procedure ClientSocket1Disconnect(Sender: TObject;Socket: TCustomWinSocket);
โพรซีเดอร์แสดงมาตรวัดหยุดการติดต่อ

procedure Button3Click(Sender: TObject); โพรซีเดอร์ส่งข้อมูลให้มาตรวัด

procedure RadioButton1Click(Sender: TObject); โพรซีเดอร์ตั้งเวลามาตรวัด

procedure RadioButton2Click(Sender: TObject); โพรซีเดอร์ติดตามการใช้ไฟฟ้า

```
procedure RadioButton3Click(Sender: TObject);โพรซีเดอร์ส่งหน่วยพลังงานรอบเดือน
procedure FormCreate(Sender: TObject);        โพรซีเดอร์แสดงข้อมูลพร้อมกับการสร้างหน้าจอ
private
        { Private declarations }
        connectsocket:tcustomwinsocket;       ประกาศตัวแปรที่ใช้ส่งผ่านข้อมูล
        status : integer;                     ประกาศตัวแปรที่ใช้ส่งรหัสส่วนการทำงาน
        procedure echo(content:shortstring);  โพรซีเดอร์แสดงข้อมูลในหน้าจอเชื่อมต่อระบบ
public
        { Public declarations }
end;
var
         Fnetamr: TFnetamr;                   ตัวแปรของหน้าจอเชื่อมต่อระบบ
implementation
uses amrmain,centralvar,meter_module;         เรียกใช้ยูนิต amrmain, centralvar และ
                                              meter_module
{$R *.DFM}
procedure Tfnetamr.Timer1Timer(Sender: TObject);
                                              โพรซีเดอร์ระบบเวลาที่ยูนิตใช้แสดง
begin
        timeseparator:='-';                   จัดข้อมูลของเวลาเป็น “ชั่วโมง-นาที-วินาที”
        panel1.caption:=timetostr(now);       แสดงข้อมูลของเวลา
        dateseparator:='-';                   จัดข้อมูลของวันเป็น “วัน-เดือน-ปี”
        panel2.caption :=datetostr(date);     แสดงข้อมูลของวันเดือนปี
end;
procedure tFnetamr.echo(content:shortstring);
                                              โพรซีเดอร์แสดงข้อมูลในหน้าจอเชื่อมต่อระบบ
begin
         memo1.Lines.add(content);            แสดงข้อมูลที่รับมาจากโพรซีเดอร์อื่น
end;
```

```
procedure TFnetamr.Button1Click(Sender: TObject);
                                        โพรซีเดอร์รอรับข้อมูล
var
        portno:integer;                 ประกาศตัวแปรหมายเลขช่องทางรับข้อมูล
begin
        echo('ตัวรวมช่องสัญญาณเริ่มทำงาน'); แสดงข้อความ “ตัวรวมช่องสัญญาณเริ่มทำงาน”
        portno:=strtoint(edit1.text);   แปลงข้อมูลตัวอักษรที่รับเข้ามาเป็นตัวเลข
        edit1.Enabled:=false;           กำหนดไม่ให้ทำการแก้ไขตัวแปรในหน้าจอ
        button1.Enabled:=false;
        serversocket1.port:=portno;     หมายเลขช่องทางในการรับข้อมูลของตัวรวมช่อง
                                        สัญญาณ
        serversocket1.Open;             เปิดช่องทางการรับข้อมูล
end;
procedure TFnetamr.Button2Click(Sender: TObject);
                                        โพรซีเดอร์เชื่อมต่อมาตรวัด
var
        hostname:shortstring;           ประกาศตัวแปรที่ใช้รับหมายเลขแอดเดรส IP
                                        มาตรวัด
        portno:integer;                 ประกาศตัวแปรหมายเลขช่องทางการส่งข้อมูล
begin
        status:=0;                      เคลียร์รหัสส่วนการทำงาน
        hostname:=edit2.text;
        if (hostname='') then           ถ้ายังไม่มีหมายเลขแอดเดรส IP มาตรวัดที่จะติด
                                        ต่อให้แสดงข้อความ “ กรุณาพิมพ์หมายเลขแอด
                                        เดรส IP ”
        begin
                echo('กรุณาพิมพ์ หมายเลขแอดเดรส IP);
                edit2.SetFocus;
                exit;                   รอรับการป้อนข้อมูล
        end;
```

```
        echo('เริ่มการติดต่อด้วย TCP');          แสดงข้อความ “ เริ่มการติดต่อด้วย TCP”
        portno:=strtoint(edit3.text);            แปลงข้อมูลตัวอักษรที่รับเข้ามาเป็นตัวเลข
        groupbox3.Enabled:=true;
        edit2.Enabled:=false;
        edit3.Enabled:=false;
        button2.enabled:=false;
        button4.Enabled:=true;                   กำหนดสภาพแวดล้อมของหน้าจอ
        radiobutton1.Enabled:=true;              ให้รับข้อมูลได้เฉพาะที่จำเป็น
        radiobutton2.Enabled:=true;
        radiobutton3.Enabled:=true;
        clientsocket1.host:=hostname;            หมายเลขแอดเดรส IP ของมาตรวัด
        clientsocket1.port:=portno;              หมายเลขช่องทางที่ใช้ในการส่งข้อมูล
        clientsocket1.open;                      เปิดช่องทางในการส่งข้อมูล
end;
procedure tFnetamr.serversocket1accept(sender:tobject;socket:tcustomwinsocket);
                                                 โพรซีเดอร์แสดงการติดต่อกับตัวรวมช่องสัญญาณ
begin
        echo('มีการเชื่อมต่อ');                  แสดงข้อความ “ มีการเชื่อมต่อ ”
        connectsocket:=socket;                   เก็บค่าช่องทางในการติดต่อ
end;
proceduretFnetamr.serversocketclienterror(sender:tobject;socket:tcustomwinsocket;
errorevnet:terrorevent ;var errorcode :integer); โพรซีเดอร์แสดงการติดต่อกับตัวรวมช่อง
                                                 สัญญาณเกิดความผิดพลาด
begin
        echo('เกิดความผิดพลาด');                แสดงข้อความ “เกิดความผิดพลาด”
end;
```

```
procedure tFnetamr.serversocket1clientread(sender:tobject;socket:tcustomwinsocket);
                                                    โพรซีเดอร์อ่านข้อมูลที่ส่งมาจากมาตรวัด
var
        s:shortstring;
        sd:pchar;
        p:pchar;                                    ประกาศตัวแปรเพื่อช่วยในการแยก
        buff:pchar;                                 ข้อมูล
        l:integer;
        size:integer;
        cust_no_i:integer;
        cust_no_x:variant;
        table_name1:string;                         ประกาศตัวแปรเพื่อช่วยในการจัด
        time_data:integer;                          เก็บข้อมูล
        unit_1:integer;
        unit_2:integer;
        sum_unit:integer;
begin
        buff:=' ';
        p:=' ';
        table_name1:=' ';                           เคลียร์ค่าตัวแปร
        unit_1:=0;
        unit_2:=0;
        fillchar(s,sizeof(s),0);
        p:=@s[1];                                   เก็บตำแหน่งของตัวแปร s
        size:=socket.receivebuf(p^,sizeof(s));      เก็บข้อมูลที่รับได้ตามตำแหน่งของตัวแปร s
        s[0]:=chr(size);                            แปลงข้อมูลที่รับได้เป็นตัวอักษร
        echo('ข้อมูลที่รับ');                       แสดงข้อความ “ข้อมูลที่รับ”
        echo(' >'+s+'< ');                          แสดงข้อมูลที่รับ
        code:=0;                                    เคลียร์ค่าตัวแปรเก็บรหัสส่วนการทำงาน
```

```
if (p<>' ')then                    ถ้ารับข้อมูลได้ให้หาตัวอักษร “-” แล้วทำการแยก
                                   ข้อมูล
begin
        sd:=strscan(p,'-');
        l:=strlen(p)-strlen(sd);
        strlcopy(buff,p,l);        แยกข้อมูลหมายเลขแอดเดรส IP เก็บ
        tcpip_x:=buff;             ไว้ที่ตัวแปร tcpip_x
        sd:=sd+1;
        strcopy(p,sd);
        sd:=strscan(p,'-');
        l:=strlen(p)-strlen(sd);
        strlcopy(buff,p,l);        แยกข้อมูลหมายเลขรหัสส่วนการ
        code:=strtoint(buff);      ทำงาน เก็บไว้ที่ตัวแปร code
        sd:=sd+1;
        strcopy(p,sd);
        case code of               เลือกรหัสส่วนการทำงาน
            2: begin               รหัสส่วนการทำงานการติดตามการใช้ไฟฟ้า
                sd:=strscan(p,'-');
                l:=strlen(p)-strlen(sd);
                strlcopy(buff,p,l);        แยกข้อมูลพลังงานหนึ่งหน่วยเก็บ
                eng_m:=strtoint(buff);     ไว้ที่ตัวแปร eng_m
                sd:=sd+1;
                strcopy(p,sd);
                sd:=strscan(p,'-');
                l:=strlen(p)-strlen(sd);
                strlcopy(buff,p,l);        แยกข้อมูลเวลา วินาที เก็บไว้ที่ตัว
                time_s:=buff;              แปร time_s
                sd:=sd+1;
                strcopy(p,sd);
```

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);          แยกข้อมูลเวลา นาที เก็บไว้ที่ตัว
time_m:=buff;                แปร time_m
sd:=sd+1;
strcopy(p,sd);
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);     แยกข้อมูลเวลา ชั่วโมง เก็บไว้ที่ตัว
strlcopy(buff,p,l);          แปร time_h
time_h:=buff;
new_time_h:=strtoint(time_h);
sd:=sd+1;
strcopy(p,sd);
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);          แยกข้อมูล วันที่ เก็บไว้ที่ตัวแปร
date_d:=buff;                date_d
sd:=sd+1;
strcopy(p,sd);
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);          แยกข้อมูล เดือน เก็บไว้ที่ตัวแปร
date_m:='mount_'+buff;       date_m
sd:=sd+1;
strcopy(p,sd);
date_y:=p;                   แยกข้อมูล ปี เก็บไว้ที่ตัวแปร
                             date_y
with meter_datamodule.meter_query do
begin          อ้างถึงฐานข้อมูลมาตรวัด
```

```
        cust_no_x := Lookup('tcpip',tcpip_x,'cust_no');
                ค้นหาข้อมูลเมื่อหมายเลขผู้ใช้ไฟฟ้า cust_no จาก
                ขอบเขตข้อมูล tcpip ที่มีหมายเลขแอดเดรส IP
                ตรงกับตัวแปร tcpip_x
        cust_no_i:=cust_no_x;
                แปลงประเภทของข้อมูล
        cust_no_s:=inttostr(cust_no_i);
                แปลงหมายเลข cust_no ให้เป็นตัวอักษร
end;
table_name1:='t_cust_h'+cust_no_s;
meter_datamodule.table1.TableName:=table_name1;
                กำหนดชื่อของตารางเป็น “t_cust_h หมายเลขผู้ใช้
                ไฟฟ้า ”
if not meter_datamodule.table1.Exists then
                ถ้ายังไม่มีตารางในฐานข้อมูลให้สร้างตาราง
        famrmain.create_table_h(cust_no_s);
table_name2:='profile'+cust_no_s;
meter_datamodule.table2.TableName:=table_name2;
                กำหนดชื่อของตารางเป็น “profile หมายเลขผู้ใช้
                ไฟฟ้า”
 if not meter_datamodule.table2.Exists then
                ถ้ายังไม่มีตารางในฐานข้อมูลให้สร้างตาราง
        famrmain.create_table_2(cust_no_s);
meter_datamodule.table2.active:=true;
                เปิดการติดต่อข้อมูล
meter_datamodule.table2.append;
                เพิ่มข้อมูล
meter_datamodule.table2.FieldValues['time_hour']:=time_data;
                บันทึกข้อมูลวัน
```

```
meter_datamodule.table2.FieldValues['time_min']:=time_m;
        บันทึกข้อมูลนาที
meter_datamodule.table2.FieldValues['time_sec']:=time_s;
        บันทึกข้อมูลวินาที
meter_datamodule.table2.FieldValues['energyunit']:=eng_m;
        บันทึกข้อมูลหน่วยพลังงาน
meter_datamodule.table2.post;
        บันทึกข้อมูล
meter_datamodule.table2.active:=false;
        ปิดการติดต่อข้อมูล
meter_datamodule.table2.active:=true;
        เปิดการติดต่อข้อมูล
if (new_time_h>old_time_h)then
        ถ้าเวลาชั่วโมงใหม่มีค่ามากกว่าเวลาชั่วโมงเดิมให้
        เก็บค่าชั่วโมงใหม่และเพิ่มค่าตัวชี้เป็น 4 เท่า
begin
    old_time_h:=new_time_h;
    ind_count:=new_time_h*4;
    if (new_time_m<=15)then
        ถ้าเวลานาทีน้อยกว่าหรือเท่ากับ 15 นาที เพิ่มค่า
        ตัวชี้และบันทึกหน่วยพลังงานลงตาราง “t_cust_h
        หมายเลขผู้ใช้ไฟฟ้า”
    begin
        stat_eng:=1;
        ind_count:=ind_count+1;
        eng_buff:=1;
        meter_datamodule.table1.append;
        เพิ่มข้อมูล
meter_datamodule.table1.FieldValues['time_read']:=ind_count;
        บันทึกข้อมูลตัวชี้
```

```
meter_datamodule.table1.FieldValues['energyunit']:=eng_m;
            บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
        end;
        if (new_time_m>15)and (new_time_m<=30)then
            ถ้าเวลานาทีมากกว่า 15 ถึง 30 นาที เพิ่มค่าตัวชี้
            และบันทึกหน่วยพลังงานลงตาราง “t_cust_h
            หมายเลขผู้ใช้ไฟฟ้า”
        begin
              stat_eng:=2;
              ind_count:=ind_count+2;
              eng_buff:=1;
              meter_datamodule.table1.append;
            เพิ่มข้อมูล
meter_datamodule.table1.FieldValues['time_read']:=ind_count;
            บันทึกข้อมูลตัวชี้
meter_datamodule.table1.FieldValues['energyunit']:=eng_m;
            บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
        end;
        if (new_time_m>30) and (new_time_m<=45)then
            ถ้าเวลานาทีมากกว่า 30 ถึง 45 นาที เพิ่มค่าตัวชี้
            และบันทึกหน่วยพลังงานลงตาราง “t_cust_h
            หมายเลขผู้ใช้ไฟฟ้า”
        begin
              stat_eng:=3;
              ind_count:=ind_count+3;
              eng_buff:=1;
              meter_datamodule.table1.append;
            เพิ่มข้อมูล
```

```
meter_datamodule.table1.FieldValues['time_read']:=indcount;
        บันทึกข้อมูลตัวชี้
meter_datamodule.table1.FieldValues['energyunit']:=eng_m;
        บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
    end;
    if (new_time_m>45) and (new_time_m<=60)then
        ถ้าเวลานาทีมากกว่า 45 ถึง 60 นาที เพิ่มค่าตัวชี้
        และบันทึกหน่วยพลังงานลงตาราง “t_cust_h
        หมายเลขผู้ใช้ไฟฟ้า”
    begin
        stat_eng:=4;
        ind_count:=ind_count+4;
        eng_buff:=1;
        meter_datamodule.table1.append;
        เพิ่มข้อมูล
meter_datamodule.table1.FieldValues['time_read']:=ind_count;
        บันทึกข้อมูลตัวชี้
meter_datamodule.table1.FieldValues['energyunit']:=eng_m;
        บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
    end;
    meter_datamodule.table1.post;
        บันทึกข้อมูลที่แก้ไข/เพิ่มเติม
    meter_datamodule.table1.active:=false;
        ปิดการติดต่อข้อมูล
    meter_datamodule.table1.active:=true;
        เปิดการติดต่อข้อมูล
end
else
        ถ้าเวลาชั่วโมงเป็นค่าเดียวกับที่ตารางเก็บไว้เดิม
        หรือน้อยกว่า
```

```
begin
        if (new_time_m<=15)then
            ถ้าเวลานาทีน้อยกว่าหรือเท่ากับ 15 นาที และค่าตัว
            แปรช่วย stat_eng ไม่เท่ากับ 1 ให้เริ่มต้นบันทึกข้อ
            มูลพลังงานใหม่
        begin
            if (stat_eng<>1)then
            begin
                    ind_count:=(new_time_h*4)+1;
                    eng_buff:=1;
                    meter_datamodule.table1.append;
            เพิ่มข้อมูล
meter_datamodule.table1.FieldValues['time_read']:=ind_count;
            บันทึกข้อมูลตัวชี้
meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
            บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                    stat_eng:=1;
                    end
                    else
            ถ้า stat_eng เป็น 1 ให้บันทึกข้อมูลพลังงานรวม
            เพิ่มขึ้น
                    begin
                    meter_datamodule.table1.last;
            เลื่อนไปตำแหน่งสุดท้ายของข้อมูล
                    eng_buff:=eng_buff+1;
                    meter_datamodule.table1.edit;
            แก้ไขข้อมูล
meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
            บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                    stat_eng:=1;
```

```
                end;
            end;
        if (new_time_m>15)and (new_time_m<=30)then
            ถ้าเวลานาทีมากกว่า 15 ถึง 30 นาที และค่าตัวแปร
            ช่วย stat_eng ไม่เท่ากับ 2 ให้เริ่มต้นบันทึกข้อมูล
            พลังงานใหม่

        begin
            if (stat_eng<>2)then
                begin
                ind_count:=(new_time_h*4)+2;
                eng_buff:=1;
                meter_datamodule.table1.append;
            เพิ่มข้อมูล
    meter_datamodule.table1.FieldValues['time_read']:=ind_count;
            บันทึกข้อมูลตัวชี้
    meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
            บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                stat_eng:=2;
            end
            else
            ถ้า stat_eng เป็น 2 ให้บันทึกข้อมูลพลังงานรวม
            เพิ่มขึ้น
            begin
                meter_datamodule.table1.last;
            เลื่อนไปตำแหน่งสุดท้ายของข้อมูล
                eng_buff:=eng_buff+1;
                meter_datamodule.table1.edit;
            แก้ไขข้อมูล
  meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
```

```
                บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                        stat_eng:=2;
                  end;
            end;
            if (new_time_m>30) and (new_time_m<=45)then
                ถ้าเวลานาทีมากกว่า 30 ถึง 45 นาที และค่าตัวแปร
                ช่วย stat_eng ไม่เท่ากับ 3 ให้เริ่มต้นบันทึกข้อมูล
                พลังงานใหม่
            begin
                  if (stat_eng<>3)then
                  begin
                        ind_count:=(new_time_h*4)+3;
                        eng_buff:=1;
                        meter_datamodule.table1.append;
                เพิ่มข้อมูล
meter_datamodule.table1.FieldValues['time_read']:=ind_count;
                บันทึกข้อมูลตัวชี้
meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
                บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                        stat_eng:=3;
                  end
                  else
                ถ้า stat_eng เป็น 3 ให้บันทึกข้อมูลพลังงานรวม
                เพิ่มขึ้น
                  begin
                        meter_datamodule.table1.last;
                เลื่อนไปตำแหน่งสุดท้ายของข้อมูล
                        eng_buff:=eng_buff+1;
                        meter_datamodule.table1.edit;
                แก้ไขข้อมูล
```

```
meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
                บันทึกข้อมูลหน่วยพลังงาน
                        stat_eng:=3;
                  end;
            end;
            if (new_time_m>45) and (new_time_m<=60)then
                ถ้าเวลานาทีมากกว่า 45 ถึง 60 นาที และค่าตัวแปร
                ช่วย stat_eng ไม่เท่ากับ 4 ให้เริ่มต้นบันทึกข้อมูล
                พลังงานใหม่
            begin
                  if (stat_eng<>4)then
                  begin
                        ind_count:=(new_time_h*4)+4;
                        eng_buff:=1;
                        meter_datamodule.table1.append;
                เพิ่มข้อมูล
      meter_datamodule.table1.FieldValues['time_read']:=ind_count;
                บันทึกข้อมูลตัวชี้
      meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
                บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                        stat_eng:=4;
                  end
                  else
                ถ้า stat_eng เป็น 4 ให้บันทึกข้อมูลพลังงานรวม
                เพิ่มขึ้น
                  begin
                        meter_datamodule.table1.last;
                เลื่อนไปตำแหน่งสุดท้ายของข้อมูล
                        eng_buff:=eng_buff+1;
```

