การพัฒนาอุปกรณ์และโพรโตคอลโปรฟิบัส

PROFIBUS Device and Protocol Development

นางสาวพัชรินทร์ พุ่มเฉี่ยว

นายภาณุ ไชยสิทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การพัฒนาอุปกรณ์และโพรโตคอลโปรฟิบัส

PROFIBUS Device and Protocol Development

โดย

นางสาวพัชรินทร์ พุ่มเฉี่ยว

นายภาณุ ไชยสิทธิ์

อาจารย์ที่ปรึกษา

ผศ.อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาอุปกรณ์และโพรโตคอลโปรฟิบัส

PROFIBUS Device and Protocol Development

คณะผู้จัดทำ นางสาวพัชรินทร์ พุ่มเฉี่ยว รหัส 43010296

นายภาณุ ไชยสิทธิ์ รหัส 43010323

.....อาจารย์ที่ปรึกษา

(ผศ.อภิเนตร อูนากูล)

การพัฒนาอุปกรณ์และโพรโตคอลโปรฟิบัส

นางสาวพัชรินทร์ พุ่มเฉี่ยว 43010296
นายภาณุ ไชยสิทธิ์ 43010323
ผศ.อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ในปัจจุบันระบบอุตสาหกรรมมีการใช้งานการควบคุมแบบอัตโนมัติ ภายในระบบดังกล่าวนิยมใช้มาตรฐานโปรโตคอล PROFIBUS DP (IEC 61158) ซึ่งอุปกรณ์ที่ใช้งานส่วนใหญ่พัฒนาโดยบริษัท Siemens จุดประสงค์หลักของโครงการนี้ คือ การสร้างแนวทางการพัฒนาอุปกรณ์ Slave บนเครือข่าย PROFIBUS DP ที่สามารถเข้าไปใช้งานกับระบบควบคุมอัตโนมัติในเครือข่าย PROFIBUS DP ได้ สำหรับโครงการนี้เป็นการถอดรหัสของโปรโตคอล PROFIBUS DP เพื่อทราบถึงรูปแบบของการสื่อสารระหว่างอุปกรณ์และพัฒนาโปรโตคอล Library ในส่วนของอุปกรณ์ PROFIBUS DP Slave เป็นการพัฒนาบน Platform x86 โดยใช้บอร์ด Com86 ใช้ระบบปฏิบัติการแบบ Real-time ชื่อว่า Micro-C OS/2 และมีการพัฒนาเป็นอุปกรณ์ต้นแบบ PROFIBUS DP-Slave เพื่อสร้างแนวทางสำหรับผู้ที่สนใจสามารถนำไปพัฒนาเป็นอุปกรณ์ Slave ต่อไปได้

PROFIBUS Device and Protocol Development

Miss.Pacharin Poomchiew 43010296
Mr.Panu Chaiyasith 43010323
Asst.Prof.Apineth Unakul Advisor
Academic Year 2546

ABSTRACT

In industry nowadays, there is widely usage of automation systems that widely use PROFIBUS DP (IEC 61158) protocol. Most devices was developed by Siemens Co, Ltd. Main goal of this project is creating a guideline for developing slave device that communicate wtih automation system on PROFIBUS DP network to reduce the number of imported devices and create a guideline for Thais to develop this device. This project provides studying PROFIBUS DP protocol to understand communication between devices, developing protocol library for PROFIBUS DP slave on X86 platform using Real-time OS named Micro-C OS/2 on single COM86 board, and developing the prototype of PROFIBUS DP slave device to be a guideline for further development.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้เป็นอย่างดี เนื่องมาจากบุคคลหลายๆฝ่ายคอยให้ความช่วยเหลืออยู่เสมอ บุคคลท่านแรกที่ขอกล่าวถึงคือ ผศ.อภิเนตร อูนากูล ท่านอาจารย์ที่ปรึกษาที่คอยให้ความรู้ คำแนะนำ และคอยดูแลเอาใจเสมอมา ซึ่งต้องขอกราบขอบพระคุณเป็นอย่างยิ่ง

อันดับต่อไปขอขอบพระคุณภาควิชาคอมพิวเตอร์ ที่มีห้องปฏิบัติการ ESL ให้เป็นแหล่งค้นคว้าหาความรู้ ใช้เป็นห้องปฏิบัติงานวิจัย และเป็นแหล่งพบปะเพื่อนร่วมงาน

สุดท้ายนี้ขอขอบพระคุณบุคคลสำคัญที่สุดคือ บิดา มารดา ผู้ที่ทำให้มีวันนี้ ผู้เป็นที่รัก และเคารพยิ่ง ผู้ซึ่งคอยเลี้ยงดู ให้ความรัก เอาใจใส่ และคอยให้กำลังใจอยู่เสมอมา ขอระลึกในพระคุณ และกราบขอบพระคุณมา ณ ที่นี้

พัชรินทร์ พุ่มเฉี่ยว
ภาณุ ไชยสิทธิ์

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญภาพ	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์งานวิจัย	2
1.3 ขอบเขตงานวิจัย	2
1.4 เป้าหมายงานวิจัย	2
1.5 วิธีการดำเนินงาน	2
บทที่ 2 มาตรฐานโปรฟีบัส	4
2.1 มาตรฐานโปรฟีบัส (PROFIBUS-Process Field Bus Standard)	4
2.1.1 โครงสร้างการสื่อสาร	4
2.1.2 สถาปัตยกรรมโพรโตคอล	6
2.1.2.1 Physical Layer	6
2.1.2.2 Data Link Layer	8
2.1.2.3 Application Layer	11
2.2 โปรฟีบัสดี-พี (PROFIBUS-DP-Process Field Bus Decentralized Periphery)	11
2.2.1 ลักษณะพื้นฐาน	11
2.2.2 สถาปัตยกรรมโพรโตคอล	12
2.3.1 สถานะการทำงาน	15
2.3.1.1 Initial Power ON/Reset	15
2.3.1.2 Parameterization	15
2.3.1.3 I/O Configuration	16
2.3.2 โครงสร้างการทำงาน	17
2.3.2.2 DDLM-FMA1/2	21
2.3.2.3 DDLM-FDL	21
2.3.3 ลำดับการทำงาน	22
2.4 GSD File	25

สารบัญ (ต่อ)

หน้าที่

2.4.1 โครงสร้าง GSD File 25

2.4.2 ตัวอย่าง GSD file 26

2.4.3 โปรแกรมแก้ไข-ตรวจสอบ GSD File 28

บทที่ 3 การออกแบบอุปกรณ์รองโปรฟิบัสดี-พี 29

3.1 PROFIBUS-DP Layer 1 (Physical Layer) 30

3.1.1 ลักษณะทางไฟฟ้าและกลไกการทำงาน 30

3.1.2 การเชื่อมต่อระหว่าง PHY กับ FDL 33

3.1.3 การเชื่อมต่อระหว่าง PHY กับ FMA 34

3.1.4 สรุปการออกแบบระดับ Physical Layer 37

3.2 PROFIBUS Layer 2 (Data Link Layer) 38

3.2.1 Fieldbus Data Link (FDL) 38

3.2.1.1 FDL User - FDL Interface 38

3.2.1.2 การออกแบบส่วน FDL 42

3.2.1.3 การออกแบบฟังก์ชันในการบริการสำหรับส่งและรับข้อมูลกับ FDL 42

3.2.2 Fieldbus Management (FMA) 45

3.2.2.1 FMA1/2 User - FMA1/2 Interface 45

3.2.3 FDL - FMA1/2 Interface 52

3.2.3.2 รายละเอียดการทำงาน 53

3.2.4 สรุปการออกแบบส่วน Layer 2 55

3.3 ส่วนติดต่อกับชั้น Data link โดยตรง DDLM (Direct Data Link Mapper) 56

3.3.1 การเชื่อมต่อระหว่าง DDLM และชั้นที่ 2 (Interface between DDLM and Layer 2) 56

3.3.2 การเชื่อมต่อระหว่าง DDLM และ User-Interface 58

3.3.3 ฟังก์ชันการทำงานระหว่าง DP-Master กับ DP-Slave 58

3.3.3.1 การอ่านค่าข้อมูลวินิจฉัยของ DP-Slave จาก DP-Master 58

3.3.3.2 การส่งและรับข้อมูล Input และ Output 59

3.3.3.3 การอ่านข้อมูล Input และ Output ของ DP-Slave 59

3.3.3.4 การส่งข้อมูลเกี่ยวกับพารามิเตอร์ 60

3.3.3.5 การตรวจสอบข้อมูลที่ตั้งไว้ (Configuration Data) 60

3.3.3.6 การอ่านค่าข้องข้อมูลที่ตั้งไว้ (Configuration Data) 60

3.3.3.7 คำสั่งควบคุมไปยัง DP-Slave 61

3.3.3.8 คำสั่งกำหนดหมายเลขตำแหน่ง (Address) ให้กับ DP-Slave 61

3.3.4 การเชื่อมต่อระหว่าง DDLM และ User-Interface และ FDL 62

สารบัญ (ต่อ)

หน้าที่

3.3.5 ฟังก์ชันภายในของ DP-Slave 62
3.3.6 สถานะการทำงานของ DDLM 64
3.3.7 การออกแบบการเชื่อมต่อระหว่างฟังก์ชั่นของ DDLM และ User-Interface และ FDL 64
3.4 ส่วนติดต่อผู้ใช้งาน (User-Interface) 65
3.4.1 ส่วน User-Interface DDLM 65
3.4.2 ส่วน User กับ User-Interface 66
3.5. ผู้ใช้งาน (User) 67
3.6 โครงสร้างข้อมูล 68
3.7 Sequence Diagram 69
บทที่ 4 การทดสอบการทำงานอุปกรณ์รองโปรฟีบัสดี-พี 71
4.1 การวิเคราะห์การส่งข้อมูลโปรฟีบัส 71
4.1.1 การรับ-ส่งข้อมูลวินิจฉัย (Diag_Data) 72
4.1.2 การรับ-ส่งข้อมูลพารามิเตอร์ (Set_Prm) 72
4.1.3 การตรวจสอบข้อมูลองค์ประกอบ (Chk_Cfg) 73
4.1.4 การส่งข้อมูลควบคุม (Global_Control) 73
4.1.5 การแลกเปลี่ยนข้อมูล (Data_Exchange) 74
4.2 การทดสอบการทำงาน 74
4.2.1 การทดสอบในระดับ Physical layer (PHY) 75
4.2.2 การทดสอบการรับและส่งข้อมูลระหว่างส่วน PHY กับ FDL 76
4.2.2.1 การทดสอบการส่งข้อมูลจาก PHY ไปหา FDL 76
4.2.2.2 การทดสอบการส่งข้อมูลจาก FDL ไปหา PHY 78
4.2.3 การทดสอบในระดับ Fieldbus Data link layer (FDL) 79
4.2.3.1 การทดสอบการตอบ-รับ Telegram ของโปรฟีบัส 79
4.3 การทดสอบการตอบ-รับบริการ SDN และ SRD ของ FDL PROFIBUS-DP 84
4.3.1 การส่งข้อมูลแบบ SDN 84
4.3.2 การส่งข้อมูลแบบ SRD 84
4.4 การทดสอบสถานะทำงาน 84
4.5 GSD File Simple Slave Device 87
บทที่ 5 บทวิจารณ์และสรุป 89
5.1 สรุปผลการดำเนินงาน 89
5.1.2 สรุปขั้นตอนการดำเนินงาน 90
5.1.3 ความถูกต้อง 90

สารบัญ (ต่อ)

	หน้าที่
5.14 ข้อจำกัด	90
5.1.5 ปัญหาในการดำเนินงาน	90
5.2 แนวทางการพัฒนาต่อ	91
5.3 บทวิจารณ์	91
ภาคผนวก ก. ระบบปฏิบัติการแบบเวลาจริง	93
ภาคผนวก ข. Micro Controller Operating System Version 2	97
ภาคผนวก ค. Com86	100
บรรณานุกรม	101

สารบัญตาราง

หน้าที่

ตารางที่ 2-1 รายละเอียดขาของดีบี-9 (DB-9) 7
ตารางที่ 2-2 การให้บริการของระดับชั้นที่ 2 8
ตารางที่ 2-3 ประเภทของจุดเริ่มต้นเทลเลแกรม 9
ตารางที่ 2-4 รูปแบบการใช้งานการควบคุมเทเลแกรมแบบที่ 1 14
ตารางที่ 2-5 รูปแบบการใช้งานการควบคุมเทเลแกรมแบบที่ 2 14
ตารางที่ 2-6 ประเภทการให้บริการโปรฟีบัสดี-พี 14
ตารางที่ 2-7 การทำงานพื้นฐานของโปรฟีบัสดี-พี 14
ตารางที่ 2-8 สถานะข้อมูลตอบกลับ 15
ตารางที่ 2-9 GSD File ทั่วๆ ไปของอุปกรณ์ต่างๆ 26
ตารางที่ 2-10 ข้อมูล GSD File ของอุปกรณ์รอง 26
ตารางที่ 3-1 การติดต่อระหว่างฟังก์ชันของ DDLM กับ Layer ผ่านจุด SSAP 57

สารบัญภาพ

หน้าที่

รูปที่ 1-1 Slave Device PROFIBUS-DP 2
รูปที่ 2-1 ระดับการสื่อสารของระบบอุสาหกรรมตามมาตรฐานโปรฟีบัส 4
รูปที่ 2-2 การสื่อสารระหว่างอุปกรณ์ตามมาตรฐานโปรฟีบัส 5
รูปที่ 2-3 ความสัมพันธ์โครงสร้างโอเอสไอ และ มาตรฐานโปรฟีบัส 6
รูปที่ 2-4 โครงสร้างมาตรฐานการสื่อสารของโปรฟีบัส 6
รูปที่ 2-5 รูปแบบเฟรมข้อมูลโปรฟีบัส 7
รูปที่ 2-6 การเชื่อมต่อสายบัสตามมาตรฐานโปรฟีบัส 7
รูปที่ 2-7 รูปแบบการรับ-ส่งเฟรมของโปรฟีบัส 9
รูปที่ 2-8 รูปแบบเฟรมเทเลแกรมโปรฟีบัส 9
รูปที่ 2-9 รูปแบบเทเลแกรมที่กำหนดความยาวคงที่ 9
รูปที่ 2-10 รูปแบบเทเลแกรมที่กำหนดความยาวข้อมูลคงที่ 10
รูปที่ 2-11 รูปแบบเทเลแกรมที่เปลี่ยนแปลงตามขนาดข้อมูล 10
รูปที่ 2-12 รูปแบบเทเลแกรมโทเคน 10
รูปที่ 2-13 รูปแบบเทเลแกรมการตอบกลับแบบสั้น 10
รูปที่ 2-14 สถาปัตยกรรมโปรฟีบัสดี-พี 12
รูปที่ 2-15 โครงสร้างการสื่อสารโปรฟีบัสดี-พี 12
รูปที่ 2-16 การรับ-ส่งข้อมูลแบบมีการรับรอง 13
รูปที่ 2-17 การส่งข้อมูลแบบไม่มีการรับรอง 13
รูปที่ 2-18 สถานะการทำงานของอุปกรณ์รอง 16
รูปที่ 2-19 โครงสร้างการทำงานของอุปกรณ์รอง 17
รูปที่ 2-20 โครงสร้างเทเลแกรม Set_Slave_Add 18
รูปที่ 2-21 โครงสร้างเทเลแกรม Chk_Cfg 18
รูปที่ 2-22 โครงสร้างเทเลแกรม Set_Prm 19
รูปที่ 2-23 โครงสร้างเทเลแกรม Data_Exchange 20
รูปที่ 2-24 โครงสร้างเทเลแกรม Diag_Data 20
รูปที่ 2-25 โครงสร้างเทเลแกรม Global_Control 21
รูปที่ 2-26 ลำดับการทำงานขั้นตอนที่ 1 22
รูปที่ 2-27 ลำดับการทำงานขั้นตอนที่ 2 23
รูปที่ 2-28 ลำดับการทำงานขั้นตอนที่ 3 23
รูปที่ 2-29 ลำดับการทำงานขั้นตอนที่ 4 24
รูปที่ 2-30 ลำดับการทำงานขั้นตอนที่ 5 25
รูปที่ 2-31 โปรแกรม GSD Editor 28

สารบัญภาพ (ต่อ)

หน้าที่

รูปที่ 3-1 สถาปัตยกรรมโครงสร้างของอุปกรณ์รองโปรฟีบัสดี-พี 29
รูปที่ 3-2 ลำดับของการส่งข้อมูล 30
รูปที่ 3-3 รูปแบบสัญญาณแบบ NRZ 31
รูปที่ 3-4 รูปแบบการต่อปลายสายสัญญาณโปรฟีบัส 31
รูปที่ 3-5 ลักษณะการเชื่อมต่อสายสัญญาณของ DB9-Connector 31
รูปที่ 3-6 Block Diagram แปลงสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485 32
รูปที่ 3-7 วงจรแปลงสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485 32
รูปที่ 3-8 แผ่น PCB ของวงจรสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485 33
รูปที่ 3-9 แสดงมุมมองการเชื่อมต่อของ PHY กับ FDL 33
รูปที่ 3-10 การเชื่อมต่อระหว่าง PHY กับ FDL 33
รูปที่ 3-11 แสดงมุมมองการเชื่อมต่อของ PHY กับ FMA1/2 34
รูปที่ 3-12 ลักษณะการทำงานของบริการ Reset PHY, Set Value PHY, Read Value 35
รูปที่ 3-13 ลักษณะการทำงานของบริการ Event PHY 35
รูปที่ 3-14 การออกแบบโครงสร้างการทำงานของ PHY 37
รูปที่ 3-15 การออกแบบโครงสร้างการรับ-ส่งข้อมูลของ PHY 37
รูปที่ 3-16 การเชื่อมต่อของ FDL User กับ FDL 38
รูปที่ 3-17 การส่งข้อมูลแบบ SDN 38
รูปที่ 3-18 การส่ง-ร้องขอข้อมูลแบบ SRD 40
รูปที่ 3-19 การออกแบบโครงสร้างการทำงานของ FDL 42
รูปที่ 3-20 การเชื่อมต่อของ FMA1/2 User กับ FMA1/2 45
รูปที่ 3-21 การทำงานของฟังก์ชัน Reset, Set, Read, Ident, LSAP, (R)SAP, SAP 46
รูปที่ 3-22 การทำงานของฟังก์ชัน Reset, Set, Read, Ident, LSAP, (R)SAP, SAP 46
รูปที่ 3-23 การเชื่อมต่อของ FDL กับ FMA1/2 Interface 52
รูปที่ 3-24 การทำงานของฟังก์ชันReset FDL, Set Value FDL, Read Value 53
รูปที่ 3-25 การทำงานของฟังก์ชัน Fault FDL 53
รูปที่ 3-26 การออกแบบโครงสร้างการทำงานของส่วน Layer 2 55
รูปที่ 3-27 การเชื่อมต่อของ DDLM กับ Layer 2 56
รูปที่ 3-28 การเชื่อมต่อของ DDLM กับ User-Interface 58
รูปที่ 3-29 การออกแบบการเชื่อมต่อระหว่าง DDLM และ User-Interface และ FDL 62
รูปที่ 3-30 State Diagram ของ DDLM ใน DP-Slave 62
รูปที่ 3-31 การเชื่อมต่อระหว่างฟังก์ชั่นของ DDLM และ User-Interface และ FDL 64
รูปที่ 3-32 การเชื่อมต่อของ User-Interface และ DDLM 65

สารบัญภาพ (ต่อ)

หน้าที่

รูปที่ 3-33 การออกแบบการเชื่อมต่อของ DDLM และ User-Interface และ User 66
รูปที่ 3-34 การออกแบบการทำงานของระดับ User 67
รูปที่ 3-35 โครงสร้างของอุปกรณ์ DP-Slave 67
รูปที่ 3-36 โครงสร้าง Diagnostic Data 68
รูปที่ 3-37 โครงสร้าง Configuration Data 69
รูปที่ 3-38 Sequence Diagram 70
รูปที่ 4-1 ตัวอย่างโปรแกรมวิเคราะห์การสื่อสารโปรฟิบัสแบบคำสั่ง 71
รูปที่ 4-2 ตัวอย่างโปรแกรมวิเคราะห์การสื่อสารโปรฟิบัสแบบเทเลแกรม 71
รูปที่ 4-3 วิเคราะห์การรับ-ส่งข้อมูลวินิจฉัย 72
รูปที่ 4-4 วิเคราะห์เทเลแกรมการรับ-ส่งข้อมูลวินิจฉัย 72
รูปที่ 4-5 วิเคราะห์เทเลแกรมการ ส่งข้อมูลวินิจฉัยของอุปกรณ์หลัก 72
รูปที่ 4-6 วิเคราะห์เทเลแกรมการ ส่งข้อมูลวินิจฉัยของอุปกรณ์รอง 72
รูปที่ 4-7 วิเคราะห์การรับ-ส่งข้อมูลพารามิเตอร์ 73
รูปที่ 4-8 วิเคราะห์เทเลแกรมการรับ-ส่งข้อมูลพารามิเตอร์ 73
รูปที่ 4-9 วิเคราะห์การตรวจสอบข้อมูลองค์ประกอบ 73
รูปที่ 4-10 วิเคราะห์เทเลแกรมการตรวจสอบข้อมูลองค์ประกอบ 73
รูปที่ 4-11 วิเคราะห์การส่งข้อมูลควบคุม 73
รูปที่ 4-12 วิเคราะห์เทเลแกรมการส่งข้อมูลควบคุม 73
รูปที่ 4-13 วิเคราะห์การแลกเปลี่ยนข้อมูล 74
รูปที่ 4-14 วิเคราะห์เทเลแกรมการแลกเปลี่ยนข้อมูลของอุปกรณ์หลัก 74
รูปที่ 4-15 วิเคราะห์เทเลแกรมการส่งข้อมูลควบคุมของอุปกรณ์รอง 74
รูปที่ 4-16 โครงสร้างของโปรโตคอลสำหรับอุปกรณ์ Slave 75
รูปที่ 4-17 การทำงานบน Com86 กับโปรแกรม test01.exe 75
รูปที่ 4-18 การทำงานบน PC 76
รูปที่ 4-19 การส่งข้อมูลจาก PC 77
รูปที่ 4-20 การรับข้อมูลบน Com86 กับโปรแกรม test02.exe 77
รูปที่ 4-21 การส่งข้อมูลระหว่าง FDL กับ PHY บน Com86 ด้วยโปรแกรม test03.exe 79
รูปที่ 4-22 การส่งข้อมูลแบบที่ 1 (SD1) 79
รูปที่ 4-23 ผลการรับข้อมูลแบบที่ 1 (SD1) 80
รูปที่ 4-24 การส่งข้อมูลแบบที่ 2 (SD2) 80
รูปที่ 4-25 ผลการรับข้อมูลแบบที่ 2 (SD2) 81
รูปที่ 4-26 การส่งข้อมูลแบบที่ 3 (SD3) 81

