การพัฒนาต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลบนมาตรฐานดีแอลเอ็มเอสและไพร์ม

นายสิวะรัฐ ลิมปพยอม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

A DEVELOPMENT OF A PROTOTYPE DATA CONCENTRATOR UNIT CONFORMED TO DLMS AND PRIME PROTOCOLS

Mr. Siwarat Limphapayom

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	การพัฒนาต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลบนมาตรฐานดีแอลเอ็มเอสและไพร์ม
โดย	นายสิวะรัฐ ลิมปพยอม
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.วันเฉลิม โปรา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร.เอกชัย ลีลารัศมี)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร.วันเฉลิม โปรา)

.....กรรมการภายนอกมหาวิทยาลัย

(ดร.วิษุวัฒน์ ปลอดประดิษฐ์)

สิวะรัฐ ลิมปพยอม : การพัฒนาต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลบนมาตรฐานดีแอลเอ็มเอสและไพร์ม. (A DEVELOPMENT OF A PROTOTYPE DATA CONCENTRATOR UNIT CONFORMED TO DLMS AND PRIME PROTOCOLS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.วันเฉลิม โปรา, 141 หน้า.

วิทยานิพนธ์ฉบับนี้นำเสนอ การพัฒนาคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม (DLMS/COSEM) เพื่อใช้เป็นโพรโทคอลในการสื่อสารระหว่างอุปกรณ์เก็บรวบรวมข้อมูลกับมาตรอัจฉริยะ ซึ่งเป็นอุปกรณ์ในระบบกริดอัจฉริยะ นอกจากนี้ได้พัฒนาต้นแบบของอุปกรณ์เก็บรวบรวมข้อมูล บนมาตรฐานดีแอลเอ็มเอส/โคเซม เพื่อใช้เป็นตัวกลางในการสื่อสารระหว่างมาตรอัจฉริยะกับระบบกลาง และจำลองระบบสื่อสารระหว่างมาตรอัจฉริยะสามเครื่องกับระบบกลาง ผ่านต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล เพื่อทดสอบการทำงานของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล โดยการสื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลองใช้โพรโทคอลดีแอลเอ็มเอส/โคเซม บนตัวกลางของการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ตามมาตรฐานไพร์ม (PRIME) ในส่วนการสื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับระบบกลางจำลองนั้น ใช้การแลกเปลี่ยนแฟ้มข้อมูลเอกซ์เอ็มแอล (XML) ที่ข้อมูลภายในเข้ารหัสโดยใช้มาตรฐานดีแอลเอ็มเอสเป็นต้นแบบ ผ่านโพรโทคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต (SFTP) บนทีซีพี/ไอพี (TCP/IP) ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลถูกสร้างโดยใช้บอร์ด BeagleBoard-XM เป็นแกนหลัก บอร์ดนี้ใช้ตัวประมวลกลางที่มีสถาปัตยกรรมแบบ ARM Cortex A8 พร้อมทั้งรองรับการทำงานของระบบปฏิบัติการอูบันตู และเชื่อมต่อกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง รวมถึงโมเด็มการสื่อสารแบบยูเอ็มทีเอส (UMTS Modem) ผ่านพอร์ตยูเอสบี (USB) ของตัวบอร์ดได้ มาตรอัจฉริยะถูกจำลองอยู่บนคอมพิวเตอร์ส่วนบุคคลที่ต่อกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง และระบบกลางถูกจำลองอยู่บนคอมพิวเตอร์ส่วนบุคคลที่ต่อกับสายแลน หรือแลนไร้สายเพื่อเชื่อมต่อเครือข่ายอินเตอร์เน็ต เมื่อทำการทดสอบระบบจำลองพบว่าการสื่อสารระหว่างระบบกลาง กับมาตรอัจฉริยะผ่านต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล โดยเรียกใช้บริการของคำสั่งต่างๆ จากคลังโปรแกรมชั้นประยุกต์ดีแอลเอ็มเอส/โคเซมนั้น สามารถแลกเปลี่ยนข้อมูลปริมาณทางไฟฟ้าที่ต้องการได้ถูกต้อง ตัวอย่างเช่น ข้อมูลแรงดันไฟฟ้า โพรไฟล์ภาระ เป็นต้น นอกจากนี้ยังได้นำคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ไปใช้ในการพัฒนาโปรแกรมประยุกต์เพื่ออ่านค่าโพรไฟล์ต่างๆ ภายในมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรมที่ได้มาตรฐานดีแอลเอ็มเอส/โคเซม ได้อย่างถูกต้อง สมบูรณ์ จึงสรุปได้ว่าคลังโปรแกรมชั้นโปรแกรมประยุกต์นั้นสามารถใช้งานได้จริง

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5470490021 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: DATA CONCENTRATOR UNIT / SMART METER / POWER LINE COMMUNICATION / CENTRAL SYSTEM

SIWARAT LIMPHAPAYOM: A DEVELOPMENT OF A PROTOTYPE DATA CONCENTRATOR UNIT CONFORMED TO DLMS AND PRIME PROTOCOLS. ADVISOR: ASST. PROF. WANCHALERM PORA, Ph.D., 141 pp.

This thesis proposes a development of a DLMS/COSEM program library. The library provides services of the DLMS/COSEM command for communication between a data concentrator unit (DCU) and a smart meter. Both of them are the devices in the smart grid system. In addition, a prototype data concentrator unit (DCU) conformed to the DLMS/COSEM standard is also developed. It is used as a representative between smart meters and a central system (CS) in communication. Moreover, a system which simulates the communication of these devices is designed as well for testing the functionality of the prototype. The devices of the system consist of a simulated CS, a prototype DCU, and three simulated smart meters. The Application Protocol Data Units (APDUs) of the communication between smart meters and a DCU are encoded conformably to the DLMS/COSEM standard and they are transferred via Power Line Communication (PLC) conformed to the PRIME standard. The encoded data of the communication between a DCU and a CS is adapted appropriately from the DLMS/COSEM standard. In addition, it is contained in an XML file which is sent through a SFTP on a TCP/IP Protocol. A BeagleBoard-XM board is used as a core of the prototype DCU. This board is powered by ARM Cortex A8; moreover, it supports an Ubuntu operating system. For the sake of communication, a PLC modem and an UMTS/HSPDA modem can be attached to USB ports of the board. The smart meters are emulated on a PC which is plugged into a PLC modem. Similarly, the CS is emulated on another PC which is connected a LAN cable or a wireless LAN hotpot. The results of the simulated system show that the exchanged data, such as a measured voltage, and stored load profiles can be transferred successfully and correctly between those devices by calling the DLMS/COSEM services that are provided by the developed library. Furthermore, the library is also deployed to develop a program application for reading profiles stored in industrial smart meters conformed to the DLMS/COSEM standard. To sum up, the library can be used to communicate to the smart meters conformed to the DLMS/COSEM standard completely and the simulated system can work functionally.

Department: Electrical Engineering Student's Signature .....

Field of Study: Electrical Engineering Advisor's Signature .....

Academic Year: 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้อย่างดี ด้วยความช่วยเหลือของ ผู้ช่วยศาสตราจารย์ ดร.วันเฉลิม โปรา ท่านอาจารย์ที่ปรึกษา ที่ให้คำปรึกษา สนับสนุนและกระตุ้นให้เกิดแรงบันดาลในการทำงานวิจัยตลอดมา รองศาสตราจารย์ ดร. เอกชัย ลีลารัศมี ที่กรุณามาเป็นประธานกรรมการสอบวิทยานิพนธ์ อีกทั้งให้คำแนะนำเพื่อนำไปปรับปรุงให้วิทยานิพนธ์ดียิ่งขึ้น ดร. วิษุวัฒน์ ปลอดประดิษฐ์ ท่านกรรมการสอบวิทยานิพนธ์ผู้ทรงคุณวุฒิภายนอกมหาวิทยาลัย พี่ สราวุฒิ เดชจรัสโยธิน ที่ช่วยให้คำปรึกษาดี ในทุก ๆ ด้านมาโดยตลอด พี่ พิศิษฎ์ สว่างวงศ์อนันต์ และพี่ ๆ ที่บริษัท NDR Solution (Thailand) Co., Ltd. ที่ช่วยให้คำแนะนำ และปรับปรุงรหัสคำสั่งโปรแกรมในวิทยานิพนธ์นี้

ขอขอบพระคุณอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาการความรู้ ตลอดจนอบรมบ่มนิสัยมาทำให้ข้าพเจ้ามีความรู้ความสามารถมากเพียงพอที่จะทำวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณบิดา มารดา ที่คอยเป็นกำลังใจ และสนับสนุนข้าพเจ้าตลอดมาจนกระทั่งวิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์

ขอขอบคุณ รุ่นพี่ เพื่อนๆ และรุ่นน้อง ทุกคนที่คอยเป็นกำลังใจและให้คำปรึกษาที่ดี ทำให้การทำวิทยานิพนธ์เป็นไปอย่างราบรื่น

สารบัญ

หน้า

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

วิทยานิพนธ์ฉบับนี้ถูกเขียนขึ้นเพื่ออธิบายการดำเนินการวิจัย ตั้งแต่เริ่มต้นจนจบ และสรุปการดำเนินการ ในบทนี้จะได้กล่าวถึงแนวเหตุผล วัตถุประสงค์ ขอบเขต วิธีการดำเนินการ และลำดับขั้นตอนในการนำเสนองานวิจัย

1.1. แนวเหตุผลในการทำวิจัย

ในเวลานี้ผู้ให้บริการสาธารณูปโภค (Utilities) ของประเทศไทย ต้องทำงานอย่างหนักเพื่อที่จะผลิตไฟฟ้าให้เพียงพอต่อความต้องการของผู้บริโภคที่สูงขึ้น และเพิ่มขึ้นอย่างรวดเร็ว ซึ่งหากผู้ผลิตไฟฟ้ายังพึ่งพาเชื้อเพลิงจาก ถ่านหิน หรือก๊าซธรรมชาติ ซึ่งมีราคาเพิ่มสูงขึ้นเรื่อยๆ และปริมาณสำรองลดลง ก็จะส่งผลต่อต้นทุนการผลิต และส่งผลกระทบกับสิ่งแวดล้อมมากขึ้นเรื่อยๆ เช่นกัน

แทนที่จะเน้นการเพิ่มจำนวนโรงไฟฟ้า มีอีกวิธีหนึ่งที่จะรองรับความต้องการที่เพิ่มขึ้นของผู้บริโภคได้คือการเพิ่มประสิทธิภาพโครงข่ายไฟฟ้า การดำเนินการแบบหนึ่งซึ่งเป็นที่นิยมกันทั่วโลกในขณะนี้คือ การปรับปรุงโครงข่ายไฟฟ้าที่มีอยู่มานานแล้ว ให้เป็นโครงข่ายไฟฟ้าอัจฉริยะ (Smart Grid) ซึ่งเป็นโครงข่ายที่เฝ้าดู แจ้งเตือน ควบคุม การใช้ไฟฟ้า ให้สอดคล้องกับความสามารถในการผลิตไฟฟ้าได้อย่างชาญฉลาด และยังสามารถรับรองพลังงานทางเลือกอื่นๆ ที่เป็นมิตรกับสิ่งแวดล้อม รวมถึงรถยนต์ไฟฟ้า และบ้านอัจฉริยะได้อีกด้วย

ในประเทศไทย มีรัฐวิสาหกิจที่เกี่ยวข้องกับการผลิตและจำหน่ายไฟฟ้าสามองค์กร ได้แก่ การไฟฟ้าฝ่ายผลิต การไฟฟ้าส่วนภูมิภาค และการไฟฟ้านครหลวง ทั้งสามองค์กรกำลังเริ่มปรับปรุงระบบโครงข่ายไฟฟ้าของตนให้เป็นโครงข่ายไฟฟ้าอัจฉริยะ อันเนื่องมาจากระบบเก่าถูกใช้มาเป็นเวลานานหลายสิบปี มีอุปกรณ์ที่ล้าสมัยอยู่มาก อุปกรณ์บางตัวก็มีอายุการใช้งานนาน ทำให้ประสิทธิภาพ รวมถึงความน่าเชื่อถือของโครงข่ายต่ำ โครงข่ายแบบปัจจุบันมีแนวโน้มว่าจะไม่สามารถรองรับปริมาณการใช้ไฟฟ้าที่เพิ่มขึ้นภายในอีกไม่เกิน 25 ปีข้างหน้า [1] อาจส่งผลต่อความมั่นคงทางไฟฟ้าของประเทศ และการตัดสินใจของนักลงทุนต่างชาติในอนาคตได้

นอกจากนี้ประเทศไทยวางแผนที่จะมีสัดส่วนการใช้พลังงานทดแทนและพลังงานทางเลือกไม่ต่ำกว่า 25% ในอีกสิบปีข้างหน้า [2] พลังงานส่วนใหญ่ของประเทศถูกใช้ในรูปของไฟฟ้า จึงหลีกเลี่ยงไม่ได้ที่จะต้องมีการผลิตไฟฟ้าจากพลังงานทดแทน (Renewable Energy) และพลังงานทางเลือก (Alternative Energy) เช่นฟาร์มเซลล์แสงอาทิตย์ (Solar Farm) ฟาร์มกังหันลม (Wind Farm) ในอัตราส่วนที่สูงขึ้นตามไปด้วย การผลิตไฟฟ้าจากพลังงานธรรมชาติทั้งสองอย่างนี้ มีกำลังการผลิตไม่แน่นอน (Non-firm) แปรตามความเข้มของแดด และความเร็วลมในขณะนั้น โครงข่าย

ไฟฟ้าในปัจจุบันยังไม่สามารถรองรับการผลิตไฟฟ้าที่มีความผันผวนสูงแบบนี้ในสัดส่วนที่มากได้ เพราะความน่าเชื่อถือจะต่ำกว่าจุดที่ยอมรับได้ จำเป็นต้องเพิ่มความชาญฉลาดในการบริหารจัดการระบบไฟฟ้าสำรอง การโยกย้ายพลังงานไฟฟ้าจากจุดหนึ่ง ไปยังอีกจุดหนึ่งได้อย่างทันท่วงทีและปลอดภัย นั่นก็คือการพัฒนาให้เป็นโครงข่ายไฟฟ้าอัจฉริยะนั่นเอง ในบางประเทศผู้ใช้ไฟฟ้านิยมติดเซลล์แสงอาทิตย์บนหลังคา (Solar Rooftop) กันมาก เช่นในประเทศเยอรมัน โครงข่ายไฟฟ้าของประเทศกลุ่มดังกล่าวได้พัฒนาไปมากแล้ว

โครงข่ายไฟฟ้าอัจฉริยะต้องการข้อมูลจากผู้บริโภค เช่นอัตราการบริโภคไฟฟ้า และการผลิตไฟฟ้าจากเซลล์แสงอาทิตย์บนหลังคา ในอัตราที่ถี่มากขึ้นเพื่อช่วยในการตัดสินใจการดำเนินการอย่างใดอย่างหนึ่งอย่างทันท่วงที นอกจากนี้ศูนย์ควบคุมอาจส่งคำสั่งไปแจ้งผู้บริโภคเป็นรายกลุ่ม หรือรายบุคคลอีกด้วย โครงข่ายไฟฟ้าอัจฉริยะจึงจำเป็นต้องวางโครงข่ายการวัดและการสื่อสารเสียใหม่ ซึ่งรู้จักกันทั่วไปว่า โครงสร้างพื้นฐานการวัดขั้นสูง (Advanced Metering Infrastructure, AMI) โครงสร้างพื้นฐานการวัดขั้นสูงอันดับแรกประกอบด้วยอุปกรณ์อัจฉริยะหลายชนิด ได้แก่ มาตรอัจฉริยะ (Smart Meter) และอุปกรณ์เก็บรวบรวมข้อมูล (Data Concentrator Unit, DCU) เป็นต้น อาจกล่าวได้ว่าขั้นตอนลำดับต้นๆ ในการพัฒนาโครงข่ายไฟฟ้าอัจฉริยะ คือการติดตั้งมาตรอัจฉริยะ และอุปกรณ์เก็บรวบรวมข้อมูลให้กระจายอยู่ครอบคลุมพื้นที่ ณ บริเวณที่ต้องการข้อมูล ได้แก่ที่อยู่อาศัย ร้านค้า โรงงานอุตสาหกรรม โรงไฟฟ้าแบบต่างๆ เป็นต้น

เพื่อให้มาตรอัจฉริยะ และอุปกรณ์เก็บรวบรวมข้อมูล รวมถึงระบบกลาง (Central System) สามารถสื่อสารกันได้ อุปกรณ์เหล่านี้ต้องใช้โพรโทคอลเดียวกัน หากทุกอุปกรณ์ปฏิบัติตามโพรโทคอลแล้ว ไม่ว่าอุปกรณ์ต่างชนิด ต่างรุ่น หรือต่างผู้ผลิต ก็จะสามารถทำงานร่วมกันได้ (Interoperability)

มาตรฐานในโลกนี้มีมากมาย มีทั้งข้อดี และข้อเสียแตกต่างกันไป แต่ในวิทยานิพนธ์นี้จะเลือกใช้มาตรฐานดีแอลเอ็มเอส/โคเซม (DLMS/COSEM) [3, 4] สำหรับการสื่อสารในชั้นโปรแกรมประยุกต์ (Application Layer) ขึ้นไป เนื่องจากเป็นมาตรฐานของ IEC [5-8] ที่สามารถรองรับโพรไฟล์ (Profile) การสื่อสารได้หลากหลาย และมีความยืดหยุ่นในการนำไปประยุกต์ใช้งาน

อุปกรณ์เก็บรวบรวมข้อมูลตั้งอยู่ตรงกลางระหว่าง มาตรอัจฉริยะที่ติดกระจายอยู่ทั่วโครงข่ายกับระบบกลาง เพื่อลดปริมาณข้อมูลและจำนวนครั้งการเชื่อมต่อ อุปกรณ์เก็บรวบรวมข้อมูลหนึ่งตัวอาจรับส่งข้อมูลให้กับมาตรอัจฉริยะได้ถึง 500 ตัว แทบจะเป็นไปไม่ได้เลยหากให้มาตรอัจฉริยะซึ่งมีอยู่หลายสิบล้านตัว ติดต่อกับระบบกลางโดยตรง

จากเหตุผลข้างต้นนี้ วิทยานิพนธ์นี้ จึงได้ออกแบบ และพัฒนาคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม (DLMS/COSEM Program Library) ขึ้นมาเพื่อให้อุปกรณ์ในโครงข่ายไฟฟ้าอัจฉริยะ ได้เรียกใช้คลาสต้นแบบสำหรับอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะ รวมถึงได้ออกแบบต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล โดยใช้บอร์ด Beagleboard-xM มาตรอัจฉริยะจำลอง และระบบกลางจำลอง เพื่อทดสอบการทำงานของคลังโปรแกรมดังกล่าวอีกด้วย

1.2. วัตถุประสงค์ของการวิจัย

1. พัฒนาซอฟต์แวร์คลังโปรแกรมชั้นโปรแกรมประยุกต์ และต้นแบบตัวเก็บรวบรวมข้อมูลตามมาตรฐานดีแอลเอ็มเอส/โคเซม เพื่อให้สามารถอ่านข้อมูลจากมาตรอัจฉริยะผ่านทางการสื่อสารผ่านสายไฟฟ้าส่งกำลัง (Power Line Communication, PLC) และส่งข้อมูลต่อให้กลับระบบกลางผ่านระบบเครือค่ายโทรศัพท์เคลื่อนที่ได้

1.3. ขอบเขตของการวิจัย

การวิจัยนี้มีงานหลักๆ อยู่สองส่วน คือ การพัฒนาคลังโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม และต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล (รวมถึงโปรแกรมประยุกต์มาตรอัจฉริยะจำลองและระบบกลางจำลอง เพื่อทดสอบระบบการสื่อสารจำลองอีกด้วย)

1.3.1. **คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม**

พัฒนาคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม โดยมีรายละเอียดดังต่อไปนี้

1. พัฒนาขึ้นทั้งฝั่งเซิร์ฟเวอร์ และฝั่งไคลเอนต์
2. รองรับการอ้างอิงชื่อทั้งแบบชื่อตรรกะ (Logical Name) และชื่อสั้น (Short Name)
3. รองรับการอ้างอิงหลายๆ อ็อบเจกต์ในคำสั่งเดียวกัน (With List)
4. รองรับการส่งข้อมูลแบบบล็อกย่อย (Data Block Transfer)
5. จำลองระบบการสื่อสารระหว่างมาตรอัจฉริยะจำลอง อุปกรณ์เก็บรวบรวมข้อมูล และระบบกลางจำลองโดยมีรายละเอียดดังต่อไปนี้

1.3.2. **การจำลองระบบการสื่อสารระหว่างมาตรอัจฉริยะจำลอง อุปกรณ์เก็บรวบรวมข้อมูลและระบบกลางจำลอง**

ขอบเขตการพัฒนาระบบการสื่อสารจำลองแบ่งออกเป็นสองหัวข้อหลัก ได้แก่ ภาคการสื่อสาร และภารฮาร์ดแวร์ มีรายละเอียดดังต่อไปนี้

1. ภาคการสื่อสาร
   - มาตรอัจฉริยะจำลองสื่อสารกับ อุปกรณ์เก็บรวบรวมข้อมูลทางการสื่อสารผ่านสายไฟฟ้าส่งกำลัง (PLC) ตามมาตรฐานไพร์ม (PRIME) ซึ่งระบุรายละเอียดการสื่อสารในชั้นกายภาพ, ชั้น MAC และชั้นคอนเวอร์เจนซ์ และมาตรฐานดีแอลเอ็มเอส/โคเซม ซึ่งระบุรายละเอียดการสื่อสารในชั้นโปรแกรมประยุกต์

- อุปกรณ์เก็บรวบรวมข้อมูลสื่อสารกับ ระบบกลางจำลองทางการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA) โดยใช้โพรไฟล์ทีซีพี/ไอพี (TCP/IP) เชื่อมต่อกับโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต (Secure File Transfer Protocol, SFTP) เพื่อส่งแฟ้มข้อมูลชนิด XML ที่บรรจุข้อมูลที่เข้ารหัสตามมาตรฐานดีแอลเอ็มเอส/โคเซม

2. ภาคฮาร์ดแวร์ และอุปกรณ์จำลอง

   - อุปกรณ์เก็บรวบรวมข้อมูลใช้บอร์ด Beagleboard-xM เป็นบอร์ดของตัวประมวลผลกลาง ซึ่งใช้สถาปัตยกรรม ARM Cortex A8 โดยมีระบบปฏิบัติการอูบันตูเป็นตัวจัดการการทำงาน

   - มาตรอัจฉริยะจำลอง 3 เครื่อง ใช้ .NET Windows Form Application จำนวน 3 โปรแกรม ทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล 1 เครื่อง

   - ระบบกลางจำลอง 1 เครื่อง ใช้ .NET Windows Form Application ทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล 1 เครื่อง

   - ใช้โมเด็มการสื่อสารผ่านสารไฟฟ้าส่งกำลัง เพื่อจัดการชั้นข้อมูลตามมาตรฐานไพร์มตั้งแต่ชั้นคอนเวอร์เจนซ์ลงไป ทั้งฝั่งอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลอง

   - ใช้โมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA Aircard) เพื่อเชื่อมต่อโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ตบนทีซีพี/ไอพี ในฝั่งอุปกรณ์เก็บรวบรวมข้อมูล

   - ใช้สายแลน หรือแลนแบบไร้สาย เพื่อเชื่อมต่อโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต บนทีซีพี/ไอพี ในฝั่งอุปกรณ์เก็บรวบรวมข้อมูล ในฝั่งระบบกลางจำลอง

1.4. วิธีดำเนินการวิจัย

1. ศึกษามาตรฐานดีแอลเอ็มเอส/โคเซม ผลการศึกษาที่อธิบายมาตรฐานดีแอลเอ็มเอส/โค-เซม โดยย่อแสดงอยู่ในหัวข้อ 2.2

2. ศึกษาการเข้ารหัสแบบ BER (IEC 8825-1) ข้อสรุปโดยสังเขปของ BER แสดงอยู่ในหัวข้อ 2.3

3. ศึกษาการเข้ารหัสแบบ A-XDR (IEC 61334-6) ข้อสรุปโดยสังเขปของ A-XDR แสดงอยู่ในหัวข้อ 2.3

4. ทดสอบการอ่านค่าโพรไฟล์ภาระ (Load Profile), ข้อมูลสำหรับการเก็บเงินค่าบริการ (Billing Data) และบันทึกเหตุการณ์ (Event log) จากมาตรอัจฉริยะยี่ห้อ Landis & Gyr รุ่น ZMD405 ที่รองรับดีแอลเอ็มเอส/โคเซม ผ่านทางโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม (GSM) ดังหัวข้อ 5.5 พร้อมทั้งศึกษาหน่วยข้อมูลโพรโตคอล (PDU) ที่ได้เปรียบเทียบกับการเข้ารหัสตามมาตรฐานในหัวข้อ 1, 2, 3

5. ออกแบบ และเขียนคลังโปรแกรมคำสั่งบริการต่างๆ ภายในชั้นโปรแกรมประยุกต์ ตามมาตรฐานดีแอลเอ็มเอส/โคเซม, BER, A-XDR ด้วยภาษา C++ มาตรฐาน พร้อมทั้งทดสอบเบื้องต้น

6. ออกแบบระบบจำลองการอ่านข้อมูลจากมาตรอัจฉริยะ โดยระบบกลาง ผ่านทางอุปกรณ์เก็บรวบรวมข้อมูลเพื่อทดสอบคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม

7. ศึกษาการสื่อสารผ่านสายไฟฟ้าส่งกำลังตามมาตรฐานไพร์ม โดยใช้โมเด็มสำเร็จรูป

8. ทดสอบการสื่อสารไพร์ม ด้วยโมเด็ม และคลังโปรแกรมไพร์มสำเร็จรูป ที่ถูกเขียนด้วยภาษา C++ มาตรฐาน โดยใช้คอมพิวเตอร์ 2 ตัว ในการจำลองการทำงานของมาตรอัจฉริยะจำลอง และอุปกรณ์เก็บรวบรวมข้อมูล

9. เลือกบอร์ดที่ใช้เพื่อสร้างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และทดสอบบอร์ดเบื้องต้นบนระบบปฏิบัติการอูบันตู พร้อมทั้งลงโปรแกรมต่างๆ ที่จำเป็นต้องใช้

10. ศึกษา และทดสอบใช้คำสั่งเอที เพื่อควบคุมโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็ม-ทีเอส/เอชเอสพีดีเอ ผ่านโปรแกรมบนอูบันตู ผ่านบอร์ด BeagleBoard-XM

11. ศึกษา และทดสอบคลังโปรแกรมสำหรับโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต และ XML สำเร็จรูป ทั้งแบบที่ถูกเขียนด้วยภาษา C++ มาตรฐาน และ Managed C++

12. ออกแบบ และเขียนโปรแกรมเพื่อควบคุมอุปกรณ์รวบรวมข้อมูล โดยใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม, คลังโปรแกรมการสื่อสารผ่านสายไฟฟ้าส่งกำลัง, คลังโปรแกรมโพรโทคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต และ คลังโปรแกรม XML บนระบบปฏิบัติการอูบันตูของบอร์ด Beagleboard-xM ด้วยภาษา C++ มาตรฐาน

13. ออกแบบ และเขียน GUI ของมาตรอัจฉริยะจำลอง โดยใช้ .NET Framework และแก้ไขคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม รวมถึงคลังโปรแกรมการสื่อสารผ่านสายไฟฟ้าส่งกำลังเป็นภาษา Managed C++ เพื่อใช้กับ .NET Windows Form Application

14. ออกแบบ และเขียน GUI ของระบบกลางจำลอง โดยใช้.NET Framework, คลังโปรแกรมโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต และ XML สำเร็จรูป ที่ถูกเขียนด้วยภาษา Managed C++ เพื่อใช้กับ .NET Windows Form Application

15. ทดสอบระบบในข้อ 6 โดยรวมอุปกรณ์ที่ได้จากข้อ 12, 13, 14

16. รวบรวมผลการทดลอง, สรุปผล และ เขียนวิทยานิพนธ์

1.5. ลำดับขั้นตอนในการเสนอผลการวิจัย

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บท รวมบทนี้ โดยบทถัดไปคือบทที่ 2 จะกล่าวถึง ความรู้พื้นฐานและมาตรฐานที่เกี่ยวข้อง เช่น การสื่อสารผ่านสายไฟฟ้าส่งกำลัง, มาตรฐานดีแอลเอ็มเอส/โคเซม, มาตรฐานไพร์ม บทที่ 3 จะกล่าวถึง การออกแบบคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมทั้งฝั่งเซิร์ฟเวอร์ และไคลเอนต์ บทที่ 4 จะกล่าวถึงการออกแบบระบบจำลองเพื่อการทดสอบคลังโปรแกรมดีแอลเอ็มเอส/โคเซม และการทำงานของอุปกรณ์เก็บรวบรวมข้อมูล รวมถึงฮาร์ดแวร์ และซอร์ฟแวร์อื่นๆ ที่เกี่ยวข้อง บทที่ 5 จะกล่าวถึง ขั้นตอนและผลการทดสอบระบบจำลอง ส่วนในบทสุดท้าย บทที่ 6 จะกล่าวถึงสรุปผลงานวิจัย ประโยชน์ที่ได้รับจากการวิจัยนี้ และข้อเสนอแนะในการพัฒนาต่อไป

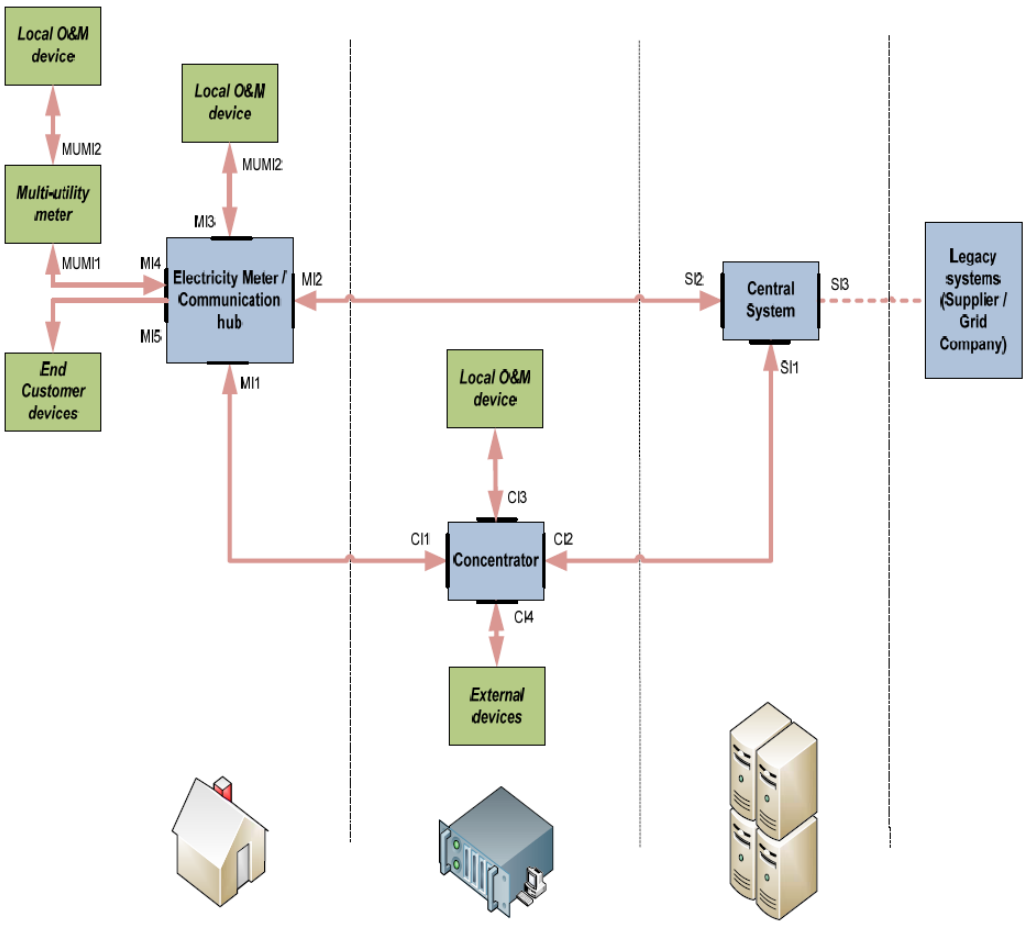
บทที่ 2

ความรู้พื้นฐานและมาตรฐานที่เกี่ยวข้อง

บทนี้จะกล่าวถึงความรู้พื้นฐาน และมาตรฐานที่เกี่ยวข้องในการพัฒนาคลังโปรแกรม ชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และการสื่อสารผ่านสายไฟฟ้าส่งกำลัง

2.1. อุปกรณ์เก็บรวบรวมข้อมูล (Data Concentrator Unit, DCU)

อุปกรณ์เก็บรวบรวมข้อมูล [9] เป็นองค์ประกอบหนึ่งในโครงสร้างการวัดขั้นสูง (Advanced Metering Infrastructure, AMI) ของโครงข่ายอัจฉริยะ (Smart Grid) ดังรูปที่ 2-1



รูปที่ 2-1 องค์ประกอบของโครงสร้างการวัดขั้นสูง

อุปกรณ์เก็บรวบรวมข้อมูลรับข้อมูลจากมาตรอัจฉริยะด้วยการสื่อสารผ่านสายไฟฟ้าส่งกำลัง [7], [8] ในชั้นเครือข่ายลงมาก่อนจะส่งต่อให้กับระบบกลางอีกทอดหนึ่ง ผ่านทางการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA) ซึ่งเป็นการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่ยุคที่สาม ในทำนองเดียวกันอุปกรณ์เก็บรวบรวมข้อมูลรับคำสั่งจากระบบกลาง แล้วส่งต่อให้กับมาตรอัจฉริยะตัวที่ถูกกำหนด

ข้อมูลที่มาตรอัจฉริยะส่งให้ระบบกลางประกอบด้วย ข้อมูลจากการวัด และที่คำนวณได้ เช่น แรงดันไฟฟ้า, กระแสไฟฟ้า, กำลังไฟฟ้า เป็นต้น รวมไปถึงการแจ้งเตือนเหตุการณ์ต่างๆ เช่น แรงดันไฟตก, ไฟดับ เป็นต้น ข้อมูลที่ระบบกลางส่งให้มาตรอัจฉริยะได้แก่ คำสั่งตัดต่อไฟฟ้าของผู้ที่ไม่จ่ายค่าไฟฟ้า เฟิร์มแวร์ใหม่ของมาตรอัจฉริยะ การปรับเปลี่ยนวิธีการคิดค่าไฟฟ้า เป็นต้น

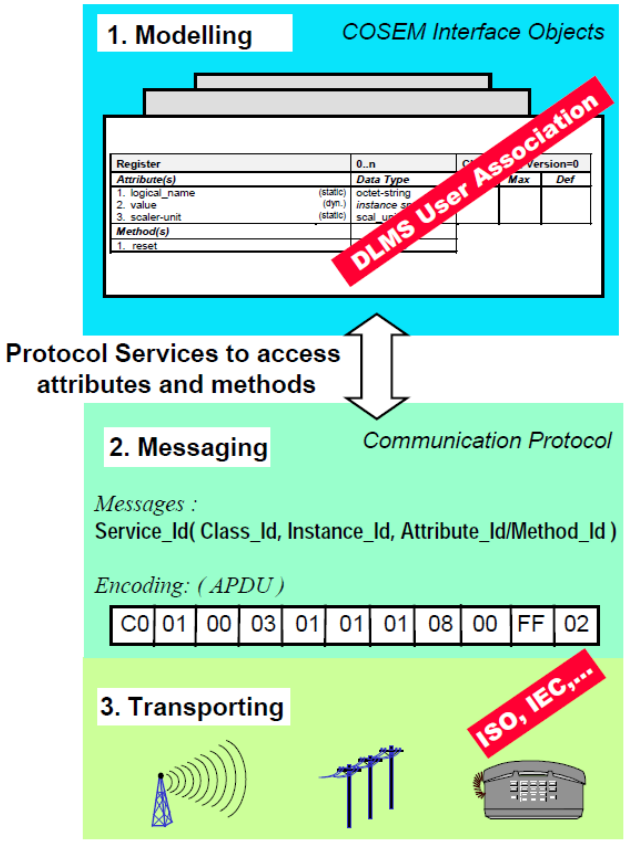
2.2. มาตรฐานดีแอลเอ็มเอส/โคเซม (DLMS/COSEM)

DLMS หรือ Device Language Message Specification คือ ข้อกำหนดต่างๆ ในโปรแกรมประยุกต์ (Application Layer) ซึ่งถูกออกแบบให้เป็นอิสระจากชั้นล่างอื่นๆ โดยได้กำหนดขั้นตอน, รูปแบบ และกลุ่มของบริการที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์ต่างๆ เพื่อสนับสนุนการจำลองการทำงาน และการเก็บข้อมูลต่างๆ ของมาตรอัจฉริยะ รวมถึงรองรับการทำงานร่วมกันของอุปกรณ์จากต่างผู้ผลิต ซึ่งถูกใช้ในการอ่าน, ควบคุม และสั่งงานมาตรอัจฉริยะชนิดต่างๆ เช่น ไฟฟ้า, น้ำ, แก๊ส และความร้อน จากระยะไกล

COSEM หรือ COmpanion Specification for Energy Metering คือแบบจำลองข้อมูลและกระบวนการทำงานสำหรับให้ดีแอลเอ็มเอสเรียกใช้งาน อุปกรณ์ที่ใช้อ่าน และอุปกรณ์ที่ใช้วัดพลังงานต้องเข้าใจโครงสร้างตรงกัน โคเซมจำลองการทำงาน และการเก็บข้อมูลต่างๆในรูปของอ็อบเจกต์ ซึ่งภายในประกอบด้วยสองส่วนคือ คุณสมบัติ (Attribute) และกระบวนการ (Method) [3-8]

อ็อบเจกต์ต่างๆ นั้น ถูกสร้างขึ้นมาจากคลาสอินเตอร์เฟส (Interface Class) ซึ่งเป็นต้นแบบของอ็อบเจกต์ มีหมายเลขเป็นตัวกำกับชนิด ยกตัวอย่างเช่น คลาสอินเตอร์เฟสลำดับที่สามมีชื่อว่า รีจิสเตอร์ (Register) ใช้ในการเก็บค่าต่างๆ ของมาตรอัจฉริยะ เช่น ค่าแรงดัน ค่ากระแสไฟฟ้า ค่าพลังงาน เป็นต้น

อ็อบเจกต์แต่ละตัวที่อยู่ในอุปกรณ์สามารถเข้าถึงได้โดยเรียกใช้บริการต่างๆ ที่อยู่ในชั้นโปรแกรมประยุกต์ ซึ่งถูกกำหนดรูปแบบไว้ในดีแอลเอ็มเอส การระบุว่าบริการหนึ่งต้องอ้างถึงอ็อบเจกต์ใด จะใช้ชื่อตรรกะ (Logical Name) เป็นตัวอ้างอิง อ็อบเจกต์ทุกตัวมีชื่อตรรกะเป็นคุณสมบัติแรก และไม่ซ้ำกัน



รูปที่ 2-2 ภาพรวมขั้นตอนการจัดการข้อมูลของมาตรฐานดีแอลเอ็มเอส/โคเซม

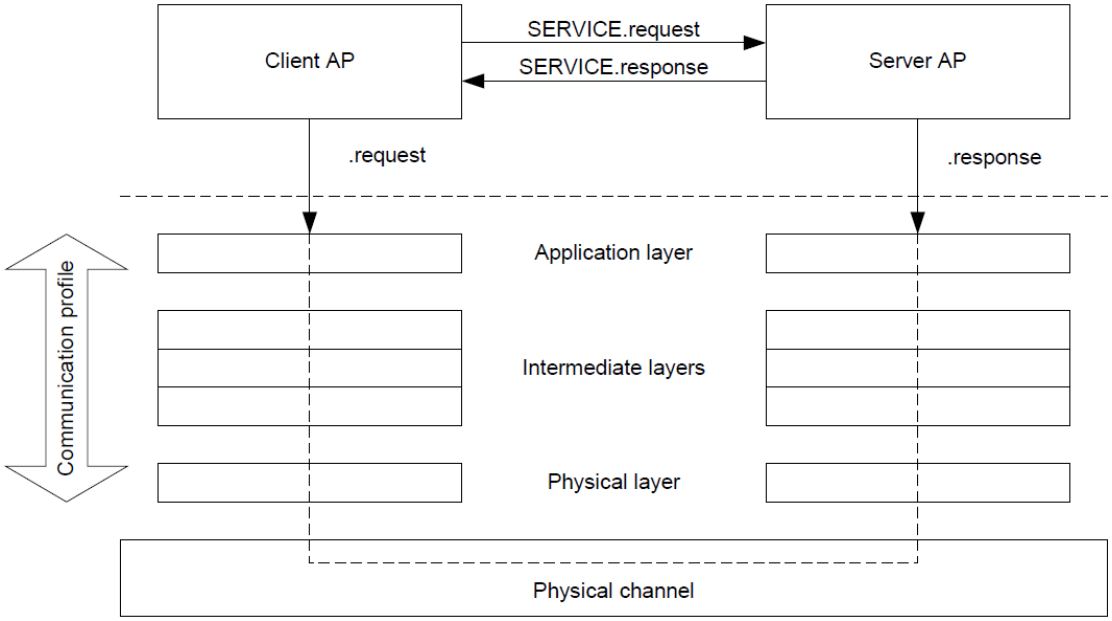
รูปที่ 2-2 เป็นภาพรวมขั้นตอนการจัดการข้อมูลของมาตรฐานดีแอลเอ็มเอส/โคเซม ซึ่งมีทั้งหมด 3 ขั้นตอนดังนี้

1. การออกแบบ (Modelling) ในขั้นตอนนี้ ผู้ออกแบบต้องเลือกว่าต้องการฟังก์ชันและข้อมูลอะไรบ้างในการสื่อสาร โดยต้องเลือกต้นแบบที่เรียกว่า คลาสอินเตอร์เฟส เพื่อนำมาสร้างอ็อบเจกต์ที่ถูกต้องเหมาะสมมาใช้งาน

2. การเรียบเรียงหน่วยข้อมูล (Messaging) ขั้นตอนนี้ทำหน้าที่แปลงอ็อบเจกต์ที่กล่าวไว้ในขั้นตอนที่ 1 ไปเป็นหน่วยข้อมูลโพรโทคอลชั้นโปรแกรมประยุกต์ (Application Protocol Data Unit, APDU) โดยใช้มาตรฐานการเข้ารหัสที่มาตรฐานดีแอลเอ็มเอส/โคเซม ระบุไว้ให้ใช้เรียบเรียงหน่วยข้อมูล ได้แก่ BER [9], และ A-XDR [10] เท่านั้น

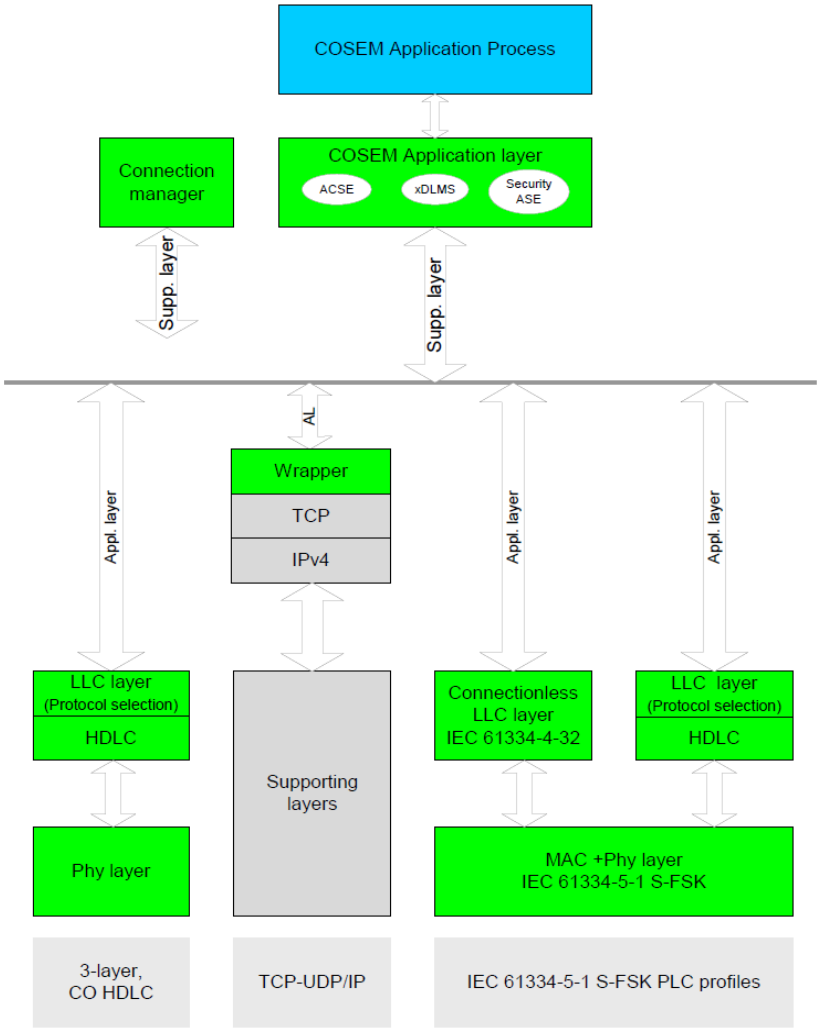
3. การส่งข้อมูล (Transporting) มาตรฐานดีแอลเอ็มเอส/โคเซม ได้ระบุโพรไฟล์ของการสื่อสารไว้หลายโพรไฟล์ สำหรับตัวกลางการสื่อสารหลายๆ ชนิด โดยมีชั้นดีแอลเอ็มเอส หรือเรียกอีกชื่อหนึ่งว่าโปรแกรมประยุกต์ของโคเซม เป็นชั้นที่อยู่บนสุดของทุกโพรไฟล์ มีหน้าที่จัดการ

สร้างช่องทางเชื่อมต่อ รวมถึงบริการต่างๆที่ใช้ในการติดต่อสื่อสารระหว่างมาตรอัจฉริยะกับอุปกรณ์เก็บรวบรวมข้อมูล



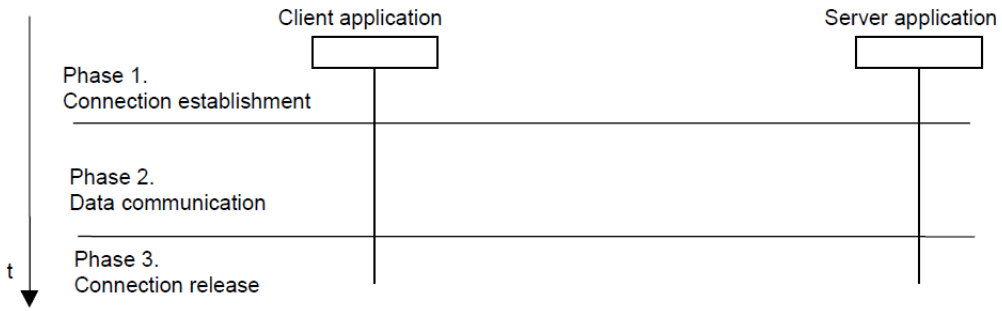
รูปที่ 2-3 เส้นทางการแลกเปลี่ยนข้อมูลระหว่างเซิร์ฟเวอร์ กับไคลเอนต์ ผ่านชั้นต่างๆ ในโพรไฟล์สื่อสาร

จากรูปที่ 2-3 การแลกเปลี่ยนข้อมูลระหว่างมาตรอัจฉริยะ (ต่อไปนี้จะเรียกว่า เซิร์ฟเวอร์) กับอุปกรณ์เก็บรวบรวมข้อมูล (ต่อไปนี้จะเรียกว่า ไคลเอนต์) เริ่มต้นที่ตัวประมวลผลโปรแกรมประยุกต์ฝั่งไคลเอนต์ (Client Application Process, Client AP) ส่งคำร้องขอไปยังตัวประมวลผลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ (Server Application Process, Server AP) ผ่านทางช่องสื่อสารตรรกะ (Logical Channel) แต่ในความเป็นจริงแล้ว Client AP จะเรียกใช้บริการของชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ก่อน แล้วบริการนั้นๆ จะไปเรียกใช้บริการในชั้นที่ติดกันของฝั่งไคลเอนต์ (Client Intermediate Layer, Client IL) ลงไปเรื่อยๆจนถึงชั้นกายภาพของฝั่งไคลเอนต์ (Client Physical Layer, Client PL) เพื่อส่งข้อมูลผ่านช่องสื่อสารกายภาพ (Physical Channel, PC) และเมื่อชั้นกายภาพของฝั่งเซิร์ฟเวอร์ (Server PL) ซึ่งเชื่อมต่ออยู่อีกฝั่งหนึ่งของช่องทางการสื่อสารกายภาพได้รับข้อมูล จะส่งต่อข้อมูลขึ้นไปยังตัว Server AP ผ่านทางชั้นต่างๆ ดังรูปที่ 2-4 แต่ละโพรไฟล์สื่อสารสามารถมีชั้นที่ติดกัน, ชั้นกายภาพ และช่องทางการสื่อสารกายภาพ ที่แตกต่างกันได้ ขึ้นอยู่กับชนิดของตัวกลางการสื่อสาร นอกจากนี้ภายในอุปกรณ์ 1 ตัว สามารถประกอบไปด้วยโพรไฟล์สื่อสารหลายๆ ชนิดได้ ซึ่งการเลือกว่าจะใช้โพรไฟล์ใดเป็นหน้าที่ของ Client AP และ Client AP 1 ตัว สามารถแลกเปลี่ยนข้อมูลกับตัว Server AP 1 ตัว หรือมากกว่าได้ ในทางกลับกัน ตัว Server AP 1 ตัว สามารถแลกเปลี่ยนข้อมูลกับ Client AP 1 ตัว หรือมากกว่าได้



รูปที่ 2-4 โพรไฟล์สื่อสารต่างๆ ของดีแอลเอ็มเอส/โคเซม

2.2.1. **ขั้นตอนการสื่อสารแบบต้องสร้างช่องทางการเชื่อมต่อก่อน (Connection Oriented, CO) ระหว่างไคลเอนต์ และเซิร์ฟเวอร์**



รูปที่ 2-5 ลำดับขั้นตอนในการแลกเปลี่ยนข้อมูลระหว่าง Client AP และ Server AP

ก่อนที่ Client AP และ ตัว Server AP จะสามารถแลกเปลี่ยนข้อมูลกันได้ ทั้ง 2 จะต้องสร้างช่องทางการเชื่อมต่อกันก่อน มีขั้นตอนการแลกเปลี่ยนข้อมูลกัน 3 ขั้นตอน ดังรูปที่ 2-5 ได้แก่

1. ขั้นตอนสร้างช่องทางการเชื่อมต่อ Client AP จะส่งบริการ COSEM-OPEN.request ซึ่งเป็นบริการของหน่วยบริการควบคุมชั้นโปรแกรมประยุกต์ (Application Control Service Element, ACSE) ที่อยู่ภายในชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ไปยัง ตัว Server AP เพื่อส่งพารามิเตอร์ สำคัญต่างๆ หลังจากตัว Server AP รับ และตรวจสอบพารามิเตอร์ต่างๆ ที่ส่งมา จะส่งบริการ COSEM-OPEN.response ซึ่งเป็นบริการของ ACSE ที่อยู่ภายในชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ กลับไปยัง Client AP ถ้าผลสรุปเป็น "ยอมรับ" การสร้างช่องทางการเชื่อมต่อ จะสำเร็จ แต่ถ้าผลสรุปเป็น "ไม่ยอมรับ" ก็จะไม่สามารถแลกเปลี่ยนข้อมูลกันได้

2. ขั้นตอนการส่งข้อมูล Client AP จะเรียกใช้บริการต่างๆ ของหน่วยบริการชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอสเพิ่มเติม (extended DLMS Application Service Element, xDLMS-ASE) ที่อยู่ภายในชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ เพื่อ อ่าน, เขียน, หรือ เรียกใช้กระบวนการ ภายในเซิร์ฟเวอร์และชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ จะเรียกใช้บริการต่างๆของ xDLMS ASE เช่นกัน ในการส่งผลสรุปของการกระทำครั้งก่อนหน้าของ Client AP กลับไป

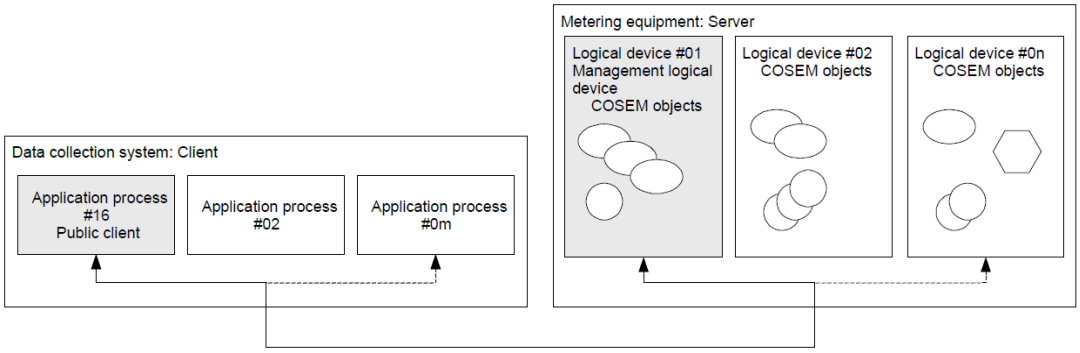
3. ขั้นตอนการยกเลิกช่องทางการเชื่อมต่อ Client AP จะส่งบริการ COSEM-RELEASE.request ซึ่งเป็นบริการของ ACSE ฝั่งไคลเอนต์ ไปยังตัว Server AP เพื่อส่งพารามิเตอร์สำคัญต่างๆ หลังจากตัวServer AP รับและตรวจสอบพารามิเตอร์ต่างๆ ที่ส่งมา จะส่งบริการ COSEM- RELEASE.response ซึ่งเป็นบริการของ ACSE ฝั่งเซิร์ฟเวอร์กลับไปยัง Client AP ถ้าผลสรุปเป็น "ตกลง" การยกเลิกจะสำเร็จ แต่ถ้าผลสรุปเป็น "ไม่ตกลง" การยกเลิกจะถือว่าไม่สำเร็จ

2.2.2. **ความสามารถในการทำงานร่วมกัน และความสามารถในการเชื่อมต่อกันได้ของ Client AP และ Server AP**

Client AP และ Server AP จะถือว่ามีความสามารถในการทำงานร่วมกันก็ต่อเมื่อ Client AP และ Server AP ทั้ง 2 นั้น ใช้คำสั่งที่เป็นมาตรฐานแบบต้องสร้างช่องทางการเชื่อมต่อก่อนของ ACSE และใช้บริบทของโปรแกรมประยุกต์ (Application Contexts) ที่เหมือนกัน

Client AP และ Server AP จะถือว่ามีความสามารถในการเชื่อมต่อกันได้ก็ต่อเมื่อทั้ง 2 นั้น มีโพรไฟล์สื่อสารที่เหมือนกัน และชั้นของโพรโทคอลในระดับชั้นเดียวกันต้องเชื่อมต่อกันอยู่

2.2.3. **แบบจำลองโปรแกรมประยุกต์**

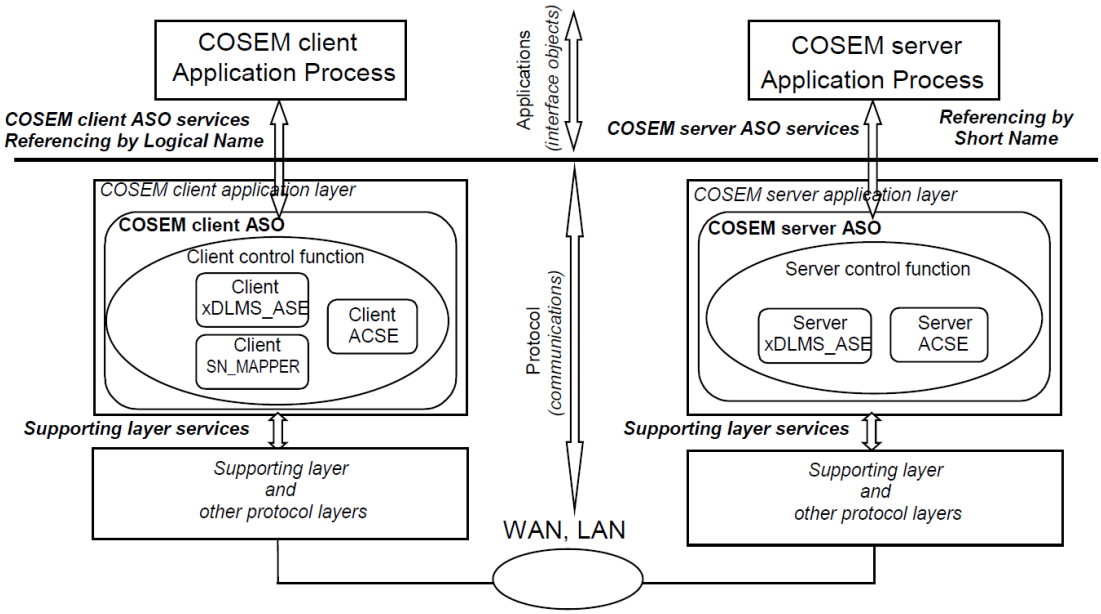


รูปที่ 2-6 แบบจำลองโปรแกรมประยุกต์โคเซมของอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะ

โคเซมจะจำลองว่าใน 1 อุปกรณ์กายภาพ สามารถมีอุปกรณ์ตรรกะหรือตัวประมวลผลโปรแกรมประยุกต์ได้หลาย ๆ อุปกรณ์ และอุปกรณ์ตรรกะแต่ละตัวจะมีความสามารถในการทำงานได้ไม่เหมือนกัน การทำงานต่าง ๆ ถูกจำลองโดยใช้อ็อบเจกต์อินเตอร์เฟส (Interface object) ของโคเซม

ภายในอุปกรณ์เก็บรวบรวมข้อมูลจะถูกจำลองให้มีหลายๆ ตัวประมวลผลโปรแกรมประยุกต์อยู่ภายใน เช่นเดียวกับมาตรอัจฉริยะที่มีหลายอุปกรณ์ตรรกะอยู่ภายใน ดังรูปที่ 2-6 แต่ละ Client AP จะมีบทบาท, สิทธิ์ในการเข้าถึงข้อมูลที่ได้รับจากมาตรอัจฉริยะที่แตกต่างกันไป

2.2.4. โครงสร้างชั้นโปรแกรมประยุกต์โคเซม



รูปที่ 2-7 โครงสร้างชั้นโปรแกรมประยุกต์ และการไหลของข้อมูลของโคเซม

โครงสร้างของชั้นโปรแกรมประยุกต์การไหลของข้อมูลของโคเซม มีลักษณะดังรูปที่ 2-7 ซึ่งองค์ประกอบสำคัญ คือ อ็อบเจกต์บริการชั้นโปรแกรมประยุกต์ (Application Service Object, ASO) ซึ่งจะเป็นตัวจัดสรรบริการต่างๆ ให้กับตัวประมวลผลโปรแกรมประยุกต์ของโคเซม และใช้บริการต่าง ๆ ของชั้นล่างที่รองรับ (Supporting layer) อีกทีหนึ่ง ภายใน COSEM ASO ประกอบด้วย 3 องค์ย่อยที่จำเป็น และ 1 องค์ย่อยประกอบเสริม ดังนี้

1. Association Control Service Element (ACSE) มีหน้าที่จัดเตรียมบริการที่ใช้ในการสร้าง, คอยดูแล และยกเลิกการเชื่อมต่อ เพื่อใช้ในโพรไฟล์สื่อสารแบบต้องสร้างช่องทางการเชื่อมต่อก่อน

2. Extended DLMS Application Service Element (xDLMS-ASE) มีหน้าที่จัดเตรียมบริการที่ใช้ในการแลกเปลี่ยนข้อมูลให้กับชั้นโปรแกรมประยุกต์ของโคเซม ทั้ง 2 ฝั่ง บริการต่าง ๆ ทำตามมาตรฐานดีแอลเอ็มเอส (IEC 61334-4-41) ซึ่งมีบางส่วนที่เพิ่มเติมเข้ามาสำหรับดีแอล-เอ็มเอส/โคเซม

ในมาตรฐานดีแอลเอ็มเอส นั้นจะมีการอ้างอิงคุณสมบัติ และกระบวนการ ที่อยู่ในเซิร์ฟเวอร์โคเซม 2 แบบ คือ 1. ชื่อตรรกะ (Logical Name, LN) 2. ชื่อสั้น (Short Name, SN) ทั้ง 2 การอ้างอิงนี้จะใช้บริการ xDLMS คนละชุดกัน