```
                    meter_datamodule.table1.edit;
                แก้ไขข้อมูล
        meter_datamodule.table1.FieldValues['energyunit']:=eng_buff;
                บันทึกข้อมูลหน่วยพลังงานไฟฟ้า
                    stat_eng:=4;
                end;
            end;
            meter_datamodule.table1.post;
                บันทึกข้อมูลที่แก้ไข/เพิ่มเติม
            meter_datamodule.table1.active:=false;
                ปิดการติดต่อข้อมูล
            meter_datamodule.table1.active:=true;
                เปิดการติดต่อข้อมูล
        end;
    if (time_data>=9) and (time_data<22) then
                ถ้าเวลา 09.00 – 21.59 น. สะสมหน่วยพลังงาน
                และบันทึกข้อมูล
        begin
            with meter_datamodule.mount1_Query do
                อ้างถึงฐานข้อมูลที่เก็บหน่วยพลังงานช่วงเวลา
                09.00 – 21.59 น.
            begin
            unit_1 := Lookup('cust_no',cust_no_x,date_m);
                ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
                เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
                edit;
                แก้ไขข้อมูลหน่วยพลังงาน
                unit_1:=unit_1+eng_m;
                สะสมข้อมูลหน่วยพลังงาน
                FieldValues[date_m]:=unit_1;
```

```
            เก็บค่าของข้อมูล
        end;
        meter_datamodule.mount1_query.post;
            บันทึกข้อมูล
        meter_datamodule.mount1_query.Active:=false;
            ยกเลิกการติดต่อกับข้อมูล
        meter_datamodule.mount1_query.active:=true;
            เปิดการติดต่อกับข้อมูล
end
else        ถ้าเวลา 22.00 – 08.59 น. สะสมหน่วยพลังงาน
            และบันทึกข้อมูล
begin
        with meter_datamodule.mount2_Query do
            อ้างถึงฐานข้อมูลที่เก็บหน่วยพลังงานช่วงเวลา
            22.00 – 08.59 น.
        begin
         unit_2 := Lookup('cust_no',cust_no_x,date_m);
            ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
            เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
              edit;
            แก้ไขข้อมูลหน่วยพลังงาน
              unit_2:=unit_2+eng_m;
            สะสมข้อมูลหน่วยพลังงาน
              FieldValues[date_m]:=unit_2;
            เก็บค่าของข้อมูล
        end;
        meter_datamodule.mount2_query.post;
            บันทึกข้อมูล
        meter_datamodule.mount2_query.active:=false;
            ยกเลิกการติดต่อกับข้อมูล
```

```
                meter_datamodule.mount2_query.active:=true;
                        เปิดการติดต่อกับข้อมูล
            end;
            with meter_datamodule.mount_Query do
                        อ้างถึงฐานข้อมูลหน่วยพลังงานรวม
            begin
                sum_unit:= Lookup('cust_no',cust_no_x,date_m);
                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
                        เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
                edit;
                        แก้ไขข้อมูลหน่วยพลังงาน
                sum_unit:=unit_1+unit_2;
                        สะสมข้อมูลหน่วยพลังงานทุกช่วงเวลา
                FieldValues[date_m]:=sum_unit;
                        เก็บค่าของข้อมูล
            end;
            meter_datamodule.mount_query.post;
                        บันทึกข้อมูล
            meter_datamodule.mount_query.active:=false;
                        ยกเลิกการติดต่อกับข้อมูล
            meter_datamodule.mount_query.active:=true;
                        เปิดการติดต่อกับข้อมูล
            meter_datamodule.total_unit_Query.active:=false;
            meter_datamodule.total_unit_Query.active:=true;
                        ยกเลิกการติดต่อและเปิดข้อมูลเพื่อนำไปแสดงผล
            meter_datamodule.Table1.active:=false;
            meter_datamodule.Table1.active:=true;
                        ยกเลิกการติดต่อและเปิดข้อมูลของตารางทั้งหมด
end;
```

3:begin ส่วนการทำงานส่งหน่วยพลังงานรอบเดือน

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);
eng_nittw:=strtoint(buff);
sd:=sd+1;
strcopy(p,sd);
```

แยกข้อมูลหน่วยพลังงาน ช่วงเวลา 09.00-21.59 น. และเก็บที่ตัวแปร eng_nittw

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);
eng_otw:=strtoint(buff);
sd:=sd+1;
strcopy(p,sd);
```

แยกข้อมูลหน่วยพลังงาน ช่วงเวลาที่เหลือเก็บไว้ที่ตัวแปร eng_otw

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);
time_s:=buff;
sd:=sd+1;
strcopy(p,sd);
```

แยกข้อมูลเวลา วินาที เก็บไว้ที่ตัวแปร time_s

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);
time_m:=buff;
sd:=sd+1;
strcopy(p,sd);
```

แยกข้อมูลเวลา นาที เก็บไว้ที่ตัวแปร time_m

```
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);
time_h:=buff;
sd:=sd+1;
```

แยกข้อมูลเวลา ชั่วโมง เก็บไว้ที่ตัวแปร time_h

```
strcopy(p,sd);
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);          } แยกข้อมูล วันที่ เก็บไว้ที่ตัว
date_d:=buff;                } แปร date_d
sd:=sd+1;
strcopy(p,sd);
sd:=strscan(p,'-');
l:=strlen(p)-strlen(sd);
strlcopy(buff,p,l);          } แยกข้อมูล เดือน เก็บไว้ที่ตัว
date_m:=buff;                } แปร date_m
sd:=sd+1;
strcopy(p,sd);
date_y:=p;                   แยกข้อมูล ปี เก็บไว้ที่ตัวแปรdate_y
with meter_datamodule.meter_query do
                             อ้างถึงฐานข้อมูลมาตรวัด
begin
        cust_no_x := Lookup('tcpip',tcpip_x,'cust_no');
                             ค้นหาข้อมูลเมื่อหมายเลขผู้ใช้ไฟฟ้า cust_no จาก
                             ขอบเขตข้อมูล tcpip ที่มีหมายเลขแอดเดรส IP
                             ตรงกับตัวแปร tcpip_x
end;
with meter_datamodule.mount1_Query do
                             อ้างถึงฐานข้อมูลที่เก็บหน่วยพลังงานช่วงเวลา
                             09.00 – 21.59 น.
begin
        unit_1 := Lookup('cust_no',cust_no_x,date_m);
                             ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
                             เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
```

```
        edit;  แก้ไขข้อมูลหน่วยพลังงาน
        unit_1:=eng_nittw;
        FieldValues[date_m]:=unit_1;
               เก็บค่าของข้อมูล
end;
meter_datamodule.mount1_query.post;
               บันทึกข้อมูล
meter_datamodule.mount1_query.Active:=false;
               ยกเลิกการติดต่อกับข้อมูล
meter_datamodule.mount1_query.active:=true;
               เปิดการติดต่อกับข้อมูล
with meter_datamodule.mount2_Query do
               อ้างถึงฐานข้อมูลที่เก็บหน่วยพลังงานช่วงเวลา
               22.00 – 08.59 น.
begin
        unit_2 := Lookup('cust_no',cust_no_x,date_m);
               ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
               เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
        edit;  แก้ไขข้อมูลหน่วยพลังงาน
        unit_2:=eng_otw;
        FieldValues[date_m]:=unit_2;
               เก็บค่าของข้อมูล
end;
meter_datamodule.mount2_query.post;
               บันทึกข้อมูล
meter_datamodule.mount2_query.active:=false;
               ยกเลิกการติดต่อกับข้อมูล
meter_datamodule.mount2_query.active:=true;
               เปิดการติดต่อกับข้อมูล
```

```
                    with meter_datamodule.mount_Query do
                                อ้างถึงฐานข้อมูลหน่วยพลังงานรวม
                    begin
                        sum_unit:= Lookup('cust_no',cust_no_x,date_m);
                                ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกที่มีหมาย
                                เลขผู้ใช้ไฟฟ้าตรงกับตัวแปร cust_no
                        edit;  แก้ไขข้อมูลหน่วยพลังงาน
                        sum_unit:=unit_1+unit_2;
                                สะสมข้อมูลหน่วยพลังงานทุกช่วงเวลา
                        FieldValues[date_m]:=sum_unit;
                                เก็บค่าของข้อมูล
                    end;
                    meter_datamodule.mount_query.post;
                                บันทึกข้อมูล
                    meter_datamodule.mount_query.active:=false;
                                ยกเลิกการติดต่อกับข้อมูล
                    meter_datamodule.mount_query.active:=true;
                                เปิดการติดต่อกับข้อมูล
                    meter_datamodule.total_unit_Query.active:=false;
                    meter_datamodule.total_unit_Query.active:=true;
                                ยกเลิกการติดต่อและเปิดข้อมูลเพื่อนำไปแสดงผล
                end;
            end;
        end;
    end;
end;
procedure tFnetamr.serversocket1listen(sender:tobject;socket:tcustomwinsocket);
                                โพรซีเดอร์แสดงการรอรับการติดต่อจากมาตรวัด
begin
    echo('รอรับการติดต่อ... ');          แสดงข้อความ “รอรับการติดต่อ…”
end;
```

```
procedure tFnetamr.clientsocket1connect(sender:tobject;socket:tcustomwinsocket);
                                        โพรซีเดอร์แสดงการติดต่อกับมาตรวัด
begin
        echo('การติดต่อระบบสำเร็จ');     แสดงข้อความ “การติดต่อระบบสำเร็จ”
end;
proceduretFnetamr.clientsocket1error(sender:tobject;socket:tcustomwinsocket;errorevent:
terrorevent;var errorcode:integer);     โพรซีเดอร์แสดงการติดต่อกับมาตรวัดเกิดความ
                                        ผิดพลาด
begin
        echo(' เกิดความผิดพลาด ');       แสดงข้อความ “ เกิดความผิดพลาด”
end;
procedure tFnetamr.clientsocket1lookup(sender:tobject;socket:tcustomwinsocket);
                                        โพรซีเดอร์มองหามาตรวัดที่ต้องการติดต่อ
begin
echo('อยู่ระหว่างการติดต่อ...');        แสดงข้อความ “อยู่ระหว่างการติดต่อ…”
end;
procedure TFnetamr.BitBtn2Click(Sender: TObject);
                                        โพรซีเดอร์กลับสู่หน้าจอหลักของโปรแกรม
begin
        famrmain.show;                  แสดงหน้าจอหลักของโปรแกรม
        fnetamr.Hide;                   ซ่อนหน้าจอย่อยของการเชื่อมต่อ AMR
end;
procedure TFnetamr.BitBtn1Click(Sender: TObject);
                                        โพรซีเดอร์ยกเลิกการเชื่อมต่อระบบ
begin
        if MessageDlg('คุณต้องการออกจากการเชื่อมต่อ'),
        mtConfirmation, [mbYes, mbNo], 0) = mrYes then
                                        แสดงข้อความ “คุณต้องการออกจากการเชื่อม ต่อ
                                        ” ถ้าเลือกตอบ “ใช่” โปรแกรมจะเคลียร์หน้า
```

จอเชื่อมต่อ ให้อยู่ในสภาวะพร้อมใช้งาน และปิด
การติดต่อของโพรโตคอล TCP/IP

```
begin
        edit1.Enabled:=true;
        edit2.Enabled:=true;
        edit3.Enabled:=true;
        groupbox3.Enabled:=false;
        button1.Enabled:=true;
        memo1.lines.Clear;
        button2.enabled:=true;
        button3.enabled:=false;
        radiobutton1.Enabled:=false;        เคลียร์หน้าจอเชื่อมต่อระบบ
        radiobutton2.Enabled:=false;
        radiobutton3.Enabled:=false;
        serversocket1.Close;
        radiobutton1.Checked:=false;
        radiobutton2.Checked:=false;
        radiobutton3.Checked:=false;
        button4.Enabled:=false;
        edit2.Clear;
        clientsocket1.Close;            ปิดการติดต่อ
end;
end;
procedure TFnetamr.Button4Click(Sender:TObject);
```

โพรซีเดอร์ยกเลิกการติดต่อกับมาตรวัด

```
begin
        button3.enabled:=false;
        radiobutton1.Enabled:=false;        เคลียร์หน้าจอในส่วนของการติด
        radiobutton2.Enabled:=false;        ต่อกับมาตรวัด และปิดการติด
        radiobutton3.Enabled:=false;        ต่อกับมาตรวัด
```

```
        radiobutton1.Checked:=false;
        radiobutton2.Checked:=false;
        radiobutton3.Checked:=false;
        groupbox3.Enabled:=false;          เคลียร์หน้าจอในส่วนของการติด
        edit2.Enabled:=true;               ต่อกับมาตรวัด และปิดการติด
        edit3.Enabled:=true;               ต่อกับมาตรวัด
        button2.Enabled:=true;
        button4.Enabled:=false;
        edit2.Clear;
        clientsocket1.Close;               ปิดการติดต่อ
end;
procedure TFnetamr.ClientSocket1Disconnect(Sender: TObject;Socket:
TCustomWinSocket);                         โพรซีเดอร์แสดงมาตรวัดหยุดการติดต่อ
begin
        echo('หยุดการติดต่อ');             แสดงข้อความ “หยุดการติดต่อ”
end;
procedure TFnetamr.Button3Click(Sender: TObject);
                                           โพรซีเดอร์ส่งข้อมูลให้มาตรวัด
var
        s:shortstring;
        p:pchar;
        data_time:string;                  ประกาศตัวแปรเก็บข้อมูลเวลา
        data_date:string;                  ประกาศตัวแปรเก็บข้อมูลวันที่
        data_buf:string;                   ประกาศตัวแปรเก็บข้อมูลเตรียมส่ง
begin
        if status=1 then                   ถ้ารหัสส่วนการทำงานเป็นการตั้งเวลา ให้จัดรูป
                                           แบบข้อมูล “ ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี ”
                                           และส่งข้อมูลให้มาตรวัดพร้อมเคลียร์รหัสการ
                                           ทำงานเดิม ถ้าไม่ใช่ส่วนการตั้งเวลาให้ส่งเฉพาะ
                                           รหัสส่วนการทำงาน
```

```
        begin
                timeseparator:='-';
                data_time:=timetostr(now);
                dateseparator:='-';                          จัดรูปแบบของข้อมูล
                data_date:=datetostr(date);
                data_buf:=data_time+'-'+data_date;
                p:=@data_buf[1];
                clientsocket1.Socket.SendBuf(p^,length(data_buf));
                                                ส่งข้อมูลให้มาตรวัด
                button3.Enabled:=false;
                status:=0;                      เคลียร์รหัสส่วนการทำงาน
        end
else
        begin
                s:=inttostr(status);            เปลี่ยนรหัสส่วนจากตัวอักษรเป็นตัวเลข
                p:=@s[1];
                clientsocket1.Socket.SendBuf(p^,length(s));
                                                ส่งข้อมูลให้มาตรวัด
                button3.Enabled:=false;
                status:=0;                      เคลียร์รหัสส่วนการทำงาน
        end;
        button4.Click;
end;
procedure TFnetamr.RadioButton1Click(Sender: TObject);
                                                โพรซีเดอร์ตั้งเวลามาตรวัด
var
        s:shortstring;
        p:pchar;
```

```
begin
        status:=1;                              รหัสส่วนการทำงานตั้งเวลา
        s:=inttostr(status);                    แปลงข้อมูลตัวเลขเป็นตัวอักษร
        p:=@s[1];                               เก็บตำแหน่งของตัวแปร s
        clientsocket1.Socket.SendBuf(p^,length(s));
                                                ส่งข้อมูลรหัสการทำงานตั้งเวลา
        button3.enabled:=true;                  ปุ่มการส่งข้อมูลพร้อมรับคำสั่ง
end;
procedure TFnetamr.RadioButton2Click(Sender: TObject);
                                                โพรซีเดอร์ติดตามการใช้ไฟฟ้า
begin
        status:=2;                              รหัสส่วนการทำงานติดตามการใช้ไฟฟ้า
        button3.enabled:=true;                  ปุ่มการส่งข้อมูลพร้อมรับคำสั่ง
end;
procedure TFnetamr.RadioButton3Click(Sender: TObject);
                                                โพรซีเดอร์ส่งหน่วยพลังงานรอบเดือน
begin
        status:=3;                              รหัสส่วนการทำงานส่งหน่วยพลังงานรอบเดือน
        button3.enabled:=true;                  ปุ่มการส่งข้อมูลพร้อมรับคำสั่ง
end;
procedure TFnetamr.FormCreate(Sender: TObject);
                                                โพรซีเดอร์แสดงข้อมูลพร้อมกับการสร้างหน้าจอ
begin
    timer1timer(timer1);                        เวลาเริ่มทำงานและสร้างหน้าจอ
end;
end.
```

ยูนิตติดต่อผ่านสายอนุกรม

```
unit uTsCom32;
interface
uses
        Windows, Messages, SysUtils, Classes,
        Graphics, Controls,Forms, Dialogs,        } ส่วนการเชื่อมต่อกับระบบวินโดส์
        uComm32, StdCtrls, Buttons, ExtCtrls,     } ที่โปรแกรมเรียกใช้งาน
        Gauges;
type
        T1KBs = array[ 0..1023 ] of char;
        TFtsCom32 = class(TForm)
        BitBtn1: TBitBtn;
        Memo1: TMemo;
        RadioGroup1: TRadioGroup;
        RadioGroup2: TRadioGroup;
        BitBtn2: TBitBtn;
        BitBtn4: TBitBtn;                         } กำหนดประเภทให้กับวัตถุที่แสดง
        Panel1: TPanel;                           } บนหน้าจอของโปรแกรม
        Panel2: TPanel;
        Panel3: TPanel;
        Panel4: TPanel;
        OpenDialog1: TOpenDialog;
        Button1: TButton;
        Button2: TButton;
procedure FormCreate(Sender: TObject);        โพรซีเดอร์การสร้างหน้าจอและเคลียร์ค่าตัวแปร
procedure BitBtn1Click(Sender: TObject);      โพรซีเดอร์เปิดการสื่อสารพอร์ตอนุกรม
procedure BitBtn2Click(Sender: TObject);      โพรซีเดอร์ส่งคำสั่งให้มาตรวัด
procedure BitBtn4Click(Sender: TObject);      โพรซีเดอร์กำหนดชื่อแฟ้มข้อมูล
procedure Button1Click(Sender: TObject);      โพรซีเดอร์กลับสู่หน้าจอหลัก
```

```
procedure Button2Click(Sender: TObject);        โพรซีเดอร์ยกเลิกการติดต่อพอร์ตอนุกรม
private
        { Private declarations }
        commInfo : TcommInfo;                   ตัวแปรกำหนดค่าของพอร์ตอนุกรม
        rxCounter : longint;                    ตัวแปรสำหรับนับจำนวนข้อมูล
procedure Echo( content : shortString );        โพรซีเดอร์แสดงข้อมูล
function  GetBaudRateValue : integer;           ฟังก์ชันกำหนดอัตราการรับส่งข้อมูล
procedure Write2File( fileName : shortString;dataP : pointer; len:integer );
                                                โพรซีเดอร์บันทึกแฟ้มข้อมูล
public
        { Public declarations }
        commProc : Tcomm32;                     ตัวแปรกลางที่โปรแกรมเรียกใช้งาน
 procedure DoProcess;                           โพรซีเดอร์รับข้อมูล
end;
var
        FtsCom32: TFtsCom32;                    ตัวแปรของหน้าจอระบบการติดต่อผ่านสาย
                                                อนุกรม
implementation
uses uComProc,netamr;
{$R *.DFM}
procedure TFtsCom32.Echo( content : shortString );
                                                โพรซีเดอร์แสดงข้อมูล
begin
        if ( memo1.lines.count>30 ) then memo1.lines.delete(0);
                                                ถ้าแสดงข้อมูลเกิน 30 บรรทัดให้แสดงข้อมูลใหม่
        memo1.lines.add( content );
end;
```