สารบัญภาพ (ต่อ)

หน้าที่

รูปที่ 4-27 ผลการรับข้อมูลแบบที่ 3 (SD3) 82
รูปที่ 4-28 การส่งข้อมูลแบบที่ 4 (SD4) 82
รูปที่ 4-29 ผลการรับข้อมูลแบบที่ 4 (SD4) 82
รูปที่ 4-30 การส่งข้อมูลแบบที่ 5 (SC) 83
รูปที่ 4-31 การรับข้อมูลแบบที่ 5(SC) 83
รูปที่ 4-32 ตัวอย่างโครงสร้างอุปกรณ์รองอย่างง่าย 84
รูปที่ 4-33 ตัวอย่างสถานะการทำงานเริ่มต้น 85
รูปที่ 4-34 ตัวอย่างสถานะการทำงานของการตรวจสอบ 85
รูปที่ 4-35 ตัวอย่างสถานะการทำงานตั้งค่า Parameter 85
รูปที่ 4-36 ตัวอย่างสถานะการตรวจสอบ Configuration Data 86
รูปที่ 4-37 ตัวอย่างสถานะการแลกเปลี่ยนข้อมูล 86
รูปที่ 4-38 ตัวอย่างสถานะคำสั่งพิเศษ 86
รูปที่ 5-1 ภาพรวมการออกแบบ PRFIBUS-DP Slave 89

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันมีการขยายตัวของระบบอัตโนมัติเป็นอย่างมาก ส่งผลให้มีการนำเทคโนโลยีระบบบัส (Bus System) มาใช้เพื่อการสื่อสารข้อมูลทางด้านอุตสาหกรรม (Industrial communication) ที่มีความซับซ้อน ให้เกิดประสิทธิภาพมากที่สุด การทำงานในระบบการสื่อสารข้อมูลต้องการระบบที่มีสายบัส (Bus Line) เพียงเส้นเดียว (Single bus System) ในการเชื่อมต่ออุปกรณ์ทั้งหมด ให้เป็นเครือข่ายเดียวกัน โดยระบบบัสสามารถติดต่อได้ทุกจุดในฟิลด์ (Field) และทุกจุดการต่อในพื้นที่นั้นๆ ทำให้ระบบฟิลด์บัส (Fieldbus System) กลายมาเป็นระบบที่สามารถสื่อสารข้อมูลที่มีประสิทธิภาพสูง

โปรฟีบัส (PROFIBUS) เป็นการนำแนวความคิดของระบบฟิลด์บัสมาใช้ โดยไม่ขึ้นกับอุปกรณ์ของผู้ผลิตรายใด มีรูปแบบของสัญญาณการสื่อสารเป็นดิจิตอล นิยมใช้กันในการใช้งานด้านการผลิตแบบอัตโนมัติ (Automatic) โดยอุปกรณ์ที่ผลิตจากผู้ผลิตที่ต่างกันสามารถเชื่อมต่อเข้าด้วยกันภายในเครือข่ายไม่ต้องมีการแปลงมาตรฐาน และสามารถส่งข้อมูลได้ที่ความเร็วสูง ใช้ในงานที่สำคัญแบบการตรวจจับเวลา (Time-critical) งานที่มีการสื่อสารที่ซับซ้อนมาก ใช้ในการสื่อสารระหว่างอุปกรณ์และเครื่องมือวัดต่างๆ ในอุตสาหกรรม เช่น อุตสาหกรรมประกอบชิ้นส่วน กระบวนการผลิตอาหาร ระบบควบคุมอาคาร เป็นต้น อุปกรณ์ในเครือข่ายโปรฟีบัส แบ่งออกเป็น 2 ชนิดคือ อุปกรณ์หลัก (Master) และ อุปกรณ์รอง (Slave) การสื่อสารแลกเปลี่ยนข้อมูลของอุปกรณ์ในเครือข่ายกระทำผ่านการ์ดฟิวบัสอินพุท-เอาท์พุต (I/O Fieldbus Card)

โปรฟีบัส- ดีพี (PROFIBUS-Decentralized Periphery) เป็นหนึ่งในรูปแบบการสื่อสารของโปรฟีบัส ออกแบบมาเพื่อใช้ในระบบอุสาหกรรมอัตโนมัติ (Industrial Automation) ในการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์หลัก และ อุปกรณ์รอง เช่น อุปกรณ์ตรวจจับ (Sensors) และ อุปกรณ์สั่งงาน (Actuators) ทำงานได้ที่ความเร็วที่ 12 Mbps

เมื่อพิจาณาถึงระบบอุตสาหกรรมอัตโนมัติภายในประเทศ ในปัจจุบันผู้เชี่ยวชาญเกี่ยวกับโปรฟีบัส ยังคงมีน้อย และ ความสามารถในการผลิตอุปกรณ์ต่าง ๆ ที่ใช้ในเครื่องข่ายโปรฟีบัสนั้นยังไม่สามารถทำได้ ต้องนำเข้าจากต่างประเทศเป็นส่วนใหญ่ จึงเกิดแนวความคิดที่จะพัฒนาต้นแบบอุปกรณ์รองในการสื่อสารของระบบโปรฟีบัส-ดีพีขึ้น เพื่อเป็นแนวทางให้ผู้ที่สนใจสามารถนำไปพัฒนาต่อได้ และช่วยลดการนำเข้าสินค้าจากต่างประเทศได้อีกทางหนึ่ง นอกจากนี้ยังเป็นการรวบรวมข้อมูลเกี่ยวกับโปรฟีบัส-ดีพี เพื่อเป็นแหล่งค้นคว้าให้กับผู้ที่สนใจได้ศึกษา สามารถนำไปประยุกต์ใช้ให้เกิดประโยชน์กับประเทศชาติต่อไป

1.2 วัตถุประสงค์งานวิจัย

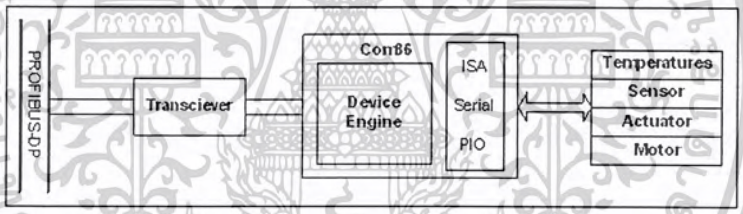
1. ถอดรหัส PROFIBUS-DP และสร้างความรู้ทางด้าน PROFIBUS DP
2. สร้าง Software Library สำหรับ Slave ที่ได้จากการถอดรหัส และแนวทางการสร้างอุปกรณ์ Slave สำหรับโรงงานอุตสาหกรรม
3. สร้างต้นแบบอุปกรณ์ Slave อย่างง่าย

1.3 ขอบเขตงานวิจัย

1. พัฒนา PROFIBUS DP Slave Library สำหรับอุปกรณ์ Slave ที่สามารถทำงานได้
2. พัฒนาแนวทางการสร้างของอุปกรณ์ Slave โดยนำ PROFIBUS DP Slave Library ไปใช้กับ Platform COM86 และ Real-time OS
3. พัฒนาต้นแบบอุปกรณ์ Slave อย่างง่าย

1.4 เป้าหมายงานวิจัย

1. Slave Device PROFIBUS-DP Platforms
2. บทสรุปจากการศึกษา ค้นคว้า การประยุกต์ใช้งาน มาตรฐาน PROFIBUS-DP



รูปที่ 1-1 Slave Device PROFIBUS-DP

1.5 วิธีการดำเนินงาน

1. ศึกษา ค้นคว้า รวบรวมข้อมูล เพื่อเข้าใจถึงโครงสร้าง รูปแบบการสื่อสาร ลำดับการทำงาน การให้บริการต่างๆ ตามมาตรฐานโปรฟิบัสดี-พี
1. วิเคราะห์หาบทสรุป กำหนดวัตถุประสงค์ ขอบเขตการทำงาน เป้าหมาย และรวบรวมข้อมูลที่สำคัญเพื่อใช้ในการออกแบบและพัฒนางานวิจัย
2. ออกแบบระบบ เป็นการออกแบบการทำงานของระบบในงานวิจัย เช่น ออกแบบโพรโตคอล-สแตก (Protocol Stack) ออกแบบลำดับการทำงานแต่ละระดับการสื่อสารของโพรโตคอล
3. พัฒนาระบบ เป็นการพัฒนาระบบให้เป็นไปตามขั้นตอนที่ได้ออกแบบไว้ โดยต้องเป็นไปตามมาตรฐานโปรฟิบัสดี-พี และพัฒนาให้มีการทำงานแบบเวลาจริง (Real-Time)
4. ทดสอบการทำงาน เป็นการนำผลงานที่ได้จากการพัฒนามาทดสอบ โดยต้องสามารถทำงานในระบบการสื่อสารตามมาตรฐานของโปรฟิบัสดี-พีได้

5. สรุปการทำงาน เป็นการนำเอาผลงานที่ได้จากการพัฒนา มาสรุปเปรียบเทียบการทำงานว่าตรงกับเป้าหมายที่ได้ตั้งไว้หรือไม่ พร้อมทั้งดำเนินการแก้ไข ปรับปรุงให้ตรงกับความต้องการที่ระบุไว้ในขั้นตอนแรก

บทที่ 2
มาตรฐานโปรฟิบัส

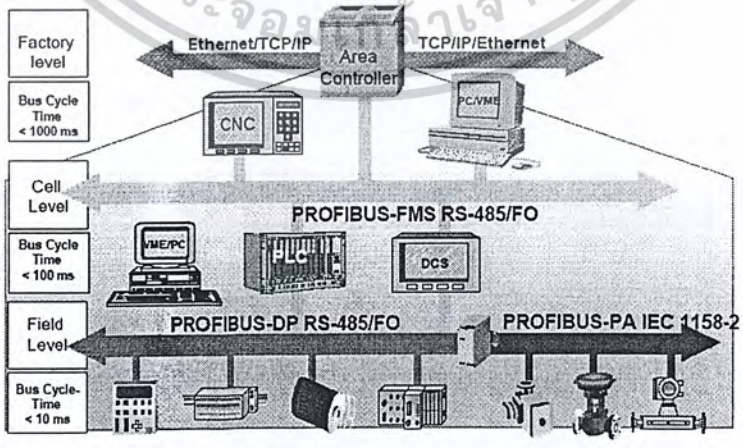
2.1 มาตรฐานโปรฟิบัส (PROFIBUS-Process Field Bus Standard)

โปรฟิบัสเป็นมาตรฐานและวิธีการสื่อสารที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่าง ระบบอัตโนมัติกับอุปกรณ์ในภาคสนาม (Distributed field devices) มีการกำหนดเทคนิควิธีและลักษณะการทำงานของระบบบัสอนุกรมที่ใช้ในอุตสาหกรรม

2.1.1 โครงสร้างการสื่อสาร

โครงสร้างการสื่อสารของระบบอุตสาหกรรมอัตโนมัติสามารถแบ่งได้เป็น 3 ระดับ คือ

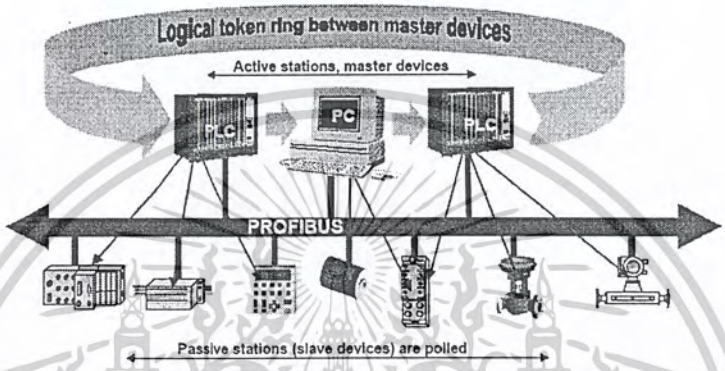
1. ระดับฟิลด์ (Field level) เป็นอุปกรณ์ที่อยู่ ณ จุดต่างๆ เช่น โมดูลอินพุท-เอาพุท (Module I/O), อุปกรณ์วิเคราะห์ (Analysis Device) และการแลกเปลี่ยนข้อมูลแบบวนรอบ (Cyclic) เป็นเครือข่ายในระดับต่ำที่สุดใช้สื่อสารระหว่างอุปกรณ์สนาม (Field device) ที่ต้องทำงานแบบสัมพันธ์ เช่น อุปกรณ์ตรวจจับและ อุปกรณ์สั่งงาน จำนวนข้อมูลในการสื่อสารไม่มาก และต้องการความเร็วในการสื่อสารสูง หรือเป็นการสื่อสารแบบเวลาจริง
2. ระดับเซลล์ (Cell level) เป็นส่วนของอุปกรณ์ควบคุมของระบบ เช่น พีแอลซี (PLC - Programmable Logic Controller) เป็นเครือข่ายการสื่อสารที่ใช้รับ-ส่งข้อมูลระหว่างหน่วย (Unit) หรือใช้สื่อสารระหว่างพีแอลซี จำนวนข้อมูลในการสื่อสารปานกลาง และต้องการความเร็วในการสื่อสารสูง
3. ระดับโรงงาน (Factory level) เป็นเครือข่ายในระดับบนสุด ใช้เป็นเครือข่ายการสื่อสารเพื่อควบคุมการทำงานของระบบโดยรวมทั้งหมด รวมทั้งสามารถรวบรวม เรียงลำดับ และจัดเก็บข้อมูลจากเครือข่ายต่ำกว่า



รูปที่ 2-1 ระดับการสื่อสารของระบบอุสาหกรรมตามมาตรฐานโปรฟิบัส

ตระกูลของโปรฟิบัส สามารถแบ่งตามรูปแบบการสื่อสาร เป็น 3 ประเภท คือ

1. โปรฟิบัสดี-พี ใช้สื่อสารระหว่างส่วนควบคุมกลางกับอุปกรณ์อินพุท-เอาพุทที่ระดับฟิลด์
2. โปรฟิบัสเอฟ-เอ็ม-เอส (Fieldbus Massage System) ใช้สื่อสารระหว่าง PLC กับ PC และแลกเปลี่ยนข้อมูลที่ Cell level สำหรับการสื่อสารข้อมูลที่มีความซับซ้อน
3. โปรฟิบัสพี-เอ (Process Automation) เป็นส่วนขยายของโปรฟิบัสดี-พี โดยสามารถรวมอุปกรณ์ของโปรฟิบัสพี-เอ และ โปรฟิบัสดี-พีด้วยกันได้โดยการใช้อุปกรณ์แยกส่วน (Segment Coupler) ใช้ในการสื่อสารที่มีความเร็วสูงและระบบอัตโนมัติ และต้องการความน่าเชื่อถือ

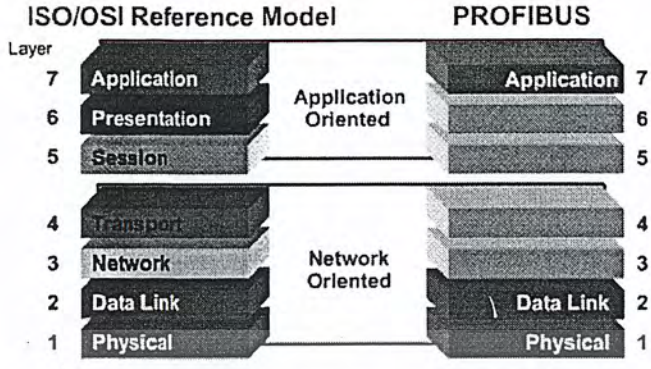


รูปที่ 2-2 การสื่อสารระหว่างอุปกรณ์ตามมาตรฐานโปรฟิบัส

อุปกรณ์ในการสื่อสารตามมาตรฐานของโปรฟิบัสแบ่งออกเป็น 2 ประเภท คือ

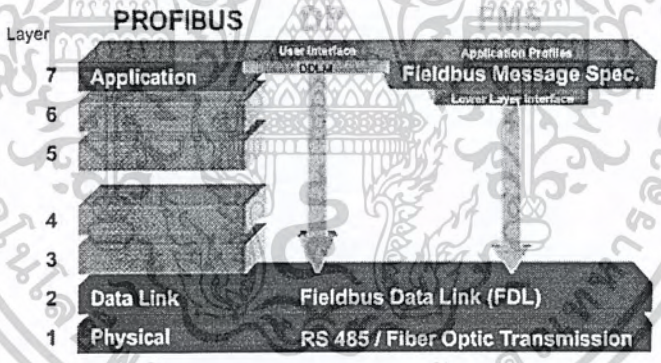
1. อุปกรณ์หลัก (Master) เป็นผู้กำหนดข้อมูลการสื่อสารบนโปรฟิบัส โดยจะส่งข้อความที่ปราศจากการกระตุ้นจากภายนอก เนื่องจากอุปกรณ์หลักเป็นผู้ถือครองบัสจึงสามารถเรียกได้อีกชื่อหนึ่งว่า สถานีกระตุ้น (Active stations) แบ่งได้ออกเป็น 2 ชนิด คือ ชนิดที่1 (Class 1) ทำหน้าที่ในการควบคุมการทำงานของอุปกรณ์รองภายในระบบ เช่น พีแอลซี และชนิดที่ 2 (Class 2) ทำหน้าที่ในการกำหนดค่าเริ่มต้นต่างๆ ให้กับระบบ เช่น การตั้งค่าข้อมูล (Configuration Data) ตัวอย่างเช่น พีซี (PC-Personal Computer) โดยอุปกรณ์หลักทั้ง 2 ชนิดนี้ จะมีอำนาจในการครอบครองบัสตามช่วงเวลาที่กำหนด หลังจากผ่านช่วงเวลาดังกล่าวจะส่งอำนาจการครอบครองให้กับอุปกรณ์หลักตัวถัดไปที่อยู่บนบัส โดยจะสื่อสารกันผ่านกระบวนการโทเค้น (Token Passing)
2. อุปกรณ์รอง (Slave) เรียกอีกอย่างหนึ่งว่า สถานีถูกกระทำ (Passive Station) เนื่องจากไม่มีอำนาจในการถือครองบัส มีความสามารถเพียงรับ-ส่งข้อมูลจากการร้องขอของอุปกรณ์หลัก ได้แก่ อุปกรณ์อินพุท-เอาพุทต่างๆ โดยที่อุปกรณ์หลักจะเป็นผู้ส่งข้อมูลเกี่ยวกับข้อกำหนดต่างๆของรูปแบบการสื่อสาร เพื่อเป็นข้อตกลงที่ใช้ระหว่างร่วมกันในการแลกเปลี่ยนข้อมูล โดยอุปกรณ์หลักจะวนมาสอบถามข้อมูลของอุปกรณ์รองทุกตัวที่อยู่ในระบบอยู่ตลอดเวลา

2.1.2 สถาปัตยกรรมโพรโตคอล



รูปที่ 2-3 ความสัมพันธ์โครงสร้างโอเอสไอ และ มาตรฐานโปรฟิบัส

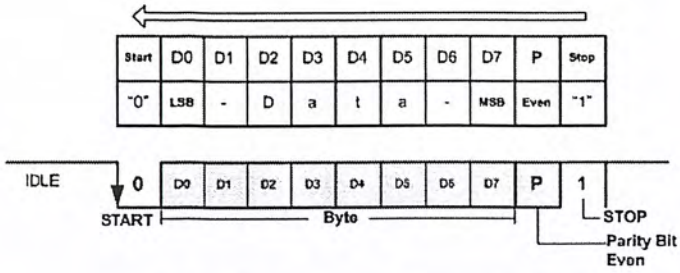
สถาปัตยกรรมโพรโตคอลของมาตรฐานโปรฟิบัสตั้งอยู่บนพื้นฐานของ โครงสร้างโอเอสไอ (OSI-Open System Interconnection Seven Layer Model) แต่ในมาตรฐานโปรฟิสบัสจะใช้เพียง 3 เลเยอร์ (Layer) เท่านั้น คือ Physical Layer, Data Link Layer และ Application Layer ในส่วนของโปรฟิบัสดี-พีจะมีการใช้งานเพียงเลเยอร์ 1 และ 2 เท่านั้น



รูปที่ 2-4 โครงสร้างมาตรฐานการสื่อสารของโปรฟิบัส

2.1.2.1 Physical Layer

สำหรับชั้นนี้ เป็นการกำหนดลักษณะทางกายภาย คุณสมบัติรูปแบบของสัญญาณ ซึ่งเทคโนโลยีที่ใช้ส่งข้อมูลของโปรฟิบัส โดยใช้กระบวนการส่งข้อมูลมาตรฐาน RS-485 ทำงานในรูปแบบสลับการส่งข้อมูล (Semi-Duplex or Half-Duplex) ส่งข้อมูลแบบไม่พร้อมกัน (Asynchronous) และใช้ช่องว่างระหว่างการส่งข้อมูล (Gap-free) ในการส่งแบบพร้อมกัน (Synchronous) จังหวะการทำงาน การส่งข้อมูลเป็นแบบครั้งละหนึ่งตัวอักษร (Character Frame) ประกอบด้วย 11 บิต เข้ารหัสแบบ รูปแบบของสัญญาณจาก 0 ไป 1 ไม่มีการเปลี่ยนแปลงขณะที่ส่ง (NRZ -Non Return to Zero)



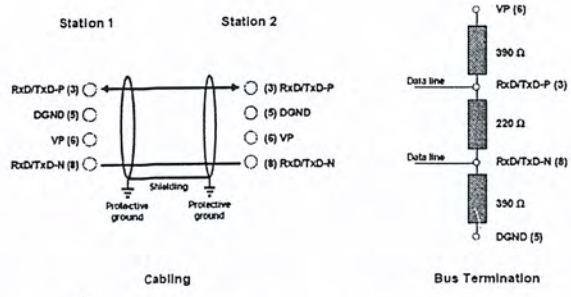
รูปที่ 2-5 รูปแบบเฟรมข้อมูลโปรฟิบัส

โดยจะมีบิตเริ่มต้น (Start Bit) เป็น 0 ตามด้วยข้อมูล 1 ไบต์ บิตตรวจสอบ (Parity Even) และ บิตสิ้นสุด (Stop Bit) เป็น 1 โดยการส่งข้อมูลจะส่งบิตต่ำไปก่อน (LSB-Least Significant Bit)

Pin No.	Signal Name	Designation
1	SHIELD	Shield or function ground
2	M24	Ground of the 24 V output voltage (Auxiliary power)
3	**RxD/TxD-P**	**Receiving/Sending-data-Plus B line**
4	CNTR-P	Signal for direction control P
5	**DGND**	**Data reference potential (ground)**
6	**VP**	**Supply Voltage-Plus**
7	P24	24 V Plus of the 24 V output voltage (Auxiliary power)
8	**RxD/TxD-N**	**Receiving/Sending-data-Plus A line**
9	CNTR-N	Signal for direction control N