3. Control Function (CF) มีหน้าที่กำหนดการเรียกใช้บริการต่างๆ ของ ACSE และ xDLMS_ASE ให้เหมาะสม รวมไปถึงบริการของชั้นล่างที่รองรับด้วย

4. SN Mapper (มีแต่ฝั่งไคลเอนต์เท่านั้น) มีหน้าที่ในการแปลงบริการระหว่างชื่อตรรกะ และชื่อสั้น ในกรณีที่ เซิร์ฟเวอร์ ใช้การอ้างอิงแบบชื่อสั้น

2.2.5. **บริการต่างๆ ในชั้นโปรแกรมประยุกต์โคเซม**

บริการต่างๆ ในชั้นโปรแกรมประยุกต์โคเซม ถูกเรียกรวมกันเป็น ASO service ซึ่งจะถูกแบ่งออกเป็น 3 กลุ่มใหญ่ๆ ได้แก่

1. บริการสำหรับการสร้างช่องทางการเชื่อมต่อ และการยกเลิก (ACSE)

1.1 COSEM-OPEN ถูกใช้ในการสร้างช่องทางการเชื่อมต่อระหว่างตัวประมวลผลโปรแกรมประยุกต์โคเซม ทั้ง 2

1.2 COSEM-RELEASE ถูกใช้ในการยกเลิกช่องทางการเชื่อมต่อระหว่างตัวประมวลผลโปรแกรมประยุกต์โคเซม ทั้ง 2

1.3 COSEM-ABORT ชั้นล่างที่รองรับจะเรียกใช้บริการนี้ เพื่อแจ้งการยกเลิกช่องทางการเชื่อมต่อระหว่างตัวประมวลผลโปรแกรมประยุกต์โคเซม ทั้ง 2 ในกรณีเกิดเหตุไม่คาดคิด เช่น ช่องทางการเชื่อมต่อถูกตัดขาด เป็นต้น

2. บริการสำหรับกลุ่มการแลกเปลี่ยนข้อมูล (xDLMS-ASE)

บริการในกลุ่มนี้อย่างที่กล่าวไปแล้วว่าเป็นการเพิ่มบริการใหม่ ให้กับมาตรฐาน DLMS (IEC 61334-4-41) จึงได้ชื่อว่า xDLMS (extended DLMS) นอกจากบริการใหม่ แล้วยังมีรายละเอียดต่างๆที่เพิ่มเข้ามาดังนี้

2.1 ชนิดของข้อมูลใหม่

2.2 เลขรุ่นดีแอลเอ็มเอสใหม่

2.3 บล็อก Conformance ใหม่

2.4 ขยายความความหมายของคำว่า "ขนาดของหน่วยข้อมูลโพรโทคอล"

บริการที่ถูกเพิ่มเข้ามาใหม่ ได้แก่ GET, SET, ACTION และ Eventnotification ในกรณีของการอ้างอิงแบบชื่อตรรกะ และ ได้เพิ่ม ความหลากหลายของบริการ Variable_Access_Specification, Read.response, Write.response เข้าไปจากมาตรฐานดีแอล-เอ็มเอสเดิม เพื่อรองรับการเข้าถึงแบบเลือกค่าได้ (Selective access) และ แลกเปลี่ยนข้อมูลแบบทีละส่วน (Block transfer) ในกรณีของการอ้างอิงแบบชื่อสั้น

3. บริการกลุ่มการจัดการชั้นต่างๆ

ในที่นี่จะกล่าวถึงบริการชั้นโปรแกรมประยุกต์โคเซม เท่านั้น นั่นก็คือ บริการ Set Mapper Table บริการนี้มีอยู่ในฝั่งของไคลเอนต์เท่านั้น หน้าที่ของมัน คือ แปลงชื่ออ็อบเจกต์แบบชื่อตรรกะ เป็นแบบชื่อสั้น เมื่อหน่วยบริการตัวเชื่อมโยงชื่อสั้นชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ (Client SN Mapper ASE) ร้องขอ

ชนิดข้อมูลที่เพิ่มเข้ามาใหม่ และวิธีเข้ารหัสของบริการต่าง ๆ ตามมาตรฐาน A-XDR ระบุอยู่ในหัวข้อ 9.5 ของรายงานทางเทคนิคเล่มสีเขียว (Technical report - Green Book) ของดี-แอลเอ็มเอส/โคเซม

2.2.6. **ความปลอดภัยในการเข้าถึงข้อมูล**

ตามมาตรฐานดีแอลเอ็มเอส/โคเซม ระบบความปลอดภัยถูกจัดการโดยอ็อบเจกต์ที่ชื่อว่า "Association LN" และ "Association SN" ในเซิร์ฟเวอร์โคเซมที่ประกอบไปด้วย อุปกรณ์ตรรกะหลาย ๆ อุปกรณ์สามารถรองรับการสร้างการเชื่อมต่อได้หลายทางพร้อมๆ กันกับไคลเอนต์

หลายตัว ไคลเอนต์แต่ละตัวมีบทบาท และความสามารถในการเข้าถึงคุณสมบัติ และกระบวนการต่างๆ ของเซิร์ฟเวอร์ที่ไม่เท่ากันขึ้นอยู่กับสิทธิ์ในการเข้าถึงข้อมูล

เพื่อที่จะได้สิทธิ์ในการเข้าถึงข้อมูล ไคลเอนต์แต่ละตัวต้องทำการพิสูจน์ตัวตนด้วยการป้อนรหัสผ่าน ในขณะที่กำลังขอสร้างช่องทางการเชื่อมต่อ มาตรฐานดีแอลเอ็มเอส/โคเซมมีระดับความปลอดภัย ในการเข้าถึงข้อมูล 3 ระดับด้วยกัน ได้แก่

1. ความปลอดภัยระดับต่ำสุด เมื่อไคลเอนต์สร้างการเชื่อมต่อด้วยระดับความปลอดภัยระดับต่ำสุด ไคลเอนต์ไม่จำเป็นต้องใส่รหัสในการยืนยันตนเอง แต่สามารถเข้าถึงข้อมูลต่างๆ ตามที่เซิร์ฟเวอร์นั้นๆ ได้ตั้งค่าเอาไว้ หรือได้เพียงข้อมูลพื้นฐานทั่วๆไปของเซิร์ฟเวอร์เท่านั้น

2. ความปลอดภัยระดับต่ำ ในการสร้างช่องทางการเชื่อมต่อที่ระดับความปลอดภัยนี้ ไคลเอนต์จำเป็นต้องใส่รหัสเพื่อยืนยันตนเองลงไปในพารามิเตอร์ที่ชื่อว่า "Calling_Authentication_Value" ของ COSEM-OPEN.request เมื่อเซิร์ฟเวอร์รับ และตรวจสอบถ้ารหัสผ่านถูกต้อง ช่องทางการเชื่อมต่อจะถูกสร้างขึ้น แต่ถ้ารหัสผ่านไม่ถูกต้อง การสร้างช่องทางการเชื่อมต่อจะถูกยกเลิก เซิร์ฟเวอร์จะส่ง COSEM-OPEN.response ซึ่งบรรจุค่า ผลของการสร้างการช่องทางการเชื่อมต่อ และเหตุผลที่การสร้างไม่สำเร็จ กลับไปยังฝั่งไคลเอนต์

3. ความปลอดภัยระดับสูง ในการสร้างช่องทางการเชื่อมต่อที่ระดับความปลอดภัยนี้ทั้งไคลเอนต์และเซิร์ฟเวอร์จำเป็นต้องพิสูจน์ตัวตนทั้งคู่

2.3. การเข้ารหัสข้อมูลแบบบีอีอาร์ (BER) และเอเอกซ์ดีอาร์ (A-XDR)

ในการพัฒนาคลังโปรแกรมชั้นโปรแกรมประยุกต์นั้นมีความจำเป็นต้องใช้การเข้ารหัสข้อมูลสองแบบ ได้แก่ การเข้ารหัสข้อมูลแบบบีอีอาร์ และเอเอกซ์ดีอาร์ โดยแต่ละแบบจะถูกนำไปใช้งานที่แตกต่างกัน บีอีอาร์นั้นถูกใช้ในการเข้ารหัสข้อมูลที่เกี่ยวกับการสร้าง และยกเลิกการเชื่อมต่อ ส่วนเอเอกซ์ดีอาร์ถูกนำไปใช้กับการเข้ารหัสข้อมูลที่เกี่ยวกับการรับ-ส่งข้อมูล

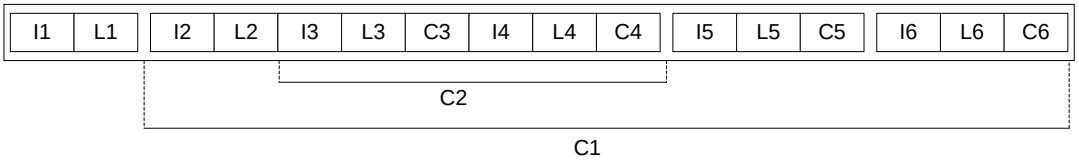
2.3.1. การเข้ารหัสข้อมูลแบบบีอีอาร์

โครงสร้างพื้นฐานการเข้ารหัสแบบบีอีอาร์ ประกอบด้วยสามส่วน ได้แก่ ตัวระบุชนิดข้อมูล (Identifier), ความยาวข้อมูล (Length), และเนื้อหา (Contents) ดังรูปที่ 2-8

Identifier	Length	Contents

รูปที่ 2-8 โครงสร้างพื้นฐานการเข้ารหัสแบบบีอีอาร์

เนื้อหาของการเข้ารหัสเป็นได้ทั้งแบบเนื้อหาเดียวไม่ซับซ้อน หรือแบบโครงสร้างพื้นฐานย่อยอื่นๆ แทรกอยู่ในเนื้อหาหลักก็ได้ ดังรูปที่ 2-9



รูปที่ 2-9 ตัวอย่างการเข้ารหัสแบบบีอีอาร์

โดยที่ Ix = ไบต์ Identifier ลำดับที่ x

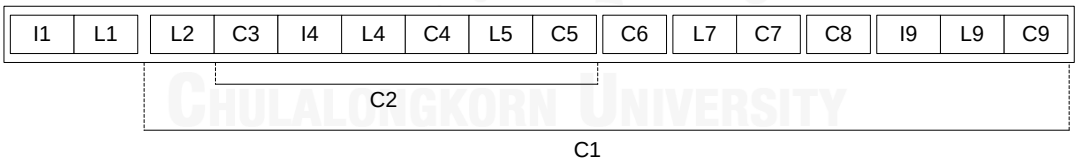
Lx = ไบต์ หรือกลุ่มไบต์ Length ลำดับที่ x

Cx = ไบต์ หรือกลุ่มไบต์ Contents ลำดับที่ x

รายละเอียดปลีกย่อยการเข้ารหัสแบบบีอีอาร์ของชนิดข้อมูลต่างๆ สามารถดูได้จาก [10]

2.3.2. **การเข้ารหัสข้อมูลแบบเอเอกซ์ดีอาร์**

การเข้ารหัสข้อมูลแบบเอเอกซ์ดีอาร์นั้นมาจากพื้นฐานเดียวกันกับบีอีอาร์ แต่มีข้อได้เปรียบตรงที่มันไม่ต้องเข้ารหัส ตัวระบุชนิดข้อมูล และ/หรือความยาวข้อมูล หรือเข้ารหัสตัวระบุชนิดข้อมูล และ/หรือความยาวข้อมูลเฉพาะกรณีที่จำเป็นจริงๆ เนื่องจากได้กำหนดวากยสัมพันธ์แบบนามธรรมนาม (Abstract Syntax) ขึ้นเพื่อให้อุปกรณ์ทั้งสองฝั่งเข้าใจตรงกัน จึงทำให้ใช้จำนวนไบต์ข้อมูลที่น้อยกว่า โดยมีลักษณะการเข้ารหัสดังรูปที่ 2-10



รูปที่ 2-10 ตัวอย่างการเข้ารหัสแบบเอเอ็กดีอาร์

โดยที่ Ix = ไบต์ Identifier ลำดับที่ x

Lx = ไบต์ หรือกลุ่มไบต์ Length ลำดับที่ x

Cx = ไบต์ หรือกลุ่มไบต์ Contents ลำดับที่ x

วากยสัมพันธ์แบบนามธรรมนามที่มาตรฐานดีแอลเอ็มเอส/โคเซมใช้นั้นถูกระบุไว้ในหัวข้อ 9.5 ของ [3] ส่วนรายละเอียดปลีกย่อยการเข้ารหัสแบบเอเอกซ์ดีอาร์ของชนิดข้อมูลต่างๆ สามารถดูได้จาก [11]

2.4. การสื่อสารผ่านสายไฟฟ้าส่งกำลัง

การสื่อสารผ่านสายไฟฟ้าส่งกำลัง (Power Line Communication, PLC) [12-14] เป็นเทคโนโลยีการสื่อการที่ส่งข้อมูลผ่านสายไฟฟ้าส่งกำลังที่เดินสายไว้อยู่แล้ว นั่นคือเราสามารถที่จะส่งได้ทั้งกำลังไฟฟ้า และส่งข้อมูลได้ในเวลาเดียวกัน การสื่อสารผ่านสายไฟฟ้าส่งกำลังแบ่งได้สองประเภทตามความกว้างของแถบความถี่ที่ใช้งาน คือ

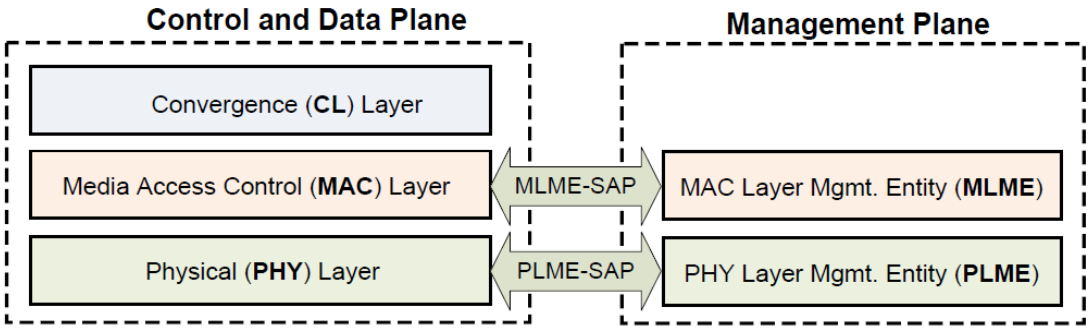
1. การสื่อสารผ่านสายไฟฟ้าส่งกำลังแบบช่วงความถี่แคบ (Narrowband PLC) ถูกใช้งานในช่วงความถี่ 3 - 500 กิโลเฮิรตซ์ มีอัตราการส่งข้อมูลต่ำไม่เกินประมาณ 100 กิโลบิตต่อวินาที แต่สามารถส่งข้อมูลได้ไกลหลายกิโลเมตร เมื่อใช้ร่วมกับอุปกรณ์ทวนสัญญาณ

2. การสื่อสารผ่านสายไฟฟ้าส่งกำลังแบบช่วงความถี่กว้าง (Broadband PLC) ถูกใช้งานในช่วงความถี่ 1.8 - 250 เมกะเฮิรตซ์ มีอัตราการส่งข้อมูลสูงถึงประมาณ 100 เมกะบิตต่อวินาที แต่ส่งข้อมูลได้ไม่ไกล

มาตรฐานของการสื่อสารผ่านสายไฟฟ้านั่นมีหลายมาตรฐานด้วยกัน เช่น G3-PLC, PRIME, IEEE P1901.2 ฯลฯ แต่ละมาตรฐาน จะใช้ช่วงความถี่ในการทำงาน, เทคโนโลยีการกล้ำ (Modulation) ต่างกัน ส่งผลให้อัตราการรับส่งข้อมูลแตกต่างกันด้วย แต่ในการวิจัยนี้จะยึดตามมาตรฐานไพร์ม (PoweRline Intelligent Metering Evolution, PRIME) เนื่องเป็นมาตรฐานที่ใช้กันมานานแล้ว และเป็นที่ยอมรับอย่างกว้างขวาง นอกจากนี้ไพร์มเป็นการสื่อสารผ่านสายไฟฟ้าส่งกำลังแบบช่วงความถี่แคบ จึงสามารถสื่อสารกันได้ระยะไกลหลายกิโลเมตรเหมาะแก่การใช้ในการสื่อสารในระบบไฟฟ้าส่งกำลังในปัจจุบัน

ไพร์มเป็นมาตรฐานที่นิยามรายละเอียดต่างๆ ในชั้นกายภาพ และชั้นการควบคุมการเข้าถึงตัวกลาง (Media Access Control, MAC) ซึ่งจะใช้เทคโนโลยีล่าสุด เพื่อรองรับการใช้งานในอนาคต ไพร์มเป็นมาตรฐานเปิด สามารถให้ผู้ผลิตอุปกรณ์เข้าร่วมได้ เพื่อที่ว่าอุปกรณ์ที่รองรับไพร์ม จะสามารถทำงานร่วมกันได้

ไพร์มใช้เทคโนโลยีการกล้ำแบบ OFDM (Orthogonal Frequency Division Multiplexing) ซึ่งกระจายข้อมูลในโดเมนเวลาไปในแถบความถี่แคบๆ หลายช่องความถี่ซึ่งตั้งฉากกันโดยใช้การแปลง IFFT โพโทคอลไพร์มใช้แถบความถี่ในช่วง CENELEC-A (3 ถึง 95 kHz) มีคลื่นความถี่ย่อย (Subcarrier) จำนวน 97 ตัว แต่ละคลื่นความถี่ย่อยห่างกัน 488.28125 Hz มีมีอัตราแลกเปลี่ยนข้อมูล 20 ถึง130 กิโลบิตต่อวินาทีโดยประมาณ ขึ้นอยู่กับวิธีการเข้ารหัสข้อมูล



รูปที่ 2-11 โครงสร้างลำดับชั้นที่ไพร์มกำหนดไว้ในมาตรฐาน

โครงสร้างลำดับชั้นที่ไพร์มกำหนดไว้ดังรูปที่ 2-11 และมีหน้าที่ ดังนี้

1. กลุ่มควบคุม และข้อมูล

1.1. ชั้นกายภาพ (Physical Layer, PL) มีหน้าที่ รับและส่งข้อมูลระหว่างจุดต่อ (Node) ที่อยู่ใกล้เคียง โดยใช้ OFDM

1.2. ชั้นการควบคุมการเข้าถึงตัวกลาง (Media Access Control Layer, MAC Layer) มีหน้าที่ ให้บริการคำสั่งต่างๆ ภายในชั้นการควบคุมการเข้าถึงตัวกลาง เช่นคำสั่ง การเข้าถึงระบบ, จองช่วงความถี่ของสื่อกลางในการส่งข้อมูล, สร้างและดูแลการเชื่อมต่อ

1.3. ชั้นคอนเวอร์เจนซ์ (Convergence Layer, CL) มีหน้าที่จัดสรรการเชื่อมโยงระหว่าง ชั้นบนที่มีโพรไฟล์ที่แตกต่างกัน ให้เข้ากันได้กับชั้นการควบคุมการเข้าถึงตัวกลางของมาตรฐานไพร์ม ในชั้นคอนเวอร์เจนซ์ มีคำสั่งบริการเฉพาะสำหรับแต่ละโพรไฟล์ที่รองรับ เช่น IPv4, IPv6 และ IEC 61334-4-32 เป็นต้น

2. กลุ่มการจัดการ

2.1. ตัวจัดการภายในชั้นกายภาพ มีหน้าที่ เก็บคุณสมบัติที่สำคัญต่างๆ ภายในชั้นกายภาพ ซึ่งเป็นส่วนหนึ่งของฐานข้อมูลสารสนเทศของการสื่อสารผ่านสายไฟฟ้าส่งกำลัง (PLC Information Base, PIB) คุณสมบัติในชั้นกายภาพ จะถูกแบ่งออกเป็น 2 กลุ่มใหญ่ๆ ได้แก่

2.1.1. กลุ่มทางสถิติ

2.1.2. กลุ่มทางการดำเนินการ

คุณสมบัติต่างๆ สามารถถูกเรียกโดยคำสั่งบริการ PLME_GET

2.2. ตัวจัดการภายในชั้นการควบคุมการเข้าถึงตัวกลาง มีหน้าที่ เก็บคุณสมบัติที่สำคัญต่างๆภายในชั้นการควบคุมการเข้าถึงตัวกลาง มีการนำคุณสมบัติในชั้นการควบคุมการเข้าถึงตัวกลางบางส่วนมาจาก PIB ซึ่งส่วนหนึ่งเป็นคุณสมบัติที่เกี่ยวข้องกับพฤติกรรมการทำงานของชั้นการควบคุมการเข้าถึงตัวกลาง อีกส่วนหนึ่งเป็นคุณสมบัติ และตัวแปรที่สร้างขึ้นเฉพาะชั้นการควบคุมการเข้าถึงตัวกลาง โดยทั่วไปแล้ว ตัวจัดการและดำเนินการยอมให้ค่าคุณสมบัติต่างๆ นั้นเปลี่ยนแปลงในขณะที่กำลังดำเนินงานอยู่โดยอัตโนมัติได้ นอกจากนั้นตัวจัดการ และดำเนินการก็สามารถเปลี่ยนแปลงคุณสมบัติเหล่านี้ได้ผ่านทางคำสั่งบริการ MLME_GET และ MLME_SET

2.2.1. ตัวแปรเฉพาะของชั้นการควบคุมการเข้าถึงตัวกลาง แบ่งได้ 2 กลุ่ม ได้แก่

2.2.1.1.ตัวแปรที่สามารถอ่านเขียนได้

2.2.1.2.ตัวแปรที่อ่านได้อย่างเดียว

2.2.2. คุณสมบัติของชั้นการควบคุมการเข้าถึงตัวกลาง แบ่งออกได้เป็น 4 กลุ่ม ได้แก่

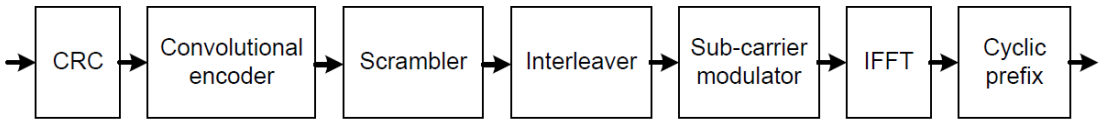
2.2.2.1.กลุ่มพฤติกรรมการทำงาน (เป็นส่วนหนึ่งของ PIB)

2.2.2.2.กลุ่มทางสถิติ

2.2.2.3.กลุ่มรายชื่อของคุณสมบัติ

2.2.2.4.กลุ่มการกระทำต่างๆ

2.4.1. **ชั้นกายภาพ**



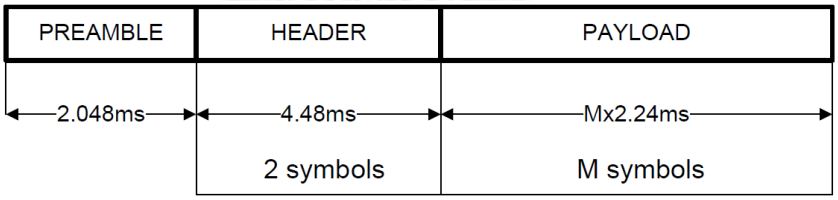
รูปที่ 2-12 ภาพรวมขั้นตอนการประมวลข้อมูลของชั้นกายภาพ

ชั้นกายภาพของไพร์ม ถูกออกแบบมาเพื่อให้รับ และส่งข้อมูลผ่านสายไฟฟ้าส่งกำลังซึ่งแต่เดิมถูกออกแบบมาเพื่อใช้ในการส่งกำลังไฟฟ้ากระแสสลับที่ความถี่ 50 - 60 เฮิรตซ์เท่านั้น ชั้นกายภาพของไพร์ม ใช้วิธีการที่มีความทนทาน, ความเร็วสูง แต่มีราคาต่ำ โดยใช้หลักการ OFDM ร่วมกับการแก้ไขความผิดพลาดไปข้างหน้า (Forward Error Correction, FEC) และการแทรกสลับ (Interleaving) มีขั้นตอนการประมวลข้อมูลของชั้นกายภาพ ดังรูปที่ 2-12

ด้านฝั่งส่ง เมื่อชั้นกายภาพรับข้อมูลอินพุทมาจากชั้นการควบคุมการเข้าถึงตัวกลาง จะนำข้อมูลนั้นมาเข้ารหัสการตรวจสอบด้วยส่วนซ้ำซ้อนแบบวน (Cyclic Redundancy Check, CRC) หลังจากนั้นก็จะเข้ารหัสแบบสังวัตนาการ (Convolution) เพื่อทำการแก้ไขความผิดพลาดไปข้างหน้า, สแครมเบลอร์ (Scrambler) และตัวแทรกสลับ (Interleaver) เป็นการจัดกลุ่ม และเพิ่มจำนวนบิต เพื่อหลีกเลี่ยงสัญญาณรบกวนแบบอิมพัลส์ (Impulsive Noise) ซึ่งอาจจะทำให้ข้อมูลผิดพลาดต่อเนื่องยาวนาน เอ้าท์พุทของกระบวนการข้างต้นจะถูกกล้ำโดยวิธีที่ต่างกันได้แก่ DBPSK, DQPSK หรือ D8PSK ซึ่งจะให้อัตราการส่งข้อมูลที่แตกต่างกันด้วย ดังตารางที่ 2-1 ขั้นตอนต่อไปคือ OFDM ซึ่งจะประกอบไปด้วยผลการแปรฟูเรียร์แบบเร็วผกผัน (Inverse Fast Fourier Transform, Inverse FFT) และตัวก่อกำเนิดช่วงเวลาป้องกันแบบวน (Cyclic Prefix Generator) เมื่อเสร็จสิ้นกระบวนการจะได้กรอบ (Frame) ชั้นกายภาพ ดังรูปที่ 2-13

ตารางที่ 2-1 เปรียบเทียบอัตราการส่งข้อมูลของแต่ละรูปแบบการกล้ำ รวมถึงการใช้งานการแก้ไขความผิดพลาดไปข้างหน้า

	DBPSK		DQPSK		D8PSK	
Convolutional Code (1/2)	On	Off	On	Off	On	Off
Information bits per subcarrier N_{BPSC}	0.5	1	1	2	1.5	3
Information bits per OFDM symbol N_{BPS}	48	96	96	192	144	288
Raw data rate (kbps approx)	21.4	42.9	42.9	85.7	64.3	128.6
Maximum MSDU length with 63 symbols (in bits)	3016	6048	6040	12096	9064	18144
Maximum MSDU length with 63 symbols (in bytes)”	377	756	755	1512	1133	2268



รูปที่ 2-13 กรอบข้อมูลชั้นกายภาพ

2.4.2. พรีแอมเบิล (Preamble)

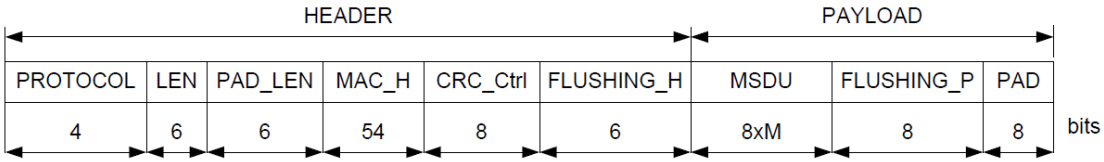
ทุกๆ การส่งข้อมูลจะต้องเริ่มต้นด้วยพรีแอมเบิลเพื่อซิงค์จุดเริ่มต้นของข้อมูล

2.4.3. ส่วนหัว (Header) และเพย์โหลด (Payload)

ส่วนหัวจะประกอบไปด้วย 2 สัญลักษณ์ของ OFDM ซึ่งใช้การกล้ำแบบ DBPSK และเข้าการรหัสแบบสังวัตนาการด้วยอัตรา 1/2 คือ มีอินพุทเข้ามา 1 บิต แต่มีเอ้าท์พุทออกไป 2 บิต เพื่อใช้เป็นการแก้ไขความผิดพลาดไปข้างหน้าเสมอ

ส่วนเพย์โหลดจะใช้การกล้ำแบบ DBPSK, DQPSK หรือ D8PSK ขึ้นอยู่กับการตั้งค่าในชั้นการควบคุมการเข้าถึงตัวกลาง โดยเลือกการกล้ำที่ส่งได้เร็วที่สุดแบบอัตโนมัติ รวมถึงการตัดสินใจว่าจะใช้การแก้ไขความผิดพลาดไปข้างหน้าด้วยหรือไม่ ซึ่งดูจากข้อมูลค่าผิดพลาดจากครั้งก่อนๆ หรือดูจากค่าอัตราส่วนของสัญญาณข้อมูลต่อสัญญาณรบกวน

สัญลักษณ์ของ OFDM 2 อันแรก ในหน่วยข้อมูลโพรโทคอลชั้นกายภาพ (Physical Protocol Data Unit, PPDU) นั้นก็คือ ส่วนหัว ซึ่งจะประกอบไปด้วยพาหะย่อยข้อมูล 84 พาหะ และพาหะย่อยนำร่องอีก 13 พาหะ หลังจากส่วนหัว ทุกสัญลักษณ์ของ OFDM จะเป็นส่วนของเพย์โหลด ซึ่งประกอบด้วยพาหะย่อยข้อมูล 96 พาหะ และพาหะย่อยนำร่องอีก 1 พาหะ ในแต่ละพาหะย่อยข้อมูลจะบรรจุ 1, 2 หรือ 3 บิต ขึ้นอยู่กับวิธีการกล้ำ ในแต่ละชุดของข้อมูล บิตที่มีนัยสำคัญสูงสุดจะถูกส่งก่อน



รูปที่ 2-14 ขบวนข้อมูลของชั้นกายภาพ ประกอบด้วยส่วนหัว และเพย์โหลด (ก่อนการเข้ารหัส)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DBPSK	DQPSK	D8PSK	RES	DBPSK_F	DQPSK_F	D8PSK_F	RES	RES	RES	RES	RES	RES	RES	RES	RES

รูปที่ 2-15 ฟิลด์ย่อยของฟิลด์ PROTOCOL

ส่วนหัว ประกอบด้วยข้อมูลส่วนหัวทั้งชั้นกายภาพ และชั้นการควบคุมการเข้าถึงตัวกลาง และประกอบด้วยฟิลด์ต่างๆ ดังรูปที่ 2-14 ดังนี้

- PROTOCOL จะบรรจุข้อมูลรูปแบบการกล้ำของเพย์โหลด ดังรูปที่ 2-15
- LEN คือ ความยาวของเพย์โหลดหลังเข้ารหัสในสัญญาลักษณ์ของ OFDM
- PED_LEN คือ ความยาวของฟิลด์ PAD ของเพย์โหลด (ก่อนการเข้ารหัส) อยู่ในรูปจำนวนไบต์
- MAC_H คือ ส่วนหัวของชั้นการควบคุมการเข้าถึงตัวกลาง
- CRC_Ctrl บรรจุผลรวมของการตรวจสอบด้วยส่วนซ้ำซ้อนแบบวน คำนวณโดยใช้ฟิลด์ PROTOCOL, LEN, PAD_LEN และ MAC_H เป็นตัวตั้งต้น

- FLUSHING_H บิตนี้ใช้ในการถอดรหัสแบบสังวัตนาการ ทุกบิตในฟิลด์นี้จะถูกตั้งเป็นศูนย์ เพื่อรีเซ็ตตัวเข้ารหัสแบบสังวัตนาการ

เพย์โหลด ประกอบด้วยฟิลด์ต่างๆ ดังนี้

- MSDU (MAC Service Data Unit) คือ หน่วยข้อมูลบริการของชั้นการควบคุมการเข้าถึงตัวกลาง ที่ยังไม่ได้เข้ารหัส

- FLUSHING_P บิตนี้ใช้ในการถอดรหัสแบบสังวัตนาการ ทุกบิตในฟิลด์นี้จะถูกตั้งเป็นศูนย์ เพื่อรีเซ็ตตัวเข้ารหัสแบบสังวัตนาการ ฟิลด์นี้จะปรากฏก็ต่อเมื่อใช้การแก้ไขความผิดพลาดไปข้างหน้า

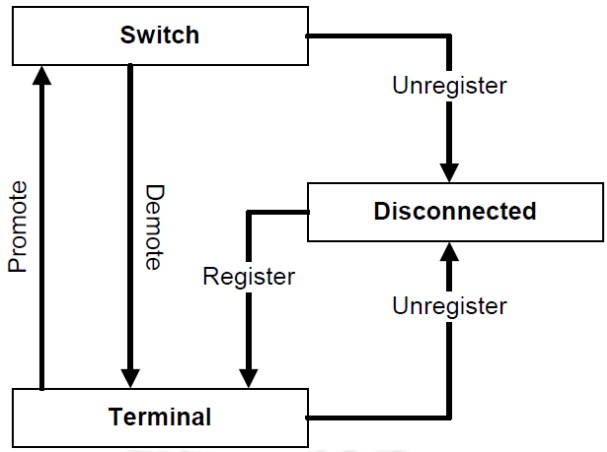
- PAD ใช้เติมลงในเพย์โหลดในกรณีที่ MSDU สั้น ทำให้เพย์โหลดมีความยาวไม่ครบ 1 สัญลักษณ์ของ OFDM แต่ละบิตของ PAD มีค่าเป็นศูนย์

สำหรับ OFDM ตามมาตรฐานไพร์ม ใช้สัญญาณนาฬิกา 250 กิโลเฮิรตซ์ มีพารามิเตอร์ต่างๆ ดังแสดงในตารางที่ 2-2

ตารางที่ 2-2 ค่าพารามิเตอร์ต่างๆ ที่ใช้ในชั้นกายภาพของมาตรฐานไพร์มซึ่งใช้ OFDM

Parameter	Value	
Base Band clock (Hz)	250000	
Subcarrier spacing (Hz)	488.28125	
Number of data subcarriers	84 (header)	96 (payload)
Number of pilot subcarriers	13 (header)	1 (payload)
FFT interval (samples)	512	
FFT interval (μs)	2048	
Cyclic Prefix (samples)	48	
Cyclic Prefix (μs)	192	
Symbol interval (samples)	560	
Symbol interval (μs)	2240	
Preamble period (μs)	2048	

2.4.4. **ชั้นการควบคุมการเข้าถึงตัวกลาง**



รูปที่ 2-16 การเปลี่ยนแปลงสถานะของจุดต่อบริการ

เครือข่ายย่อยของชั้นนี้ใช้โครงสร้างแบบต้นไม้ ซึ่งประกอบไปด้วยจุดต่อ 2 ชนิด ได้แก่

1. จุดต่อฐาน (Base Node) เปรียบเสมือนเป็นรากของโครงสร้างต้นไม้ มีหน้าที่เป็นจุดเริ่มต้นของเครือข่ายย่อย จัดการทรัพยากร และการเชื่อมต่อ ในหนึ่งเครือข่ายย่อยจะมีจุดต่อฐานได้เพียงจุดต่อเดียวเท่านั้น ในตอนเริ่มต้นของเครือข่ายย่อย จุดต่อฐานจะเป็นตัวสร้างเครือข่ายย่อย หลังจากนั้นจุดต่ออื่นๆ (จุดต่อบริการ) ต้องร้องขอเพื่อเข้าร่วมเครือข่ายย่อยนั้นๆ ไปยังจุดต่อฐาน

2. จุดต่อบริการ (Service Node) เปรียบเสมือนใบ หรือกิ่งของโครงสร้างต้นไม้ ในตอนเริ่มต้นจุดต่อบริการจะอยู่ในสถานะ "ไม่เชื่อมต่อ" จุดต่อบริการต้องร้องขอเข้าร่วมเครือข่ายย่อยกับจุดต่อฐาน เมื่อเข้าร่วมได้แล้ว จะเปลี่ยนสถานะเป็น "เครื่องปลายทาง" คือ สามารถรับส่งข้อมูลกับจุดต่อฐานได้ ในกรณีที่มีจุดต่อบริการอื่นมาขอเข้าร่วมเครือข่ายย่อยผ่านทางจุดต่อบริการที่มีสถานะเป็น "เครื่องปลายทาง" มันจะร้องขอไปยังจุดต่อฐาน เพื่อให้เลื่อนตำแหน่ง (Promote) ไปอยู่ในสถานะ "สวิตช์" เพื่อเป็นจุดเชื่อมต่อระหว่างจุดต่อฐาน กับจุดต่อบริการใหม่ที่ขอเข้าร่วมเครือข่ายย่อยได้ การเปลี่ยนแปลงสถานะของจุดต่อบริการ จะเป็นดังรูปที่ 2-16

2.4.5. **เหตุการณ์ที่เกี่ยวข้องกับการเปลี่ยนแปลงสถานะของจุดต่อบริการ**

เหตุการณ์ที่เกี่ยวข้องกับการเปลี่ยนแปลงสถานะของจุดต่อบริการมีดังนี้

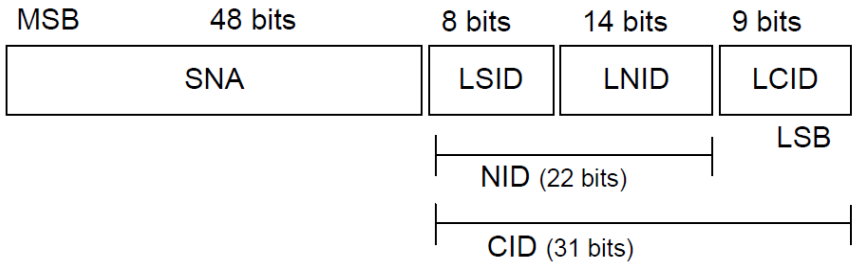
การลงทะเบียน (Registration) คือการที่จุดต่อบริการร้องขอเข้าร่วมเครือข่ายย่อยนั้นๆ ไปยังจุดต่อฐาน เมื่อจุดต่อฐานตอบรับการร้องขอ มันจะบันทึกจุดต่อบริการนั้นลงในรายชื่อจุดต่อที่ลงทะเบียนแล้ว ขั้นตอนการลงทะเบียนเป็นขั้นตอนที่เปลี่ยนสถานะของจุดต่อบริการจาก "ไม่เชื่อมต่อ" ไปเป็น "เครื่องปลายทาง"

การยกเลิกลงทะเบียน (Unregistration) คือ การที่จะนำจุดต่อบริการ ออกจากเครือข่ายย่อย รวมถึงนำชื่อของจุดต่อบริการนั้น ออกจากรายชื่อลงทะเบียนด้วย ในกรณีที่จุดต่อบริการหนึ่งที่มีสถานะเป็น "สวิตช์"เป็นจุดต่อที่ให้บริการจุดต่อบริการอื่นเพียงจุดเดียว เมื่อจุดต่อปลายทางยกเลิกการลงทะเบียนสำเร็จ จะทำให้จุดต่อบริการที่เป็นสวิตช์ถูกลดตำแหน่งเป็น "เครื่องปลายทาง" ด้วย การยกเลิกลงทะเบียนนั้น ทั้งจุดต่อฐาน และจุดต่อบริการสามารถร้องขอได้ทั้งคู่ การยกเลิกลงทะเบียนจะเป็นขั้นตอนที่เปลี่ยนสถานะของจุดต่อบริการจาก "เครื่องปลายทาง" หรือ "สวิตช์" ไปเป็น "ไม่เชื่อมต่อ"

การเลื่อนตำแหน่ง (Promotion) คือการที่จุดต่อบริการที่อยู่ในสถานะ "เครื่องปลายทาง" ต้องการเปลี่ยนสถานะไปเป็น "สวิตช์" อันเนื่องมาจากมีจุดต่อบริการอื่นร้องขอเข้าร่วมเครือข่ายผ่านตัวมัน มันจะร้องขอการเลื่อนตำแหน่งไปยังจุดต่อฐาน เมื่อเปลี่ยนสถานะสำเร็จ มันจะทำหน้าที่เป็นเหมือนกิ่งไม้ของโครงสร้างต้นไม้ ค่อยส่งต่อข้อมูลจากราก (จุดต่อฐาน) ไปยังใบ (จุดต่อบริการอื่นที่อยู่ในสถานะ "เครื่องปลายทาง") และในทางกลับกัน

การลดตำแหน่ง (Demotion) คือการที่จุดต่อบริการที่เป็นสวิตช์ร้องขอไปยังจุดต่อฐาน เพื่อขอลดตำแหน่งจาก "สวิตช์" ไปเป็น "เครื่องปลายทาง" อันเนื่องมาจากไม่มีจุดต่อบริการอื่นต่อจากตัวมันเองอีกแล้ว

2.4.6. **การอ้างที่อยู่**



รูปที่ 2-17 โครงสร้างของที่อยู่ภายในชั้น MAC

โครงสร้างของที่อยู่ภายในชั้น MAC เป็นไปตามรูปที่ 2-17

แต่ละจุดต่อมีที่อยู่ประจำจุดต่อ ตามมาตรฐาน IEEE Std 802-2001 เรียกว่า EUI-48 ซึ่งมีขนาด 48 บิต แต่ละจุดต่อได้รับที่อยู่ EUI-48 ตั้งแต่ตอนที่ถูกผลิตขึ้นมา และนำไปใช้เพื่อระบุตัวตนของจุดต่อนั้นๆ ในขั้นตอนลงทะเบียน ที่อยู่ EUI-48 ของจุดต่อฐานถูกใช้เป็นหมายเลขเครือข่ายย่อยที่จุดต่อฐานนั้นสร้างขึ้นด้วย และถูกเรียกว่า ที่อยู่เครือข่ายย่อย (Subnetwork Address, SNA)

ตัวระบุลำดับสวิตช์เฉพาะที่ (Local Switch Identifier, LSID) มีขนาด 8 บิต ใช้ระบุจุดต่อบริการที่อยู่ในสถานะ "สวิตช์" (ต่อไปจะเรียกว่า จุดต่อสวิตช์) ที่อยู่ในเครือข่ายย่อย ซึ่งจุด

ต่อฐานแจกจ่าย LSID ให้กับจุดต่อสวิตช์ในขั้นตอนการเลื่อนตำแหน่ง โดยค่า LSID = 0x00 ถูกใช้เพื่อระบุจุดต่อฐานเท่านั้น และค่า LSID = 0xFF จะหมายถึง ยังไม่ได้ถูกให้ค่า

จุดต่อบริการทุกจุดต่อจะได้รับตัวระบุจุดต่อเฉพาะที่ (Local Node Identifier, LNID) ในขั้นตอนการลงทะเบียน ซึ่งมีขนาด 14 บิต ตัวระบุจุดต่อเฉพาะที่นี้ จะสามารถซ้ำกันได้ ถ้าเป็นจุดต่อที่ต่ออยู่กันคนละจุดต่อสวิตช์ ตัวระบุจุดต่อเฉพาะที่ที่มีค่าเท่ากับ 0x0000 จะใช้ระบุจุดต่อสวิตช์เท่านั้น สำหรับจุดต่อบริการที่อยู่ในสถานะ "เครื่องปลายทาง" (ต่อไปจะเรียกว่า จุดต่อเครื่องปลายทาง) จะใช้ตัวระบุจุดต่อเฉพาะที่ที่มีค่าตั้งแต่ 0x0001 ขึ้นไป การรวมกันของตัวระบุลำดับสวิตช์กับตัวระบุจุดต่อเฉพาะที่ จะเรียกว่าตัวระบุจุดต่อ (Node Identifier, NID) มีขนาด 22 บิต

ตัวระบุการเชื่อมต่อเฉพาะที่ (Local Connection Identifier, LCID) ขนาด 9 บิต ใช้ระบุการเชื่อมต่อภายในจุดต่อ เมื่อรวมตัวระบุจุดต่อเฉพาะที่ เข้ากับตัวระบุการเชื่อมต่อเฉพาะที่จะได้ ตัวระบุการเชื่อมต่อ (Connection Identifier, CID) ขนาด 31 บิต มีหน้าที่ในการระบุการเชื่อมต่อ 1 การเชื่อมต่อที่อยู่ภายในเครือข่ายย่อย การเชื่อมต่อใด ๆ จะถูกระบุได้ด้วยที่อยู่เครือข่ายย่อย และตัวระบุการเชื่อมต่อ ค่าของตัวระบุการเชื่อมต่อที่ถูกจองด้วยกฎดังนี้

ตัวระบุการเชื่อมต่อเฉพาะที่มีค่าเท่ากับ 0x000 ถึง 0x0FF ใช้สำหรับการเชื่อมต่อที่ถูกร้องขอโดยจุดต่อฐาน การจองจะถูกทำโดยจุดต่อฐาน

ตัวระบุการเชื่อมต่อเฉพาะที่มีค่าเท่ากับ 0x100 ถึง 0x1FF ใช้สำหรับการเชื่อมต่อที่ถูกร้องขอโดยจุดต่อบริการ การจองจะถูกทำโดยจุดต่อบริการ

2.4.7. **การเริ่มต้น และดูแลเครือข่ายย่อย**

หน้าที่ในการตั้ง และดูแลเครือข่ายย่อย จะเป็นของจุดต่อฐาน มีหน้าที่ต่าง ๆ ดังนี้

1. การส่งบีคอน (Beacon) จุดต่อฐาน และจุดต่อสวิตช์ทุกจุดต่อ จะส่งบีคอนถึงทุกจุดต่อในเครือข่ายย่อยนั้นๆ

2. การเลื่อนตำแหน่ง และลดตำแหน่งของจุดต่อบริการ จุดต่อเครื่องปลายทางจะร้องขอการเลื่อนตำแหน่งไปยังจุดต่อฐาน ซึ่งเป็นผู้จัดการดูแลตารางของจุดต่อสวิตช์ และส่งตัวระบุลำดับสวิตช์ให้กับจุดต่อเครื่องปลายทางที่กำลังเปลี่ยนเป็นจุดต่อสวิตช์ นอกจากนั้นจุดต่อฐานยังรับผิดชอบการลดตำแหน่งของจุดต่อสวิตช์ที่ลงทะเบียนแล้วอีกด้วย

3.การจัดการการลงทะเบียนของอุปกรณ์ เมื่อจุดต่อบริการที่ต้องการเข้าร่วมเครือข่ายย่อย ร้องขอการลงทะเบียนไปยังจุดต่อฐาน ซึ่งเป็นผู้ตัดสินใจ และจะตอบรับ หรือตอบปฏิเสธคำขอดังกล่าวทุกครั้งที่ตอบตกลงจุดต่อฐานจะส่งตัวระบุจุดต่อไปให้จุดต่อบริการนั้นๆ นอกจากนี้หน้าที่ในการยกเลิกลงทะเบียนก็เป็นของจุดต่อฐาน เช่นกัน

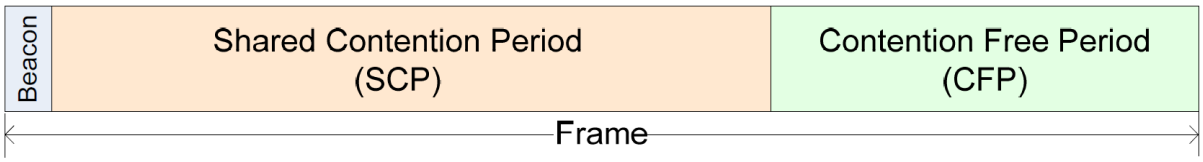
4. การตั้งค่า และจัดการการเชื่อมต่อ การเชื่อมต่อในชั้นการควบคุมการเข้าถึงตัวกลาง เป็นแบบที่ต้องสร้างช่องทางการเชื่อมต่อก่อนจึงจะสามารถส่งข้อมูลกันได้ เนื่องจากทุกจุดต่อจะต้องร้องขอการสร้างการเชื่อมต่อมายังจุดต่อฐานเสมอ จุดต่อฐานจึงมีบทบาทสำคัญในทุกเครือข่ายย่อย

5. การอนุญาตให้เข้าถึงช่องทางการส่งข้อมูล การใช้ช่องทางการส่งข้อมูลของอุปกรณ์ต่างๆ ตามมาตรฐานนี้ มีทั้งช่วงเวลาปลอดการช่วงชิง (Contention Free Period, CFP) และช่วงเวลาช่วงชิงร่วม (Shared Contention Period, SCP) จุดต่อฐานจะเป็นผู้กำหนดกลไกการใช้งานว่าจะใช้กลไกแบบไหน เวลาใด และนานเท่าใด นอกจากนี้ยังเป็นผู้อนุญาตให้อุปกรณ์ใดๆ ใช้ช่องทางการสื่อสารในช่วงเวลาปลอดการช่วงชิงอีกด้วย

6. การแจกจ่ายกุญแจที่ใช้ในการเข้ารหัสลับ (Encryption) คำสั่งควบคุมทุกคำสั่งในชั้นการควบคุมการเข้าถึงตัวกลาง จำเป็นที่จะต้องเข้ารหัสลับก่อนการส่ง รวมไปถึงข้อมูลก็อาจจะมีการเข้ารหัสลับด้วย กุญแจสำหรับการเข้ารหัสลับจะมีขนาด 128 บิต ได้จากการสุ่มค่า ซึ่งเป็นหน้าที่ของจุดต่อฐานที่ต้องสร้าง และแจกจ่ายไปทั้งเครือข่ายย่อย

7. การจัดการกลุ่มของการสื่อสารแบบกลุ่ม (Multicast) จุดต่อฐานจะเป็นผู้ดูแลกลุ่มของการสื่อสารแบบกลุ่มภายในเครือข่ายย่อย ซึ่งหมายถึงต้องตัดสินใจเกี่ยวกับการร้องขอเข้ากลุ่ม หรือออกจากกลุ่มของจุดต่อบริการ และต้องมีการสร้าง และส่งข้อความตอบรับแก่จุดต่อบริการนั้นๆ ด้วย

2.4.8. **การเข้าถึงช่องทางการสื่อสาร**



รูปที่ 2-18 โครงสร้างของกรอบชั้นการควบคุมการเข้าถึงตัวกลาง

โครงสร้างของกรอบชั้นการควบคุมการเข้าถึงตัวกลาง แสดงดังรูปที่ 2-18

การเข้าถึงช่องทางการสื่อสารของไพร์ม ใช้หลักการการเข้าถึงหลายทางแบบตรวจรู้พาหะ/ตรวจหาการชน (Carrier Sense Multiple Access with Collision Avoidance, CSMA/CA) ร่วมกับการรวมส่งสัญญาณแบบแบ่งเวลา (Time Division Multiplexing, TDM)

เวลาสำหรับการเข้าถึงช่องทางการสื่อสารถูกแบ่งออกเป็นหน่วยย่อยๆ เรียกว่ากรอบ (Frame) จุดต่อฐาน และจุดต่อบริการ สามารถเข้าถึงช่องทางการสื่อสารได้ทั้งช่วงเวลาการช่วงชิงร่วมและช่วงเวลาปลอดการช่วงชิง

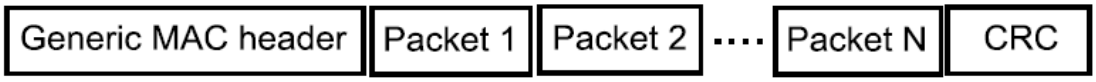
การเข้าถึงช่องทางการสื่อสารในช่วงเวลาปลอดการช่วงชิง อุปกรณ์นั้นๆ จำเป็นต้องร้องขอไปยังจุดต่อฐาน จุดต่อฐานอาจจะอนุญาต หรือไม่ ขึ้นอยู่กับสภาวะของช่องทางการสื่อสารในขณะนั้น ส่วนการเข้าถึงช่องทางการสื่อสารในช่วงเวลาการช่วงชิงร่วม อุปกรณ์นั้นๆ ไม่จำเป็นต้องขออนุญาต แต่ต้องส่งข้อมูลภายในระยะเวลาของช่วงเวลาการช่วงชิงร่วม

ในกรอบหนึ่งๆ ต้องมีอย่างน้อยหนึ่งหน่วยข้อมูลโพรโทคอลบีคอน (Beacon PDU, BPDU) , หนึ่งช่วงเวลา SCP แต่จะมีช่วงเวลา CFP หรือไม่ก็ได้ขึ้นอยู่กับข้อมูลใน BPDU ของกรอบนั้นๆ ความยาวของช่วงเวลา SCP จะถูกระบุอยู่ BPDU ด้วย

2.4.9. **รูปแบบของหน่วยข้อมูลโพรโทคอลชั้นการควบคุมการเข้าถึงตัวกลาง**

รูปแบบของหน่วยข้อมูลโพรโทคอลนั้นมีหลายชนิด ขึ้นอยู่กับจุดประสงค์การใช้ ดังนี้

2.4.9.1. **หน่วยข้อมูลโพรโทคอลทั่วไป**



รูปที่ 2-19 หน่วยข้อมูลโพรโทคอลทั่วไป

การแลกเปลี่ยนข้อมูลในชั้นการควบคุมการเข้าถึงตัวกลาง หน่วยข้อมูลโพรโทคอลทั่วไปถูกใช้เป็นส่วนใหญ่ และมีวัตถุประสงค์ในการใช้หลายประการ ทั้งส่งข้อมูล, ควบคุมการทำงาน และใช้สำหรับการทำงานทั่วๆ ไปในเครือข่ายย่อย ยกเว้นในบางกรณีที่ต้องการใช้หน่วยข้อมูลโพรโทคอลเฉพาะงาน หน่วยข้อมูลโพรโทคอลทั่วไปประกอบด้วย ส่วนหัว, กลุ่มข้อมูล (Packet) อย่างน้อยหนึ่งกลุ่ม และ CRC ดังรูปที่ 2-19

ส่วนหัวของหน่วยข้อมูลโพรโทคอลทั่วไป มีขนาด 3 ไบต์ และบรรจุรายละเอียดที่เกี่ยวข้องกับกลุ่มข้อมูล มีรายละเอียดดังนี้

1. ทิศทางการส่งข้อมูล

2. ระดับของอุปกรณ์ที่ส่ง (เทียบจากจุดต่อฐาน)

3. การตรวจสอบลำดับส่วนหัว (Header Check Sequence, HCS)

โครงสร้างกลุ่มข้อมูล ในแต่ละกลุ่มข้อมูลประกอบด้วยส่วนหัว และเพย์โหลด ดังรูปที่ 2-20

รูปที่ 2-20 โครงสร้างกลุ่มข้อมูล

ส่วนหัวของกลุ่มข้อมูลมีขนาด 6 ไบต์ ซึ่งบรรจุข้อมูลดังนี้

1. จุดเริ่มต้น และจุดสิ้นสุดของกลุ่มข้อมูล

2. ความสำคัญของการส่งของข้อมูล (Transmission priority of contents)

3. ความยาวข้อมูล

4. ลำดับข้อมูล ใช้ในการตรวจสอบลำดับการรับส่ง

2.4.9.2. **หน่วยข้อมูลโพรโทคอลขอเลื่อนตำแหน่ง (Promotion Needed PDU)**

จุดต่อบริการที่อยู่ในสถานะ "ไม่เชื่อมต่อ" จะเป็นผู้ส่งหน่วยข้อมูลโพรโทคอลชนิดนี้ เมื่อมันไม่ได้รับบีคอนภายในเวลาที่ได้ตั้งไว้ หน่วยข้อมูลโพรโทคอลขอเลื่อนตำแหน่งจะถูกส่งแบบถึงทุกจุดต่อโดยใช้วิธีการกล้ำที่ทนทานที่สุด เมื่อจุดต่อบริการอื่นที่อยู่ในสถานะ "เครื่องปลายทาง" ได้รับหน่วยข้อมูลโพรโทคอลนี้ มันจะเข้าสู่กระบวนการเลื่อนตำแหน่งเป็น "สวิตช์" และจะส่งบีคอนให้กับจุดต่อบริการที่อยู่ในสถานะ "ไม่เชื่อมต่อ" ที่ร้องขอมา หรือจุดต่อบริการอื่นๆ ที่เพิ่มเข้ามาใหม่

2.4.9.3. **หน่วยข้อมูลโพรโทคอลบีคอน (Beacon PDU)**

หน่วยข้อมูลโพรโทคอลนี้ จะถูกส่งโดยจุดต่อสวิตช์ทุกจุดต่อที่อยู่ในเครือย่อย รวมถึงจุดต่อฐานด้วย เพื่อเป็นการส่งข้อมูลบนโครงสร้างของกรอบข้อมูลสำหรับการเข้าถึงช่องทางการสื่อสารไปยังทุกอุปกรณ์ภายในเครือข่ายย่อย รายละเอียดของบีคอนมีดังนี้

1. โพรไฟล์ระบบความปลอดภัยที่เครือข่ายย่อยใช้

2. ที่อยู่ของผู้ส่งบีคอน

3. ที่อยู่ของเครือข่ายย่อย

4. ระดับของผู้ส่งบีคอนภายในเครือข่ายย่อย

5. รายละเอียดของโครงสร้างกรอบข้อมูล (ช่วง SCP และ CFP) ที่ต่อจากหน่วยข้อมูลโพรโทคอลบีคอนนี้

2.4.10. **กลุ่มข้อมูลควบคุมในชั้นการควบคุมการเข้าถึงตัวกลาง (MAC Control Packets)**

ข้อมูลควบคุมของชั้นการควบคุมการเข้าถึงตัวกลาง จะถูกส่งโดยใช้กลุ่มข้อมูลควบคุม กลุ่มข้อมูลควบคุมที่มีทิศทางลงจะถูกส่งจากจุดต่อฐาน และจุดต่อสวิตช์ ไปยังจุดต่อเครื่องปลายทางอื่นๆ ที่ต่ออยู่ ในทำนองเดียวกันกลุ่มข้อมูลควบคุมที่มีทิศทางขึ้นก็จะถูกส่งจากจุดต่อบริการไปยังจุดต่อฐาน หรือจุดต่อสวิตช์ที่จุดต่อบริการนั้นๆ ต่ออยู่

กลุ่มข้อมูลควบคุมนั้นมีหลายชนิด เพย์โหลดในแต่ละชนิดของกลุ่มข้อมูลที่แตกต่างกัน ก็จะต่างกันด้วย ชนิดของกลุ่มข้อมูลมีดังนี้

1. กลุ่มข้อมูลการจัดการลงทะเบียน (Registration Management, REG)

2. กลุ่มข้อมูลการจัดการการเชื่อมต่อ (Connection Management, CON)

3. กลุ่มข้อมูลการจัดการเลื่อนตำแหน่ง (Promotion Management, PRO)

4. กลุ่มข้อมูลแจ้งเตือนช่องบีคอน (Beacon Slot Indication, BSI)

5. กลุ่มข้อมูลการเปลี่ยนแปลงโครงสร้างของกรอบข้อมูล (Frame structure change, FRA)

6. กลุ่มข้อมูลร้องขอช่วงเวลาปลอดการช่วงชิง (Contention Free Period, CFP)

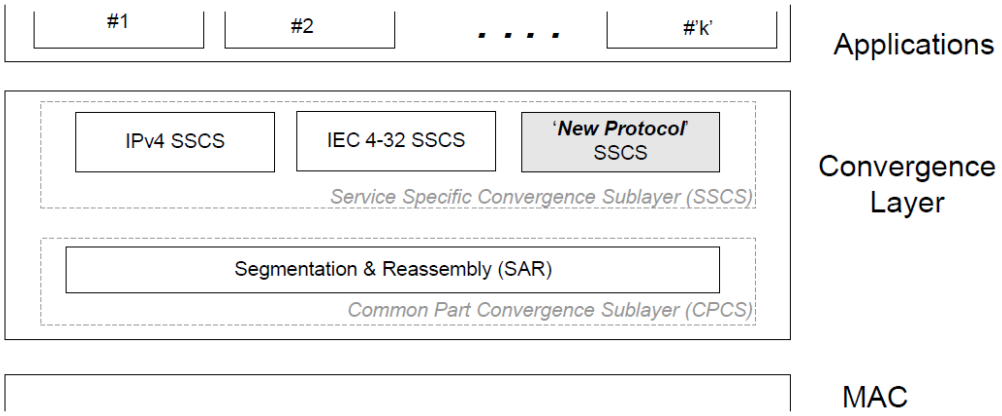
7. กลุ่มข้อมูลรักษาการเชื่อมต่อ (Keep Alive, ALV)

8. กลุ่มข้อมูลการจัดการการสื่อสารแบบกลุ่ม (Multicast Management, MUL)

9. กลุ่มข้อมูลการจัดการความทนทานชั้นกายภาพ (PHY Robustness Management, PRM)

10. กลุ่มข้อมูลข้อมูลรักษาความปลอดภัย (Security Information, SEC)