```
function TFtsCom32.GetBaudRateValue : integer;
                                        ฟังก์ชันกำหนดอัตราการรับส่งข้อมูล
var
        brsel : integer;
begin
        brsel:=CBR_19200;               กำหนดให้พอร์ตอนุกรมมีอัตราการส่งเริ่มต้นที่
                                        19200 บิตต่อวินาที
        case RadioGroup2.ItemIndex of   เลือกความเร็วในการส่งข้อมูล
                0 : brsel:=CBR_2400;
                1 : brsel:=CBR_4800;
                2 : brsel:=CBR_9600;
                3 : brsel:=CBR_14400;
                4 : brsel:=CBR_19200;
                5 : brsel:=CBR_38400;
                6 : brsel:=CBR_56000;
                7 : brsel:=CBR_115200;
 end;
        result:=brsel;
procedure TFtsCom32.Write2File( fileName : shortString;dataP : pointer; len:integer );
                                        โพรซีเดอร์บันทึกแฟ้มข้อมูล
var
        fp : file;
        ret : integer;
begin
        if ( not fileExists( fileName ) ) then  ถ้าไม่มีชื่อแฟ้มข้อมูลให้กำหนดชื่อแฟ้มข้อมูล
        begin
                assignFile( fp, fileName ); rewrite( fp,1 );
        end
        else
```

```
        begin
                assignFile( fp, fileName ); reset( fp,1 );
        end;
        seek( fp, fileSize( fp ) );                 ตรวจสอบความยาวของแฟ้มข้อมูล
        blockWrite( fp, dataP^, len, ret );         บันทึกแฟ้มข้อมูล
        closeFile( fp );                            ปิดแฟ้มข้อมูล
        inc( rxCounter, len );                      เพิ่มและแสดงจำนวนข้อมูลที่รับได้
        Panel2.caption:=IntToStr( rxCounter );
end;
procedure TFtsCom32.DoProcess;                      โพรซีเดอร์รับข้อมูล
var
        data : T1KBs;
        dataP : pchar;                              ประกาศตัวแปรช่วยในการรับข้อมูล
        ret : integer;
begin
        if ( commProc=nil ) then exit;              ถ้าไม่มีข้อมูลให้ออกจากการสื่อสาร
        dataP:=@data[0];                            เคลียร์ค่าตัวแปร
        ret:=commProc.ReadData( dataP, 1024 );
                                                    รับข้อมูลจากพอร์ตอนุกรม
        if ( ret>0 ) then                           ถ้ามีการรับข้อมูลให้บันทึกแฟ้มข้อมูล
        begin
                Write2File( OpenDialog1.fileName, dataP, ret );
                                                    เรียกใช้โพรซีเดอร์บันทึกแฟ้มข้อมูล
        end;
end;
procedure TFtsCom32.FormCreate(Sender: TObject);
                                                    โพรซีเดอร์การสร้างหน้าจอและเคลียร์ค่าตัวแปร
begin
```

```
        commproc:=nil;
        rxCounter:=0;                    เคลียร์ค่าตัวแปร
        fillchar(commInfo,sizeof(TcommInfo'0);
end;
procedure TFtsCom32.BitBtn1Click(Sender: TObject);
                                         โพรซีเดอร์เปิดการสื่อสารพอร์ตอนุกรม
begin
        with commInfo do                 กำหนดค่าของพอร์ตอนุกรม
        begin
            commPort:=RadioGroup1.ItemIndex+1;
                                         กำหนดหมายเลขพอร์ต
            rxBuffer:=2048;              กำหนดความจำสำรองสำหรับรับข้อมูล
            txBuffer:=2048;              กำหนดความจำสำรองสำหรับส่งข้อมูล
            baudRate:=GetBaudRateValue;  กำหนดความเร็วในการสื่อสาร
            byteSize:=8;                 กำหนดความยาวของโพรโตคอล
            parity:=NOPARITY;            กำหนดขั้วของข้อมูล
            stopBits:=ONESTOPBIT;        กำหนดบิตหยุด
        end;
end;
procedure TFtsCom32.BitBtn2Click(Sender: TObject);
                                         โพรซีเดอร์ส่งคำสั่งให้มาตรวัด
var
        s : shortString;
        sP : pChar;
begin
        if ( commProc=nil ) then exit;   ถ้าไม่มีข้อมูลให้ออกจากการสื่อสาร
        s:='11';                         กำหนดคำสั่งเป็น "11"
        sP:=@s[1];
        commProc.SendData( sP, length( s ) );  ส่งข้อมูลให้พอร์ตอนุกรม
end;
```

```
procedure TFtsCom32.BitBtn4Click(Sender: TObject);
                                        โพรซีเดอร์กำหนดชื่อแฟ้มข้อมูล
begin
    if ( commProc=nil ) then exit;      ถ้าไม่มีข้อมูลให้ออกจากการสื่อสาร
    if ( OpenDialog1.Execute ) then     เปิดหน้าจอการบันทึกแฟ้มข้อมูล
    begin
        rxCounter:=0;
        Echo( 'บันทึกแฟ้มเป็น>'+OpenDialog1.fileName );
    end;
end;
procedure TFtsCom32.Button1Click(Sender: TObject);
                                        โพรซีเดอร์กลับสู่หน้าจอหลัก
begin
    ftsCom32.hide;                      ซ่อนหน้าจอระบบการติดต่อผ่านสายอนุกรม
    fnetamr.show;                       แสดงหน้าจอเชื่อมต่อระบบ
end;
procedure TFtsCom32.Button2Click(Sender: TObject);
                                        โพรซีเดอร์ยกเลิกการติดต่อพอร์ตอนุกรม
begin
    if (commProc<>nil) then
    commProc.Free;                      เคลียร์ค่าตัวแปร
end;
end.
```

ยูนิตติดตามการใช้ไฟฟ้า

```
unit mon;
interface
uses
    Windows, Messages, SysUtils, Classes,
    Graphics, Controls, Forms, Dialogs,
    ExtCtrls, TeeProcs, TeEngine, Chart,
    DBChart, StdCtrls, Buttons, DBCtrls,
    Mask, Series;
type
    TFmon = class(TForm)
    DBChart1: TDBChart;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    DBEdit2: TDBEdit;
    DBEdit4: TDBEdit;
    DBNavigator1: TDBNavigator;
    Label4: TLabel;
    DBText1: TDBText;
    Series1: TBarSeries;
    Label5: TLabel;
    DBEdit5: TDBEdit;
    Label6: TLabel;
    Label7: TLabel;
```

ส่วนการเชื่อมต่อกับระบบวินโดวส์ที่โปรแกรมเรียกใช้งาน

กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
        label8: TLabel;
        Edit1: TEdit;
        Edit2: TEdit;
        Edit3: TEdit;
```
} กำหนดประเภทให้กับวัตถุที่แสดงบนหน้าจอของโปรแกรม

```
procedure BitBtn1Click(Sender: TObject);
```
โพรซีเดอร์กลับสู่หน้าจอหลัก

```
procedure DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
```
โพรซีเดอร์แสดงข้อมูลกราฟตามฐานข้อมูล

```
procedure FormCreate(Sender: TObject);
```
โพรซีเดอร์แสดงข้อมูลพร้อมกับการสร้างหน้าจอ

```
procedure RadioButton1Click(Sender: TObject);
```
โพรซีเดอร์แสดงค่าไฟฟ้า TOD น้อยกว่า 150 หน่วย

```
procedure RadioButton2Click(Sender: TObject);
```
โพรซีเดอร์แสดงค่าไฟฟ้า TOD มากกว่า 150 หน่วย

```
procedure RadioButton3Click(Sender: TObject);
```
โพรซีเดอร์แสดงค่าไฟฟ้า TOU

```
private
        { Private declarations }
        procedure con_dbtext(var cust_x:string);
```
โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง

```
        procedure clearedit;
```
โพรซีเดอร์เคลียร์ส่วนการคำนวณค่าไฟฟ้า

```
        procedure tod_less(unit_t:integer;var cost_x:real);
```
โพรซีเดอร์คำนวณค่าไฟฟ้า TOD น้อยกว่า 150 หน่วย

```
        procedure tod_more(unit_t:integer;var cost_x:real);
```
โพรซีเดอร์คำนวณค่าไฟฟ้า TOD มากกว่า 150 หน่วย

```
        procedure tou(cust_c:integer;mount_x:string;var cost_x:real);
```
โพรซีเดอร์คำนวณค่าไฟฟ้า TOU

```
public
        { Public declarations }
end;
var
```

```
        Fmon: TFmon;                    ตัวแปรของหน้าจอติดตามการใช้พลังงาน
        cust_x:string;                  ตัวแปรกลางที่โปรแกรมใช้งาน
        cust_b:string;
implementation
uses amrmain,meter_module,centralvar;
{$R *.DFM}
procedure TFmon.BitBtn1Click(Sender: TObject);
                                        โพรซีเดอร์กลับสู่หน้าจอหลัก
begin
        famrmain.show;                  แสดงหน้าจอหลักของโปรแกรม
        fmon.Hide;                      ซ่อนหน้าจอติดตามการใช้พลังงาน
end;
procedure TFmon.FormCreate(Sender: TObject);
                                        โพรซีเดอร์แสดงข้อมูลพร้อมกับการสร้างหน้าจอ
var
        table_name:string;
begin
        label8.Visible:=false;
        con_dbtext(cust_b);             เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง
        table_name:='t_cust_h'+cust_b;
        meter_datamodule.table1.tablename:=table_name;
                                        อ้างถึงมอดูลข้อมูลตารางชื่อ “t_cust_h หมายเลขผู้
                                        ใช้ไฟฟ้า”
        if not meter_datamodule.table1.Exists then
                                        ถ้าไม่มีตารางในมอดูลข้อมูลให้แสดงข้อความ
                                        “ไม่มีข้อมูลติดตามการใช้พลังงาน”
        begin
                label8.Caption:='ไม่มีข้อมูลติดตามการใช้พลังงาน';
                label8.Visible:=true;
        end
```

```
        else                                    ถ้ามีตารางในมอดูลข้อมูลให้ติดต่อข้อมูลและ
                                                แสดงกราฟ
        begin
                meter_datamodule.table1.active:=true;
                                                ติดต่อตารางข้อมูล
                series1.Active:=true;           แสดงกราฟข้อมูล
        end;
end;
procedure tfmon.clearedit;                      โพรซีเดอร์เคลียร์ส่วนการคำนวณค่าไฟฟ้า
begin
        radiobutton1.Checked:=false;
        radiobutton2.Checked:=false;            } เคลียร์ปุ่มการสั่งงาน
        radiobutton3.Checked:=false;
        edit1.Text:=' ';
        edit2.Text:=' ';                        } เคลียร์ส่วนการแสดงผลการคำนวณ
        edit3.Text:=' ';
end;
procedure tfmon.con_dbtext(var cust_x:string);
                                                โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง
var
        Buffer: PChar;                          ประกาศตัวแปรช่วยในการแปลงข้อมูล
        Size: Byte;
begin
        Size := dbtext1.getTextLen;             หาความยาวของข้อมูล
        Inc(Size);                              เพิ่มความยาวข้อมูลอีกหนึ่งตัวอักษร
        GetMem(Buffer, Size);                   จองหน่วยความจำที่จะใช้งาน
        dbtext1.GetTextBuf(Buffer,Size);        เก็บข้อมูลตามความยาวเข้าสู่หน่วยความจำ
        cust_x := StrPas(Buffer);               แปลงข้อมูลเป็นแบบสตริง
        FreeMem(Buffer, Size);                  เคลียร์หน่วยความจำ
end;
```

```
procedure TFmon.DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
                                        โพรซีเดอร์แสดงข้อมูลกราฟตามฐานข้อมูล
var
        table_name:string;
begin
        case Button of                  เลือกกรณีการทำงานของปุ่มนำทางข้อมูล
                nbFirst:                รายการบันทึกข้อมูลแรก
                 begin
                        meter_datamodule.table1.active:=false;
                                        ปิดการติดต่อตารางข้อมูล
                        label8.Visible:=false;
                        con_dbtext(cust_b);     เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นสตริง
                        table_name:='t_cust_h'+cust_b;
                        meter_datamodule.table1.tablename:=table_name;
                                        อ้างถึงมอดูลข้อมูลตารางชื่อ “t_cust_h หมายเลขผู้
                                        ใช้ไฟ”
                        if not meter_datamodule.table1.Exists then
                                        ถ้าไม่มีตารางในมอดูลข้อมูลให้แสดงข้อความ
                                        “ไม่มีข้อมูลติดตามการใช้พลังงาน”
                        begin
                                label8.Caption:='ไม่มีข้อมูลติดตามการใช้พลังงาน';
                                label8.Visible:=true;
                        end
                        else            ถ้ามีตารางในมอดูลข้อมูลให้ติดต่อข้อมูลและ
                                        แสดงกราฟ
                        begin
                                meter_datamodule.table1.active:=true;
                                        ติดต่อตารางข้อมูล
                                series1.Active:=true;
                                        แสดงกราฟข้อมูล
                        end;
```

```
        clearedit;
end;
nbPrior:                    รายการบันทึกข้อมูลก่อนหน้า
begin
        meter_datamodule.table1.active:=false;
                            ปิดการติดต่อตารางข้อมูล
        label8.Visible:=false;
        con_dbtext(cust_b);  เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นสตริง
        table_name:='t_cust_h'+cust_b;
        meter_datamodule.table1.tablename:=table_name;
                            อ้างถึงมอดูลข้อมูลตารางชื่อ “t_cust_h หมายเลขผู้
                            ใช้ไฟ”
        if not meter_datamodule.table1.Exists then
                            ถ้าไม่มีตารางในมอดูลข้อมูลให้แสดงข้อความ
                            “ไม่มีข้อมูลติดตามการใช้พลังงาน”
        begin
                label8.Caption:='ไม่มีข้อมูลติดตามการใช้พลังงาน';
                label8.Visible:=true;
        end
        else                ถ้ามีตารางในมอดูลข้อมูลให้ติดต่อข้อมูลและ
                            แสดงกราฟ
        begin
                meter_datamodule.table1.active:=true;
                            ติดต่อตารางข้อมูล
                series1.Active:=true;
                            แสดงกราฟข้อมูล
        end;
        clearedit;
end;
```

```
nbNext:                          รายการบันทึกข้อมูลต่อไป
begin
        meter_datamodule.table1.active:=false;
                                 ปิดการติดต่อตารางข้อมูล
        label8.Visible:=false;
        con_dbtext(cust_b);      เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นสตริง
        table_name:='t_cust_h'+cust_b;
        meter_datamodule.table1.tablename:=table_name;
                                 อ้างถึงมอดูลข้อมูลตารางชื่อ “t_cust_h หมายเลขผู้
                                 ใช้ไฟ”
        if not meter_datamodule.table1.Exists then
                                 ถ้าไม่มีตารางในมอดูลข้อมูลให้แสดงข้อความ
                                 “ไม่มีข้อมูลติดตามการใช้พลังงาน”
        begin
                label8.Caption:='ไม่มีข้อมูลติดตามการใช้พลังงาน';
                label8.Visible:=true;
        end
        else                     ถ้ามีตารางในมอดูลข้อมูลให้ติดต่อข้อมูลและ
                                 แสดงกราฟ
        begin
                meter_datamodule.table1.active:=true;
                                 ติดต่อตารางข้อมูล
                series1.Active:=true;
                                 แสดงกราฟข้อมูล
        end;
        clearedit;
end;
nbLast:                          รายการบันทึกข้อมูลสุดท้าย
```

```
            begin
                meter_datamodule.table1.active:=false;
                                        ปิดการติดต่อตารางข้อมูล
                label8.Visible:=false;
                con_dbtext(cust_b);     เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นสตริง
                table_name:='t_cust_h'+cust_b;
                meter_datamodule.table1.tablename:=table_name;
                                        อ้างถึงมอดูลข้อมูลตารางชื่อ “t_cust_h หมายเลขผู้
                                        ใช้ไฟ”
                if not meter_datamodule.table1.Exists then
                                        ถ้าไม่มีตารางในมอดูลข้อมูลให้แสดงข้อความ
                                        “ไม่มีข้อมูลติดตามการใช้พลังงาน”
                begin
                    label8.Caption:='ไม่มีข้อมูลติดตามการใช้พลังงาน';
                    label8.Visible:=true;
                end
                else                    ถ้ามีตารางในมอดูลข้อมูลให้ติดต่อข้อมูลและ
                                        แสดงกราฟ
                begin
                    meter_datamodule.table1.active:=true;
                                        ติดต่อตารางข้อมูล
                    series1.Active:=true;
                                        แสดงกราฟข้อมูล
                end;
                clearedit;
            end;
        end;
end;
```

```
procedure TFmon.tod_less(unit_t:integer;var cost_x:real);
                                        โพรซีเดอร์คำนวณค่าไฟฟ้า TOD น้อยกว่า 150
                                        หน่วย
var
        unit_x:integer;
        cost:Variant;
begin
        cost_x:=0;
        if unit_t > 5 then              ถ้าหน่วยพลังงานไฟฟ้าน้อยกว่า 5 หน่วย คิด
                                        เฉพาะค่าบริการ
        begin
                if (unit_t > 5) and (unit_t <= 15) then
                                        คำนวณค่าไฟฟ้าถ้าหน่วยพลังงานระหว่าง 6 ถึง 15
                                        หน่วย
                begin
                        unit_x:=unit_t-5;       5 หน่วยแรกคิดเฉพาะค่าบริการ
                        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
                        cost_x:=cost*unit_x;    คำนวณค่าไฟฟ้าหน่วยที่เหลือ
                end;
                if (unit_t > 15) and (unit_t <= 25) then
                                        คำนวณค่าไฟฟ้าถ้าหน่วยพลังงานระหว่าง 16 ถึง
                                        25 หน่วย
                begin
                        unit_x:=unit_t-15;
                        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
                        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
                        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
```

```
        cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
end;
if (unit_t > 25) and (unit_t <= 35) then
begin
        unit_x:=unit_t-25;
        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[4].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 26 ถึง 35 หน่วย
        cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
end;
if (unit_t > 35) and (unit_t <= 100) then
begin
        unit_x:=unit_t-35;
        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[4].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 26 ถึง 35 หน่วย
```

```
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[5].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 36 ถึง 10 หน่วย
        cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
end;
if (unit_t > 100) and (unit_t <= 150) then
begin
        unit_x:=unit_t-100;
        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[4].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 26 ถึง 35 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[5].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 36 ถึง 100
                                หน่วย
        cost_x:=cost_x+(cost*65);
                                คำนวณค่าไฟฟ้า 65 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[6].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 101 ถึง 150
                                หน่วย
```

```
        cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
end;
if (unit_t > 150) and (unit_t <= 400) then
begin
        unit_x:=unit_t-150;
        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[4].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 26 ถึง 35 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[5].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 36 ถึง 100
                                หน่วย
        cost_x:=cost_x+(cost*65);
                                คำนวณค่าไฟฟ้า 65 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[6].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 101 ถึง 150
                                หน่วย
        cost_x:=cost_x+(cost*50);
                                คำนวณค่าไฟฟ้า 50 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[7].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 151 ถึง 400
                                หน่วย
```

```
        cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
end;
if unit_t > 400 then
begin
        unit_x:=unit_t-400;
        cost:= meter_datamodule.cost_Table.Fields[2].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 6 ถึง 15 หน่วย
        cost_x:=cost*10;        คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[3].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 16 ถึง 25 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[4].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 26 ถึง 35 หน่วย
        cost_x:=cost_x+(cost*10);
                                คำนวณค่าไฟฟ้า 10 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[5].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 36 ถึง 100
                                หน่วย
        cost_x:=cost_x+(cost*65);
                                คำนวณค่าไฟฟ้า 65 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[6].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 101 ถึง 150
                                หน่วย
        cost_x:=cost_x+(cost*50);
                                คำนวณค่าไฟฟ้า 50 หน่วย
        cost:= meter_datamodule.cost_Table.Fields[7].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 151 ถึง 400
                                หน่วย
```

```
                cost_x:=cost_x+(cost*250);
                                คำนวณค่าไฟฟ้า 250 หน่วย
                cost:= meter_datamodule.cost_Table.Fields[8].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วงมากกว่า 400
                                หน่วย
                cost_x:=cost_x+(cost*unit_x);
                                คำนวณค่าไฟฟ้าหน่วยที่เหลือ
            end;
        end;
        cost:= meter_datamodule.cost_Table.Fields[0].value;
                                อ้างถึงอัตราค่าบริการ TOD น้อยกว่า 150 หน่วย
        cost_x:=cost_x+cost;    รวมค่าไฟฟ้า
end;
procedure TFmon.tod_more(unit_t:integer;var cost_x:real);
                                โพรซีเดอร์คำนวณค่าไฟฟ้า TOD มากกว่า 150
                                หน่วย
var
        unit_x:integer;
        cost:Variant;
begin
        cost_x:=0;
        if unit_t <= 150 then
        begin
            cost:= meter_datamodule.cost_Table.Fields[10].value;
                                อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 0 ถึง 150 หน่วย
            cost_x:=cost*unit_t;    คำนวณค่าไฟฟ้าหน่วยที่เหลือ
        end;
        if (unit_t > 150) and (unit_t <= 400) then
        begin
            unit_x:=unit_t-150;
```

```
                cost:= meter_datamodule.cost_Table.Fields[10].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 0 ถึง 150 หน่วย
                cost_x:=cost*150;       คำนวณค่าไฟฟ้า 150 หน่วย
                cost:= meter_datamodule.cost_Table.Fields[11].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 151 ถึง 400
                                        หน่วย
                cost_x:=cost_x+(cost*unit_x); คำนวณค่าไฟฟ้าหน่วยที่เหลือ
        end;
        if (unit_t > 400)then
        begin
                unit_x:=unit_t-400;
                cost:= meter_datamodule.cost_Table.Fields[10].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 0 ถึง 150 หน่วย
                cost_x:=cost*150;       คำนวณค่าไฟฟ้า 150 หน่วย
                cost:= meter_datamodule.cost_Table.Fields[11].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วง 151 ถึง 400
                                        หน่วย
                cost_x:=cost_x+(cost*250);  คำนวณค่าไฟฟ้า 250 หน่วย
                cost:= meter_datamodule.cost_Table.Fields[12].value;
                                        อ้างถึงอัตราค่าพลังงานไฟฟ้าช่วงมากกว่า 400
                                        หน่วย
                cost_x:=cost_x+(cost*unit_x); คำนวณค่าไฟฟ้าหน่วยที่เหลือ
        end;
        cost:= meter_datamodule.cost_Table.Fields[9].value;
                                        อ้างถึงอัตราค่าบริการ TOD มากกว่า 150 หน่วย
cost_x:=cost_x+cost;                    รวมค่าไฟฟ้า
end;
```

```
procedure TFmon.tou(cust_c:integer;mount_x:string;var cost_x:real);
                                        โพรซีเดอร์คำนวณค่าไฟฟ้า TOU
var
        unit_1:integer;
        unit_2:integer;
        cost:Variant;
begin
        unit_1:= meter_datamodule.mount1_Query.Lookup('cust_no',cust_c,'mount_'+mount_x);
                                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                        เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        cost:= meter_datamodule.cost_Table.Fields[14].value;
                                        อ้างถึงข้อมูลอัตราค่าไฟฟ้า TOU ช่วงเวลา 09.00 –
                                        21.59 น.
        cost_x:=unit_1*cost;            คำนวณค่าไฟฟ้าช่วงเวลาดังกล่าว
        unit_2:= meter_datamodule.mount2_Query.Lookup('cust_no',cust_c,'mount_'+mount_x);
                                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                        เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        cost:= meter_datamodule.cost_Table.Fields[15].value;
                                        อ้างถึงข้อมูลอัตราค่าไฟฟ้า TOU ช่วงเวลา 22.00 –
                                        08.59 น.
        cost_x:=cost_x+(unit_2*cost);   คำนวณค่าไฟฟ้าช่วงเวลาดังกล่าว
        cost:= meter_datamodule.cost_Table.Fields[13].value;
                                        อ้างถึงข้อมูลค่าบริการ TOU
        cost_x:=cost_x+cost;            รวมค่าไฟฟ้า
end;
procedure TFmon.RadioButton1Click(Sender: TObject);
                                        โพรซีเดอร์แสดงค่าไฟฟ้า TOD น้อยกว่า 150
                                        หน่วย
var
```

```
        unit_t:integer;
        cost_x:real;
        cust_c:integer;                    } ประกาศตัวแปรช่วยในการแสดงผล
        mount_x:integer;
        pr_mount:string;
begin
        con_dbtext(cust_b);                เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง
        cust_c:=strtoint(cust_b);          แปลงข้อมูลสตริงเป็นตัวเลข
unit_t:= meter_datamodule.mount_Query.Lookup('cust_no',cust_c,'mount_'+date_mserv);
                                           ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                           เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        edit3.Text:=inttostr(unit_t);      แสดงหน่วยพลังงานไฟฟ้า
        tod_less(unit_t,cost_x);           เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOD น้อยกว่า
                                           150 หน่วย
        edit1.Text:=currtostr(cost_x);     แสดงค่าไฟฟ้าเดือนปัจจุบัน
        mount_x:=strtoint(date_mserv);
        if mount_x = 1 then                ถ้าเป็นเดือนมกราคมข้อมูลเดือนก่อนหน้าเป็น
                                           เดือนธันวาคม
                mount_x:=12
        else                               ถ้าเป็นเดือนอื่น ให้กลับไปเดือนก่อนหน้า
                mount_x:=mount_x-1;
        pr_mount:=inttostr(mount_x);
        unit_t:= meter_datamodule.mount_Query.Lookup('cust_no',cust_c,'mount_'+pr_mount);
                                           ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                           เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        tod_less(unit_t,cost_x);           เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOD น้อยกว่า
                                           150 หน่วย
        edit2.Text:=currtostr(cost_x);     แสดงค่าไฟฟ้าเดือนก่อน
end;
```

```
procedure TFmon.RadioButton2Click(Sender: TObject);
                                        โพรซีเดอร์แสดงค่าไฟฟ้า TOD มากกว่า 150
                                        หน่วย
var     unit_t:integer;
        cost_x:real;
        cust_c:integer;                 ประกาศตัวแปรช่วยในการแสดงผล
        mount_x:integer;
        pr_mount:string;
begin
        con_dbtext(cust_b);             เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง
        cust_c:=strtoint(cust_b);       แปลงข้อมูลสตริงเป็นตัวเลข
unit_t:= meter_datamodule.mount_Query.Lookup('cust_no',cust_c,'mount_'+date_mserv);
                                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                        เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        edit3.Text:=inttostr(unit_t);   แสดงหน่วยพลังงานไฟฟ้า
        tod_more(unit_t,cost_x);        เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOD มากกว่า
                                        150 หน่วย
        edit1.Text:=currtostr(cost_x);  แสดงค่าไฟฟ้าเดือนปัจจุบัน
        mount_x:=strtoint(date_mserv);
        if mount_x = 1 then             ถ้าเป็นเดือนมกราคมข้อมูลเดือนก่อนหน้าเป็น
                                        เดือนธันวาคม
                mount_x:=12
        else                            ถ้าเป็นเดือนอื่น ให้กลับไปเดือนก่อนหน้า
        pr_mount:=inttostr(mount_x);
        unit_t:= meter_datamodule.mount_Query.Lookup('cust_no',cust_c,'mount_'+pr_mount);
                                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                        เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        tod_more(unit_t,cost_x);        เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOD มากกว่า
                                        150 หน่วย
        edit2.Text:=currtostr(cost_x);  แสดงค่าไฟฟ้าเดือนก่อน
end;
```

```
procedure TFmon.RadioButton3Click(Sender: TObject);
                                        โพรซีเดอร์แสดงค่าไฟฟ้า TOU
var
        unit_t:integer;
        cost_x:real;
        cust_c:integer;                 ประกาศตัวแปรช่วยในการแสดงผล
        mount_x:integer;
        pr_mount:string;
begin
        cost_x:=0;
        con_dbtext(cust_b);             เรียกใช้โพรซีเดอร์แปลงข้อมูลเป็นแบบสตริง
        cust_c:=strtoint(cust_b);       แปลงข้อมูลสตริงเป็นตัวเลข
unit_t:= meter_datamodule.mount_Query.Lookup('cust_no',cust_c,'mount_'+date_mserv);
                                        ค้นหาข้อมูลเมื่อเดือนจากรายการบันทึกมีหมาย
                                        เลขผู้ใช้ไฟตรงกับตัวแปร cust_no
        edit3.Text:=inttostr(unit_t);   แสดงหน่วยพลังงานไฟฟ้า
        tou(cust_c,date_mserv,cost_x);  เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOU
        edit1.Text:=currtostr(cost_x);  แสดงค่าไฟฟ้าเดือนปัจจุบัน
        mount_x:=strtoint(date_mserv);
        if mount_x = 1 then             ถ้าเป็นเดือนมกราคมข้อมูลเดือนก่อนหน้าเป็น
                                        เดือนธันวาคม
                mount_x:=12
        else                            ถ้าเป็นเดือนอื่น ให้กลับไปเดือนก่อนหน้า
                mount_x:=mount_x-1;
        pr_mount:=inttostr(mount_x);
        tou(cust_c,pr_mount,cost_x);    เรียกใช้โพรซีเดอร์คำนวณค่าไฟฟ้า TOU
        edit2.Text:=currtostr(cost_x);  แสดงค่าไฟฟ้าเดือนก่อน
end;
end.
```