ตารางที่ 2-1 รายละเอียดขาของดีบี-9 (DB-9)

มาตรฐานโปรฟิบัสใช้คอนเน็กเตอร์แบบชนิดดี (D-Type-DB9) เชื่อมต่อระหว่างสถานีเข้ากับสายของบัส สำหรับสถานีจะมี Socketซึ่งจะถูกต่อกับเข้าสายบัสด้วย Connecter ที่จุดปลายทั้งสองข้างจะต้องมีการปิดสาย ด้วยส่วนดึงสัญญาณไฟ (Pull-Down resister and Pull-Up resister)



รูปที่ 2-6 การเชื่อมต่อสายบัสตามมาตรฐานโปรฟิบัส

2.1.2.2 Data Link Layer

การทำงานในระดับนี้ เกี่ยวกับการควบคุมการใช้งานบัส (Bus Access Control) การตรวจสอบความถูกต้องของข้อมูล และการดูแลโพรโตคอลที่ใช้ส่งข้อมูลในระดับชั้นนี้ ซึ่งเรียกว่า เอฟ-ดี-แอล (FDL-Fieldbus Data Link) เป็นการบริการในเรื่องการส่ง (Send) ข้อมูล หรือการร้องขอ (Request) ผ่านบัส การส่งข้อมูลผ่านเครือข่ายโปรฟิบัส (Transmission of data) มีบริการอยู่ 4 แบบ คือ

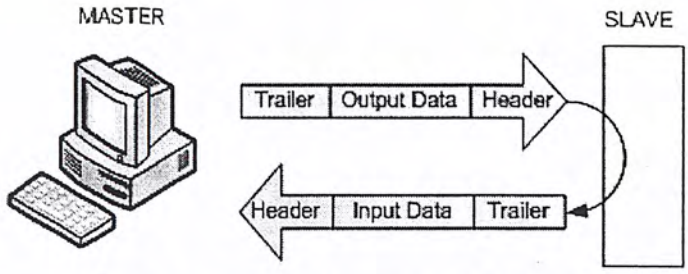
1. การส่งข้อมูลแบบมีการรับรอง (SDA-Send Data with Acknowledge) เป็นบริการส่งข้อมูลไปยังยังที่อยู่สถานีที่ต้องการ เมื่อได้รับข้อมูลประเภทนี้ ผู้รับต้องตอบการรับรองกลับมา
2. การรับ-ส่งข้อมูลแบบมีการรับรอง (SRD-Send and Request Data with acknowledge) เป็นการบริการรับ-ส่งข้อมูลไปยังที่อยู่สถานีที่ต้องการ หรือสถานีที่พร้อมจะรับข้อมูลโดยสถานีผู้รับจะตรวจสอบที่อยู่ ว่าเป็นของตนหรือไม่ ถ้าไม่ใช่ก็จะไม่รับ ถ้าใช่จะมีการตอบการรับรองกลับไปยังผู้ส่งทันที
3. การส่งข้อมูลแบบไม่มีการรับรอง (SDN-Send Data with NO Acknowledge) เป็นบริการส่งข้อมูลไปยังสถานีเดียวหรือหลายสถานีพร้อมกัน เมื่อได้รับข้อมูลประเภทนี้ ผู้รับไม่ต้องส่งรับรองกลับมา ใช้การสื่อสารแบบกระจาย (Broadcast) และ เฉพาะกลุ่ม (Multicast)
4. การรับ-ส่งข้อมูลที่มีการรับรองแบบวนรอบ (CRSD-Cyclic Send and Request Data with Acknowledge) เป็นการให้บริการรับ-ส่งข้อมูลโดยที่ขณะมีการส่งข้อมูลไปยังปลายทางจะมีการรับข้อมูลจากปลายทางกลับมาด้วย

ในแต่ละประเภทของโปรฟิบัสจะมีการให้บริการที่แตกต่างกัน สามารถแสดงรายละเอียดการให้บริการของแต่ละประเภทโปรฟิบัสได้ดังตารางที่ 2-2

Service	Function	DP	PA	FMS
SDA	Send Data with Acknowledge			x
SRD	Send and Request Data with Acknowledge	x	x	x
SDN	Send Data without Acknowledge	x	x	x
CRSD	Cyclic Send and Request Data with Acknowledge			x

ตารางที่ 2-2 การให้บริการของระดับชั้นที่ 2

รูปแบบเทเลแกรม (Telegram) ของโปรฟิบัสสามารถบรรจุข้อมูลได้ถึง 256 ไบต์ ต่อการส่ง 1 ข้อความ โดยจะแบ่งเป็นส่วนหัวของข้อมูล 11 ไบต์ ยกเว้นในกรณีที่ต้องการแลกเปลี่ยนข้อมูลจะใช้เพียง 9 ไบต์ และส่วนของข้อมูล 244 ไบต์ ดังนั้นโปรฟิบัสจึงใช้ประโยชน์ในการส่งข้อมูลขนาดใหญ่ นอกจากนี้ยังมีส่วนสถานะเปล่า (Idle State) อย่างน้อย 33T บิต ซึ่งจะต้องถูกทำงานก่อนเริ่มมีการรับ-ส่งเทเลแกรม



รูปที่ 2-7 รูปแบบการรับ-ส่งเฟรมของโปรฟิบัส

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU-Data Unit	FCS	ED
1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte	Var(1-244)	1byte	1byte

รูปที่ 2-8 รูปแบบเฟรมเทเลแกรมโปรฟิบัส

โครงสร้างของเทเลแกรมประกอบไปด้วยส่วนต่างๆ ซึ่งมีความหมายและรายละเอียดดังต่อไปนี้

1. SD (Start Delimiter) เป็นจุดเริ่มต้นของเทเลแกรม มีขนาด 1 ไบต์ แบ่งออกได้เป็น 5 ประเภท ตามตารางที่ 2-3

Telegram Format	Value	Data Field Length
SD1	10H	0 bytes (No Data Field)
SD2	68H	1 to 32 bytes (or up to 244)
SD3	A2H	8 bytes fixed
SD4	DCH	0 bytes (No Data Field)
SC	E5H	0 bytes (No Data Field), Short Acknowledge

ตารางที่ 2-3 ประเภทของจุดเริ่มต้นเทเลแกรม

ในแต่ละประเภทของจุดเริ่มต้นเทเลแกรม ทำให้มาตรฐานรูปแบบของข้อมูลที่ใช้ส่งภายในเครือข่ายแบ่งออกได้เป็น 5 รูปแบบคือ

- รูปแบบที่กำหนดความยาวของเทเลแกรมคงที่
- รูปแบบที่กำหนดความยาวของข้อมูลในเทเลแกรมคงที่
- รูปแบบที่เปลี่ยนแปลงได้ตามขนาดของข้อมูลในเทเลแกรม
- รูปแบบการตอบกลับการรับรองแบบสั้น
- รูปแบบเทเลแกรมโทเคน

SD1	DA	SA	FC	FCS	ED
10H	xx	xx	x	x	16H

รูปที่ 2-9 รูปแบบเทเลแกรมที่กำหนดความยาวคงที่

SD3	DA	SA	FC	DU	FCS	ED
A2H	xx	xx	x	x.....x	xx	16H

รูปที่ 2-10 รูปแบบเทเลแกรมที่กำหนดความยาวข้อมูลคงที่

SD2	LE	LEr	SD2	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	xx	xx	68H	xx	xx	x	3CH	3EH	x.....x	xx	16H

VAR LENGTH

รูปที่ 2-11 รูปแบบเทเลแกรมที่เปลี่ยนแปลงตามขนาดข้อมูล

SD3	DA	SA	ED
DCH	xx	xx	16H

รูปที่ 2-12 รูปแบบเทเลแกรมโทเคน

SC
E5H

รูปที่ 2-13 รูปแบบเทเลแกรมการตอบกลับแบบสั้น

2. LE & LEr (Length of Telegram) ส่วนเก็บความยาวของข้อมูลตั้งแต่ส่วนของ DA ถึง Data Unit มีขนาด 1 ไบต์ โดยค่า LEr จะเหมือนกับ LE ทั้งนี้เพื่อทำการสำรองป้องกันความผิดพลาด
3. DA & SA(Destination Address & Source Address) เป็นส่วนที่ใช้เก็บ ที่อยู่สถานีปลายทาง และที่อยู่สถานีต้นทาง โดยมีขนาดอย่างละ 1 ไบต์ โดยจะมีค่าอยู่ระหว่าง 0 – 127 โดยค่า 127 จะสงวนไว้สำหรับการกระจายข้อมูล (Broadcast)
4. FC (Function Code) เป็นส่วนเก็บข้อมูลควบคุมโดยจะกำหนดชนิดของเทเลแกรม ว่าเป็นชนิดการร้องขอ หรือ การตอบสนอง หรือ การตอบกลับ บ่งบอกชนิดของสถานี เป็นต้น
5. SSAP & DSAP (Source/Destination Service Access Point) เป็นส่วนที่ใช้เก็บชนิดของการให้บริการในการรับ-ส่งข้อมูล มีขนาดอย่างละ 1 ไบต์
6. DU (Data Unit) เป็นส่วนบรรจุเนื้อข้อมูลที่ใช้ในการสื่อสาร-แลกเปลี่ยนระหว่างอุปกรณ์ มีขนาด 1 - 224 ไบต์

7. FCS (Frame Check Sequence) เป็นส่วนตรวจสอบความผิดพลาดของเทเลแกรม มีขนาด 1 ไบต์ โดยจะคิดจากค่าผลรวมของ DA+SA+FC+DU mod 256
8. ED (End Delimiter) ส่วนเก็บจุดสิ้นสุดของเทเลแกรม มีขนาด 1 ไบต์ มีค่าคงที่คือ 16H

เมื่อพบความผิดพลาดของเทเลแกรม ในระดับชั้นที่ 2 จะทำการส่งอีกครั้งอย่างอัตโนมัติ โดยสามารถตั้งให้พยายามส่งได้สูงสุด 8 ครั้ง การตรวจสอบชนิดของความผิดพลาดสามารถตรวจสอบได้จาก

- รูปแบบของตัวอักษรที่ได้รับผิดพลาด อาจเกิดจากบิตที่ติดกัน (Parity)
- ความผิดพลาดที่เกิดจากโพรโตคอล
- ความผิดพลาดจากการกำหนดจุดเริ่ม และจุด จบ (Start and End)
- ความผิดพลาดที่เกิดจากตรวจสอบเฟรม (Frame Check byte)
- ความผิดพลาดที่เกิดจาก ความยาวของ เทเลแกรม

2.1.2.3 Application Layer

โปรฟีบัส-ดีพี จะมีการทำงานของ ดีดีแอลเอ็ม (DDLM-Direct Data Link Mapper) เป็นตัวกลางในการเชื่อต่อระหว่างระดับชั้นที่ 2 กับผู้ใช้ ส่วนโปรฟีบัส-เอฟเอ็มเอส ในชั้นนี้จะเป็นการสื่อสารแบบสากล ที่สามารถกำหนดรูปแบบการสื่อสารได้ตามต้องการ โดยอาศัยการทำงานของ เอฟเอ็มเอส (FMS-Fieldbus Message Specification) และ แอลแอลไอ (LLI-Lower Layer Interface) ซึ่งเป็นส่วนที่แสดงความสัมพันธ์ระหว่างการทำงานของแอพลิเคชั่นและการส่งผ่านข้อมูล

2.2 โปรฟีบัสดี-พี (PROFIBUS-DP-Process Field Bus Decentralized Periphery)

เป้าหมายหลักในการสื่อสารระบบอัตโนมัติ คือ การแลกเปลี่ยนระหว่างอุปกรณ์กับระบบควบคุมที่มีความรวดเร็ว และถูกต้อง จากรูปแบบการสื่อสารของโปรฟีบัส โปรฟีบัส- ดีพี ออกแบบมาเพื่อใช้ในระบบอุสาหกรรมอัตโนมัติในการแลกเปลี่ยนข้อมูลด้วยความเร็วสูงระหว่างอุปกรณ์ควบคุม และ อุปกรณ์ฟิวส์ เนื่องจากเป็นมาตรฐานการสื่อสาร (Protocol) ที่เปิดกว้าง อิสระจากผู้ค้า ติดตั้งแล้วสามารถใช้งานได้ โปรฟีบัสดี-พีจึงได้รับความนิยมในโรงงานอุตสาหกรรม

2.2.1 ลักษณะพื้นฐาน

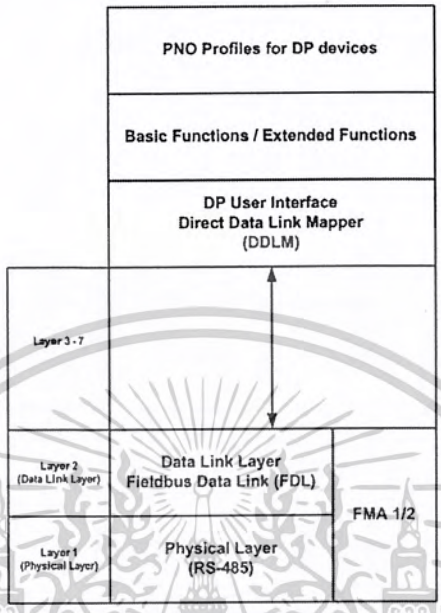
โปรฟีบัสดี-พี เป็นรูปแบบหนึ่งของมาตรฐานโปรฟีบัส มีความสามารถในการรับส่งข้อมูลได้ที่ความเร็ว 12 Mbit/sec สามารถรับส่งข้อมูลได้จำนวน 244 ไบต์ต่อการส่งข้อความ 1 ครั้ง นอกจากนี้สามารถเชื่อมต่อสถานีได้ถึง 126 สถานีภายในบัสเส้นเดียวกัน และ สามารถเชื่อมต่อได้ถึง 32 สถานีในแต่ละส่วนของบัส (Segment)

อุปกรณ์ภายในเครือข่ายของโปรฟีบัสดี-พี แบ่งได้ออกเป็น 3 ชนิดคือ

1. อุปกรณ์หลักชนิดที่ 1 (DP-Master Class1) เป็นส่วนควบคุมกลางในการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์รอง และทำหน้าที่ในการจัดการโทเคนระหว่างอุปกรณ์หลัก
2. อุปกรณ์หลักชนิดที่ 2 (DP-Mater Class 2) เป็นส่วนที่จัดการเกี่ยวกับการตรวจสอบ หรือ การจัดการ อุปกรณ์รอง โดยสามารถควบคุมอุปกณ์รองได้เพียงตัวเดียวในช่วงเวลาหนึ่งเท่านั้น ไม่สามารถที่จะเข้าไปจัดการภายใน อุปกรณ์รองได้

3. อุปกรณ์รอง (DP-Slave) เป็นส่วนที่ตอบสนองต่อการทำงานจากการร้องขอของอุปกณ์หลัก และการตอบกลับการรับรอง โดยจะไม่สามารถถือครองบัสได้

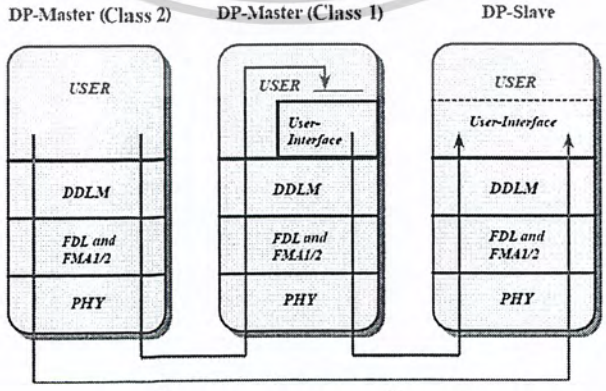
2.2.2 สถาปัตยกรรมโพรโตคอล



รูปที่ 2-14 สถาปัตยกรรมโปรฟิบัสดี-พี

สถาปัตยกรรมของโปรฟิบัสดี-พี ในระดับชั้นที่ 1 มีรูปแบบเป็นไปตามมาตรฐานทั่วไปของโปรฟิบัส ส่วนในระดับชั้นที่ 2 โปรฟิบัสดี-พี จะประกอบไปด้วยส่วนสำคัญ 2 ส่วน คือ

1. FDL (FDL-Fieldbus Data Link) เป็นส่วนจัดการในเรื่องของความถูกต้องของข้อมูล เป็นตัวกลางระหว่างเลเอยร์บน กับ เลเยอร์ 1
2. FMA1/2 (Fieldbus Management) เป็นส่วนควบคุมการทำงานระหว่างเลเยอร์ 1 และ 2 โดยจะรับรูปแบบการควบคุมมาจากเลเยอร์ที่อยู่สูงกว่า

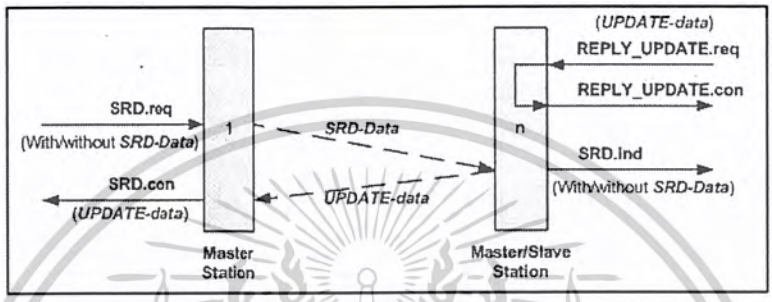


รูปที่ 2-15 โครงสร้างการสื่อสารโปรฟิบัสดี-พี

โครงสร้างการสื่อสารของโปรฟิบัสดี-พี การทำงานของแต่ละอุปกรณ์ในระบบจะเริ่มจากส่วนผู้ใช้ ผ่าน DDLL FMA และ FDL ก่อนที่จะส่งผ่านข้อมูลไปยังปลายทาง และทางกลับกัน ส่วนของการรับข้อมูลจะกระทำตั้งแต่ระดับล่างขึ้นมายังผู้ใช้

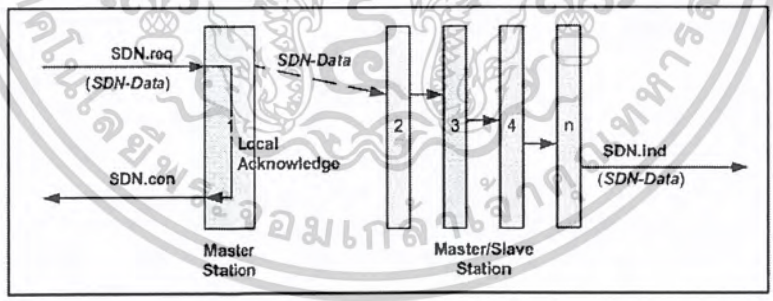
นอกจากนี้ในโปรฟิบัสดี-พี มีการให้บริการในการรับส่งข้อมูลอยู่ 2 แบบ คือ

1. การรับ-ส่งข้อมูลแบบมีการรับรอง (SRD) อุปกรณ์หลักจะส่งข้อมูลไปยังอุปกรณ์รอง หลังจากที่อุปกรณ์รองได้รับข้อมูลแล้ว จะทำการตอบกลับในช่วงเวลาที่กำหนด ในช่วงเวลาเดียวกันขณะที่ส่งข้อมูล อุปกรณ์หลักสามารถทำการรับข้อมูลจากอุปกรณ์รองได้ด้วย



รูปที่ 2-16 การรับ-ส่งข้อมูลแบบมีการรับรอง

2. การส่งข้อมูลแบบไม่มีการรับรอง (SDN) การบริการนี้ใช้ในการส่งข้อมูลไปยังกลุ่มของอุปกรณ์รอง หรืออุปกรณ์รองทั้งหมด พร้อมๆ กัน โดยที่อุปกรณ์รองจะไม่มีการตอบกลับเมื่อได้รับข้อมูล



รูปที่ 2-17 การส่งข้อมูลแบบไม่มีการรับรอง

ภายในโครงสร้างเทเลแกรม โปรฟิบัสดี-พีจะมีข้อกำหนดที่ใช้เฉพาะในการสื่อสารอยู่ 2 ส่วนคือ

1. FC ในการควบคุมเทเลแกรมของโปรฟิบัสดี-พี มีรูปแบบการใช้งานดังตารางที่ 2-4 และ 2-5
2. SSAP และ DSAP ประเภทของการให้บริการในการส่งข้อมูลของโปรฟิบัสดี-พีจะมีการใช้งานอยู่ระหว่าง 54 ถึง 62 รวมถึงค่าปกติด้วย ตามตารางที่ 2-6 โดยเงื่อนไขการใช้งาน คือ ประเภท 56-57 จะไม่สามารถใช้งานได้จนกว่า อุปกรณ์รองจะเข้าสู่สถานะในการแลกเปลี่ยนข้อมูล ส่วนค่าอื่นๆ สามารถใช้ได้ตามปกติ

FC Code	Function (The MSB in FC = 1)
4	SDN low (Send Data with No acknowledge)
6	SDN high (Send Data with No acknowledge)
7	Reserved/Request Diagnostic Data
9	Request FDL Status With Reply
12	SRD low (Send and Request Data with acknowledge)
13	SRD high (Send and Request Data with acknowledge)
14	Request ID With Reply
15	Request LSAP Status With Reply

ตารางที่ 2-4 รูปแบบการใช้งานการควบคุมเทเลแกรมแบบที่ 1

FC Code	Function (The MSB in FC = 0)
0	ACK Positive
1	ACK Negative (FDL/FMA1/2 user error UE, interface error)
2	ACK Negative (No resource/memory space for Send Data (RR).
3	ACK Negative (No service activated (RS), SAP not activated).
8	Response FDL/FMA ½ Data low and Send Data OK)
9	ACK Negative (No response FDL/FMA1/2 Data & Send Data OK).
10	Response FDL Data High and Send Data OK.
12	Response FDL Data Low, No resource for Send Data.
13	Response FDL Data High Resource For Send Data.