2.4.11. **ชั้นคอนเวอร์เจนซ์**



รูปที่ 2-21 โครงสร้างชั้นคอนเวอร์เจนซ์

ชั้นนี้แบ่งออกเป็น 2 ชั้นย่อย ดังรูปที่ 2-21 ได้แก่

1. ชั้นคอนเวอร์เจนซ์ย่อยส่วนใช้ร่วมกัน (Common Part Convergence Sublayer, CPCS)เชื่อมต่อชั้น MAC ด้วยชุดคำสั่งที่เหมือนกัน ไม่ว่าจะใช้โพรโทคอลในระดับบนใดกับชั้นย่อย SSCS

2. ชั้นคอนเวอร์เจนซ์ย่อยเฉพาะ (Service Specific Convergence Sublayer, SSCS) เชื่อมต่อชั้นย่อย CPCS กับโพรโทคอลระดับบน ชุดคำสั่งของ SSCS จะแตกต่างกันตามการเลือกโพรโทคอลระดับบน หรือกล่าวได้ว่า ชุดคำสั่งของ SSCS ที่บริการ IPv4 จะต่างกับชุดคำสั่งของ SSCS ที่บริการ IEC 4-32 อย่างสิ้นเชิง

2.4.11.1. **ชั้นคอนเวอร์เจนซ์ย่อยส่วนใช้ร่วมกัน**

ขณะนี้มีเพียงบริการเดียวเท่านั้น คือ การแยกส่วน และประกอบใหม่ (Segmentation and Reassembly, SAR)

หน่วยข้อมูลบริการ (SDU) ที่มีขนาดเกินขนาดของ 1 ชิ้นส่วนย่อย (Segment) ต้องถูกซอยเป็นหลายส่วนย่อยๆ ให้แต่ละส่วนมีขนาดไม่เกิน 1 ชิ้นส่วนย่อย จึงจะได้รับอนุญาตให้ส่งได้ เมื่อไปถึงจุดหมาย ชิ้นส่วนย่อยของหน่วยข้อมูลบริการจะถูกนำมาประกอบใหม่อีกครั้ง ก่อนที่จะถูกส่งขึ้นไปยังชั้นย่อย SSCS มาตรฐานไพร์ม กำหนดให้ใช้ขนาดของชิ้นส่วนย่อยร่วมกันสำหรับทุกๆ ชั้นย่อย SSCS เพื่อลดความซับซ้อนในการนำไปใช้จริง

2.4.11.2. **ชั้นคอนเวอร์เจนซ์ย่อยเฉพาะ IEC 61334-4-32**

ชั้นคอนเวอร์เจนซ์ย่อยจัดสรรการทำงานสำหรับชั้นโปรแกรมประยุกต์ที่ต้องการใช้บริการผ่านบริการตาม IEC 61334-4-32 ซึ่งเป็นบริการในชั้นการควบคุมการเชื่อมต่อเชิงตรรกะ

Logical Link Control, LLC) ที่ระบุไว้ในส่วนที่ 2 ของมาตรฐาน IEC 61334-4-32 ฉบับ 1996-09 นอกจากนี้ ไพร์มยังเพิ่มบริการที่นอกเหนือจาก IEC 61334-4-32 ขึ้นมา เพื่อช่วยในการแปลงโพร-โทคอล แบบไม่ต้องสร้างช่องทางการเชื่อมต่อ ให้เป็นแบบที่ต้องสร้างการเชื่อมต่อก่อน ให้เหมือนกับชั้นการควบคุมการเข้าถึงตัวกลางของไพร์ม

ในชั้นคอนเวอร์เจนซ์ย่อยเฉพาะนี้ จุดต่อบริการสามารถแลกเปลี่ยนข้อมูลได้เฉพาะกับจุดต่อฐานเท่านั้น ไม่สามารถแลกเปลี่ยนข้อมูลกับจุดต่อบริการอื่นได้ นอกจากนี้ยังมีรายละเอียดต่างๆ ดังนี้

แต่ละชั้นคอนเวอร์เจนซ์ย่อยเฉพาะนี้ของจุดต่อบริการจะสร้างการเชื่อมต่อในชั้น MAC เพื่อแลกเปลี่ยนข้อมูลแบบหนึ่งต่อหนึ่งกับจุดต่อฐาน

จุดต่อบริการมีหน้าที่ในการเริ่มต้นขอสร้างช่องทางการเชื่อมต่อกับจุดต่อฐานเท่านั้น จุดต่อฐานไม่สามารถเป็นผู้เริ่มต้นได้

เมื่อช่องทางการสื่อสารถูกสร้างขึ้นแล้ว จุดต่อฐานจะเป็นผู้เริ่มการส่งข้อมูลก่อนเสมอ

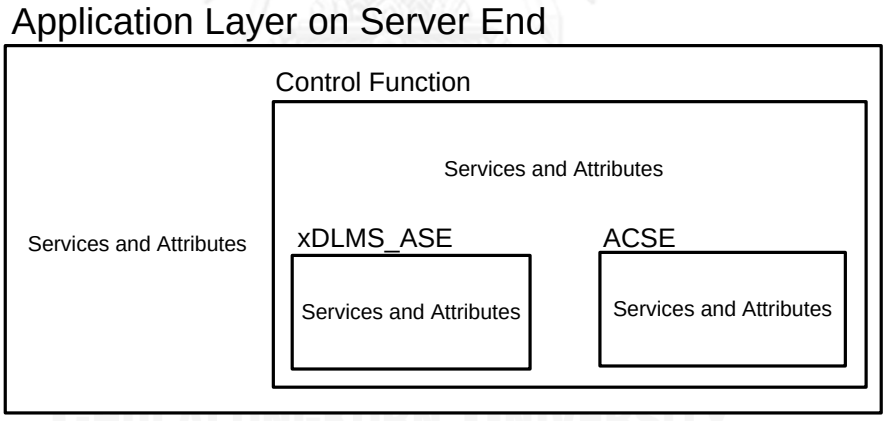
บทที่ 3

การออกแบบคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม

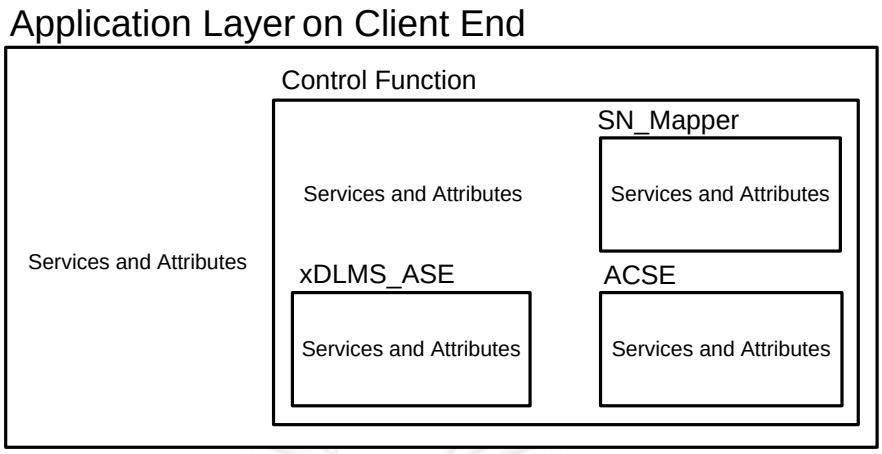
บทนี้จะกล่าวถึงแนวคิดในการออกแบบคลังโปรแกรม (Program Library) ชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส โคเซม (DLMS/COSEM) ทั้งฝั่งเซิร์ฟเวอร์ (Server) และฝั่งไคลเอนต์ (Client) แนวคิดในการนำมาใช้ และออกแบบเทรดจัดการชั้นโปรแกรมประยุกต์ การทำงานร่วมกันระหว่างเทรดหลัก (Main Thread) เทรดจัดการชั้นโปรแกรมประยุกต์ และเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง (Power Line Communication Modem Interface Thread) ผังงาน (Flow Chart) การทำงานของเทรดจัดการชั้นโปรแกรมประยุกต์ รวมถึงปัจจัยต่างๆ ที่มีผลต่อการเปลี่ยนแปลงสถานะของเทรดจัดการ

3.1. โครงสร้างคลังโปรแกรมชั้นโปรแกรมประยุกต์ของดีแอลเอ็มเอส/โคเซม

การสื่อสารระหว่างอุปกรณ์ 2 อุปกรณ์ใดๆ ตามมาตรฐานดีแอลเอ็มเอส/โคเซม จำเป็นต้องสื่อสารกันแบบเซิร์ฟเวอร์-ไคลเอนต์เสมอ การออกแบบคลังโปรแกรมชั้นโปรแกรมประยุกต์จึงต้องออกแบบเพื่อรองรับระบบการสื่อสารแบบนี้



รูปที่ 3-1 โครงสร้างคลาสชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์



รูปที่ 3-2 โครงสร้างคลาสชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์

ดังนั้น ภายในคลังโปรแกรมถูกแบ่งออกเป็น 2 คลาสใหญ่ที่บรรจุบริการ และตัวแปรที่สำคัญในการสื่อสารสำหรับอุปกรณ์ทั้งสองฝั่ง ได้แก่ คลาสชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ และคลาสชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ ทั้งสองคลาส ได้ถูกสืบทอดมาจากคลาสสำคัญต่างๆ ดังมีโครงสร้างคลาสแสดงในรูปที่ 3-1 และรูปที่ 3-2 ตามลำดับ คลาสชั้นโปรแกรมประยุกต์ทั้งสอง ถูกออกแบบให้มีรูปแบบโครงสร้าง และการไหลของข้อมูล (Data Flow) เดียวกันกับรูปที่ 2-7 ซึ่งเป็นไปตามมาตรฐานดีแอลเอ็มเอส/โคเซมเล่มสีเขียว (Green Book)

คลาสชั้นโปรแกรมประยุกต์ทั้ง 2 ฝั่ง ได้ถูกสืบทอดมาจากคลาส Control Function ของแต่ละฝั่ง คลาส Control Function ฝั่งเซิร์ฟเวอร์ได้ถูกสืบทอดมาจากคลาส xDLMS_ASE และ ACSE แต่ของฝั่งไคลเอนต์ได้ถูกสืบทอดเพิ่มมาอีก 1 คลาส นั่นคือ คลาส SN_Mapper

จากแนวคิดการสืบทอดคลาส (คลาสลูกสามารถเรียกใช้บริการ และตัวแปรต่างๆ ของคลาสแม่ที่เป็นแบบสาธารณะ (Public) และแบบป้องกัน (Protected) ได้ พร้อมทั้งอาจมีบริการและตัวแปรต่างๆ เป็นของตนเอง) คลาสชั้นโปรแกรมประยุกต์ทั้ง 2 จึงมีบริการ และตัวแปรที่เป็นแบบสาธารณะ และแบบป้องกัน ของคลาสแม่ที่อยู่ภายในทั้งหมด ยกตัวอย่างเช่น คลาส Control Function ฝั่งเซิร์ฟเวอร์ ซึ่งเป็นคลาสลูก จะสามารถเข้าถึงตัวแปร และบริการต่างๆ ที่เป็นแบบสาธารณะ และแบบป้องกันของคลาส xDLMS_ASE และ ACSE ซึ่งเป็นคลาสแม่ทั้งสองของมันได้ทั้งหมด

คลาสต่างๆ ที่ถูกกล่าวถึงในตอนต้นนั้น มีหน้าที่ดังนี้

คลาส xDLMS_ASE บรรจุบริการ และตัวแปรที่ใช้สำหรับรับ-ส่งข้อมูลระหว่างทั้ง 2 ฝั่ง เพื่อให้คลาส Control Function ได้สืบทอดไปใช้

คลาส ACSE บรรจุบริการ และตัวแปรที่ใช้สำหรับการสร้างช่องทางการเชื่อมต่อระหว่างฝั่งเซิร์ฟเวอร์ และฝั่งไคลเอนต์ เพื่อให้คลาส Control Function ได้สืบทอดไปใช้

คลาส SN_Mapper บรรจุบริการ และตัวแปรที่ใช้สำหรับแปลงบริการแบบชื่อตรรกะ (Logical Name) ไปเปลี่ยนแบบชื่อสั้น (Short Name) ในกรณีที่ฝั่งเซิร์ฟเวอร์เป็นแบบชื่อสั้น เพื่อให้คลาส Control Function ฝั่งไคลเอนต์ได้สืบทอดไปใช้

คลาส Control Function ได้เพิ่มส่วนตรวจสอบการทำงานของบริการต่างๆ ของคลาสแม่ เพื่อให้การใช้บริการต่างๆ ถูกต้อง และเป็นไปตามมาตรฐาน เช่น ส่วนตรวจสอบว่าสถานะขณะนั้นสามารถใช้บริการใดๆ ได้บ้าง และส่วนตรวจสอบว่าการเรียกใช้บริการต่างๆ ในครั้งนั้นๆ สำเร็จหรือไม่ เป็นต้น รวมถึงบรรจุตัวแปรโครงสร้างที่ใช้ในการเก็บข้อมูลที่ใช้ในการรับส่งด้วย

คลาสชั้นโปรแกรมประยุกต์ ได้เพิ่มบริการสำหรับเทรด (Thread) ที่ใช้จัดการการรับ-ส่งแบบบล็อกย่อยๆ เพื่อความสะดวกในการใช้งานของผู้ใช้คลังโปรแกรมนี้ ในกรณีข้อมูลที่ต้องการส่งมีขนาดใหญ่กว่าความสามารถของตัวกลางที่จะส่งออกไปได้ ข้อมูลจะถูกแบ่งออกเป็นบล็อกย่อยที่ฝั่งส่ง และไปประกอบเข้าด้วยกันอีกครั้งที่ฝั่งรับโดยอัตโนมัติ ก่อนที่จะถูกส่งไปยังชั้นตัวประมวลผลโปรแกรมประยุกต์ (Application Process) โดยการใส่เก็บไว้ในตัวแปรโครงสร้างที่ใช้ในการเก็บข้อมูลในการรับ-ส่งของคลาสโปรแกรมประยุกต์ ซึ่งจะมีขนาดใหญ่กว่าของคลาส Control Function เนื่องจากต้องใช้เก็บข้อมูลที่ได้จากการประกอบข้อมูลจากบล็อกย่อยๆ ด้วย และได้เพิ่มบริการที่ใช้เพื่อสั่งการเทรดเพื่อเรียกใช้บริการต่างๆ ของคลาส Control Function ด้วย

3.2. การออกแบบบริการต่างๆ ในชั้นโปรแกรมประยุกต์ของดีแอลเอ็มเอส/โคเซม

การออกแบบในส่วนนี้จะถูกแบบออกเป็น 2 ส่วน ได้แก่ การออกแบบบริการของ xDLMS_ASE และการออกแบบบริการของ ACSE

3.2.1. การออกแบบบริการของ xDLMS_ASE

ดำเนินการตามโครงสร้าง Abstract syntax of ACSE and COSEM APDUs ใน DLMS User Association, COSEM Architecture and Protocols, Seventh Edition (Green Book) หน้า 197 – 213 เป็นหลัก โดยวากยสัมพันธ์แบบนามธรรม (Abstract syntax) ที่ใช้กับ xDLMS_ASE ถูกเข้ารหัสแบบ A-XDR

3.2.2. การออกแบบบริการของ ACSE

ดำเนินการเช่นเดียวกันกับข้อ 3.2.1 แต่ใช้การเข้ารหัสทั้งแบบ BER และ A-XDR

3.3. เทรดที่เกี่ยวข้อง

ในการใช้งานคลังโปรแกรมจำเป็นต้องใช้ระบบปฏิบัติการที่รองรับเทรด เพราะได้ออกแบบการใช้งานคลาสต่างๆ ผ่านหลายเทรด ดังนี้

3.3.1. **เทรดหลัก**

เทรดหลัก ซึ่งก็คือตัวประมวลผลโปรแกรมประยุกต์ ทำหน้าที่เป็นผู้สั่งการ ควบคุม และผสานงานกับเทรดอื่นที่เกี่ยวข้อง ได้แก่ เทรดจัดการชั้นโปรแกรมประยุกต์ และเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง

3.3.2. **เทรดจัดการชั้นโปรแกรมประยุกต์**

ใช้เพื่อจัดการการรับส่งข้อมูลที่ยาวกว่าบัฟเฟอร์ (Buffer) ของทั้งฝั่งไคลเอนต์ และเซิร์ฟเวอร์จะรองรับได้ ในการรับส่งแต่ละหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ (APDU) ในกรณีการส่งข้อมูลออก เทรดจัดการชั้นโปรแกรมประยุกต์จะทำการแบ่งข้อมูลที่มีขนาดยาวกว่า ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง ออกเป็นบล็อกข้อมูลย่อยๆ ก่อนเข้ารหัสและส่งทีละบล็อก หรือในทางกลับกันกรณีของการรับข้อมูลเข้า เทรดจัดการชั้นโปรแกรมประยุกต์จะรับข้อมูลทีละบล็อก ถอดรหัส และประกอบบล็อกข้อมูลย่อยๆ เข้าด้วยกัน และในกรณีที่ข้อมูลมีขนาดสั้นกว่า ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง เทรดจัดการชั้นโปรแกรมประยุกต์จะทำหน้าที่เป็นเพียงแค่ตัวส่งผ่านข้อมูลเท่านั้น โดยที่ผู้ใช้ไม่จำเป็นต้องทราบกลไกการทำงานของการแยกส่วน (Segmentation) และการประกอบข้อมูล (Assembly) เลย

3.3.3. **เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง**

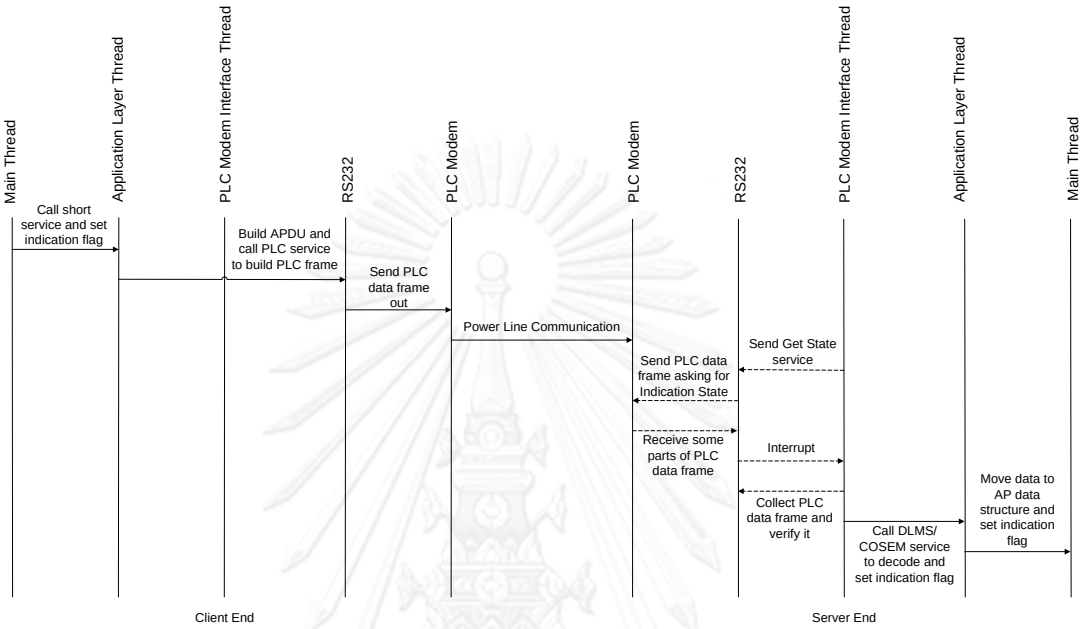
ใช้เพื่อสื่อสารกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังตามมาตรฐานไพร์ม (PRIME) ผ่านทางพอร์ตอนุกรมแบบ RS232 โดยมันจะคอยตรวจสอบ และรวบรวมข้อมูลที่เข้าใหม่จากโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ทุกๆ ช่วงเวลาที่ตั้งเอาไว้ เมื่อได้ข้อมูลครบสมบูรณ์แล้วมันจะส่งต่อให้กับเทรดจัดการชั้นโปรแกรมประยุกต์ เพื่อประมวลผลต่อไป

3.4. การเข้าจังหวะ (Synchronization) ของเทรด

การเข้าจังหวะกันของเทรด คือ การสื่อสารกันระหว่างแต่ละเทรด เพื่อให้การทำงานเป็นไปอย่างราบลื่น เนื่องจากการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม มีเทรดที่เกี่ยวข้องหลายเทรด ดังนั้นจึงต้องมีการเข้าจังหวะกัน ซึ่งจะใช้วิธีการตั้งค่าตัวบ่งชี้ (Flag) เป็นหลัก การเข้าจังหวะกันของเทรดสามารถแบ่งออกตามลักษณะการส่งข้อมูลได้ 2 แบบ คือ การส่งข้อมูลสมบูรณ์ในคราวเดียว และการส่งข้อมูลแบบบล็อกข้อมูลย่อย

3.4.1. **การส่งข้อมูลสมบูรณ์ในคราวเดียว**

กรณีนี้เกิดขึ้นเมื่อฝั่งไคลเอนต์ หรือเซิร์ฟเวอร์ฝั่งใดฝั่งหนึ่ง (แต่ในการอธิบายนี้จะพูดถึงฝั่งไคลเอนต์เป็นหลัก) ต้องการที่จะส่งข้อมูลที่มีขนาดไม่เกิน ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง ซึ่งบริการในคลาสชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ ที่ถูกเรียกใช้จะเป็นบริการในกลุ่มที่ 1 ดังตารางที่ 3-1



รูปที่ 3-3 การทำงานร่วมกันของเทรด กรณีส่งข้อมูลสมบูรณ์ในคราวเดียว

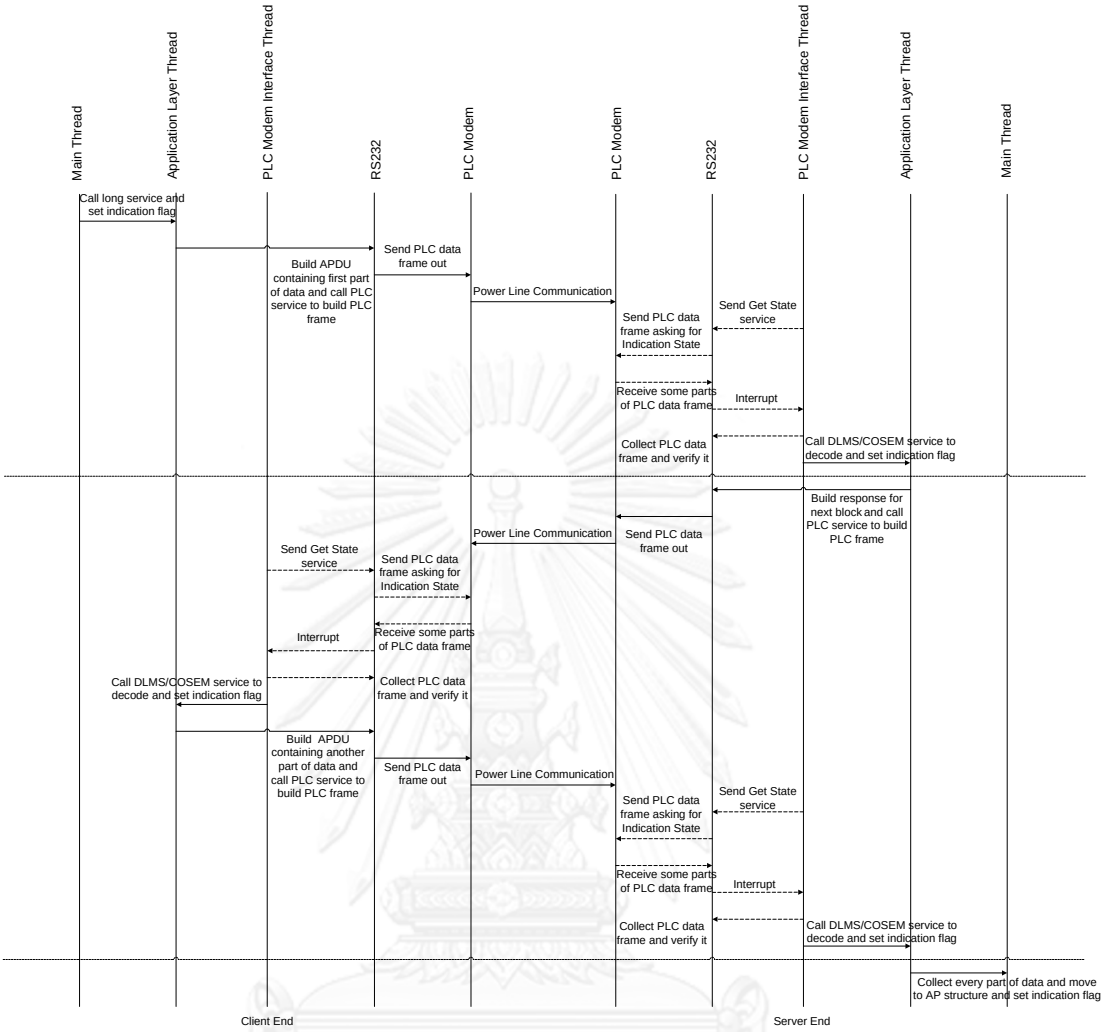
จากรูปที่ 3-3 เริ่มต้นที่เทรดหลักฝั่งไคลเอนต์ ต้องการส่งข้อมูลไปยังอุปกรณ์ฝั่งเซิร์ฟเวอร์ โดยมันจะใส่ข้อมูลที่ต้องการส่งไว้ในโครงสร้างข้อมูลสำหรับบริการนั้นๆ จากนั้นเรียกใช้บริการนั้นๆ ของคลาสชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ เพื่อตั้งค่าตัวบ่งชี้ให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า มันต้องสร้าง APDU ของคำสั่งดังกล่าว เมื่อเทรดจัดการชั้นโปรแกรมประยุกต์สร้าง APDU เสร็จเรียบร้อย จากนั้นมันจะเรียกใช้บริการของคำสั่ง Unicast จากคลาส PLC_API เพื่อสร้างกรอบข้อมูลการสื่อสารผ่านสายไฟฟ้าส่งกำลัง (PLC Data Frame) (กรอบข้อมูลการสื่อสารผ่านสายไฟฟ้าส่งกำลังของคำสั่งต่างๆ ที่ใช้ควบคุมโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ได้ถูกกำหนดใน [15]) และส่งข้อมูลออกไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เพื่อส่งข้อมูลไปยังอุปกรณ์อีกฝั่งหนึ่ง

เมื่อโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังฝั่งเซิร์ฟเวอร์ได้รับข้อมูลแล้ว มันจะตั้งค่าตัวบ่งชี้ของการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ว่ามีข้อมูลใหม่เข้ามา เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังฝั่งเซิร์ฟเวอร์จำเป็นต้องสอบถามไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังอยู่เรื่อยๆ โดยการส่งกรอบข้อมูลการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ที่มีชื่อว่า Prime Get State เมื่อโมเด็มรับกรอบข้อมูลแล้ว มันจะส่งกรอบข้อมูลที่บรรจุตัวบ่งชี้ของการสื่อสารผ่านสายไฟฟ้าส่ง

กำลัง กลับมาบอกว่ามีข้อมูลใหม่เข้ามาหรือไม่ เทรดติดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง จะเรียกใช้บริการเพื่อส่งคำสั่งไปยังโมเด็ม ให้โมเด็มส่งกรอบข้อมูลการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ที่บรรจุข้อมูลใหม่มาให้ แต่กรอบข้อมูลการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ที่มาจากโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ผ่านทางพอร์ตอนุกรมแบบ RS232 จะมาไม่ครบในครั้งเดียว เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง จึงต้องนำชิ้นส่วนข้อมูล (Data Part) มาต่อกันเพื่อให้ได้กรอบข้อมูลที่สมบูรณ์ หลังจากได้รับกรอบข้อมูลที่สมบูรณ์แล้ว เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังจะถอดกรอบข้อมูลออก ให้เหลือแต่หน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ (APDU) เท่านั้น และจะเรียกใช้บริการชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โค-เซม เพื่อถอดรหัสหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ และตั้งค่าตัวบ่งชี้ชั้นโปรแกรมประยุกต์ เพื่อบอกให้เทรดจัดการชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์รับทราบว่า มีข้อมูลเข้ามาใหม่ให้ไปตรวจสอบด้วย ซึ่งเทรดจัดการชั้นโปรแกรมประยุกต์จะตรวจสอบว่าคำสั่งที่ถูกส่งเข้ามานั้น จะต้องถอดรหัสด้วยบริการกลุ่มไหน ซึ่งในกรณีนี้จะเป็นกลุ่มที่ 1 มันจึงย้ายข้อมูลไปไว้ในโครงสร้างข้อมูลของบริการนั้นๆ และตั้งค่าตัวบ่งชี้ตัวประมวลผลโปรแกรมประยุกต์ เพื่อแจ้งให้เทรดหลักทราบว่ามีข้อมูลใหม่เข้ามา โดยเทรดหลักจะต้องคอยตรวจสอบตัวบ่งชี้ตัวประมวลผลโปรแกรมประยุกต์อยู่เรื่อยๆ

3.4.2. **การส่งข้อมูลแบบบล็อกข้อมูลย่อย**

กรณีนี้เกิดขึ้นเมื่อฝั่งไคลเอนต์ หรือเซิร์ฟเวอร์ฝั่งใดฝั่งหนึ่ง (แต่ในการอธิบายนี้จะพูดถึงฝั่งไคลเอนต์เป็นหลัก) ต้องการที่จะส่งข้อมูลที่มีขนาดใหญ่เกิน ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง ซึ่งบริการในคลาสชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์ ที่ถูกเรียกใช้จะเป็นบริการในกลุ่มที่ 2 และกลุ่มที่ 3 ดังตารางที่ 3-2 และตารางที่ 3-3 ตามลำดับ



รูปที่ 3-4 การทำงานร่วมกันของเทรด กรณีส่งข้อมูลแบบบล็อกข้อมูลย่อย

จากรูปที่ 3-4 ในช่วงเริ่มต้นของกรณี (ก่อนเส้นประแนวนอนเส้นที่ 1) เหมือนกับหัวข้อ 3.4.1 แต่จะต่างกันที่ เทรดจัดการฝั่งไคลเอนต์จะส่งคำสั่งโดยใช้บริการในกลุ่มที่ 2 มาแทน เพื่อสร้างหน่วยข้อมูลโพรโตคอล ซึ่งบรรจุข้อมูลที่ต้องการส่งเพียงบางส่วนเท่านั้น (บล็อกข้อมูลย่อย) และเมื่อเทรดจัดการชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ตรวจพบคำสั่ง ที่ต้องใช้บริการในกลุ่มที่ 2 เพื่อถอดรหัส มันจะตอบกลับด้วยคำสั่งที่สร้างโดยใช้บริการในกลุ่มที่ 3 เพื่อแจ้งให้เทรดฝั่งไคลเอนต์ทราบว่ามันได้รับข้อมูลบล็อกล่าสุดที่ส่งมาให้แล้ว และให้เริ่มส่งบล็อกต่อไปได้

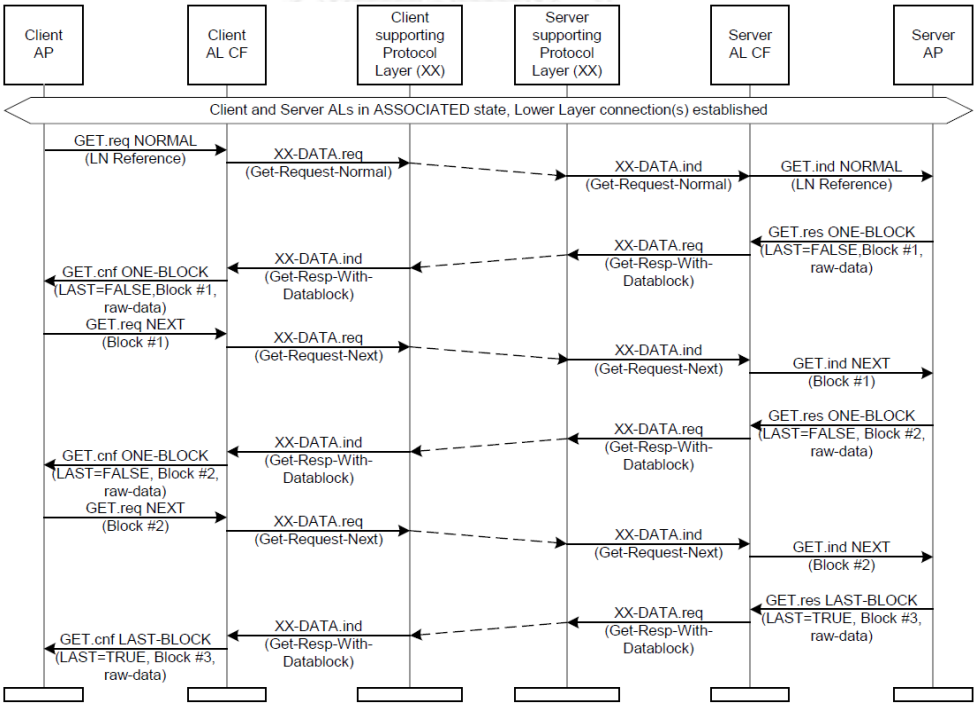
โดยขั้นตอนการรับข้อมูลของฝั่งไคลเอนต์ก็จะเหมือนกับฝั่งเซิร์ฟเวอร์ เมื่อเทรดจัดการฝั่งไคลเอนต์ตรวจพบคำสั่งที่ต้องใช้บริการในกลุ่มที่ 3 เพื่อถอดรหัส มันจะเรียกใช้บริการในกลุ่มที่ 2 เพื่อสร้างหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ที่บรรจุบล็อกข้อมูลในส่วนถัดไป และส่งออกไปยังฝั่งเซิร์ฟเวอร์อีกครั้ง

ขั้นตอนที่อยู่ระหว่างเส้นประแนวนอนจะเกิดขึ้นซ้ำๆ จนกว่าเทรดจัดการชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์จะได้รับข้อมูลบล็อกสุดท้าย และในระหว่างการรับข้อมูลบล็อกย่อย มันจะนำข้อมูลบล็อกล่าสุดที่ได้ เก็บไว้ในโครงสร้างข้อมูลชั้นโปรแกรมประยุกต์ก่อน โดยต่อเข้ากับข้อมูลที่มีอยู่เดิม เมื่อมันได้รับข้อมูลที่บล็อกข้อมูลย่อยครบแล้ว มันจะตั้งค่าตัวบ่งชี้ตัวประมวลผลโปรแกรมประยุกต์ เพื่อให้เทรดหลักทราบว่ามีข้อมูลใหม่เข้ามา โดยเทรดหลักจะต้องคอยตรวจสอบตัวบ่งชี้ตัวประมวลผลโปรแกรมประยุกต์อยู่เรื่อยๆ

ในกรณีเทรดหลัก หรือตัวประมวลผลชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ รับข้อมูลแล้ว ต้องการส่งข้อมูลกลับ ขั้นตอนต่างๆ ก็จะมีลักษณะเหมือนกับขั้นตอนด้านบน

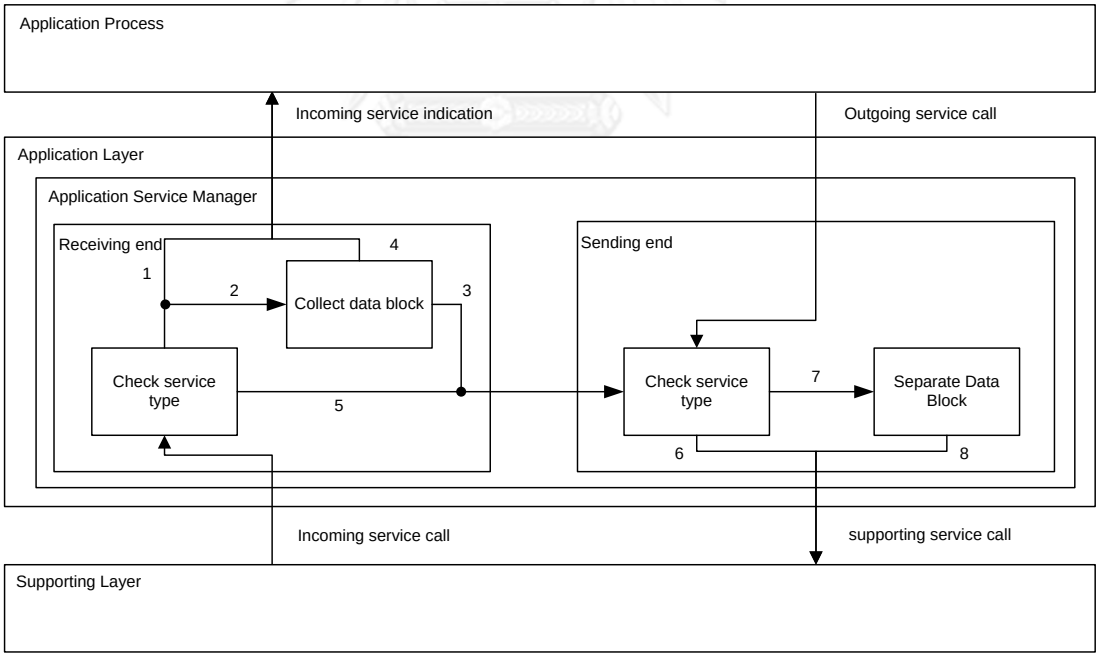
3.5. การออกแบบเทรดจัดการภายในชั้นโปรแกรมประยุกต์ของดีแอลเอ็มเอส/โคเซม

เมื่อตัวประมวลผลโปรแกรมประยุกต์มีความต้องการที่จะส่งข้อมูลที่มีขนาดยาวกว่าขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง มันจำเป็นต้องแยกส่วนข้อมูล ก่อนส่งออกไป และเมื่อมันรับข้อมูลที่เป็นบล็อกข้อมูลย่อยๆ เข้ามา มันจำเป็นต้องประกอบข้อมูลก่อนนำไปใช้ รวมถึงต้องเรียกใช้บริการ เพื่อแจ้งให้อุปกรณ์ฝั่งตรงข้ามทราบว่า มันได้รับบล็อกข้อมูลถึงลำดับที่เท่าไรแล้ว ดังรูปที่ 3-5 [3, 6] คือ ตัวอย่างลำดับการส่งข้อมูลเพื่อรับส่งข้อมูลแบบบล็อกย่อยของบริการ GET ซึ่งเป็นงานที่ยุ่งยากมาก สำหรับผู้ใช้งานคลังโปรแกรม แนวคิดในการใช้เทรดจัดการชั้นโปรแกรมประยุกต์เพื่อทำงานแทนตัวประมวลผลโปรแกรมประยุกต์จึงเกิดขึ้น



รูปที่ 3-5 ตัวอย่างการส่งข้อมูลที่มีขนาดยาวเกินกว่าความสามารถในการส่งของตัวกลาง

ในตอนเริ่มต้น ตัวประมวลผลโปรแกรมประยุกต์ฝั่งไคลเอนต์ เรียกใช้บริการ GET.req แบบ Normal ส่งไปยัง ตัวประมวลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ ซึ่งเป็นบริการในกลุ่มที่ 1 เมื่อตัวประมวลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ทราบความต้องการแล้ว มันจะเตรียมข้อมูลที่จะส่งกลับ โดยข้อมูลที่จะส่งกลับในกรณีนี้ มันยาวกว่าขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง ตัวประมวลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์จึงต้องแบ่งข้อมูลออกเป็นบล็อกย่อยก่อนจะเรียกใช้บริการ GET.res แบบ ONE-BLOCK หรือก็คือ GET.res แบบ With Datablock ที่มีพารามิเตอร์ดังนี้ LAST = FALSE และ Block Number = 1 ซึ่งเป็นบริการในกลุ่มที่ 2 แทน GET.res แบบ Normal ซึ่งเป็นบริการในกลุ่มที่ 1 เมื่อตัวประมวลผลโปรแกรมประยุกต์ฝั่งไคลเอนต์ได้รับข้อมูลบล็อกที่ 1 มันจะตอบด้วยการเรียกใช้บริการ GET.req แบบ Next ที่มีพารามิเตอร์ Block Number = 1 เพื่อแจ้งตัวประมวลผลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ว่า มันได้รับข้อมูลบล็อกที่ 1 เรียบร้อยแล้ว และให้ส่งบล็อกถัดไปได้ เมื่อตัวประมวลผลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์รับทราบมันจะเริ่มส่งบล็อกถัดไป และเป็นดังขั้นตอนข้างบนเรื่อยๆ จนกว่าตัวประมวลผลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์จะส่งบล็อกข้อมูลสุดท้าย โดยมันจะเรียกใช้บริการ GET.res แบบ LAST-BLOCK หรือก็คือ GET.res แบบ With Datablock ที่มีพารามิเตอร์ดังนี้ LAST = TRUE และ Block Number = n โดยในตัวอย่างนี้ n จะมีค่าเท่ากับ 3 เมื่อตัวประมวลผลโปรแกรมประยุกต์ฝั่งไคลเอนต์ได้รับข้อมูลบล็อกสุดท้ายแล้ว มันจะประกอบข้อมูลทั้งหมดเข้าด้วยกัน ก่อนนำไปใช้ต่อไป



รูปที่ 3-6 โครงสร้างการทำงานของเทรดภายในชั้นโปรแกรมประยุกต์

ในกรณีที่ขนาดของข้อมูลมีขนาดใหญ่กว่าความสามารถในการส่งของตัวกลาง ตัวอย่างเช่น ขนาดบัฟเฟอร์ (Buffer) ของชั้นล่างที่รองรับ (Supporting Layer) มีขนาดเล็กกว่าข้อมูลที่ต้องการส่ง จึงจำเป็นต้องมีการแบ่งข้อมูลขนาดใหญ่ ออกเป็นข้อมูลหลาย ๆ บล็อกที่มีขนาด

น้อยกว่า หรือเท่ากับความสามารถในการส่งของตัวกลาง ซึ่งอาจเรียกกระบวนการนี้ว่าการแยกส่วนข้อมูล

ในการส่งข้อมูลแบบบล็อกนี้ ต้องมีการจัดการเพื่อตรวจสอบลำดับของบล็อก และต้องเรียกใช้บริการที่เกี่ยวข้องหลายบริการ ทำให้ผู้ใช้งานคลังโปรแกรมเกิดความยุ่งยากในการใช้งาน เทรดนี้จึงถูกสร้างขึ้นมาในชั้นนี้เพื่อจัดการงานดังกล่าว โดยที่ผู้ใช้ไม่จำเป็นต้องทราบกลไกการแยกส่วนดังกล่าวเลย

จากรูปที่ 3-6 ทางด้านฝั่งรับ (Receiving End) เมื่อชั้นล่างที่รองรับ เรียกบริการในชั้นโปรแกรมประยุกต์เพื่อถอดรหัสข้อมูล บริการนั้นจะตั้งค่าตัวบ่งชี้ เพื่อแจ้งให้เทรดจัดการทราบว่ามีข้อมูลของบริการใดเข้ามา โดยบริการต่างๆ สามารถแบ่งออกได้เป็น 3 กลุ่ม ได้แก่

1. บริการรับ-ส่งข้อมูลสมบูรณ์ในคราวเดียว ดังตารางที่ 3-1 เป็นบริการที่ถูกเทรดจัดการฝั่งตรงข้ามเรียกใช้ เมื่อมันตรวจพบว่าขนาดของข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ฝั่งตรงข้ามต้องการส่งนั้น มีความยาวน้อยกว่า ขนาดบัฟเฟอร์ในการรับ-ส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง

ตารางที่ 3-1 บริการกลุ่มรับ-ส่งข้อมูลสมบูรณ์ในคราวเดียว

บริการสำหรับไคลเอนต์	บริการสำหรับเซิร์ฟเวอร์
COSEM-Open-Request	COSEM-Open-Response
COSEM-Release-Request	COSEM-Release-Response
Get-Request-Normal	Get-Response-Normal
Get-Request-With-List	Get-Response-With-List
Set-Request-Normal	Set-Response-Normal
Set-Request-With-List	Set-Response-With-List
Action-Request-Normal	Action-Response-Normal
Action-Request-With-List	Action-Response-With-List
COSEM-Open-Request	Event-Notification-Request
	Confirmed-Service-Error
	Exception-Response

2. บริการส่งข้อมูลแบบบล็อกย่อย ๆ ดังตารางที่ 3-2 เป็นบริการที่ถูกเทรดจัดการฝั่งตรงข้ามเรียกใช้ เมื่อมันตรวจพบว่าขนาดของข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ฝั่งตรงข้ามต้องการส่งนั้น มีความยาวมากกว่า ขนาดบัฟเฟอร์ในการรับ-ส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง มันจะทยอยส่งบล็อกข้อมูลย่อยๆ มาทีละบล็อก

ตารางที่ 3-2 บริการกลุ่มส่งข้อมูลแบบบล็อกย่อย

บริการสำหรับไคลเอนต์	บริการสำหรับเซิร์ฟเวอร์
Set-Request-With-First-Datablock	Get-Response-With-Datablock
Set-Request-With-Datablock	Action-Response-With-Pblock
Set-request-With-List-And-First-Datablock	
Action-Request-With-First-Block	
Action-Request-With-List-And-First-Pblock	
Action-Request-With-Pblock	

3. บริการยืนยันการรับบล็อกย่อย และร้องขอบล็อกย่อยถัดไป ดังตารางที่ 3-3 เป็นบริการที่ถูกเทรดจัดการฝั่งตรงข้ามเรียกใช้ เมื่อมันได้รับบริการในกลุ่มที่ 2 จากเทรดจัดการฝั่งนี้ เพื่อแจ้งให้เทรดจัดการฝั่งนี้ทราบว่า เทรดจัดการฝั่งตรงข้ามได้รับบล็อกข้อมูลถึงลำดับบล็อกที่เท่าไรแล้ว และให้ส่งบล็อกถัดไปได้เลย

ตารางที่ 3-3 บริการยืนยันการรับบล็อกย่อย และร้องขอบล็อกย่อยถัดไป

บริการสำหรับไคลเอนต์	บริการสำหรับเซิร์ฟเวอร์
Get-Request-Next	Set-Response-Datablock
Action-Request-Next-Pblock	Set-Response-Last-Datablock
Action-Response-Next-Pblock	Set-Response-Last-Datablock-With-List

เมื่อเทรดจัดการตรวจพบว่าบริการที่รับเข้ามานั้นอยู่ในกลุ่มที่ 1 ข้อมูลจะถูกถอดรหัสแล้วไปเก็บไว้ในโครงสร้างข้อมูลที่ได้เตรียมไว้สำหรับบริการในกลุ่มที่ 1 พร้อมทั้งไปปรับตัวบ่งชี้ (Flag) เพื่อแจ้งให้ตัวประมวลผลโปรแกรมประยุกต์ทราบ ดังรูปที่ 3-6 เส้นหมายเลข 1

เมื่อเทรดจัดการตรวจพบว่าบริการที่รับเข้ามานั้นอยู่ในกลุ่มที่ 2 ข้อมูลจะถูกถอดรหัส และเก็บไว้ หลังจากนั้นเทรดจะเข้าสู่สถานะส่งออก (sending end) ดังรูปที่ 3-6 หมายเลข 3 เพื่อส่งบริการในกลุ่มที่ 3 ไปบอกอุปกรณ์อีกฝั่งหนึ่งว่าได้รับข้อมูลถูกต้องแล้ว และร้องขอข้อมูลบล็อกต่อไป ออกไปยังเส้นหมายเลข 6 ดังรูปที่ 3-6 จนได้รับข้อมูลในบล็อกสุดท้าย เมื่อถอดรหัสและนำข้อมูลบล็อกสุดท้ายไปต่อรวมกับบล็อกก่อนหน้า ข้อมูลจะถูกนำไปใส่ไว้ในโครงสร้างข้อมูลที่เตรียมไว้ ซึ่งเป็นโครงสร้างข้อมูลเดียวกันกับที่ใช้สำหรับเก็บข้อมูลที่รับเข้ามาโดยบริการในกลุ่มที่ 1 พร้อมทั้งไปปรับตัวบ่งชี้ (Flag) เพื่อแจ้งให้ตัวประมวลผลโปรแกรมประยุกต์ทราบ ดังรูปที่ 3-6 หมายเลข 4

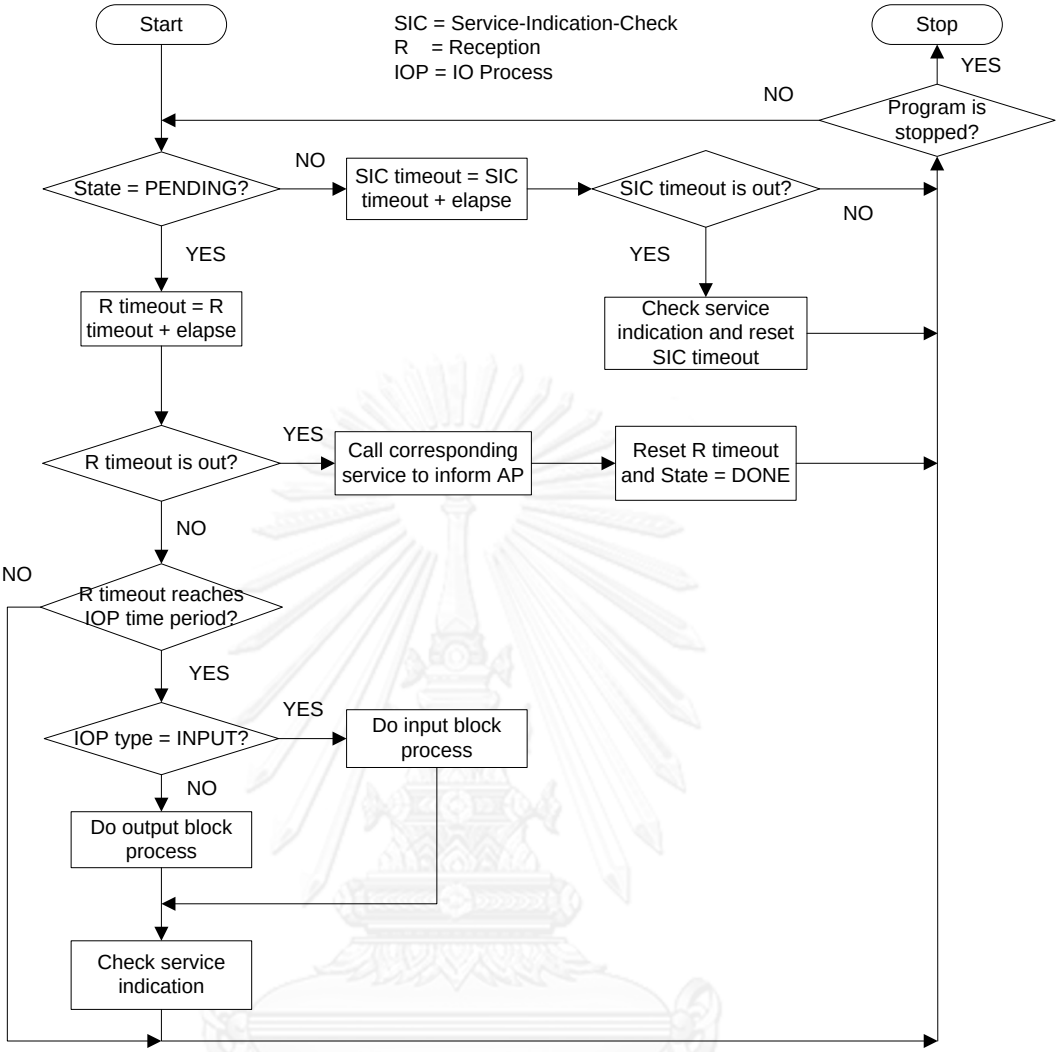
จากรูปที่ 3-6 ทางด้านฝั่งส่ง (sending end) เมื่อตัวประมวลผลโปรแกรมประยุกต์เรียกใช้บริการเพื่อส่งข้อมูลไปยังอุปกรณ์ฝั่งตรงข้าม เทรดจะตรวจสอบว่าเป็นบริการของคำสั่งใด และมีขนาดข้อมูลกี่ไบต์ ถ้าความยาวไม่เกินความสามารถในการส่งของตัวกลาง ข้อมูลจะถูกส่งออกทันที ทางหมายเลข 6 รูปที่ 3-6

แต่ในกรณีที่ขนาดข้อมูลมีความยาวเกินกว่าที่ตัวกลางจะสามารถส่งได้ ข้อมูลจะถูกส่งเข้าสู่กระบวนการแบ่งส่วน ดังรูปที่ 3-6 หมายเลข 7 เพื่อแบ่งเป็นบล็อกย่อยๆ ก่อนจะถูกทยอยส่งออกไปยังอุปกรณ์อีกฝั่งหนึ่ง โดยการเรียกใช้บริการในกลุ่มที่ 2 เพื่อเข้ารหัสข้อมูลก่อนจะส่งโดยเรียกใช้บริการในชั้นล่างที่รองรับอีกครั้งหนึ่ง ทางหมายเลข 8 ดังรูปที่ 3-6 เมื่อส่งบล็อกข้อมูลแล้ว เทรดจัดการจะเข้าสู่สถานะทำงานเสร็จ (Done) และจะคอยตรวจสอบบริการต่างๆ ที่ถูกส่งมาจากอุปกรณ์อีกฝั่งหนึ่งทุกๆ 200 ms หรือทุกๆ ช่วงเวลาใดๆ ที่กำหนดไว้ตอนเรียกใช้บริการ เมื่ออุปกรณ์อีกฝั่งได้รับบล็อกข้อมูลแล้ว จะตอบกลับด้วยบริการในกลุ่มที่ 3 เพื่อบอกว่าได้รับบล็อกข้อมูลที่ส่งไปก่อนหน้าแล้ว ให้ส่งบล็อกต่อไปได้ เมื่อบริการในกลุ่มที่ 3 ถูกเรียกใช้ เพื่อถอดรหัสหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ที่ได้รับเข้ามา พร้อมทั้งปรับตัวบ่งชี้ให้เทรดจัดการได้ทราบว่า มีบริการในกลุ่มที่ 3 เข้ามา เทรดจัดการจะเข้าสู่สถานะรับข้อมูล และตรวจสอบข้อมูลในโครงสร้างข้อมูลที่ได้เตรียมไว้ ว่าการโอนถ่ายบล็อกสำเร็จถึงลำดับบล็อกที่เท่าไรแล้ว และเตรียมตัวส่งบล็อกลำดับต่อไป เทรดจัดการจะเข้าสู่สถานะส่งข้อมูล ดังรูปที่ 3-6 หมายเลข 5 แล้วทำซ้ำจนกว่าจะครบทุกบล็อก นั่นคือ เมื่อเทรดจัดการตรวจสอบพารามิเตอร์ของข้อมูลที่รับเข้ามาแล้ว พบว่าเป็นข้อมูลบล็อกสุดท้าย เทรดจัดการจะเข้าสู่สถานะทำงานเสร็จ

จากที่กล่าวมาข้างต้น การแจ้งเตือนบริการเข้าใหม่ไปยังตัวประมวลผลโปรแกรมประยุกต์ และบริการที่ถูกเรียกใช้โดยตัวประมวลผลโปรแกรมประยุกต์นั้นจะเป็นบริการที่มีลักษณะเหมือนกับบริการในกลุ่มที่ 1 ทั้งหมด แต่จะมีโครงสร้างข้อมูลทั้งในกรณีรับเข้า และส่งออก ที่มากกว่าเพื่อให้ตัวประมวลผลโปรแกรมประยุกต์ส่งข้อมูลได้ยาวเท่าที่ต้องการ ถ้าข้อมูลนั้นไม่ยาวกว่าขนาดบัฟเฟอร์ในการรับ-ส่งที่ได้ตกลงกันไว้ของอุปกรณ์ทั้งสองฝั่ง เทรดจัดการจะเรียกใช้บริการของคำสั่งในกลุ่มที่ 1 เพื่อส่งข้อมูลทั้งหมดได้เลย แต่ถ้ายาวกว่า มันจะเรียกใช้บริการของคำสั่งในกลุ่มที่ 2 และ 3 แทน โดยหน้าที่ของกระบวนการการแบ่งส่วน และการประกอบข้อมูลจะยกให้เป็นหน้าที่ของเทรดจัดการทั้งฝั่งไคลเอนต์ และเซิร์ฟเวอร์ทั้งหมด

3.6. ผังงานการทำงานของเทรดจัดการชั้นโปรแกรมประยุกต์

การทำงานของเทรดจัดการชั้นโปรแกรมประยุกต์ เป็นแบบทำซ้ำทุกๆ ช่วงเวลาที่ผู้ใช้กำหนดให้ สถานะของเทรดจัดการมี 2 ค่า คือ ทำงานเสร็จ (Done) และกำลังทำงาน (Pending) หน้าที่ในขณะที่อยู่ในสถานะ “ทำงานเสร็จ” คือ คอยตรวจสอบว่าชั้นล่างที่รองรับเรียกใช้บริการใดในการถอดรหัสข้อมูล เมื่อมีการเรียกใช้บริการใดๆ ตัวบ่งชี้จะถูกตั้งค่า และสถานของเทรดจัดการจะเข้าสู่สถานะ “กำลังทำงาน” ต่อไป หน้าที่ในขณะที่อยู่ในสถานะ “กำลังทำงาน” คือ ประมวลผล และจัดการข้อมูลเข้า-ออก ทั้งแบบสมบูรณ์ในคราวเดียว แบบบล็อกย่อย และยืนยันการรับบล็อกย่อยและร้องขอบล็อกย่อยถัดไป โดยตรวจสอบจากตัวบ่งชี้ที่ถูกตั้งค่า พร้อมทั้งนับเวลาที่ตั้งเอาไว้เพื่อตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ มีผังงานการทำงานดังรูปที่ 3-7



รูปที่ 3-7 ผังงานการทำงานหลักของเทรดภายในชั้นโปรแกรมประยุกต์

เมื่อเริ่มโปรแกรมเทรดจัดการเข้าสู่สถานะทำงานเสร็จ (Done) และให้ค่าตัวแปรที่ใช้เก็บเวลาที่ต้องเข้าไปตรวจสอบว่ามีข้อมูลรับเข้ามาหรือไม่ (Service-Indication-Check timeout) และเวลาที่ตั้งเอาไว้เพื่อตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ (Reception timeout) เท่ากับศูนย์

เมื่อเข้าสู่ฟังก์ชันการทำงานหลักของเทรด มันจะตรวจสอบว่าตอนนี้อยู่ในสถานะใด ทำงานเสร็จ หรือกำลังทำงาน (Pending)

กรณีเทรดอยู่ในสถานะทำงานเสร็จ มันจะเพิ่มเวลาของ SIC timeout ขึ้นเท่ากับระยะเวลาผ่านไป (elapse time) ซึ่งเป็นค่าคงที่ที่ใส่เข้ามาเป็นตัวแปรรับเข้าของฟังก์ชันนี้ และเป็นระยะเวลาเดียวกันกับเวลาที่ให้เทรดนอนหลับ (Sleep) ก่อนที่จะปลุกขึ้นมาทำงานอีกครั้งหนึ่ง

จากนั้นมันจะตรวจสอบว่าค่า SIC timeout ถึงค่าๆ หนึ่งที่กำหนดหรือยัง ถ้าถึงแล้วจะเข้าสู่ฟังก์ชันตรวจสอบว่ามีข้อมูลรับเข้ามาหรือไม่ พร้อมทั้งตั้งค่า SIC timeout เท่ากับศูนย์ และถ้ายังไม่ถึงค่านั้นก็จะจบการทำงานในรอบนี้

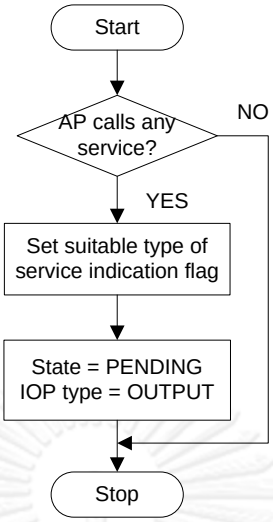
กรณีเข้าเทรดอยู่ในกำลังทำงาน มันจะเพิ่มเวลาของ R timeout ขึ้นเท่ากับระยะเวลาผ่านไปเช่นกัน จากนั้นมันจะตรวจสอบว่าค่า R timeout ถึงค่าๆ หนึ่งที่กำหนดไว้หรือยัง ถ้าถึงแล้ว มันจะไปเรียกฟังก์ชันที่เกี่ยวข้องเพื่อแจ้งตัวประมวลผลชั้นโปรแกรมประยุกต์ว่าอุปกรณ์ฝั่งตรงข้ามมีปัญหาเกิดขึ้น เนื่องจากไม่ตอบกลับมาในเวลาที่กำหนด แต่ถ้ายังไม่ถึงค่านั้น และ R timeout ยังไม่เข้าสู่ช่วงการประมวลผลอินพุตเอาท์พุต (IO Process Time Period) มันจะจบการทำงานในรอบนี้ แต่ถ้า R timeout เข้าสู่ช่วงดังกล่าวพอดี ก็จะทำหน้าที่หลักในโหมดนี้ คือ การรับ-ส่งข้อมูล โดยมันจะตรวจสอบรูปแบบการประมวลผลอินพุตเอาท์พุต (IO Process type) เพื่อให้ทราบว่าในขณะนี้มันต้องส่งข้อมูลออก หรือต้องจัดการกับข้อมูลรับเข้าที่ถูกถอดรหัสแล้ว หลังจากนั้นมันจะต้องตรวจสอบว่ามีข้อมูลรับเข้ามาหรือไม่ เป็นขั้นตอนสุดท้ายในรอบนี้