ภาคผนวก ข.

โปรแกรมรับส่งข้อมูลสำหรับไมโครคอนโทรลเลอร์แรบบิท 2000

โปรแกรมรับส่งข้อมูลสำหรับไมโครคอนโทรลเลอร์แรบบิท 2000

(ผู้พัฒนา นายศักดิ์ชัย ไวยลาภ, 2546)

#define MY_IP_ADDRESS "192.168.0.2"	กำหนดหมายเลขแอดเดรส IP ของมาตรวัดพลังงานไฟฟ้า
#define MY_NETMASK "255.255.255.0"	กำหนดหมายเลขเครือข่ายย่อย
#define ADDRESS"192.168.0.1"	กำหนดหมายเลขแอดเดรส IP ของเครื่องเซิร์ฟเวอร์ที่ต้องการติดต่อ
#define PORT_L 23	กำหนดหมายเลขช่องทาง ในการรอรับข้อมูลทางด้านซอฟแวร์
#define PORT_O 24	กำหนดหมายเลขช่องทาง ในการส่ง ข้อมูลทางด้านซอฟแวร์
#define EEPROM_ADDRESS 0xA0	กำหนดหมายเลขแอดเดรสให้หน่วยความจำ
#define Rev 1200	กำหนดจำนวนรอบในการหมุนของมาตรวัด
#define WRITE_TIME 10	กำหนดเวลาสำหรับการเขียนข้อมูลบนหน่วยความจำ
#define DINBUFSIZE 63	กำหนดขนาดของหน่วยความจำสำรองให้กับพอร์ตอนุกรมดี
#define DOUTBUFSIZE 63	กำหนดขนาดของหน่วยความจำสำรองให้กับพอร์ตอนุกรมดี
#define TIMEOUT 20UL	กำหนดเวลาที่ใช้ในการติดต่อ
#use "dcrtcp.lib"	เรียกใช้ไลบารีของ TCP/IP
#use "i2c_devices.lib"	เรียกใช้ไลบารีของ I^2C
#define FS_RAM #use "filesystem.lib" #define FORMAT #define RESERVE 0L #define BLOCKS 50 #define TESTFILE 1	เรียกใช้ไลบารีของการสร้างแฟ้มข้อมูลและกำหนดรูปแบบของแฟ้มข้อมูลที่ใช้ในการเก็บข้อมูล
int tcp_estab_mon();	ฟังก์ชันติดตามการใช้ไฟฟ้า