ตารางที่ 2-5 รูปแบบการใช้งานการควบคุมเทเลแกรมแบบที่ 2

SAP	SERVICE
Default SAP=0	Cyclical Data Exchange (Write_Read_Data)
SAP54	Master-to-Master SAP (M-M Communication)
SAP55	Change Station Address (Set_Slave_Add)
SAP56	Read Inputs (Rd_Inp)
SAP57	Read Outputs (Rd_Outp)
SAP58	Control Commands to a DP Slave (Global_Control)
SAP59	Read Configuration Data (Get_Cfg)
SAP60	Read Diagnostic Data (Slave_Diagnosis)
SAP61	Send Parameterization Data (Set_Prm)
SAP62	Check Configuration Data (Chk_Cfg)

ตารางที่ 2-6 ประเภทการให้บริการโปรฟิบัสดี-พี

การทำงานของโปรฟิบัสดี-พี จะแบ่งตามประเภทของการให้บริการ ทำให้ในแต่ละอุปกรณ์จำเป็นต้องมีการทำงานพื้นฐานดังตารางที่ 2-7

Functions	DP Slave	Class 1 Master
Data_Exchange	√	√
Rd_Inp	√	
Rd_Outp	√	
Slave_Diag	√	√
Set_Prm	√	√
Chk_Cfg	√	√
Get_Cfg	√	
Global_Control	√	√
Set_Slave_Address	√ (Optional)	
Get_Master_Diag		√
Start_Seq		√ (Optional)
Download		√ (Optional)
Upload		√ (Optional)
End_Seq		√ (Optional)
Act_Para_Brct		√ (Optional)
Act_Para		√ (Optional)

ตารางที่ 2-7 การทำงานพื้นฐานของโปรฟิบัสดี-พี

ในส่วนของการตอบกลับของข้อมูลจะมีข้อมูลเกี่ยวกับสถานะของตัวแปรที่อธิบายถึงความสำเร็จจากการร้องขอ โดยสามารถแสดงเงื่อนไขที่เป็นไปได้ตามตารางที่ 2-8

Status Value	Description
OK	Acknowledgement positive.
IV	Invalid parameters in request.
NO	Service in this state not possible.
DS	Local FDL/PHY entity is not in logical token ring or disconnected from line.
NA	Negative Acknowledge – No reaction from remote station.
RS	Service or remote-address at remote-LSAP or remote-LSAT not activated: Remote station is no DP-Station, Remote station is not ready for these functions, Remote station is associated with another requestor, Optional service not available.
RR	Resources of the remote-FDL entity not sufficient or not available.
UE	Remote-DDLM/FDL interface error.
NR	No response data.
TO	Function timeout expired.
FE	Format error in request frame.
RE	Format error in response frame.
LE	Data block length too large (Upload/Download)
NI	Function not implemented.
EA	Area too large (Upload/Download)
AD	Access denied.
IP	Invalid parameter.
SC	Sequence conflict.
SE	Sequence error.
NE	Area Non-existent.
DI	Data incomplete.
NC	Master parameter set not compatible.

ตารางที่ 2-8 สถานะข้อมูลตอบกลับ

2.3 อุปกรณ์รองดี-พี (DP-Slave)

ในส่วนของงานวิจัยชิ้นนี้จะเป็นการพัฒนาการ์ดฟิวบัสของอุปกรณ์รองของโปรฟิบัสดี-พี ดังนั้นจึงต้องทำความเข้าใจในการทำงานของอุปกณ์รองของโปรฟิบัสดี-พี ในส่วนของอุปกรณ์หลักซึ่งมีการทำงานที่ซับซ้อนจึงไม่ขอกล่าวถึงรายละเอียด

2.3.1 สถานะการทำงาน

อุปกรณ์รองของโปรฟิบัสดี-พี จะต้องมีสถานะการทำงานอยู่ 3 ขั้นตอนหลักก่อนที่จะเริ่มมีการแรกเปลี่ยนข้อมูลกับอุปกรณ์หลัก โดยในแต่ละสถานะจะมีการทำงานที่แตกต่างกัน รวมถึงรูปแบบของเทเลแกรมที่ใช้ในการสื่อสารจะแตกต่างกันด้วย

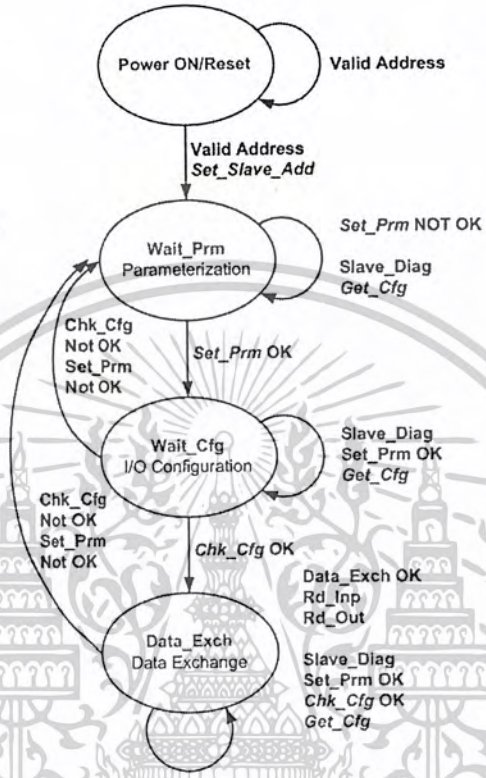
2.3.1.1 Initial Power ON/Reset

ในสถานะเริ่มต้นหลังจากเริ่มเปิดอุปกรณ์ให้เริ่มทำงาน ในตัวอุปกณ์รองจะจัดระบบของตนเองอัตโนมัติ เช่น การตรวจสอบความเร็วของสายบัส และปรับความเร็วของตัวอุปกรณ์เองให้สามารถสื่อสารได้ หลังจากนั้นจะทำการกำหนดที่อยู่ให้กับตัวอุปกรณ์ จุดประสงค์หลักของการทำงานในสถานะนี้คืออุปกรณ์รองจำเป็นต้องมีค่าที่อยู่ของตัวเองที่ถูกต้อง คือ ค่าระหว่าง 0-125 แต่ถ้าตัวอุปกรณ์มีค่าที่อยู่ 126 จำเป็นต้องให้อุปกรณ์หลักชนิดที่ 2 เป็นผู้จัดการที่อยู่ให้ใหม่ โดยจะใช้ประเภทการบริการที่มีค่าเป็น 55 (SAP55)

2.3.1.2 Parameterization

ถ้าที่อยู่ของอุปกรณ์รองไม่ถูกต้อง (เป็น 126) จำเป็นต้องมีการตั้งค่าให้ถูกต้องก่อนที่จะเข้าสู่สถานะนี้ หลังจากที่อุปกรณ์รองมีค่าที่อยู่ที่ถูกต้องแล้ว จะเข้าสู่ขั้นตอนการรอเทเลแกรม เพื่อใช้ในตั้ง

ค่าตัวแปรในการกำหนดรูปแบบการทำงานจากอุปกรณ์หลัก (Parameter Telegram) โดยจะใช้ประเภทการบริการที่มีค่าเป็น 61 (SAP61) ในขณะนี้อุปกรณ์รอเทลเลแกรมตั้งค่าตัวแปร จะปฏิเสธทุกเทเลแกรมที่เข้ามายกเว้นเทเลแกรมตรวจสอบ (Diagnostic Telegram) และ เทเลแกรมองค์ประกอบระบบ (Configuration Telegram)



รูปที่ 2-18 สถานะการทำงานของอุปกรณ์รอง

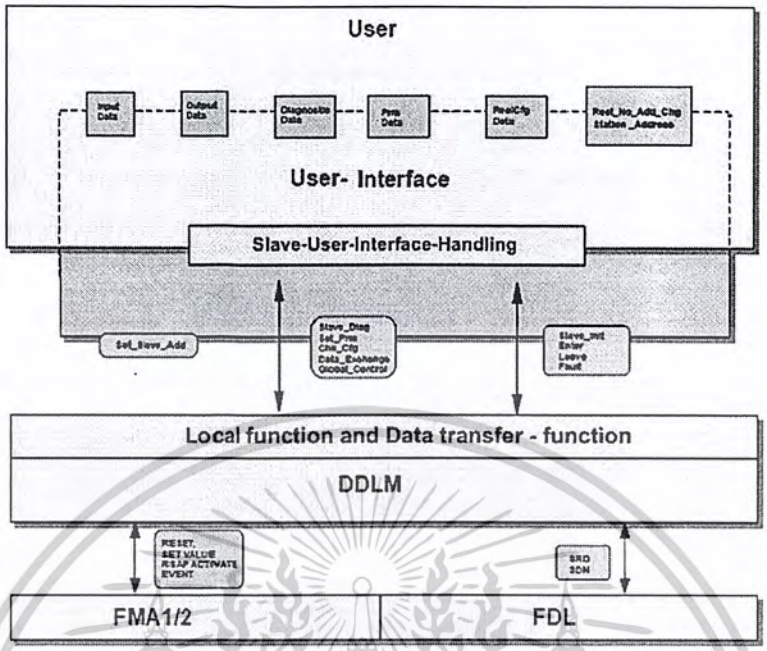
2.3.1.3 I/O Configuration

หลังจากผ่านสถานการณ์ตั้งค่าตัวแปรแล้ว อุปกรณ์รองจะเข้าสู่สถานการณ์ตรวจสอบองค์ประกอบของระบบ โดยที่จะทำการกำหนดจำนวนของอินพุท-เอาพุท ที่ใช้ในการแลกเปลี่ยนของแต่ละรอบการสื่อสาร โดยอุปกรณ์รองจะนำค่าที่ได้จากเทเลแกรมมาตรวจสอบกับค่าที่ตัวเองได้กำหนดไว้ หากค่าไม่ถูกต้องอุปกรณ์หลักจะทำการสอบถามข้อมูลที่ถูกต้องจากอุปกรณ์รอง

2.3.1.4 Data Exchange State

ช่วงนี้อุปกรณ์หลักจะเริ่มทำการแรกเปลี่ยนข้อมูลระหว่างอุปกรณ์รอง โดยอุปกรณ์จะทำงานอัตโนมัติในการตรวจสอบ ข้อมูลที่ส่งผ่าน ข้อมูลที่รับเข้ามา ข้อความตอบกลับ โดยจำนวนของอินพุท-เอาพุทที่ใช้ในการรับส่งจะถูกกำหนดไว้ในองค์ประกอบของระบบ ในการส่งข้อมูล อุปกรณ์หลักจะส่งข้อมูลไปให้อุปกรณ์รอง และอุปกรณ์รองมีการตอบกลับข้อมูลแบบสั้น ในสถานะนี้จะไม่มีการกำหนดประเภทของการให้บริการ ถ้าจำเป็นต้องมีการกำหนดรูปแบบองค์ประกอบใหม่ อุปกรณ์รองจำเป็นต้องเข้าไปอยู่ในสถานะรอการตั้งค่าตัวแปรใหม่

2.3.2 โครงสร้างการทำงาน



รูปที่ 2-19 โครงสร้างการทำงานของอุปกรณ์รอง

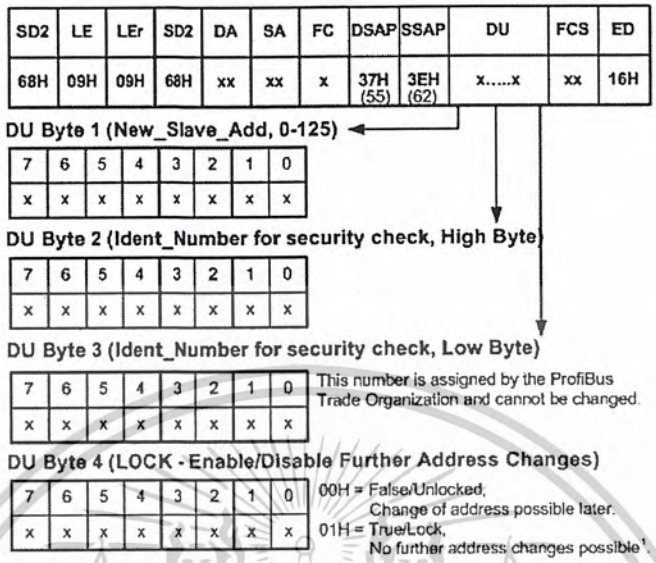
เมื่อพิจารณาถึงโครงสร้างการทำงานตามรูปแบบของสถาปัตยกรรมอุปกรณ์รอง จะเห็นได้ว่า ในแต่ละ การเชื่อมต่อระหว่างชั้นการสื่อสารจะมีรูปแบบในการเรียกใช้งานที่แตกต่างกันไปตามการใช้งาน สามารถแสดงรายละเอียดในแต่ละประเภทการของการเชื่อมต่อระหว่างชั้นการสื่อสารได้ดังนี้

2.3.2.1 DDLM-User Interface

ในส่วนนี้จะมีโครงสร้างการทำงานที่แบ่งออกเป็น 2 ส่วน คือ ส่วนที่มีการเรียกใช้การบริการ ในส่วนนี้จะมีการกำหนดโครงสร้างของเทเลแกรมที่ต้องใช้ส่ง เนื่องจากจำเป็นต้องติดต่อกับอุปรณ์อื่น และอีกส่วนหนึ่งคือ ดารติดต่อภายในโครงสร้างของระบบเอง ไม่จำเป็นต้องมีสร้างเทเลแกรมในการติดต่อ ในการเรียกใช้การบริการประเภทต่างๆ จะเรียกใช้ตามลักษณะของงาน โดยในแต่ละประเภทจะมีรูปแบบการใช้งานแตกต่างกัน ดังนี้

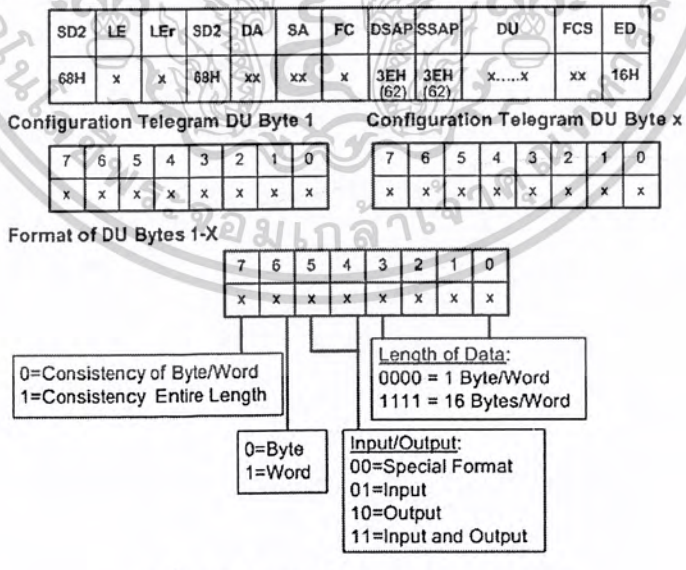
1. Slave_Int เป็นส่วนในการทำงานแรกเริ่มของอุปกณ์รอง โดยจะทำการกำหนดค่าเริ่มต้นต่างๆ ให้กับอุปกรณ์รองใหม่ทั้งหมด
2. Salve_Enter เป็นการทำงานในส่วนของการตั้งค่าเริ่มในการแลกเปลี่ยนข้อมูล
3. Slave_Fault เกี่ยวข้องกับความผิดพลาดที่เกิดขึ้นในระหว่างการเปลี่ยนแปลงส่งผ่านข้อมูล
4. Slave_Set_minTsdr เกี่ยวข้องกับการกำหนดค่าของเวลาในการตอบกลับเทเลแกรม
5. Set_Slave_Add จะใช้ประเภทการบริการ 55 โดยจะถูกใช้โดยอุปกรณ์ชนิดที่ 2 ในการเปลี่ยนแปลงที่อยู่ของอุปกรณ์รอง ในกรณีที่ไม่สามารถเปลี่ยนแปลงได้จากภายนอก เพื่อความปลอดภัย จะมีการตรวจสอบหมายเลขประจำของอุปกรณ์รองด้วย (Ident_Number) ถ้า

หมายเลขประจำเครื่องไม่ตรงกันจะไม่สามารถเปลี่ยนแปลงค่าที่อยู่ได้ สามารถแสดงโครงสร้างเทเลแกรมได้ดังนี้



รูปที่ 2-20 โครงสร้างเทเลแกรม Set_Slave_Add

6. Chk_Cfg ใช้บริการประเภท 62 เป็นการตรวจสอบองค์ประกอบของระบบของอุปกรณ์รอง ถ้าได้ค่าที่ไม่ตรงกัน อุปกรณ์รองจะส่งรายงานความผิดพลาดกลับไป โดยที่อุปกรณ์หลักจะตรวจสอบหาข้อมูลของอุปกรณ์รองอยู่เสมอ สามารถแสดงโครงสร้างเทเลแกรมได้ดังนี้



รูปที่ 2-21 โครงสร้างเทเลแกรม Chk_Cfg

7. Set_Prm ใช้การบริการประเภท 61 ใช้ในการตั้งค่าตัวแปรของอุปกรณ์รองในการเริ่มต้นหลังจากการเริ่มต้นการทำงานก่อนเริ่มการแลกเปลี่ยนข้อมูล โดยมาตรฐานแล้วจะมีโครงสร้างเทเลแกรมจำนวน 7 ไบต์ ดังนี้

SD2	LE	LEr	SD2	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	x	x	68H	8x	8x	x	3DH (61)	3EH (62)	x.....x	xx	16H

Set_Prm Parameterization Telegram DU Byte 1 – Station_status

7	6	5	4	3	2	1	0
Lock_Req	Unlock_Req	Sync_Req	Freeze_Req	WD_On	Reserve – Set = 0		

WD_On (Watchdog On)
Set this bit to 1 to activate the watchdog control. Refer to the Watchdog section for more information.
Freeze_Req (Freeze Mode Request) If this bit is set, the slave will operate in the Freeze Mode as soon as the Global_Control function is received. If a slave does not support the Freeze control, it sets the Diag.Not_Supported bit within the diagnostic information.
Sync_Req (Sync Mode Request) If this bit is set, the slave will operate in Sync Mode as soon as the Global_Control function is received. If a slave does not support the Sync control, it sets the Diag.Not_Supported bit within the diagnostic information.

Lock Bit 7	Unlock Bit 6	Description
0	0	The Min TSDR parameter may be changed. All other parameters remain unchanged.
0	1	DP Slave is unlocked/released for other masters.
1	0	DP Slave locked for other masters. All parameters are accepted and can be carried over (except min TSDR = 0).
1	1	DP Slave is unlocked/released for other masters.

Set_Prm Telegram DU Byte 2 – WD_Fact_1, Range 1 to 255

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

WD_Fact_1

Set_Prm Telegram DU Byte 3 – WD_Fact_2, Range 1 to 255

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

WD_Fact_2

The watchdog is switched on or off via the WD_On bit of DU byte 1.
Bytes 2 & 3 are factors used for setting the watchdog control (TWD) time.
The watchdog time is calculated between 10ms and 650 seconds as follows:
TWD =10ms * WD_Fact_1 * WD_Fact_2
The watchdog control causes the slave outputs to go to a failsafe state if the master fails to communicate with the slave before this time expires.

Set_Prm Telegram DU Byte 4 – Min T_{SDR} , Range 11 to 255 Tbit

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

Min T_{SDR}: Minimum time in Tbit (bit time) after which a slave is allowed to answer. This value must be less than Max T_{SDR}. 11 Tbits are specified permanently in the standard.

Byte 4 sets the minimum T_{SDR} time (in bit time, 11-255) a slave will wait before it is allowed to send a response to the master. If 00H is specified, the previous or default value is used.

Set_Prm Telegram DU Byte 6 (Ident_Number, Low Byte)

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

Set_Prm Telegram DU Byte 7 (Group_Ident)

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

Group_Ident

Each bit represents a unique group.
ProfiBus DP supports multi-cast messaging via a **Global_Control** telegram function directed to a specific group of connected slaves identified via this group number.
*Note that **Group_Ident** is only accepted if the **Lock_Req** bit is also set.

รูปที่ 2-22 โครงสร้างเทเลแกรม Set_Prm

8. Data_Exchange จะไม่มีการใช้บริการเนื่องจากจะมีส่วนหัวของข้อมูลเพียง 9 ไบต์ โดยค่าของบิตแรกในที่อยู่ของต้นทางและปลายทางจะมีค่าเป็น 1

SD	LE	LEr	SD	DA	SA	FC	DU	FCS	ED
68H	X	X	68H	8x	8x	X	X..	X	16H

รูปที่ 2-23 โครงสร้างเทเลแกรม Data_Exchange

9. Diag_Data ใช้ประเภทการบริการ 60 อุปกรณ์ชนิดที่ 1 จะเป็นผู้ใช้ในการร้องขอจากอุปกรณ์รอง โดยขณะเริ่มต้นอุปกรณ์หลักจะร้องขอข้อมูลตรวจสอบจากอุปกรณ์รองก่อนที่จะทำการตั้งค่าตัวแปร และจะเรียกใช้อีกครั้งก่อนการแลกเปลี่ยนข้อมูล ตามมาตรฐานแล้วจะมีโครงสร้างเทเลแกรมทั้งหมด 6 ไบต์ ดังนี้

SD2	LE	LEr	SD2	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	x	x	68H	xx	xx	x	3EH (62)	3EH (60)	x....x	xx	16H

The diagnostic information of a DP slave consists of 6 bytes of standard diagnostic information, plus any user-diagnostic information (slave specific).

A set bit (1) in a position means the linked definition has occurred.
The parameter Status can be tested to indicate the success or failure of the Diag_Data function with possible values of: OK, DS, NA, RS, UE, NR, & RE.

Diag_Data Response Telegram DU Byte 1 – Station_status_1

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x

Bit	DIAGNOSTIC
0	Diag.Station_Non_Existent: Set to 1 by the master if slave cannot be reached over the line. Slave sets this bit to 0.
1	Diag.Station_Not_Ready: Set by slave if slave is not ready for data transfer.
2	Diag.Cfg_Fault: Set by slave if it detects a mismatch in config data.
3	Diag.Ext_Diag: Set by slave to indicate a diagnostic entry is in the slave-specific diagnostic area (see below).
4	Diag.Not_Supported: Set by slave if requested function/service is not supported.
5	Diag.Invalid_Slave_Response: Slave sets this bit to 0. Set to 1 by the master if it receives an implausible response from the slave.
6	Diag.Prm_Fault: Set by slave if last parameter frame was faulty (wrong parameterization , bad length, bad ident_number, etc.).
7	Diag.Master_Lock: Set by a class 1 master to indicate slave has been parameterized by another master (if address in DU byte 4 is not 255 and differs from its own address). Set to 0 by slave.

Diag_Data Response Telegram DU Byte 2 – Station_status_2

Bit	DIAGNOSTIC
0	Diag.Prm_Req: Set by a slave if it needs to be parameterized and cleared once parameterization is complete.
1	Diag.Stat_Diag: Static diagnostics. Slave sets this bit to cause the master to retrieve diagnostic information until this bit is cleared (the slave sets it if it's not able to provide user data).
2	Slave sets this bit to 1.
3	Diag.WD_ON: Set by slave to indicate Watchdog is active.
4	Diag.Freeze_Mode: Set by slave after it has received the Freeze control command.
5	Diag.Sync_Mode: Set by slave after it has received a Sync command.
6	Reserved.
7	Diag.Deactivated: Set by the master if slave has been marked inactive within the slave parameter set and is removed from cyclic processing. Slave sets this bit to 0.

Diag_Data Response Telegram DU Byte 3 – Station_status_3

Bit	DIAGNOSTIC
0-6	Reserved.
7	Diag.Ext_Diag_Overflow: Set if there is more diagnostic information than specified in Ext_Diag_Data. For example, slave sets if slave has more diagnostics than it can enter into its send buffer. Set by master if slave sends more diagnostic information than it can enter into its diagnostic buffer.