ปัจจัยต่างๆ ที่มีผลต่อการเปลี่ยนแปลงสถานะ (State) และรูปแบบการประมวลผลอินพุตเอาท์พุต (IOP type) ของเทรด มีดังต่อไปนี้

1. การเรียกใช้บริการต่างๆ ในชั้นโปรแกรมประยุกต์ เพื่อเข้ารหัสข้อมูล โดยตัวประมวลผลชั้นโปรแกรมประยุกต์ การเปลี่ยนแปลงสถานะ และรูปแบบการประมวลอินพุตเอาท์พุตนั้น จะแตกต่างกัน ขึ้นอยู่กับว่าบริการใดถูกเรียกใช้

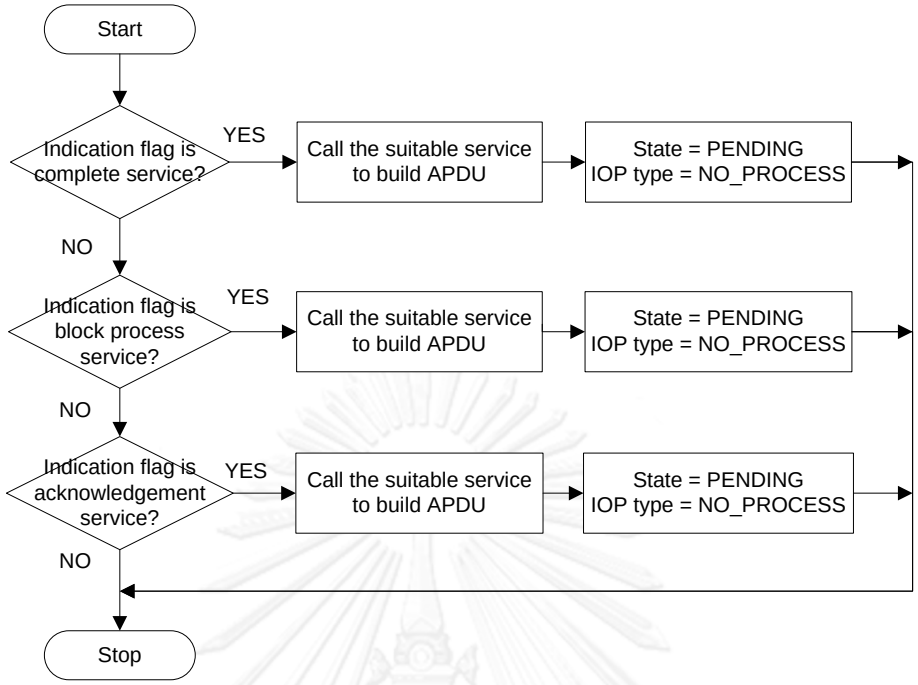
2. การตรวจพบว่ามีการรับข้อมูลเข้ามา เมื่อตรวจพบว่ามีข้อมูลเข้ามาจากภายนอกเทรดจะเข้าสู่สถานะ “กำลังทำงาน” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “อินพุต”

3. การเรียกใช้บริการต่างๆ ในชั้นโปรแกรมประยุกต์ เพื่อจัดการกับข้อมูลที่ถูกถอดรหัสข้อมูลแล้ว เช่น การจัดเรียงบล็อกข้อมูลย่อย และย้ายข้อมูลไปยังโครงสร้างข้อมูลชั้นโปรแกรมประยุกต์ เป็นต้น โดยตัวเทรดเอง ซึ่งมีผลเช่นเดียวกันกับข้อ 1. คือ การเปลี่ยนแปลงสถานะและรูปแบบการประมวลอินพุตเอาท์พุตนั้น จะแตกต่างกัน ขึ้นอยู่กับว่าบริการใดถูกเรียกใช้



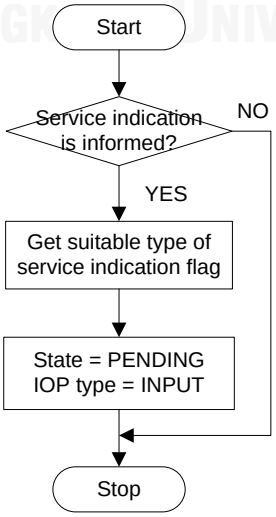
รูปที่ 3-8 ผังงานผลจากตัวประมวลผลชั้นโปรแกรมประยุกต์เรียกใช้บริการใดๆ ในชั้นโปรแกรมประยุกต์ของทั้งฝั่งไคลเอนต์ และเซิร์ฟเวอร์ เพื่อตั้งค่าตัวบ่งชี้สำหรับเทรด ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด

จากรูปที่ 3-8 เมื่อตัวประมวลผลโปรแกรมประยุกต์ฝั่งไคลเอนต์เรียกใช้ บริการใดๆ จะเป็นการไปตั้งค่าให้กับตัวบ่งชี้ (Flag) ของบริการนั้น และสั่งให้เทรดเข้าสู่สถานะ “กำลังทำงาน” รวมถึงเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” เพื่อแจ้งให้เทรดทราบว่าต้องเรียกใช้บริการดังกล่าวเพื่อเข้ารหัสข้อมูล และสร้างหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ (APDU) ในระหว่างนั้นเทรดจะเริ่มนับเวลาที่ตั้งเอาไว้ตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ (R timeout) ด้วย จากที่กล่าวไว้ข้างต้นเมื่อเทรดอยู่ในสถานะ “กำลังทำงาน” และมีรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” ในขณะที่ R timeout ยังไม่หมดลง และเข้าสู่ช่วงเวลาประมวลผลอินพุตเอาท์พุตพอดี มันจะเข้าสู่ฟังก์ชันประมวลผลเอาท์พุต (output block process) ดังผังงานรูปที่ 3-7



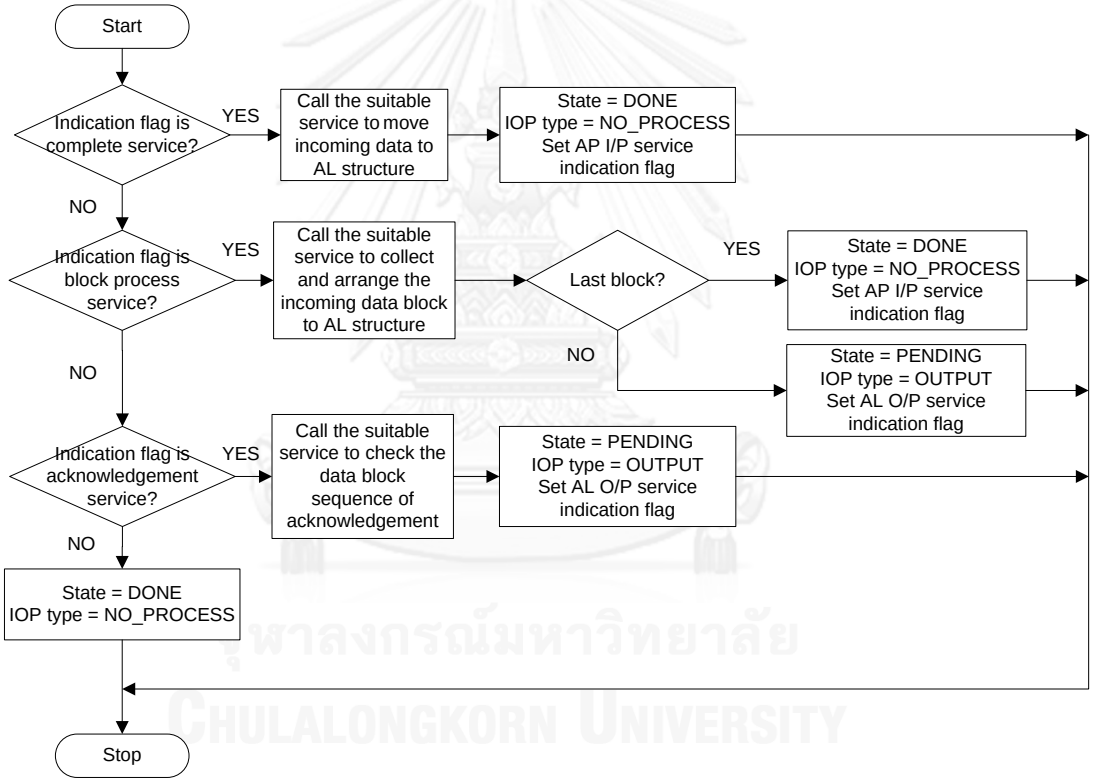
รูปที่ 3-9 ผังงานผลจากตัวประมวลผลชั้นโปรแกรมประยุกต์ตรวจสอบตัวบ่งชี้ และเรียกใช้บริการที่เหมาะสมในชั้นโปรแกรมประยุกต์ฝั่งไคลเอนต์เพื่อเข้ารหัส และส่งออกข้อมูล ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด (Client Output Block Process)

รูปที่ 3-9 คือ ผังงานการทำงานของฟังก์ชันประมวลผลเอาท์พุต เมื่อเริ่มทำงานเทรดจะตรวจสอบตัวบ่งชี้ของบริการใดๆ ว่าถูกตั้งค่าหรือไม่ ถ้าถูกตั้งค่า มันจะรีเซต และเรียกใช้บริการนั้นเพื่อเข้ารหัส และสร้างหน่วยข้อมูลโพรโตคอล ถ้าไม่มีตัวบ่งชี้ใดถูกตั้งค่าเลย มันก็จะเข้าสู่สถานะ “ทำงานเสร็จ” หลังจากนั้น มันจะเปลี่ยนรูปแบบการประมวลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล (NO_PROCESS)” เพื่อเป็นการรีเซต จากนั้นข้อมูลที่เข้ารหัสแล้วจะถูกส่งออกโดยชั้นล่างที่รองรับอีกครั้งหนึ่ง



รูปที่ 3-10 ผังงานผลการรับบริการใดๆ ที่เข้ามาจากภายนอกของทั้งฝั่งไคลเอนต์ และเซิร์ฟเวอร์ ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด

เมื่อเทรดฝั่งเซิร์ฟเวอร์ตรวจพบว่ามีข้อมูลของบริการใดๆ เข้ามา มันจะเปลี่ยนสถานะเข้าสู่ “กำลังทำงาน” และรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “อินพุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของบริการนั้นๆ เพื่อแจ้งให้ตัวเทรดเองทราบว่าต้องเรียกใช้บริการดังกล่าว เพื่อถอดรหัสข้อมูลที่เข้ามาใหม่ ในช่วงเวลาประมวลอินพุตเอาท์พุตที่กำลังจะมาถึง ดังรูปที่ 3-10 จากที่กล่าวไว้ข้างต้นเมื่อเทรดอยู่ในสถานะ “กำลังทำงาน” และมีรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “อินพุต” ในขณะที่ R timeout ยังไม่หมดลง และเข้าสู่ช่วงเวลาประมวลผลอินพุตเอาท์พุตพอดี มันจะเข้าสู่ฟังก์ชันประมวลผลอินพุตฝั่งเซิร์ฟเวอร์ (Server Input Block Process) ดังผังงานรูปที่ 3-7



รูปที่ 3-11 ผังงานผลการเรียกใช้บริการเพื่อจัดการข้อมูลที่รับเข้าจากภายนอกฝั่งเซิร์ฟเวอร์ ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด (Server Input Block Process)

รูปที่ 3-11 คือ ผังงานการทำงานของฟังก์ชันประมวลผลอินพุตฝั่งเซิร์ฟเวอร์ เมื่อเริ่มทำงาน เทรดตรวจพบว่ามีตัวบ่งชี้ของคำสั่งใดๆ ถูกตั้งค่าอยู่ในขณะนั้น เทรดจะรีเซตค่านั้น และเรียกใช้บริการของคำสั่งนั้น เพื่อจัดการกับข้อมูล โดยการนำข้อมูลนั้นไปใส่ไว้ที่โครงสร้างข้อมูลชั้น

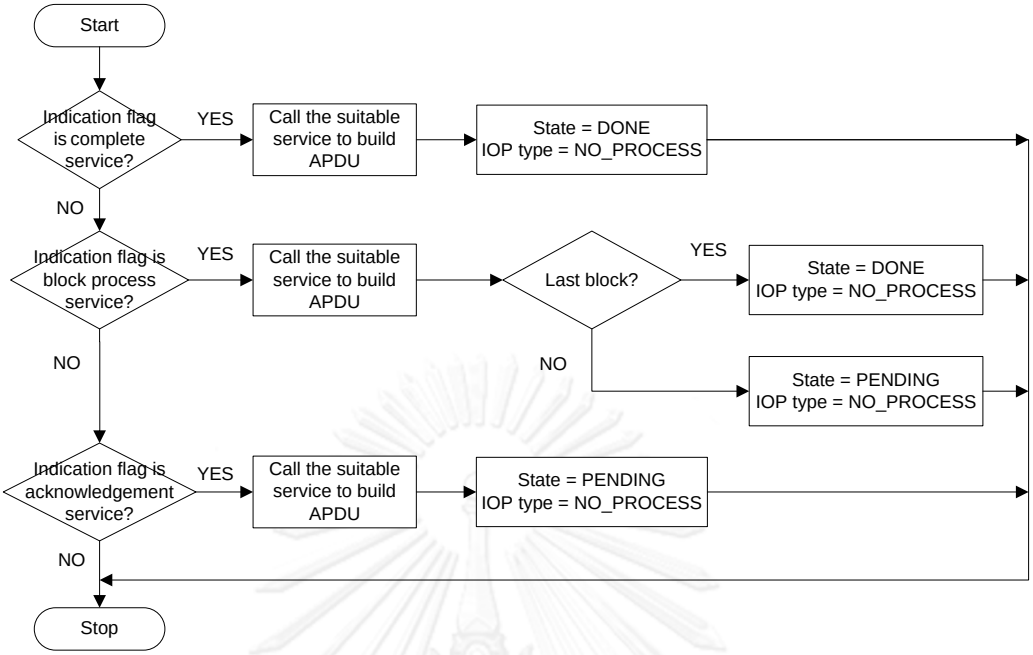
โปรแกรมประยุกต์ของคำสั่งดังกล่าวที่เตรียมไว้ หลังจากนั้นมันจะเปลี่ยนสถานะ เป็นค่าใดๆ ที่สอดคล้องกับการเรียกใช้บริการของคำสั่งนั้นๆ ที่อยู่ในแต่ละกลุ่ม ดังนี้

1.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มสมบูรณ์ในคราวเดียว ดังตารางที่ 3-1 เพื่อจัดการกับข้อมูลที่รับเข้ามา เทรดจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” พร้อมทั้งแจ้งให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า ข้อมูลที่รับเข้ามานั้น พร้อมใช้งานแล้ว ผ่านทางตัวบ่งชี้

2.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มบล็อกย่อย ดังตารางที่ 3-2 เพื่อจัดการกับข้อมูลที่รับเข้ามา ในกรณีที่บล็อกย่อยที่รับเข้ามานั้น ไม่ใช่บล็อกย่อยสุดท้าย เทรดจะคงอยู่สถานะ “กำลังทำงาน” และเปลี่ยน รูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของคำสั่งที่สอดคล้อง เพื่อให้ตัวเทรดเองทราบว่าในรอบหน้าของการประมวลผลอินพุตเอาท์พุตนั้น จะต้องเรียกใช้บริการของคำสั่งใดในการเข้ารหัสข้อมูลเพื่อส่งออก แต่ถ้าบล็อกย่อยที่รับเข้ามานั้น เป็นบล็อกย่อยสุดท้าย เทรดจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” หลังจากนั้นจะแจ้งให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า ข้อมูลที่รับเข้ามานั้น พร้อมใช้งานแล้ว ผ่านทางตัวบ่งชี้

3.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มยืนยันการรับบล็อกย่อย และร้องขอบล็อกย่อยถัดไป ดังตารางที่ 3-3 เพื่อจัดการกับข้อมูลที่รับเข้ามา เทรดจะคงอยู่สถานะ “กำลังทำงาน” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของคำสั่งที่สอดคล้อง เพื่อให้ตัวเทรดเองทราบว่าในรอบหน้าของการประมวลผลอินพุตเอาท์พุตนั้น จะต้องเรียกใช้บริการของคำสั่งใดในการเข้ารหัสข้อมูลเพื่อส่งออก

หลังจากที่ตัวประมวลผลโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ได้รับข้อมูลจากฝั่งไคลเอนต์เรียบร้อยแล้ว มันจะประมวลผลข้อมูลที่รับเข้ามา พร้อมทั้งจัดหาข้อมูลที่ต้องส่งกลับไปยังฝั่งไคลเอนต์ และเรียกใช้บริการใดๆ ในชั้นโปรแกรมประยุกต์ เพื่อตั้งค่าตัวบ่งชี้ของคำสั่งที่ต้องการให้เทรดจัดการเรียกใช้บริการของคำสั่งนั้น เพื่อเข้ารหัสข้อมูลให้ ก่อนส่งต่อให้ชั้นล่างที่รองรับอีกครั้ง เช่นเดียวกับฝั่งไคลเอนต์ ผลของการเรียกใช้บริการนั้น จะทำให้สถานะ และรูปแบบการประมวลอินพุตเอาท์พุตของเทรดเปลี่ยนไปตามผังงานรูปที่ 3-8



รูปที่ 3-12 ผังงานผลจากตัวประมวลผลชั้นโปรแกรมประยุกต์ตรวจสอบตัวบ่งชี้ และเรียกใช้บริการที่เหมาะสมในชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์เพื่อเข้ารหัส และส่งออกข้อมูล ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด (Server Output Block Process)

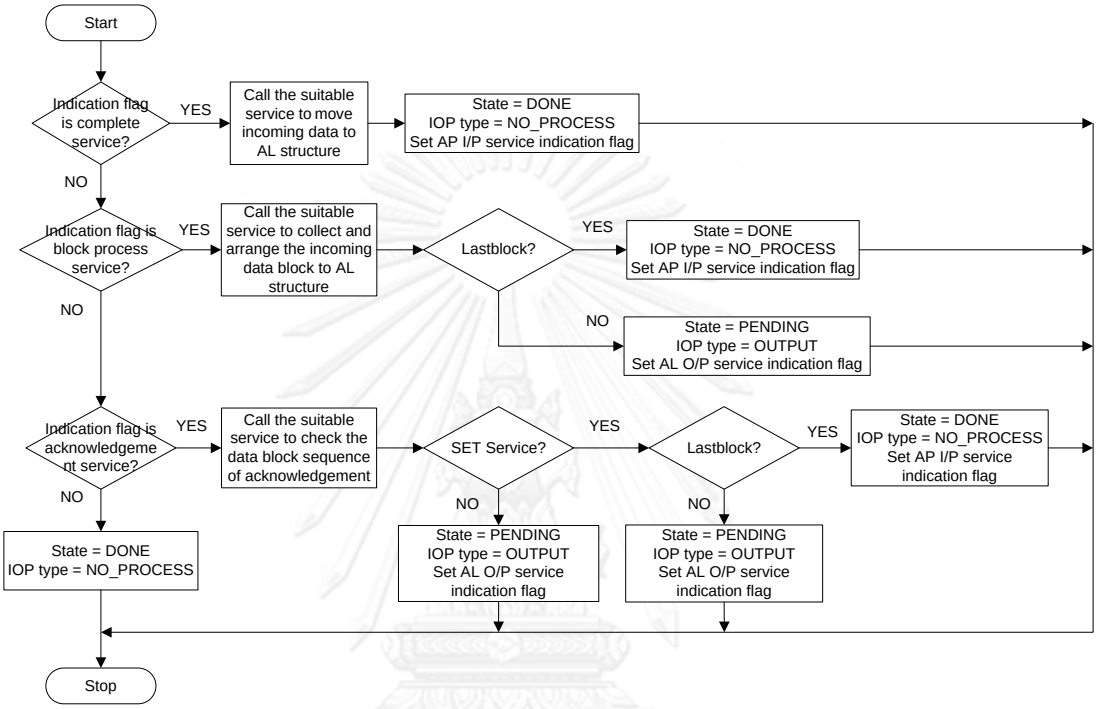
เมื่อเทรดจัดการฝั่งเซิร์ฟเวอร์เข้าสู่ช่วงเวลาประมวลผลอินพุตเอาท์พุต มันจะเข้าสู่ฟังก์ชันประมวลผลเอาท์พุต ที่มีการทำงานตามผังงานรูปที่ 3-12 เมื่อมันตรวจพบตัวบ่งชี้ของบริการใดๆ ที่ตัวประมวลผลโปรแกรมประยุกต์ตั้งค่าไว้ มันจะเรียกบริการที่สอดคล้องกันเพื่อเข้ารหัสข้อมูล ซึ่งมีผลต่อการเปลี่ยนสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรดจัดการฝั่งเซิร์ฟเวอร์ดังต่อไปนี้

1.การเรียกใช้บริการในกลุ่มสมบูรณ์ในคราวเดียว ดังตารางที่ 3-1 เพื่อเข้ารหัสข้อมูล และสร้างหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ มีผลทำให้เทรดจัดการฝั่งเซิร์ฟเวอร์ เปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล”

2.การเรียกใช้บริการในกลุ่มบล็อกย่อย ดังตารางที่ 3-2 เพื่อเข้ารหัสข้อมูล และสร้างหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ มีผลทำให้เทรดจัดการฝั่งเซิร์ฟเวอร์ ยังคงสถานะอยู่ที่ “กำลังงาน” เพื่อนับเวลาที่ตั้งเอาไว้ตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล”

3.การเรียกใช้บริการในกลุ่มยืนยันการรับบล็อกย่อย และร้องขอบล็อกย่อยถัดไป ดังตารางที่ 3-3 เพื่อเข้ารหัสข้อมูล และสร้างหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ ในกรณีที่

บล็อกย่อยที่จะส่งออกนั้น ไม่ใช่บล็อกย่อยสุดท้าย เทรดจะคงอยู่สถานะ “กำลังทำงาน” เพื่อนับเวลาที่ตั้งเอาไว้ตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” แต่ถ้าบล็อกย่อยที่จะส่งออกนั้น เป็นบล็อกย่อยสุดท้าย เทรดจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล”



รูปที่ 3-13 ผังงานผลการเรียกใช้บริการเพื่อจัดการข้อมูลที่รับเข้าจากภายนอกฝั่งไคลเอนต์ ต่อสถานะ และรูปแบบการประมวลผลอินพุตเอาท์พุตของเทรด (Client Input Block Process)

เมื่อเทรดฝั่งไคลเอนต์ตรวจพบว่ามีข้อมูลของบริการใดๆ ตอบสนองกลับมา มันจะเปลี่ยนสถานนะเข้าสู่ “กำลังทำงาน” และรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “อินพุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของคำสั่งนั้นๆ เพื่อแจ้งให้ตัวเทรดเอง ทราบว่าต้องเรียกใช้บริการของคำสั่งดังกล่าว เพื่อจัดการกับข้อมูลที่เข้ามาใหม่ ในช่วงเวลาประมวลอินพุตเอาท์พุตที่กำลังจะมาถึง ดังรูปที่ 3-10 จากที่กล่าวไว้ข้างต้นเมื่อเทรดอยู่ในสถานะ “กำลังทำงาน” และมีรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “อินพุต” ในขณะที่ R timeout ยังไม่หมดลง และเข้าสู่ช่วงเวลาประมวลผลอินพุตเอาท์พุตพอดี มันจะเข้าสู่ฟังก์ชันประมวลผลอินพุตฝั่งไคลเอนต์ (Client Input Block Process) ดังผังงานรูปที่ 3-7

รูปที่ 3-13 คือ ผังงานการทำงานของฟังก์ชันประมวลผลอินพุตฝั่งไคลเอนต์ เมื่อเริ่มทำงาน เทรดตรวจพบว่ามีตัวบ่งชี้ของคำสั่งใดๆ ถูกตั้งค่าอยู่ในขณะนั้น เทรดจะรีเซตค่านั้น และเรียกใช้บริการของคำสั่งนั้น เพื่อจัดการกับข้อมูล โดยการนำข้อมูลนั้นไปใส่ไว้ที่โครงสร้างข้อมูลชั้น

โปรแกรมประยุกต์ของคำสั่งดังกล่าวที่เตรียมไว้ หลังจากนั้นมันจะเปลี่ยนสถานะ เป็นค่าใดๆ ที่สอดคล้องกับการเรียกใช้บริการของคำสั่งนั้นๆ ที่อยู่ในแต่ละกลุ่ม ดังนี้

1.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มสมบูรณ์ในคราวเดียว ดังตารางที่ 3-1 เพื่อจัดการกับข้อมูลที่รับเข้ามา เทรดจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” พร้อมทั้งแจ้งให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า ข้อมูลที่รับเข้ามานั้น พร้อมใช้งานแล้ว ผ่านทางตัวบ่งชี้

2.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มบล็อกย่อย ดังตารางที่ 3-2 เพื่อจัดการกับข้อมูลที่รับเข้ามา ในกรณีที่บล็อกย่อยที่รับเข้ามานั้น ไม่ใช่บล็อกย่อยสุดท้าย เทรดจะคงอยู่สถานะ “กำลังทำงาน” และเปลี่ยน รูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของคำสั่งที่สอดคล้อง เพื่อให้ตัวเทรดเองทราบว่าในรอบหน้าของการประมวลผลอินพุตเอาท์พุตนั้น จะต้องเรียกใช้บริการของคำสั่งใดในการเข้ารหัสข้อมูลเพื่อส่งออก แต่ถ้าบล็อกย่อยที่รับเข้ามานั้น เป็นบล็อกย่อยสุดท้าย เทรดจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” หลังจากนั้นจะแจ้งให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า ข้อมูลที่รับเข้ามานั้น พร้อมใช้งานแล้ว ผ่านทางตัวบ่งชี้

3.เมื่อเรียกใช้บริการของคำสั่งในกลุ่มยืนยันการรับบล็อกย่อย และร้องขอบล็อกย่อยถัดไป ดังตารางที่ 3-3 เพื่อจัดการกับข้อมูลที่รับเข้ามา ในกรณีที่ไม่ใช่บริการตั้งค่า (Set) หรือเป็นบริการตั้งค่า และไม่ใช้บล็อกย่อยสุดท้าย เทรดจะคงอยู่สถานะ “กำลังทำงาน” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “เอาท์พุต” พร้อมทั้งตั้งค่าตัวบ่งชี้ของคำสั่งที่สอดคล้อง เพื่อให้ตัวเทรดเองทราบว่าในรอบหน้าของการประมวลผลอินพุตเอาท์พุตนั้น จะต้องเรียกใช้บริการของคำสั่งใดในการเข้ารหัสข้อมูลเพื่อส่งออก ส่วนในกรณีเป็นบริการตั้งค่า และเป็นบล็อกย่อยสุดท้าย มันจะเปลี่ยนสถานะเป็น “ทำงานเสร็จ” และเปลี่ยนรูปแบบการประมวลผลอินพุตเอาท์พุตเป็น “ไม่มีการประมวลผล” หลังจากนั้นจะแจ้งให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า ข้อมูลที่รับเข้ามานั้น พร้อมใช้งานแล้ว ผ่านทางตัวบ่งชี้

3.7. ตัวแปร และบริการสาธารณะของคลาสชั้นโปรแกรมประยุกต์ที่สำคัญ

คลาสชั้นโปรแกรมประยุกต์ถูกแบ่งออกเป็น สองชนิด ได้แก่ คลาสชั้นโปรแกรมประยุกต์ไคลเอนต์ และเซิร์ฟเวอร์

3.7.1. **คลาสชั้นโปรแกรมประยุกต์ไคลเอนต์**

ภายในคลาสนี้ประกอบด้วยสององค์ประกอบหลัก คือ ตัวแปร และบริการ ซึ่งแต่ละอย่างจะถูกแบบชนิดออกอีกเป็น สาธารณะ ปกป้อง ส่วนบุคคล ในหัวข้อนี้จะกล่าวถึงแค่แบบสาธารณะ ซึ่งเป็นส่วนที่ผู้ที่จะนำคลังโปรแกรมนี้ไปประยุกต์ใช้ต้องทราบ

3.7.1.1. **ตัวแปรสาธารณะที่สำคัญของคลาส**

ตัวแปรสาธารณะที่ถูกใช้ในคลาสโปรแกรมประยุกต์นี้ ถูกแบ่งออกเป็นสามกลุ่ม ได้แก่ ตัวแปรสำหรับให้ผู้ใช้ใส่ข้อมูลการร้องขอการกระทำใดๆ ตัวแปรสำหรับให้ผู้ใช้อ่านค่าผลตอบสนองการกระทำใดๆ ครั้งก่อนหน้า จากฝั่งเซิร์ฟเวอร์ และตัวแปรที่ใช้เก็บตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์ โดยมีรายละเอียดต่างๆ ดังนี้

1. **dataComServices_req_param** เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้ใส่ข้อมูลร้องขอต่างๆ ลงไปในโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายใน ซึ่งผู้ใช้ต้องการเรียกใช้ เพื่อร้องขอการกระทำใดๆ ที่เกี่ยวกับการรับ-ส่งข้อมูลไปยังฝั่งเซิร์ฟเวอร์ ตัวแปรนี้เป็นตัวแปรชนิด AL_Data_Communication_Req_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยหกตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Data_Communication_Req_Parameters
{
        AL_C_Get_Req_Ind_Normal get_req_normal;
        AL_C_Get_Req_Ind_With_List get_req_with_list;
        AL_C_Set_Req_Ind_Normal set_req_normal;
        AL_C_Set_Req_Ind_With_List set_req_with_list;
        AL_C_Action_Req_Ind_Normal action_req_normal;
        AL_C_Action_Req_Ind_With_List action_req_with_list;
};
```

2. **dataComServices_cnf_param** เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้มาอ่านข้อมูลผลตอบสนองต่าง ๆ จากโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายใน ซึ่งเซิร์ฟเวอร์ได้ส่งผลตอบสนองคำสั่งร้องขอการกระทำใดๆ ที่เกี่ยวกับการรับ- ส่งข้อมูลครั้งก่อนหน้ากลับมายังฝั่งไคลเอนต์ ตัวแปรนี้เป็นตัวแปรชนิด AL_Data_Communication_Cnf_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยเจ็ดตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Data_Communication_Cnf_Parameters
{
```

```
        AL_C_Get_Res_Cnf_Normal get_cnf_normal;
        AL_C_Get_Res_Cnf_With_List get_cnf_with_list;
        AL_C_Set_Res_Cnf_Normal set_cnf_normal;
        AL_C_Set_Res_Cnf_With_List set_cnf_with_list;
        AL_C_Action_Res_Cnf_Normal action_cnf_normal;
        AL_C_Action_Res_Cnf_With_List action_cnf_with_list;
        AL_EventNotification_Req_Ind event_notification_ind;
};
```

3. conManServices_req_param เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้ใส่ข้อมูลร้องขอต่าง ๆ ลงไปในโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายใน ซึ่งผู้ใช้ต้องการเรียกใช้ เพื่อร้องขอการสร้างการเชื่อมต่อ หรือการยกเลิกการเชื่อมต่อ ไปยังฝั่งเซิร์ฟเวอร์ ตัวแปรนี้เป็นตัวแปรชนิด AL_Connection_Manage_Req_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยสองตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Connection_Manage_Req_Parameters
{
        AL_COSEM_OPEN_REQ cosem_open_req;
        AL_COSEM_RELEASE_REQ cosem_release_req;
};
```

4. conManServices_cnf_param เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้มารับข้อมูลผลตอบสนองต่าง ๆ จากโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายใน ซึ่งเซิร์ฟเวอร์ได้ส่งผลตอบสนองคำสั่งร้องขอการสร้างการเชื่อมต่อ และการยกเลิกการเชื่อมต่อครั้งก่อนหน้า กลับมายังฝั่งไคลเอนต์ ตัวแปรนี้เป็นตัวแปรชนิด AL_Connection_Manage_Cnf_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยสองตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Connection_Manage_Cnf_Parameters
{
        AL_COSEM_OPEN_CNF cosem_open_cnf;
        AL_COSEM_RELEASE_CNF cosem_release_cnf;
```

```
};
```

5. **al_client_received_services_state** เป็นตัวแปรที่สร้างขึ้นเพื่อเพิ่มความสะดวกในการเรียกใช้ตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์ โดยจะสามารถเรียกใช้ได้ทั้งแบบทีละตัวบ่งชี้ของคำสั่ง (ทีละบิต) หรือแบบทุกตัวบ่งชี้ของคำสั่งพร้อมๆกัน (รวมทุกไบต์) ตัวแปรนี้เป็นตัวแปรชนิด AL_ClientReceivedServiceState ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ภายในประกอบด้วยสองตัวแปรย่อย ได้แก่ ตัวแปร **clientServiceStateBit** เป็นตัวแปรที่ใช้เก็บค่าตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์เพื่อบอกให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า มีผลตอบสนองของคำสั่งใหม่เข้ามา เป็นตัวแปรชนิด AL_ClientReceivedServiceStateBit ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้าง และตัวแปร **clientServiceState** เป็นตัวแปรชนิด uint16_t ดังด้านล่างนี้

```
union AL_ClientReceivedServiceState
{
        AL_ClientReceivedServiceStateBit clientServiceStateBit;
        uint16_t clientServiceState;
};
struct AL_ClientReceivedServiceStateBit
{
        uint16_t b11_exception_cnf              : 1;
        uint16_t b10_event_notification_ind     : 1;
        uint16_t b9_action_withList_cnf         : 1;
        uint16_t b8_action_normal_cnf           : 1;
        uint16_t b7_set_withList_cnf            : 1;
        uint16_t b6_set_normal_cnf              : 1;
        uint16_t b5_get_withList_cnf            : 1;
        uint16_t b4_get_normal_cnf              : 1;
        //---------------------------------------//
        uint16_t b3_confirmed_service_error_cnf : 1;
        //---------------------------------------//
        uint16_t b2_abort_ind                   : 1;
        uint16_t b1_releasecnf                  : 1;
        uint16_t b0_open_cnf                    : 1;
};
```

3.7.1.2. **บริการสาธารณที่สำคัญของคลาส**

บริการสาธารณะที่ถูกใช้ในคลาสโปรแกรมประยุกต์นี้ เป็นบริการของคำสั่งการกระทำใดๆ ที่ต้องการให้ฝั่งเซิร์ฟเวอร์กระทำให้ ถูกแบ่งออกเป็นสองกลุ่ม ได้แก่ บริการที่ใช้เพื่อสร้างและขอยกเลิกการเชื่อมต่อ และบริการที่ใช้รับ-ส่งข้อมูล โดยมีรายละเอียดต่างๆ ดังนี้

1. **void AP_COSEM_OPEN_Req();** เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการขอสร้างการเชื่อมต่อไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **conManServices_req_param.cosem_open_req** ไปเข้ารหัส และส่งออกไปยังเซิร์ฟเวอร์

2. **void AP_COSEM_RELEASE_Req();** เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการขอยกเลิกการเชื่อมต่อไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **conManServices_req_param.cosem_release_req** ไปเข้ารหัส และส่งออกไปยังเซิร์ฟเวอร์

3. **void AP_GET_Req_Normal();** เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอค่าคุณสมบัติ (Attribute) ที่อยู่ภายในอ็อบเจกต์เพียงค่าเดียวไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.get_req_normal** ไปเข้ารหัสและส่งออกไปยังเซิร์ฟเวอร์

4. **void AP_GET_Req_With_List();** เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์หลายๆค่าพร้อมๆ กันไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.get_req_with_list** ไปเข้ารหัสและส่งออกไปยังเซิร์ฟเวอร์

5. **void AP_SET_Req_Normal();** เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอการตั้งค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์เพียงค่าเดียวไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.set_req_normal** ไปเข้ารหัสและส่งออกไปยังเซิร์ฟเวอร์

6. void AP_SET_Req_With_List(); เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอการตั้งค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์หลายๆค่าพร้อมๆ กันไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.set_req_with_list** ไปเข้ารหัสและส่งออกไปยังเซิร์ฟเวอร์

7. void AP_ACTION_Req_Normal(); เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอการเรียกใช้กระบวนการ (Method) ที่อยู่ภายในอ็อบเจกต์เพียงกระบวนการเดียวไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.action_req_normal** ไปเข้ารหัส และส่งออกไปยังเซิร์ฟเวอร์

8. void AP_ACTION_Req_With_List(); เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการร้องขอการเรียกใช้กระบวนการ ที่อยู่ภายในอ็อบเจกต์หลายๆ กระบวนการพร้อมๆ กันไปยังเซิร์ฟเวอร์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_req_param.action_req_with_list** ไปเข้ารหัส และส่งออกไปยังเซิร์ฟเวอร์

3.7.2. คลาสชั้นโปรแกรมประยุกต์เซิร์ฟเวอร์

เช่นเดียวกันกับ คลาสโปรแกรมประยุกต์ภายในคลาสนี้ประกอบด้วยสององค์ประกอบหลัก คือ ตัวแปร และบริการ ซึ่งแต่ละอย่างจะถูกแบบชนิดออกอีกเป็น สาธารณะ ปกป้อง ส่วนบุคคล ในหัวข้อนี้จะกล่าวถึงแค่แบบ สาธารณะ ซึ่งเป็นส่วนที่ผู้ที่จะนำคลังโปรแกรมนี้ไปประยุกต์ใช้ต้องทราบ

3.7.2.1. ตัวแปรสาธารณะที่สำคัญของคลาส

ตัวแปรสาธารณะที่ถูกใช้ในคลาสโปรแกรมประยุกต์นี้ ถูกแบ่งออกเป็นสามกลุ่ม ได้แก่ ตัวแปรสำหรับให้ผู้ใช้อ่านข้อมูลการร้องขอการกระทำใดๆ ที่รับมาจากไคลเอนต์ ตัวแปรสำหรับให้ผู้ใช้ใส่ค่าผลตอบสนองของการกระทำนั้นๆ ที่ถูกร้องขอมาจากฝั่งไคลเอนต์ และตัวแปรที่ใช้เก็บตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์ โดยมีรายละเอียดต่างๆ ดังนี้

1. dataComServices_res_param เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้ใส่ข้อมูลผลตอบสนองของการร้องขอต่าง ๆ ลงไปในโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายในซึ่งผู้ใช้ต้องการเรียกใช้เพื่อส่งผลตอบสนองการร้องขอการกระทำนั้นๆ ที่เกี่ยวกับการรับ-ส่งข้อมูลกลับไปยังฝั่งไคลเอนต์ ตัวแปรนี้เป็นตัวแปรชนิด AL_Data_Communication_Res_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยสิบสองตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Data_Communication_Res_Parameters
{
        AL_Read_Res_Cnf read_res;
        AL_Write_Res_Cnf write_res;
        AL_InformationReport_Req_Ind information_report_req;
        AL_Confirmed_Service_Error confirmed_service_error_res;
        AL_S_Get_Res_Cnf_Normal get_res_normal;
        AL_S_Get_Res_Cnf_With_List get_res_with_list;
        AL_S_Set_Res_Cnf_Normal set_res_normal;
        AL_S_Set_Res_Cnf_With_List set_res_with_list;
        AL_S_Action_Res_Cnf_Normal action_res_normal;
        AL_S_Action_Res_Cnf_With_List action_res_with_list;
        AL_EventNotification_Req_Ind event_notification_req;
        AL_Exception_Res_Cnf exception_res;
};
```

2. dataComServices_ind_param เป็นตัวแปรยูเนียนสำหรับให้ผู้ใช้มาอ่านข้อมูลการร้องขอการกระทำใดๆ ที่เกี่ยวกับการรับ- ส่งข้อมูลต่าง ๆ ซึ่งไคลเอนต์ได้ส่งมา จากโครงสร้างข้อมูลย่อยๆ ของคำสั่งที่อยู่ภายใน ตัวแปรนี้เป็นตัวแปรชนิด AL_Data_Communication_Ind_Parameters ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ซึ่งภายในประกอบด้วยเก้าตัวแปรย่อย ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้างอีกทีหนึ่ง ดังด้านล่างนี้ ส่วนรายละเอียดชนิดตัวแปรย่อยต่างๆ จะถูกอธิบายในคู่มือการใช้คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมในซีดีรอมวิทยานิพนธ์

```
union AL_Data_Communication_Ind_Parameters
{
        AL_Read_Req_Ind read_ind;
```

```
        AL_Write_Req_Ind write_ind;
        AL_UnconfirmedWrite_Req_Ind unconfirmed_write_ind;
        AL_S_Get_Req_Ind_Normal get_ind_normal;
        AL_S_Get_Req_Ind_With_List get_ind_with_list;
        AL_S_Set_Req_Ind_Normal set_ind_normal;
        AL_S_Set_Req_Ind_With_List set_ind_with_list;
        AL_S_Action_Req_Ind_Normal action_ind_normal;
        AL_S_Action_Req_Ind_With_List action_ind_with_list;
};
```

3. al_server_received_services_state เป็นตัวแปรที่สร้างขึ้นเพื่อเพิ่มความสะดวกในการเรียกใช้ตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์ โดยจะสามารถเรียกใช้ได้ทั้งแบบทีละตัวบ่งชี้ของคำสั่ง (ทีละบิต) หรือแบบทุกตัวบ่งชี้ของคำสั่งพร้อมๆกัน (รวมทุกไบต์) ตัวแปรนี้เป็นตัวแปรชนิด AL_ServerReceivedServiceState ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรยูเนียน ภายในประกอบด้วยสองตัวแปรย่อย ได้แก่ ตัวแปร serverServiceStateBit เป็นตัวแปรที่ใช้เก็บค่าตัวบ่งชี้การรับข้อมูลชั้นโปรแกรมประยุกต์เพื่อบอกให้ตัวประมวลผลชั้นโปรแกรมประยุกต์ทราบว่า มีผลตอบสนองของคำสั่งใหม่เข้ามา เป็นตัวแปรชนิด AL_ServerReceivedServiceStateBit ซึ่งเป็นชนิดตัวแปรที่นิยามขึ้นจากชนิดตัวแปรโครงสร้าง และตัวแปร servertServiceState เป็นตัวแปรชนิด uint16_t ดังด้านล่างนี้

```
union AL_ServerReceivedServiceState
{
        AL_ServerReceivedServiceStateBit serverServiceStateBit;
        uint16_t serverServiceState;
};
struct AL_ServerReceivedServiceStateBit
{
        uint16_t b11_action_withList_ind        : 1;
        uint16_t b10_action_normal_ind          : 1;
        uint16_t b9_set_withList_ind            : 1;
        uint16_t b8_set_normal_ind              : 1;
        uint16_t b7_get_withList_ind            : 1;
        uint16_t b6_get_normal_ind              : 1;
        //---------------------------------------//
        uint16_t b5_unconfirmed_write_ind       : 1;
        uint16_t b4_write_ind                   : 1;
```