int tcp_estab_eng();	ฟังก์ชันส่งหน่วยพลังงานในรอบเดือน
char* tcp_xrec();	ฟังก์ชันที่ใช้ในการรับข้อมูล
tcp_xsend(char* data_xsend);	ฟังก์ชันที่ใช้ในการส่งข้อมูล
set_time();	ฟังก์ชันปรับตั้งเวลาของมาตรวัด
char* read_date_x();	ฟังก์ชันอ่านค่า วัน-เดือน-ปี
char* read_time_x();	ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที
char* date_time();	ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี
char* databuf(int eng_1,int eng_2);	ฟังก์ชันเตรียมข้อมูลที่ใช้ส่ง
monitor_x();	ฟังก์ชันช่วยในการติดตามการใช้ไฟฟ้า
energy_mon(int old_mon);	ฟังก์ชันช่วยในการส่งหน่วยพลังงานในรอบเดือน
int eng_sum();	ฟังก์ชันที่ใช้ในการนับรอบการหมุนของมาตรวัด
store_eng();	ฟังก์ชันที่ใช้เก็บหน่วยของพลังงานในแต่ละช่วงเวลา
stopserD();	ฟังก์ชันที่ใช้ในการหยุดส่งข้อมูลจากพอร์ตอนุกรมดี
char msg[DINBUFSIZE];	ประกาศตัวแปรที่ใช้เก็บข้อมูลของพอร์ตอนุกรมดี
longword host; tcp_Socket echosock;	โปรแกรมมาโคร ที่ช่วยในการเปิดและปิดช่องทางในการสื่อสาร
struct tm rtc;	ประกาศตัวแปรของเวลาเป็นแบบโครงสร้าง
unsigned long time_a;	ประกาศตัวแปรช่วยเก็บค่าเวลา
int stats; int sta_buf;	ประกาศตัวแปรที่ช่วยในส่วนการทำงานของโปรแกรมหลัก
int sta_mon;	ประกาศตัวแปรที่ช่วยในวงจรวนซ้ำเพื่อรอรับการติดต่อจากเวิร์ฟเวอร์ในขณะที่โปรแกรมทำงานตามคำสั่งหลัก
int eng;	ประกาศตัวแปรที่ช่วยในการเก็บหน่วยพลังงาน
int eng_rev;	ประกาศตัวแปรที่ใช้เก็บจำนวนรอบในการหมุนของมาตรวัด

```
int eng_nitottwo;
int eng_zetotfour;                          ประกาศตัวแปรที่ใช้เก็บหน่วยพลังงาน
int mon;                                    ประกาศตัวแปรช่วยในส่วนของการส่งหน่วย
                                            พลังงานในรอบเดือน
unsigned long old_mon;                      ประกาศตัวแปรที่ใช้เก็บค่าของเดือนเดิม เพื่อ
                                            ใช้ในการเปรียบเทียบ
File file;                                  ประกาศตัวแปรช่วยในการทำงานของระบบ
                                            แฟ้มข้อมูล
static char bufferm[2048];                  ประกาศตัวแปรช่วยในการทำงานของระบบ
                                            แฟ้มข้อมูล
static int wd;                              ประกาศตัวแปรช่วยในการทำงานของระบบ
                                            วอตซ์ด็อก
main()
{
eng=0;
mon=0;
stats=0;
sta_buf=0;                                  เคลียร์ค่าตัวแปรก่อนการใช้งาน
sta_mon=0;
eng_rev=0;
eng_nitottwo=0;
eng_zetotfour=0;
WrPortI(PEDDR, &PEDDRShadow, 0x00);         กำหนดให้พอร์ตอีเป็นช่องทางในการรับข้อมูล
SetVectExtern2000(1,eng_sum);               เกิดการอินเตอร์รัฟให้เรียกใช้ฟังก์ชันนับรอบ
                                            การหมุนของมาตรวัด
WrPortI(I0CR, NULL, 0x2B);                  พอร์ต อี เริ่มทำงานด้วยการตรวจจับขอบขาขึ้น
WrPortI(I1CR, NULL, 0x2B);                  และขอบขาลงของอินพุต
```

```
I2CRead(EEPROM_ADDRESS,0,statbuff,stats_x);          อ่านค่าสภาวะการทำงานเดิม
while(MS_TIMER - t < WRITE_TIME);                    ที่ค้างในหน่วยความจำ
stats=atoi(statbuff);
I2CRead(EEPROM_ADDRESS,1,eng_ni_buff,store_x);       อ่านค่าหน่วยพลังงาน ก. เดิม
while(MS_TIMER - t < WRITE_TIME);                    จากหน่วยความจำ
eng_nitottwo=atoi(eng_ni_buff);
I2CRead(EEPROM_ADDRESS,4,eng_ze_buff,store_x);       อ่านค่าหน่วยพลังงาน ข. เดิม
while(MS_TIMER - t < WRITE_TIME);                    จากหน่วยความจำ
eng_zetotfour=atoi(eng_ze_buff);
#ifdef FORMAT
        fs_format(RESERVE,BLOCKS,0);
        if(fcreate(&file,TESTFILE));                 สร้างรูปแบบและค่าเริ่มต้น
#else                                                ของแฟ้มข้อมูลที่ใช้สำหรับ
        fs_init(RESERVE,BLOCKS);                     บันทึกข้อมูล
        if(fopen_wr(&file,TESTFILE));
#endif
serDopen(19200);                                     เปิดพอร์ตอนุกรมดี
sock_init();
host=resolve(ADDRESS);                               เปลี่ยนหมายเลขแอดเดรสจากตัวอักษรให้เป็น
                                                     ตัวเลข
loop1:
tcp_listen(&echosock,PORT_L,host,0,NULL,0);          รอรับการติดต่อจากเซิร์ฟเวอร์
loop2:
stats=atoi(tcp_xrec());                              แปลงข้อมูลที่รับได้จากตัวอักษรเป็นตัวเลข
while (stats!=0)                                     วงจรวนซ้ำของโปรแกรมหลัก
        {
        loop3:
        statxr=stats;                                บันทึกค่าสภาวะในการทำงาน
        I2CWrite(EEPROM_ADDRESS,statbuff,stats_x);   ลงหน่วยความจำ
        while(MS_TIMER - t < WRITE_TIME);
```

```
switch (stats)
        {
        case 1:                         ส่วนของการปรับตั้งเวลา
                {
                set_time();             เรียกใช้ฟังก์ชันปรับตั้งเวลาของมาตรวัด
                if ((sta_buf!=0)&&(sta_buf!=1))
                        {               ถ้ามีการทำงานกับส่วนงานอื่นอยู่ก่อนเมื่อปรับ
                                        ตั้งเวลาของมาตรวัดแล้วให้กลับไปทำงานที่
                                        ส่วนงานเดิม
                        stats=sta_buf;
                        goto loop3;
                        }
                        else
                        {               ถ้าไม่มีส่วนงานเดิมให้ไปรอรับการติดต่อจาก
                                        เซิร์ฟเวอร์
                        sta_buf=1;
                        goto loop1;
                        }
                }
        case 2:                         ส่วนของการติดตามการใช้ไฟฟ้า
                {
                tcp_listen(&echosock,PORT_L,host,0,NULL,0);
                                        รอรับการติดต่อจากเซิร์ฟเวอร์
                while (sta_mon!=1)
                        {
                        wd  = VdGetFreeWd(9);
                                        เริ่มต้นการทำงานของวอตช์ด็อก
                        sta_mon=tcp_estab_mon();
                                        เรียกใช้ฟังก์ชันติดตามการใช้ไฟฟ้า
```

```
            store_eng();        เรียกใช้ฟังก์ชันเก็บหน่วยของพลังงานแต่ละ
                                ช่วงเวลา
            }
          sta_mon=0;            เคลียร์ค่าตัวแปรที่ช่วยในวงจรวนซ้ำ
          sta_buf=2;            เก็บค่าส่วนการทำงานเดิมเมื่อมีการติดต่อจาก
                                เซิร์ฟเวอร์ให้มีการปรับตั้งเวลา
          goto loop2;
        }
case 3:                         ส่วนของการส่งหน่วยพลังงานในรอบเดือน
        {
        time_a = read_rtc();    } อ่านค่าเวลาแล้วแปลงให้อยู่ในรูปของตัวแปร
        mktm(&rtc,time_a);      } แบบโครงสร้าง
        old_mon=rtc.tm_mon;     เก็บค่าของเดือน
        tcp_listen(&echosock,PORT_L,host,0,NULL,0);
                                รอรับการติดต่อจากเซิร์ฟเวอร์
        while (sta_mon!=1)
            {
            wd  = VdGetFreeWd(9);
                                เริ่มต้นการทำงานของวอตช์ด็อก
            time_a = read_rtc();
                                อ่านค่าเวลาแล้วแปลงให้อยู่ในรูปของตัวแปร
                                แบบโครงสร้าง
            mktm(&rtc,time_a);
            if ((rtc.tm_hour==0)&&(rtc.tm_min==0)&&(rtc.tm_sec==0)
            &&(rtc.tm_mon!=old_mon))
                    mon=1;      ถ้าค่า ชั่วโมง-นาที-วินาที เป็น 00:00:00 และ
                                เดือนมีการเปลี่ยนแปลง ตัวแปรช่วยมีค่าเป็นหนึ่ง
            else
                    mon=0;
```

```
                    sta_mon=tcp_estab_eng();
                                เรียกใช้ฟังก์ชันส่งหน่วยพลังงานในรอบเดือน
                    if (eng_rev==Rev)
                                ถ้าจำนวนรอบการหมุนครบตามกำหนดจะได้
                                พลังงานหนึ่งหน่วยและให้เคลียร์จำนวนรอบ
                                การหมุนเดิม
                          {
                          eng_rev=0;
                          eng=1;
                          }
                    else
                          eng=0;
                    store_eng();      เรียกใช้ฟังก์ชันเก็บหน่วยของพลังงานแต่ละ
                                      ช่วงเวลา
                          }
               sta_mon=0;             เคลียร์ค่าตัวแปรช่วยที่ใช้ในวงจรวนซ้ำ
               sta_buf=3;             เก็บค่าส่วนการทำงานเดิมเมื่อมีการติดต่อจาก
                                      เซิร์ฟเวอร์ให้ปรับตั้งเวลา
               goto loop2;
               }
          }
           sta_buf=0;                 เคลียร์ค่าตัวแปรช่วยที่เก็บค่าส่วนการทำงาน
                                      เดิม
     }
goto loop1;
WrPortI(I0CR, NULL, 0x00);            ปิดการรับข้อมูลของพอร์ต อี
}
```

```
char* tcp_xrec()                                ฟังก์ชันการรับข้อมูล
{
auto char data_xrec[256];                       ประกาศตัวแปรเพื่อเก็บข้อมูล
unsigned int len;                               ประกาศตัวแปรเพื่อเก็บค่าความยาวข้อมูล
int  status;                                    ประกาศตัวแปรที่ใช้แจ้งความผิดพลาด
xrec:
sock_wait_input(&echosock,0,NULL,&status);      รอรับข้อมูลจากเซิร์ฟเวอร์
len=sock_dataready(&echosock);                  เก็บค่าความยาวของข้อมูลที่รับได้
sock_read(&echosock,data_xrec,len);             รับข้อมูลตามความยาวของข้อมูลที่เก็บไว้
data_xrec[len]=0;                               เคลียร์ข้อมูล
return (data_xrec);                             คืนค่าข้อมูลที่รับได้ให้โปรแกรมหลัก
sock_err:
 if (status!=0)                                 ถ้าเกิดความผิดพลาดให้รอรับการติดต่อจาก
                                                เซิร์ฟเวอร์แล้วกลับไปที่ลูบ xrec
        {
        tcp_listen(&echosock,PORT_L,host,0,NULL,0);
        status=0;
        goto xrec;
        }
}
tcp_xsend(char* data_xsend)                     ฟังก์ชันการส่งข้อมูล
{
int status;                                     ประกาศตัวแปรที่ใช้แจ้งความผิดพลาด
xsend:
tcp_open(&echosock,PORT_L,host,PORT_O,NULL);
                                                เรียกการติดต่อกับเซิร์ฟเวอร์
sock_wait_established(&echosock,0,NULL,NULL);
                                                รอการติดต่อกับเซิร์ฟเวอร์ให้สำเร็จ
```

```
sock_write(&echosock,data_xsend,strlen(data_xsend));
                                        ส่งข้อมูลให้เซิร์ฟเวอร์
sock_close(&echosock);                  เรียกปิดการติดต่อกับเซิร์ฟเวอร์
sock_wait_closed(&echosock,0,NULL,NULL);  รอการปิดการติดต่อกับเซิร์ฟเวอร์ให้สำเร็จ
sock_err:
if (status!=0)                          ถ้าเกิดความผิดพลาดให้ไปที่ลูบ xsend
        {
        status=0;
        goto xsend;
        }
}
set_time()                              ฟังก์ชันปรับตั้งเวลา
{
struct tm  rtc;                         ประกาศตัวแปรของเวลาเป็นแบบโครงสร้าง
unsigned int set_sec;
unsigned int set_min;
unsigned int set_hour;                  ประกาศตัวแปรที่ใช้เก็บค่า ชั่วโมง-นาที-
                                        วินาที-วัน-เดือน-ปี
unsigned int set_day;
unsigned int set_mon;
unsigned int set_year;
auto char*      buff;
auto char       buff_0[32];
auto char*      buff_1;                 ประกาศตัวแปรที่ช่วยในการแยกข้อมูล
auto char       buff_2[4];
auto char  buff_3[4];
unsigned int len_x;                     ประกาศตัวแปรที่ช่วยในการแยกข้อมูล
auto char td_send[34];
buff=tcp_xrec();                        รับข้อมูล ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี ที่ส่ง
                                        มาจากเซิร์ฟเวอร์
```

```
strcpy(buff_0,buff);                          เก็บข้อมูลที่ตัวแปร buff_0
buff_1=strchr(buff_0,'-');                    ค้นหาตัวอักษร “-” แล้วเก็บข้อมูลส่วนที่เหลือ
                                              ไว้ที่ตัวแปร buff_1

len_x=strlen(buff_0)-strlen(buff_1);          หาความยาวของข้อมูลชุดแรก
strcpy(buff_2," ");                           เคลียร์ข้อมูลที่ตัวแปร buff_2
strncpy(buff_2,buff_0,len_x);                 เก็บข้อมูลชุดแรกที่ตัวแปร buff_2
set_hour=atoi(buff_2);                        แปลงตัวแปรตัวอักษรชุดแรกให้เป็นตัวเลขเพื่อ
                                              เตรียมปรับตั้งเวลา

buff_2[len_x]=0;                              เคลียร์ข้อมูลที่ตัวแปร buff_2
*buff_1++;                                    เพิ่มค่าตัวชี้ไปที่ตัวอักษรลำดับต่อไป
strcpy(buff_0,buff_1);                        เก็บข้อมูลส่วนที่เหลือไว้ที่ตัวแปร buff_0
buff_1=strchr(buff_0,'-');                    ค้นหาตัวอักษร “-” แล้วเก็บข้อมูลส่วนที่เหลือ
                                              ไว้ที่ตัวแปร buff_1

len_x=strlen(buff_0)-strlen(buff_1);          หาความยาวของข้อมูลชุดที่สอง
strcpy(buff_2," ");                           เคลียร์ข้อมูลที่ตัวแปร buff_2
strncpy(buff_2,buff_0,len_x);                 เก็บข้อมูลชุดแรกที่ตัวแปร buff_2
set_min=atoi(buff_2);                         แปลงตัวแปรตัวอักษรชุดแรกให้เป็นตัวเลขเพื่อ
                                              เตรียมปรับตั้งเวลา

buff_2[len_x]=0;                              เคลียร์ข้อมูลที่ตัวแปร buff_2
*buff_1++;                                    เพิ่มค่าตัวชี้ไปที่ตัวอักษรลำดับต่อไป
strcpy(buff_0,buff_1);                        เก็บข้อมูลส่วนที่เหลือไว้ที่ตัวแปร buff_0
buff_1=strchr(buff_0,'-');                    ค้นหาตัวอักษร “-” แล้วเก็บข้อมูลส่วนที่เหลือ
                                              ไว้ที่ตัวแปร buff_1

len_x=strlen(buff_0)-strlen(buff_1);          หาความยาวของข้อมูลชุดที่สาม
strcpy(buff_2," ");                           เคลียร์ข้อมูลที่ตัวแปร buff_2
strncpy(buff_2,buff_0,len_x);                 เก็บข้อมูลชุดแรกที่ตัวแปร buff_2
set_sec=atoi(buff_2);                         แปลงตัวแปรตัวอักษรชุดแรกให้เป็นตัวเลขเพื่อ
                                              เตรียมปรับตั้งเวลา
```

```
buff_2[len_x]=0;                          เคลียร์ข้อมูลที่ตัวแปร buff_2
*buff_1++;                                เพิ่มค่าตัวชี้ไปที่ตัวอักษรลำดับต่อไป
strcpy(buff_0,buff_1);                    เก็บข้อมูลส่วนที่เหลือไว้ที่ตัวแปร buff_0
buff_1=strchr(buff_0,'-');                ค้นหาตัวอักษร “-” แล้วเก็บข้อมูลส่วนที่เหลือ
                                          ไว้ที่ตัวแปร buff_1

len_x=strlen(buff_0)-strlen(buff_1);      หาความยาวของข้อมูลชุดที่สี่
strcpy(buff_3," ");                       เคลียร์ข้อมูลที่ตัวแปร buff_3
strncpy(buff_3,buff_0,len_x);             เก็บข้อมูลชุดแรกที่ตัวแปร buff_3
set_day=atoi(buff_3);                     แปลงตัวแปรตัวอักษรชุดแรกให้เป็นตัวเลขเพื่อ
                                          เตรียมปรับตั้งเวลา

buff_3[len_x]=0;                          เคลียร์ข้อมูลที่ตัวแปร buff_3
*buff_1++;                                เพิ่มค่าตัวชี้ไปที่ตัวอักษรลำดับต่อไป
strcpy(buff_0,buff_1);                    เก็บข้อมูลส่วนที่เหลือไว้ที่ตัวแปร buff_0
buff_1=strchr(buff_0,'-');                ค้นหาตัวอักษร “-” แล้วเก็บข้อมูลส่วนที่เหลือ
                                          ไว้ที่ตัวแปร buff_1

len_x=strlen(buff_0)-strlen(buff_1);      หาความยาวของข้อมูลชุดที่ห้า
strcpy(buff_3," ");                       เคลียร์ข้อมูลที่ตัวแปร buff_3
strncpy(buff_3,buff_0,len_x);             เก็บข้อมูลชุดแรกที่ตัวแปร buff_3
set_mon=atoi(buff_3);                     แปลงตัวแปรตัวอักษรชุดแรกให้เป็นตัวเลขเพื่อ
                                          เตรียมปรับตั้งเวลา

buff_3[len_x]=0;                          เคลียร์ข้อมูลที่ตัวแปร buff_3
*buff_1++;                                เพิ่มค่าตัวชี้ไปที่ตัวอักษรลำดับต่อไป
strcpy(buff_0,buff_1);                    เก็บข้อมูลส่วนที่เหลือไว้ที่ตัวแปร buff_0
set_year=atoi(buff_0);                    แปลงข้อมูลส่วนที่เหลือให้เป็นตัวเลขและปรับ
set_year=set_year-2000;                   แต่ง เพื่อเตรียมปรับตั้งเวลา
rtc.tm_sec = set_sec;
rtc.tm_min = set_min;
rtc.tm_hour = set_hour;                   ปรับตั้งเวลา
rtc.tm_mday = set_day;
rtc.tm_mon = set_mon;
```

```
if (set_year < 80)
set_year = 100 + set_year;
rtc.tm_year = set_year;
tm_wr(&rtc);
```

ปรับตั้งเวลา

```
strcpy(td_send," ");
strcpy(td_send,MY_IP_ADDRESS);
strcat(td_send,"-1-");
strcat(td_send,date_time());
```

เรียกใช้ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี และเพิ่มหมายเลข IP แอดเดรส รหัสส่วนการทำงานให้ข้อมูล

```
tcp_xsend(td_send);
```

ส่งข้อมูลเวลาที่ผ่านการปรับตั้งแล้วให้เซิร์ฟเวอร์

```
}
int tcp_estab_mon()
```

ฟังก์ชันติดตามการใช้ไฟฟ้า

```
{
int stat;
```

ประกาศตัวแปรช่วยในวงจรวนซ้ำของส่วนการติดตามการใช้ไฟฟ้า

```
stat=0;
sock_wait_established(&echosock,0,monitor_x,NULL);
```

รอรับการติดต่อกับเซิร์ฟเวอร์ให้สำเร็จและเรียกใช้ฟังก์ชันช่วยในการติดตามการใช้ไฟฟ้าระหว่างการรอ

```
stat=1;
sock_err:
return (stat);
```

คืนค่าตัวแปรช่วยวงจรวนซ้ำให้โปรแกรมหลัก

```
}
monitor_x()
```

ฟังก์ชันช่วยในการติดตามการใช้ไฟฟ้า

```
{
auto char buf_x[30];
```

ประกาศตัวแปรสำหรับเก็บข้อมูลที่ส่งให้เซิร์ฟเวอร์

```
const char spac[2] ="-";
auto char eng_con[2];
VdHitWd(wd);                                    กระตุ้นการทำงานของวอตซ์ด็อก
        if (serDrdUsed()!=0)                    ถ้ามีการรับข้อมูลที่พอร์ตดี ให้หยุด
        {                                       การทำงานของวอตซ์ด็อกและรับข้อมูล
                VdReleaseWd(wd);
                while ((n = serDread(msg,DINBUFSIZE, TIMEOUT)) == 0) ;
                        n=atoi(msg);
                        if( n==11 )             ถ้าข้อมูลคำสั่งเป็น “11” ให้ส่งแฟ้ม
                                                ข้อมูลที่เก็บไว้ในหน่วยความจำสำรอง
                        {
                        serDwrite(bufferm,strlen(bufferm));
                        while (serDwrFree() != DOUTBUFSIZE);
                        stopserD();
                        n=0;
                        }
        }
if (eng_rev==Rev)                               ถ้าจำนวนรอบการหมุนครบตามกำหนดจะได้
                                                พลังงานหนึ่งหน่วยให้เคลียร์จำนวนรอบการ
                                                หมุนเดิมและทำการส่งข้อมูลให้เซิร์ฟเวอร์ตาม
                                                รูปแบบ “หมายเลข IP แอดเดรส-รหัสส่วน
                                                การทำงาน-หน่วย-ชั่วโมง-นาที-วินาที-วัน-
                                                เดือน-ปี”
        {
        VdReleaseWd(wd);                        หยุดการทำงานของวอตซ์ด็อก
        eng_rev=0;
        eng=1;
        strcpy(buf_x," ");
        strcpy(buf_x,MY_IP_ADDRESS);
```

```
        strcat(buf_x,"-2-");                    จัดรูปแบบของข้อมูล
        ltoa(eng,eng_con);
        strcat(buf_x,eng_con);                  จัดรูปแบบของข้อมูล
        strcat(buf_x,spac);                     จัดรูปแบบของข้อมูล
        strcat(buf_x,date_time());              จัดรูปแบบของข้อมูล
        tcp_xsend(buf_x);
        fopen_wr(&file,TESTFILE);               เปิดแฟ้มข้อมูล
        fshift(&file,50,buf_xa);
        fwrite(&file,buf_x,50);                 บันทึกข้อมูลลงแฟ้ม
        fclose(&file);                          ปิดแฟ้มข้อมูล
        fopen_rd(&file,TESTFILE);               เปิดแฟ้มข้อมูล
        while(fread(&file,buf_x,50)>0)          อ่านแฟ้มข้อมูลเพื่อเก็บข้อมูลไว้ที่
                                                หน่วยความจำสำรองสำหรับการส่ง
                {
                l=strlen(buf_x);
                strncat(bufferm,buf_x,l);
                strcat(bufferm,spac);
                }
        fclose(&file);                          ปิดแฟ้มข้อมูล
        tcp_listen(&echosock,PORT_L,host,0,NULL,0);
        }
}
int tcp_estab_eng()                             ฟังก์ชันส่งหน่วยพลังงานในรอบเดือน
{
int stat;                                       ประกาศตัวแปรช่วยในวงจรวนซ้ำของส่วนการ
                                                ส่งหน่วยพลังงานในรอบเดือน
sock_wait_established(&echosock,0,energy_mon,NULL);
                                                รอรับการติดต่อกับเซิร์ฟเวอร์ให้สำเร็จและเรียก
                                                ใช้ฟังก์ชันช่วยส่งหน่วยพลังงานในระหว่างการรอ
```

```
stat=1;
sock_err:
return (stat);                          คืนค่าตัวแปรช่วยวงจรวนซ้ำให้โปรแกรมหลัก
}
energy_mon()                            ฟังก์ชันช่วยในการส่งหน่วยพลังงานในรอบเดือน
{
auto char data_send[64];                ประกาศตัวแปรสำหรับเก็บข้อมูลที่ส่งให้
                                        เซิร์ฟเวอร์
VdHitWd(wd);                            กระตุ้นการทำงานของวอตซ์ด็อก
if (mon==1)                             ถ้าเวลาเป็น “00:00:00” และมีการเปลี่ยนแปลง
                                        เดือนให้ทำการส่งข้อมูลให้เซิร์ฟเวอร์ตามรูป
                                        แบบ “หมายเลข IP แอดเดรส-รหัสส่วนการ
                                        ทำงาน-หน่วยพลังงานช่วงที่หนึ่ง-หน่วยพลัง
                                        งานช่วงที่สอง-ชั่วโมง-นาที-วินาที-วัน-เดือน-
                                        ปี” และเคลียร์ตัวแปรที่ใช้เก็บหน่วยพลังงาน
        {
        VdReleaseWd(wd);
        strcpy(data_send," ");
        strcpy(data_send,MY_IP_ADDRESS);
        strcat(data_send,"-3-");
        strcpy(data_send,databuf(eng_nitottwo,eng_zetotfour));
        tcp_xsend(data_send);
        eng_nitottwo=0;
        eng_zetotfour=0;
        I2CWrite(EEPROM_ADDRESS,1,eng_ni_buff,store_x);
                                        บันทึกค่าหน่วยพลังงาน ก. ลงหน่วยความจำ
        while(MS_TIMER - t < WRITE_TIME);
        I2CWrite(EEPROM_ADDRESS,4,eng_ze_buff,store_x);
                                        บันทึกค่าหน่วยพลังงาน ข. ลงหน่วยความจำ
```

```
        while(MS_TIMER - t < WRITE_TIME);
        old_mon=rtc.tm_mon;
        tcp_listen(&echosock,PORT_L,host,0,NULL,0);
        }
}
```

char* read_date_x() ฟังก์ชันอ่าน วัน-เดือน-ปี

```
{
unsigned long date_a;
char date_b[3],mon_x[3],year_x[5];
auto char date_c[24];
const char spac[2]="-";
```

ประกาศตัวแปรช่วยในการจัดรูปแบบข้อมูล

```
date_a = read_rtc();
mktm(&rtc,date_a);
```

อ่านค่าเวลาแล้วแปลงให้อยู่ในรูปของตัวแปรแบบโครงสร้าง

```
ltoa(rtc.tm_mday,date_b);
ltoa(rtc.tm_mon,mon_x);
ltoa((1900+rtc.tm_year),year_x);
```