Diag_Data Response Telegram DU Byte 4 (Para Master Address)

Bit	DIAGNOSTIC
0-7	Diag.Master_Add: The master's address that parameterized this slave is entered here. If no master has parameterized this slave, then the DP Slave inserts 255 here (FF without parameterization).

Diag_Data Response Telegram DU Byte 5 - Ident_Number High Byte

Bit	DIAGNOSTIC
0-7	Manufacturer Identification Number High byte for ID & verification

Diag_Data Response Telegram DU Byte 6 - Ident_Number Low Byte

Bit	DIAGNOSTIC
0-7	Manufacturer Identification Number Low byte for ID & verification

รูปที่ 2-24 โครงสร้างเทเลแกรม Diag_Data

10. Global_Control ใช้การบริการประเภท 58 ในการส่งรูปแบบการทำงานที่พิเศษของอุปกรณ์รองเพียงตัวเดียว หรือกลุ่มของอุปกรณ์รอง โดยจะใช้การส่งข้อมูลการบริการแบบ SDN อุปกรณ์รองจะรับค่าขากอุปกรณ์หลักแล้วตั้งค่ารูปแบบให้ตรงกับที่อุปกรณ์หลักกำหนด และไม่ต้องมีการตอบกลับ

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	X	68H	8x	8x	X	3EH (62)	3AH (58)	X..	X	16H

DU Byte 1 (Control_Command To Be Executed)

7	6	5	4	3	2	1	0
Res.	*Res.*	Sync	Unsync	Freeze	Unfreeze	Clear_Data All outputs cleared.	*Res.*
0	0	00=No Function 01=Deactivated 10=Activated 11=Deactivated		00=No Function 01=Deactivated 10=Activated 11=Deactivated		0=Do Not Clear Outp 1=Clear Outputs	0

DU Byte 2 (Group_Select or Group_Ident Number)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

Group_Select: 0-255, Number must match the Group_Ident number of the parameterization data (multi-cast). If set to 0, all slaves are addressed (broadcast).

Sync: The output states transferred in Data_Exchange are delivered and frozen. The output data which follows is held until the next Sync command or Unsync command.

Unsync: This control cancels the sync command.

Freeze: This causes the states of the inputs to be read and frozen until the next Freeze command or Unfreeze. Slaves must ensure that following a freeze command, the last frozen values of the inputs must be transferred in the next data exchange cycle.

Unfreeze: This control cancels the freeze command.

Clear_Data: A class 1 master may use a global control telegram to inform the slaves that it is switching from Operate Mode to Clear Mode. This bit is set in Clear Mode (02 00H), and cleared in Operate Mode (00 00H). A slave will respond by either clearing its outputs, or it may optionally assume a user-defined state with its master in Clear Mode. Please refer to Failsafe Operation for more information on the use of this bit.

รูปที่ 2-25 โครงสร้างเทเลแกรม Global_Control

2.3.2.2 DDLM-FMA1/2

การทำงานในส่วนนี้จะเกี่ยวข้องกับการจัดการค่าให้กับข้อมูลในตัวอุปกรณ์ โดยจะมีการทำงานพื้นฐานดังต่อไปนี้ คือ

1. Reset เป็นการทำงานที่ต้องการให้ระบบมีการตั้งค่าตัวแปรเป็นค่าเริ่มต้นก่อนการทำงานใหม่ทั้งหมด
2. Set_value เป็นการทำงานในการที่ต้องการที่จะกำหนดค่าให้กับตัวแปรที่ต้องการ
3. RSAP_Activate เป็นการทำงานในลักษณะของการตอบกลับการทำงาน
4. Event เป็นการทำงานในลักษณะที่เกิดความผิดพลาดขึ้นในการทำงาน

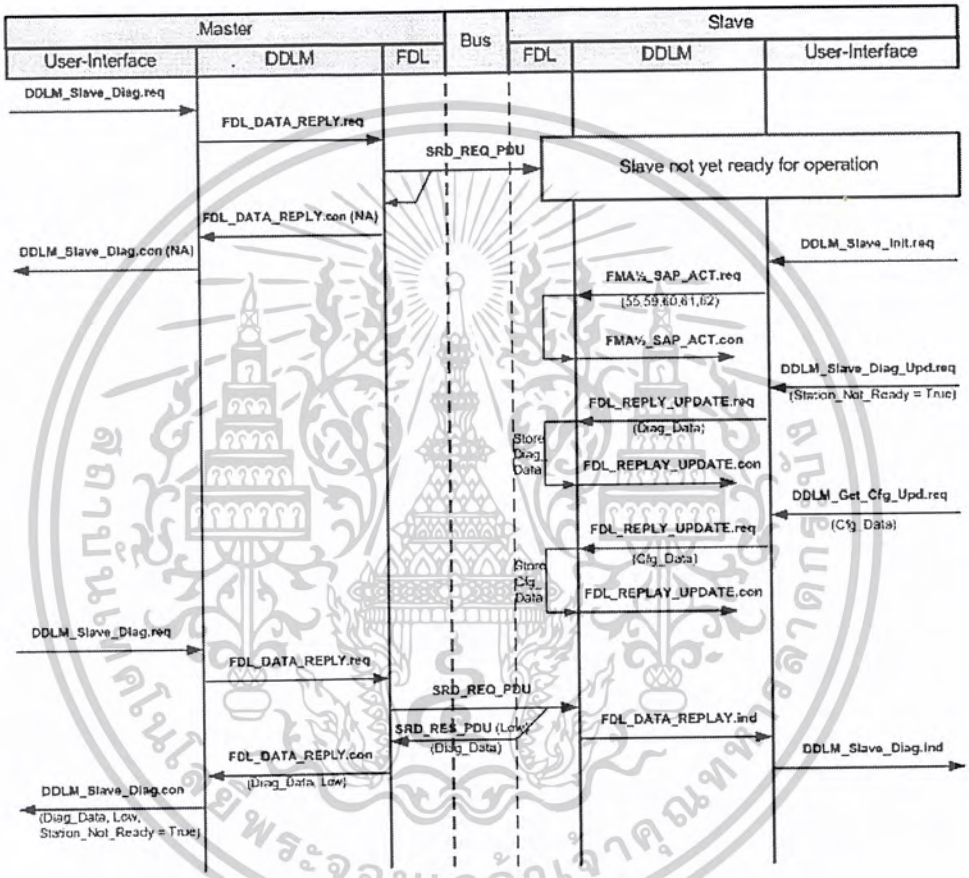
2.3.2.3 DDLM-FDL

การทำงานจะเกี่ยวข้องกับการให้บริการที่ชั้น DDLM โดยจะมีการส่งข้อมูลไปให้ในส่วน FDL จัดการรวบรวมข้อมูลที่เกี่ยวข้องทั้งหมด ก่อนที่จะส่งไปยังปลายทาง โดยแบ่งตามลักษณะการให้บริการได้ 2 ประเภท คือ

1. SDN เมื่อได้รับข้อมูลจากผู้ใช้แล้วจะทำการส่งข้อมูลไปยัง FDL และ FDL จัดการค้นหาข้อมูลที่เกี่ยวข้องกับการให้บริการประเภทนี้ และในทางกลับกัน เมื่อได้รับข้อมูลเข้ามา FDL จะจัดการแยกข้อมูลที่เกี่ยวข้องไปยังประเภทการบริการที่ต้องการ
2. SRD มีรูปแบบการทำงานคล้าย SDN แต่จะมีแจ้งการตอบกลับของข้อมูลหลังจากที่ส่งไปแล้วกลับมายังต้นทางด้วย

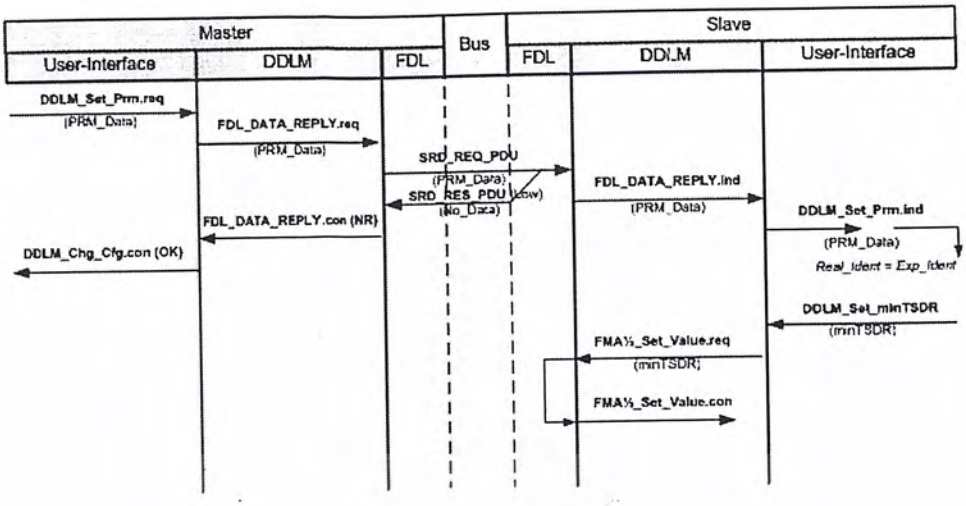
2.3.3 ลำดับการทำงาน

ในการสื่อสารระหว่างอุปกรณ์หลักละอุปกรณ์รองจะมีขั้นตอนในการทำงานดังต่อไปนี้



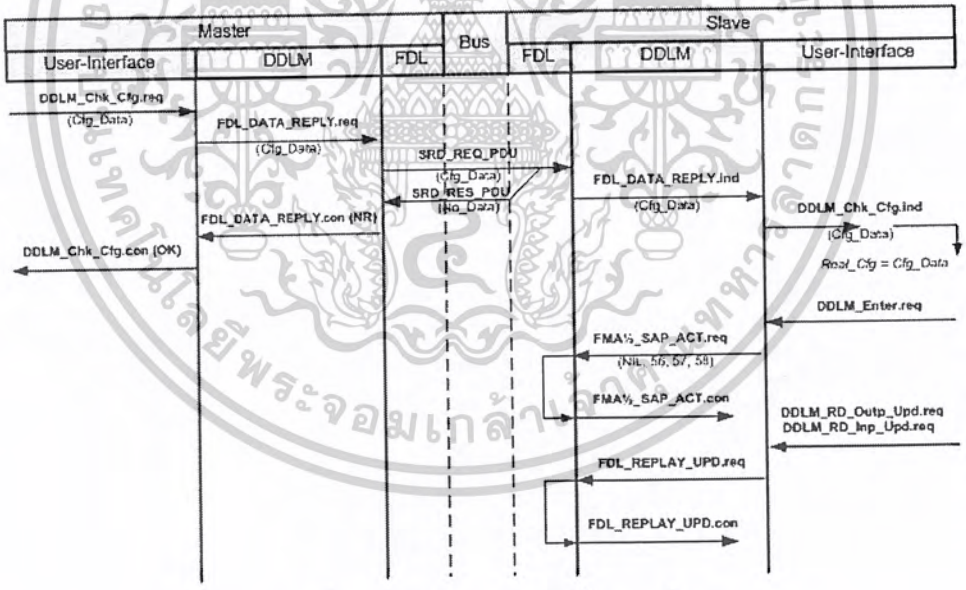
รูปที่ 2-26 ลำดับการทำงานขั้นตอนที่ 1

ในขั้นที่ 1 เริ่มแรกอุปกรณ์รองยังไม่พร้อมที่จะทำการสื่อสาร จึงต้องทำการตั้งค่าระบบต่างๆ ให้กับตัวอุปกรณ์เอง เช่น ข้อมูลการตรวจสอบ ข้อมูลองค์ประกอบของระบบ เป็นต้น ในขณะเดียวกันอุปกณ์หลักจะวนมาสอบถามความพร้อมของอุปกรณ์รองเสมอ หากอุปกรณ์รองได้ตั้งค่าต่างๆให้กับตัวเองพร้อมแล้ว เมื่ออุปกรณ์หลักวนมาสอบถามอีกครั้ง จะทำการอ่านค่าต่างๆ ที่อุปกรณ์รองได้กำหนดไว้กลับไปด้วย



รูปที่ 2-27 ลำดับการทำงานขั้นตอนที่ 2

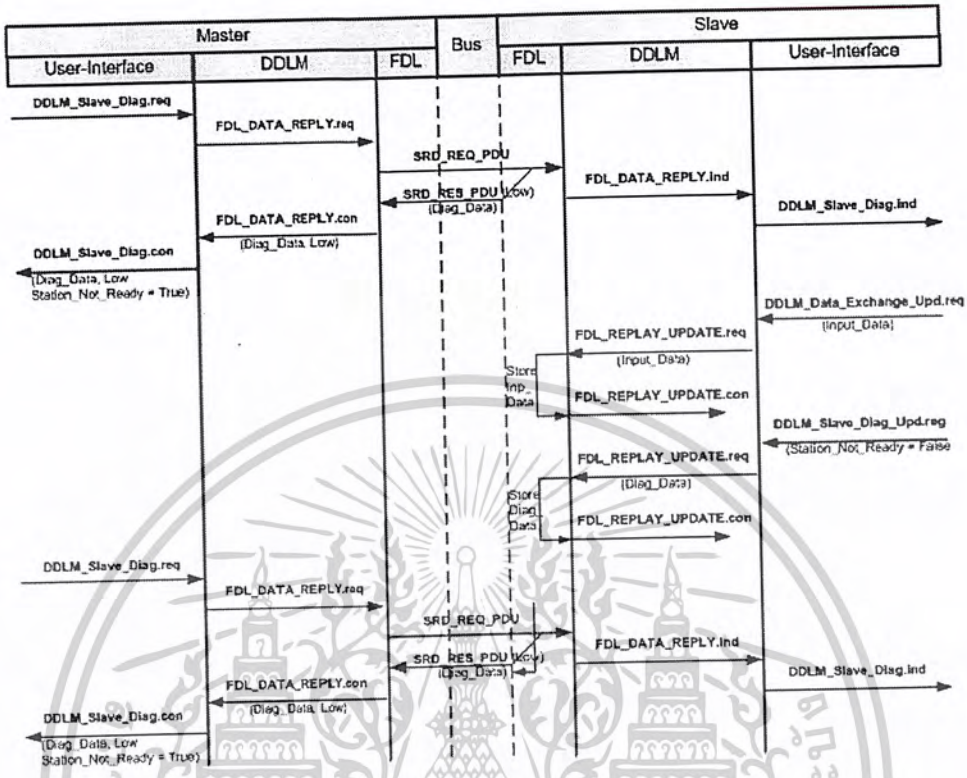
ในขั้นตอนที่ 2 หลังจากที่อุปกรณ์หลักรับทราบความพร้อมของอุปกรณ์รองแล้วจะทำการส่งข้อมูลตัวแปร เพื่อมากำหนดค่าต่างๆให้กับตัวอุปกรณ์รอง เมื่ออุปกรณ์รองได้รับค่าดังกล่าวจะนำค่าที่ได้มากำหนดให้กับตัวแปรของตนเอง



รูปที่ 2-28 ลำดับการทำงานขั้นตอนที่ 3

ในขั้นตอนที่ 3 อุปกรณ์หลักจะส่งค่าองค์ประกอบของระบบมายังอุปกณ์รองเพื่อทำการ กำหนดรูปแบบในการแลกเปลี่ยนข้อมูลระหว่างกัน หากข้อมูลที่ส่งมามีค่าไม่ตรงกัน จะไม่สามารถเริ่มการแลกเปลี่ยนได้ ตัวอุปกรณ์หลักต้องทำการอ่านค่าที่ถูกต้องไปจากอุปกรณ์รอง กรณีข้อมูลที่ส่งมามีค่า

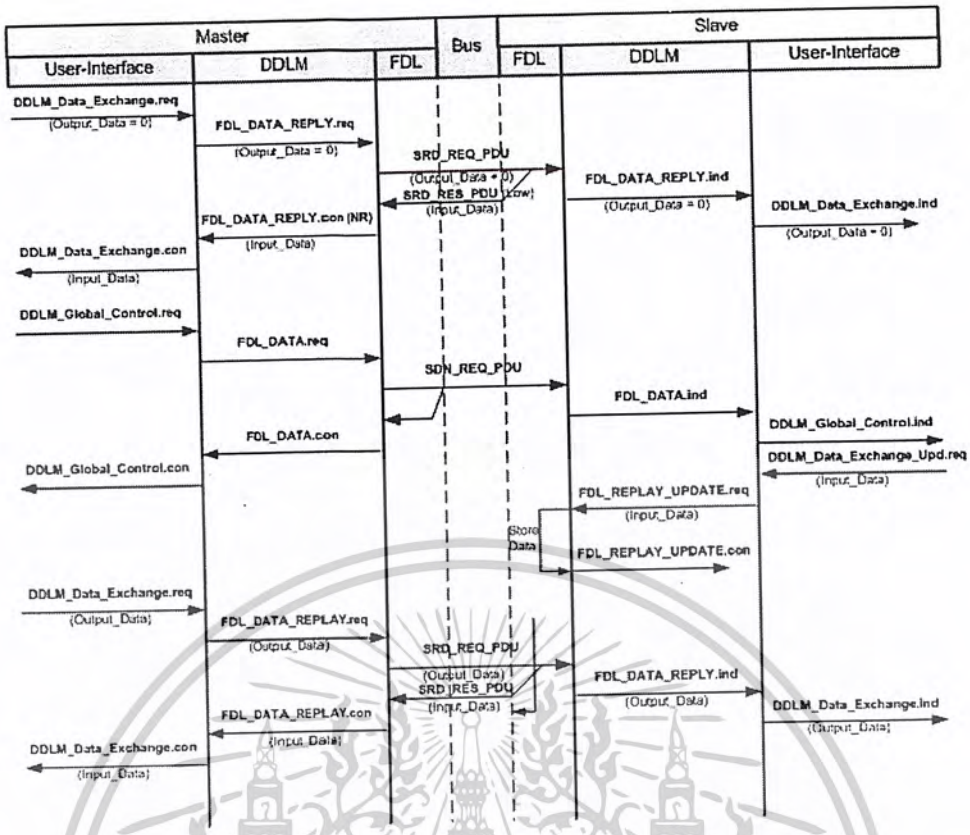
ตรงกันอุปกรณ์รองก้อจะใช้ข้อมูลดังกล่าวเป็นเกณฑ์ในการกำหนดรูปแบบในการสื่อสาร ในสภาวะนี้แสดงถึงความพร้อมของอุปกรณ์รองที่จะแลกเปลี่ยนข้อมูลกับอุปกรณ์หลัก



รูปที่ 2-29 ลำดับการทำงานขั้นตอนที่ 4

ในขั้นตอนที่ 4 หากอุปกรณ์รองยังไม่กำหนดค่าให้กับค่าความพร้อมในการสื่อสาร อุปกรณ์หลักยังจะไม่เริ่มทำการส่งข้อมูล หลังจากที่อุปกรณ์รองกำหนดค่าความพร้อมในการสื่อสารแล้ว ในขั้นตอนต่อไปจะเริ่มการแลกเปลี่ยนข้อมูลระหว่างกัน

ในขั้นตอนที่ 5 เป็นการแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์หลักและอุปกรณ์รอง โดยในการแลกเปลี่ยนอาจจะมีการแลกเปลี่ยนทั้งข้อมูลอินพุทและเอาพุทในช่วงเวลาเดียวกันได้ ในช่วงนี้อาจจะมีการส่งข้อมูลควบคุมพิเศษจากอุปกรณ์หลักมายังอุปกรณ์รองบางตัว หรือบางกลุ่มให้มีการทำงานที่พิเศษ และ ตลอดช่วงการแลกเปลี่ยนข้อมูลอุปกรณ์หลักสามารถที่จะวนมาตรวจสอบข้อมูลของอุปกรณ์รองได้ตลอดเวลา



รูปที่ 2-30 ลำดับการทำงานขั้นตอนที่ 5

2.4 GSD File

ส่วนที่สำคัญอีกส่วนของการสร้างอุปกรณ์รองขึ้นมา คือ GSD File ซึ่งข้อมูลที่อยู่ใน GSD File จะเป็นข้อมูลพื้นฐานของอุปกรณ์ ลักษณะจำเพาะของอุปกรณ์นั้นๆ โดยที่อุปกรณ์หลักจะมี GSD File เป็นของตนเอง (profile ของแต่ละอุปกรณ์) การใช้งาน GSD File ของโปรฟิบัสจะแตกต่างกับกระบวนการผลิตอื่น ๆ ตรงที่ ไม่ได้อยู่ภายในตัวอุปกรณ์เอง แต่จะแยกออกมาเป็น disk /drive มีเป็นลักษณะของ text file โดยอุปกรณ์หลัก จะเป็นผู้ใช้งาน ดังนั้นเมื่อเราต้องการใช้งาน อุปกรณ์รองจึงจำเป็นต้อง up load ข้อมูลอุปกรณ์รอง โดยใช้ GSD File ให้กับอุปกรณ์หลัก

2.4.1 โครงสร้าง GSD File

ประกอบไปด้วย ข้อมูลจำเพราของอุปกรณ์ เช่น ชื่อผู้ผลิต จำนวน Input/output ค่า baud rate ซึ่งมีมากมายหลากหลาย GSD file จะขึ้นต้นด้วย #Profibus_DP ในส่วนรายการจะเก็บชื่อ และ ค่าตัวแปรต่างๆ ในลักษณะ “ตัวแปร = ค่าตัวแปร” การ comment จะใช้ “;” แต่ละ บรรทัดจะเก็บข้อมูลได้มากสุดเพียง 80 ตัวอักษร แต่สามารถเชื่อมบรรทัดได้ด้วยการใช้ “\” ต่อท้ายบรรทัด GSD file แบ่งเป็น 2 ส่วนสำคัญคือ ส่วนข้อมูลทั่วๆ ไป คือ ข้อมูลทั่วๆ ไป และ ข้อมูลอุปกรณ์รอง

GSD General Specifications	Keyword	Keyword
This section contains information on vendor and device names, hardware and software revisions, ident_number, supported baud rates, reaction time intervals at supported baud rates for monitoring times, and optional signal support at the bus connector.	Vendor_name	187.5_supp
	Model_Name	500_supp
	Revision	1.5M_supp
	Ident_Number	MaxTsdr_9.6
	Protocol_Ident	MaxTsdr_19.2
	Station_type	MaxTsdr_93.75
	FMS_Support	MaxTsdr_187.5
	Hardware_Release	MaxTsdr_500
	Software_Release	MaxTsdr_1.5M
	9.6_supp	Redundancy
	19.2_supp	Repeater_Ctrl_Sig
	93.75_supp	24V_Pins