```
        uint16_t b3_read_ind                    : 1;
        //-------------------------------------//
        uint16_t b2_abort_ind                   : 1;
        uint16_t b1_release_ind                 : 1;
        uint16_t b0_open_ind                    : 1;
};
```

3.7.2.2. **บริการสาธารณะที่สำคัญของคลาส**

บริการสาธารณะที่ถูกใช้ในคลาสโปรแกรมประยุกต์นี้ เป็นบริการของคำสั่งเพื่อส่งผลตอบสนองการกระทำใดๆ ที่ฝั่งไคลเอนต์ร้องขอมาก่อนหน้า หรือบริการของคำสั่งเพื่อแจ้งเตือนเหตุการณ์สำคัญ ถูกแบ่งออกเป็นสามกลุ่ม ได้แก่ บริการที่ใช้เพื่อสร้างและขอยกเลิกการเชื่อมต่อบริการที่ใช้รับ-ส่งข้อมูล และบริการที่ใช้เพื่อแจ้งเตือนเหตุการณ์สำคัญ โดยมีรายละเอียดต่างๆ ดังนี้

1. void AP_COSEM_OPEN_Res(); เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการขอสร้างการเชื่อมต่อกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า conManServices_res_param.cosem_open_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

2. void AP_COSEM_RELEASE_Res(); เป็นบริการสำหรับตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการขอยกเลิกการเชื่อมต่อกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า conManServices_res_param.cosem_release_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

3. void AP_READ_Res(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อสั้น (Short Name) เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอค่าคุณสมบัติ หรือการเรียกใช้กระบวนการที่**ต้องมี**ข้อมูลส่งกลับไป ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียว หรือหลายๆ ผลตอบสนอง พร้อมๆ กัน กลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.read_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

4. void AP_WRITE_Res(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อสั้น เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการตั้งค่าคุณสมบัติ หรือการเรียกใช้กระบวนการที่**ไม่ต้องมี**ข้อมูลส่งกลับไป ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียว หรือหลายๆ ผลตอบสนอง พร้อมๆ กันกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.write_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

5. void AP_WRITE_Res(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อสั้น เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการตั้งค่าคุณสมบัติ หรือการเรียกใช้กระบวนการที่**ไม่ต้องมี**ข้อมูลส่งกลับไป ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียว หรือหลายๆ ผลตอบสนอง พร้อมๆ กันกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.write_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

6. void AP_INFORMATION_REPORT_Req(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อสั้น เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการแจ้งเตือนค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์ยังไคลเอนต์ ในกรณีที่เกิดเหตุการณ์สำคัญต่างๆ โดยสามารถแจ้งเพียงค่าเดียว หรือหลายๆ ค่าพร้อมๆ กันต่อการเรียกใช้หนึ่งครั้ง ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.information_report_req ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

7. void AP_CONFIRMED_SERVICE_ERROR_Res(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ หรือชื่อสั้นก็ได้ และใช้ในกรณีที่เซิร์ฟเวอร์ไม่มีบริการของคำสั่ง ExceptionResponse (AP_EXCEPTION_Res();) ให้ใช้งาน เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการแจ้งเตือนว่าการเรียกใช้คำสั่งร้องขอการกระทำใดๆ ที่ไคลเอนต์ส่งมานั้น มีรูปแบบที่ไม่เหมาะสม หรือสอดคล้องกับลักษณะการเชื่อมต่อ หรือกรณีอื่นๆ และบอกเหตุผลของความไม่เหมาะสมกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.confirmed_service_error_res ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

8. void AP_GET_Res_Normal(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียวกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.get_res_normal ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

9. void AP_GET_Res_With_List(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์หลายๆ ผลตอบสนองพร้อมๆ กันกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.get_res_with_list ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

10. void AP_SET_Res_Normal(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการตั้งค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียวกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.set_res_normal ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

11. void AP_SET_Res_With_List(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการตั้งค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์หลายๆ ผลตอบสนองพร้อมๆ กันกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า dataComServices_res_param.set_res_with_list ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

12. void AP_ACTION_Res_Normal(); เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการเรียกใช้กระบวนการ ที่อยู่ภายในอ็อบเจกต์เพียงผลตอบสนองเดียวกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัว

แปรโครงสร้างที่ชื่อว่า **dataComServices_res_param.action_res_normal** ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

13. **void AP_ACTION_Res_With_List();** เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการส่งผลตอบสนองการร้องขอการเรียกใช้กระบวนการ ที่อยู่ภายในอ็อบเจกต์หลายๆ ผลตอบสนองพร้อมๆ กันกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_res_param.action_res_with_list** ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

14. **void AP_EVENT_NOTIFICATION_Req();** เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการแจ้งเตือนค่าคุณสมบัติ ที่อยู่ภายในอ็อบเจกต์ยังไคลเอนต์ ในกรณีที่เกิดเหตุการณ์สำคัญต่างๆ โดยสามารถแจ้งได้เพียงค่าเดียวต่อการเรียกใช้หนึ่งครั้งเท่านั้น ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_res_param.event_notification_req** ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

15. **void AP_EXCEPTION_Res();** เป็นบริการที่ถูกเรียกใช้เมื่อ เซิร์ฟเวอร์นั้นใช้การอ้างอิงแบบชื่อตรรกะ หรือชื่อสั้นก็ได้ เพื่อตั้งค่าตัวบ่งชี้การส่งชั้นโปรแกรมประยุกต์เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า ตัวประมวลผลชั้นโปรแกรมประยุกต์ต้องการแจ้งเตือนว่าการเรียกใช้คำสั่งร้องขอการกระทำใดๆ ที่ไคลเอนต์ส่งมานั้น มีรูปแบบที่ไม่เหมาะสม หรือสอดคล้องกับลักษณะการเชื่อมต่อ หรือกรณีอื่นๆ และบอกเหตุผลของความไม่เหมาะสมกลับไปยังไคลเอนต์ ให้เทรดนำข้อมูลที่ตัวประมวลผลชั้นโปรแกรมประยุกต์ใส่ไว้ในตัวแปรโครงสร้างที่ชื่อว่า **dataComServices_res_param.exception_res** ไปเข้ารหัส และส่งออกไปยังไคลเอนต์

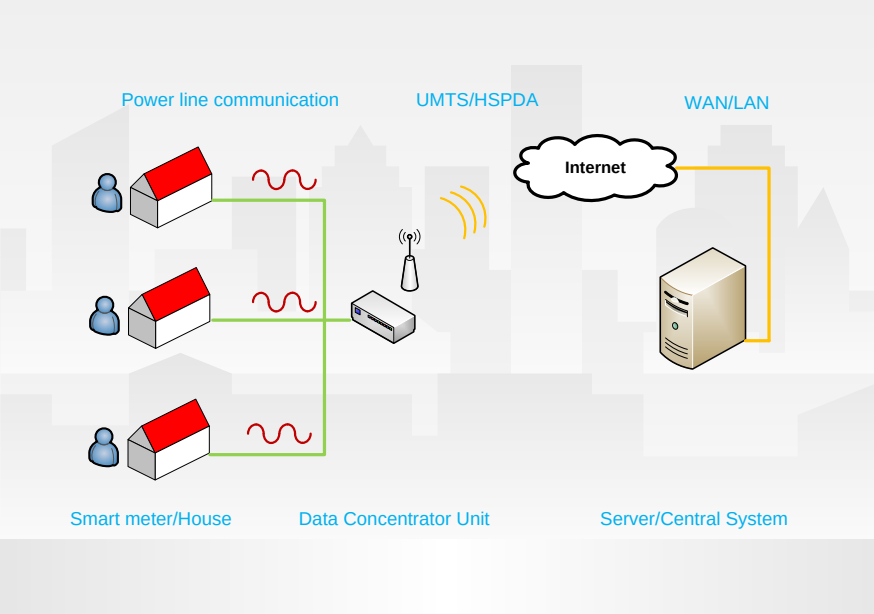
บทที่ 4

การออกแบบระบบจำลองเพื่อการทดสอบคลังโปรแกรมดีแอลเอ็มเอส/โคเซม และการทำงานของอุปกรณ์เก็บรวบรวมข้อมูล

ในบทนี้จะพูดถึงการออกแบบอุปกรณ์หลักๆ ทั้งสามชนิด ในระบบโครงสร้างการวัดขั้นสูง (Advanced Metering Infrastructure, AMI) จำลอง ได้แก่ ระบบกลางจำลอง ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลอง โดยจะพูดถึงการออกแบบ ซึ่งเน้นไปที่ด้านซอร์ฟแวร์ และฮาร์ดแวร์เฉพาะส่วนของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล

4.1. ภาพรวมของระบบ

หัวข้อย่อยนี้จะกล่าวถึงภาพรวมของระบบการสื่อสารกันระหว่างอุปกรณ์ทั้งสามผ่านทางสื่อต่างๆ ได้แก่ การสื่อสารผ่านสายไฟฟ้าส่งกำลัง (Power Line Communication, PLC) ระหว่างอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะ การสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA) และการสื่อสารผ่านอินเทอร์เน็ตแบบแลน (Local Area Network, LAN) หรือแบบแวน (Wide Area Network, WAN) ระหว่างระบบกลาง และอุปกรณ์เก็บรวบรวมข้อมูลดังรูปที่ 4-1

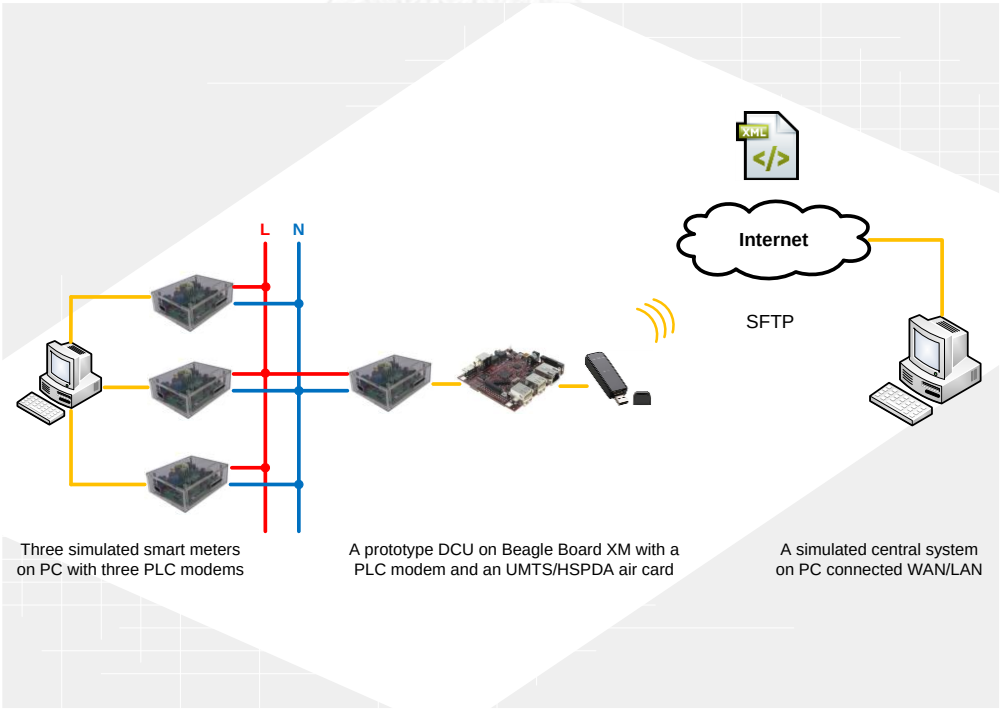


รูปที่ 4-1 ภาพรวมของระบบที่เสนอในการวิจัย

ระบบที่เสนอในการวิจัยมีลักษณะดังรูปที่ 4-1 ประกอบด้วย เครื่องคอมพิวเตอร์เซิร์ฟเวอร์ตั้งอยู่ที่ระบบกลางของผู้ให้บริการไฟฟ้า โดยเครื่องเซิร์ฟเวอร์ต้องเชื่อมต่อเครือข่ายอินเทอร์เน็ตแบบแลน หรืออื่นๆ เพื่อใช้ในการสื่อสารกับอุปกรณ์เก็บรวบรวมข้อมูลผ่านทางโพรโตคอลทีซีพี/ไอพี (TCP/IP) และอุปกรณ์รวบรวมข้อมูลต้องเชื่อมต่อเครือข่ายอินเทอร์เน็ตเช่นกัน เนื่องอุปกรณ์เก็บรวบรวมข้อมูลส่วนใหญ่จะถูกติดตั้งไว้ในที่ที่หาเครือข่ายอินเทอร์เน็ตแบบแลนได้ยาก ในงานวิจัยนี้เลือกใช้ตัวกลางเป็นระบบเครือข่ายโทรศัพท์เคลื่อนที่ในยุคที่ 3 นั่นคือ ยูเอ็มทีเอส/เอช-เอสพีดีเอ ซึ่งสามารถเชื่อมต่อเครือข่ายอินเทอร์เน็ตได้ทุกที่ที่มีสัญญาโทรศัพท์เคลื่อนที่ และรองรับการสื่อสารผ่านโพรโตคอลทีซีพี/ไอพี ได้เช่นเดียวกัน อุปกรณ์เก็บรวบรวมข้อมูลถูกออกแบบเพื่อใช้รวบรวมข้อมูลสภาพของระบบไฟฟ้า การใช้ไฟฟ้า และเหตุการณ์สำคัญต่างๆ จากมาตรอัจฉริยะจำนวนหนึ่งที่ถูกติดตั้งตามที่พักอาศัย ซึ่งเชื่อมต่อกันผ่านทางการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ตามมาตรฐานไพร์ม เมื่ออุปกรณ์เก็บรวบรวมข้อมูลได้รับข้อมูลครบแล้ว จะทำการบีบอัดข้อมูล และส่งต่อให้ระบบกลาง โดยขั้นตอนการสื่อสาร และหน่วยข้อมูลโพรโทคอลอ้างอิงตามมาตรฐาน ดีแอลเอ็มเอส/โคเซม (DLMS/COSEM)

4.2. ระบบภาพรวมระบบจำลองที่ใช้ทดสอบการทำงานของระบบที่เสนอในการทดสอบ

ในหัวข้อย่อยนี้จะกล่าวลึกลงในรายละเอียดว่า ระบบจำลองของอุปกรณ์ทั้งสามนั้นใช้รูปแบบของฮาร์ดแวร์ และซอร์ฟแวร์แบบใด รวมถึงการนำโพรโทคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต (Secure File Transfer Protocol, SFTP) และระบบเครือข่ายส่วนบุคคลเสมือน (Virtual Private Network, VPN) มาใช้

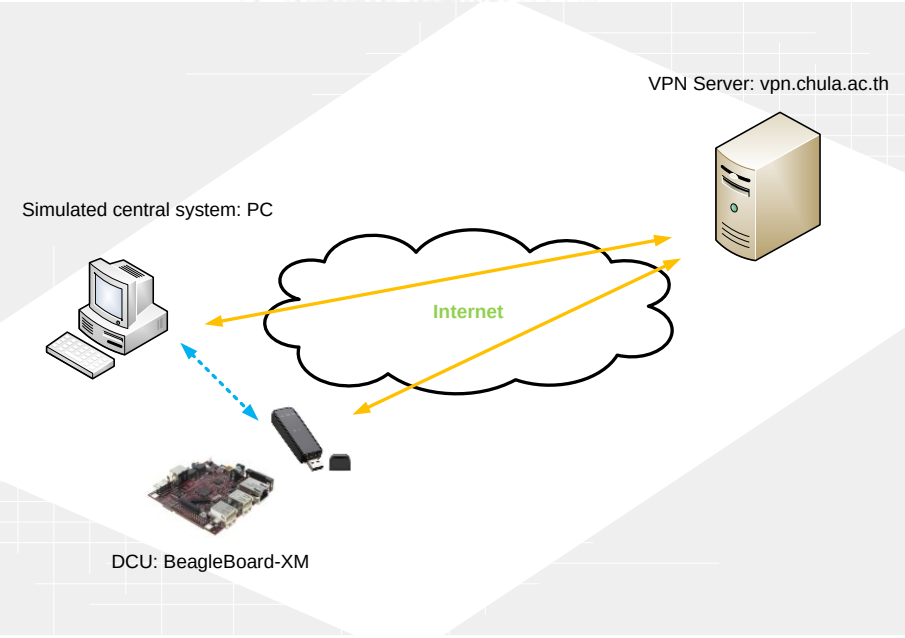


รูปที่ 4-2 ภาพรวมของระบบจำลองที่ใช้ทดสอบการทำงานของระบบที่เสนอในการวิจัย

ทางฝั่งอุปกรณ์เก็บรวบรวมข้อมูลถูกพัฒนาขึ้นโดยใช้บอร์ดการพัฒนา (Development Board) ที่มีชื่อว่า Beagleboard-xM [16] โดยมี ARM Cortex A8 เป็นตัวประมวลผลกลาง มีระบบปฏิบัติการอูบันตู เพื่อจัดการการใช้ทรัพยากรต่างๆ บนบอร์ดได้อย่างสะดวก และมีสิทธิภาพ พร้อมทั้งมีช่อง USB เพื่อเชื่อมต่อกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง [17] เพื่อใช้สื่อสารกับมาตรอัจฉริยะ และเชื่อมต่อโมเด็มการสื่อสารในยุคที่สาม ในการวิจัยนี้ใช้ 3G UMTS/HSPDA Air card [18] เพื่อเชื่อมต่อกับระบบกลางทางอินเตอร์โดยใช้โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต ในการถ่ายโอนแฟ้มข้อมูล XML ที่ใช้บรรจุคำสั่ง และผลตอบสนอง

ทางฝั่งมาตรอัจฉริยะ 3 เครื่อง ถูกจำลองเป็นโปรแกรมประยุกต์บนคอมพิวเตอร์ส่วนบุคคล 1 เครื่อง โดยที่คอมพิวเตอร์เครื่องนั้นเชื่อมต่อกับโมเด็มสื่อสารผ่านสายไฟฟ้าส่งกำลัง 3 ตัวเช่นกัน ผ่านสาย USB เพื่อใช้สื่อสารกับโมเด็มสื่อสารผ่านสายไฟฟ้าส่งกำลังของฝั่งอุปกรณ์เก็บรวบรวมข้อมูล

ทางฝั่งระบบกลาง ถูกจำลองเป็นโปรแกรมประยุกต์บนคอมพิวเตอร์ส่วนบุคคล 1 เครื่อง โดยที่คอมพิวเตอร์นั้นเชื่อมต่อเครือข่ายอินเทอร์เน็ตแบบเฉพาะที่ เพื่อใช้สื่อสารกับอุปกรณ์เก็บรวบรวมข้อมูล



รูปที่ 4-3 รูปแบบการสื่อสารระหว่างอุปกรณ์เก็บรวบรวมข้อมูล และระบบกลางผ่านระบบเครือข่ายส่วนบุคคลเสมือน

เพื่อให้บอร์ดการพัฒนา Beagleboard-xM เสมือนเชื่อมต่ออยู่ภายในเครือข่ายเดียวกันกับคอมพิวเตอร์ที่ใช้จำลองการทำงานของระบบกลาง ทั้งสองอุปกรณ์จึงจำเป็นต้องเชื่อม

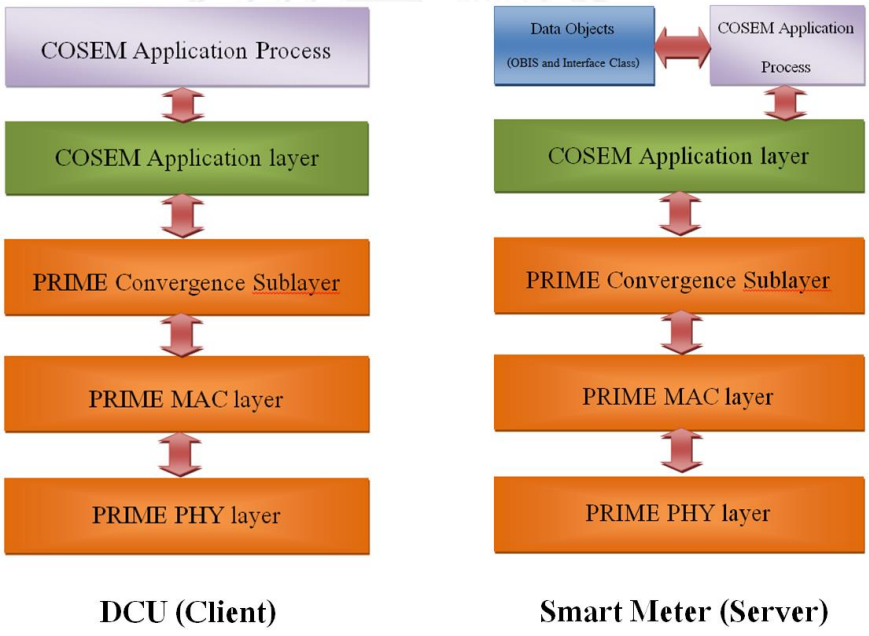
ต่อไปยังเครื่องเซิร์ฟเวอร์กลางที่ทำหน้าที่เป็นเครื่องเซิร์ฟเวอร์เครือข่ายส่วนบุคคลเสมือน ในการวิจัยนี้จะเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์เครือข่ายเสมือนของจุฬาลงกรณ์มหาวิทยาลัยซึ่งมีชื่อเซิร์ฟเวอร์ DNS คือ vpn.chula.ac.th ดังรูปที่ 4-3 โดยเส้นสีส้ม คือ เส้นทางการสื่อสารจริง และเส้นสีฟ้า คือ เส้นทางการสื่อสารเสมือน

4.3. โพรไฟล์สื่อสารของอุปกรณ์ภายในระบบจำลอง

โพรไฟล์สื่อสารของระบบจำลองถูกแบ่งออกเป็นสองส่วน ได้แก่ โพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลอง และโพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับระบบกลางจำลอง

4.3.1. โพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลอง

โพรไฟล์สื่อสารระหว่างทั้งสองอุปกรณ์ ได้อ้างอิงมาจากมาตรฐาน OPEN meter ฉบับที่ D 5.2.1 หน้า 13 [19] และ ฉบับที่ D 3.1 หน้า 42 [20] ซึ่งมีลักษณะดังรูปที่ 4-4



รูปที่ 4-4 โพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลอง

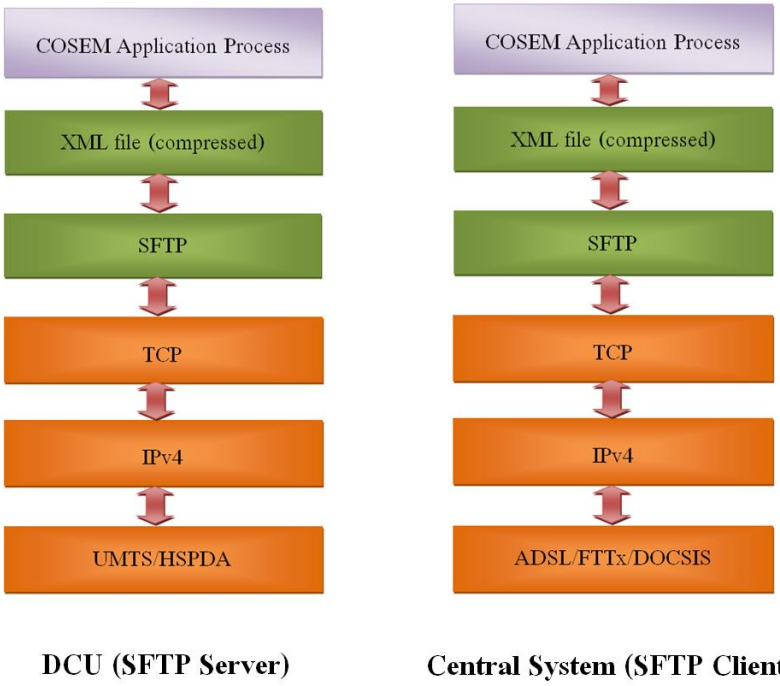
โพรไฟล์สื่อสารระหว่างทั้งสองอุปกรณ์นั้นมีลักษณะคล้ายกับรูปที่ 2-4 เนื่องจากมีตัวประมวลผลโปรแกรมประยุกต์ และชั้นโปรแกรมประยุกต์โคเซมอยู่บนสุดของโพรไฟล์สื่อสาร แต่โพรไฟล์สื่อสารระหว่างอุปกรณ์ทั้งสองในรูปที่ 4-4 นั้น โพรโทคอลในชั้นตั้งแต่ชั้นโปรแกรมประยุกต์โคเซมลงไป ใช้เป็นชั้นโพรโทคอลของไพร์มแทน ซึ่งเป็นชั้นโพรโทคอลดังรูปที่ 2-11 ในกลุ่มควบคุม และข้อมูล โพรไฟล์สื่อสารแบบนี้ กำลังจะถูกประกาศลงในมาตรฐานดีแอลเอ็มเอส/โคเซม เช่นกัน

ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลทำหน้าที่เป็นไคลเอนต์ ซึ่งเป็นผู้เริ่มการสื่อสารก่อน มันต้องร้องขอการสร้างช่องทางการเชื่อมต่อไปยังมาตรอัจฉริยะจำลองซึ่งทำหน้าที่เป็นเซิร์ฟเวอร์เสียก่อน จึงจะสามารถร้องขอการกระทำใดๆ จากมาตรอัจฉริยะจำลองได้

โพรไฟล์สื่อสารระหว่างทั้งสองอุปกรณ์นั้นแตกต่างกันตรงที่โพรไฟล์ของมาตรอัจฉริยะจำลองมีอ็อบเจกต์ที่ใช้เก็บค่าของข้อมูลปริมาณทางไฟฟ้าที่วัดได้ และอ็อบเจกต์ที่ใช้เก็บฟังก์ชันการทำงานต่างๆ ของมาตรอัจฉริยะจำลอง เพื่อให้ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลร้องขอการกระทำใดๆ ต่อค่าต่างๆ หรือให้กระทำฟังก์ชันใดๆ ที่บรรจุอยู่ภายในอ็อบเจกต์ของมาตรอัจฉริยะจำลอง

4.3.2. **โพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับระบบกลางจำลอง**

โพรไฟล์สื่อสารระหว่างทั้งสองอุปกรณ์ ได้อ้างอิงมาจากมาตรฐาน OPEN meter ฉบับที่ D 3.1 หน้า 76 [20] ซึ่งมีลักษณะดังรูปที่ 4-5



รูปที่ 4-5 โพรไฟล์สื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับระบบกลางจำลอง

การสื่อสารกันของอุปกรณ์ทั้งสองจะใช้การส่งแฟ้มข้อมูลที่บรรจุข้อมูลที่มีรูปแบบการเข้ารหัสแบบดีแอลเอ็มเอส/โคเซม โดยจะถูกบีบอัดข้อมูลก่อนถูกส่งผ่านโพรโทคอลการโอนถ่ายแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต บนโพรโทคอลทีซีพี/ไอพี

ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลทำหน้าที่เป็นเซิร์ฟเวอร์ของการโอนถ่ายแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต โดยระบบกลางจำลองซึ่งเป็นไคลเอนต์ ต้องเชื่อมต่อไปยังต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลก่อนเสมอ จึงจะเริ่มถ่ายโอนแฟ้มข้อมูลได้

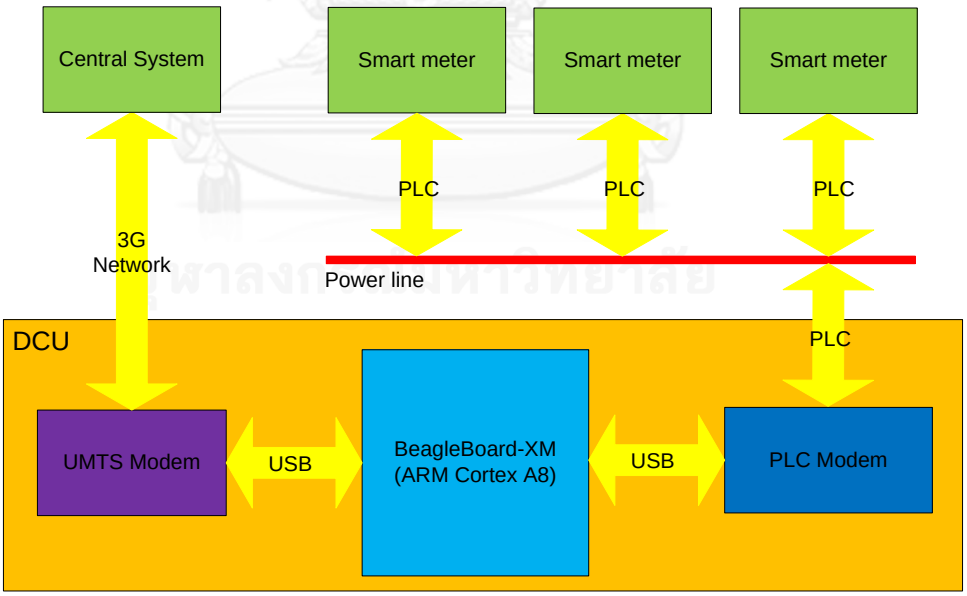
การสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA) ถูกใช้เป็นตัวกลางทางฝั่งต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และส่วนทางฝั่งระบบกลางจำลองอาจจะใช้การสื่อสารผ่านอินเทอร์เน็ตเอดีเอสแอล (ADSL) แบบใยแก้วนำแสง (FTTx) หรือแบบเคเบิล (DOCSIS) แต่ในการวิจัยนี้ใช้แบบเคเบิล

4.4. การออกแบบต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล

ในหัวข้อนี้จะกล่าวถึงการออกแบบในรายละเอียดของส่วนต่างๆ ดังต่อไปนี้ การออกแบบฮาร์ดแวร์ และซอร์ฟแวร์ ซึ่งประกอบไปด้วย ผังงานการทำงานของโปรแกรม คลังโปรแกรมที่ใช้ และซอร์ฟแวร์บนระบบปฏิบัติการอูบันตูที่เกี่ยวข้องกับอุปกรณ์เก็บรวบรวมข้อมูล

4.4.1. การออกแบบด้านฮาร์ดแวร์

การออกแบบนี้จะใช้บอร์ดพัฒนาที่มีชื่อว่า Beagleboard-xM เป็นแกนหลัก และเชื่อมต่อกับโมเด็มการสื่อสารอื่นๆ ได้แก่ โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง และโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ โดยมีองค์ประกอบต่างๆ ดังรูปที่ 4-6



รูปที่ 4-6 แผนภาพบล็อกการออกแบบฮาร์ดแวร์

ARM Cortex A8 เป็นตัวประมวลผลกลางมีสถาปัตยกรรมแบบ ARMv7-A มีหน้าที่ประมวลผลโปรแกรม และควบคุมมอดูลต่างๆ รองรับระบบปฏิบัติการลินุกซ์แบบฝังตัว (Embedded

Linux) ทำให้ผู้ใช้จัดการการทำงานของโปรแกรม และควบคุมมอดูลต่างๆ ได้ง่าย ซึ่งตัวประมวลผลกลางนี้จะอยู่บนบอร์ดพัฒนา BeagleBoard-XM

โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังมีหน้าที่รับ และส่งข้อมูลผ่านสายไฟฟ้าส่งกำลังเพื่อใช้ในการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะที่ต่อกับสายไฟฟ้าส่งกำลังในเฟสเดียวกัน

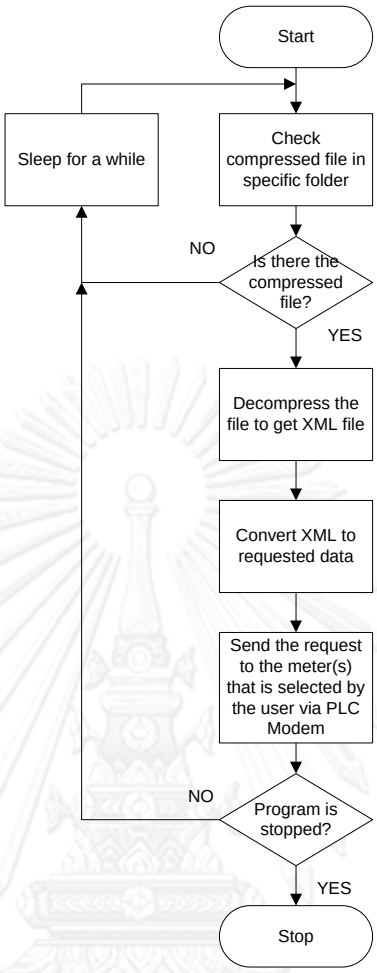
โมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA Modem) มีหน้าที่รับ และส่งข้อมูลผ่านเครือข่ายระบบโทรศัพท์เคลื่อนที่ในยุคที่ 3 เพื่อใช้ในการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์เก็บรวบรวมข้อมูล กับคอมพิวเตอร์ที่ระบบกลาง

4.4.2. **การออกแบบด้านซอฟต์แวร์**

หัวข้อย่อยนี้จะกล่าวถึงผังงานการทำงาน คลังโปรแกรมที่เกี่ยวข้อง และซอร์ฟแวร์บนระบบปฏิบัติการอูบันตูที่เกี่ยวข้องกับต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล

4.4.2.1. **ผังงานการทำงาน**

หัวข้อย่อยนี้จะกล่าวถึงการทำงานของเทรดหลัก เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง และเทรดตัวประมวลผลชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมฝั่งไคลเอนต์ของอุปกรณ์เก็บรวบรวมข้อมูล



รูปที่ 4-7 ผังงานการทำงานของเทรดหลักของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล

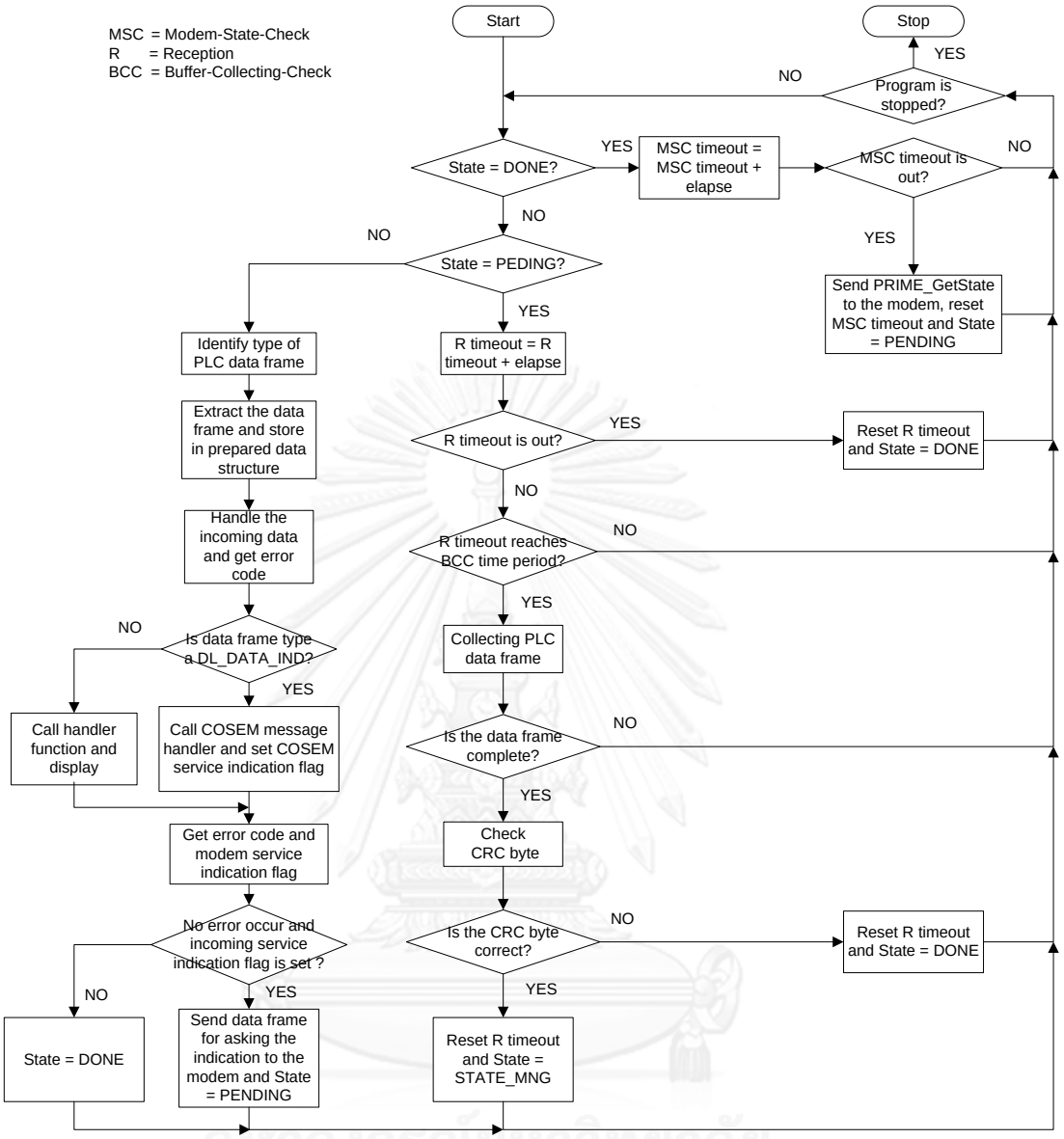
เมื่อเริ่มต้นเทรดหลักตรวจสอบหาแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ตั้งเอาไว้ ได้แก่ GetRequestNormal.xml.gz ในกรณีของคำสั่ง GET-Req, SetRequestNormal.xml.gz ในกรณีของคำสั่ง SET-Req และ ActionRequestNormal.xml.gz ในกรณีของคำสั่ง ACTION-Req ถ้าตรวจไม่พบ จะหยุดทำงานสักพักหนึ่ง แล้วจึงเริ่มหาใหม่ ถ้าตรวจพบจะทำการคลายการบีบอัดเพื่อให้ได้แฟ้มข้อมูล XML ที่บรรจุคำขอต่างๆ จากนั้นจะนำข้อมูลคำขอที่ได้ไปใส่ไว้ที่โครงสร้างข้อมูลคำขอที่ได้เตรียมไว้ เพื่อให้ง่ายแก่การเรียกใช้งาน

เมื่อเทรดทราบว่าคำขอใด ต้องขอไปที่มาตรอัจฉริยะตัวใด ก็จะเริ่มร้องขอไปทีละตัวจนครบ โดยเริ่มต้นด้วยการใส่ข้อมูลที่ได้รับมา ลงไปในโครงสร้างข้อมูลของคำสั่งดังกล่าว แล้วจึงเรียกใช้บริการของคำสั่งนั้นของชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ซึ่งเป็นการตั้งค่าตัวบ่งชี้เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมทราบว่า เทรดหลักต้องการส่งข้อมูล มันจะทำงานตามผังงานตามรูปที่ 3-7 ถึง รูปที่ 3-9

เมื่อถึงขั้นตอนสุดท้ายที่บรรยายตามรูปที่ 3-9 คือ ข้อมูลที่เข้ารหัสแล้ว หรือหน่วยข้อมูลโพรโทคอลชั้นโปรแกรมประยุกต์ (APDU) จะถูกส่งออกโดยชั้นล่างที่รองรับอีกครั้งหนึ่ง นั่นคือ เทรดจัดการชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมจะไปเรียกใช้บริการของคำสั่ง DL_Data_Request เพื่อส่งข้อมูลผ่านการสื่อสารผ่านสายไฟฟ้าส่งกำลัง บริการดังกล่าวจะสร้างกรอบข้อมูลของคำสั่ง DL_Data_Request แบบร้องขอ (Request Data Frame) และส่งไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ผ่านพอร์ตอนุกรมแบบ RS232

เมื่อโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังได้รับกรอบข้อมูลดังกล่าว มันจะตอบกลับด้วยกรอบข้อมูลของคำสั่ง DL_Data_Request แบบรายงานผล (Return Data Frame) ที่บรรจุค่ารหัสผิดพลาด (Error Code) ถ้ารหัสดังกล่าวแจ้งว่ามีข้อผิดพลาด เทรดจัดการชั้นโปรแกรมประยุกต์จะต้องแก้ไขข้อผิดพลาด และส่งข้อมูลใหม่ ถ้าไม่มีข้อผิดพลาด โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังจะพยายามส่งข้อมูลไปยังโมเด็มอีกฝั่งหนึ่ง ที่เชื่อมต่ออยู่บนสายไฟฟ้าส่งกำลังในเฟส (Phase) เดียวกัน ตามที่อยู่ที่ระบุให้มัน เมื่อส่งเสร็จ มันจะตั้งค่าตัวบ่งชี้ของคำสั่ง DL_Data_Confirm ในตัวมัน โดยเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง มีหน้าที่ที่ต้องส่งกรอบข้อมูลของคำสั่ง PRIME_GetState แบบร้องขอ ไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เพื่อสอบถามสถานะของตัวบ่งชี้ของคำสั่งทั้งหมด รวมถึงมันต้องคอยรวบรวม และตรวจสอบกรอบข้อมูลของทุกๆคำสั่งแบบรายงานผลที่ถูกส่งมาจากโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังด้วย เนื่องจากการส่งข้อมูลผ่านพอร์ตอนุกรมแบบ RS232 ข้อมูลจะไม่ถูกส่งมาทั้งหมดในคราวเดียว ขั้นตอนดังกล่าวเป็นไปตามผังงานรูปที่ 4-8

เมื่อเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังได้รับกรอบข้อมูลของคำสั่ง PRIME_GetState แบบรายงานผลแล้ว มันจะทราบว่าสามารถเรียกบริการ ของคำสั่ง DL_Data_Confirm เพื่อเรียกดูผลของการส่งข้อมูลด้วยคำสั่ง DL_Data_Request ได้ว่าส่งสำเร็จหรือไม่ จากสถานะส่งข้อมูล (Transmission Status) ซึ่งถูกบรรจุอยู่บนกรอบข้อมูลของคำสั่ง DL_Data_Confirm แบบรายงานผล ถ้าสถานะส่งข้อมูลเป็น “Sent” แสดงว่าส่งสำเร็จ ถ้าเป็น “Timeout” แสดงว่าส่งไม่สำเร็จ เทรดจัดการชั้นโปรแกรมประยุกต์ต้องเป็นผู้จัดการส่งข้อมูลใหม่อีกครั้ง จนกว่าจะส่งสำเร็จ



รูปที่ 4-8 ผังงานการทำงานของเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังของอุปกรณ์เก็บรวบรวมข้อมูล

จากผังงานรูปที่ 4-8 เมื่อเริ่มโปรแกรมเทรดติดต่อกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังเข้าสู่สถานะทำงานเสร็จ (Done) และให้ค่าตัวแปรที่ใช้เก็บเวลาที่ต้องคำสั่ง PRIME_GetState แบบร้องขอ ไปยังโมเด็มเพื่อสอบถามค่าของตัวบ่งชี้ของโมเด็ม (Modem-State-Check Timeout) และเวลาที่ตั้งเอาไว้เพื่อตรวจสอบว่าอุปกรณ์ฝั่งตรงข้ามส่งข้อมูลกลับมาหรือไม่ (Reception Timeout) เท่ากับศูนย์

เมื่อเข้าสู่ฟังก์ชันการทำงานหลักของเทรด มันจะตรวจสอบว่าตอนนี้อยู่ในสถานะใด โดยสถานะของเทรด จะมีดังนี้ ทำงานเสร็จ, กำลังทำงาน (Pending), จัดการตัวบ่งชี้ของโมเด็ม (State Management)

กรณีเทรดอยู่ในสถานะทำงานเสร็จ มันจะเพิ่มเวลาของ MSC Timeout ขึ้นเท่ากับระยะเวลาผ่านไป (Elapse Time) ซึ่งเป็นค่าคงที่ที่ใส่เข้ามาเป็นตัวแปรรับเข้าของฟังก์ชันนี้ และเป็นระยะเวลาเดียวกันกับเวลาที่ให้เทรดนอนหลับ (Sleep) ก่อนที่จะปลุกขึ้นมาทำงานอีกครั้งหนึ่ง จากนั้นมันจะตรวจสอบว่าค่า MSC Timeout ถึงค่าๆ หนึ่งที่กำหนดหรือยัง ถ้าถึงแล้วมันจะส่งกรอบข้อมูลของคำสั่ง PRIME_GetState แบบร้องขอ ไปยังโมเด็มเพื่อสอบถามค่าของตัวบ่งชี้ของโมเด็ม และเข้าสู่สถานนะ “กำลังทำงาน” เพื่อรอรับกรอบข้อมูลของคำสั่ง PRIME_GetState แบบรายงานผลจากโมเด็ม พร้อมทั้งตั้งค่า MSC Timeout เท่ากับศูนย์ แต่ถ้ายังไม่ถึงค่านั้น ก็จะจบการทำงานในรอบนี้

กรณีเทรดอยู่ในสถานะกำลังทำงาน มันจะเพิ่มเวลาของ R timeout ขึ้นเท่ากับระยะเวลาผ่านไปเช่นกัน จากนั้นมันจะตรวจสอบว่าค่า R timeout ถึงค่าๆ หนึ่งที่กำหนดไว้หรือยัง ถ้าถึงแล้ว มันจะสันนิษฐานว่าอุปกรณ์ฝั่งตรงข้ามมีปัญหาเกิดขึ้น เนื่องจากไม่ตอบกลับมาในเวลาที่กำหนด และมันจะเข้าสู่สถานะทำงานเสร็จ แต่ถ้ายังไม่ถึงค่านั้น และ R timeout ยังไม่เข้าสู่ช่วงการตรวจสอบการสะสมบัฟเฟอร์ (Buffer-Collecting-Check Time Period) ที่รับเข้ามา มันจะจบการทำงานในรอบนี้ แต่ถ้า R timeout เข้าสู่ช่วงดังกล่าวพอดี ก็จะทำหน้าที่หลักในโหมดนี้ คือ การตรวจสอบบัฟเฟอร์ที่รับเข้ามา โดยอัตโนมัติจากเทรดจัดการพอร์ตอนุกรมแบบ RS232 ของคลังโปรแกรมบูสท์ (Boost Library) ดังที่กล่าวไว้ข้างต้นว่า กรอบข้อมูลที่รับเข้ามานั้น จะไม่เข้ามาพร้อมกันในคราวเดียว เทรดต่อประสานโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังจึงต้องคอยตรวจสอบว่ากรอบข้อมูลที่อยู่ในบัฟเฟอร์นั้นครบสมบูรณ์หรือไม่ ถ้าไม่ครบให้รอก่อน แล้วเข้ามาตรวจสอบในรอบถัดไป แต่ถ้าครบแล้วมันจะตรวจสอบไบท์ CRC ว่ามีค่าถูกต้องหรือไม่ ซึ่งนั้นก็หมายถึงว่ากรอบข้อมูลที่รับเข้ามานั้นถูกต้องสมบูรณ์หรือไม่ ถ้าผลตรวจ CRC ไม่ถูกต้อง มันจะรีเซ็ต R timeout เท่ากับศูนย์ และเข้าสู่สถานะทำงานเสร็จ แต่ถ้าถูกต้อง มันจะรีเซ็ต R timeout เท่ากับศูนย์เช่นกัน และเข้าสู่สถานะจัดการตัวบ่งชี้ของโมเด็มต่อไป

กรณีเทรดอยู่ในสถานะจัดการตัวบ่งชี้ของโมเด็ม เทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง จะตรวจสอบรูปแบบของกรอบข้อมูลของคำสั่งที่ผ่านการตรวจสอบความสมบูรณ์และความถูกต้องในช่วงเวลาตรวจสอบการสะสมบัฟเฟอร์ที่รับเข้ามาแล้ว ว่าเป็นรูปแบบใด เมื่อทราบแล้วจะเรียกใช้บริการของคำสั่งนั้นเพื่อแยกข้อมูลรายงานผลที่ได้จากโมเด็ม ไปเก็บไว้ที่โครงสร้างข้อมูลที่ได้เตรียมเอาไว้ จากนั้นจะเรียกบริการที่สอดคล้องกับรูปแบบของกรอบข้อมูลของคำสั่งต่างๆ ยกตัวอย่างเช่น ในกรณีกรอบข้อมูลของคำสั่ง DL_Data_Confirm แบบรายงานผล เทรดจะไปเรียกบริการที่แสดงผลบนเอาท์พุตมาตรฐาน (Standard Output) ว่าการส่งข้อมูลด้วยคำสั่ง DL_Data_Request แบบร้องขอครั้งล่าสุดนั้นสำเร็จหรือไม่ หรือในกรณีกรอบข้อมูลของคำสั่ง DL_Data_Indication แบบรายงานผล เทรดต่อประสานกับโมเด็มการสื่อสารผ่าน