แปลงตัวแปรที่เป็นตัวเลขให้เป็นตัวอักษร

```
strcpy(date_c," ");
strcat(date_c,date_b);
strcat(date_c,spac);
strcat(date_c,mon_x);
strcat(date_c,spac);
strcat(date_c,year_x);
```

จัดรูปแบบข้อมูลเป็น “ วัน-เดือน-ปี ”

```
return (date_c);
```

คืนค่าข้อมูลที่จัดรูปแบบแล้วให้กับโปรแกรมหลัก

```
}
```

char* read_time_x() ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที

```
{
unsigned long time_a;
char sec_x[3],min_x[3],hour_x[3];
auto char time_b[24];
const char spac[2]="-";
```

ประกาศตัวแปรช่วยในการจัดรูปแบบข้อมูล

```
time_a = read_rtc();                    อ่านค่าเวลาแล้วแปลงให้อยู่ในรูปของตัวแปร
mktm(&rtc,time_a);                      แบบโครงสร้าง
ltoa(rtc.tm_sec,sec_x);
ltoa(rtc.tm_min,min_x);                 แปลงตัวแปรที่เป็นตัวเลขให้เป็นตัวอักษร
ltoa(rtc.tm_hour,hour_x);
strcpy(time_b," ");
strcat(time_b,hour_x);
strcat(time_b,spac);                    จัดรูปแบบข้อมูลเป็น " ชั่วโมง-นาที-วินาที "
strcat(time_b,min_x);
strcat(time_b,spac);
strcat(time_b,sec_x);
return (time_b);                        คืนค่าข้อมูลที่จัดรูปแบบแล้วให้กับโปรแกรม
                                        หลัก
}
char* date_time()                       ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี
{
auto char time_x[24];
auto char date_x[24];                   ประกาศตัวแปรช่วยในการจัดรูปแบบข้อมูล
auto char td_buff[24];
strcpy(time_x," ");                     เคลียร์ค่าตัวแปร time_x
strcpy(time_x,read_time_x());           เรียกใช้ฟังก์ชันอ่านค่า ชั่วโมง-นาที-วินาที
                                        แล้วเก็บที่ตัวแปร time_x
strcpy(date_x," ");                     เคลียร์ค่าตัวแปร date_x
strcpy(date_x,read_date_x());           เรียกใช้ฟังก์ชันอ่านค่า วัน-เดือน-ปี แล้วเก็บที่
                                        ตัวแปร date_x
strcpy(td_buff," ");
strcat(td_buff,time_x);                 เชื่อมต่อข้อมูล
strcat(td_buff,"-");
strcat(td_buff,date_x);
```

```
return(td_buff);                                คืนค่าข้อมูลที่จัดรูปแบบแล้วให้กับโปรแกรมหลัก
}
char* databuf(int eng_1,int eng_2)              ฟังก์ชันเตรียมข้อมูลที่ใช้ส่ง
{
auto char buf_x[64];                            ประกาศตัวแปรช่วยในการจัดรูปแบบข้อมูล
const char spac[2] ="-";
auto char eng_x[7];
strcpy(buf_x," ");                              จัดรูปแบบข้อมูลเป็น “ หน่วยพลังงานช่วงที่หนึ่ง-หน่วยพลังงานช่วงที่สอง-ชั่วโมง-นาที-วินาที-วัน-เดือน-ปี ”
ltoa(eng_1,eng_x);
strcat(buf_x,eng_x);
strcat(buf_x,spac);
ltoa(eng_2,eng_x);
strcat(buf_x,eng_x);
strcat(buf_x,spac);
strcat(buf_x,date_time());
return (buf_x);                                 คืนค่าข้อมูลที่จัดรูปแบบแล้วให้กับโปรแกรมหลัก
}
int eng_sum()                                   ฟังก์ชันนับรอบการหมุนของเครื่องวัด
{
eng_rev++;                                      เพิ่มจำนวนรอบการหมุน
}
store_eng()                                     ฟังก์ชันเก็บหน่วยของพลังงานในแต่ละช่วงเวลา
{
time_a = read_rtc();                            อ่านค่าเวลาแล้วแปลงให้อยู่ในรูปของตัวแปรแบบโครงสร้าง
mktm(&rtc,time_a);
```

```
if (eng==1)                                   ถ้าพลังงานครบหนึ่งหน่วยให้ตัวแปรเก็บข้อมูล
                                              ตามช่วงเวลาที่หนึ่ง เวลา 9.00 – 21.59 น. เวลา
                                              นอกเหนือจากนี้ให้เก็บเป็นช่วงเวลาที่สอง
        {
        if ((rtc.tm_hour>=9)&&(rtc.tm_hour<22))
                eng_nitottwo++;               เพิ่มหน่วยพลังงานช่วงเวลาที่หนึ่ง
                itoa(engxr,eng_ni_buff);
                I2CWrite(EEPROM_ADDRESS,1,eng_ni_buff,store_x);
                                              บันทึกค่าหน่วยพลังงาน ก. ลงหน่วยความจำ
                while(MS_TIMER - t < WRITE_TIME);
        else
                eng_zetotfour++;              เพิ่มหน่วยพลังงานช่วงเวลาที่สอง
                I2CWrite(EEPROM_ADDRESS,4,eng_ze_buff,store_x);
                                              บันทึกค่าหน่วยพลังงาน ข. ลงหน่วยความจำ
                while(MS_TIMER - t < WRITE_TIME);
        eng=0;
        }
}
```

sendserD() ฟังก์ชันที่ใช้ในการหยุดส่งข้อมูลจากพอร์ตอนุกรมดี

```
{
const char sp[]=" ";
while (serDwrFree() != DOUTBUFSIZE);          ส่งข้อมูลว่างให้พอร์ตอนุกรม ดี
serDputs(sp,strlen(sp));
}
```

ภาคผนวก ค.

รายละเอียดไมโครคอนโทรลเลอร์แรบบิท 2000 และ ไอซีที่ใช้งาน

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E10937	INTIAL RELEASE, TRACKS A/W @ REV-B	DM	04MAY00	KAH	03MAY00
B	E11182	CHANGE R14 FROM 1K TO 1.3K TO CONFORM WITH UL REGULATIONS ON CUREENT LIMITING RESISTORS	RJH	31AUG00	KAH	31AUG00
C	E11620	CHANGE R14 FROM 1.3K TO 2K. ADDED EXTERNAL 32KHz OSCILLATOR. MODIFIED VBAT CIRCUITRY				

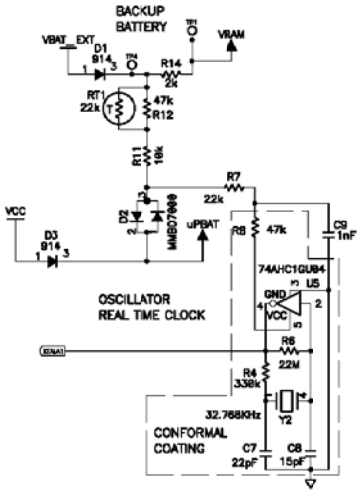
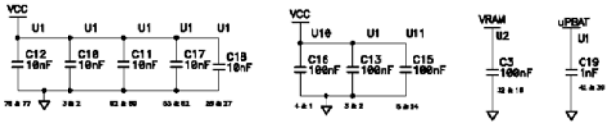


TABLE A

REF DES	DEVICE	DEVICE VOLTAGE INFORMATION					DEVICE: FILTER CAP REF DES(s)
		AGND	GND	VCC	VRAM	NO CONNECTS	
U1	RABBIT 2000		2,27,39 52,77,89	3,16,53, 78,92	42		C13,16 – PIN3, C16,17,12,11 C19 – PIN42 (uPbat)
U2	SRAM+		15		32		C3
U10	ETC811		1	4			C16
U11	FLASH+		24	8			C15



STUFFING TABLE

CIRCUIT		PART	MODEL		
			RCM2000	RCM2010	RCM2020
SRAM	MAIN	U2	512K SRAM	128K SRAM	128K SRAM
	SRAM SELECT	JP1	ZERO ohm ACROSS PINS 2–3	ZERO ohm ACROSS PINS 1–2	ZERO ohm ACROSS PINS 1–2
FLASH	MAIN	U11	256K FLASH	256K FLASH	256K FLASH
	FLASH SELECT	JP2	ZERO ohm ACROSS PINS 1–2	ZERO ohm ACROSS PINS 1–2	ZERO ohm ACROSS PINS 1–2
		JP3	ZERO ohm ACROSS PINS 1–2	ZERO ohm ACROSS PINS 1–2	ZERO ohm ACROSS PINS 1–2
PROCESSOR CRYSTAL		Y1	12.9024MHz	12.9024MHz	9.216MHz

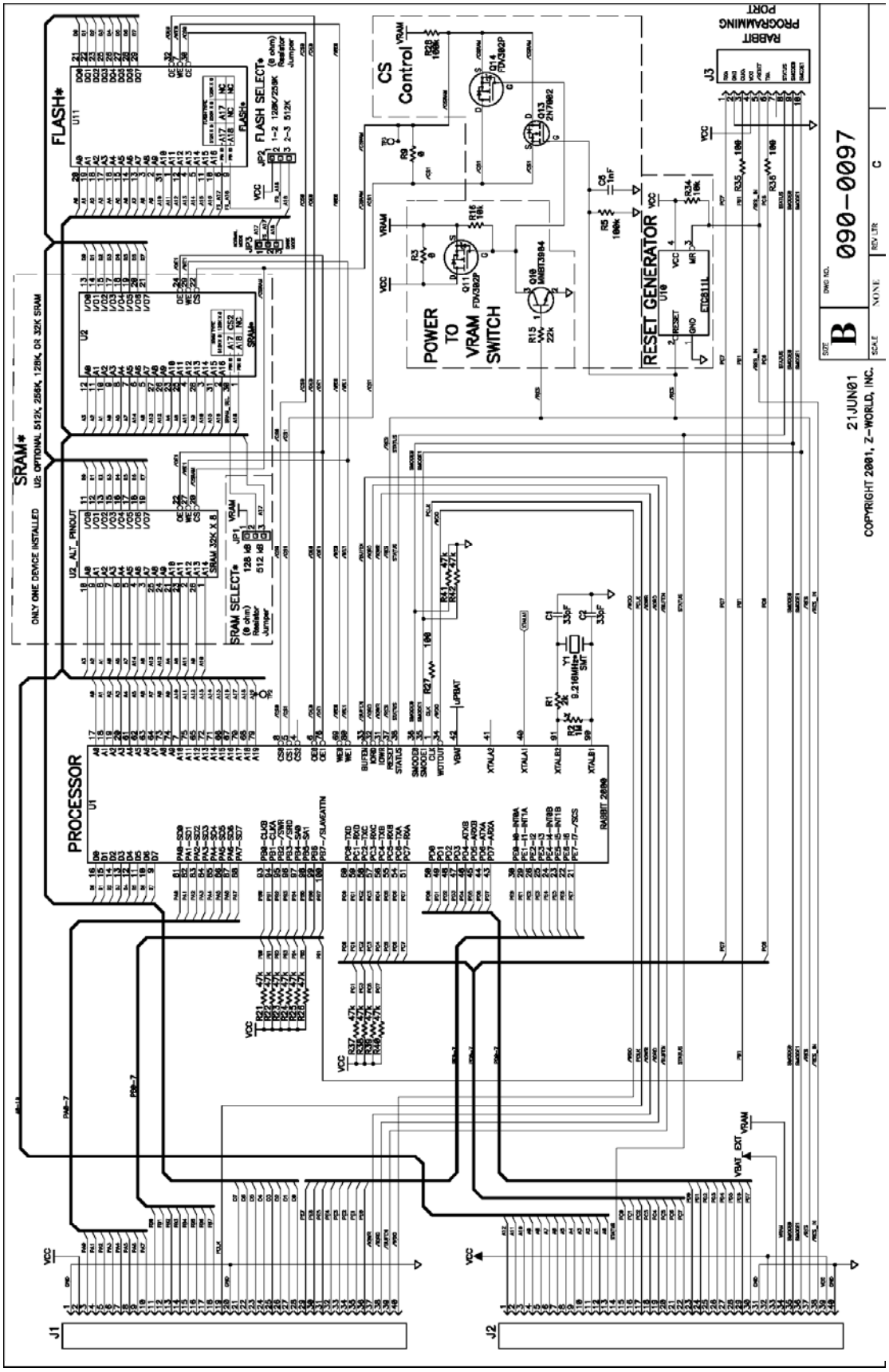
NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (VCC), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (VCC).
4. OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL, SEE STUFFING CHART FOR CLARIFICATION.
5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:	DRAWING CONTENT:		TITLE
	DRAWN BY: (INITIAL RELEASE)	31JAN00	SCHEMATIC DIAGRAM RCM2000 SERIES CORE MODULE
	DM		
	REVISED BY:	21JUN01	
	KAH		
	APPROVALS: INITIAL RELEASE		
	PROJECT ENGINEER:	04MAY00	
	DM		
	ENGINEERING MANAGER:	---	

	SIGNATURES	DATE	

Z-WORLD
2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757 - 4616

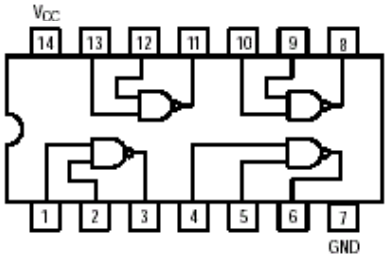
SIZE B | DWG NO. 090-0097 | SCALE NONE | RELEASE DATE 03MAY00


PROCESSOR
U1
RABBIT 2000
FLASH#
U11
SRAM#
ONLY ONE DEVICE INSTALLED
U2: OPTIONAL 512K, 256K, 128K, OR 32K SRAM
U2_ALT_PINOUT
SRAM 32K X 8
FLASH SELECT
SRAM SELECT
CS Control
POWER TO VRAM SWITCH
RESET GENERATOR
RABBIT PROGRAMMING PORT
J1
J2
J3
VRAM
VRAM_EXT
VBAT_EXT
SIZE
B
DWG NO.
090-0097
SCALE
NONE
REV
C
21JUN01
COPYRIGHT 2001, Z-WORLD, INC.

SN74LS00

Quad 2-Input NAND Gate

- ESD > 3500 Volts



LOW
POWER
SCHOTTKY

GUARANTEED OPERATING RANGES

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current – High			–0.4	mA
I_{OL}	Output Current – Low			8.0	mA

PLASTIC
N SUFFIX
CASE 646

SOIC
D SUFFIX
CASE 751A

ORDERING INFORMATION

Device	Package	Shipping
SN74LS00N	14 Pin DIP	2000 Units/Box
SN74LS00D	14 Pin	2500/Tape & Reel

SN74LS00

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions	
		Min	Typ	Max			
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs	
V_{IK}	Input Clamp Diode Voltage		−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	2.7	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
V_{OL}	Output LOW Voltage		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
			0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current			20	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
				0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current			−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)	−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current						
	Total, Output HIGH			1.6	mA	V_{CC} = MAX	
	Total, Output LOW			4.4			

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t_{PLH}	Turn–Off Delay, Input to Output		9.0	15	ns	V_{CC} = 5.0 V C_L = 15 pF
t_{PHL}	Turn–On Delay, Input to Output		10	15	ns	

AM24LC16

2-Wire Serial 16K-bits (2048 x 8) CMOS Electrically Erasable PROM

■ Features

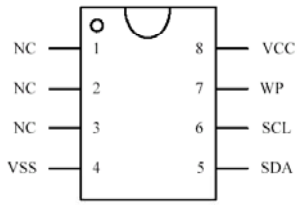
- State- of- the- Art Architecture
  - Non-volatile data storage
  - Full range Vcc = 2.7V to 5.5V
- 2 wire I^2C serial interface
  - Provides bi-directional data transfer protocol
- Hard-ware Write Protection
  - With WP PIN to disable programming command
- 16 bytes page write mode
  - Minimizes total write time per word
- Self-timed write-cycle(including auto-erase)
- Durable and Reliable
  - 40 years data retention
  - Minimum of 1M write/erase cycles per word
  - Unlimited read cycles
  - ESD protection
- Low standby current
- Package: PDIP-8L, SOP-8L

■ General Description

The AM24LC16 is a non-volatile, 16384-bit serial EEPROM with enhanced security device and conforms to all specifications in I^2C 2 wire protocol. The whole memory can be disabled (Write Protected) by connecting the WP pin to Vcc. This section of memory then becomes unalterable unless WP is switched to Vss. The AM24LC16's communication protocol uses CLOCK (SCL) and DATA I/O (SDA) lines to synchronously clock data between the master (for example: a microcomputer)and the slave EEPROM devices (s).

ATC EEPROMs are designed and tested for application requiring high endurance, high reliability, and low power consumption.

■ Connection Diagram

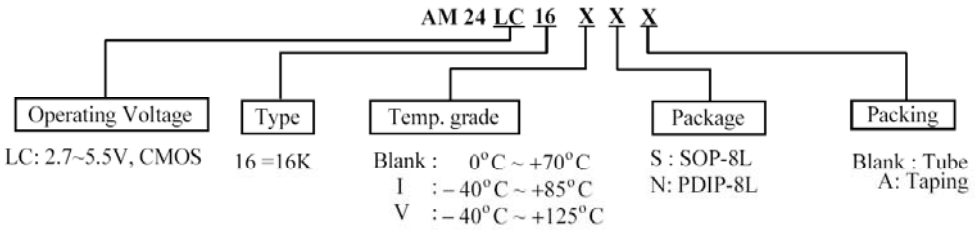


PDIP / SOP

■ Pin Assignments

Name	Description
NC	No connect
VSS	Ground
SDA	Data I/O
SCL	Clock input
WP	Write protect
VCC	Power pin

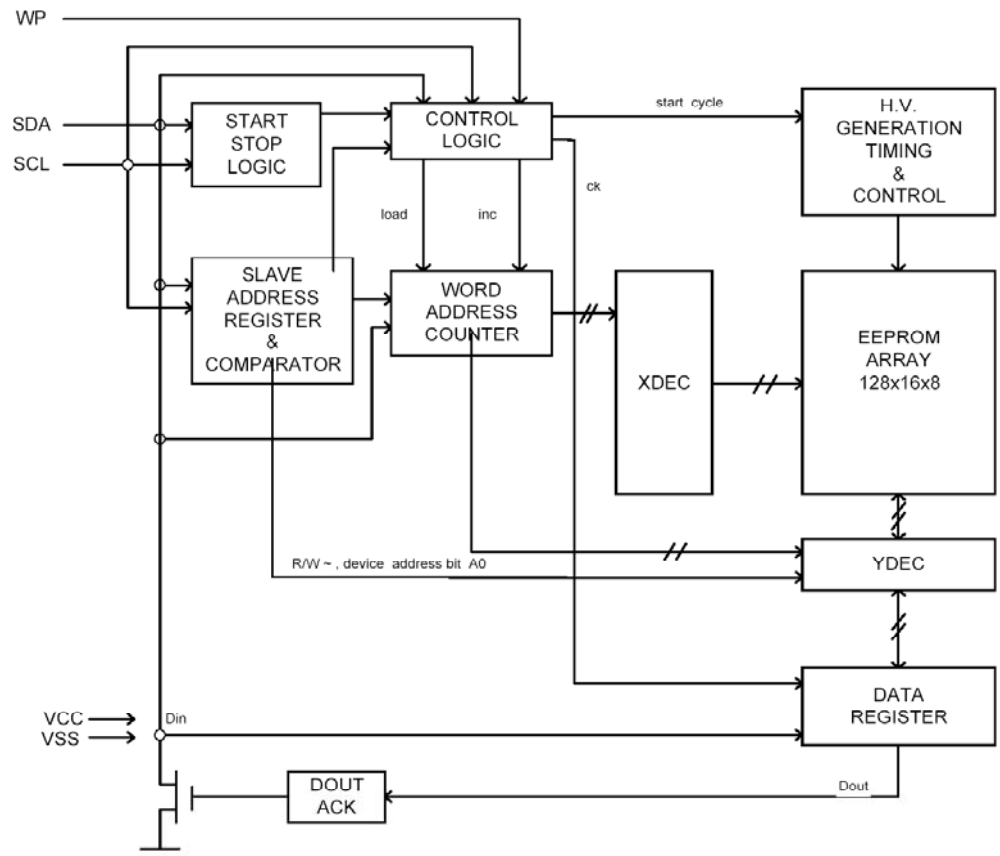
■ Ordering Information



AM24LC16

2-Wire Serial 16K-bits (2048 x 8) CMOS Electrically Erasable PROM

■ Block Diagram



■ Absolute Maximum Ratings

- Storage temperature.....-65°C to + 125°C
- Voltage with respect to ground.....-0.3 to + 6.5 V

NOTE: These are STRESS rating only. Appropriate conditions for operating these devices given elsewhere may permanently damage the part. Prolonged exposure to maximum ratings may affect device reliability.