ตารางที่ 2-9 GSD File ทั่วๆ ไปของอุปกรณ์ต่างๆ

GSD Slave Specifications	Keyword	Keyword
This section contains all slave-related specifications, such as the number and type of I/O channels, specification of diagnostic text, auto-baud support, alternate mode support and information on available modules with modular devices.	Freeze_Mode_supp	Max_Input_Len
	Sync_Mode_supp	Max_Output_Len
	Auto_Baud_supp	Max_Data_Len
	Set_Slave_Add_supp	Unit_Diag_Bit
	User_Prm_Data_Len	Diag_Text
	User_Prm_Data	Unit_Diag_Area
	Min_Slave_Intervall	Module
	Modular_Station	Channel_Diag
	Max_Module	

ตารางที่ 2-10 ข้อมูล GSD File ของอุปกรณ์รอง

2.4.2 ตัวอย่าง GSD file

```
#Profibus_DP
GSD_Revision          =    1                    ;Version of the GSD file
Vendor_Name           =    "Acromag, Inc."      ;Vendor name
Model_Name            =    "983PB-2012"         ;Product name
Revision = "A"
Ident_Number          =    0x06F1               ;Ident Number
Protocol_Ident        =    0                    ;ProfiBusDP Only (1-DP/FMS)
Station_Type          =    0                    ;Type of device (Slave)
Hardware_Release      =    "A"                  ;Hardware version of the device
Software_Release      =    "A"                  ;Software version of the device
;
9.6_supp              =    1                    ;9600bps Supported
19.2_supp             =    1
93.75_supp            =    1
187.5_supp            =    1
500_supp              =    1
1.5M_supp             =    1
```

```
3M_supp                 =       1
6M_supp                 =       1
12M_supp                =       1
MaxTsdr_9.6             =       60              ;Maximum response time
MaxTsdr_19.2            =       60              ;at different baud rates.
MaxTsdr_93.75           =       60
MaxTsdr_187.5           =       60
MaxTsdr_500             =       100
MaxTsdr_1.5M            =       150
MaxTsdr_3M              =       250
MaxTsdr_6M              =       450
MaxTsdr_12M             =       800
;
Redundancy              =       0               ;Redundancy Not Supported
Repeater_Ctrl_Sig       =       2               ;Includes RTS Support w/TTL
Implementation_Type     =       "SPC3"          ;Uses Siemens SPC3 ASIC
24V_Pins                =       0               ;Does Not Include 24V
Fail_Safe               =       1               ;Supports Fail-Safe Mode
Freeze_Mode_supp        =       1               ;Supports FREEZE
Sync_Mode_supp          =       1               ;Supports SYNC
Auto_Baud_supp          =       1               ;Includes Auto Baud Detection
Set_Slave_Add_supp      =       1               ;Addr can be set via ProfiBus
User_Prm_Data_Len       =       3
User_Prm_Data           =       0x00, 0x00, 0x00        ;Module Specific Parameters
; 00H = Set outputs to 0
; 01H = Maintain Last Output Values
; 02H = Set output to user-defined values in bytes 1 and 2
; Byte 1 is the lower byte of user-defined output data which is outputs 0 to 7
; Byte 2 is the upper byte of user-defined output data which is outputs 8 to
Slave_Family            =       3
Min_Slave_Interval      =       1               ;Min_Slave_Interval is 100us
Modular_Station         =       0               ;0-compact, 1-modular
Max_Diag_Data_Len       =       6               No User Diagnostics are Sent
; I/O Byte
```

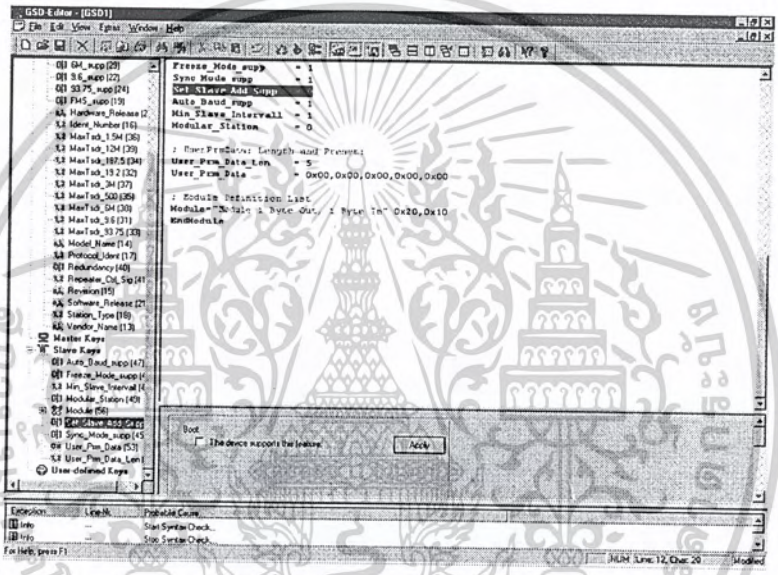
Module = "12 CH DIG I/O:xxxx1198 76543210" 0x11, 0x21

EndModule

;

2.4.3 โปรแกรมแก้ไข-ตรวจสอบ GSD File

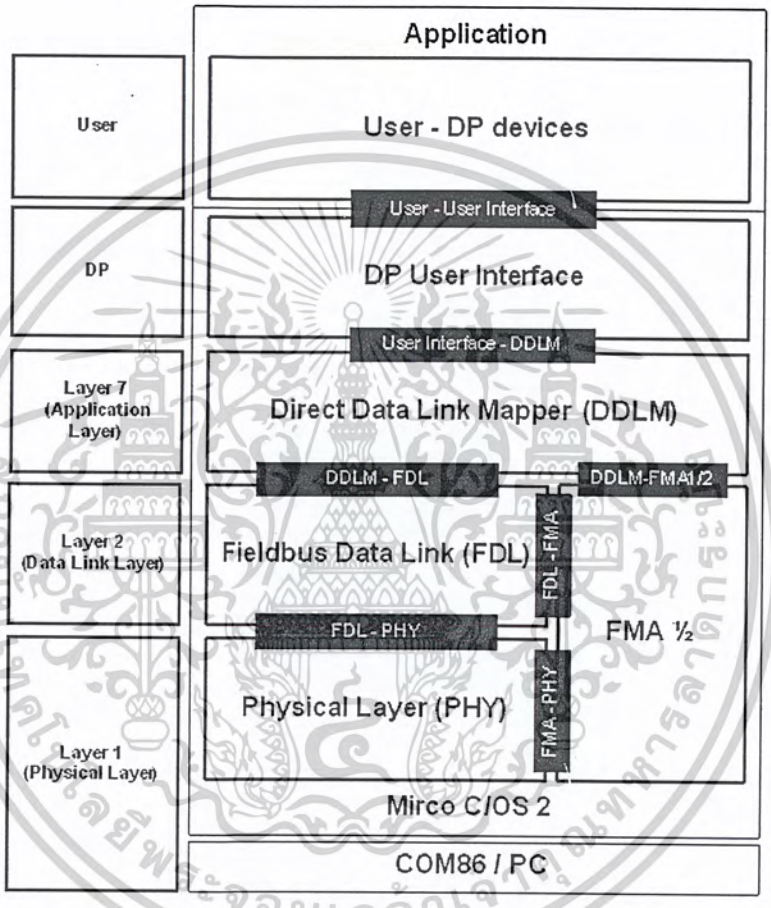
ในปัจจุบันมีโปรแกรมพัฒนาเพื่ออำนวยความสะดวกในการแก้ไข GSD File ขึ้นมามาก ทั้งนี้ขอยกตัวอย่างโปรแกรมแก้ไข GSD File มี่ชื่อว่า GSD Editor นอกจากนี้ยังมีโปรแกรมที่พัฒนามาเพื่อตรวจสอบความถูกต้องของ GSD File ด้วย ยกตัวอย่างเช่น GSD Checker หรือในบางโปรแกรมจะรวมเอาความสามารถทั้ง 2 อย่างไว้ในโปรแกรมเดียว



รูปที่ 2-31 โปรแกรม GSD Editor

บทที่ 3
การออกแบบอุปกรณ์รองโปรฟิบัสดี-พี

สถาปัตยกรรมโครงสร้างของอุปกรณ์รองโปรฟิบัสดี-พี (PROFIBUS-DP Slave Device) ทั้งหมดที่ได้ออกแบบไว้ประกอบด้วยส่วนต่าง ๆ ดังนี้



รูปที่ 3-1 สถาปัตยกรรมโครงสร้างของอุปกรณ์รองโปรฟิบัสดี-พี

- ส่วน User หมายถึงส่วนที่ผู้ใช้งานนำไปพัฒนาต่อหรือพัฒนานำให้เข้ากับที่ผู้พัฒนาต้องการ
- ส่วน DP หมายถึงส่วนจัดการเกี่ยวกับโปรโตคอล PROFIBUS DP สำหรับอุปกรณ์ Slave เช่น กำหนดของอุปกรณ์ กำหนดค่าการส่งข้อมูล กำหนดชนิดของอุปกรณ์ ตามมาตรฐานจะเรียกว่า User-Interface
- ชั้น Application Layer หมายถึงชั้นที่สร้างฟังก์ชันส่งข้อมูลและกำหนดค่าของระบบให้กับชั้น DP และจะติดต่อโดยตรงกับชั้น Data Link Layer ตามมาตรฐานจะเรียกว่า Direct Data Link Mapper (DDLM)

- ชั้น Data Link Layer ในชั้นนี้จะประกอบด้วย 2 ส่วน ตามมาตรฐาน ส่วนแรกจะเรียกว่า Field Data Link (FDL) ทำหน้าที่ในการรับและส่งข้อมูลจากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์หนึ่งโดยมีการสร้างบริการส่งข้อมูล SRD และ SDN ส่วนที่สองคือ Fieldbus Management จะสร้างฟังก์ชันบริการเพื่อกำหนดค่าการทำงานของระบบ ทั้ง 2 ส่วนนี้จะเรียกใช้โดย DDLM รวมถึงรับส่งข้อมูลกับ Physical Layer
- ชั้น Physical Layer ส่วนนี้จะรับ ส่งข้อมูลอนุกรม 1 ไบต์จากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์หนึ่งผ่านตัวกลาง ซึ่งจะรวมถึงการติดกับกับ UART และพอร์ตอนุกรมของ Platform ใด ๆ ไปยังมาตรฐาน RS-485

ดังนั้นในการออกแบบการทำงานของโปแกรมจึงต้องพิจารณาในส่วนของการเชื่อมต่อของแต่ละ Layer ประกอบด้วยส่วนต่าง ๆ ดังนี้

1. Physical Layer
2. Physical เชื่อมต่อกับ Filedbus Management
3. Physical เชื่อมต่อกับ Filedbus Data Link
4. Filedbus Management เชื่อมต่อกับ Filedbus Data Link
5. Filedbus Management เชื่อมต่อกับ Direct Data link Mapper
6. Filedbus Data Link เชื่อมต่อกับ Direct Data link Mapper
7. Direct Data link Mapper เชื่อมต่อกับ User Interface
8. User Interface เชื่อมต่อกับ User

3.1 PROFIBUS-DP Layer 1 (Physical Layer)

3.1.1 ลักษณะทางไฟฟ้าและกลไกการทำงาน

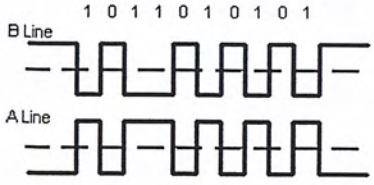
มาตรฐานของ PROFIBUS-DP กำหนดให้ใช้การส่งข้อมูลแบบสมดุล (balanced line transmission) เป็นไปตามมาตรฐานทางไฟฟ้า EIA RS-485 (EIA: Electronic Industries Association; RS-485) สามารถส่งและรับข้อมูลดิจิตอลในระบบระหว่างจุดหลาย ๆ จุด (multipoint systems) หรือโครงสร้างแบบบัส (Topology bus) ตัวกลางที่ใช้คือ สายตีเกลียวคู่มีฉนวน (Shielded Twisted Pair) ความเร็วในการส่งข้อมูล 9,600 bps ถึง12 Mbps

การส่งข้อมูลเป็นแบบครั้งละหนึ่งตัวอักษร (Character Frame) ประกอบด้วย 11 บิต เข้ารหัสแบบ NRZ (Non Return to Zero) ประกอบด้วย 1 start bit, Data 8 bit ,parity 1 bit และ 1 stop bit ดังรูปที่ 3-2

Start	D0	D1	D2	D3	D4	D5	D6	D7	Parity	Stop
“0”	0	1	2	3	4	5	6	7	**even**	**“1”**
←	LSB	←	←	←	←	←	←	MSB	←	←

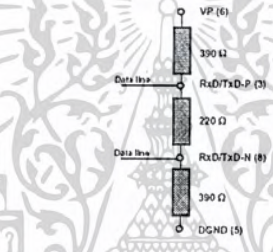
รูปที่ 3-2 ลำดับของการส่งข้อมูล

การส่งข้อมูลจะส่งแบบสมดุล โดยใช้สาย 2 เส้นคือ A, B และอีก 1 เส้นเป็น GND โดยแต่ละบิตจะใช้วิธีเข้ารหัส (Encoding) แบบ NRZ (Non Return to Zero) ดังรูปที่ 3-3

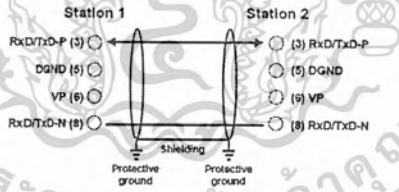


รูปที่ 3-3 รูปแบบสัญญาณแบบ NRZ

สายส่งข้อมูล A และ B ของมาตรฐาน EIA RS 485 ที่จุดปลายทั้งสองข้าง จะต้องมีการปิดสายที่เรียกว่า Bus Terminator สำหรับสายของโปรฟิบัสด้วย Pull-Down resister และ Pull-Up resister ดังรูปที่ 3-4

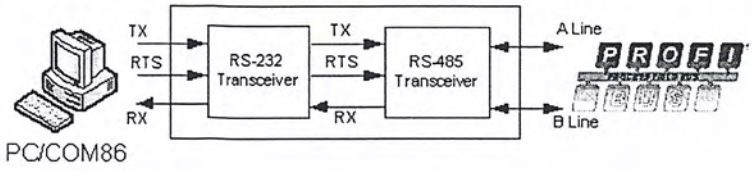


รูปที่ 3-4 รูปแบบการต่อปลายสายสัญญาณโปรฟิบัส



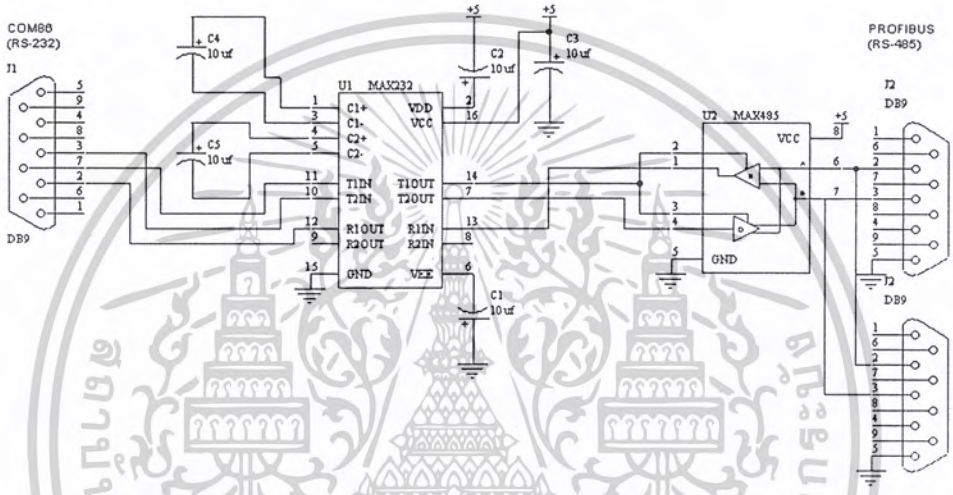
รูปที่ 3-5 ลักษณะการเชื่อมต่อสายสัญญาณของ DB9-Connector

สำหรับอุปกรณ์ COM86 มีพอร์ตสื่อสารอนุกรม (serial port) จำนวน 2 พอร์ต ในการใช้งานพอร์ตCOM1 จะใช้สำหรับ Monitor และสั่งงาน สำหรับ COM2 ซึ่งเป็นมาตรฐาน RS-232 จะติดต่อกับเครือข่าย PROFIBUS DP ที่เป็นเครือข่าย RS-485 จึงต้องมีการสร้างอุปกรณ์แปลงสัญญาณ (แรงดันไฟฟ้า)



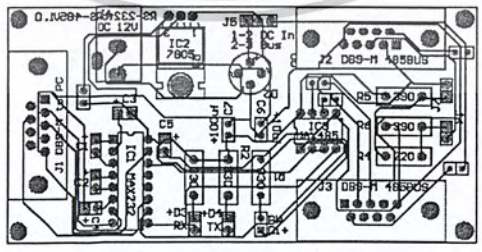
รูปที่ 3-6 Block Diagram แปลงสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485

จากรูป Block Diagram ข้างต้นสามารถออกแบบวงจรแปลงสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485 ดังนี้



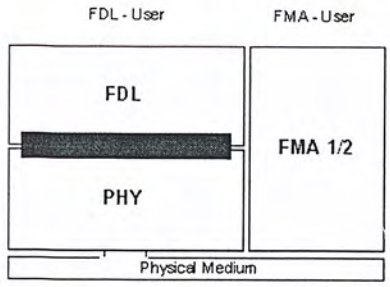
รูปที่ 3-7 วงจรแปลงสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485

สำหรับ RS-485 Transceiver อุปกรณ์ที่ใช้คือ MAX-485 หรือ IC เบอร์ 75176 และ RS-232 Transceiver อุปกรณ์ที่ใช้คือ MAX-232 หรือ ICL232 และการควบคุมการส่งข้อมูลจาก RS-232 จะใช้สัญญาณ RTS ด้วยเมื่อต้องการส่งข้อมูล TX



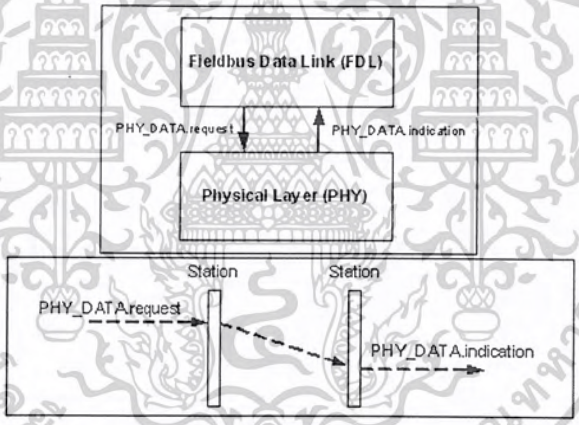
รูปที่ 3-8 แผ่น PCB ของวงจรสัญญาณไฟฟ้ามาตรฐาน RS-232/RS-485

3.1.2 การเชื่อมต่อระหว่าง PHY กับ FDL



รูปที่ 3-9 แสดงมุมมองการเชื่อมต่อของ PHY กับ FDL

การทำงานของ Physical Layer (PHY) จะเป็นทำงานในระดับล่างสุด คือ ติดต่อกับอุปกรณ์ส่งข้อมูลอนุกรม (UART) โดยตรงและจัดการส่งข้อมูลเป็น Byte จากผู้ส่งไปยังผู้รับตามความเร็วที่กำหนด ภายใน Physical Layer (PHY) จะสร้างบริการที่เรียกว่า "PHY Data Service" ซึ่งจะลักษณะการทำงานดังรูปที่ 3-10



รูปที่ 3-10 การเชื่อมต่อระหว่าง PHY กับ FDL

บริการที่จัดเตรียมไว้สำหรับส่งและรับข้อมูลกับ FDL จะมี 2 บริการคือ

1. PHY_DATA.request (FDL_symbol) เป็นฟังก์ชันสำหรับส่งข้อมูล
2. PHY_DATA.indication (FDL_symbol) เป็นฟังก์ชันสำหรับรับข้อมูลเมื่อมีข้อมูลเข้ามา

FDL_symbol หมายถึง ข้อมูลที่ส่งในระดับบิต ตามมาตรฐานของ PROFIBUS จะมีค่า 0 กับ 1

สำหรับการออกแบบส่วนนี้ FDL_symbol จะกำหนดให้เป็น Byte หรือส่งอักษร 1 ตัวอักษรแทน จึงสร้างเป็นบริการสำหรับส่งและรับข้อมูลกับ FDL ด้วยฟังก์ชันดังนี้

1. INT8U PHY_DATA_PUT (INT8U)
   เป็นฟังก์ชันสำหรับส่งข้อมูลจำนวน 1 ตัวอักษร
2. INT8U PHY_DATA_GET ()

เป็นฟังก์ชันสำหรับรับข้อมูลจำนวน 1 ตัวอักษร

3. OSSemPost (sem_rx_phy_to_fdl)

เป็นการส่งสัญญาณให้ FDL เพื่อบอกว่ามีข้อมูลเข้ามาแล้วให้ FDL ทำการรับข้อมูลไปได้

4. OSSemPend (sem_rx_fdl_to_phy, 1000, &err)

เป็นการรอสัญญาณจาก FDL เพื่อบอกว่า FDL ทำการรับข้อมูลไปเสร็จแล้ว

5. OSSemPend (PHY_PUT, 0, &err)

เป็นการจัดการการขอเข้าไปทำการส่งข้อมูลทีละ 1 ตัวอักษร ได้ทีละ 1 ครั้ง (Acquire semaphore)

6. OSSemPost (PHY_PUT)

เป็นการจัดการการออกจากการส่งข้อมูลทีละ 1 ตัวอักษร (Release semaphore)

7. INT8U int_putch (INT8U)

เป็นฟังก์ชันในการส่งข้อมูลในระดับล่างออกทาง UART ทีละ 1 ตัวอักษรประกอบด้วย บิตเริ่มต้น (Start Bit) จำนวน 1 บิต ข้อมูล (Data) จำนวน 8 บิต บิตตรวจสอบ (Even Parity) จำนวน 1 บิตและบิตหยุดจำนวน 1 บิต รวมทั้งหมด1 บิต

8. INT8U int_getch ()

เป็นฟังก์ชันในการรับข้อมูลในระดับล่างจากทาง UART จำนวน 1 ตัวอักษร โดยมีความสามารถเก็บข้อมูลบัฟเฟอร์ไว้ในตัวด้วย

9. INT8U CheckInterrupt ()

เป็นฟังก์ชันในการตรวจสอบว่ามีข้อมูลเข้ามาจาก UART หรือยัง

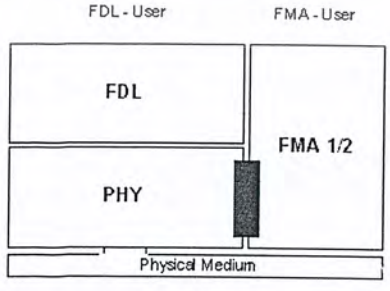
10. INT8U InitComport ()