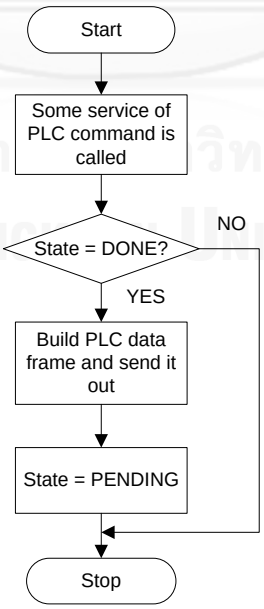
สายไฟฟ้าส่งกำลัง จะไปเรียกบริการการถอดรหัสหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ พร้อมทั้งตั้งค่าตัวบ่งชี้ของชั้นโปรแกรมประยุกต์ เพื่อแจ้งให้เทรดชั้นโปรแกรมประยุกต์รับทราบว่ามีหน่วยข้อมูลโพรโทคอลชั้นโปรแกรมประยุกต์เข้ามาใหม่

หลังจากนั้นมันจะตรวจสอบว่าเกิดการผิดพลาดในการสื่อสารกับโมเด็มหรือไม่ ผ่านรหัสผิดพลาด และตรวจสอบตัวบ่งชี้ของโมเด็มว่าถูกตั้งค่าหรือไม่ ในกรณีที่กรอบข้อมูลของคำสั่งที่รับเข้ามา คือ PRIME_GetState แบบรายงานผล หรืออาจจะเป็นกรอบข้อมูลคำสั่งอื่นๆ แบบรายงานผลก็ได้ (ในกรณีที่ตัวบ่งชี้ของโมเด็มถูกตั้งค่า หลายๆ ค่าพร้อมๆ กัน) ถ้าผลการตรวจสอบพบว่าไม่มีข้อผิดพลาด และมีตัวบ่งชี้ค่าใดค่าหนึ่งถูกตั้งค่า มันจะเรียกใช้บริการของคำสั่งที่ตัวบ่งชี้ถูกตั้งค่า เพื่อสร้างกรอบข้อมูลของคำสั่งดังกล่าวแบบร้องขอ และส่งไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เพื่อให้โมเด็มส่งกรอบข้อมูลของคำสั่งนั้นๆ แบบรายงานผลกลับมา และเทรดจะเข้าสู่สถานนะ “กำลังทำงาน” เพื่อรอรับกรอบข้อมูลแบบรายงานผล ที่ส่งกลับมาจากโมเด็ม แต่ถ้าผลการตรวจสอบเป็นกรณีอื่นๆ เทรดจะเข้าสู่สถานนะ “ทำงานเสร็จ” เพื่อเริ่มทำงานในรอบต่อไป

ปัจจัยต่างๆ ที่มีผลต่อการเปลี่ยนแปลงสถานะ (State) ของเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง มีดังต่อไปนี้

1. การเปลี่ยนแปลงสถานะโดยตัวเทรดเอง ดังผังงานรูปที่ 4-8 ที่ได้อธิบายไปด้านบน

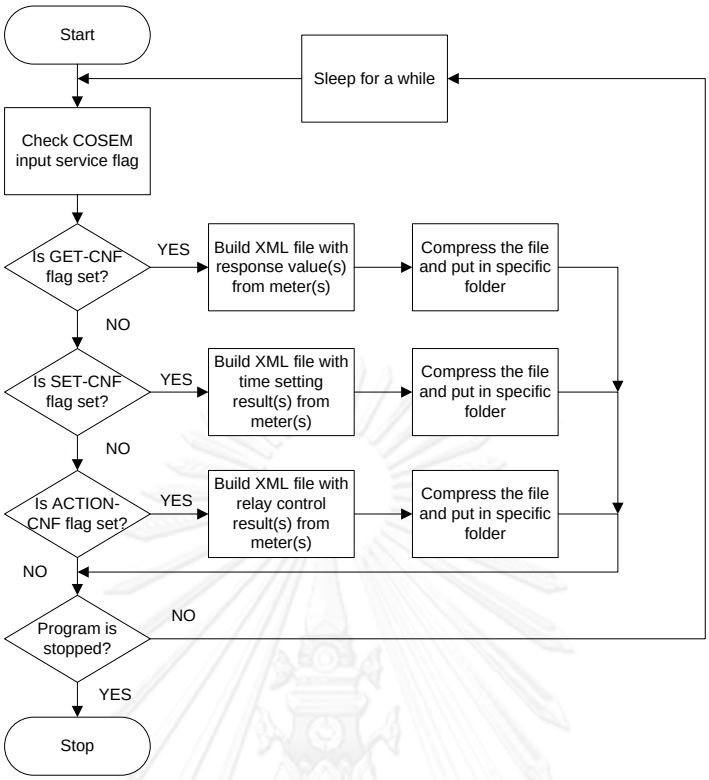
2.การเปลี่ยนแปลงสถานะเมื่อบริการของคำสั่งการสื่อสารผ่านสายไฟฟ้าส่งกำลังถูกเรียกใช้จากภายนอก ดังรูปที่ 4-9



รูปที่ 4-9 ผังงานการเรียกใช้บริการใดๆ ของการสื่อสารผ่านสายไฟฟ้าส่งกำลังที่ส่งผลต่อการเปลี่ยนแปลงสถานะของเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง

จากผังงานรูปที่ 4-9 เมื่อบริการของคำสั่งใดๆ ของการสื่อสารผ่านสายไฟฟ้าส่งกำลังถูกเรียกใช้เมื่อเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังอยู่ในสถานนะ “ทำงานเสร็จ” เทรดจะสร้างกรอบข้อมูลของคำสั่งนั้นๆ แบบร้องขอ และส่งไปยังโมเด็ม พร้อมทั้งเปลี่ยนสถานนะเข้าสู่ “กำลังทำงาน” แต่ถ้าเทรดไม่ได้อยู่ในสถานะ “ทำงานเสร็จ” การเรียกใช้บริการในครั้งนี้จะถือว่าไม่มีผลใดๆ ต้องเรียกใหม่ในภายหลัง

เมื่อส่งข้อมูลของคำสั่งร้องขอของชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมด้วยโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ไปยังมาตรอุปกรณ์ได้สำเร็จ มันจะตอบกลับด้วยคำสั่งตอบสนองซึ่งถูกบรรจุอยู่บนกรอบข้อมูลของคำสั่ง DL_Data_Indication แบบรายงานผล และเมื่อเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังได้แยกหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ และเรียกใช้บริการการถอดรหัสหน่วยข้อมูลโพรโตคอลชั้นโปรแกรมประยุกต์ พร้อมทั้งตั้งค่าตัวบ่งชี้ของชั้นโปรแกรมประยุกต์แล้ว ดังที่กล่าวไปด้านบน เทรดจัดการชั้นโปรแกรมประยุกต์จะรับทราบว่ามีหน่วยข้อมูลโพรโตคอลใหม่เข้ามา มันจะเข้าไปจัดการหน่วยข้อมูลโพรโตคอลที่ถูกถอดรหัสข้อมูลแล้ว ซึ่งวิธีการจัดการก็ขึ้นอยู่กับว่าคำสั่งตอบสนองดังกล่าว ต้องใช้บริการในกลุ่มใดในการถอดรหัส เช่น ถ้าต้องใช้บริการในกลุ่มที่ 1 ดังตารางที่ 3-1 มันจะย้ายข้อมูลที่ถอดรหัสแล้วไปยังโครงสร้างข้อมูลชั้นโปรแกรมประยุกต์ แล้วจึงตั้งค่าตัวบ่งชี้ชั้นตัวประมวลผลชั้นโปรแกรมประยุกต์ แต่ถ้าใช้บริการในกลุ่มที่ 2 ดังตารางที่ 3-2 มันค่อยๆ จัดเรียงบล็อกข้อมูลย่อยไปไว้ยังโครงสร้างข้อมูลชั้นโปรแกรมประยุกต์ จนกว่าจะได้ข้อมูลที่สมบรูณ์ แล้วจึงตั้งค่าตัวบ่งชี้ชั้นตัวประมวลผลชั้นโปรแกรมประยุกต์เช่นกัน เพื่อแจ้งให้ตัวประมวลชั้นโปรแกรมประยุกต์ทราบว่าข้อมูลที่รับเข้ามาใหม่พร้อมใช้งานแล้ว ให้เข้าไปตรวจสอบได้ที่โครงสร้างข้อมูลชั้นโปรแกรมประยุกต์ที่เตรียมไว้ เพื่อใช้ในการประมวลผลต่อไป ดังรูปที่ 4-10



รูปที่ 4-10 ผังงานการทำงานของเทรดตัวประมวลผลชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมของอุปกรณ์เก็บรวบรวมข้อมูล

เมื่อตัวประมวลชั้นโปรแกรมประยุกต์ทราบว่ามีข้อมูลใหม่เข้ามา จากการตรวจสอบตัวบ่งชี้ ถ้าตัวบ่งชี้ที่ถูกตั้งค่าเป็นของคำสั่ง GET-Cnf มันจะนำค่าการวัดของมาตรอัจฉริยะที่อ่านได้บรรจุลงในแฟ้มข้อมูล XML หลังจากนั้นทำการบีบอัดข้อมูล และใส่ลงในโฟลเดอร์ที่เตรียมไว้ แต่ถ้าตัวบ่งชี้ที่ถูกตั้งค่าเป็นคำสั่ง SET-Cnf และ ACTION-Cnf เทรดจะนำผลการตั้งค่าฐานเวลาของมาตรอัจฉริยะ และผลการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะ ตามลำดับ บรรจุลงในแฟ้มข้อมูลแทน ซึ่งแต่ละแฟ้มข้อมูลที่ถูกบีบอัดแล้วจะมีชื่อต่างกัน ได้แก่ GetResponseNormal.xml.gz, SetResponseNormal.xml.gz และ ActionResponseNormal.xml.gz หลังจากนั้นเทรดจะตรวจสอบว่าโปรแกรมถูกหยุดการทำงานหรือไม่ ถ้าไม่ เทรดจะหลับไปสักพักหนึ่ง ก่อนจะตื่นมาตรวจสอบตัวบ่งชี้อีกครั้งในรอบถัดไป แต่ถ้าโปรแกรมถูกหยุดการทำงาน เทรดนี้จะจบการทำงานไปด้วย

4.4.2.2. **คลังโปรแกรมที่เกี่ยวข้อง**

หัวข้อย่อยนี้จะกล่าวถึงคลังโปรแกรมที่ถูกใช้ในการพัฒนาต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล ซึ่งถูกเขียนด้วยภาษา C++ มาตรฐาน ทั้งหมด ดังต่อไปนี้

DLMS/COSEM Library (C++ มาตรฐาน) เป็นคลังโปรแกรมแบบ DLL ที่ให้บริการต่างๆ ในชั้นโปรแกรมประยุกต์

Boost Library version 1.47 (C++ มาตรฐาน) ใช้คลังโปรแกรมย่อยดังนี้

Boost-asio ใช้สำหรับรับ-ส่งพอร์ตอนุกรม

Boost-iostreams เป็นแฟ้มข้อมูลที่ คลังโปรแกรม gzip ต้องเรียกใช้

Boost-thread ใช้เพื่อสร้าง และควบคุมการทำงานของเทรด

ST PLC Library (C++ มาตรฐาน) ใช้ติดต่อสื่อสารกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง

Gzip Library (C++ มาตรฐาน) ใช้เพื่อบีบอัด และคลายการบีบอัดข้อมูล gzip

TinyXML2 Library (C++ มาตรฐาน) ใช้เพื่อสร้างแฟ้มข้อมูลชนิด XML เพื่อบรรจุข้อมูลที่ต้องการสื่อสาร

4.4.2.3. **ซอร์ฟแวร์โปรแกรมประยุกต์บนระบบปฏิบัติการอูบันตูที่เกี่ยวข้อง**

หัวข้อย่อยนี้จะกล่าวถึงซอร์ฟแวร์โปรแกรมประยุกต์บนระบบปฏิบัติการอูบันตูที่ถูกใช้ เพื่อช่วยให้อุปกรณ์เก็บรวบรวมข้อมูลสามารถสื่อสารผ่านระบบอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอได้ ดังต่อไปนี้

wvdial ใช้เพื่อเชื่อมอินเตอร์ผ่านโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ (UMTS/HSPDA modem)

openvpn ใช้เพื่อเชื่อมต่อไปยังเครือข่ายส่วนบุคคลเสมือน

4.5. การออกแบบมาตรอัจฉริยะจำลอง

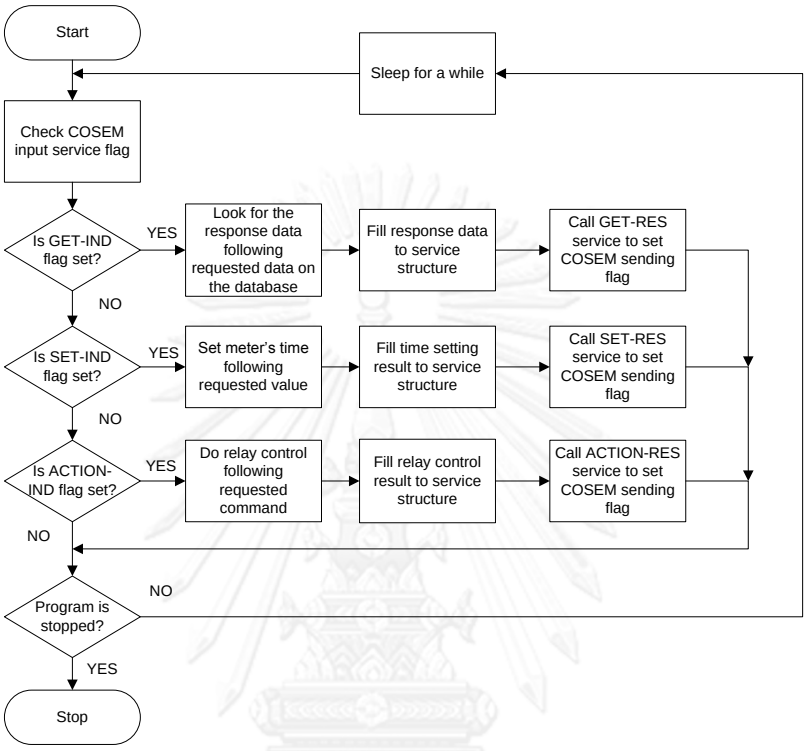
ในหัวข้อนี้จะกล่าวถึงการออกแบบในรายละเอียดของส่วนต่างๆ ดังต่อไปนี้ การออกแบบซอร์ฟแวร์ ซึ่งประกอบไปด้วย ผังงานการทำงานของโปรแกรม และคลังโปรแกรมที่ใช้ของมาตรอัจฉริยะจำลอง

4.5.1. **การออกแบบด้านซอฟต์แวร์**

หัวข้อย่อยนี้จะกล่าวถึงผังงานการทำงาน และคลังโปรแกรมที่เกี่ยวข้องกับมาตรอัจฉริยะจำลอง

4.5.1.1. **ผังงานการทำงาน**

หัวข้อย่อยนี้จะกล่าวถึงการทำงานของเทรดตัวประมวลผลชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมฝั่งเซิร์ฟเวอร์ของมาตรอัจฉริยะ



รูปที่ 4-11 ผังงานการทำงานของเทรดตัวประมวลผลชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมของมาตรอัจฉริยะจำลอง

เมื่อข้อมูลจากฝั่งอุปกรณ์เก็บรวบรวมข้อมูลถูกส่งมาถึงโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังฝั่งมาตรอัจฉริยะ เทรดต่อประสานกับโมเด็มต้องร้องขอข้อมูลดังกล่าวไปยังโมเด็ม โดยขั้นตอนการติดต่อสื่อสารระหว่างเทรดต่อประสานกับโมเด็ม กับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ในการรับกรอบข้อมูลจากโมเด็มฝั่งมาตรอัจฉริยะจะทำงานเหมือนกันกับผังงานรูปที่ 4-8 และรูปที่ 4-9 ที่ได้บรรยายไว้ด้านบนแล้ว

เมื่อหน่วยข้อมูลโพรโทคอลชั้นโปรแกรมประยุกต์ถูกถอดรหัส และตัวบ่งชี้ชั้นโปรแกรมประยุกต์ถูกตั้งค่า โดยเทรดต่อประสานกับโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังฝั่งมาตรอัจฉริยะแล้ว และเทรดจัดการชั้นโปรแกรมประยุกต์ตรวจพบ มันจะเรียกใช้บริการในการจัดการข้อมูลให้พร้อมสำหรับตัวประมวลผลชั้นโปรแกรมประยุกต์นำไปประมวลผลต่อไป โดยขั้นตอนการ

ทำงานของเทรดจัดการชั้นโปรแกรมประยุกต์นั้น ได้บรรยายไว้แล้วดังรูปที่ 3-7 และ รูปที่ 3-10 ถึง รูปที่ 3-11เมื่อเทรดจัดการชั้นโปรแกรมประยุกต์นำข้อมูลไปใส่ในโครงสร้างข้อมูลชั้นโปรแกรมประยุกต์เรียบร้อย มันตั้งค่าตัวบ่งชี้ของคำสั่งที่รับเข้ามาของตัวประมวลผลชั้นโปรแกรมประยุกต์ เพื่อแจ้งให้ตัวประมวลชั้นโปรแกรมประยุกต์ทราบว่าข้อมูลพร้อมแล้ว

จากรูปที่ 4-11 เมื่อตัวประมวลชั้นโปรแกรมประยุกต์ตรวจพบว่าตัวบ่งชี้ของคำสั่งใดถูกตั้งค่า มันจะเข้าไปอ่านข้อมูลในโครงสร้างข้อมูลของคำสั่งนั้นที่ได้เตรียมเอาไว้ ว่าความต้องการของคำร้องขอที่ส่งมาคืออะไร

ในกรณีตัวบ่งชี้ของคำสั่ง GET_Ind ถูกตั้งค่า ตัวประมวลชั้นโปรแกรมประยุกต์จะนำค่าชื่อตรรกะ (Logical Name), เลขคลาสต้นแบบ (Class Id), และเลขลำดับของคุณสมบัติ (Attribute Id) จากโครงสร้างข้อมูล เพื่อใช้ในการอ้างถึงค่าของอ็อบเจกต์ที่อยู่ในฐานข้อมูลที่ตัวมันต่ออยู่ (เปรียบเสมือนเป็นค่าที่ได้จากการวัดจริงแล้วนำมาบันทึกไว้) ซึ่งเป็นค่าของอ็อบเจกต์ที่ถูกร้องขอจากอุปกรณ์เก็บรวบรวมข้อมูล จากนั้นเมื่อได้ข้อมูลที่ต้องการแล้ว มันจะนำค่าเหล่านั้นไปใส่ในโครงสร้างข้อมูลของคำสั่งตอบสนอง นั่นคือ GET_Res แล้วจึงเรียกใช้บริการของคำสั่งดังกล่าวของชั้นโปรแกรมประยุกต์ เพื่อตั้งค่าตัวบ่งชี้ชั้นโปรแกรมประยุกต์ เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า มันต้องการส่งข้อมูลออกไปยังอุปกรณ์เก็บรวบรวมข้อมูล

แต่ถ้าเป็นกรณีตัวบ่งชี้ของคำสั่ง SET_Ind ถูกตั้งค่า ตัวประมวลชั้นโปรแกรมประยุกต์จะนำค่าชื่อตรรกะ (Logical Name), เลขคลาสต้นแบบ (Class Id), และเลขลำดับของคุณสมบัติ (Attribute Id) จากโครงสร้างข้อมูล เพื่อใช้ในการอ้างถึงอ็อบเจกต์ที่อยู่ในฐานข้อมูลที่ตัวมันต่ออยู่เช่นกัน แต่จะเป็นการเข้าถึงแบบเข้าไปเปลี่ยนแปลงค่าของอ็อบเจกต์ที่อ่านถึงแทน แล้วจึงนำผลการเปลี่ยนแปลงค่านั้นว่าสำเร็จหรือไม่ ส่งกลับไปยังอุปกรณ์เก็บรวบรวมข้อมูล โดยการนำผลการเปลี่ยนแปลงไปใส่ในโครงสร้างข้อมูลของคำสั่งตอบสนอง นั่นคือ SET_Res แล้วจึงเรียกใช้บริการของคำสั่งดังกล่าวของชั้นโปรแกรมประยุกต์ เพื่อตั้งค่าตัวบ่งชี้ชั้นโปรแกรมประยุกต์ เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า มันต้องการส่งข้อมูลออกไปยังอุปกรณ์เก็บรวบรวมข้อมูล

สุดท้ายในกรณีตัวบ่งชี้ของคำสั่ง ACTION_Ind ถูกตั้งค่า ตัวประมวลชั้นโปรแกรมประยุกต์จะนำค่าชื่อตรรกะ (Logical Name), เลขคลาสต้นแบบ (Class Id), และเลขลำดับของกระบวนการ (Method Id) จากโครงสร้างข้อมูล เพื่อใช้ในการอ้างถึงอ็อบเจกต์ที่อยู่ในฐานข้อมูลที่ตัวมันต่ออยู่ แต่จะเป็นการเข้าถึงแบบเข้าไปเรียกใช้บริการของอ็อบเจ็กต์ดังกล่าวแทน แล้วจึงนำผลการเรียกใช้บริการดังกล่าวว่าสำเร็จหรือไม่ ส่งกลับไปยังอุปกรณ์เก็บรวบรวมข้อมูล โดยการนำผลการเปลี่ยนแปลงไปใส่ในโครงสร้างข้อมูลของคำสั่งตอบสนอง นั่นคือ ACTION_Res แล้วจึงเรียกใช้บริการของคำสั่งดังกล่าวของชั้นโปรแกรมประยุกต์ เพื่อตั้งค่าตัวบ่งชี้ชั้นโปรแกรมประยุกต์ เพื่อแจ้งให้เทรดจัดการชั้นโปรแกรมประยุกต์ทราบว่า มันต้องการส่งข้อมูลออกไปยังอุปกรณ์เก็บรวบรวมข้อมูล

เมื่อเทรดจัดการชั้นโปรแกรมประยุกต์ฝั่งเซิร์ฟเวอร์ตรวจสอบตัวบ่งชี้ชั้นโปรแกรมประยุกต์ และทราบว่า จะต้องส่งข้อมูลดังกล่าวกลับไปยังอุปกรณ์เก็บรวบรวมข้อมูล มันจะดำเนินการตามขั้นตอนการทำงานดังผังงานรูปที่ 3-7 และรูปที่ 3-12 ดังที่ได้บรรยายไว้ข้างต้นแล้ว ซึ่งในขั้นตอนสุดท้ายเทรดจัดการชั้นโปรแกรมประยุกต์จะเรียกใช้บริการของคลังโปรแกรมการสื่อสารผ่านสายไฟฟ้าส่งกำลังเพื่อสร้างกรอบข้อมูลของคำสั่ง DL_Data_Request แบบร้องขอไปยังโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เพื่อส่งข้อมูลดังกล่าวกลับไปยังอุปกรณ์เก็บรวบรวมข้อมูล

ในระหว่างการทำงานของตัวประมวลชั้นโปรแกรมประยุกต์ หากโปรแกรมถูกปิดลง จะทำให้การทำงานของตัวประมวลผลชั้นโปรแกรมประยุกต์หยุดการทำงานไปด้วย ถ้าหากโปรแกรมยังทำงานต่อไป ตัวประมวลชั้นโปรแกรมประยุกต์ จะทำงานในรอบนั้นๆ จนเสร็จและหลับไปสักพัก แล้วจึงตื่นมาตรวจสอบตัวบ่งชี้ของตัวมันเองอีกครั้งในรอบถัดไป

4.5.1.2. **คลังโปรแกรมที่เกี่ยวข้อง**

หัวข้อย่อยนี้จะกล่าวถึงคลังโปรแกรมที่ถูกใช้ในการพัฒนามาตรอัจฉริยะจำลอง ซึ่งประกอบด้วยทั้งภาษา C++ มาตรฐาน และ Managed C++ ดังต่อไปนี้

DLMS/COSEM Library (C++ มาตรฐาน) เป็นคลังโปรแกรมแบบ DLL ที่ให้บริการต่างๆ ในชั้นโปรแกรมประยุกต์

ST PLC Library (Managed C++) เป็นคลังโปรแกรมเดียวกันกับคลังโปรแกรมในข้อ 4.4.2 แต่ถูกปรับเปลี่ยนเพื่อให้ทำงานร่วมกับ Managed C++ ได้

Windows Forms (.NET Framework, Managed C++) เป็นคลังโปรแกรมที่ใช้สร้าง GUI ของมาตรอัจฉริยะ

Base Class Library (.NET Framework, Managed C++) เป็นคลังโปรแกรมที่ให้บริการคำสั่งทั่วไป ที่ประกอบด้วยหลายคลังโปรแกรมย่อยๆ โดยคลังโปรแกรมย่อยที่ระบบจำลองเรียกใช้คำสั่งที่สำคัญ ดังนี้

System.IO.Ports.SerialPort เป็นคลังโปรแกรมย่อยที่ให้บริการคำสั่งรับ-ส่งพอร์ตอนุกรม

System.Threading เป็นคลังโปรแกรมย่อยที่ให้บริการคำสั่งสร้าง และควบคุมการทำงานของเทรด

ADO.NET (.NET Framework, Managed C++) เพื่อเชื่อมต่อกับฐานข้อมูลที่เก็บข้อมูลอ็อบเจกต์ที่อยู่ภายในมาตรอัจฉริยะจำลอง

4.6. การออกแบบระบบกลางจำลอง

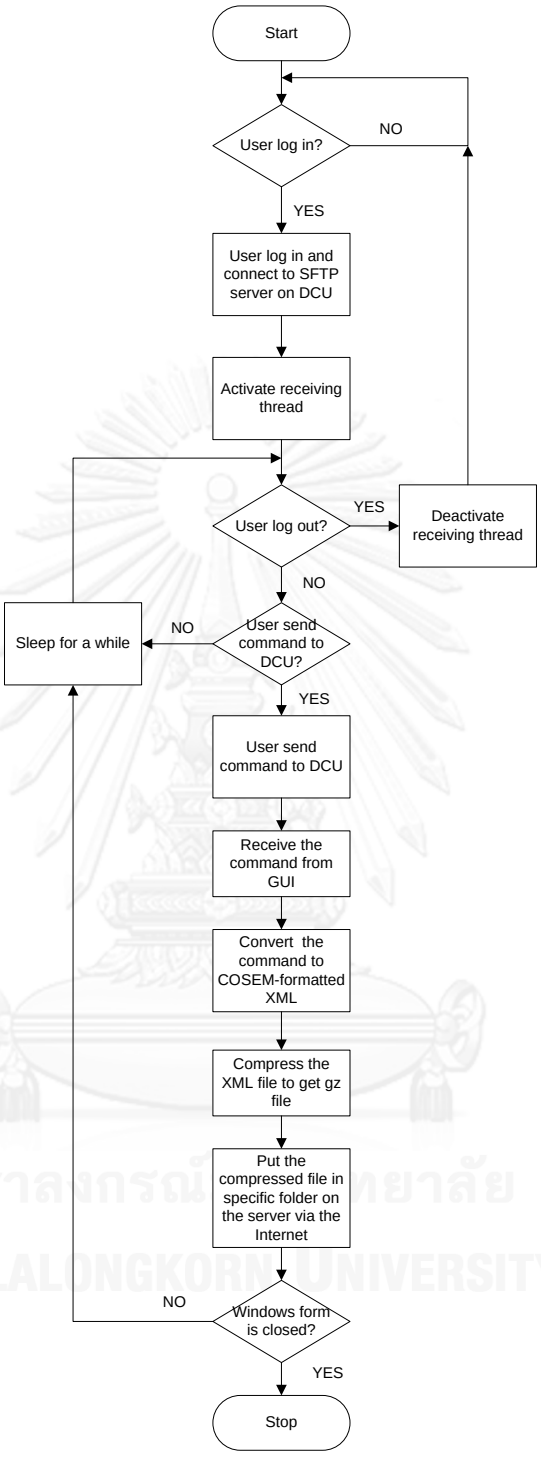
ในหัวข้อนี้จะกล่าวถึงการออกแบบในรายละเอียดของส่วนต่างๆ ดังต่อไปนี้ การออกแบบซอร์ฟแวร์ ซึ่งประกอบไปด้วย ผังงานการทำงานของโปรแกรม และคลังโปรแกรมที่ใช้ของระบบกลางจำลอง

4.6.1. **การออกแบบด้านซอฟต์แวร์**

หัวข้อย่อยนี้จะกล่าวถึงผังงานการทำงาน และคลังโปรแกรมที่เกี่ยวข้องกับระบบกลางจำลอง

4.6.1.1. **ผังงานการทำงาน**

หัวข้อย่อยนี้จะกล่าวถึงการทำงานของเทรดหลัก และเทรดตรวจสอบผลตอบสนองจากอุปกรณ์เก็บรวบรวมข้อมูลของระบบกลางจำลอง

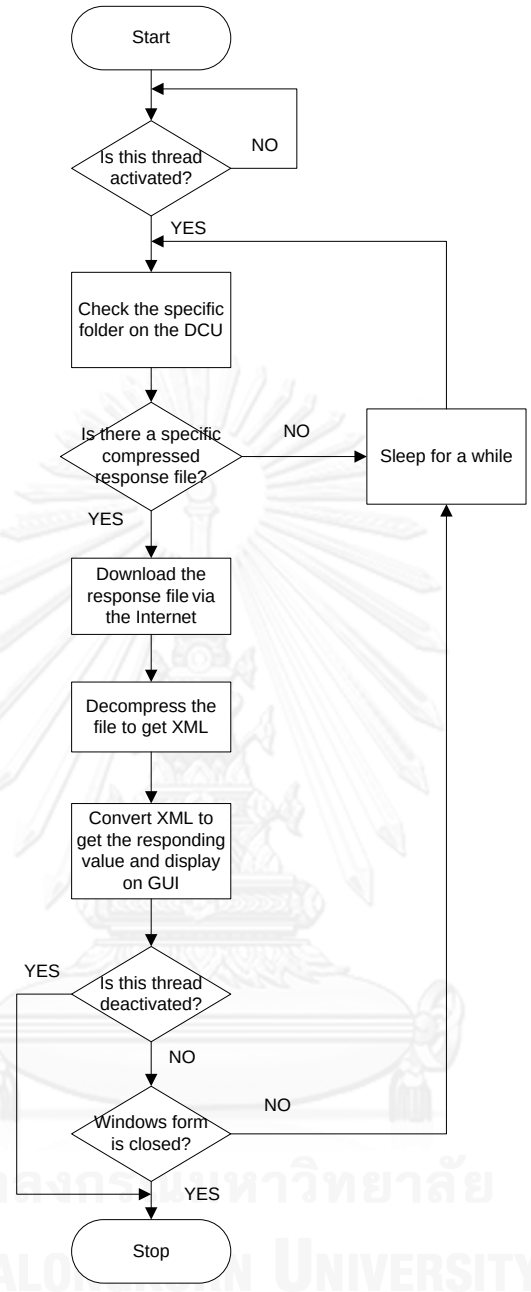


รูปที่ 4-12 ผังงานการทำงานของเทรดหลักของระบบกลางจำลอง

จากรูปที่ 4-12 การทำงานเริ่มต้นด้วยผู้ใช้ กรอกชื่อผู้ใช้ (Username) รหัสผู้ใช้ (Password) และเลขที่อยู่ไอพี (IP Address) ของเซิร์ฟเวอร์โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต (SFTP Server) ลงในหน้าต่างส่วนติดต่อผู้ใช้ (Graphic User Interface, GUI) แล้วกดลงบันทึกเข้า (Log In) เพื่อเชื่อมต่อไปยังเซิร์ฟเวอร์โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ตบนอุปกรณ์เก็บรวบรวมข้อมูล เมื่อผู้กดปุ่มเพื่อเชื่อมต่อนั้น เทรดหลักของระบบกลางจำลองจะสร้างเทรดตรวจสอบผลตอบสนองจากอุปกรณ์เก็บรวบรวมข้อมูล ซึ่งเป็นเทรดย่อย และสั่งให้เริ่มทำงานตามผังงานการทำงานในรูปที่ 4-13

หลังจากนั้นเทรดหลักจะตรวจสอบว่าผู้ใช้ได้กดลงบันทึกออก (Log Out) หรือไม่ ถ้ากด เทรดหลักจะหยุดการทำงานของเทรดตรวจสอบผลตอบสนองจากอุปกรณ์เก็บรวบรวมข้อมูล และกลับเข้าจุดเริ่มต้น แต่ถ้าผู้ใช้ไม่ได้กด เทรดหลักจะตรวจสอบต่อไปว่าผู้ใช้ได้กดส่งคำร้องขอไปยังอุปกรณ์เก็บรวบรวมข้อมูลหรือไม่ ถ้าไม่มีการกดส่ง เทรดหลักจะหลับไปสักพักหนึ่ง ก่อนตื่นมาทำงานในรอบถัดไป แต่ถ้ามีการกดส่ง เทรดหลักจะทำการเปลี่ยนข้อมูลคำร้องขอให้อยู่ในรูปของแฟ้มข้อมูล XML จากนั้นจะทำการบีบอัดแฟ้มข้อมูล XML ดังกล่าวให้มีขนาดเล็กลง เพื่อสะดวกในการส่งข้อมูล และส่งข้อมูลที่ถูกบีบอัดไปไว้ในโฟลเดอร์ที่เตรียมเอาไว้ของเซิร์ฟเวอร์โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต หรือก็คือ อุปกรณ์เก็บรวบรวมข้อมูลนั่นเอง จากนั้นเทรดหลักจะตรวจสอบว่าโปรแกรมถูกหยุดทำงานโดยผู้ใช้หรือไม่ ถ้าไม่ เทรดหลักจะหลับไปสักพักก่อนจะตื่นมาทำงานในรอบถัดไป แต่ถ้าโปรแกรมถูกหยุด เทรดหลักจะถูกหยุดการทำงานไปด้วย

จากรูปที่ 4-13 เทรดตรวจสอบผลตอบสนองจากอุปกรณ์เก็บรวบรวมข้อมูลจะตรวจสอบหาแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ตกลงกันเอาไว้ และอยู่ในโฟลเดอร์ที่ได้เตรียมเอาไว้แล้ว ว่ามีหรือไม่ ถ้าไม่มี มันจะหลับไปสักพักหนึ่ง แต่ถ้ามี มันจะบรรจุลง (Download) มาเก็บไว้ที่คอมพิวเตอร์เครื่องที่มันกำลังทำงานอยู่ และทำการคลายการบีบอัดแฟ้มข้อมูลเพื่อให้ได้แฟ้มข้อมูล XML หลังจากนั้นเปลี่ยนแฟ้มข้อมูล XML นั้นให้เป็นข้อมูลผลตอบสนอง และไปเก็บไว้ในโครงสร้างข้อมูลของคำสั่งที่ได้เตรียมเอาไว้ พร้อมทั้งนำค่าไปแสดงผลบนส่วนติดต่อผู้ใช้ จากนั้นมันจะตรวจสอบว่าตัวมันถูกหยุดการทำงานโดยเทรดหลักหรือไม่ ถ้าใช่ มันจะจบการทำงาน แต่ถ้าไม่ใช่ มันจะตรวจสอบต่อไปว่าโปรแกรมถูกหยุดหรือไม่ ถ้าใช่ มันก็จะจบการทำงานเช่นเดียวกัน แต่ถ้าไม่ใช่ มันจะหลับไปสักพักหนึ่งก่อนจะตื่นขึ้นมาทำงานในรอบต่อไป



รูปที่ 4-13 ผังงานการทำงานของเทรดตรวจสอบผลตอบสนองจากอุปกรณ์เก็บรวบรวมข้อมูลของระบบกลางจำลอง

4.6.1.2. **คลังโปรแกรมที่เกี่ยวข้อง**

หัวข้อย่อยนี้จะกล่าวถึงคลังโปรแกรมที่ถูกใช้ในการพัฒนาระบบกลางจำลองดังต่อไปนี้

Windows Forms (.NET Framework, managed C#) เป็นคลังโปรแกรมที่ใช้สร้าง GUI ของมาตรอัจฉริยะ

Base Class Library (.NET Framework, managed C#) เป็นคลังโปรแกรมที่ให้บริการคำสั่งทั่วไป ที่ประกอบด้วยหลายคลังโปรแกรมย่อยๆ โดยคลังโปรแกรมย่อยที่ระบบจำลองเรียกใช้คำสั่งที่สำคัญ ดังนี้

System.Xml เป็นคลังโปรแกรมย่อยสำหรับคำสั่งสร้างแฟ้มข้อมูลชนิด XML

System.IO.Compression.GZipStream เป็นคลังโปรแกรมย่อยสำหรับคำสั่งบีบอัดข้อมูลแบบ GZip

System.Threading เป็นคลังโปรแกรมย่อยที่ให้บริการคำสั่งสร้าง และควบคุมการทำงานของเทรด

SFTP Library – SharpSSH (managed C#) เป็นคลังโปรแกรมที่ให้บริการคำสั่งในการส่งแฟ้มข้อมูลด้วยโพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต บนโพรโตคอลทีซีพี/ไอพี

บทที่ 5

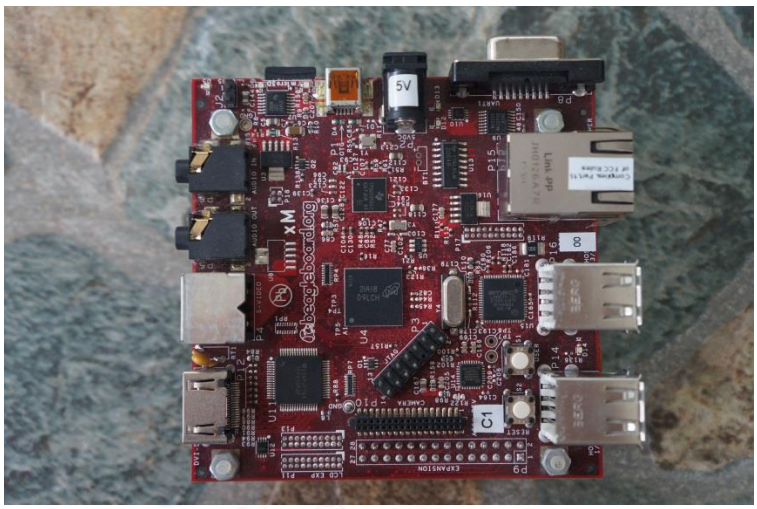
การทดสอบ และผลการทดสอบ

ในบทนี้จะกล่าวถึง การตั้งค่า การทดสอบ และผลการทดสอบการทำงานร่วมกันและการสื่อสารของอุปกรณ์ทั้งสาม ซึ่งลักษณะการเชื่อมต่อของอุปกรณ์ทั้งสามเป็นไปตามรูปที่ 4-2 และรูปที่ 4-3 ซึ่งประกอบด้วยซอร์ฟแวร์โปรแกรมประยุกต์ระบบกลางจำลองทำงานอยู่บนคอมพิวเตอร์ส่วนบุคคล ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล และซอร์ฟแวร์โปรแกรมประยุกต์มาตร-อัจฉริยะจำลอง 3 ชุด บนคอมพิวเตอร์ส่วนบุคคล โดยการทดลองจะถูกแบ่งหัวข้อออกตามการเรียกใช้บริการของคำสั่ง ดังนี้ การใช้บริการของคำสั่ง GET SET และ ACTION

นอกจากนี้ยังมีการทดสอบนำคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โค-เซมที่พัฒนาขึ้น ไปใช้ในโปรแกรมประยุกต์สำหรับการอ่านค่าโพรไฟล์ต่าง ๆ จากมาตรอัจฉริยะที่ใช้งานจริงในอุตสาหกรรมอีกด้วย

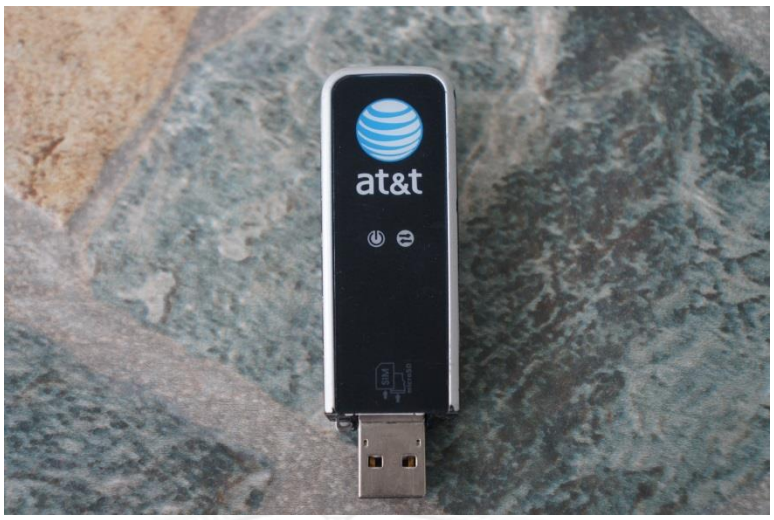
5.1. การตั้งค่าอุปกรณ์ต่างๆ

ในการทดสอบนี้จะใช้ฮาร์ดแวร์ต่างๆ ดังนี้ คอมพิวเตอร์ส่วนบุคคลหนึ่งเครื่อง เพื่อจำลองการทำงานของมาตรอัจฉริยะจำลองทั้งสามเครื่อง และระบบกลางจำลอง หรือใช้คอมพิวเตอร์ส่วนบุคคลหลายเครื่องก็ได้ (เพื่อให้เห็นภาพการทำงานจริง) บอร์ด Beagleboard-xM ดังรูปที่ 5-1 ที่ลงระบบปฏิบัติการอูบันตู เพื่อสร้างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล สาย USB-to-RS232 Converter ดังรูปที่ 5-2 สำหรับเชื่อมต่อบอร์ด Beagleboard-xM กับคอมพิวเตอร์ส่วนบุคคลเพื่อส่งคำสั่งไปยังบอร์ด และแสดงผลไปยังคอมพิวเตอร์ โมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ ดังรูปที่ 5-3 ใช้เชื่อมต่อบอร์ด Beagleboard-xM เข้ากับอินเทอร์เน็ต สายแลนใช้เชื่อมต่อคอมพิวเตอร์ส่วนบุคคลที่โปรแกรมประยุกต์ระบบกลางจำลองทำงานอยู่ และบอร์ด Beagleboard-xM (อีกทางเลือกในการทดสอบ) เข้ากับอินเทอร์เน็ต และโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง ดังรูปที่ 5-4 ใช้เป็นตัวกลางในการสื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลองทั้งสามเครื่อง



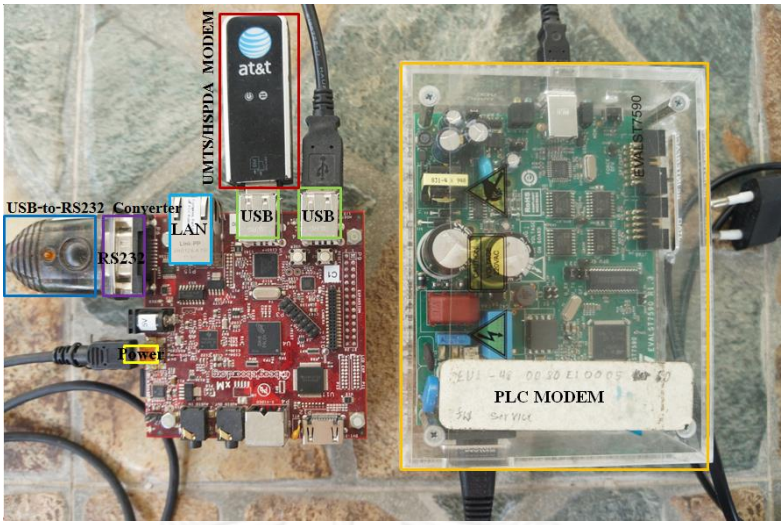
รูปที่ 5-1 บอร์ด BeagleBoard-XM

รูปที่ 5-2 สาย USB-to-RS232 Converter



รูปที่ 5-3 โมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสพีดีเอ

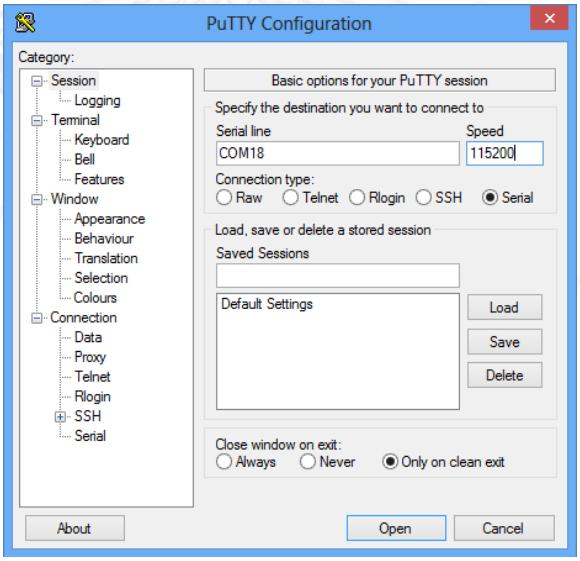
รูปที่ 5-4 โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง



รูปที่ 5-5 บอร์ด Beagleboard-xM ที่ถูกเชื่อมต่อด้วยอุปกรณ์ต่างๆ

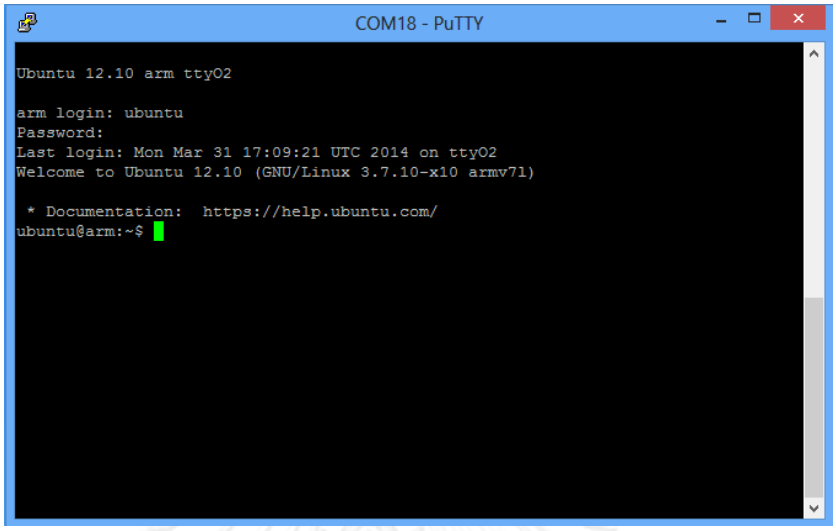
เมื่อเชื่อมต่ออุปกรณ์ต่างๆ เข้ากับบอร์ด Beagleboard-xM จะได้ต้นแบบอุปกรณ์เก็บรวมรวบข้อมูล ดังรูปที่ 5-5

เริ่มต้นด้วยการเข้าถึงบอร์ด Beagleboard-xM ผ่านสาย USB-to-RS232 Converter ซึ่งด้านที่เป็นหัว DB-9 จะต่อเข้ากับตัวบอร์ด Beagleboard-xM ส่วนด้านที่เป็น USB จะต่อเข้ากับคอมพิวเตอร์ส่วนบุคคลที่ได้ติดตั้งโปรแกรมประยุกต์ชื่อ putty เอาไว้



รูปที่ 5-6 หน้าต่างตั้งค่าการเชื่อมต่อของโปรแกรมประยุกต์ putty

จากรูปที่ 5-6 ต้องกรอกค่าในช่อง Serial line ให้ตรงกับ COM Port ที่สาย USB-to-RS232 ต่ออยู่ และในช่อง Speed ให้ใส่ 115200 แล้วจึงกดปุ่ม Open



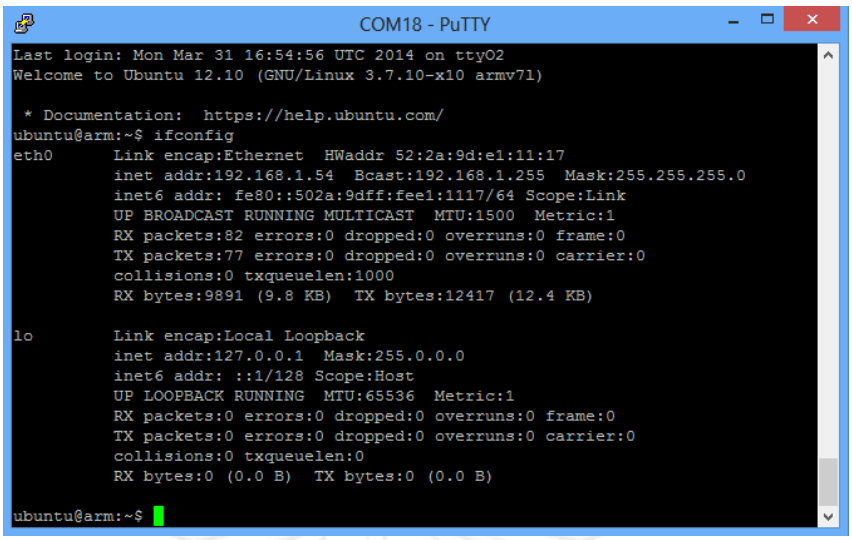
รูปที่ 5-7 หน้าต่างแสดงผลของโปรแกรมประยุกต์ putty เมื่อเชื่อมต่อเข้ากับระบบปฏิบัติการอูบันตู บนบอร์ด Beagleboard-xM

จากนั้นให้กรอกชื่อผู้ใช้ (Username) และรหัสผู้ใช้ (Password) ของระบบปฏิบัติการอูบันตู จากนั้นจะพบกับหน้าต่างดังรูปที่ 5-7

การเชื่อมต่ออินเทอร์เน็ตของบอร์ด Beagleboard-xM ด้วยการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอในปัจจุบันของประเทศไทยยังไม่มีเสถียรที่ดีพอ การวิจัยนี้จึงขอเพิ่มทางเลือกอีกหนึ่งวิธี คือ การทดสอบในเครือข่ายแลนเดียวกัน

เริ่มต้นด้วยวิธีการเชื่อมต่ออินเทอร์เน็ตแบบแลน ให้ต่อสายแลนจากอุปกรณ์จัดเส้นทาง (Router) หรือฮับ (Hub) ที่อยู่ในเครือข่ายแลนเดียวกันกับคอมพิวเตอร์ส่วนบุคคลที่ติดตั้งโปรแกรมประยุกต์ระบบกลางจำลอง

จากนั้นพิมพ์คำสั่ง ifconfig เพื่อดูเลขที่อยู่ไอพี (IP Address) ของบอร์ด Beagleboard-xM ดังรูปที่ 5-8



รูปที่ 5-8 ผลการใช้คำสั่ง ifconfig บนระบบปฏิบัติการอูบันตู เพื่อดูเลขที่อยู่ไอพีของบอร์ด BeagleBoard-XM

หรืออีกวิธีหนึ่ง คือ การเชื่อมต่อการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอ วิธีนี้ต้องเรียกใช้โปรแกรมประยุกต์ wvdial และ openConnect ของระบบปฏิบัติการอูบันตูของบอร์ด Beagleboard-xM เพื่อเชื่อมต่ออินเทอร์เน็ต และเซิร์ฟเวอร์วีพีเอ็น (VPN server) ตามลำดับ ในการทดสอบนี้จะเชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นของจุฬาลงกรณ์มหาวิทยาลัยที่ที่อยู่ vpn.chula.ac.th รวมถึงบนคอมพิวเตอร์ส่วนบุคคลที่มีโปรแกรมประยุกต์ระบบกลางจำลองทำงานอยู่ต้องเชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นเดียวกัน เพื่อให้เสมือนว่าอุปกรณ์ทั้งสอง เชื่อมต่ออยู่ในเครือข่ายแลนเดียวกัน

การเชื่อมต่อการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอด้วยโปรแกรมประยุกต์ wvdial นั้น จำเป็นต้องมีแฟ้มข้อมูลบทคำสั่ง (Script) โดยการสร้างแฟ้มข้อมูลบทคำสั่งใดๆ จะต้องใช้โปรแกรมประยุกต์แบบบรรณาธิการ (Editor) ในการทดสอบนี้จะใช้โปรแกรมประยุกต์ที่มีชื่อว่า vim แฟ้มข้อมูลบทคำสั่งที่ต้องใช้ร่วมกับโปรแกรมประยุกต์ wvdial มีดังนี้

wvdial.conf เป็นบทคำสั่งที่จัดเก็บการตั้งค่าของโปรแกรมประยุกต์ wvdial เพื่อใช้ในการเชื่อมต่ออินเทอร์เน็ต บทคำสั่งนี้สามารถเก็บไว้ในโฟลเดอร์ใดก็ได้ แต่ในการทดสอบเก็บไว้ที่ /etc/wvdial.conf โดยมีเนื้อความดังรูปที่ 5-9 จากรูปเป็นการตั้งค่าสำหรับซิมเครือข่ายดีแทค แต่ถ้าเป็นซิมของเคือข่ายอื่น การตั้งค่าจะต่างกันไปดังนี้

การตั้งค่าสำหรับเครือข่ายดีแทค มีรายละเอียดดังนี้

APN: internet

Access Number: *99#

Username: dtac

Password: dtac

การตั้งค่าสำหรับเครือข่ายเอไอเอส มีรายละเอียดดังนี้

APN: internet

Access Number: *99***1#

Username: ais

Password: ais

การตั้งค่าสำหรับเครือข่ายทรู มีรายละเอียดดังนี้

APN: hinternet

Access Number: *99***1#

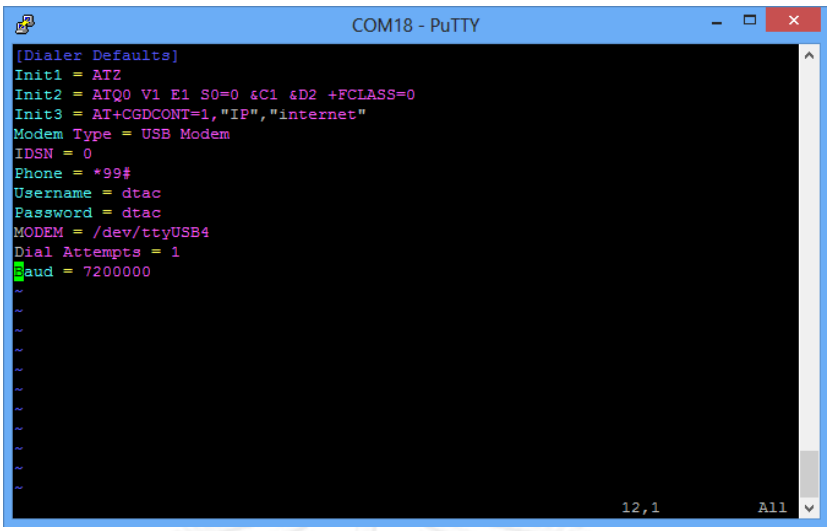
Username: true

Password: true

การตั้งค่าของเครือข่ายดีแทค และเอไอเอสไม่จำเป็นต้องใช้ค่า Username และ Password แต่โปรแกรมประยุกต์ wvdial บังคับให้ใส่ค่าดังกล่าว

ค่า APN คือ ค่าสุดท้ายของบรรทัด Init3 ในรูปที่ 5-9

ค่า MODEM ในรูปที่ 5-9 ต้องกำหนดให้ตรงกับพอร์ตของบอร์ด Beagleboard-xM ที่ต่อโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอ

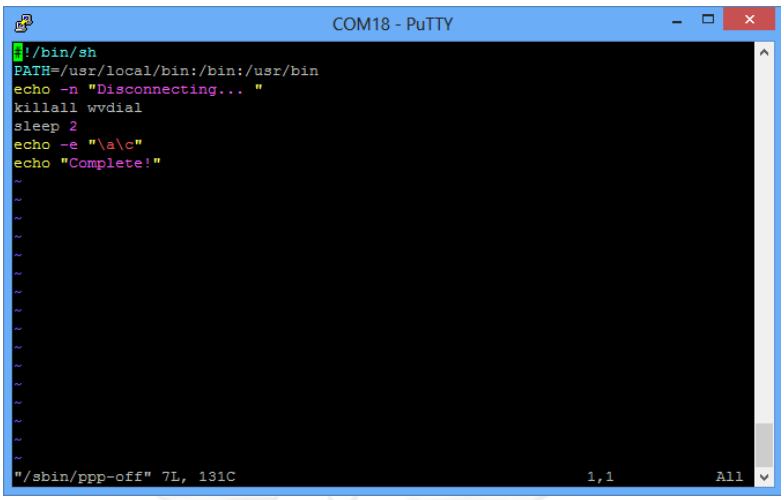


รูปที่ 5-9 เนื้อความบทคำสั่ง wvdial.conf

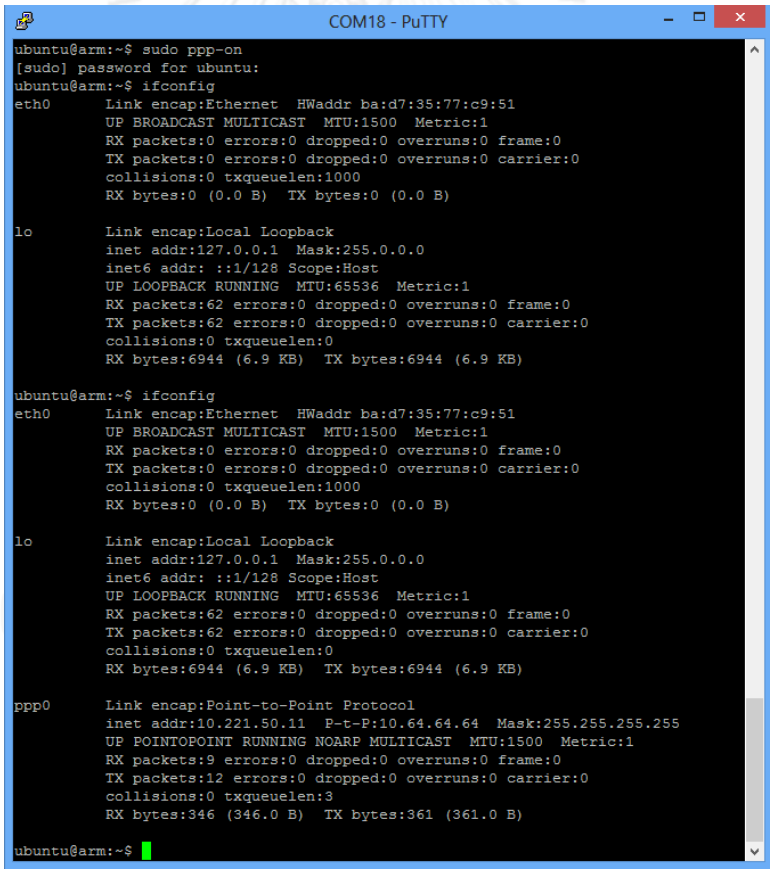
ppp-on เป็นบทคำสั่งสำหรับใช้สั่งให้โปรแกรมประยุกต์ wvdial เชื่อมต่ออินเทอร์เน็ตแบบทำงานในภาวะพื้นหลัง (Background) เนื่องจากปกติแล้วโปรแกรมประยุกต์ wvdial เมื่อผู้ใช้สั่งเชื่อมต่ออินเทอร์เน็ตโดยใช้คำสั่งโดยตรง มันจะเข้ายึดส่วนเฝ้าคุม (Console) ทำให้ผู้ใช้ไม่สามารถพิมพ์คำสั่งใดๆ ได้อีก บทคำสั่งนี้ต้องเก็บไว้ในโฟลเดอร์ /sbin/ppp-on โดยมีเนื้อความดังรูปที่ 5-10