■ Operating Conditions

Temperature under bias: AM24LC16.....0°C to + 70°C
AM24LC16I..... -40°C to + 85°C
AM24LC16V..... -40°C to +125°C

AM24LC16

2-Wire Serial 16K-bits (2048 x 8) CMOS Electrically Erasable PROM

■ Electrical Characteristics

DC Electrical Characteristics (Vcc =2.7~5.5V, Ta = 25°C)

Symbol	Parameter	Conditions	AM24LC16		Units
			Min	Max	
I_{CC1}	Operating Current (Program)	SCL = 100KHZ CMOS Input Levels	—	3	mA
I_{CC2}	Operating Current (Read)	SCL = 100KHZ CMOS Input Levels	—	200	µA
I_{SB1}	Standby Current	SCL=SDA=0V, Vcc=5V	—	10	µA
I_{SB2}	Standby Current	SCL=SDA=0V, Vcc=3V	—	1	
I_{IL}	Input Leakage	VIN = 0 V to VCC	-1	+1	µA
I_{OL}	Output Leakage	VOUT = 0 V to Vcc	-1	+1	µA
V_{IL}	Input Low Voltage**		-0.1	Vcc x 0.3	V
V_{IH}	Input High Voltage**		Vcc x 0.7	V_{CC}+ 0.2	V
V_{OL1}	Output Low Voltage	IOL = 2.1mA TTL	—	0.4	V
V_{OL2}	Output Low Voltage	IOL = 10uA CMOS	—	0.2	V
V_{LK}	VCC Lockout Voltage	Programming Command Can Be Executed	Default	—	V

Note. ** VIL min and VIH max are reference only and are not tested

■ Switching Characteristics (Under Operating Conditions)

AC Electrical Characteristics (Vcc =2.7~5.5V)

Parameter	Symbol	AM24LC16		Units
		Min	Max	
Clock frequency	Fscl	0	100	kHz
Clock high time	Thigh	4000	—	ns
Clock low time	Tlow	4700	—	ns
SDA and SCL rise time**	Tr	—	1000	ns
SDA and SCL fall time**	Tf	—	300	ns
START condition hold time	Thd:Sta	4000	—	ns
START condition setup time	Tsu:Sta	4700	—	ns
Data input hold time	Thd:Dat	0	—	ns
Data input setup time	Tsu:Dat	250	—	ns
STOP condition setup time	Tsu:Sto	4000	—	ns
Output valid from clock	Taa	300	3500	ns
Bus free time **	Tbuf	4700	—	ns
Data out hold time	Tdh	300	—	ns
Write cycle time	Twr	—	10	ms
5V, 25°C, Byte Mode	Endurance**	1M	—	write cycles

Note. ** This parameter is characterized and is not 100% tested.

Capacitance TA= 25°C , f=250KHz

Symbol	Parameter	Max	Units
C_{OUT}	Output capacitance	5	pF
C_{IN}	Input capacitance	5	pF

AC Conditions of Test

Input Pulse Levels	Vcc x 0.1 to Vcc x 0.9
Input Rise and Fall times	10 ns
Input and Output Timming level	Vcc x 0.5
Output Load	1 TTL Gate and CL = 100pf

ภาคผนวก ง.

บทความที่ได้รับการตีพิมพ์เผยแพร่

การอ่านหน่วยมาตรวัดกำลังงานไฟฟ้าหนึ่งเฟสด้วยแลนแบบไร้สาย

Tele-meter Reading Single Phase Power Meter Using Wireless Local Area Network

ศักดิ์ชัย ไวยลาภ และ อนันท์ อุ่นศิวิไลย์

โทร 0-44411251, 01-5485930 E-mail:mrwiyalap@thaimail.com, anant@ccs.sut.ac.th

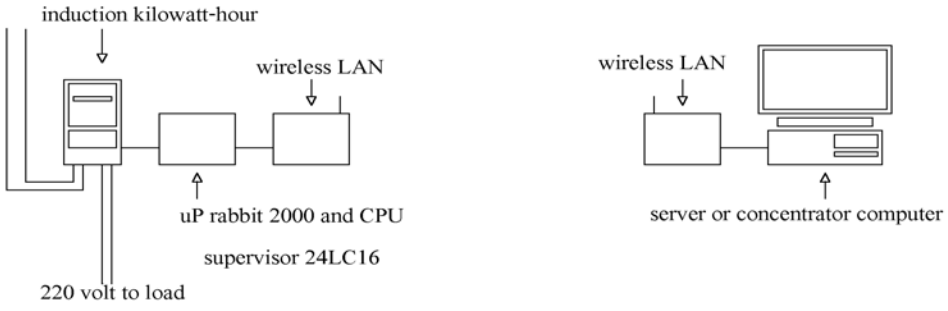
บทคัดย่อ

บทความนี้กล่าวถึงการพัฒนาระบบเครือข่ายของแลนแบบไร้สาย สำหรับการอ่านหน่วยกำลังงานไฟฟ้าจากมาตรวัดกำลังงานระบบหนึ่งเฟสโดยอัตโนมัติ โปรแกรมด้านอุปกรณ์รวมช่องสัญญาณใช้เดลไฟ 5 มาตรวัดกำลังงานเป็นชนิดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ และเชื่อมโยงผ่านไมโครคอนโทรล-เลอร์โดยใช้โปรแกรมไดนามิคซี 7.06 ใช้โพรโตคอล TCP/IP เพื่อกำหนดหมายเลข IP แอดเดรสที่เป็นชั้นซีและเป็นแบบเครือข่ายส่วนบุคคลให้กับมาตรวัดกำลังงาน ข้อมูลที่ได้รับจัดเก็บไว้ในฐานข้อมูลของโปรแกรมแอคเซส 97 ผ่านระบบปฏิบัติการวินโดวส์ ระบบสามารถติดตามการใช้กำลังงานไฟฟ้าได้ในรอบวัน หรือในรอบเดือน และคำนวณค่ากำลังงานไฟฟ้าตามอัตราของการไฟฟ้าส่วนภูมิภาค

1. บทนำ

การใช้งานมาตรวัดชนิดกิโลวัตต์-ชั่วโมงแบบเหนี่ยวนำ (induction kilowatt-hour meter) สำหรับวัดหน่วยกำลังงานไฟฟ้าในปัจจุบันมีปริมาณมาก การจดหน่วยกำลังงานไฟฟ้าจากมาตรวัดชนิดนี้ต้องอาศัยบุคลากรที่มีความชำนาญในพื้นที่เข้าไปบันทึกหน่วย ทำให้เกิดต้นทุนที่สูงและใช้เวลาหลายวัน ข้อมูลที่ได้มีระยะห่างของรอบการใช้กำลังงานไม่คงที่ การนำข้อมูลที่ได้มาใช้ในการวิเคราะห์หรือจัดการด้านกำลังงานไฟฟ้าจึงมีความคาดเคลื่อน

ปัจจุบันมีเทคโนโลยีที่ใช้ในการอ่านหน่วยกำลังงานไฟฟ้าอัตโนมัติหลายแบบ [4] เช่น การส่งคลื่นพาห์ผ่านสายส่งกำลัง (power line carrier) ซึ่งการออกแบบระบบให้มีความเร็วในการรับส่งข้อมูลสูงหรือให้มีความน่าเชื่อถือ ทำได้ยากเพราะสายส่งไฟฟ้ากำลังเป็นตัวกลางที่มีการรบกวนรุนแรงเมื่อโหลดมีการเปลี่ยนแปลง [5] การใช้ระบบเครือข่ายโทรศัพท์ หน่วยงานที่รับผิดชอบจะต้องสูญเสียค่าใช้จ่ายในการเช่าระบบตลอดทั้งเดือนแต่มีการรับส่งข้อมูลเพียงเดือนละครั้ง



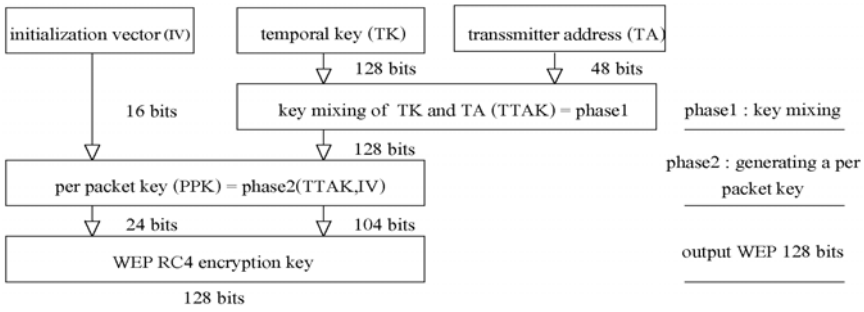
รูปที่ 1 ระบบต้นแบบในการทดสอบ

ในบทความนี้จึงมีวัตถุประสงค์เพื่อการเพิ่มประสิทธิภาพให้กับมาตรวัดกำลังงานไฟฟ้าระบบเดิมให้อ่านและบันทึกข้อมูลการใช้กำลังงานไฟฟ้าในรอบเดือนได้เองอัตโนมัติโดยเพิ่มส่วนของการตรวจจับการหมุนและไมโครคอนโทรลเลอร์ให้กับมาตรวัด อาศัยเครือข่ายของระบบแลนแบบไร้สาย (wireless local area network) ตามมาตรฐาน IEEE 802.11 ในการส่งผ่านข้อมูลเข้าสู่อุปกรณ์รวมช่องสัญญาณ

2. หลักการรับส่งและโพรโตคอล

การสื่อสารที่ใช้ในการรับส่งข้อมูลหน่วยกำลังงานไฟฟ้าในบทความอาศัยเทคโนโลยีของระบบแลนแบบไร้สายที่มีการมอดูเลตชันแบบแบ่งรหัสลำดับโดยตรง [2] (direct sequence code division multiple access) ซึ่งเป็นการเข้าใช้ช่องสัญญาณแบบหลายทางอีกรูปแบบหนึ่ง ซึ่งสัญญาณข้อมูลแต่ละช่องจะถูกคูณด้วยรหัสที่มีความแตกต่างกันก่อนนำไปมอดูเลตกับคลื่นพาห์ที่มีแถบความถี่ในย่าน 2.4 GHz

wired equivalent privacy (WEP) เป็นการเข้ารหัสเพื่อความปลอดภัยของข้อมูลในระบบแลนแบบไร้สาย โดยอาศัยอัลกอริทึมการเข้ารหัสแบบ ron's code pseudo random number generator (RC4PRNG) เป็นรหัสแบบต่อเนื่องภายใต้ระบบเครือข่ายเดียวกัน มาตรวัดกำลังงานเครื่องใดที่จะเข้าเครือข่ายต้องใช้กุญแจสำหรับเข้ารหัสหรือถอดรหัสตัวเดียวกันกับอุปกรณ์รวมช่องสัญญาณ



รูปที่ 2 การเข้ารหัสความปลอดภัยแบบ WEP ที่ใช้ในระบบแลนแบบไร้สาย

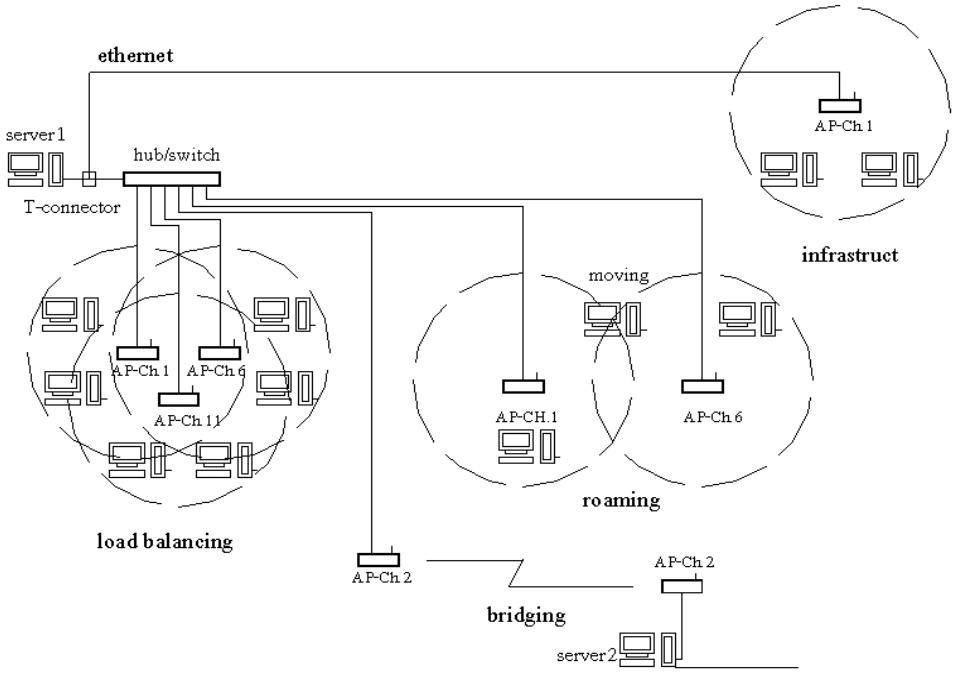
รัศมีของพื้นที่การรับส่งข้อมูลจะแตกต่างกันออกไปขึ้นกับสภาพแวดล้อมของตัวกลางที่มีปัจจัยต่อการลดทอนสัญญาณ หลากมีความผิดผลาดของข้อมูลขึ้นจะทำการส่งข้อมูลซ้ำอีกในอัตราการส่งที่ต่ำลงมาอีก

การใช้โพรโตคอล transmission control protocol/internet protocol (TCP/IP) [6] ในการขนถ่ายข้อมูล โดยกำหนดหมายเลข IP แอดเดรสเป็นชั้นซีและเป็นแบบเครือข่ายส่วนตัวที่ไม่สามารถติดต่อจากระบบเครือข่ายภายนอกได้ สามารถรองรับจำนวนมาตรวัดกำลังงานไฟฟ้าได้สูงสุดถึง 254 เครื่อง โพรโตคอล media access control (MAC) [1] สนับสนุนการเข้าใช้ตัวกลางร่วมกันแต่แบ่งเวลาในการใช้ด้วยวิธีการเข้าถึงแบบสุ่มทำให้มาตรวัดกำลังงานไฟฟ้าแต่ละเครื่องมีโอกาสใช้ตัวกลางเท่าๆ กัน

3. ฐานข้อมูล

ฐานข้อมูลที่ใช้เป็นมอดูลแบบไคลเอ็นต์เซิร์ฟเวอร์ โดยอาศัยการทำงานร่วมกันระหว่างโปรแกรมเดลไฟ 5 กับโปรแกรมแอคเซส 97 ผ่านการเชื่อมโยงข้อมูลโดยอาศัย open data base connectivity (ODBC) ในระบบปฏิบัติการของวินโดวส์ ทำให้อุปกรณ์รวมช่องสัญญาณสามารถรองรับการทำงานของมาตรวัดหลายเครื่องและจัดเก็บข้อมูลที่ซ้ำซ้อนได้

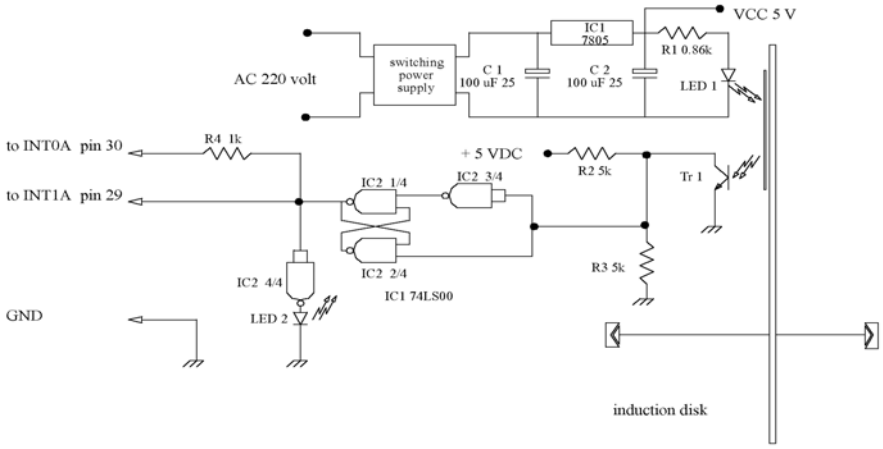
โดยมีหมายเลข IP แอดเดรสเป็นกุญแจหลักสำคัญที่ ใช้ในการอ้างถึงข้อมูลของมาตรวัดแต่ละเครื่อง



รูปที่ 3 สถาปัตยกรรมการเชื่อมต่อระบบเครือข่ายแลนแบบไร้สาย

4. การออกแบบระบบ

การออกแบบได้ใช้แรงดันภายในมาตรวัดกำลังงานไฟฟ้าเป็นแหล่งจ่ายกำลังงานให้ไดโอดเปล่งแสงและวงจรตรวจจับการหมุนมีการติดตั้งแถบทึบแสงขนาดเล็กไว้ที่จานเหนี่ยวนำของมาตรวัด เพื่อให้โฟโต้ทรานซิสเตอร์ซึ่งเป็นอุปกรณ์สำหรับรับแสงสร้างสัญญาณที่มีขอบขาขึ้นและขอบขาลงตามจังหวะที่แถบทึบแสงหมุนผ่าน



รูปที่ 4 วงจรส่วนที่ตรวจจับการหมุนของมาตรวัดกำลังงาน

การปรับปรุงรูปแบบของสัญญาณให้ดีขึ้นด้วยการใช้เกตแนนด์มาต่อเป็นดีฟลิปฟลอป ซึ่งสัญญาณเอาต์พุตจะแปรตามสัญญาณอินพุตก่อนส่งสัญญาณต่อไป

ไมโครคอนโทรลเลอร์ต้องการการอินเตอร์-รัฟเมื่อจานเหนี่ยวนำของมาตรวัดหมุนครบหนึ่งรอบ เพื่อนำไปนับจำนวนรอบการหมุนจนกว่าจะได้จำนวนตามที่มาตรวัดนั้นกำหนดให้เป็นหนึ่งหน่วยกำลังงาน แล้วทำการประมวลผลตามเงื่อนไขซึ่งมีรหัสส่วนการทำงานเป็นตัวกำหนดมาจากอุปกรณ์รวมช่องสัญญาณก่อนจะส่งข้อมูลให้ระบบแลนแบบไร้สายทำการส่งข้อมูล

อุปกรณ์รวมช่องสัญญาณจะทำการส่งเงื่อนไขตามรหัสส่วนการทำงานที่ผู้ใช้งานต้องการให้กับมาตรวัดกำลังงานและรอรับข้อมูลที่มาตรวัดแต่ละเครื่องส่งเข้ามา ทำการแยกข้อมูลออกเป็นส่วนให้ได้ข้อมูลหน่วยกำลังงาน เพื่อนำไปทำการบันทึกและประมวลผลต่อไป