เป็นฟังก์ชันในการกำหนดค่าเริ่มต้นของพอร์ตอนุกรม

11. Physical Task ()

เป็นงานของชั้น PHY ทำหน้าที่ดูแลการรับข้อมูลที่เข้ามาจากพอร์ตอนุกรมผ่าน UART โดยจะถามการ Poll ถามว่ามีข้อมูลเข้ามาหรือยัง ถ้ามีข้อมูลเข้ามาแล้วให้ส่งสัญญาณไปแจ้ง FDL ให้มารับข้อมูล

3.1.3 การเชื่อมต่อระหว่าง PHY กับ FMA

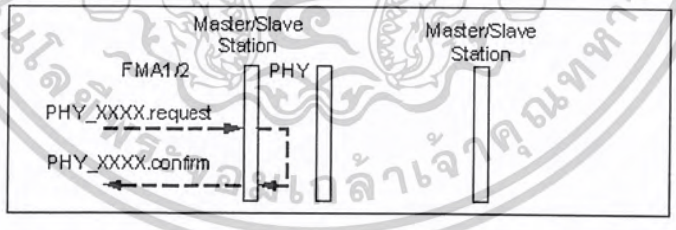


รูปที่ 3-11 แสดงมุมมองการเชื่อมต่อของ PHY กับ FMA1/2

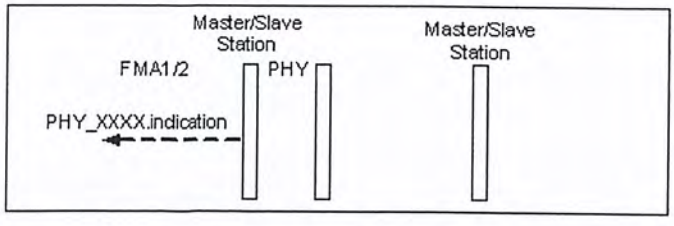
สำหรับส่วนนี้ FMA จะเป็นใช้งาน PHY ดังนั้น PHY จะต้องมี Service ที่ต้องให้บริการฟังก์ชันการทำงานดังต่อไปนี้

1. Reset PHY เป็นฟังก์ชันที่ใช้ในการสั่งให้ reset การทำงานของ PHY
   - PHY_RESET.request ฟังก์ชันสั่งให้ PHY ทำการ reset
   - PHY_RESET.confirm ฟังก์ชันแจ้งกลับมาว่าได้ทำการ reset แล้ว
2. Set Value เป็นฟังก์ชันที่ใช้ในการกำหนดค่าให้กับตัวแปรภายใน PHY
   - PHY PHY_SET_VALUE.request ฟังก์ชันกำหนดค่าให้กับตัวแปรภายใน PHY
   - PHY_SET_VALUE.confirm ฟังก์ชันการแจ้งกลับมาว่าได้กำหนดค่าแล้วได้หรือไม่
3. Read Value เป็นฟังก์ชันที่ใช้ในอ่านค่าให้กับตัวแปรภายใน PHY
   - PHY PHY_READ_VALUE.request ฟังก์ชันอ่านค่าให้กับตัวแปรภายใน PHY
   - PHY_READ_VALUE.confirm ฟังก์ชันการแจ้งกลับว่าพร้อมค่าของตัวแปรที่ต้องการอ่าน
4. Event PHY
   - PHY_EVENT.indication ฟังก์ชันการแจ้งว่ามี เหตุการณ์ (event) ใดเกิดขึ้น
5. ตัวแปรของ PHY
   - Transmitter_output กำหนดการส่งข้อมูล ทำงานหรือไม่ทำงาน (enabled, disabled)
   - Received_signal_source กำหนดช่องสัญญาณที่จะรับตัวหลักหรือตัวอื่น (primary, alternate)
   - Loop กำหนดการส่งข้อมูลส่งแบบวนกลับมาหรือ (enabled, disabled)

ลักษณะการทำงานของบริการการเชื่อมต่อระหว่าง Physical Layer (PHY) กับ Fieldbus Management (FMA)



รูปที่ 3-12 ลักษณะการทำงานของบริการ Reset PHY, Set Value PHY, Read Value



รูปที่ 3-13 ลักษณะการทำงานของบริการ Event PHY

สำหรับการออกแบบในส่วนนี้ FMA จะเรียกใช้งานบริการของ PHY ผ่านฟังก์ชันที่สร้างขึ้นมา ตามจึงได้ออกแบบฟังก์ชันเพื่อเรียกใช้งานดังนี้

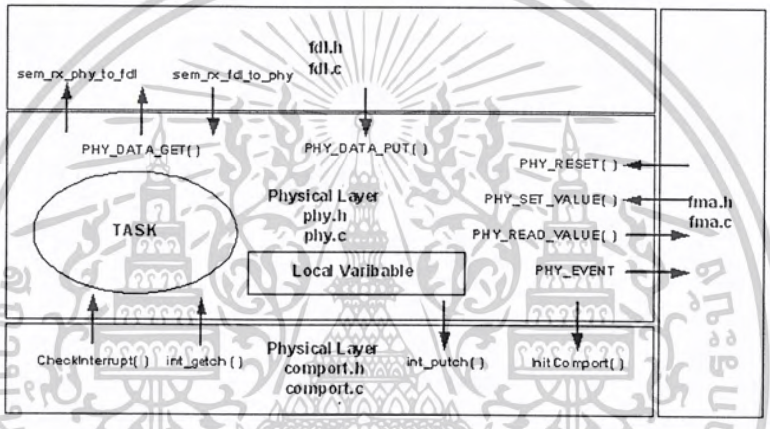
1. INT8U PHY_RESET ()
   เป็นฟังก์ชันที่สั่งให้ PHY ทำการ reset
2. INT8U PHY_SET_VALUE_Transmitter_output (INT8U Desired_value)
   เป็นฟังก์ชันที่ใช้กำหนดค่าให้กับ Transmitter_output ด้วยค่าที่ต้องการกำหนด (Desired_value)
3. INT8U PHY_SET_VALUE_Received_signal_source (INT8U Desired_value)
   เป็นฟังก์ชันที่ใช้กำหนดค่าให้กับ Received_signal_source ด้วยค่าที่ต้องการกำหนด (Desired_value)
4. INT8U PHY_SET_VALUE_Loop (INT8U Desired_value)
   เป็นฟังก์ชันที่ใช้กำหนดค่าให้กับ Loop ด้วยค่าที่ต้องการกำหนด (Desired_value)
5. INT8U PHY_READ_VALUE_Transmitter_output ()
   เป็นฟังก์ชันที่ใช้อ่านค่าของ Transmitter_output โดยจะส่งค่าของ Transmitter_output กลับมา
6. INT8U PHY_READ_VALUE_Received_signal_source ()
   เป็นฟังก์ชันที่ใช้อ่านค่าของ Received_signal_source โดยจะส่งค่าของ Received_signal_source
7. INT8U PHY_READ_VALUE_Loop ()
   เป็นฟังก์ชันที่ใช้อ่านค่าของ Loop โดยจะส่งค่าของ Loop
8. OSMboxPost (PHY_EVENT, &PHY_EVENT_VAR)
   เป็นการส่งสัญญาณด้วยข้อความ (Message) ไปยังกล่องข้อความ (Message Box) เพื่อบอกว่ามีการเปลี่ยนแปลงของตัวแปรใดแล้ว
9. ตัวแปรที่สำคัญของ PHY ที่ต้องกำหนด
   - Transmitter_output ชนิด INT8U กำหนดการส่งข้อมูล
     - 1 = enabled
     - 0 = disabled
   - Received_signal_source ชนิด INT8U กำหนดช่องสัญญาณที่จะรับ
     - 0 = primary
     - 1 = alternate
     - 2 = random
   - Loop ชนิด INT8U กำหนดการส่งข้อมูลส่งแบบวนภายในตัวเอง 1 = enabled, 0 = disabled
10. ตัวแปรเพิ่มเติม
    - Baud_rate อัตราเร็วในการส่งข้อมูล
      - 0 = 9,6 kbit/s
      - 1 = 19,2 kbit/s
      - 2 = 93,75 kbit/s

- 3 = 187,5 kbit/s
- 4 = 500 kbit/s
- 5 = 38,4 kbit/s
- 6 = 1500 kbit/s

- Medium_red กำหนดจำนวนของการเข้าใช้งานตัวกลาง (Single, Redundant)
  - 0 = no_redundancy
  - 1 = bus_A_highprior
  - 2 = bus_B_highprior

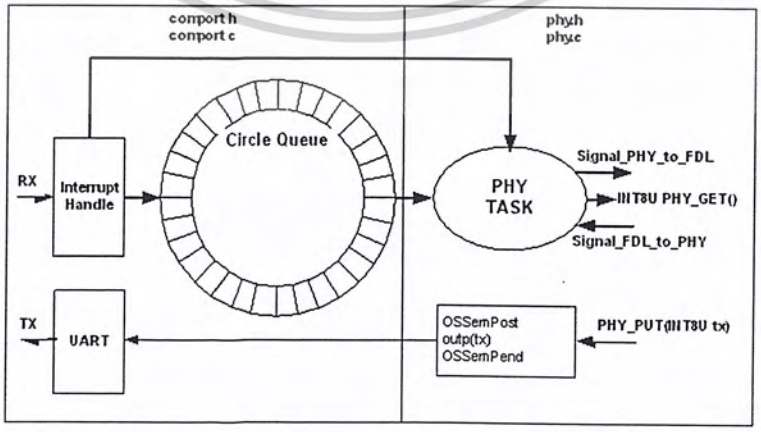
3.1.4 สรุปการออกแบบระดับ Physical Layer

สามารถสรุปการออกแบบการทำงานของระดับ Physical Layer ทั้งหมด ดังรูปที่ 3-11



รูปที่ 3-14 การออกแบบโครงสร้างการทำงานของ PHY

โครงสร้างการทำงานของ PHY ในส่วนการติดต่อกับ FDL จะทำหน้าที่ในการส่งข้อมูลที่อ่านได้จาก Comport และ รับข้อมูลเพื่อส่งออกทาง Comport ในส่วนของการติดต่อกับ FMA จะทำหน้าที่ในการ Set ค่าตัวแปร การ Read ค่าตัวแปร รวมถึงการ Reset ค่าตัวแปรทั้งหมดด้วย



รูปที่ 3-15 การออกแบบโครงสร้างการรับ-ส่งข้อมูลของ PHY

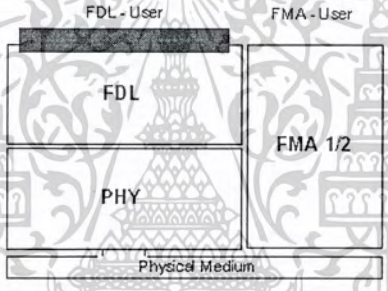
ในส่วนของการติดต่อกับ Comport จะใช้ Function CheckInterrupt() ในการตรวจสอบการเข้ามาของข้อมูล หลังจากนั้นจะใช้ Function int_getch() ในการดึงเอาข้อมูล Character ออกจาก Circular Buffer โดยก่อนเริ่มการแลกเปลี่ยนข้อมูลจะทำการ InitComport() ก่อน และ หากต้องการส่งข้อมูลออกไปยัง Comport จะเรียกใช้ Function int_putch()

3.2 PROFIBUS Layer 2 (Data Link Layer)

การเชื่อมต่อกับโปรฟิบัสในชั้นที่ 2 (PROFIBUS Layer 2 Interface) จะมีการเชื่อมต่อ 2 ส่วนผ่านบริการของโปรฟิบัส คือ ส่วนแรกสำหรับส่งและรับข้อมูลผ่านซึ่งเป็นบริการของโปรฟิบัสที่เรียกว่า "FDL" ย่อมาจาก "Fieldbus Data Link" และส่วนที่สองใช้สำหรับบริหารและจัดการซึ่งเป็นบริการของโปรฟิบัสที่เรียกว่า "FMA" ย่อมาจาก "Fieldbus Management" การสร้างทั้งสองส่วนนี้สามารถทำได้บน hardware, firmware, software อย่างใดอย่างหนึ่งก็ได้

3.2.1 Fieldbus Data Link (FDL)

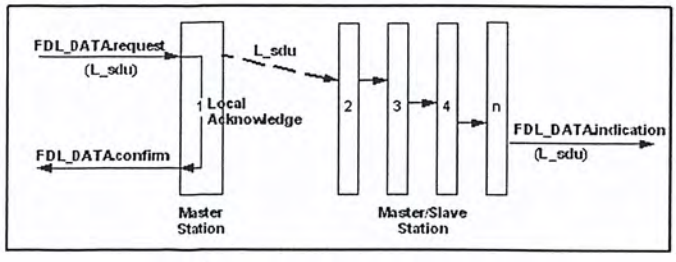
3.2.1.1 FDL User - FDL Interface



รูปที่ 3-16 การเชื่อมต่อของ FDL User กับ FDL

การส่งและรับข้อมูลระหว่างสถานีหนึ่งไปยังอีกสถานีหนึ่งจะรับผิดชอบโดย FDL การใช้งานนั้นจะ เรียกใช้งานผ่านส่วนเชื่อมต่อระหว่าง FDL กับผู้ใช้งาน FDL (FDL User) สำหรับบริการที่FDL ต้องเตรียมไว้สำหรับการส่งและรับข้อมูลตามมาตรฐานที่กำหนดไว้สำหรับอุปกรณ์ PROFIBUS DP Slave มีดังนี้ คือ

3.2.1.1.1 ส่งข้อมูลโดยไม่มีการตอบรับ - Send Data with No Acknowledge (SDN)



รูปที่ 3-17 การส่งข้อมูลแบบ SDN

การส่งข้อมูลแบบนี้จะใช้ส่งข้อมูลแบบ Multicast กับแบบ Broadcast โดยไม่มีการตอบรับกลับจากผู้ที่ได้รับข้อมูล บริการนี้จะไม่รับประกันว่าทุกคนจะได้รับ ข้อมูลที่จะส่งจะเก็บใน L_sdu และเมื่อต้องการส่งข้อมูลจะเรียกผ่านฟังก์ชัน FDL_DATA.request (b7=1, Code No 4/6)

เมื่อข้อมูลได้ถูกส่งออกไปจะมีการบอกว่าได้ส่งข้อมูลออกไปด้วย FDL_DATA.confirm และเมื่อข้อมูลที่ส่งไปถึงสถานีผู้รับ ผู้รับจะได้ FDL_DATA.indication แจ้งเมื่อได้รับข้อมูลแล้ว สำหรับบริการพื้นฐานของการส่งข้อมูลแบบ SDN จะมีฟังก์ชันเรียกที่ตามมาตรฐานดังต่อไปนี้

1. FDL_DATA.request

   (SSAP, DSAP, Rem_add, L_sdu, Serv_class)

   เป็นฟังก์ชันสำหรับส่งข้อมูล (L_sdu) ไปหาเครื่องปลายทาง(Rem_add) (สำหรับ Master)

2. FDL_DATA.indication

   (SSAP, DSAP, Loc_add, Rem_add, L_sdu, Serv_class)

   เป็นฟังก์ชันเมื่อ FDL ได้รับข้อมูลจากสถานีอื่นที่ส่งเข้ามา จะแจ้งบอกว่าได้รับข้อมูลกับผู้ที่รับข้อมูลพร้อมแจ้งสถานการณ์ทำงาน (สำหรับ Master/Slave)

3. FDL_DATA.confirm

   (SSAP, DSAP, Rem_add, Serv_class, L_status)

   เป็นฟังก์ชันเมื่อ FDL ได้ทำการส่งข้อมูลออกไปแล้วให้ผู้ที่ส่งข้อมูลที่ทราบสถานะการทำงาน (สำหรับ Master)

   สำหรับความหมายของพารามิเตอร์ที่เกี่ยวข้องมีดังนี้

   - SSAP (Source Service Access Point)

     หมายถึง กำหนดจุดเข้าถึงบริการของสถานีต้นทาง ซึ่งมีค่าทั้งหมด 0-62 สำหรับ SSAP = 63 เรียกว่า Global Access Address ไม่อนุญาตให้ใช้

   - DSAP (Destination Service Access Point)

     หมายถึง กำหนดจุดเข้าถึงบริการของสถานีปลายทาง

   - Rem_add (Remote Address)

     หมายถึง กำหนดหมายเลขตำแหน่งของเครื่องปลายทาง

   - Loc_add (Local Address)

     หมายถึง กำหนดหมายเลขตำแหน่งของเครื่องต้นทาง

   - L_sdu (Link service data unit)

     หมายถึงข้อมูลที่ผู้ใช้งานต้องการส่งจากสถานีหนึ่งไปยังอีกสถานีหนึ่ง สามารถบรรจุข้อมูลได้น้อยที่สุด 1 ไบต์และมากที่สุด 246 ไบต์ ถ้ามีการกำหนด SSAP, DSAP จะส่งข้อมูลได้มากสุด 242 ไบต์

   - Serv_class (Service Class)

     หมายถึงการกำหนดระดับความสำคัญของข้อมูลที่จะส่งดังนี้

ความสำคัญสูง (High) ใช้กับข้อมูลแบบ Time-critical messages เช่น alarms, synchronization และ Coordination data

ความสำคัญต่ำ (Low) ใช้กับข้อมูลแบบ less urgent messages เช่น process, diagnostic, program

- L_status

หมายถึง บอกสถานะในว่าการส่งข้อมูลนั้นสำเร็จหรือล้มเหลว

OK การส่งข้อมูลสำเร็จโดย FDL/PHY Controller ภายในตัวเอง

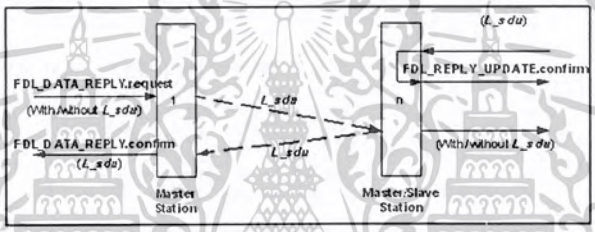
LS บริการของ LSAP ไม่ทำงาน

LR ทรัพยากรของสถานีนี้ไม่เพียงพอหรือไม่มี

DS การส่งข้อมูลไม่ได้ Token ring หรือสายขาด

IV พารามิเตอร์ที่ใช้ส่งไม่ถูกต้อง

3.2.1.1.2 ส่งและขอข้อมูลโดยมีการตอบกลับ - Send and Request Data with Reply (SRD)



รูปที่ 3-18 การส่ง-ร้องขอข้อมูลแบบ SRD

การส่งข้อมูล ผู้สั่งนั้นจะเตรียมข้อมูลที่จะส่งสำหรับผู้รับ (L_sdu) ข้อมูลจะส่งจาก FDL Controller ของผู้ส่งด้วย FDL_DATA_REPLY.request และขอข้อมูลจากผู้รับในเวลาเดียว เมื่อตอบรับการใช้บริการนี้จะทำการส่งข้อมูลและขอข้อมูลจากเครื่องปลายทาง (b7=1, Code No 12/13)

เมื่อสถานีปลายทางได้รับข้อมูลและเฟรมที่ไม่มีผิดพลาดแล้ว จะส่งข้อมูลกลับมา (b7=0, Code No 8/10) โดยข้อมูลที่จะส่งกลับมานั้น ก่อนหน้านั้น สถานีปลายทางจะต้องนำข้อมูลมาเตรียมรอไว้ก่อน โดยเรียกใช้คำสั่ง FDL_REPLY_UPDATE.request สำหรับกรณีที่มีความผิดพลาดเกิดขึ้นหรือ SSAP ไม่ถูกต้อง สถานีปลายทางจะส่งการตอบรับกลับ (b7=0, Code No 1/2/3/9/12/13)

เมื่อสถานีปลายได้รับข้อมูลที่สถานีต้นทางส่งมาหาสถานีปลายจะได้รับโดย FDL Controller ของต้นปลายทางและใช้ฟังก์ชัน FDL_DATA_REPLY.Indication แจ้งให้ผู้รับที่สถานีปลายทางทราบ

สถานีปลายทางจะส่งข้อมูลกลับทันที ถ้าการตอบกลับไม่ได้รับภายในเวลาที่กำหนด (Slot time T_{SL}) FDL Controller ต้นทางจะส่งข้อมูลอีกครั้งหนึ่ง ถ้าไม่สามารถส่งถึงได้ฟังก์ชัน FDL_DATA_REPLY.confirm แจ้งให้ผู้ใช้งานของสถานีต้นทางหรือถ้าข้อมูลที่สถานีปลายทางส่งข้อมูลที่เตรียมไว้กลับมาหาสถานีต้นทางในเวลาที่กำหนด FDL Controller ของต้นทางจะใช้ฟังก์ชัน FDL_DATA_REPLY.confirm แจ้งให้ผู้ใช้งานของสถานีต้นทางทราบ

สำหรับบริการพื้นฐานของการส่งข้อมูลแบบ SRD จะมีฟังก์ชันเรียกที่ตามมาตรฐานดังต่อไปนี้

1. FDL_DATA_REPLY.request

   (SSAP, DSAP, Rem_add, L_sdu, Serv_class)

   เป็นฟังก์ชันสำหรับส่งและขอข้อมูล (L_sdu) ไปหาเครื่องปลายทาง(Rem_add) (สำหรับ Master)

2. FDL_DATA_REPLY.indication

   (SSAP, DSAP, Loc_add, Rem_add, L_sdu, Serv_class, Update_status)

   เป็นฟังก์ชันเมื่อ FDL ได้รับข้อมูลจากสถานีอื่นที่ส่งเข้ามา จะแจ้งบอกว่าได้รับข้อมูลกับผู้ที่รับข้อมูลพร้อมแจ้งสถานการณ์ทำงาน (สำหรับ Master/Slave)

   Update_status

   - NO ไม่มีการข้อมูล L_sdu ที่ส่งมา
   - LO มีข้อมูลที่ส่งมาและมีความสำคัญต่ำ
   - HI มีข้อมูลที่ส่งมาและมีความสำคัญสูง

3. FDL_DATA_REPLY.confirm

   (SSAP, DSAP, Rem_add, L_sdu, Serv_class, L_status)

   เป็นฟังก์ชันเมื่อ FDL ได้ทำการส่งข้อมูลออกไปแล้วให้ผู้ที่ส่งข้อมูลที่ทราบสถานะการทำงาน (สำหรับ Master)