รูปที่ 5-10 เนื้อความบทคำสั่ง ppp-on

ppp-off เป็นบทคำสั่งสำหรับใช้สั่งให้โปรแกรมประยุกต์ wvdial ตัดการเชื่อมต่ออินเทอร์เน็ต บทนี้คำสั่งนี้ต้องเก็บไว้ในโฟลเดอร์ /sbin/ppp-off โดยมีเนื้อความดังรูปที่ 5-11



รูปที่ 5-11 เนื้อความบทคำสั่ง ppp-off



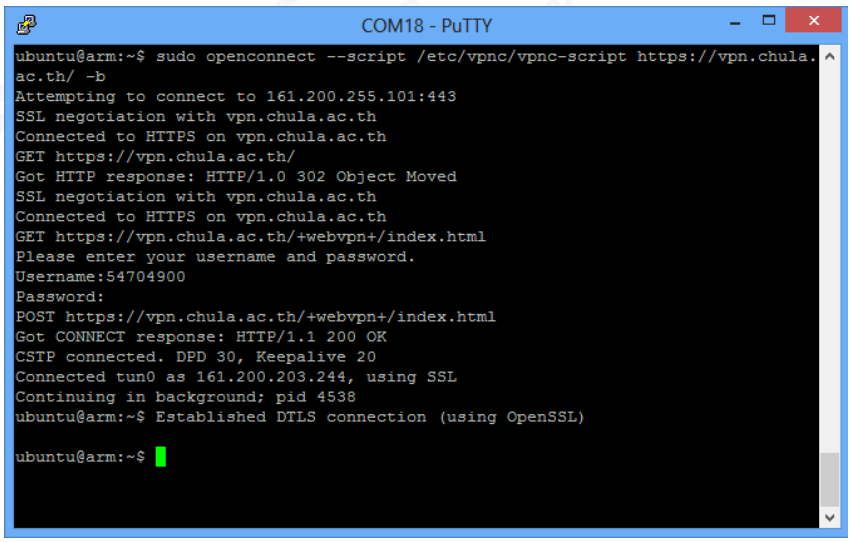
รูปที่ 5-12 เชื่อมต่ออินเทอร์เน็ตผ่านโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอ สำเร็จ

ให้เชื่อมต่ออินเทอร์เน็ตผ่านโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็มทีเอส/เอชเอสดีพีเอ โดยใช้คำสั่ง ppp-on เพื่อเรียกบทคำสั่งที่ได้สร้างเอาไว้ก่อนหน้า เมื่อเรียกเสร็จบทคำสั่งจะคืนส่วนเฝ้าคุมให้กับผู้ใช้ทันที แต่การเชื่อมต่อนั้นยังไม่เรียบร้อยดูได้จากการเรียกใช้คำสั่ง ifconfig เพื่อดูเลขที่อยู่ไอพีที่บอร์ด Beagleboard-xM ได้รับ ทันทีหลังจากใช้คำสั่ง ppp-on เสร็จ ปรากฏว่ายังไม่ได้รับเลขที่ไอพี แต่เมื่อรอสักครู่หนึ่ง แล้วจึงเรียกใช้คำสั่ง ifconfig อีกครั้ง คราวนี้ได้รับเลขที่อยู่ไอพีโพรโทคอลการสื่อสารแบบจุดต่อจุด (Point-to-Point Protocol, PPP) มา แสดงว่าการเชื่อมต่ออินเทอร์เน็ตเสร็จสมบูรณ์ ดังรูปที่ 5-12

ขั้นตอนต่อไป คือ การเชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นของจุฬาลงกรณ์มหาวิทยาลัย โดยใช้โปรแกรมประยุกต์ที่ชื่อว่า openconnect และจำเป็นต้องมีบทคำสั่งที่ชื่อว่า vpnc-script สำหรับโปรแกรมประยุกต์นี้ด้วย สามารถบรรจุลงได้จากเวปไซต์ http://git.infradead.org/users/dwmw2/vpnc-scripts.git/blob_plain/HEAD:/vpnc-script บทคำสั่งนี้สามารถเก็บไว้ในโฟลเดอร์ใดก็ได้ แต่การทดสอบนี้จะเก็บไว้ที่ /etc/vpnc/vpnc-script

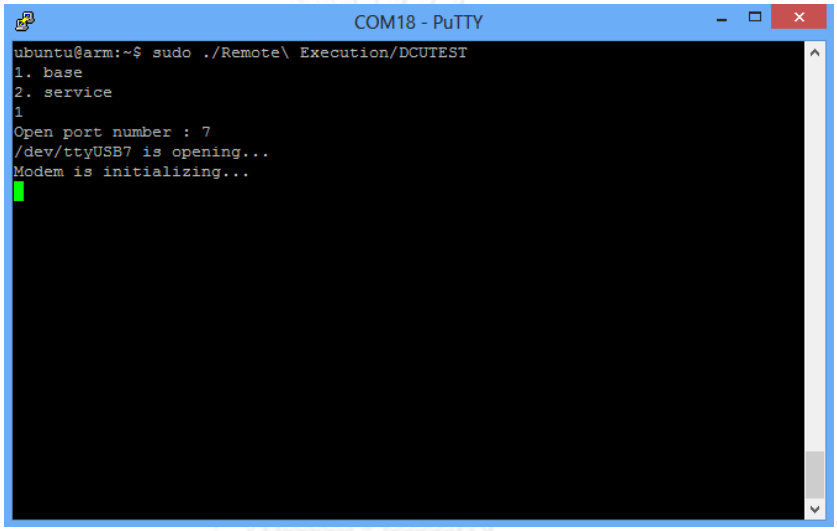
เชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นของจุฬาลงกรณ์มหาวิทยาลัย โดยใช้คำสั่ง openconnect --script /etc/vpnc/vpnc-script vpn.chula.ac.th –b เมื่อเชื่อมต่อไปยังเซิร์ฟเวอร์ได้ จะต้องกรอกชื่อผู้ใช้ และรหัสผู้ใช้ของนิสิต หรือบุคลากรจุฬาฯ เมื่อกรอกเสร็จ และเชื่อมต่อสำเร็จ บอร์ด Beagleboard-xM จะได้รับเลขที่อยู่ไอพีใหม่ นั่นคือ 161.200.203.244 ซึ่งเป็นเลขที่อยู่ไอพีภายในวงแลนเสมือนของเซิร์ฟเวอร์จุฬาฯ หลังจากนั้นโปรแกรมประยุกต์ openconnect จะเข้าสู่การทำงานในภาวะพื้นหลังดังรูปที่ 5-13

คำสั่ง sudo ที่อยู่ก่อนหน้า openconnect หมายความว่าต้องการให้โปรแกรมประยุกต์ openconnect ถูกอนุญาตให้ทำงาน โดยผู้ใช้แบบราก (root) กล่าวคือ โปรแกรมประยุกต์ดังกล่าว จะสามารถเข้าถึงทรัพยากร หรือแฟ้มข้อมูลต่างๆ ได้ทั้งหมด



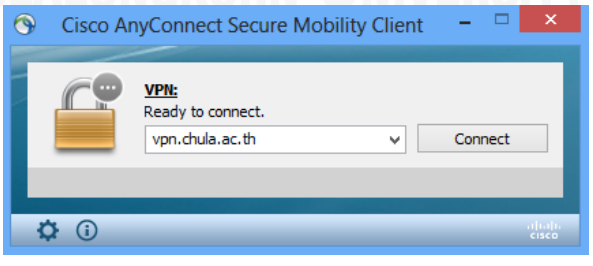
รูปที่ 5-13 ผลการเชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นของจุฬาฯ

หลังจากนั้นจึงสั่งโปรแกรมประยุกต์ของอุปกรณ์เก็บรวบรวมข้อมูลเพื่อให้ทำงาน โดยใช้คำสั่ง ./Remote\ Execution/DCUTEST (แฟ้มข้อมูลที่บรรจุรหัสคำสั่ง (Code) มีชื่อว่า DCUTEST ที่อยู่ในโฟลเดอร์ชื่อ Remote Execution) ดังรูปที่ 5-14



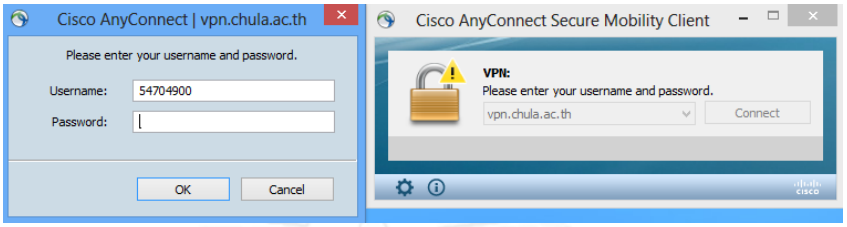
รูปที่ 5-14 โปรแกรมประยุกต์อุปกรณ์เก็บรวบรวมข้อมูล เริ่มทำงาน

ส่วนการเชื่อมต่อไปยังเซิร์ฟเวอร์วีพีเอ็นของจุฬาฯ ทางด้านคอมพิวเตอร์ส่วนบุคคลที่โปรแกรมประยุกต์ระบบกลางจำลองทำงานอยู่ จะใช้โปรแกรมประยุกต์ที่มีชื่อว่า Cisco AnyConnect Secure Mobility Client เมื่อเปิดโปรแกรมประยุกต์ขึ้นมาจะพบหน้าต่างดังรูปที่ 5-15

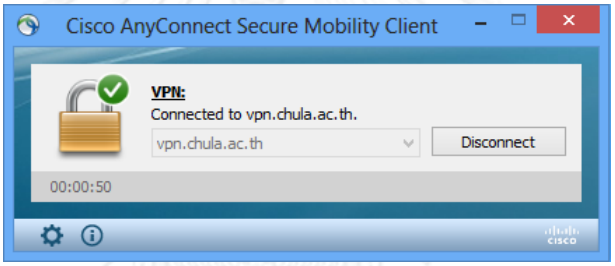


รูปที่ 5-15 หน้าต่างโปรแกรมประยุกต์ AnyConnect

กดเชื่อมต่อ หลังจากโปรแกรมประยุกต์ตรวจพบเซิร์ฟเวอร์ มันจะร้องขอชื่อผู้ใช้และรหัสผู้ใช้ ให้กรอกรหัสนิสิต หรือบุคลากรจุฬาฯ เพื่อเข้าใช้งานดังรูปที่ 5-16 ถ้าใส่รหัสถูกต้อง จะสามารถเชื่อมต่อเซิร์ฟเวอร์วีพีเอ็นได้สำเร็จดังรูปที่ 5-17

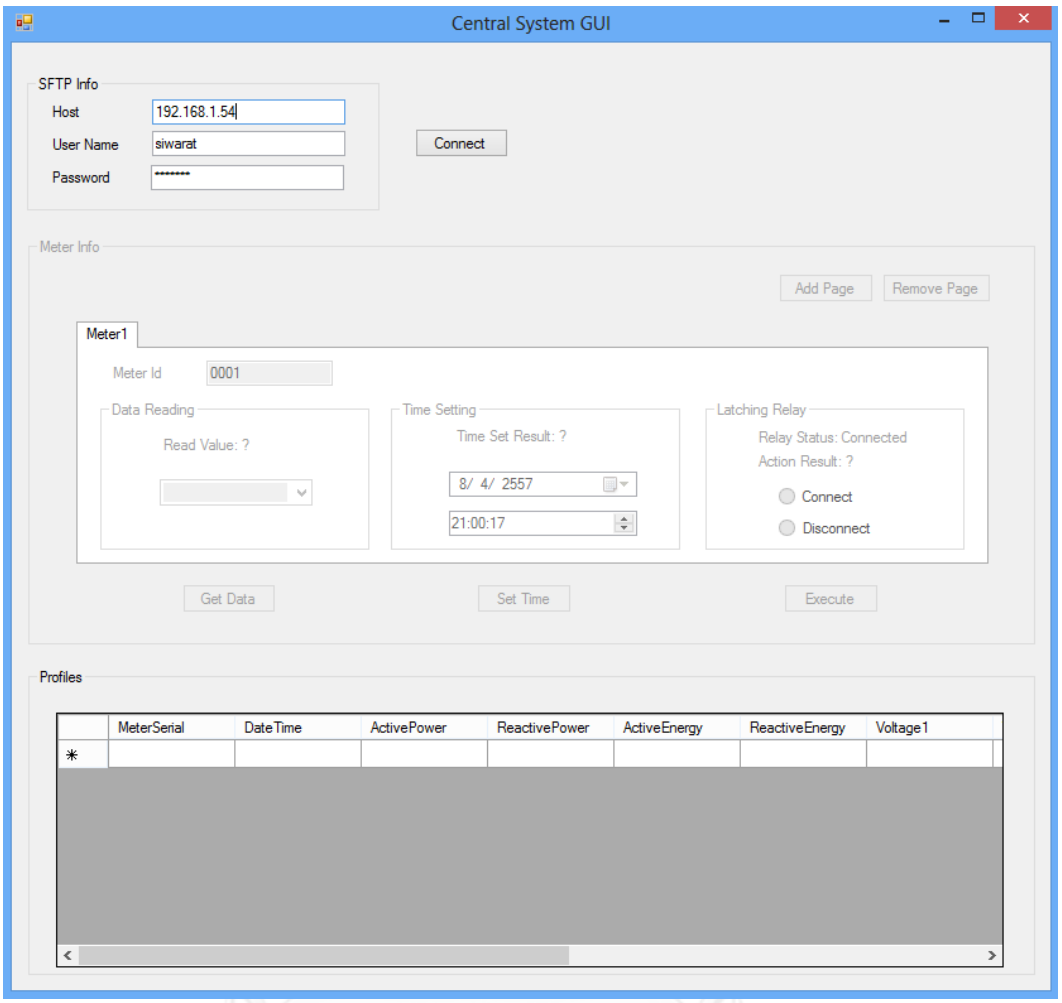


รูปที่ 5-16 โปรแกรมประยุกต์ AnyConnect ร้องขอชื่อผู้ใช้ และรหัสผู้ใช้

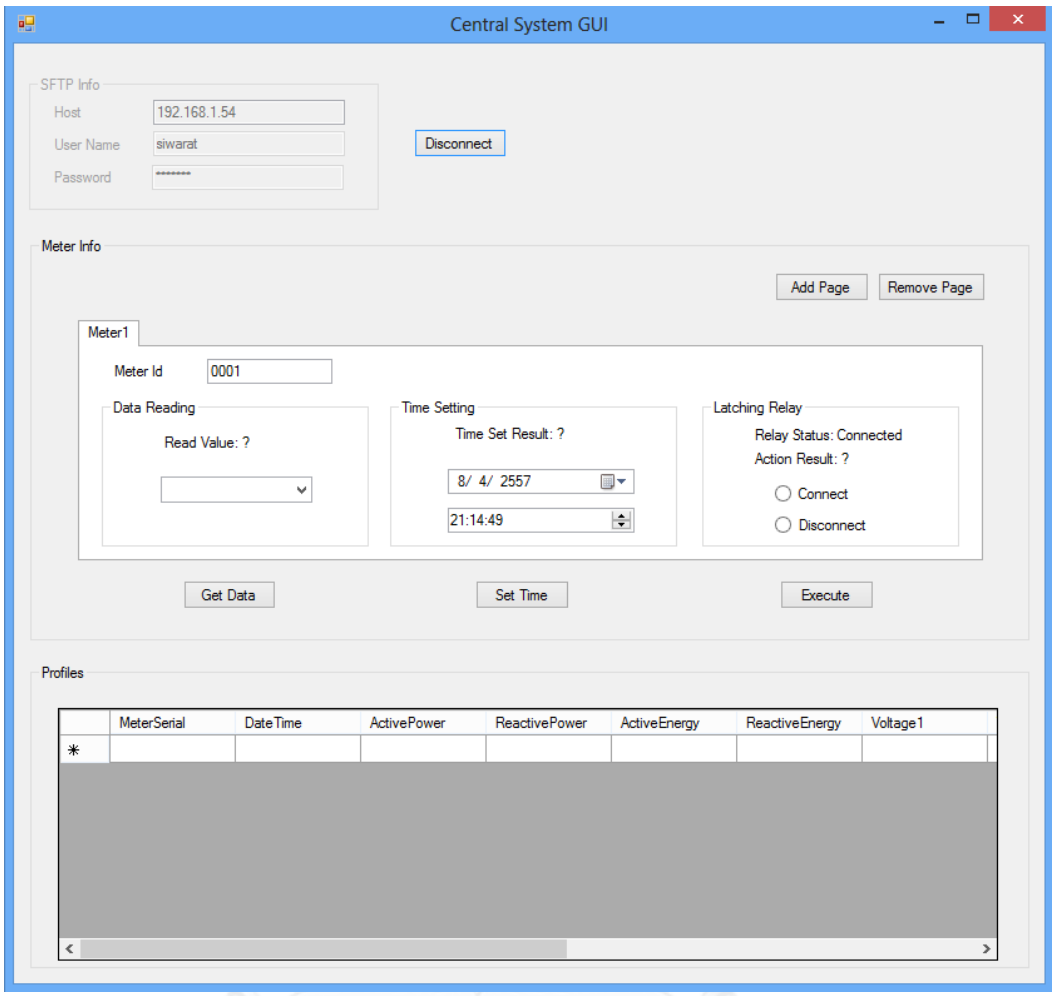


รูปที่ 5-17 การเชื่อมต่อเซิร์ฟเวอร์วีพีเอ็นสำเร็จ

หลังจากนั้นเปิดโปรแกรมประยุกต์ระบบกลางจำลองบนคอมพิวเตอร์ส่วนบุคคล จะพบหน้าต่างโปรแกรมประยุกต์ดังรูปที่ 5-18 จากนั้นให้กรอกข้อมูลในกรอบด้านบนสุด ที่เขียนว่าข้อมูลโพรโตคอลการถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต (SFTP Info) ในช่องแรก ให้ใส่เลขที่อยู่ไอพีของบอร์ด Beagleboard-xM ช่องที่สองและสาม ให้ใส่ชื่อผู้ใช้ และรหัสผู้ใช้ของระบบปฏิบัติการอูบันตู ตามลำดับ จากนั้นกดปุ่มเชื่อมต่อ (Connect) หน้าต่างโปรแกรมประยุกต์จะเปลี่ยนไปดังรูปที่ 5-19

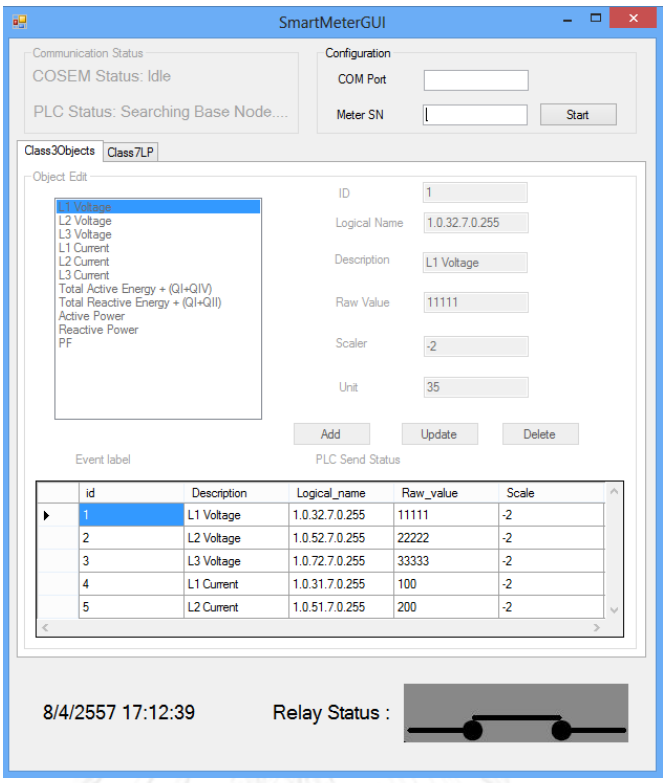


รูปที่ 5-18 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลองบนคอมพิวเตอร์ส่วนบุคคล ในตอนเริ่มต้น



รูปที่ 5-19 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลองบนคอมพิวเตอร์ส่วนบุคคล เมื่อเชื่อมต่อเสร็จ

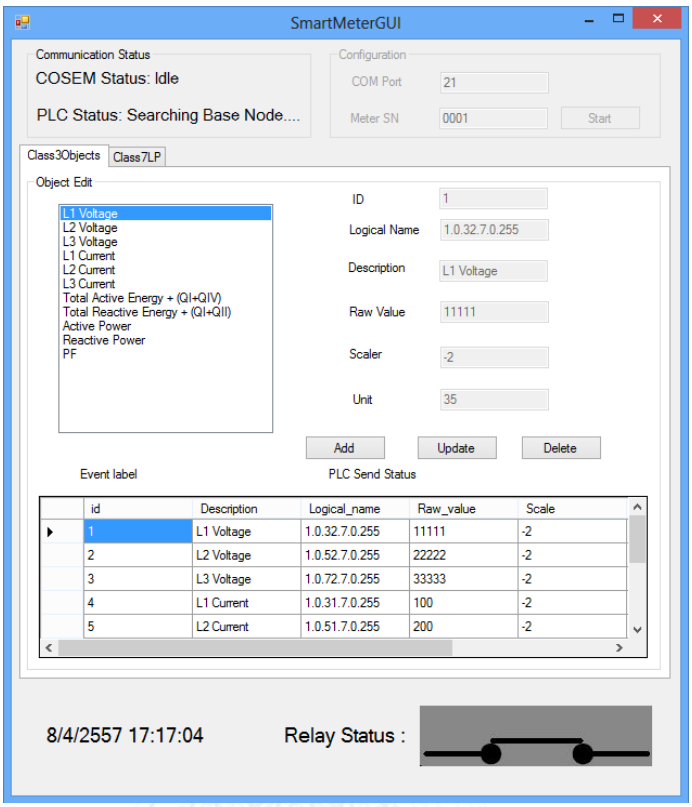
หลังจากนั้นเปิดโปรแกรมประยุกต์มาตรอัจฉริยะจำลองบนคอมพิวเตอร์ส่วนบุคคล จะพบหน้าต่างโปรแกรมประยุกต์รูปที่ 5-20



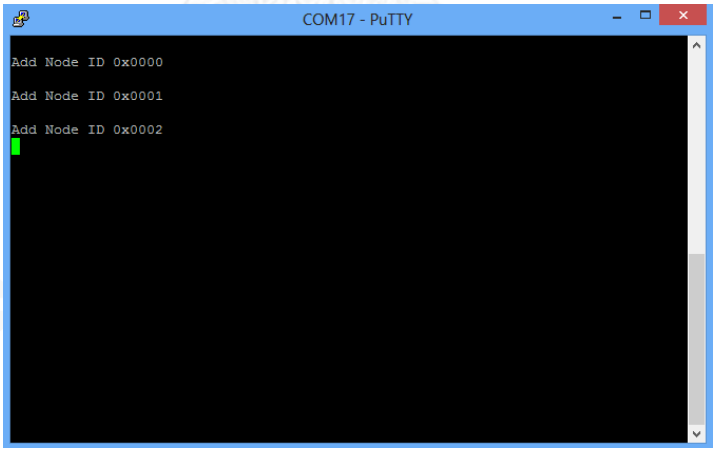
รูปที่ 5-20 หน้าต่างโปรแกรมประยุกต์มาตรอัจฉริยะจำลองบนคอมพิวเตอร์ส่วนบุคคล ในตอนเริ่มต้น

จากนั้นกรอก COM Port ที่คอมพิวเตอร์ส่วนบุคคลต่ออยู่โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลัง และเลขซีเรียลของมาตรอัจฉริยะจำลอง (Meter Serial Number) แล้วกดปุ่ม Start เพื่อเริ่มทำงาน หน้าต่างโปรแกรมประยุกต์จะเปลี่ยนไปดังรูปที่ 5-21 โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังที่ต่ออยู่กับมาตรอัจฉริยะจำลองนั้น เป็นแบบจุดต่อบริการ (Service Node) มันจะค้นหาจุดต่อฐาน (Base Node) ที่อยู่ใกล้ที่สุด นั้นคือ โมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังที่ต่ออยู่บอร์ด Beagleboard-xM ของอุปกรณ์เก็บรวบรวมข้อมูล ที่อยู่ในสายไฟฟ้าส่งกำลังเฟสเดียวกัน

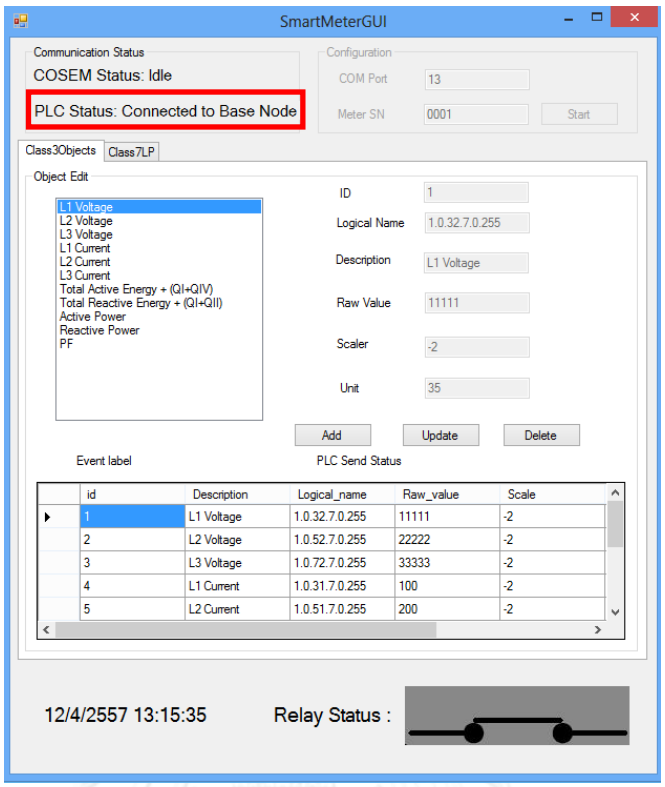
เมื่อจุดต่อบริการค้นหาจุดต่อฐานพบ และลงทะเบียนกับจุดต่อฐานสำเร็จแล้ว หน้าต่างโปรแกรมประยุกต์มาตรอัจฉริยะจำลองจะเป็นดังรูปที่ 5-23 และต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลจะแสดงผลดังรูปที่ 5-22 ผ่านทางโปรแกรมประยุกต์ putty ซึ่งเป็นรูปที่แสดงผลว่ามาตรอัจฉริยะจำลองลงทะเบียนกับต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลครบทั้งสามเครื่องแล้ว



รูปที่ 5-21 หน้าต่างโปรแกรมประยุกต์มาตรอัจฉริยะจำลองบนคอมพิวเตอร์ส่วนบุคคล เมื่อเริ่มทำงาน



รูปที่ 5-22 ต้นอุปกรณ์เก็บรวบรวมข้อมูลแสดงผลการลงทะเบียนของมาตรอัจฉริยะจำลองทั้งสาม



รูปที่ 5-23 สถานะของการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เมื่อมาตรอัจฉริยะจำลอง (จุดต่อบริการ) ลงทะเบียนกับต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล (จุดต่อฐาน) สำเร็จ

5.2. การทดสอบ และผลการทดสอบการเรียกใช้บริการของคำสั่ง GET

การทดสอบนี้เรียกใช้บริการถูกแบ่งออกเป็นสองส่วน คือ การสื่อสารโดยใช้บริการของคำสั่ง GET แบบปกติ และแบบบล็อกข้อมูลย่อย ความแตกต่างของสองแบบนี้ คือ

แบบแรกเมื่ออุปกรณ์เก็บรวบรวมข้อมูลได้รับคำสั่งจากระบบกลาง มันจะเรียกใช้บริการของคำสั่ง GET-Request-Normal ไปยังมาตรอัจฉริยะ และด้วยข้อมูลที่มาตรอัจฉริยะจะต้องตอบกลับมามีขนาดสั้นกว่าขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันเอาไว้ของทั้งสองอุปกรณ์ มันจะเรียกใช้บริการของคำสั่ง GET-Response-Normal ซึ่งเป็นบริการในกลุ่มที่ 1 ตามตารางที่ 3-1

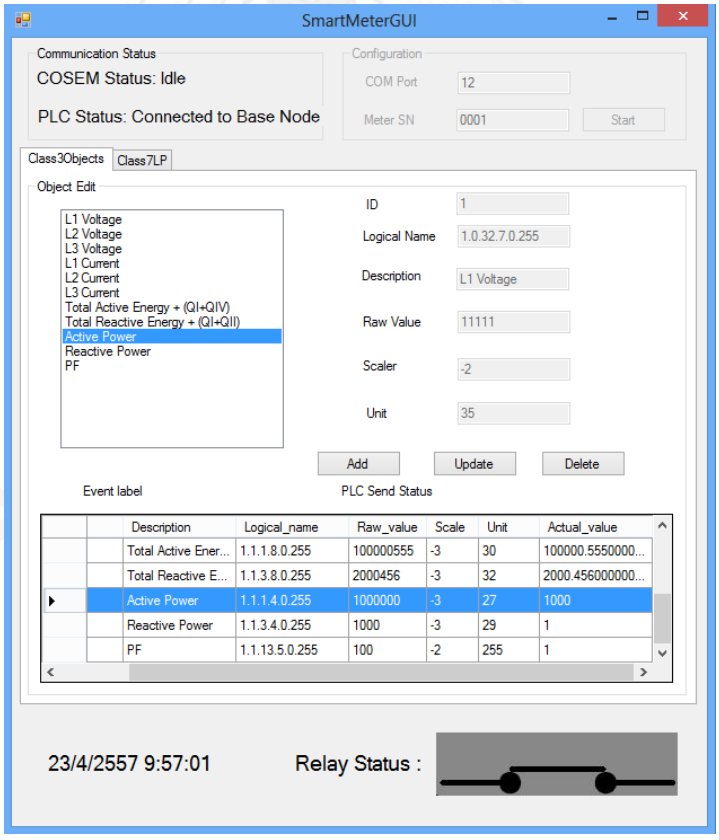
แบบที่สองเมื่ออุปกรณ์เก็บรวบรวมข้อมูลได้รับคำสั่งจากระบบกลาง มันจะเรียกใช้บริการของคำสั่ง GET-Request-Normal ไปยังมาตรอัจฉริยะ และด้วยข้อมูลที่มาตรอัจฉริยะจะต้องตอบกลับมามีขนาดยาวกว่าขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันเอาไว้ของทั้งสองอุปกรณ์ มันจะเรียกใช้บริการของคำสั่ง GET-Response-With-Datablock ซึ่งเป็นบริการในกลุ่มที่ 2 ตามตารางที่ 3-2 เมื่ออุปกรณ์เก็บรวบรวมข้อมูลได้รับข้อมูลในบล็อกย่อยแรก มันจะตอบกลับด้วย GET-Request-

Next ซึ่งเป็นบริการในกลุ่มที่ 3 ตามตารางที่ 3-3 ไปยังมาตรอัจฉริยะ เพื่อร้องขอบล็อกย่อยต่อไปเป็นแบบนี้ไปเรื่อยๆ จนครบทุกบล็อกย่อย

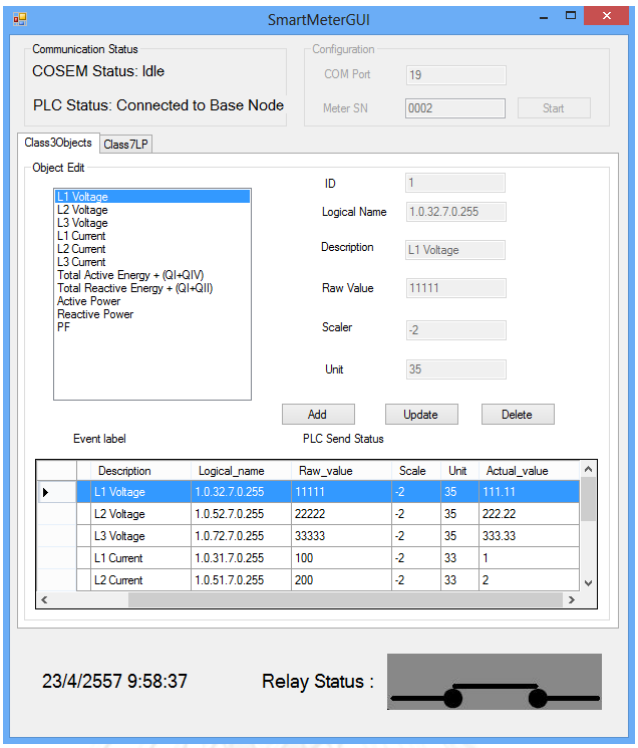
5.2.1. การทดสอบการเรียกใช้บริการของคำสั่ง GET แบบปกติ

การทดสอบนี้ผู้ใช้จะร้องขอค่าปริมาณทางไฟฟ้าที่มาตรอัจฉริยะทั้งสามเครื่องวัดได้ ซึ่งอยู่ในรูปของอ็อบเจกต์ที่บรรจุอยู่ในฐานข้อมูล เช่น ค่าแรงดันไฟฟ้า ค่าพลังงานไฟฟ้า ค่ากำลังไฟฟ้า เป็นต้น ซึ่งเป็นค่าที่จำนวนไบต์ข้อมูลไม่ยาวเกินกว่า ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลอง โดยใช้คำสั่ง GET

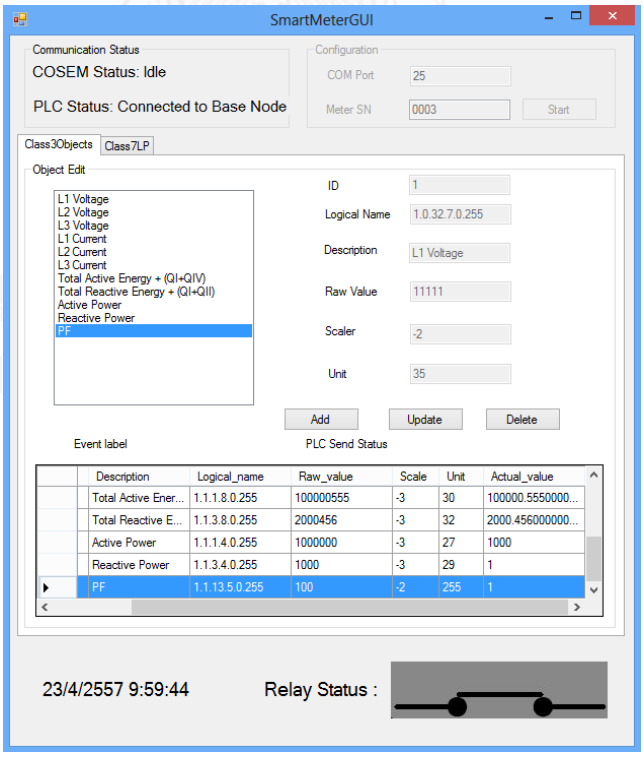
มาตรอัจฉริยะจำลองแต่ละเครื่องมีอ็อบเจกต์ต่างๆ ที่เหมือนกัน เนื่องจากการทดสอบนี้มาตรอัจฉริยะทุกเครื่องจะเชื่อมต่อไปยังฐานข้อมูลเดียวกัน แต่การทดสอบนี้จะสั่งระบบกลางจำลองให้เรียกอ่านมาตรอัจฉริยะแต่ละเครื่อง ด้วยค่าของอ็อบเจ็กต์ที่ต่างกัน ดังรูปที่ 5-24 ถึงรูปที่ 5-26



รูปที่ 5-24 อ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 1 ที่ต้องการอ่านค่า



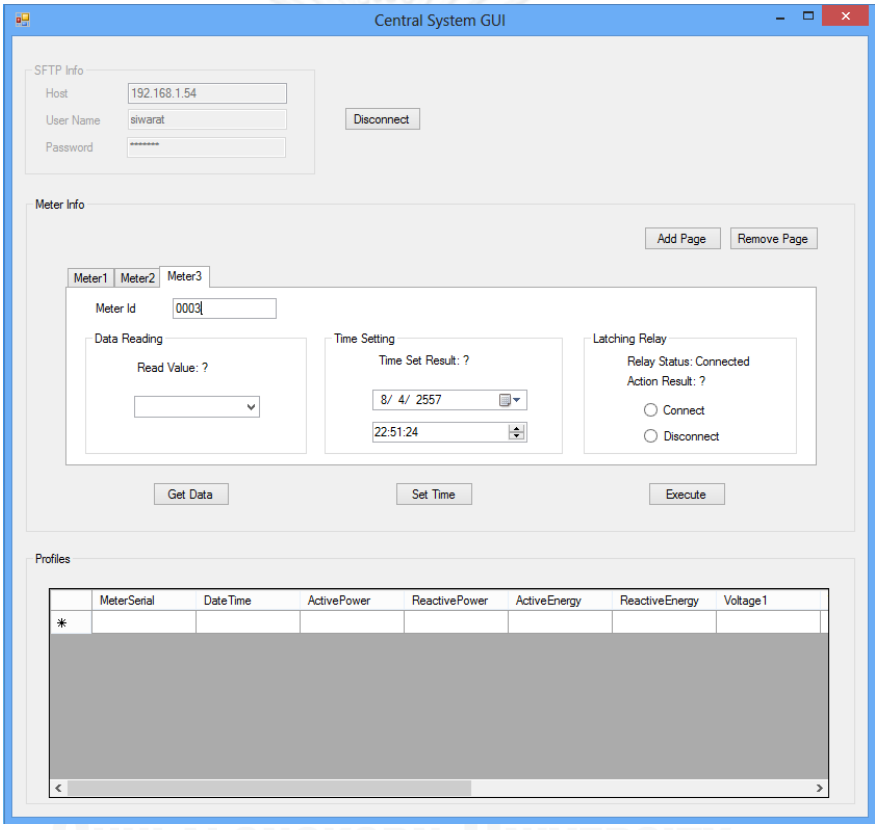
รูปที่ 5-25 อ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 2 ที่ต้องการอ่านค่า



รูปที่ 5-26 อ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 3 ที่ต้องการอ่านค่า

เริ่มต้นที่ระบบกลางจำลองจะร้องขอค่าของอ็อบเจกต์จากมาตรอัจฉริยะจำลองทั้ง 3 เครื่อง ผ่านอุปกรณ์เก็บรวบรวมข้อมูล ในการร้องขอเพียงครั้งเดียว

หลังจากตั้งค่าอุปกรณ์ทั้งสามตามหัวข้อ 5.1 แล้ว กดปุ่มเพิ่มหน้า (Add Page) เพื่อรองรับการอ่านมาตรอัจฉริยะทั้งสามเครื่องในการร้องขอเพียงครั้งเดียว หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลองจะเป็นดังรูปที่ 5-27 พร้อมทั้งใส่เลขซีเรียลของมาตรอัจฉริยะในช่องตัวระบุมาตร-อัจฉริยะ (Meter Id)

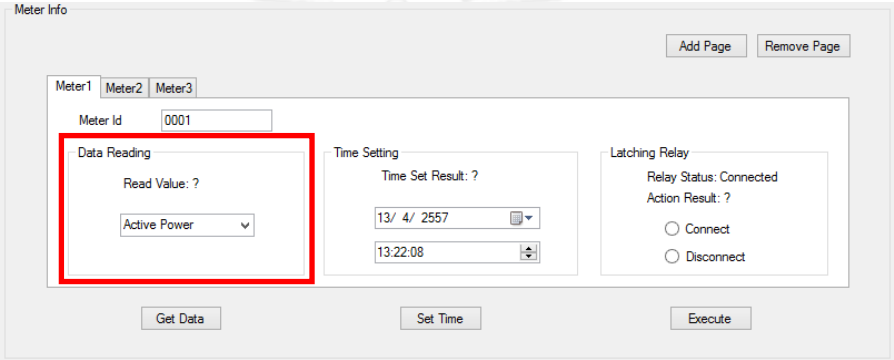


รูปที่ 5-27 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลอง เมื่อกดปุ่มเพิ่มหน้า

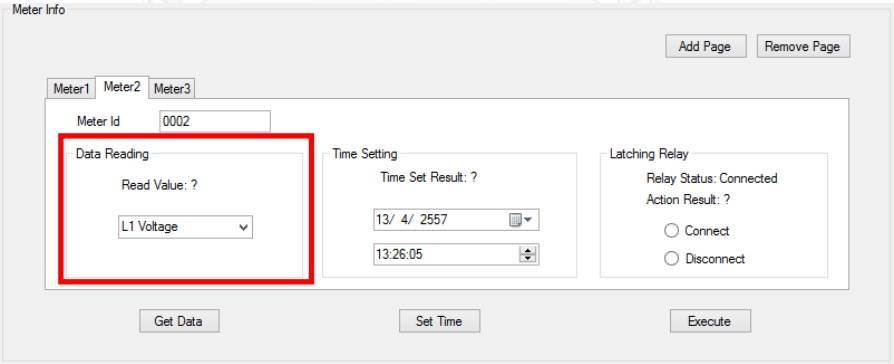
จากนั้นเลือกค่าของอ็อบเจกต์ภายในมาตรอัจฉริยะที่ต้องการอ่าน โดยกดเลือกที่ ComboBox ในกรอบที่มีชื่อว่า การอ่านข้อมูล (Data Reading) ของทั้งสามแท็บของมาตรอัจฉริยะจำลอง ในการทดสอบนี้ จะอ่านค่ากำลังไฟฟ้า (Active Power), ค่าแรงดันไฟฟ้าเฟสที่ 1 (L1 Voltage), และค่า แฟกเตอร์กำลัง (Power Factor) จากมาตรอัจฉริยะเครื่องที่ 1, 2, และ3 ดังรูปที่ 5-28 ถึง รูปที่ 5-30 ตามลำดับ เมื่อเลือกค่าที่ต้องการอ่านเสร็จแล้ว ให้กดปุ่ม รับข้อมูล (Get Data)

โปรแกรมประยุกต์ระบบกลางจำลองจะสร้างแฟ้มข้อมูล XML จากการข้อมูลที่ได้รับผ่านส่วนติดต่อผู้ใช้ หลังจากนั้นจะบีบอัดข้อมูล และส่งไปยังเซิร์ฟเวอร์โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ตที่อยู่บนบอร์ด Beagleboard-xM

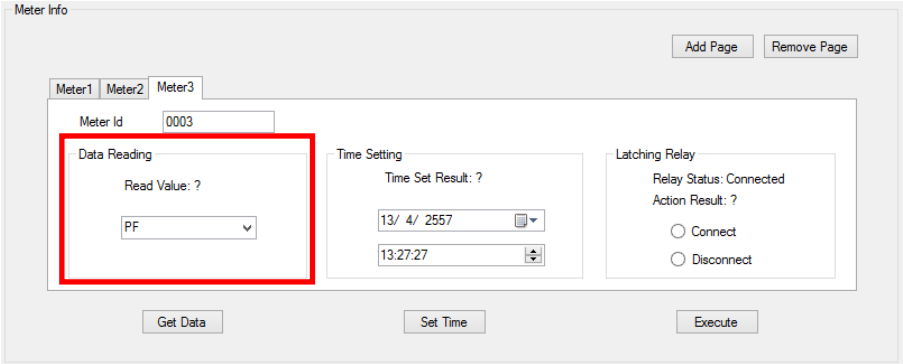
เมื่ออุปกรณ์เก็บรวบรวมข้อมูลตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ได้ตกลงกันไว้ (GetRequestNormal.xml.gz) มันจะคลายการบีบอัด และนำข้อมูลการร้องขอในแฟ้มข้อมูล XML เพื่อร้องขอการอ่านค่าอ็อบเจกต์ไปยังมาตรอัจฉริยะทั้งสามต่อไป โดยระหว่างนั้นมันจะแสดงผลผ่านโปรแกรมประยุกต์ putty ดังรูปที่ 5-31 กรอบสีแดง



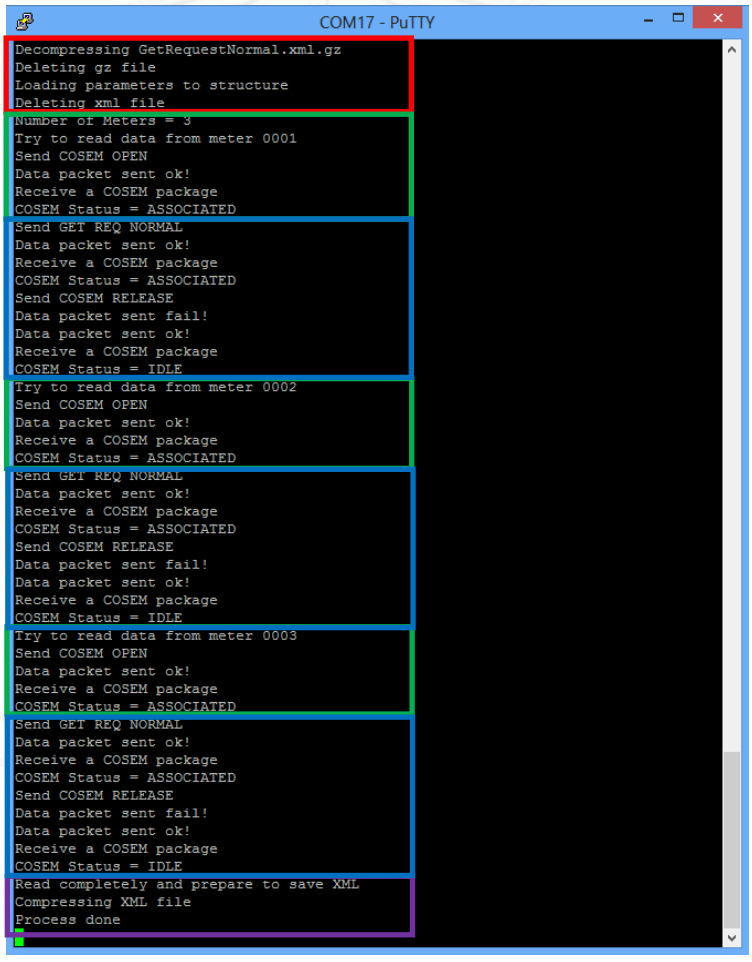
รูปที่ 5-28 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลอง เมื่อเลือกค่าที่ต้องการอ่านมาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-29 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลอง เมื่อเลือกค่าที่ต้องการอ่านมาตรอัจฉริยะเครื่องที่ 2



รูปที่ 5-30 หน้าต่างโปรแกรมประยุกต์ระบบกลางจำลอง เมื่อเลือกค่าที่ต้องการอ่านมาตรอัจฉริยะเครื่องที่ 3



รูปที่ 5-31 ผลตอบสนองขั้นตอนการทำงานของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลในการอ่านค่ามาตรอัจฉริยะจำลอง

ก่อนอุปกรณ์เก็บรวบรวมข้อมูลจะอ่านค่าอ็อบเจกต์ที่อยู่ภายในมาตรอัจฉริยะ มันต้องขอสร้างการเชื่อมต่อไปยังมาตรอัจฉริยะก่อน ดังรูปที่ 5-31 กรอบสีเขียว โดยในการทดสอบนี้จะใช้พารามิเตอร์ในกำหนดรูปแบบการสื่อสารชั้นประยุกต์ ซึ่งเป็นข้อตกลงของอุปกรณ์ทั้งสองก่อนการแลกเปลี่ยนข้อมูลกัน ดังนี้

dlms-version-number เท่ากับ 6

application-context-name เท่ากับ logical name

mechanism-name เท่ากับ lowest

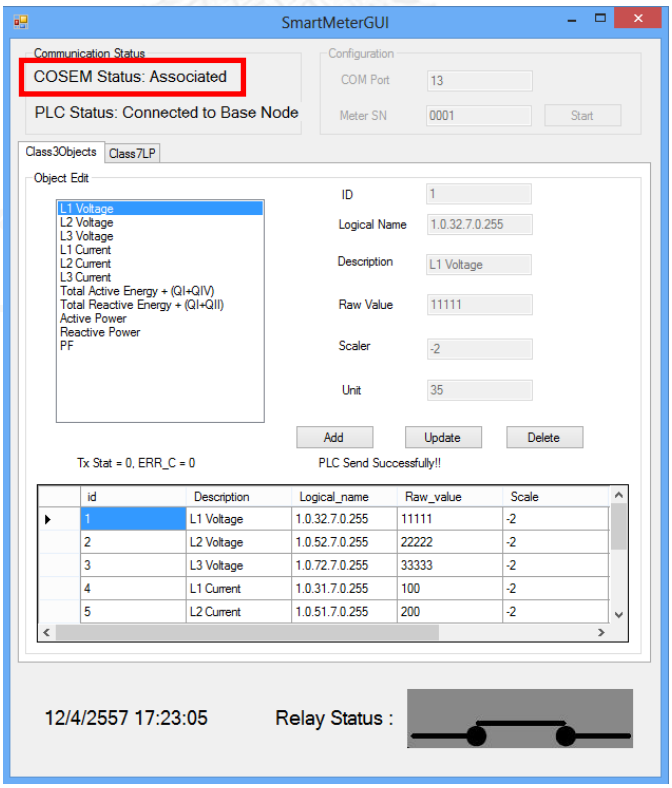
conformance เท่ากับ 0x00003E1F (ดูเพิ่มเติมได้ที่ [3] หัวข้อ 9.4.6.1)

*client-max-receive-pdu-size เท่ากับ 50 ไบต์

*server-max-receive-pdu-size เท่ากับ 50 ไบต์

* คือ ขนาดบัฟเฟอร์ในการรับส่งที่ได้ตกลงกันไว้ของอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลอง

เมื่อมันขอเชื่อมต่อเสร็จแล้ว สถานะโคเซม (COSEM State) ของมาตรอัจฉริยะจะเปลี่ยนเป็น เชื่อมต่อแล้ว (Associated) ดังรูปที่ 5-32



รูปที่ 5-32 สถานะโคเซมของมาตรอัจฉริยะเมื่อถูกเชื่อมต่อเข้ากับอุปกรณ์เก็บรวบรวมข้อมูล

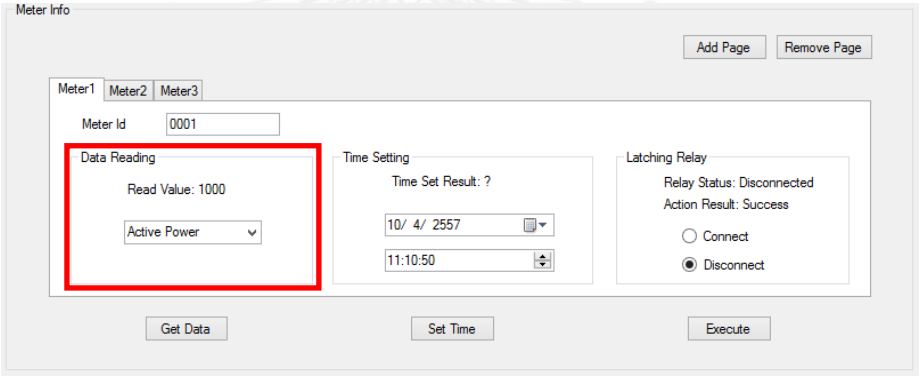
จากนั้นมันค่อยส่งคำร้องขอการอ่านค่าในอ็อบเจกต์ แล้วจึงขอยกเลิกการเชื่อมต่อไปยังมาตรอัจฉริยะ ดังรูปที่ 5-31 กรอบสีฟ้า สถานะโคเซมของมาตรอัจฉริยะจะกลับมาเป็นเดินเครื่องเปล่า (Idle) อีกครั้ง

เมื่ออุปกรณ์เก็บรวบรวมข้อมูลอ่านค่าอ็อบเจกต์ของมาตรอัจฉริยะตัวแรกเสร็จแล้ว มันจะอ่านของมาตรอัจฉริยะต่อไปที่อยู่ในรายชื่อคำร้องขอในแฟ้มข้อมูล XML ที่รับมาจากระบบกลางจำลอง

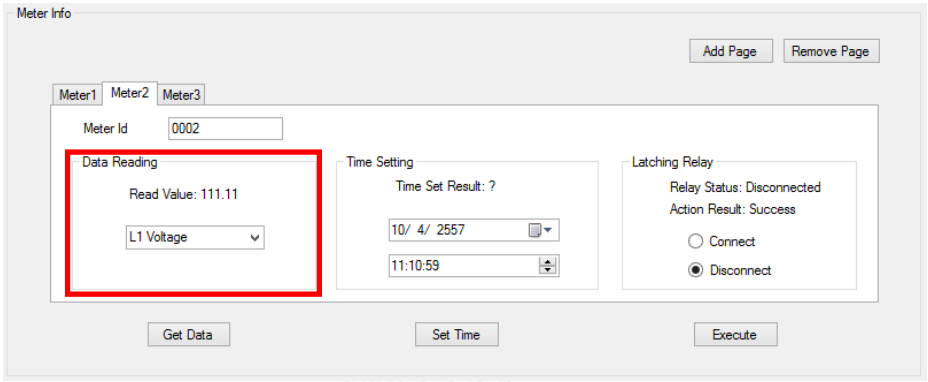
เมื่ออ่านครบทุกตัวมันจะสร้างแฟ้มข้อมูล XML ที่บรรจุผลตอบสนองจากมาตรอัจฉริยะที่มันอ่านมาได้ บีบอัดข้อมูล และใส่ไว้ในโฟลเดอร์ที่ได้ตกลงกันไว้ โดยจะแสดงผลออกทางโปรแกรมประยุกต์ putty ดังรูปที่ 5-31 กรอบสีม่วง

5.2.2. ผลการทดสอบการเรียกใช้บริการของคำสั่ง GET แบบปกติ

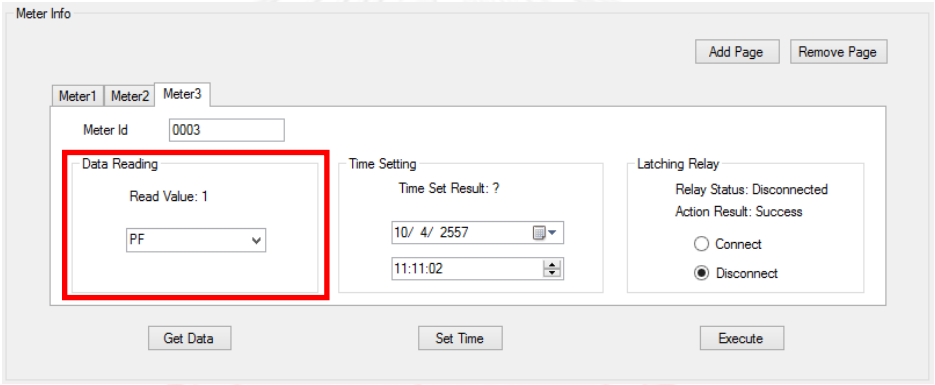
เมื่อระบบกลางจำลองตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อว่า GetResponseNomal.xml.gz มันจะบรรจุลง (Download) บนคอมพิวเตอร์ส่วนบุคคลที่มันกำลังทำงานอยู่ แล้วจึงคลายการบีบอัด แล้วนำผลตอบสนองในแฟ้มข้อมูล XML ไปถอดรหัส และแสดงผลได้ผลการทดสอบดังรูปที่ 5-33 ถึง รูปที่ 5-35



รูปที่ 5-33 ผลการอ่านค่าของอ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-34 ผลการอ่านค่าของอ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 2



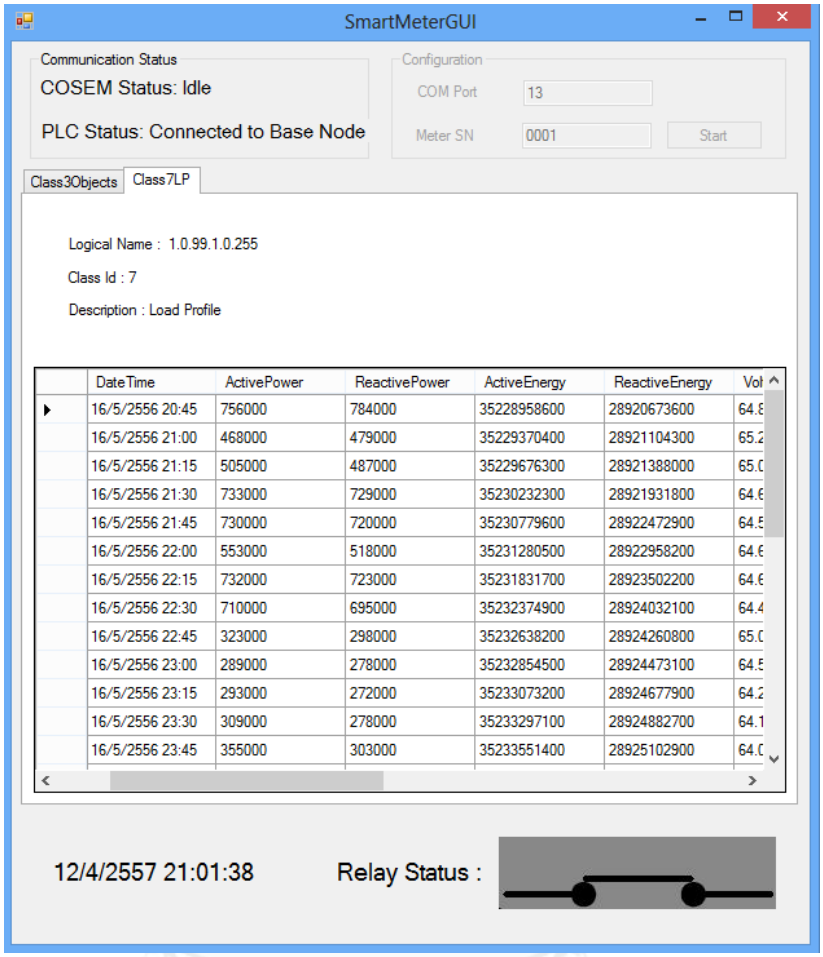
รูปที่ 5-35 ผลการอ่านค่าของอ็อบเจกต์ภายในมาตรอัจฉริยะเครื่องที่ 3

จากผลการอ่านค่าที่วัดได้จากมาตรอัจฉริยะจำลองทั้งสามเครื่อง ดังรูปที่ 5-33 ถึง รูปที่ 5-35 พบว่าได้ค่าตรงกับค่าของอ็อบเจกต์ที่มาตรอัจฉริยะจำลองเก็บอยู่ดังรูปที่ 5-24 ถึง รูปที่ 5-26 นั่นคือ ได้ค่ากำลังไฟฟ้าเท่ากับ 1000 วัตต์, ค่าแรงดันไฟฟ้าเฟสที่ 1 เท่ากับ 111.11 โวลต์, และค่าแฟกเตอร์กำลังเท่ากับ 1

5.2.3. การทดสอบการเรียกใช้บริการของคำสั่ง GET แบบบล็อกข้อมูลย่อย

การทดสอบนี้ผู้ใช้จะร้องขอโพรไฟล์ภาระ (Load Profile) ที่อยู่ภายในมาตร-อัจฉริยะจำลองทั้งสามเครื่อง (โดยปกติมาตรอัจฉริยะที่ใช้จริงจะวัด และบันทึกเก็บเอาไว้ทุกๆ สิบห้านาที) ซึ่งอยู่ในรูปของอ็อบเจกต์ที่บรรจุอยู่ในฐานข้อมูลที่โปรแกรมประยุกต์มาตรอัจฉริยะจำลองเชื่อมต่ออยู่ โดยมาตรอัจฉริยะแต่ละเครื่องจะมีโพรไฟล์ภาระที่เหมือนกัน เนื่องจากการทดสอบนี้มาตรอัจฉริยะทุกเครื่องจะเชื่อมต่อไปยังฐานข้อมูลเดียวกัน

เมื่อกดเลือกที่แท็บที่ชื่อว่าคลาสเลขที่เจ็ดโพรไฟล์ภาระ (Class7LP) ของโปรแกรมประยุกต์มาตรอัจฉริยะจำลอง จะพบหน้าต่างโพรไฟล์ภาระ ดังรูปที่ 5-36



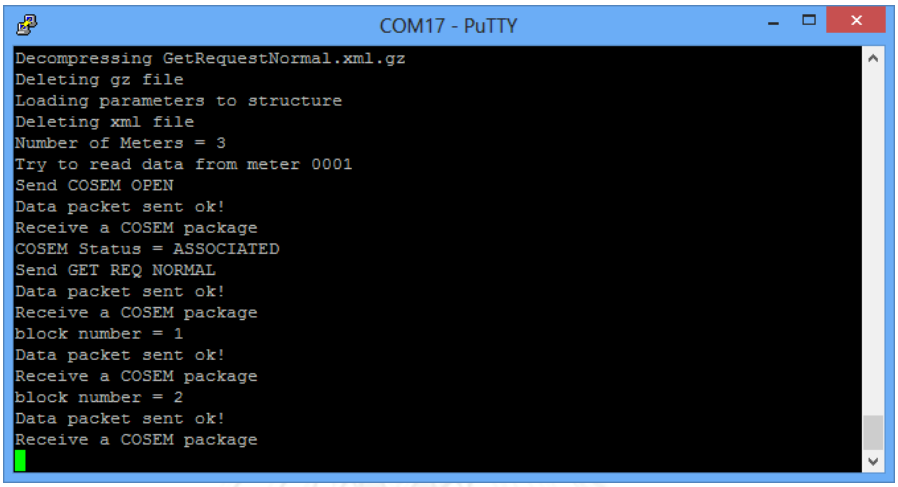
รูปที่ 5-36 โพรไฟล์ภาระที่เก็บอยู่ภายในมาตรอัจฉริยะ

เนื่องจากโพรไฟล์ภาระนั้นถูกเก็บสะสมมานาน และมีจำนวนไบต์มากกว่า ขนาดบัฟเฟอร์ในการรับส่ง ที่ได้ตกลงกันไว้ระหว่างอุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะ จึงทำให้ต้องส่งแบบบล็อกข้อมูลย่อยทีละบล็อก โดยใช้คำสั่ง GET

โดยขั้นตอนต่างๆ นั้น เกือบทั้งหมดจะเหมือนกันกับหัวข้อ 5.2.1 จะต่างกันตรงที่ขั้นตอนการรับส่งข้อมูล ในการทดสอบนี้จะส่งทีละบล็อกย่อย จนกว่าจะครบ โดยอุปกรณ์เก็บรวบรวมข้อมูลจะแสดงผลผ่านทางโปรแกรมประยุกต์ putty

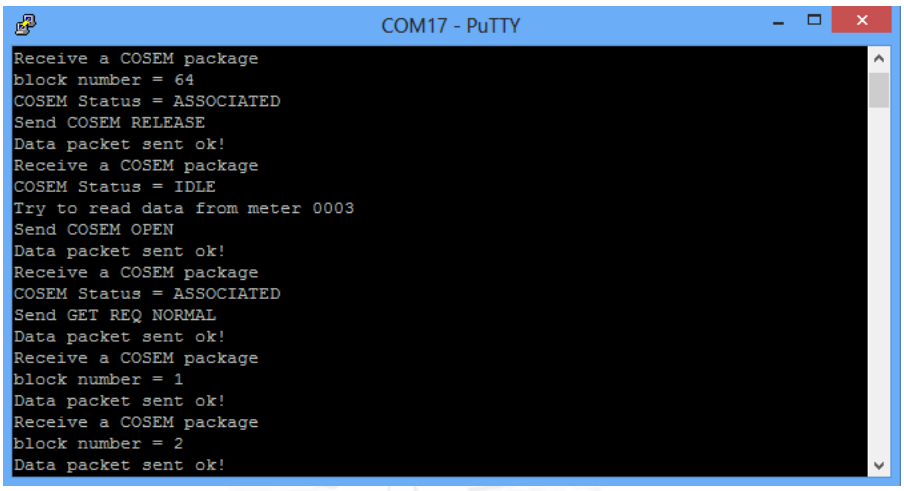
ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลรับคำสั่งจากระบบกลางจำลอง และเริ่มอ่านมาตรอัจฉริยะจำลองเครื่องที่ 1 ตั้งแต่บล็อกที่ 1 ดังรูปที่ 5-37 เมื่ออ่านครบถึงบล็อกสุดท้ายของมาตรอัจฉริยะจำลองเครื่องที่ 1 มันจะเริ่มอ่านมาตรอัจฉริยะจำลองเครื่องที่ 2 ดังรูปที่ 5-38 และเมื่ออ่าน

ครบถึงบล็อกสุดท้ายของมาตรอัจฉริยะจำลองเครื่องที่ 2 มันจะอ่านมาตรอัจฉริยะจำลองเครื่องสุดท้าย ดังรูปที่ 5-39 พออ่านครบทุกบล็อก มันจะสร้างแฟ้มข้อมูล XML ที่เก็บข้อมูลโพร์ไฟล์ภาระของมาตรอัจฉริยะทั้งสามเครื่อง จากนั้นมันจะบีบอัดแฟ้มข้อมูล ดังรูปที่ 5-40

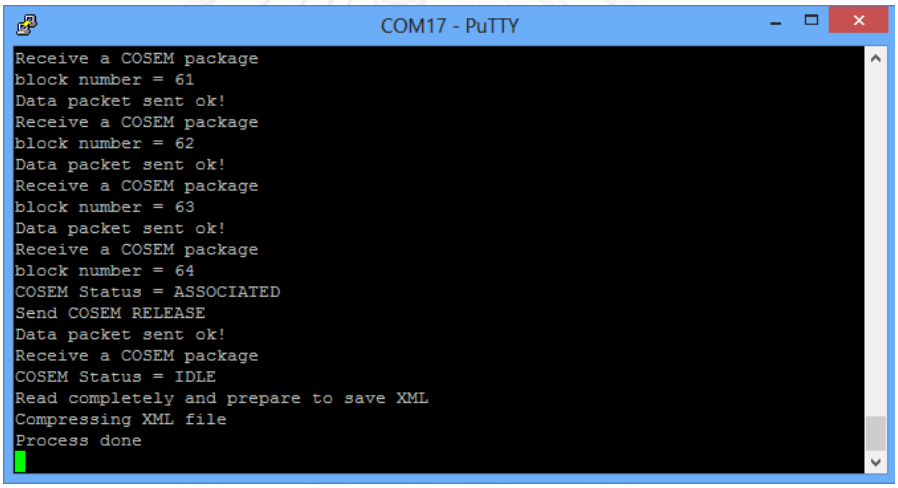


รูปที่ 5-37 ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลรับคำสั่ง และเริ่มอ่านมาตรอัจฉริยะจำลองเครื่องที่ 1

รูปที่ 5-38 ต้นแบบอุปกรณ์เก็บรวบรวมอ่านถึงบล็อกสุดท้ายของเครื่องที่ 1 และเริ่มอ่านเครื่องที่ 2



รูปที่ 5-39 ต้นแบบอุปกรณ์เก็บรวบรวมอ่านถึงบล็อกสุดท้ายของเครื่องที่ 2 และเริ่มอ่านเครื่องที่ 3

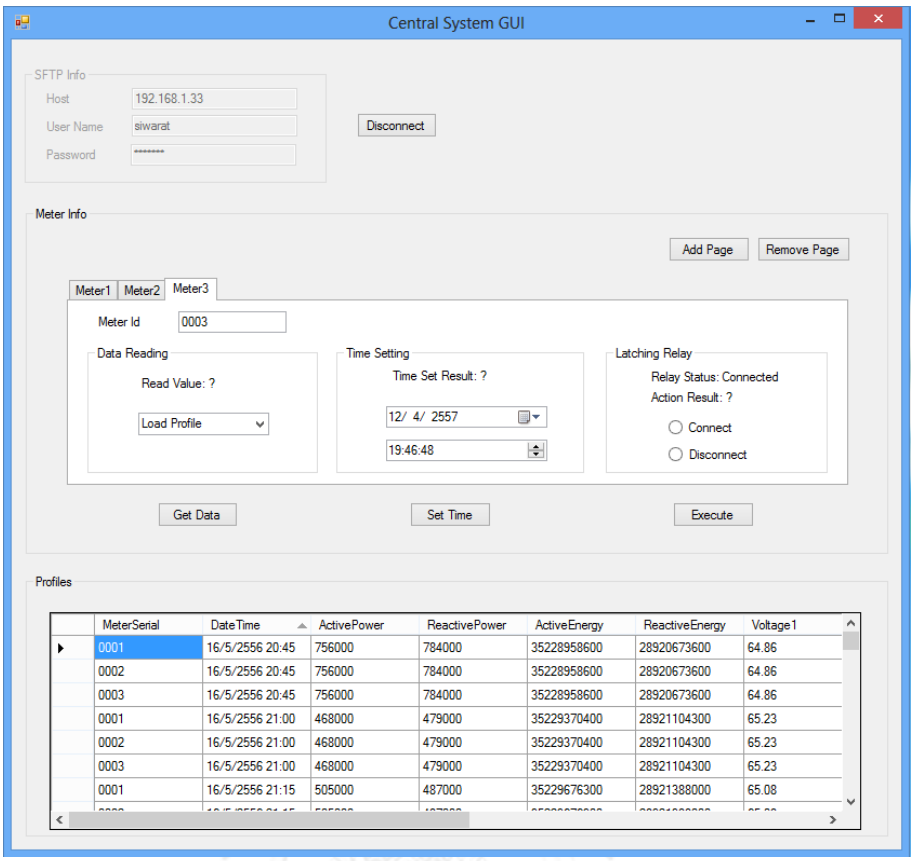


รูปที่ 5-40 ต้นแบบอุปกรณ์เก็บรวบรวมอ่านถึงบล็อกสุดท้ายของเครื่องที่ 3 สร้างแฟ้มข้อมูล XML และบีบอัด

เมื่ออุปกรณ์เก็บรวบรวมข้อมูลรับข้อมูลทุกบล็อกย่อยครบแล้ว มันจึงค่อยสร้างไฟล์ XML และบีบอัดข้อมูลโพรไฟล์ภาระของมาตรอัจฉริยะทั้งสามตัว แล้วจึงนำไปใส่ในโฟลเดอร์ที่ตกลงกันเอาไว้ ดังรูปที่ 5-40

5.2.4. ผลการทดสอบการเรียกใช้บริการของคำสั่ง GET แบบบล็อกข้อมูลย่อย

เมื่อระบบกลางจำลองตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อว่า GetResponseNomal.xml.gz มันจะบรรจุลง (Download) บนคอมพิวเตอร์ส่วนบุคคลที่มันกำลังทำงานอยู่ แล้วจึงคลายการบีบอัด แล้วนำผลตอบสนองในแฟ้มข้อมูล XML ไปถอดรหัส และแสดงผลได้ผลการทดสอบดังรูปที่ 5-41



รูปที่ 5-41 ผลการอ่านค่าโพรไฟล์ภาระของมาตรอัจฉริยะทั้งสาม

จากผลการอ่านโพรไฟล์ภาระจากมาตรอัจฉริยะทั้งสามเครื่อง ดังรูปที่ 5-41 พบว่าได้ค่าเท่ากับโพรไฟล์ภาระที่อยู่ภายในมาตรอัจฉริยะ ดังรูปที่ 5-36

5.3. การทดสอบ และผลการทดสอบการเรียกใช้บริการของคำสั่ง SET

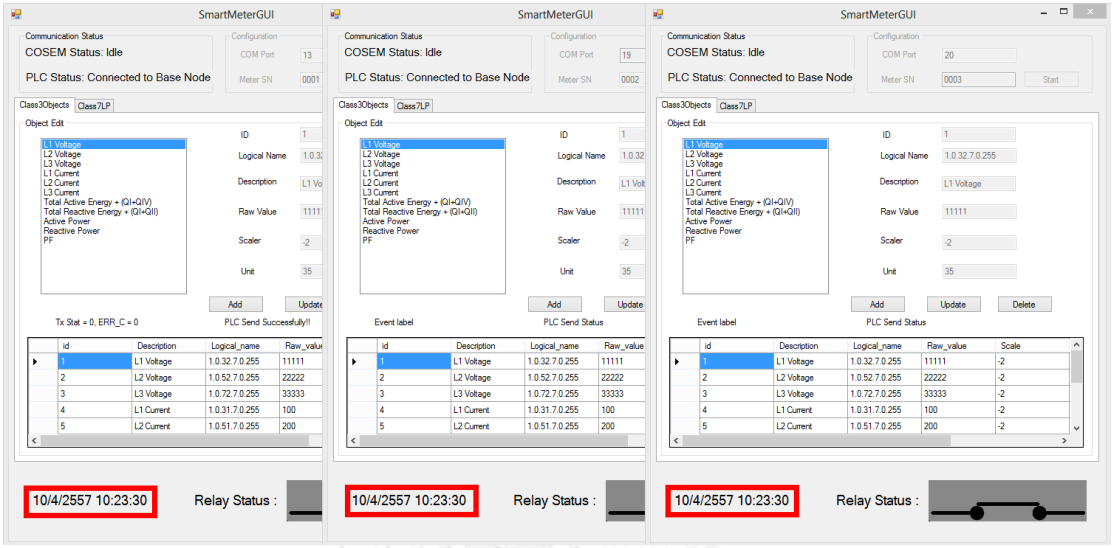
การทดสอบนี้ผู้ใช้จะร้องขอการตั้งค่าฐานเวลาในมาตรอัจฉริยะทั้งสามเครื่อง โดยใช้คำสั่ง SET แบบปกติ

5.3.1. การทดสอบการเรียกใช้บริการของคำสั่ง SET แบบปกติ

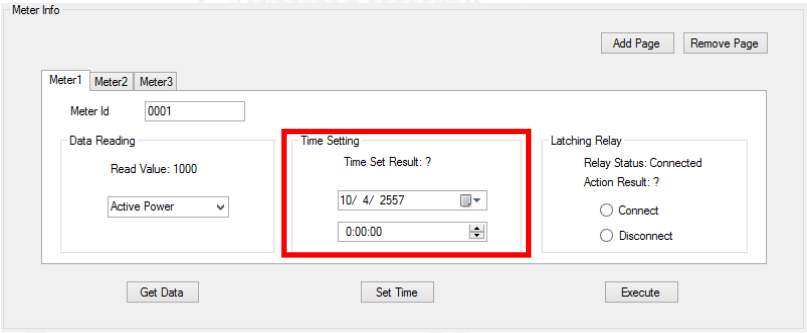
ต่อเนื่องจากหัวข้อ 5.1 และ 5.2 ในกรอบตั้งค่าเวลา (Time Setting) ทดสอบใส่ค่า ปี เดือน วันที่ ชั่วโมง นาที และวินาที ที่ต้องการตั้งค่าฐานเวลาให้กับมาตรอัจฉริยะ โดยค่าฐานเวลาปัจจุบันของมาตรอัจฉริยะทั้งสามเป็นดังรูปที่ 5-42

การทดสอบนี้จะตั้งค่าเวลาใหม่ให้กับมาตรอัจฉริยะเครื่องที่ 1 ดังรูปที่ 5-43 ส่วนมาตรอัจฉริยะเครื่องที่ 2 จะถูกตั้งวันที่ใหม่ ดังรูปที่ 5-44 และมาตรอัจฉริยะเครื่องที่ 3 จะถูกตั้งทั้ง

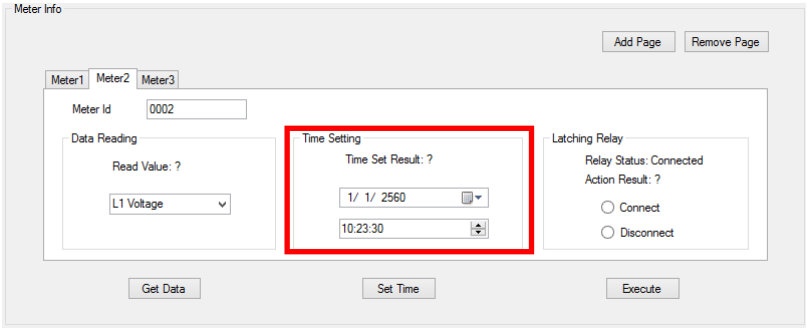
เวลา และวันที่ ดังรูปที่ 5-45 หลังจากนั้นกดปุ่ม ตั้งเวลา (Set Time) เพื่อส่งคำสั่งให้กับอุปกรณ์เก็บรวบรวมข้อมูล



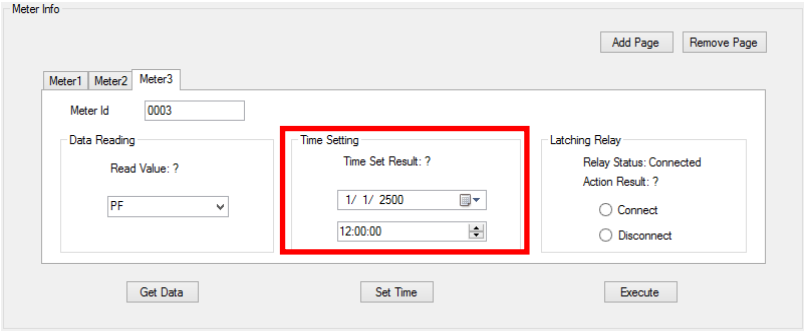
รูปที่ 5-42 ค่าฐานเวลาของมาตรอัจฉริยะทั้งสามเครื่อง ก่อนการตั้งค่าใหม่



รูปที่ 5-43 ค่าฐานเวลาที่ต้องการตั้งให้มาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-44 ค่าฐานเวลาที่ต้องการตั้งให้มาตรอัจฉริยะเครื่องที่ 2

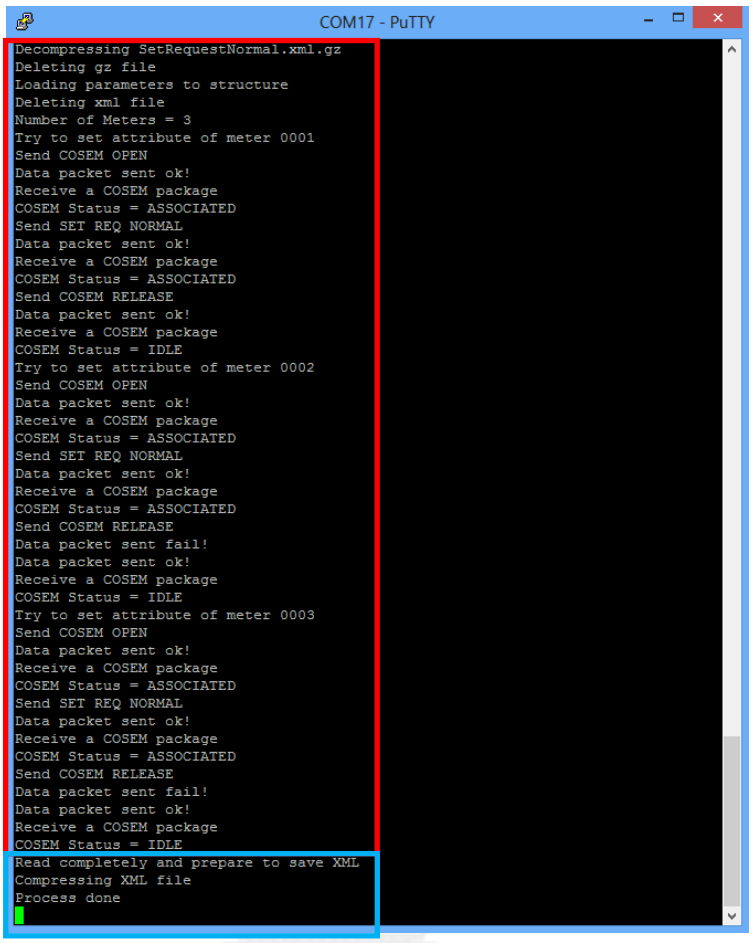


รูปที่ 5-45 ค่าฐานเวลาที่ต้องการตั้งให้มาตรอัจฉริยะเครื่องที่ 3

โปรแกรมประยุกต์ระบบกลางจำลองจะสร้างแฟ้มข้อมูล XML จากการข้อมูลที่ได้รับผ่านส่วนติดต่อผู้ใช้ หลังจากนั้นจะบีบอัดข้อมูล และส่งไปยังเซิร์ฟเวอร์โพรโตคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ตที่อยู่บนบอร์ด BeagleBoard-XM

เมื่ออุปกรณ์เก็บรวบรวมข้อมูลตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ได้ตกลงกันไว้ (SetRequestNormal.xml.gz) มันจะคลายการบีบอัด และนำข้อมูลการร้องขอในแฟ้มข้อมูล XML เพื่อร้องขอการตั้งฐานเวลาไปยังมาตรอัจฉริยะทั้งสามต่อไป โดยระหว่างนั้นมันจะแสดงผลผ่านโปรแกรมประยุกต์ putty ดังรูปที่ 5-46 กรอบสีแดง

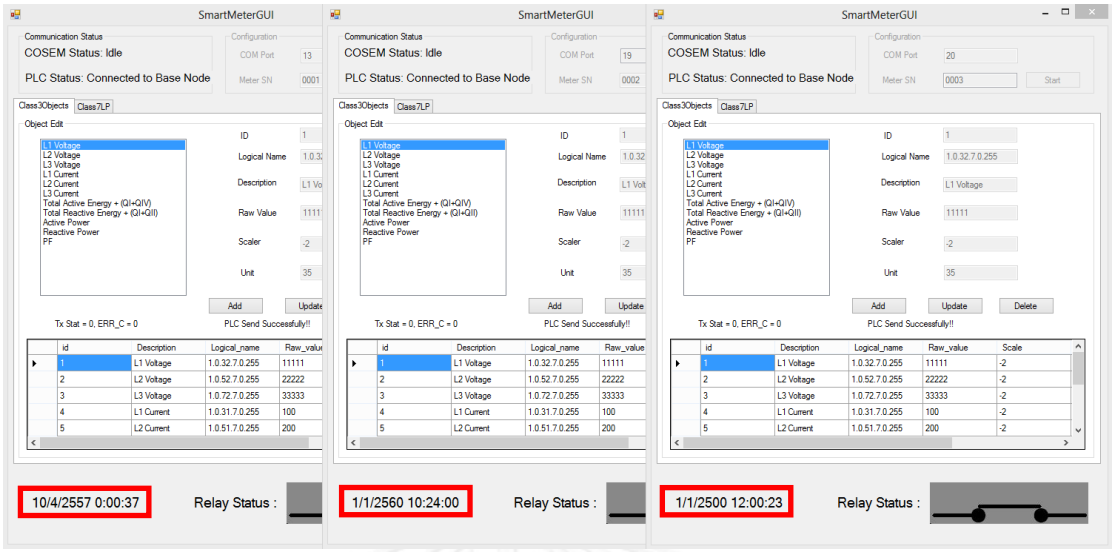
เมื่ออุปกรณ์เก็บรวบรวมข้อมูลตั้งค่าฐานเวลาของมาตรอัจฉริยะครบทุกตัว มันจะสร้างแฟ้มข้อมูล XML ที่บรรจุผลสรุปการตั้งค่าฐานเวลาของมาตรอัจฉริยะ จากนั้นบีบอัดข้อมูล และใส่ไว้ในโฟลเดอร์ที่ได้ตกลงกันไว้ โดยจะแสดงผลออกทางโปรแกรมประยุกต์ putty ดังรูปที่ 5-46 กรอบสีฟ้า



รูปที่ 5-46 ผลตอบสนองขั้นตอนการทำงานของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลในการตั้งค่าฐานเวลาของมาตรอัจฉริยะจำลอง

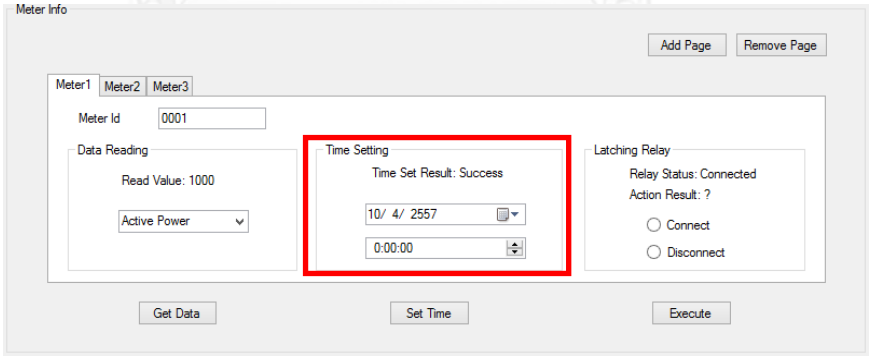
5.3.2. ผลการทดสอบการเรียกใช้บริการของคำสั่ง SET แบบปกติ

เมื่อมาตรอัจฉริยะทั้งสามได้รับคำร้องขอเพื่อตั้งค่า มันจะปรับฐานเวลาตัวเอง และได้ผลดังรูปที่ 5-47

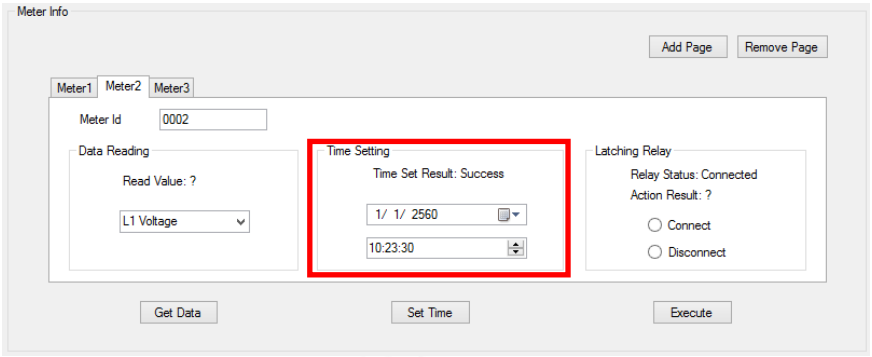


รูปที่ 5-47 ผลการตั้งค่าฐานเวลาของมาตรอัจฉริยะทั้งสามเครื่อง

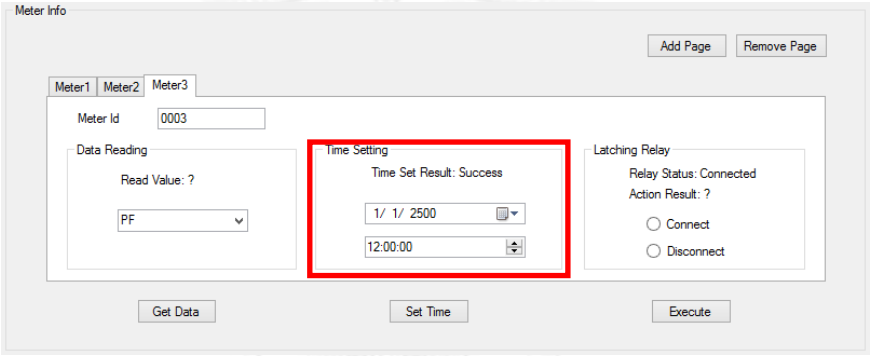
เมื่อระบบกลางจำลองตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ได้ตกลงกันไว้ (SetResponseNormal.xml.gz) บนเซิร์ฟเวอร์โพรโทคอลการโอนถ่ายแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต มันจะบรรจุลงคอมพิวเตอร์เครื่องที่มันทำงานอยู่ คลายการบีบอัด และนำผลสรุปการตั้งค่าฐานเวลาของมาตรอัจฉริยะทั้งสามเครื่องแสดงผลบนส่วนติดต่อผู้ใช้ ดังรูปที่ 5-48 ถึง รูปที่ 5-50



รูปที่ 5-48 ผลสรุปการตั้งค่าฐานเวลาของมาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-49 ผลสรุปการตั้งค่าฐานเวลาของมาตรอัจฉริยะเครื่องที่ 2



รูปที่ 5-50 ผลสรุปการตั้งค่าฐานเวลาของมาตรอัจฉริยะเครื่องที่ 3

จากผลการตั้งค่าฐานเวลาของมาตรอัจฉริยะทั้งสามเครื่อง ดังรูปที่ 5-47 พบว่าตรงกับค่าเวลาที่ระบบกลางจำลองสั่งไปยังมาตรอัจฉริยะแต่ละเครื่อง ดังรูปที่ 5-43 ถึง รูปที่ 5-45 (เวลาในรูปที่ 5-47 อาจคาดเคลื่อนไปบ้าง เนื่องจากเวลาในการส่งคำสั่งไปยังมาตรอัจฉริยะแต่ละเครื่องและเวลาในการจับภาพ) และผลตอบสนองที่กลับมายังระบบกลางจำลอง บ่งชี้ว่า การตั้งค่าฐานเวลาสำเร็จ ดังรูปที่ 5-48 ถึง รูปที่ 5-50

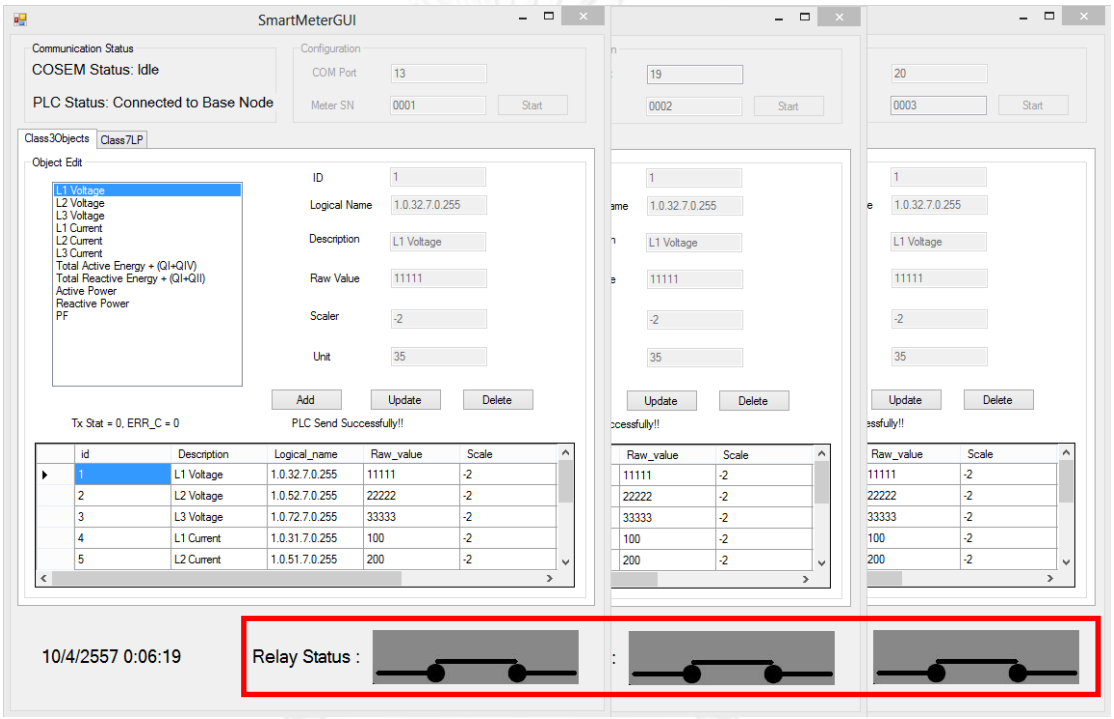
5.4. การทดสอบ และผลการทดสอบการเรียกใช้บริการของคำสั่ง ACTION

การทดสอบนี้ผู้ใช้จะร้องขอการเรียกใช้บริการภายในมาตรอัจฉริยะทั้งสามเครื่อง เพื่อทำการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ (Latching Relay) ภายในมาตรอัจฉริยะ โดยใช้คำสั่ง ACTION แบบ Normal

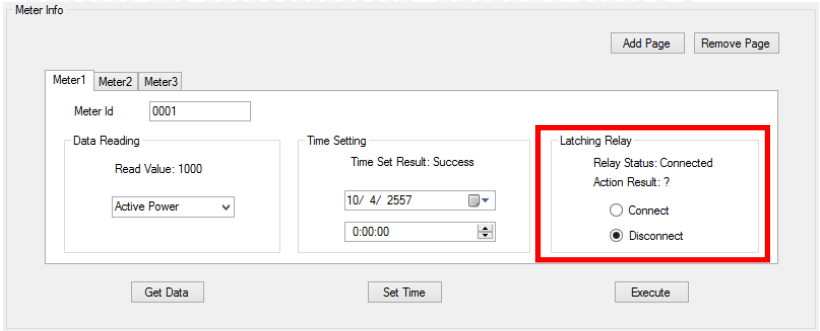
5.4.1. การทดสอบการเรียกใช้บริการของคำสั่ง ACTION แบบปกติ

ต่อเนื่องจากหัวข้อ 5.1 หัวข้อ 5.2 และหัวข้อ 5.3 การทดสอบนี้ต้องการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะทั้งสามที่มีสถานะเริ่มทำงานคือ ปิดวงจร (Connected) ดังรูปที่ 5-51

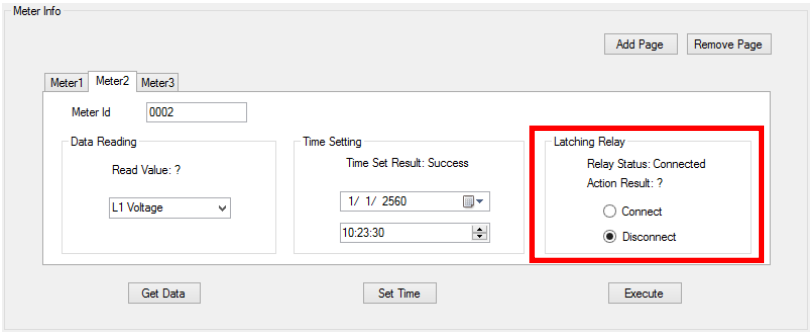
ในกรอบแลทชิ่งรีเลย์ เลือกเปลี่ยนสถานนะของแลทชิ่งรีเลย์เป็น เปิดวงจร (Disconnect) เพื่อสั่งให้แลทชิ่งรีเลย์ที่อยู่ในมาตรอัจฉริยะทุกตัวเปิดวงจร ผ่านส่วนติดต่อผู้ใช้ดังรูปที่ 5-52 ถึง รูปที่ 5-54 หลังจากนั้นกดปุ่ม กระทำการ (Execute) เพื่อส่งคำสั่งให้กับอุปกรณ์เก็บรวบรวมข้อมูล



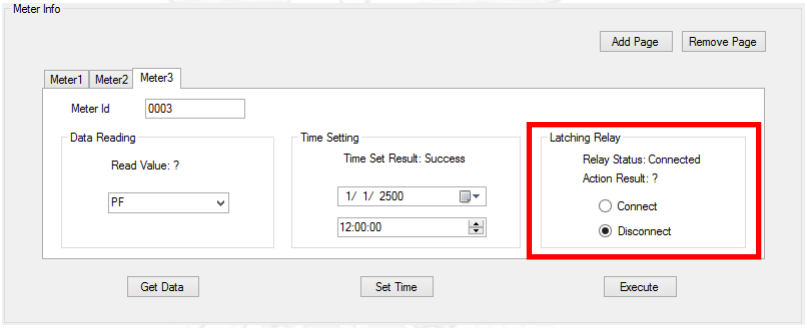
รูปที่ 5-51 สถานะรีเลย์ของมาตรอัจฉริยะทั้งสาม ก่อนการสั่งควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์



รูปที่ 5-52 ค่าควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ที่จะสั่งไปยังมาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-53 ค่าควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ที่จะสั่งไปยังมาตรอัจฉริยะเครื่องที่ 2

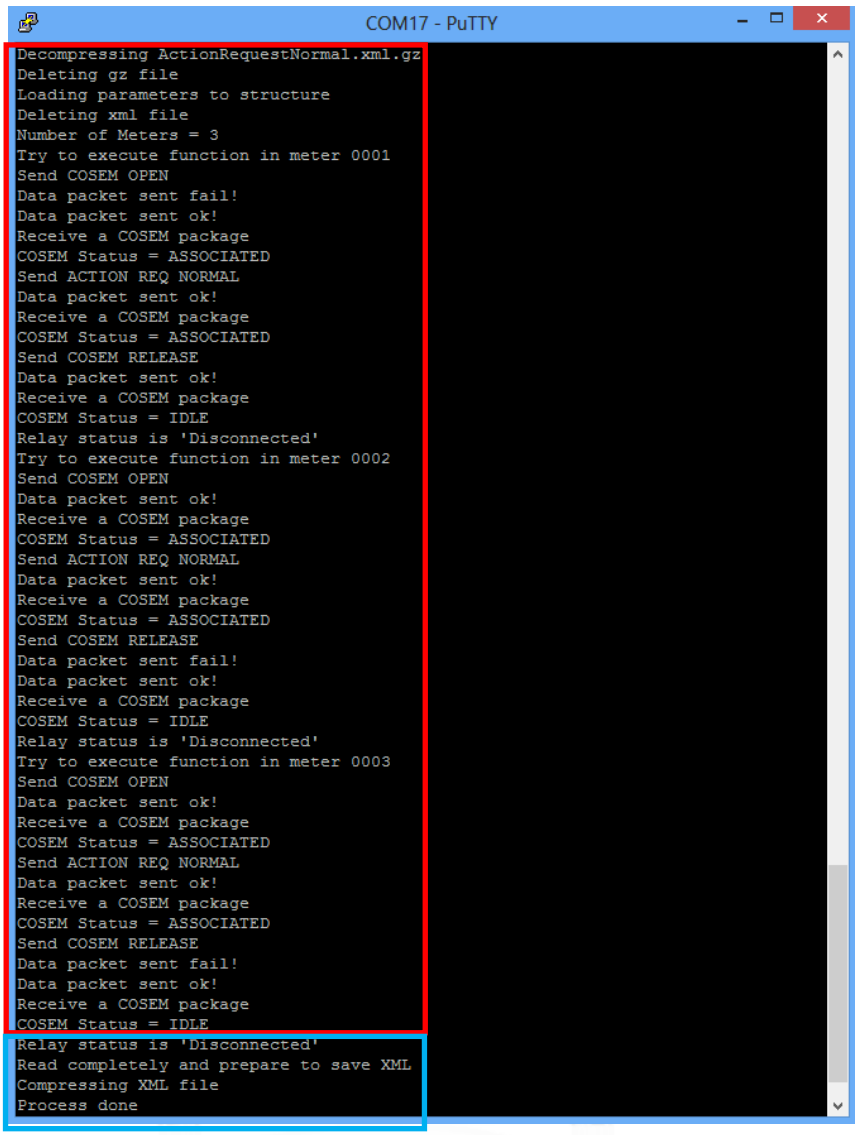


รูปที่ 5-54 ค่าควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ที่จะสั่งไปยังมาตรอัจฉริยะเครื่องที่ 3

โปรแกรมประยุกต์ระบบกลางจำลองจะสร้างแฟ้มข้อมูล XML จากการข้อมูลที่ได้รับผ่านส่วนติดต่อผู้ใช้ หลังจากนั้นจะบีบอัดข้อมูล และส่งไปยังเซิร์ฟเวอร์โพรโทคอลถ่ายโอนแฟ้มข้อมูลแบบปลอดภัยที่อยู่บนบอร์ด BeagleBoard-XM

เมื่ออุปกรณ์เก็บรวบรวมข้อมูลตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ได้ตกลงกันไว้ (ActionRequestNormal.xml.gz) มันจะคลายการบีบอัด และนำข้อมูลการร้องขอในแฟ้มข้อมูล XML เพื่อร้องขอการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ไปยังมาตรอัจฉริยะทั้งสามต่อไป โดยระหว่างนั้นมันจะแสดงผลผ่านโปรแกรมประยุกต์ putty ดังรูปที่ 5-55 กรอบสีแดง

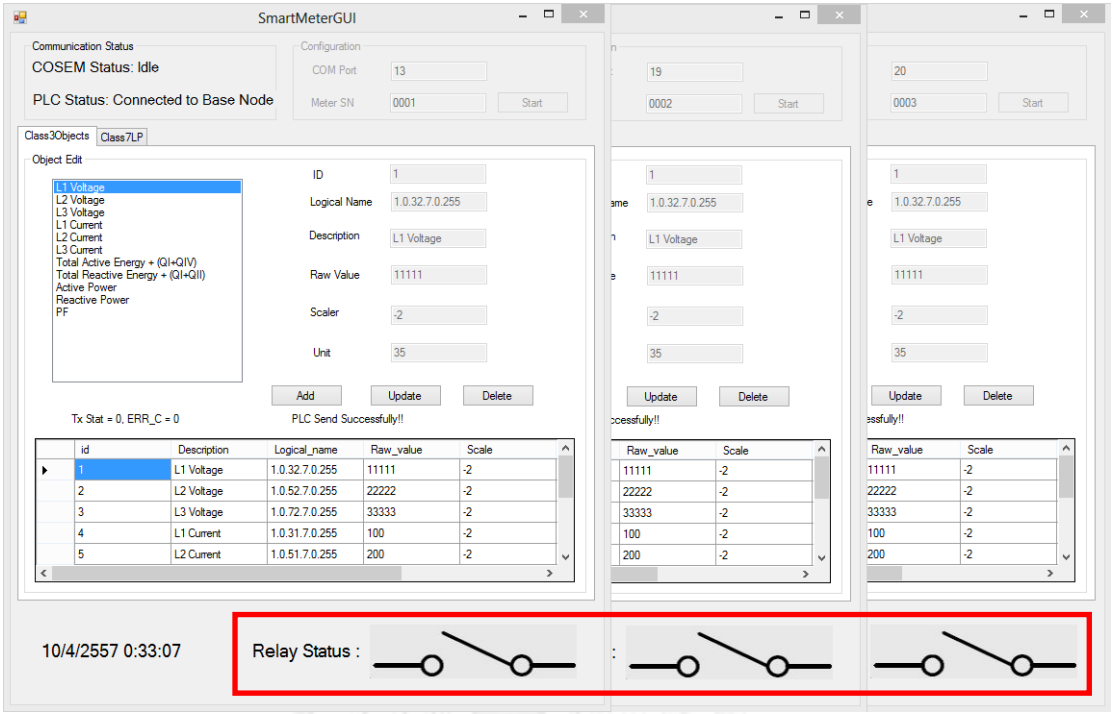