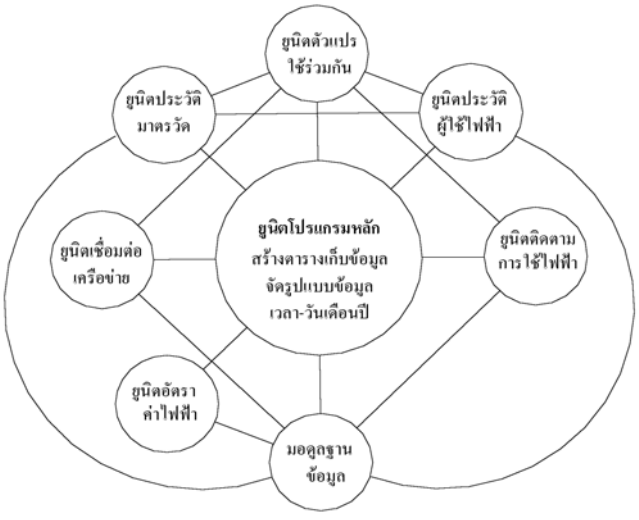
5. การออกแบบโปรแกรม

การติดต่อกับผู้ใช้งานและข้อมูลที่อุปกรณ์รวมช่องสัญญาณได้พัฒนาเป็นขั้นตอนดังนี้

ขั้นตอนที่ 1 พัฒนาโปรแกรมให้บันทึกประวัติของผู้ใช้ไฟฟ้าและประวัติของมาตรวัด พร้อมส่งผ่านให้โปรแกรมแอคเซส 97 จัดทำเป็นฐานข้อมูล

ขั้นตอนที่ 2 พัฒนาโปรแกรมในส่วนการเชื่อมต่อระบบเครือข่ายของอุปกรณ์รวมช่องสัญญาณกับมาตรวัด โดยอาศัยหลักการทำงานของโพรโตคอล TCP/IP และระบบแลนแบบไร้สาย

ขั้นตอนที่ 3 พัฒนาโปรแกรมส่วนของการแสดงผลข้อมูลหน่วยกำลังงานและการคำนวณอัตราค่าไฟฟ้า



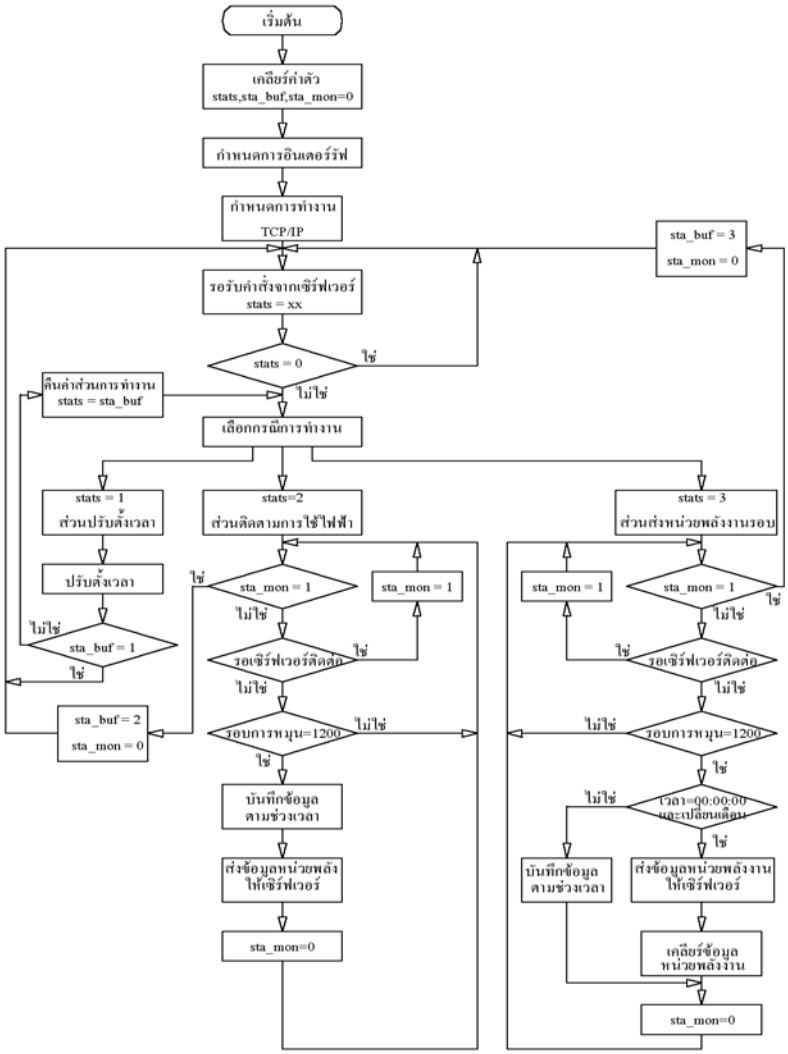
รูปที่ 5 การเชื่อมต่อยูนิตแต่ละส่วนของโปรแกรม

ด้านมาตรวัดกำลังงานไฟฟ้า ได้พัฒนาโดยอาศัยโปรแกรมไดนามิคซี 7.06 [3] ให้ไมโครคอนโทรลเลอร์แรบบิท 2000 [7] เลือกการประมวลผลโดยอาศัยรหัสส่วนการทำงานที่รับได้จากอุปกรณ์รวมช่องสัญญาณ คือ

- ส่วนการปรับตั้งเวลาของไมโครคอนโทรลเลอร์ที่มาตรวัดกำลังงาน เพื่อให้มาตรวัดแต่ละเครื่องใช้อ้างอิงในการบันทึกหน่วยกำลังงานตามช่วงเวลาและจัดส่งข้อมูลให้อุปกรณ์รวมช่องสัญญาณ

- ส่วนติดตามการใช้กำลังงาน ทำการส่งข้อมูลให้อุปกรณ์รวมช่องสัญญาณทุกครั้งที่มีการใช้กำลังงานไฟฟ้าครบหนึ่งหน่วย

- ส่วนการส่งหน่วยกำลังงานในรอบเดือน ทำการส่งข้อมูลให้อุปกรณ์รวมช่องสัญญาณที่เวลาศูนย์นาฬิกาในวันที่หนึ่งของเดือนถัดไป



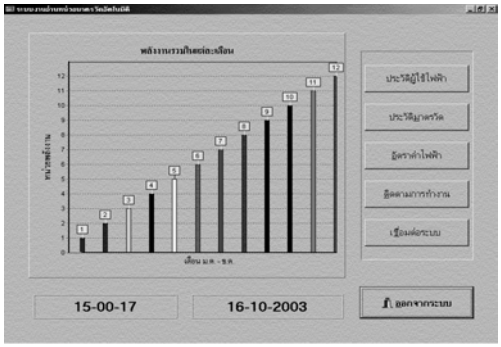
รูปที่ 6 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์

6. ตัวอย่างการใช้งาน

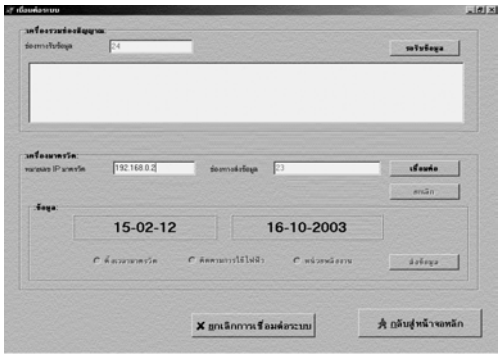
โปรแกรมหลักจะแสดงกราฟการใช้กำลังงานรวมเป็นรายเดือนในรอบหนึ่งปีและมีตัวเลือกย่อยเพื่อช่วยสนับสนุนการทำงาน

ตัวเลือกประวัติผู้ใช้ไฟฟ้าและตัวเลือกประวัติมาตรวัดจะแสดงรายละเอียดเกี่ยวกับการแก้ไข การเพิ่มเติมประวัติของผู้ใช้ไฟฟ้าและประวัติมาตรวัด ตัวเลือกอัตราค่าไฟฟ้าจะแสดงรายละเอียดของอัตราค่าไฟฟ้าอ้างอิงตามการไฟฟ้าส่วนภูมิภาค

ตัวเลือกเชื่อมต่อระบบเครือข่ายจะแสดงรายละเอียดของการเชื่อมต่อมาตรวัดเข้ากับเครือข่ายของระบบแลนและการกำหนดเงื่อนไขรหัสส่วนการทำงานให้กับมาตรวัดแต่ละเครื่อง

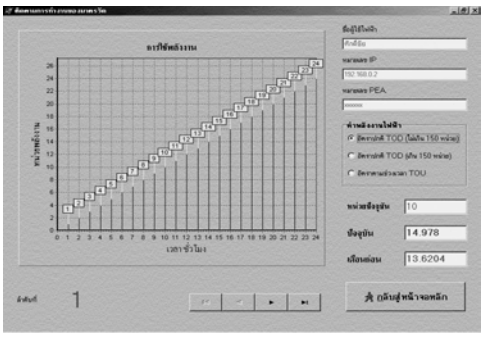


รูปที่ 7 หน้าจอหลักของโปรแกรม



รูปที่ 8 หน้าจอเชื่อมต่อระบบเครือข่าย

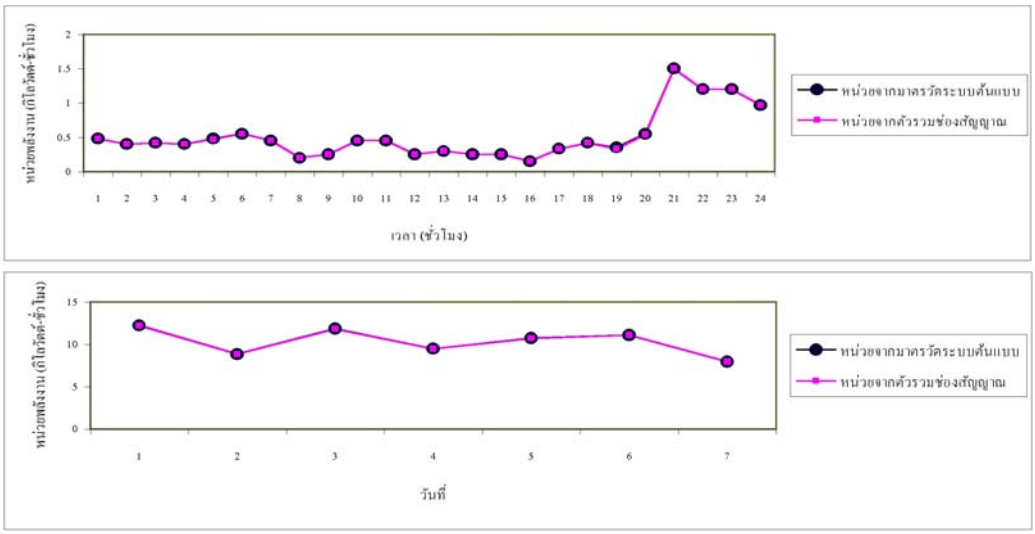
ตัวเลือกติดตามการใช้กำลังงานไฟฟ้าจะแสดงปริมาณการใช้กำลังงานไฟฟ้าของมาตรวัด แต่ละเครื่องพร้อมทั้งการคำนวณค่ากำลังงานไฟฟ้า



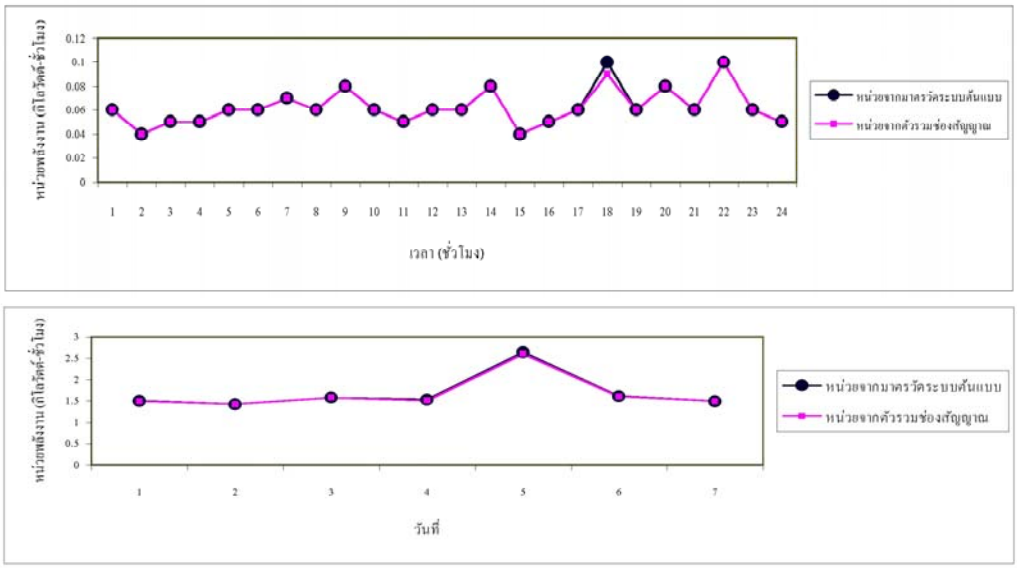
รูปที่ 9 หน้าจอติดตามการใช้กำลังไฟฟ้า

7.ผลการทดสอบ

มีการติดตั้งเพื่อทดสอบระบบเป็นจำนวน 2 แห่ง ซึ่งมีระยะทางและสภาพแวดล้อมที่แตกต่างกัน เป็นระยะเวลา 24 ชั่วโมง และ 7 วัน นำผลที่ได้มาหาความผิดพลาดของระบบเทียบกับหน่วยที่ได้จากมาตรวัดกำลังงานของการไฟฟ้าส่วนภูมิภาค ผลการทำงานของอุปกรณ์และโปรแกรมที่พัฒนาขึ้นในระบบสามารถทำงานได้กว่าร้อยเปอร์เซ็นต์



รูปที่ 10 ข้อมูลจุดทดสอบที่หนึ่งระยะทางในการทดสอบ 10 เมตร



รูปที่ 11 ข้อมูลจุดทดสอบที่สองระยะทางในการทดสอบ 180 เมตร

อีกทั้งสามารถติดตามการใช้กำลังงานไฟฟ้าในแต่ละชั่วโมงในรอบวัน ทำให้สามารถนำหน่วยพลังงานนั้นมาคำนวณอัตราค่าไฟฟ้าในอัตราต่างๆ ได้

8. สรุปและเสนอแนะ

บทความนี้แสดงการเพิ่มประสิทธิภาพให้มาตรวัดกิโลวัตต์-ชั่วโมง แบบเหนี่ยวนำระบบหนึ่ง เฟส ให้สามารถติดตามการใช้กำลังงานไฟฟ้า บันทึกและส่งข้อมูลหน่วยกำลังงานไฟฟ้าได้เองอัตโนมัติเมื่อครบรอบเดือน เป็นการเพิ่มความถูกต้องของ ข้อมูลหน่วยกำลังงานในรอบเดือน ทำให้ทราบรายละเอียดเกี่ยวกับการใช้กำลังงานของผู้ใช้ไฟฟ้าและสามารถคำนวณค่ากำลังงานไฟฟ้าตามอัตราของ การไฟฟ้าส่วนภูมิภาค

การติดตั้งรีพีตเตอร์หรือการเปลี่ยนแปลงเสาสัญญาณให้มีค่าอัตราการขยายสูงขึ้นจะทำมีสัญญาณที่ดีขึ้น เป็นการเพิ่มหรือขยายพื้นที่ในการใช้งานและลดความผิดผลาดที่อาจเกิดขึ้นกับการรับส่งข้อมูลลงได้

9. เอกสารอ้างอิง

[1] อภิชาต อัศวาดิศยางกูร. (2539). ทฤษฎีและการใช้ระบบเครือข่าย. สำนักพิมพ์ฟิสิกส์เซ็นเตอร์. กรุงเทพฯ.

[2] Artecch, R.P.,(1996). CDMA for wireless personal communication. House Publishers, New York.

[3] Dynamic C Premier User's Manual, (2001). Z-World, California.

[4] Ghajar, R., Khalife,J., and Richani,B., (2000). Design and cost analysis of an automatic meter reading system for Electricite' du Liban. Utilities Policy.

[5] Hooijen, O.G., (1998). A channel Model for the Residential Power circuit Used as a Digital Communication Medium. IEEE Transactions on Electromagnetic Compatibility. 40(4): 331-336.

[6] Parker, T., (1994). Teach yourself TCP/IP in 14 days. Sams Publishing, New York.

[7] Rabbit 2000^{TM}Microprocessor User's Manual. (2001). Rabbit Semiconductor, California.

ที่ ศธ 0515(15)/6/ 007

สถานจัดการและอนุรักษ์พลังงาน
มหาวิทยาลัยเชียงใหม่
239 ถ.ห้วยแก้ว ต.สุเทพ
อ.เมือง จ.เชียงใหม่ 50200

2 ธันวาคม 2546

เรื่อง แจ้งผลการพิจารณาบทความเพื่อลงเผยแพร่ในวารสารโลกพลังงาน

เรียน คุณศักดิ์ชัย ไวยลาภ

ตามที่ท่านได้ให้ความสนใจเขียนบทความ เรื่อง "การอ่านหน่วยมาตรวัดกำลังงานไฟฟ้าหนึ่งเฟสด้วยแลนแบบไร้สาย" (Tele-meter Reading Single Phase Power Meter Using Wireless Local Area Network) ส่งมาให้กองบรรณาธิการวารสารโลกพลังงานเพื่อพิจารณาลงเผยแพร่ในวารสาร ความละเอียดทราบแล้วนั้น

ในการนี้ สถานฯ โดยบรรณาธิการวารสารโลกพลังงาน ได้พิจารณาบทความดังกล่าวเป็นที่เรียบร้อยแล้ว จึงใคร่ขอแจ้งให้ท่านทราบว่า บทความของท่านจะได้รับการตีพิมพ์เผยแพร่ลงในวารสารโลกพลังงาน ประจำฉบับที่ 22 เดือนมกราคม-มีนาคม 2547 และจะได้จัดส่งให้ท่านในลำดับต่อไป

จึงเรียนมาเพื่อโปรดทราบและขอขอบคุณมา ณ โอกาสนี้ด้วย

ขอแสดงความนับถือ

(รองศาสตราจารย์ประเสริฐ ฤกษ์เกรียงไกร)
ผู้อำนวยการสถานจัดการและอนุรักษ์พลังงาน
มหาวิทยาลัยเชียงใหม่

ฝ่ายประชาสัมพันธ์
โทรศัพท์ 0-5394-2007-9 ต่อ 108
โทรสาร 0-5389-2375

ประวัติผู้เขียน

นายศักดิ์ชัย ไวยลาภ สำเร็จการศึกษาจากโรงเรียนช่างการไฟฟ้าส่วนภูมิภาค เมื่อปี พ.ศ. 2534 ภายหลังสำเร็จการศึกษาได้เข้าทำงานกับการไฟฟ้าส่วนภูมิภาคอำเภอสีคิ้ว จังหวัดนครราชสีมา ในปี พ.ศ. 2536 ได้เข้าศึกษาระดับอนุปริญญา ที่ภาควิชาช่างไฟฟ้ากำลัง สถาบันเทคโนโลยีราชมงคล วิทยาเขตภาคตะวันออกเฉียงเหนือ จังหวัดนครราชสีมา และระดับปริญญาตรีที่ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ ศูนย์กลางสถาบันเทคโนโลยีราชมงคล จังหวัดปทุมธานี สำเร็จการศึกษาเมื่อปี พ.ศ. 2540 จากการทำงานด้านวิศวกรรมกับการไฟฟ้าส่วนภูมิภาค จึงทำให้เกิดแรงจูงใจที่จะศึกษาต่อในระดับปริญญามหาบัณฑิต โดยได้เข้าศึกษาต่อในสาขาวิชาวิศวกรรมไฟฟ้า สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารีในปี พ.ศ. 2544 ปัจจุบันดำรงตำแหน่งวิศวกรประจำการไฟฟ้าส่วนภูมิภาคอำเภอสีคิ้ว จังหวัดนครราชสีมา