เมื่ออุปกรณ์เก็บรวบรวมข้อมูลควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะครบทุกตัว มันจะสร้างแฟ้มข้อมูล XML ที่บรรจุผลสรุปการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะ จากนั้นบีบอัดข้อมูล และใส่ไว้ในโฟลเดอร์ที่ได้ตกลงกันไว้ โดยจะแสดงผลออกทางโปรแกรมประยุกต์ putty ดังรูปที่ 5-55 กรอบสีฟ้า



รูปที่ 5-55 ผลตอบสนองขั้นตอนการทำงานของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลในการควบคุมการเปิด-ปิดของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะจำลอง

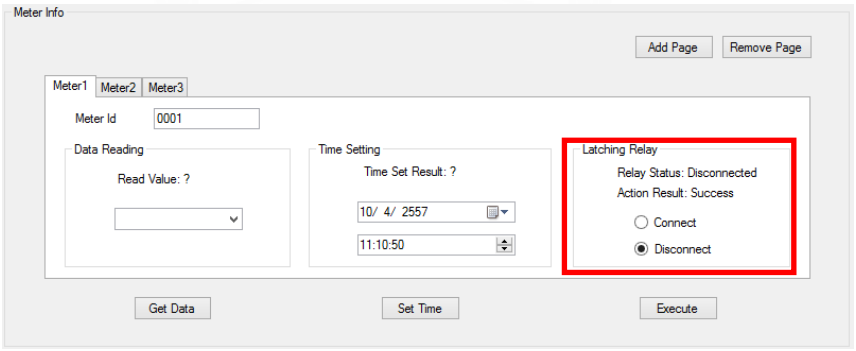
5.4.2. ผลการทดสอบการเรียกใช้บริการของคำสั่ง ACTION แบบปกติ

เมื่อมาตรอัจฉริยะทั้งสามได้รับคำร้องขอเพื่อควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ มันจะเปลี่ยนสถานะของแลทชิ่งรีเลย์ตามคำร้องขอ และได้ผลดังรูปที่ 5-56

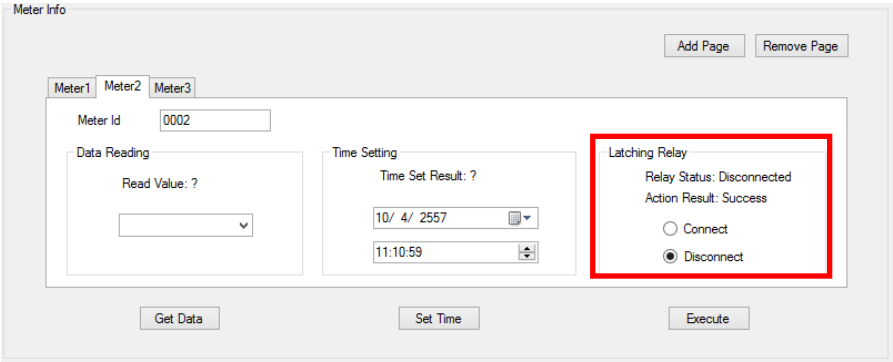


รูปที่ 5-56 ผลการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะทั้งสาม

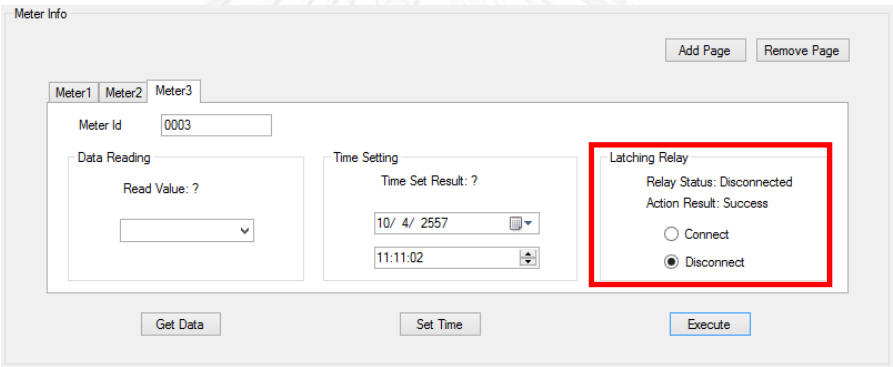
เมื่อระบบกลางจำลองตรวจพบแฟ้มข้อมูลที่ถูกบีบอัดที่มีชื่อตามที่ได้ตกลงกันไว้ (ActionResponseNormal.xml.gz) บนเซิร์ฟเวอร์โพรโทคอลการโอนถ่ายแฟ้มข้อมูลแบบปลอดภัยผ่านอินเทอร์เน็ต มันจะบรรจุลงคอมพิวเตอร์เครื่องที่มันทำงานอยู่ คลายการบีบอัด และนำผลสรุปการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะทั้งสามเครื่องแสดงผลบนส่วนติดต่อผู้ใช้ ดังรูปที่ 5-57 ถึง รูปที่ 5-59



รูปที่ 5-57 สถานะปัจจุบันของแลทชิ่งรีเลย์ และผลสรุปการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะเครื่องที่ 1



รูปที่ 5-58 สถานะปัจจุบันของแลทชิ่งรีเลย์ และผลสรุปการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะเครื่องที่ 2



รูปที่ 5-59 สถานะปัจจุบันของแลทชิ่งรีเลย์ และผลสรุปการควบคุมการเปิด-ปิดวงจรของแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะเครื่องที่ 3

จากผลการควบคุมการเปิด-ปิดแลทชิ่งรีเลย์ของมาตรอัจฉริยะทั้งสามเครื่อง ดังรูปที่ 5-56 พบว่าแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะแต่ละเครื่อง ทำตามคำสั่งดังรูปที่ 5-52 ถึง รูปที่ 5-54 ได้อย่างถูกต้อง คือ แลทชิ่งรีเลย์ภายในมาตรอัจฉริยะแต่ละเครื่องอยู่ในสถานะเปิดวงจร และผลตอบสนองที่กลับมายังระบบกลางจำลอง บ่งชี้ว่า การควบคุมการเปิด-ปิดแลทชิ่งรีเลย์สำเสร็จ ดังรูปที่ 5-57 ถึง รูปที่ 5-59

5.5. การทดสอบ และผลการทดสอบการอ่านค่าโพรไฟล์ภาระ ข้อมูลสำหรับการเก็บเงินค่าบริการ และบันทึกเหตุการณ์จากมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรม และใช้มาตรฐานดีแอล-เอ็มเอส/โคเซมในการสื่อสาร

การทดสอบนี้เป็นการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม และใช้โพรไฟล์สื่อสารแบบสามชั้นแบบเชื่อมต่อก่อนเอชดีแอลซี (The 3-layer, connection-oriented, HDLC-

based communication profile) [3, 5] เพื่อแลกเปลี่ยนข้อมูลกับมาตรอัจฉริยะที่อยู่ห่างไกลออกไปได้ ตราบเท่าที่เครือข่ายโทรศัพท์เคลื่อนที่จีเอสเอ็มครอบคลุมไปถึงจุดที่ทั้งโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม และมาตรอัจฉริยะตั้งอยู่ [21]

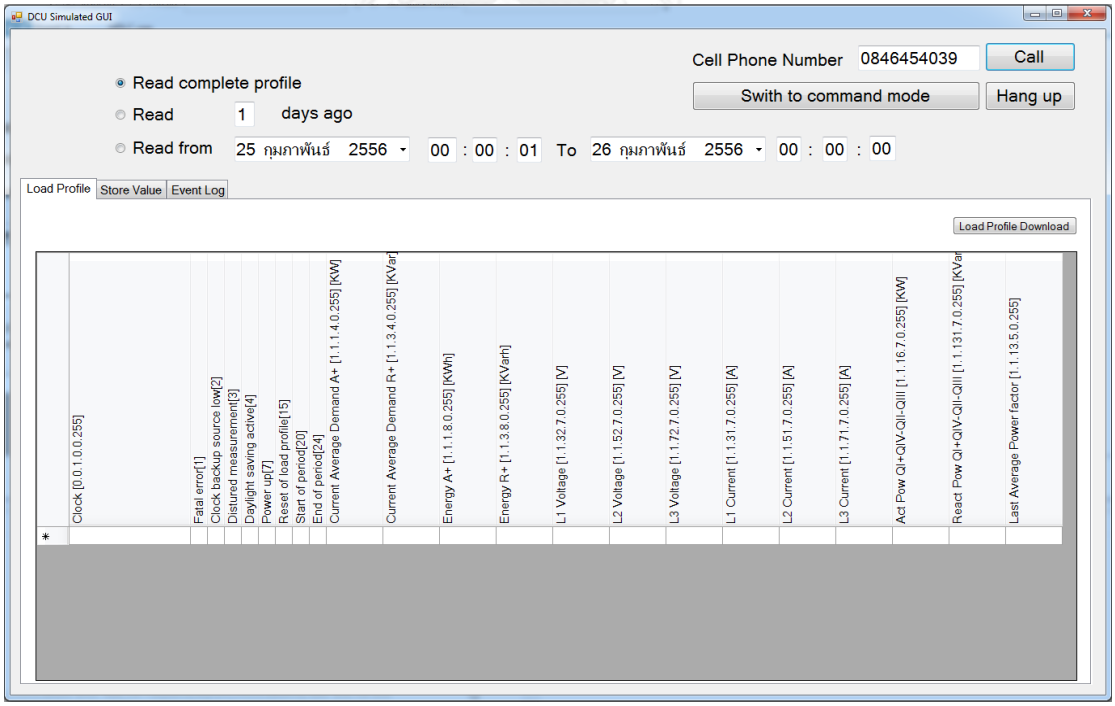
5.5.1. **การทดสอบการอ่านค่าโพรไฟล์ภาระ ข้อมูลสำหรับการเก็บเงินค่าบริการ และบันทึกเหตุการณ์**

การทดสอบนี้ใช้อุปกรณ์สองประกอบด้วย มาตรอัจฉริยะที่ได้มาตรฐานดีแอลเอ็ม-เอส/โคเซม ที่ใช้การอ้างอิงแบบชื่อสั้น (Short Name) ดังรูปที่ 5-60 และโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็มที่เชื่อมต่อกับคอมพิวเตอร์ส่วนบุคคลผ่านพอร์ตยูเอสบี ดังรูปที่ 5-61

รูปที่ 5-60 มาตรอัจฉริยะที่ใช้งานจริงในอุตสาหกรรม ที่ได้มาตรฐานดีแอลเอ็มเอส และโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม

รูปที่ 5-61 โมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็มที่เชื่อมต่อกับคอมพิวเตอร์ส่วนบุคคลผ่านพอร์ตยูเอสบี (USB Port)

นอกจากฮาร์ดแวร์ทั้งสอง ยังมีซอร์ฟแวร์อ่านโพรไฟล์จากมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรม ที่พัฒนาขึ้นโดยใช้คลังโปรแกรมดีแอลเอ็มเอส/โคเซม ดังรูปที่ 5-62



รูปที่ 5-62 หน้าต่างโปรแกรมประยุกต์ที่ใช้คลังโปรแกรมดีแอลเอ็มเอส/โคเซมที่พัฒนาขึ้น เพื่ออ่านค่าโพรไฟล์ต่างๆ จากมาตรอัจฉริยะที่ใช้งานจริง

เมื่อปุ่มโทร (Call) บนหน้าต่างของโปรแกรมประยุกต์ ดังรูปที่ 5-62 มันจะร้องขอเพื่อสร้างช่องทางเชื่อมต่อของการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม ไปยังโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม ผ่านทางพอร์ตยูเอสบีของคอมพิวเตอร์ส่วนบุคคล หลังจากนั้นโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็มจะสร้างการเชื่อมต่อให้ โดยโทรไปหาโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็มที่อยู่ในมาตรอัจฉริยะ เมื่อโมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม ที่มาตรอัจฉริยะตอบรับการเชื่อมต่อ โมเด็มการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม ฝั่งคอมพิวเตอร์ส่วนบุคคลจะส่งข้อความว่า "..CONNECT 9600.." กลับไปแจ้งโปรแกรมประยุกต์ ดังรูปที่ 5-63 ในกรอบสีแดง

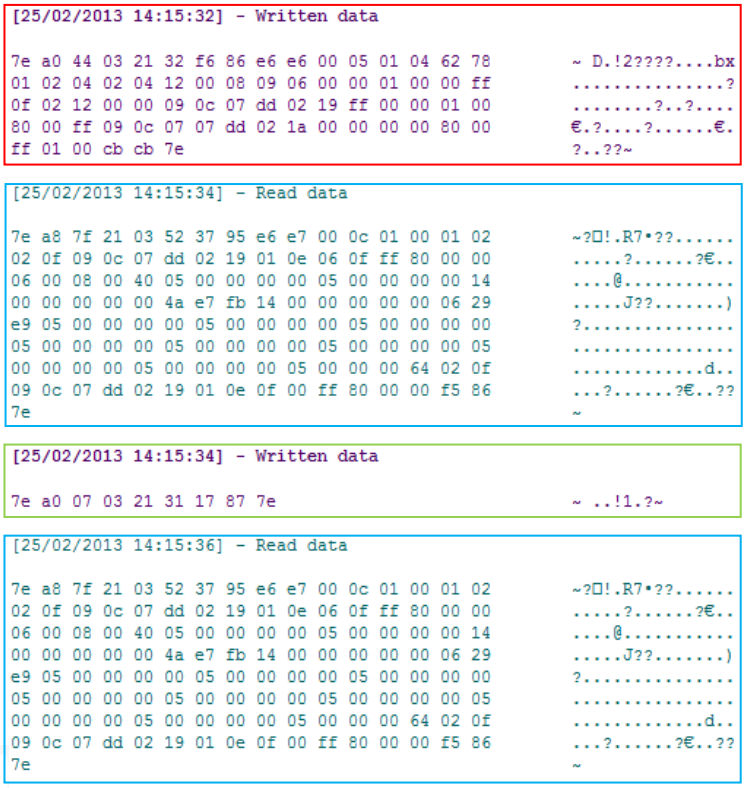
เมื่อช่องทางสื่อสารของการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็มเชื่อมต่อแล้ว โปรแกรมประยุกต์จะร้องขอเพื่อสร้างการเชื่อมต่อในชั้นเชื่อมต่อข้อมูล (Data Link Layer) ซึ่งจะใช้โพรโทคอลแบบเอชดีแอลซี กับมาตรอัจฉริยะ และเมื่อมาตรอัจฉริยะตอบตกลง ทำให้การเชื่อมต่อในชั้นเชื่อมต่อข้อมูลถูกสร้างขึ้น ดังรูปที่ 5-63 ในกรอบสีฟ้า

โปรแกรมประยุกต์จะเรียกใช้คำสั่ง COSEM_OPEN_Req จากคลังโปรแกรมดีแอล-เอ็มเอส/โคเซม เพื่อร้องขอการสร้างช่องทางการเชื่อมต่อในชั้นประยุกต์ พร้อมทั้งส่งค่าพารามิเตอร์ที่เป็นข้อกำหนดต่างๆ ในการสื่อสารเพื่อต่อรองกับมาตรอัจฉริยะ และเมื่อมาตรอัจฉริยะตอบตกลง การเชื่อมต่อชั้นประยุกต์ จะถูกสร้างขึ้น ดังรูปที่ 5-63 กรอบสีเขียว

รูปที่ 5-63 การส่งข้อมูลเพื่อร้องขอ และตอบรับการสร้างช่องทางการเชื่อมต่อในชั้นการเชื่อมต่อต่างๆ ตามมาตรฐานดีแอลเอ็มเอส/โคเซม

เมื่อช่องทางการสื่อสารได้ถูกสร้างขึ้นแล้ว เลือกตัวเลือกอ่านหนึ่งวันล่าสุด (Read 1 days ago) และกดปุ่มบรรจุลงโพรไฟล์ภาระ (Load Profile Download) ที่หน้าต่างโปรแกรมประยุกต์ดังรูปที่ 5-62 โปรแกรมประยุกต์จะเรียกใช้คำสั่ง GET_Req_Normal จากคลังโปรแกรมดีแอลเอ็มเอส/โคเซม และส่งข้อความไปยังมาตรอัจฉริยะดังรูปที่ 5-64 ในกรอบสีแดง

เมื่อมาตรอัจฉริยะได้รับข้อความ ก็จะตอบกลับด้วยข้อมูลที่ถูกร้องขอ โดยแบ่งเป็นส่วนย่อยๆ ดังรูปที่ 5-64 ในกรอบสีฟ้า และทุกครั้งที่รับข้อมูลส่วนย่อยๆ โปรแกรมประยุกต์จะส่งคำสั่งในชั้นเชื่อมต่อข้อมูลเพื่อบอกมาตรอัจฉริยะว่าได้รับข้อมูลย่อยที่ส่งมาก่อนหน้าแล้ว ดังรูปที่ 5-64 ในกรอบสีเขียว



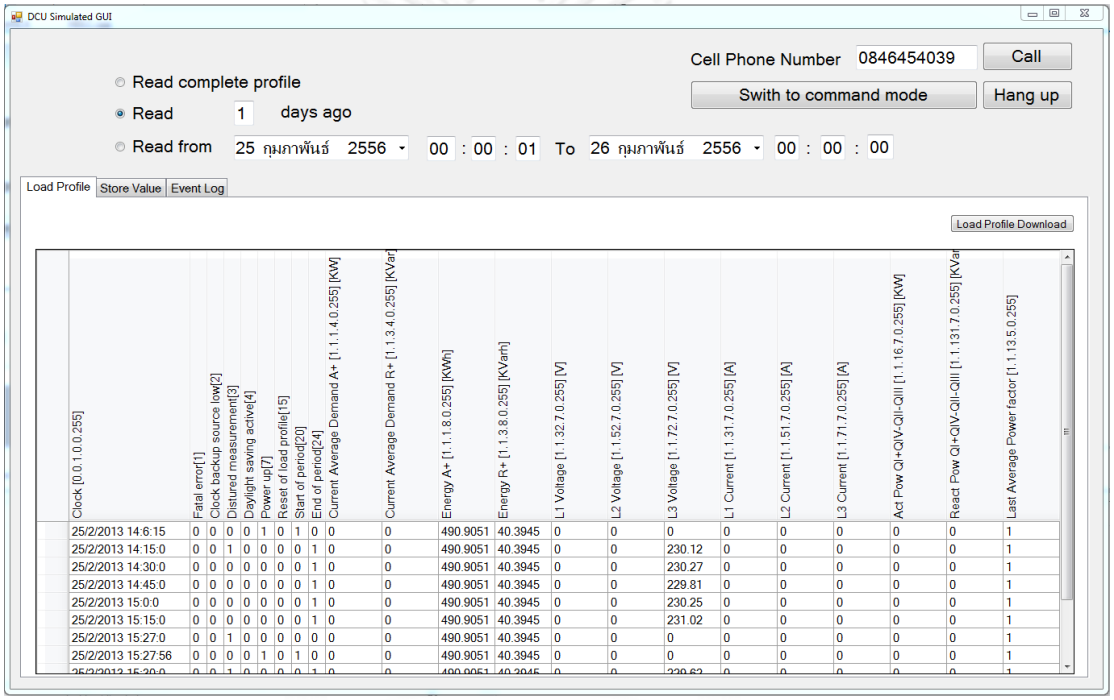
รูปที่ 5-64 การร้องขอ และตอบกลับข้อมูลโหลดโพรไฟล์โดยเลือกแบบ 1 วันย้อนหลัง (บางส่วน)

5.5.2. ผลการทดสอบการอ่านค่าโพรไฟล์ภาระ ข้อมูลสำหรับการเก็บเงินค่าบริการ และบันทึกเหตุการณ์

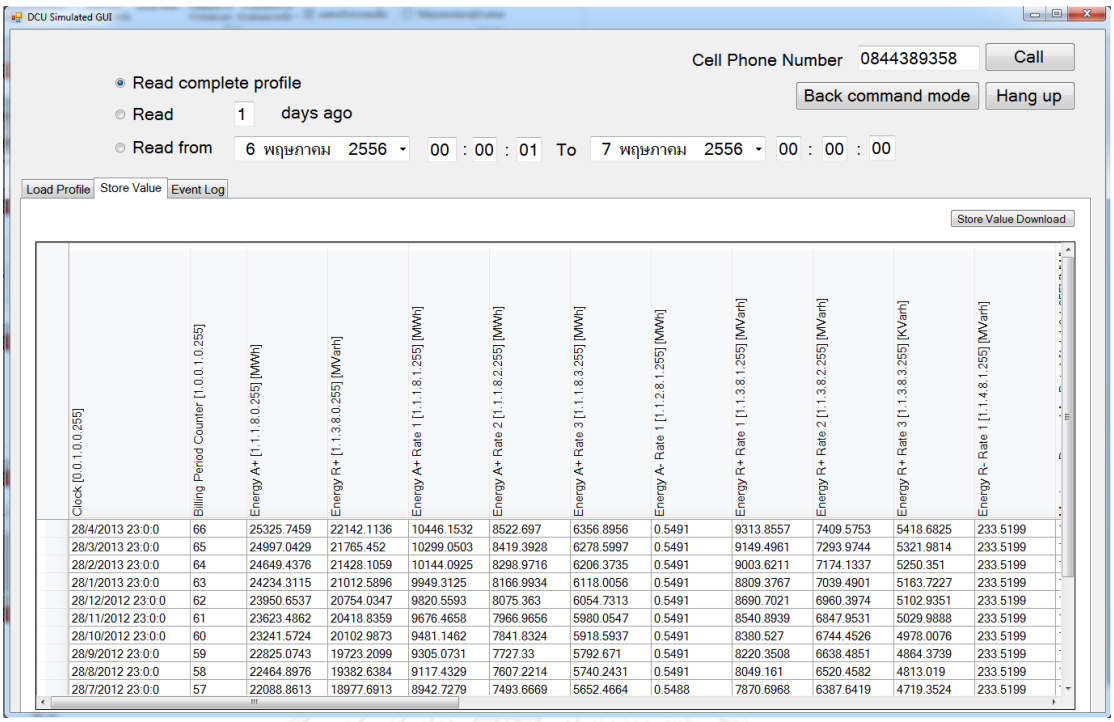
เมื่อรับข้อมูลส่วนย่อยๆ ครบทุกส่วนแล้ว ชั้นเชื่อมต่อข้อมูลจะเรียกใช้บริการของคำสั่ง GET_Cnf_Normal จากคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม เพื่อส่งข้อมูล

ทั้งหมดไปให้กับชั้นประยุกต์ เพื่อนำข้อมูลไปแสดงผลที่หน้าต่างโปรแกรมประยุกต์ และได้ผลการอ่านโพรไฟล์ภาระ ดังรูปที่ 5-65

กรณีของโพรไฟล์ข้อมูลสำหรับการเก็บเงินค่าบริการ และโพรไฟล์บันทึกเหตุการณ์ก็มีขั้นตอนในทำนองเดียวกันกับโพรไฟล์ภาระ โดยเลือกที่แท็บของหน้าต่างโปรแกรมประยุกต์ตามโพร-ไฟล์ที่ต้องการอ่าน เมื่อกดปุ่มบรรจุลงค่าที่ถูกเก็บ (Stored Values Download) และปุ่มบรรจุลงบันทึกเหตุการณ์ (Event Log Download) จะได้ผลดังรูปที่ 5-66 และรูปที่ 5-67 ตามลำดับ

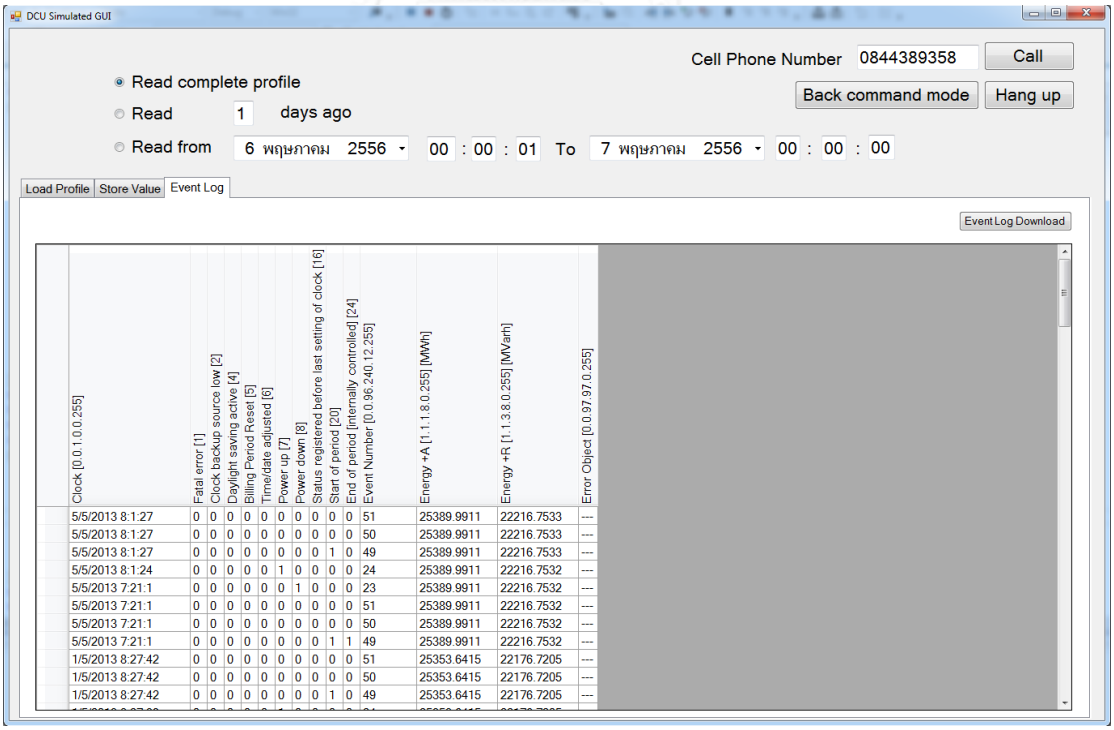


รูปที่ 5-65 ผลการอ่านโพรไฟล์ภาระจากมาตรอัจฉริยะที่ได้มาตรฐานดีแอลเอ็มเอส/โคเซม



Clock [0.0.1.0.0.255]	Billing Period Counter [1.0.0.1.0.255]	Energy A+ [1.1.1.8.0.255] [MWh]	Energy R+ [1.1.3.8.0.255] [MVarh]	Energy A+ Rate 1 [1.1.1.8.1.255] [MWh]	Energy A+ Rate 2 [1.1.1.8.2.255] [MWh]	Energy A+ Rate 3 [1.1.1.8.3.255] [MWh]	Energy A- Rate 1 [1.1.2.8.1.255] [MWh]	Energy R+ Rate 1 [1.1.3.8.1.255] [MVarh]	Energy R+ Rate 2 [1.1.3.8.2.255] [MVarh]	Energy R+ Rate 3 [1.1.3.8.3.255] [KVarh]	Energy R- Rate 1 [1.1.4.8.1.255] [MVarh]
28/4/2013 23:0:0	66	25325.7459	22142.1136	10446.1532	8522.697	6356.8956	0.5491	9313.8557	7409.5753	5418.6825	233.5199
28/3/2013 23:0:0	65	24997.0429	21765.452	10299.0503	8419.3928	6278.5997	0.5491	9149.4961	7293.9744	5321.9814	233.5199
28/2/2013 23:0:0	64	24649.4376	21428.1059	10144.0925	8298.9716	6206.3735	0.5491	9003.6211	7174.1337	5250.351	233.5199
28/1/2013 23:0:0	63	24234.3115	21012.5896	9949.3125	8166.9934	6118.0056	0.5491	8809.3767	7039.4901	5163.7227	233.5199
28/12/2012 23:0:0	62	23950.6537	20754.0347	9820.5593	8075.363	6054.7313	0.5491	8690.7021	6960.3974	5102.9351	233.5199
28/11/2012 23:0:0	61	23623.4862	20418.8359	9676.4658	7966.9656	5980.0547	0.5491	8540.8939	6847.9531	5029.9888	233.5199
28/10/2012 23:0:0	60	23241.5724	20102.9873	9481.1462	7841.8324	5918.5937	0.5491	8380.527	6744.4526	4978.0076	233.5199
28/9/2012 23:0:0	59	22825.0743	19723.2099	9305.0731	7727.33	5792.671	0.5491	8220.3508	6638.4851	4864.3739	233.5199
28/8/2012 23:0:0	58	22464.8976	19382.6384	9117.4329	7607.2214	5740.2431	0.5491	8049.161	6520.4582	4813.019	233.5199
28/7/2012 23:0:0	57	22088.8613	18977.6913	8942.7279	7493.6669	5652.4664	0.5488	7870.6968	6387.6419	4719.3524	233.5199

รูปที่ 5-66 ผลการอ่านข้อมูลสำหรับการเก็บเงินค่าบริการจากมาตรอัจฉริยะที่ได้มาตรฐานดีแอลเอ็มเอส/โคเซม



Clock [0.0.1.0.0.255]	Fatal error [1]	Clock backup source low [2]	Daylight saving active [4]	Billing Period Reset [5]	Time/date adjusted [6]	Power up [7]	Power down [8]	Status registered before last setting of clock [16]	Start of period [20]	End of period [internally controlled] [24]	Event Number [0.0.96.240.12.255]	Energy +A [1.1.1.8.0.255] [MWh]	Energy +R [1.1.3.8.0.255] [MVarh]	Error Object [0.0.97.97.0.255]
5/5/2013 8:1:27	0	0	0	0	0	0	0	0	0	0	51	25389.9911	22216.7533	---
5/5/2013 8:1:27	0	0	0	0	0	0	0	0	0	0	50	25389.9911	22216.7533	---
5/5/2013 8:1:27	0	0	0	0	0	0	0	0	1	0	49	25389.9911	22216.7533	---
5/5/2013 8:1:24	0	0	0	0	0	1	0	0	0	0	24	25389.9911	22216.7532	---
5/5/2013 7:21:1	0	0	0	0	0	0	1	0	0	0	23	25389.9911	22216.7532	---
5/5/2013 7:21:1	0	0	0	0	0	0	0	0	0	0	51	25389.9911	22216.7532	---
5/5/2013 7:21:1	0	0	0	0	0	0	0	0	0	0	50	25389.9911	22216.7532	---
5/5/2013 7:21:1	0	0	0	0	0	0	0	0	0	1	49	25389.9911	22216.7532	---
1/5/2013 8:27:42	0	0	0	0	0	0	0	0	0	0	51	25353.6415	22176.7205	---
1/5/2013 8:27:42	0	0	0	0	0	0	0	0	0	0	50	25353.6415	22176.7205	---
1/5/2013 8:27:42	0	0	0	0	0	0	0	0	1	0	49	25353.6415	22176.7205	---

รูปที่ 5-67 ผลการอ่านบันทึกเหตุการณ์จากมาตรอัจฉริยะที่ได้มาตรฐานดีแอลเอ็มเอส/โคเซม

โปรแกรมประยุกต์อ่านค่าจากมาตรอัจฉริยะที่ใช้งานจริงในอุตสาหกรรม ที่ได้มาตรฐานดีแอลเอ็มเอส/โคเซม สามารถอ่านค่าโพรไฟล์ต่างๆ จากมาตรอัจฉริยะได้ถูกต้อง โดยใช้บริการจากคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมที่ถูกพัฒนาขึ้นในการวิจัยนี้

จากการทดสอบในหัวข้อ 5.2 พบว่าระบบกลางจำลองสามารถอ่านค่าทางปริมาณทางไฟฟ้า และโพรไฟล์ภาระจากมาตรอัจฉริยะจำลองทั้งสามเครื่อง ผ่านทางต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล โดยใช้บริการของคำสั่ง GET จากคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ได้อย่างถูกต้อง และเป็นที่น่าพอใจ

จากการทดสอบในหัวข้อ 5.3 พบว่าสามารถตั้งค่าฐานเวลาภายในมาตรอัจฉริยะโดยใช้บริการของคำสั่ง SET จากคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ได้อย่างถูกต้องตามค่าเวลาที่ป้อนให้กับระบบกลางจำลอง แต่มีความคลาดเคลื่อนจากความหน่วงของการสื่อสาร ก่อนจะไปถึงมาตรอัจฉริยะจำลองอยู่บ้าง

จากการทดสอบในหัวข้อ 5.4 พบว่าสามารถเรียกใช้บริการภายในมาตรอัจฉริยะเพื่อควบคุมการเปิด-ปิดของแลทชิ่งรีเลยภายในมาตรอัจฉริยะ โดยใช้บริการของคำสั่ง ACTION จากคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ได้อย่างถูกต้อง และเป็นที่น่าพอใจ

จากผลการทดสอบในหัวข้อ 5.5 พบว่าโปรแกรมประยุกต์ที่พัฒนาขึ้นมา โดยใช้บริการของคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม เพื่อใช้อ่านค่าโพรไฟล์ต่างๆ ภายในมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรม สามารถอ่านโพรไฟล์ต่างๆ ได้อย่างถูกต้อง และเป็นที่น่าพอใจ

บทที่ 6

ข้อสรุปและข้อเสนอแนะ

6.1. ข้อสรุป

วิทยานิพนธ์ฉบับนี้ได้นำเสนอ รายละเอียดการออกแบบคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม ซึ่งเป็นโพรโทคอลการสื่อสารระหว่างอุปกรณ์เก็บรวบรวมข้อมูล (DCU) กับมาตรอัจฉริยะที่ได้รับความนิยมในทวีปยุโรป ซึ่งได้ถูกพัฒนาขึ้นเป็นภาษา C++ มาตรฐาน เนื่องจากสามารถนำไปประยุกต์ใช้ได้หลายแพลตฟอร์ม (Platform) และรายละเอียดการออกแบบเทรดจัดการชั้นโปรแกรมประยุกต์เพื่อจัดการส่งข้อมูลที่มีขนาดยาวกว่า บัฟเฟอร์ที่ได้ตกลงกันไว้ระหว่างอุปกรณ์ทั้งสอง เพื่อสะดวกต่อผู้ที่นำคลังโปรแกรมประยุกต์นี้ไปใช้ต่อ รวมถึงรายละเอียดการออกแบบของต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล ที่พัฒนาขึ้นจากบอร์ดพัฒนา Beagleboard-xM มี ARM Cortex A8 เป็นตัวประมวลหลัก ทำงานบนระบบปฏิบัติการอูบันตู และเรียกใช้บริการของคำสั่งภายในคลังโปรแกรมชั้นโปรแกรมประยุกต์ที่พัฒนาขึ้นในการวิจัยนี้ นอกจากนี้ยังได้พัฒนาซอร์ฟแวร์โปรแกรมประยุกต์ระบบกลางจำลอง และมาตรอัจฉริยะจำลอง ซึ่งเป็นซอร์ฟแวร์ที่ทำงานได้ทั้งบนระบบปฏิบัติการวินโดว์ 7 และวินโดว์ 8 และเชื่อมต่อกับฐานข้อมูลเอสคิวแอล (SQL Database) เพื่อเก็บค่าปริมาณทางไฟฟ้าจำลอง (ที่มาตร-อัจฉริยะของจริงจะได้จากการวัด) และข้อมูลโพรไฟล์ภาระ (Load Profile) จำลอง และเก็บค่าทุกๆ สิบห้านาที (กรณีมาตรอัจฉริยะจำลอง) หรือค่าปริมาณทางไฟฟ้า และโพรไฟล์ภาระที่อ่านได้จากมาตรอัจฉริยะจำลองแต่ละเครื่อง (กรณีระบบกลางจำลอง) เพื่อจำลองระบบโครงสร้างพื้นฐานการวัดขั้นสูง (AMI) พร้อมทั้งทดสอบฟังก์ชันการทำงานเบื้องต้นของระบบดังกล่าว ในด้านการสื่อสาร ระบบกลางจำลองส่งคำสั่งไปยัง และรับผลตอบสนองจากต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลในรูปแฟ้มข้อมูล XML ที่ถูกบีบอัดแบบ Gzip ผ่านอินเทอร์เน็ต โดยอุปกรณ์ทั้งคู่ต้องเชื่อมต่อเข้าสู่เครือข่ายส่วนบุคคลเสมือน (Virtual Private Network, VPN) เสียก่อน เพื่อให้เสมือนว่าเชื่อมต่ออยู่ในวงแลนเดียวกัน คอมพิวเตอร์ที่ระบบกลางจำลองทำงานอยู่นั้นต้องเชื่อมต่ออินเทอร์เน็ตผ่านสายแลน หรือแลนไร้สาย และใช้โปรแกรมประยุกต์ AnyConnect เพื่อเชื่อมต่อไปยังเครือข่ายส่วนบุคคลเสมือน บนระบบปฏิบัติการวินโดว์ และต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลเชื่อมต่ออินเทอร์เน็ตผ่านโมเด็มการสื่อสารผ่านอินเทอร์เน็ตแบบยูเอ็ม-ทีเอส/เอชเอสพีดีเอ ด้วยโปรแกรมประยุกต์ wvdial และเชื่อมต่อไปยังเครือข่ายส่วนบุคคลเสมือนด้วยโปรแกรมประยุกต์ openconnect บนระบบปฏิบัติการอูบันตู ส่วนการสื่อสารระหว่างต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลองนั้น ใช้การสื่อสารผ่านสายไฟฟ้าส่งกำลัง (PLC) ตามมาตรฐานไพร์ม (PRIME) โดยมีโมเด็มการสื่อสารผ่านสายไฟฟ้าส่งกำลังเป็นตัวกลาง และอุปกรณ์ทั้งสองเรียกใช้บริการของคำสั่งจากคลังโปรแกรมการสื่อสารผ่านสายไฟฟ้าส่งกำลัง เพื่อสั่งงานโมเด็มในการแลกเปลี่ยนข้อมูลกัน

การทดสอบแบบออกตามชนิดของคำสั่งที่เรียกใช้ได้แก่ GET, SET, ACTION ผลจากการเรียกใช้บริการของคำสั่ง GET เพื่อเรียกอ่านค่าที่อยู่ภายในอ็อบเจกต์ของมาตรอัจฉริยะจำลองทั้งสามเครื่อง โดยระบบกลางจำลอง ผ่านต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลนั้นพบว่า ในกรณีการส่งข้อมูลที่สั้นกว่า บัฟเฟอร์รับ-ส่งที่ได้ตกลงกันไว้ระหว่างอุปกรณ์เก็บรวบรวมข้อมูล กับมาตรอัจฉริยะจำลอง สามารถอ่านปริมาณทางไฟฟ้าได้ถูกต้อง และสมบูรณ์ ส่วนในกรณีการส่งข้อมูลที่ยาวกว่าบัฟเฟอร์รับ-ส่งที่ได้ตกลงกันไว้ระหว่างอุปกรณ์ทั้งสอง ทดสอบด้วยการอ่านค่าโพรไฟล์ภาระ ผลปรากฏว่าสามารถอ่านโพรไฟล์ภาระจากมาตรอัจฉริยะทั้งสามเครื่อง มาเก็บไว้ที่ฐานข้อมูลของระบบกลางจำลองได้ครบ และสมบูรณ์เช่นกัน จึงสรุปได้ว่าการทำงานของเทรดจัดการชั้นโปรแกรมประยุกต์ทั้งฝั่งเซิร์ฟเวอร์ และไคลเอนต์นั้นสามารถรับ-ส่งข้อมูลบล็อกย่อยๆ ได้โดยอัตโนมัติ โดยที่ผู้นำคลังโปรแกรมชั้นโปรแกรมประยุกต์ไปใช้ต่อ ไม่จำเป็นต้องทราบถึงกลไกการทำงานดังกล่าว

ส่วนการทดสอบเรียกใช้คำสั่ง SET เพื่อตั้งค่าฐานเวลาภายในมาตรอัจฉริยะจำลองทั้งสามเครื่องนั้น สามารถตั้งฐานเวลาได้ถูกต้อง แต่อาจเกิดความคลาดเคลื่อนเนื่องจากเวลาที่ใช้ในการส่งคำสั่งไปยังมาตรอัจฉริยะแต่ละเครื่อง และมาตรอัจฉริยะจำลองสามารถส่งผลยืนยันการตั้งค่าฐานเวลากลับไปยังระบบกลางจำลองได้

ส่วนการทดสอบเรียกใช้คำสั่ง ACTION เพื่อเรียกใช้บริการการควบคุมการเปิด-ปิดแลทชิ่งรีเลย์ภายในมาตรอัจฉริยะจำลอง สามารถทำงานได้ถูกต้องตามที่ระบบกลางจำลองร้องขอ และมาตรอัจฉริยะจำลองสามารถส่งผลยืนยันการควบคุมการเปิด-ปิดแลทชิ่งรีเลย์กลับไปยังระบบกลางจำลองได้

นอกจากนี้ยังมีการทดสอบนำคลังโปรแกรมชั้นประยุกต์ดีแอลเอ็มเอส/โคเซม ไปใช้ในพัฒนาโปรแกรมประยุกต์บนระบบปฏิบัติการวินโดว์ เพื่อการอ่านค่าโพรไฟล์ต่างๆ ได้แก่ โพรไฟล์ภาระ โพรไฟล์ข้อมูลสำหรับการเก็บเงินค่าบริการ (Billing Data) และโพรไฟล์บันทึกเหตุการณ์ (Event Log) จากมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรม ผ่านการสื่อสารผ่านระบบโทรศัพท์เคลื่อนที่จีเอสเอ็ม (GSM) โดยใช้โพรไฟล์สื่อสารแบบสามชั้นแบบเชื่อมต่อก่อนเอชดีแอลซี (Three-layer, connection-oriented, HDLC-based communication profile) ผลการทดสอบปรากฏว่าโปรแกรมประยุกต์ดังกล่าวสามารถอ่านโพรไฟล์ต่างๆ จากมาตรอัจฉริยะที่ใช้จริงในอุตสาหกรรมได้อย่างถูกต้อง และสมบูรณ์

จากผลการทดสอบที่ได้กล่าวไปแล้วข้างต้น ทำให้สรุปได้ว่าคลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซมที่พัฒนาขึ้นในการวิจัยนี้ สามารถนำไปใช้งานได้จริง

6.2. ประโยชน์ที่ได้รับ

1. ได้ซอร์ฟแวร์คลังโปรแกรมชั้นโปรแกรมประยุกต์ที่เป็นไปตามมาตรฐานดีแอลเอ็มเอส/โคเซม (DLMS/COSEM)

2. ได้ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลตามมาตรฐานดีแอลเอ็มเอส/โคเซม ที่ติดต่อกับมาตรอัจฉริยะผ่านทางการสื่อสารผ่านสายไฟฟ้าส่งกำลังตามมาตรฐานไพร์ม และส่งข้อมูลให้กับระบบกลางผ่านทางระบบเครือค่ายโทรศัพท์เคลื่อนที่ในยุคที่ 3

3. ได้โปรแกรมประยุกต์ระบบกลางจำลองที่ติดต่อกับต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลผ่านอินเทอร์เน็ต

4. ได้โปรแกรมประยุกต์มาตรอัจฉริยะจำลองที่ติดต่อกับต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลผ่านทางการสื่อสารผ่านสายไฟฟ้าส่งกำลังตามมาตรฐานไพร์ม

6.3. ข้อเสนอแนะ

1. คลังโปรแกรมชั้นโปรแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม สามารถทำให้เหมาะที่สุด (Optimize) ได้ เช่น ด้านหน่วยความจำ อาจจะนำบัฟเฟอร์ที่จำเป็นน้อยออก เพิ่มประหยัดพื้นที่หน่วยความจำ แต่อาจต้องแลกมาด้วยรหัสคำสั่ง (Code) ที่ยาวขึ้น

2. คลังโปรแกรมชั้นโปนแกรมประยุกต์ดีแอลเอ็มเอส/โคเซม อาจปรับปรุงให้สามารถใช้งานง่ายขึ้นได้อีก เช่น เพิ่มบริการสำเร็จรูปที่ใช้บ่อยๆ

3. เพิ่มฟังก์ชันการทำงานของอุปกรณ์จำลองทั้งสาม ได้แก่ ระบบกลางจำลอง อุปกรณ์เก็บรวบรวมข้อมูล และมาตรอัจฉริยะจำลอง เพื่อให้มีฟังก์ชันใกล้เคียงกับของจริงมากที่สุด

   - ระบบกลางจำลอง อาจจะเพิ่มฟังก์ชันพล็อตกราฟจากข้อมูลที่ได้รับมาจากมาตรอัจฉริยะจำลอง หรือฟังก์ชันการอ่านค่าโพรไฟล์แบบกำหนดช่วงเวลา เป็นต้น

   - อุปกรณ์เก็บรวบรวมข้อมูล อาจจะเพิ่มฟังก์ชัน การเรียกอ่านโพรไฟล์ภาระจากมาตรอัจฉริยะทุกตัว โดยอัตโนมัติทุกๆ หนึ่งชั่วโมง เป็นต้น

   - มาตรอัจฉริยะจำลอง อาจจะเพิ่มฟังก์ชัน การแจ้งเตือนโดยอัตโนมัติเมื่อเกิดเหตุการณ์สำคัญต่างๆ เช่น การเกิดไฟฟ้าดับ มาตรอัจฉริยะจำลองต้องแจ้งไปยังต้นแบบอุปกรณ์เก็บรวบรวมข้อมูล เพื่อให้ต้นแบบอุปกรณ์เก็บรวบรวมข้อมูลประมวลผลบริเวณที่เกิดไฟดับ แล้วจึงแจ้งไปยังระบบกลางจำลอง และผู้ให้บริการระบบไฟฟ้าทราบ เป็นต้น

4. อินเทอร์เน็ตที่ใช้ควรมีความเร็วสูง และมีเสถียรภาพที่ดี ไม่เช่นนั้น แฟ้มข้อมูลที่ถูกรับ-ส่งอาจเสีย หรือไม่ครบสมบูรณ์

5. การตั้งค่าฐานเวลาภายในมาตรอัจฉริยะจำลองควรทำการชดเชยเวลาสำหรับมาตรอัจฉริยะที่มีลำดับการถูกตั้งค่าฐานเวลาเป็นเครื่องท้ายๆ เนื่องจากผลของการหน่วง (Delay) จากการสื่อสาร สามารถชดเชยได้โดยการส่งข้อมูลทดสอบไปหลาย ๆ ครั้งก่อน

และจับเวลาในการส่งกลับ เพื่อหาเวลาเฉลี่ยของการรับ-ส่งข้อมูล แล้วจึงนำมาชดเชยให้กับเวลาที่ต้องการจะตั้งค่าให้มาตรอัจฉริยะเครื่องนั้นๆ หรืออาจส่งคำสั่งตั้งค่าฐานเวลาเพียงครั้งละเครื่องเท่านั้น และทำการชดเชยเวลาร่วมด้วย เพื่อความแม่นยำที่ดีกว่า

รายการอ้างอิง

1. Provincial Electricity Authority. *Smart Grid Frequently Asked Questions (FAQs)*. 2012 [cited 2012 29/12/2012]; Available from: http://161.200.85.41/pea-smartgrid/index.php/smart-grid/9-uncategorised.
2. Cheokul, R., *The Renewable and Alternative Energy Development Plan for* 25 *Percent in* 10 *Years (AEDP* 2012-2021). 2012.
3. DLMS User Association, *DLMS/COSEM Architecture and Protocols*, in *Green Book*. 2009.
4. DLMS User Association, *DLMS/COSEM Identification System and Interface Classes*, in *Blue Book*. 2010.
5. IEC, *Electricity metering - Data exchange for meter reading, tariff and load control - Part* 46: *Data link layer using HDLC protocol*, in *IEC* 62056-46. 2007. p. 1-72.
6. IEC, *Electricity metering - Data exchange for meter reading, tariff and load control - Part* 53: *COSEM application layer*, in *IEC* 62056-53. 2006. p. 1-144.
7. IEC, *Electricity metering - Data exchange for meter reading, tariff and load control - Part* 61: *Object identification system (OBIS)*, in *IEC* 62056-61. 2006. p. 1-38.
8. IEC, *Electricity metering - Data exchange for meter reading, tariff and load control - Part* 62: *Interface classes*, in *IEC* 62056-62. 2006. p. 1-125.
9. OPEN meter, *Requirements of AMI*, in *D* 1.1. 2009.
10. IEC, *Information technology-ASN.*1 *encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, in *ISO/IEC* 8825-1. 2002, ISO: Switzerland.
11. IEC, *Distribution automation using distribution line carrier systems-Part* 6: *A-XDR encoding rule*, in *IEC* 61334-6. 2000.
12. PRIME Alliance Technical Working Group, *Draft Specification for Powerline Intelligent Metering Evolution*.
13. PRIME Project, *PHY, MAC and Convergence layers*, in *Technology Whitepaper*. 2008.
14. Cypress Semiconductor Corporation. *What is Power Line Communication*. designlines 2011 [cited 2014 16/04/2014]; Available from: http://www.eetimes.com/document.asp?doc_id=1279014.
15. DORA, *ST*7590 *External Controller Interface*. 2009, STMicroelectronics.
16. beagleboard.org. *BeagleBoard-xM Rev C System Reference Manual*. 2010 [cited 2014 02/05/2014]; Available from: http://beagleboard.org/static/BBxMSRM_latest.pdf.

17. STMicroelectronics. *Quick Start Guide - ST7590 Power Line Modem Demo Board*. 2009 [cited 2014 02/05/2014]; Available from: http://www.mitracon.ru/pdf/e/evalst7590%20quick%20start%20guide.pdf.
18. ren-ben.en.alibaba.com. *Sierra Compass 885 Aircard USB Modem Specifications*. [cited 2014 02/05/2014]; Available from: http://ren-ben.en.alibaba.com/product/356396033-210092000/Sierra_Compass_885_aircard_usb_Modem.html.
19. meter, O., *Communication profile: PLC based on PRIME OFDM*, in *D* 5.2.1. 2011. p. 1-64.
20. meter, O., *Design of the overall System Architecture*, in *D* 3.1. 2010. p. 1-83.
21. Limphapayom, S. and W. Pora. *An emulation of data concentrator units conformed to DLMS-HDLC protocols*. in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON),* 2013 10*th International Conference on*. 2013.

ภาคผนวก

ประวัติผู้เขียนวิทยานิพนธ์

นายสิวะรัฐ ลิมปพยอม เกิดเมื่อวันที่ 15 ธันวาคม พ.ศ. 2531 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือในปีการศึกษา 2553 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2554