   L_status

   - DL ข้อมูลที่ส่งมีการตอบรับที่ดีและได้รับข้อมูลตอบกลับมาโดยที่ข้อมูลมีความสำคัญต่ำ
   - DH ข้อมูลที่ส่งมีการตอบรับที่ดีและได้รับข้อมูลตอบกลับมาโดยที่ข้อมูลมีความสำคัญสูง
   - NR ข้อมูลที่ส่งมีการตอบรับที่ดีแต่ไม่ได้รับข้อมูลตอบกลับ
   - RDL ข้อมูลที่ส่งมีการตอบรับที่ไม่ดีและได้รับข้อมูลตอบกลับมาโดยที่ข้อมูลมีความสำคัญต่ำ เนื่องจากทรัพยากรของ FDL Controller ปลายทางไม่พร้อมหรือไม่เพียงพอ
   - RDH ข้อมูลที่ส่งมีการตอบรับที่ไม่ดีและได้รับข้อมูลตอบกลับมาโดยที่ข้อมูลมีความสำคัญสูงเนื่องจากทรัพยากรของ FDL Controller ปลายทางไม่พร้อมหรือไม่เพียงพอ
   - RDH มีการตอบรับที่ไม่ดีและเนื่องจากทรัพยากรของ FDL Controller ปลายทางไม่พร้อมหรือไม่เพียงพอ

4. FDL_REPLY_UPDATE.request

   (SSAP, L_sdu, Serv_class, Transmit)

   เป็นฟังก์ชันสำหรับเตรียมข้อมูลที่ต้องการจะส่ง (L_sdu) ณ ตำแหน่งของ SSAP

   (สำหรับ Master/Slave)

Transmit

Single ข้อมูล L_sdu สามารถอ่านข้อมูลได้ครั้งเดียว

Multiple ข้อมูล L_sdu สามารถอ่านข้อมูลได้หลายครั้ง

5. FDL_REPLY_UPDATE.confirm

(SSAP, Serv_class, L_status)

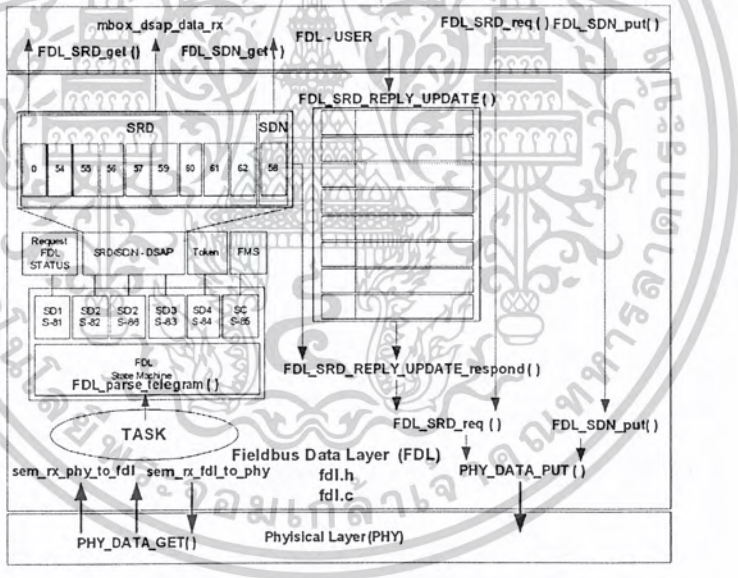
เป็นฟังก์ชันสำหรับตอบรับเมื่อเตรียมข้อมูลที่ต้องการจะส่ง (L_sdu) ณ ตำแหน่งของ SSAP (สำหรับ Master/Slave)

L_status

- OK ข้อมูล L_sdu ได้นำเข้าไปเก็บที่ SSAP แล้ว
- LS บริการที่ SSAP ไม่ทำงาน
- LR ทรัพยากรของสถานีนี้ไม่เพียงพอหรือไม่มี
- IV พารามิเตอร์ที่ใช้ส่งไม่ถูกต้อง

3.2.1.2 การออกแบบส่วน FDL

สำหรับการออกแบบส่วน FDL มีโครงสร้างการทำงานดังรูปที่ 3-18



รูปที่ 3-19 การออกแบบโครงสร้างการทำงานของ FDL

ในส่วนการทำงานของ FDL ทำหน้าที่ในการสร้าง Telegram ส่ง Telegram ที่สร้างให้กับ PHY โดยจะแยกฟังก์ชั่นตามรูปแบบของการบริการเป็น SRD และ SDN นอกจากนี้ยังทำหน้าที่ในการสร้าง Telegram ที่รับมาจาก PHY ด้วย

3.2.1.3 การออกแบบฟังก์ชันในการบริการสำหรับส่งและรับข้อมูลกับ FDL

สำหรับฟังก์ชันที่ได้ออกแบบไว้ในการบริการสำหรับส่งและรับข้อมูลกับ FDL มีฟังก์ชัน คือ การทำงานนี้ตามที่ออกแบบนั้น มี Task ( ) จัดการข้อมูลที่ได้รับเข้ามาจาก PHY ทีละ 1 ไบต์ โดย Task จะรอ

สัญญาณจาก PHY ชื่อ sem_rx_phy_to_fdl และทันทีเมื่อได้รับสัญญาณเข้ามาจะอ่านข้อมูลด้วยฟังก์ชัน PHY_DATA_GET ( ) เก็บในบัฟเฟอร์ของตัวเองชื่อ phy_data_rx ซึ่งเก็บข้อมูลทีละ 1 ไบต์นำมาประกอบเป็น PROFIBUS Telegram ตามรูปแบบโปรโตคอลของโปรฟิบัสด้วยฟังก์ชัน FDL_parse_telegram ( ) โดยจะมีสถานะของ FDL คือ fdl_state เก็บสถานะปัจจุบัน โดยจะใช้ สถานะนี้จัดการจัดเก็บลำดับข้อมูลที่ได้รับเข้ามา สำหรับข้อมูลที่รับเข้ามาเก็บนำมาจะเก็บในตัวแปร telegram_rx ซึ่งได้มีโครงสร้างข้อมูลดังนี้

```
typedef struct
{
INT8U  SD2_sd;                  เก็บค่าเริ่มต้นที่บอกชนิดของ Telegram แบบ sd2
INT8U  SD2_da;                  เก็บค่าตำแหน่งสถานีปลายทาง
INT8U  SD2_sa;                  เก็บค่าตำแหน่งสถานีต้นทาง
INT8U  SD2_fc;                  เก็บค่าควบคุมเฟรม
INT8U  SD2_dsap;                เก็บค่าจุดเข้าถึงบริการตำแหน่งสถานีปลายทาง
INT8U  SD2_ssap;                เก็บค่าจุดเข้าถึงบริการตำแหน่งสถานีต้นทาง
INT8U  SD2_data_unit[MTU_DATA]; เก็บข้อมูลที่ได้รับเข้ามา L_sdu
INT8U  SD2_data_length;         เก็บความยามของข้อมูลนี้
} TELEGRAM_SD2;
```

ถ้าการประกอบ Telegram นั้นไม่มีความผิดพลาดเกิดขึ้นจะส่ง Telegram ไปยังจุดเข้าถึงบริการ (SAP) เพื่อรับข้อมูลและดูแลข้อมูลที่ส่งเข้ามาต่อไป สำหรับข้อมูลที่ได้รับเข้ามาจะมีบริการที่ FDL นี้เกี่ยวข้องอยู่คือต้องส่งข้อมูลไปให้ถูกบริการคือ SRD และ SDN เพื่อแจ้งให้ผู้รับทำการรับข้อมูลซึ่งจะกำหนดโดยหมายเลข SAP และหมายเลขนี้จะมีการกำหนดว่าใช้บริการส่งข้อมูลแบบใด สำหรับอุปกรณ์ Slave โดยทั่วไปนี้ หมายเลข SAP เบอร์ 58 จะเป็นบริการแบบ SDN ส่วนหมายเลข SAP เบอร์ 0, 54-62 ยกเว้น 58 จะเป็นบริการแบบ SRD โดยจะส่งข้อความไปยังผู้รับด้วยตัวแปร mbox_dsap_data_rx ซึ่งเป็น Message Box ไปหาผู้ที่ใช้งาน FDL

สำหรับการนำข้อมูลมาเก็บเตรียมไว้ก่อนของบริการแบบ SRD นั้นจึงต้องออกแบบฟังก์ชันจัดการในส่วนนี้กำหนดเรียกใช้งานฟังก์ชัน FDL_SRD_REPLY_UPDATE พร้อมกับพารามิเตอร์ที่จะต้องป้อนเข้ามาด้วยดังนี้

1. FDL_SRD_REPLY_UPDATE

   (INT8U SRD_SSAP, INT8U *SRD_data_unit, INT8U SRD_data_length,

   INT8U SRD_Serv_class, INT8U SRD_Transmit);

   ฟังชันก์ในการนำข้อมูลที่ต้องจะเก็บเตรียมไว้เพื่อตอบกลับ

   - SRD_SSAP เก็บจุดเข้าให้บริการ (SAP) ของ SRD
   - SRD_data_unit เก็บข้อมูลที่ต้องการจัดเก็บ
   - SRD_data_length เก็บความยาวข้อมูลที่จะจัดเก็บ

- SRD_Serv_class เก็บระดับของความสำคัญ
- SRD_Transmit เก็บลักษณะของข้อมูลนี้สามารถอ่านได้ครั้งเดียวหรือหลายครั้ง

2. INT8U FDL_SRD_REPLY_UPDATE_respond

(INT8U SRD_SSAP, INT8U SRD_DSAP, INT8U SRD_Rem_add);

เป็นฟังก์ชันที่ใช้ตอบข้อมูลกลับไปให้สถานีผู้ส่งที่ได้ขอข้อมูลตอบกลับมาฟังก์ชันนี้จะนำค่าของข้อมูลที่ได้เตรียมไว้ส่งกลับไป ตามจุดให้บริการเข้าถึงข้อมูล (SSAP) ที่ได้เตรียมไว้ก่อนหน้า

โครงสร้างข้อมูลของ SSAP ที่ใช้เก็บข้อมูล

```
typedef struct
{
INT8U   FDL_SSAP_Enable;              เก็บสถานะ (1 = Enable, 0 = Disable)
INT8U   FDL_SSAP_NO;                  เก็บจุดให้บริการ (0-63)
INT8U   FDL_SSAP_data_unit[MTU_DATA]; เก็บข้อมูลที่ต้องการ
INT8U   FDL_SSAP_data_len;            เก็บความยาวของข้อมูล
INT8U   FDL_SSAP_Serv_class;          1 = สำคัญสูง (high) , 1 = สำคัญต่ำ (low)
INT8U   FDL_SSAP_Transmit;            2 = หลายครั้ง, 1 = ครั้งเดียว, 0 = ไม่มี
                                      ข้อมูล
} FDL_SSAP;
```

3. INT8U FDL_SRD_get

(TELEFRAM_SD2 * temp);

เป็นฟังก์ชันของบริการ SRD ในการอ่าน telegram ที่ได้รับมาเก็บไว้หรือนำไปใช้งาน

4. INT8U FDL_SRD_put

(INT8U SRD_SSAP, INT8U SRD_DSAP, INT8U SRD_Rem_add,

INT8U * SRD_data_unit, INT8U SRD_data_length, INT8U SRD_Serv_class);

เป็นฟังก์ชันของบริการ SRD ในการส่งข้อมูล telegram จากสถานีหนึ่งไปยังสถานีหนึ่ง โดยจะภายในจะต้องสร้างส่วน header ของ telegram ส่วนข้อมูล ส่วนความตรวจถูกต้อง ตามรูปแบบของโปรโตคอล

5. INT8U FDL_SDN_get

(TELEFRAM_SD2 * temp);

เป็นฟังก์ชันของบริการ SDN ในการอ่าน telegram ที่ได้รับมาเก็บไว้หรือนำไปใช้งาน

6. INT8U FDL_SDN_put

(INT8U SRD_SSAP, INT8U SRD_DSAP, INT8U SRD_Rem_add,

INT8U * SRD_data_unit, INT8U SRD_data_length, INT8U SRD_Serv_class);

เป็นฟังก์ชันของบริการ SDN ในการส่งข้อมูล telegram จากสถานีหนึ่งไปยังสถานีหนึ่ง โดยจะภายในจะต้องสร้างส่วน header ของ telegram ส่วนข้อมูล ส่วนความตรวจถูกต้อง ตามรูปแบบของโปรโตคอล

7. FDL_parse_telegram

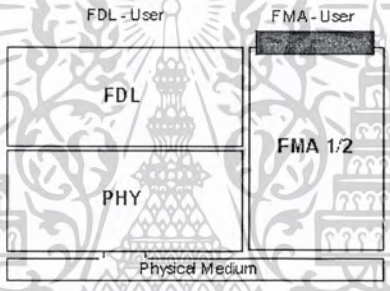
(INT8U phy_data_rx)

เป็นฟังก์ชันที่ใช้ภายใน FDL_TASK เพื่อจัดการการประกอบข้อมูลไบต์เป็นข้อมูล Telegram

INT8U fdl_state ;	ก็บสถานะของการประกอบข้อมูล
INT8U fdl_data_rx_telegram_index;	เก็บตำแหน่งของข้อมูลที่ชี้อยู่
INT8U fdl_data_rx_all;	เก็บความยาวของข้อมูลทั้งหมด
INT8U fdl_data_rx_telegram[MTU_RX];	เก็บข้อมูลทั้ง Telegram

3.2.2 Fieldbus Management (FMA)

3.2.2.1 FMA1/2 User - FMA1/2 Interface



รูปที่ 3-20 การเชื่อมต่อของ FMA1/2 User กับ FMA1/2

บริการที่ต้องเตรียมไว้ใช้ระหว่างผู้ใช้งาน FMA1/2 และ FMA1/2 ต้องมีฟังก์ชันต่อดังนี้

- สั่งการ Reset ในระดับชั้นที่ 1 และ 2
- กำหนดและแก้ไขค่าพารามิเตอร์การทำงานของ FDL และ PHY รวมถึงตัวนับ (Coutner)
- แจ้งเหตุต่าง ๆ ที่เกิดขึ้น (event), ความผิดพลาด (error) และสถานการณ์เปลี่ยนแปลง
- กำหนดค่าจุดเชื่อมต่อเพื่อเข้าถึงบริการ (LSAP) ให้ทำงานหรือยกเลิกการทำงาน

บริการพื้นฐานของเพื่อจัดการการทำงานโดย FMA1/2 User สำหรับอุปกรณ์ Slave ที่ต้องพัฒนามีบริการดังนี้

1. Reset FMA1/2

ฟังก์ชันสั่งให้ Reset ใน FMA1/2 โดยผู้ใช้งาน FMA1/2 สั่งให้อุปกรณ์ทำการเริ่มต้นการทำงานใหม่ (reset) เสมือนกับการเปิดเครื่อง

2. Set Value, Read Value FMA1/2

ฟังก์ชัน Set Value จะทำการกำหนดค่าใหม่ให้กับตัวแปรที่ต้องการ

ฟังก์ชัน Read Value จะทำการอ่านค่าตัวแปรที่ต้องการ

3. Event FMA1/2

   แจ้ง Event ที่เกิดขึ้นของในระดับชั้นที่ 1 และ 2 ให้ผู้ใช้งานทราบ

4. Ident FMA1/2

   ฟังก์ชันบริการแจ้งข้อมูลของตนเอง (Identification) สำหรับสถานีที่เป็น Slave station อาจกำหนดข้อมูลของตัวเองไว้เป็น Hardware หรือ Software

5. LSAP Status FMA1/2

   ผู้ใช้งาน FMA1/2 ใช้ฟังก์ชันนี้เพื่อบอกค่าของบริการจุดเข้าถึงบริการที่ได้กำหนดไว้ภายในตัวเองสำหรับอุปกรณ์ Slave จะใช้ได้เฉพาะ local LSAPs เท่านั้น

6. Live List FMA1/2

   กำหนดรายการของเครื่องที่ทำงานอยู่ (Master เท่านั้น)

7. (R) SAP Activate / SAP Deactivate FMA1/2

   - SAP Activate

     ฟังก์ชันนี้บริการให้กับผู้ใช้งาน FMA1/2 เพื่อกำหนดค่าของจุดเข้าถึงบริการ Link Service Access Point (Local LSAP) สำหรับบริการ FDL แต่ละอันที่ต้องการทำงาน

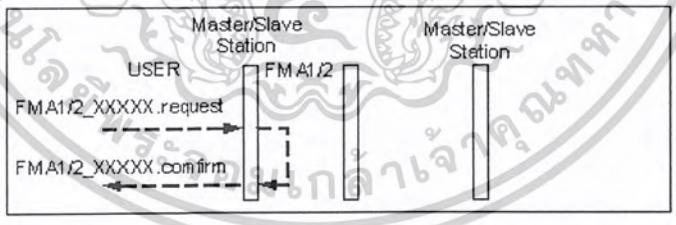
   - RSAP Activate FMA1/2

     ฟังก์ชันนี้บริการให้กับผู้ใช้งาน FMA1/2 เพื่อสั่งให้จุดเข้าถึงบริการตอบกลับทำงาน

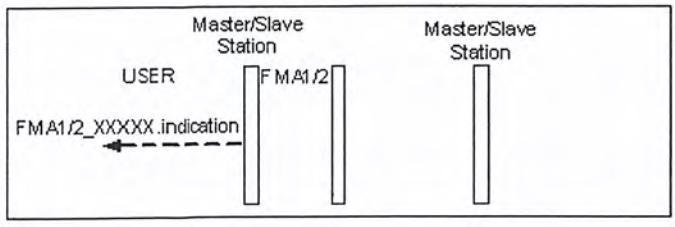
   - SAP Deactivate FMA1/2

     ฟังก์ชันนี้บริการให้กับผู้ใช้งาน FMA1/2 เพื่อกำหนดค่าของจุดเข้าถึงบริการ Link Service Access Point (Local LSAP) สำหรับบริการ FDL แต่ละอันที่ต้องไม่การทำงาน

3.2.2.2 ลักษณะการทำงาน



รูปที่ 3-21 การทำงานของฟังก์ชัน Reset, Set, Read, Ident, LSAP, (R)SAP, SAP



รูปที่ 3-22 การทำงานของฟังก์ชัน Reset, Set, Read, Ident, LSAP, (R)SAP, SAP

3.2.2.3 รายละเอียดการทำงาน

1. Reset FMA1/2 ฟังก์ชันนี้ประกอบด้วยฟังก์ชัน 2 คำสั่งคือ

   - FMA1/2_RESET.request

     (MSAP_0)

     เป็นฟังก์ชันร้องขอให้ทำการรีเซตผ่านจุดให้บริการการจัดการ

   - FMA1/2_RESET.confirm

     (MSAP_0, M_status)

     เป็นฟังก์ชันยืนยันเมื่อได้สั่งให้ทำการรีเซตผ่านจุดให้บริการการจัดการแล้วจะส่งค่าสถานะกลับ

     MSAP_0 คือจุดให้บริการการจัดการ

     M_status

     - OK การสั่งฟังก์ชันรีเซตทำได้สำเร็จ
     - NO การสั่งฟังก์ชันรีเซตทำไม่สำเร็จ
     - IV พารามิเตอร์ที่ส่งมาผิด

2. Set Value FMA1/2, Read Value FMA1/2

   ฟังก์ชันกำหนดค่าและอ่านค่าของตัวแปรเกี่ยวกับการทำงานที่ต้องการ

   ○ Set Value FMA1/2 ฟังก์ชันกำหนดค่าตัวแปรที่ต้องการ

   - FMA1/2_SET_VALUE.request

     (MSAP_0, Variable_name 1 to z, Desired_value 1 to z)

     เป็นฟังก์ชันร้องขอให้กำหนดตัวแปรตามที่ต้องการพร้อมค่าที่ต้องการผ่านจุดให้บริการ

   - FMA1/2_SET_VALUE.confirm

     (MSAP_0, M_status 1 to z)

     เป็นฟังก์ชันยืนยันกลับมาเมื่อมีการร้องขอให้กำหนดตัวแปรตามที่ต้องการพร้อมค่าที่ต้องการผ่านจุดให้บริการ พร้อมแจ้งสถานการณ์กำหนดค่าส่งกลับมา

     Variable_name ตัวแปรที่ต้องการกำหนดค่า

     Desired_value ค่าใหม่ที่ต้องการกำหนด

     M_status

     - OK ตัวแปรได้มีการกำหนดค่าใหม่แล้ว
     - NO ไม่มีตัวแปรที่ต้องการ หรือไม่สามารถกำหนดค่าให้กลับตัวแปรนี้ได้
     - IV พารามิเตอร์ที่ส่งมาผิด

   ○ Read Value FMA1/2

   - FMA1/2_READ_VALUE.request

     (MSAP_0, Variable_name 1 to z)

     เป็นฟังก์ชันร้องขอเพื่ออ่านตัวแปรตามที่ต้องผ่านจุดให้บริการ

- FMA1/2_READ_VALUE.confirm

  (MSAP_0, Current_value 1 to z, M_status 1 to z)

  เป็นฟังก์ชันยืนยันกลับมาเมื่อมีการร้องขออ่านตัวแปรที่ต้องการผ่านจุดให้บริการ พร้อมแจ้งค่าของตัวแปรปัจจุบันและสถานะการอ่านส่งกลับมา

  M_status

  - OK ตัวแปรที่ต้องการสามารถอ่านได้
  - NO ไม่มีตัวแปรที่ต้องการ หรือตัวแปรที่ต้องการไม่ได้มีการกำหนดค่าไว้
  - IV พารามิเตอร์ที่ส่งมาผิด

3. Event FMA1/2

เมื่อ FMA1/2 ได้รับ PHY_EVENT.indiacation หรือได้รับ FDL_Fault.indication จากนั้น FMA1/2 จะทำการส่งค่าผ่าน FMA1/2_EVENT.indication ไปแจ้งกลับผู้ใช้งาน FMA1/2 เพื่อแจ้งเหตุการณ์ที่เกิดขึ้น

- FMA1/2_EVENT.indication

  (MSAP_1, Event/Fault, Add_info)

  เป็นฟังก์ชันแจ้งเหตุการณ์จาก FMA1/2 ไปยังผู้ใช้งาน FMA1/2

  MSAP_1 คือจุดให้บริการการจัดการ

  Event/Fault เป็นค่าที่แจ้งกลับมาอยู่ที่ PHY Event และ FDL Event

  Add_info ข้อมูลเพิ่มเติมที่เกิดจากเหตุการณ์ที่แจ้งเข้ามา

4. Ident FMA1/2

ผู้ใช้งาน FMA1/2 จะเรียกใช้คำสั่งพื้นฐาน FMA1/2_IDENT.request เพื่อขอข้อมูลประจำสถานีนี้สำหรับอุปกรณ์ Slave จากนั้น FDL Layer จะส่งข้อมูลตอบกลับโดยผ่านฟังก์ชัน FDL_IDENT.confirm

สำหรับแบบระยะไกล (Remote) เมื่อมีการร้องขอข้อมูลจาก Master โดย FDL_IDENT (b7=1, Code No 14) ในระดับ FDL Layer จะส่งข้อมูลกลับโดย FDL_IDENT.confirm (b7=0, Code No 1/8/9)

- FMA1/2_IDENT.request

  (MSAP_0, Rem/Loc_add)

- FMA1/2_IDENT.confirm

  (MSAP_0, Rem/Loc_add, Ident_list, M_status)

  Rem/Loc_add กำหนดตำแหน่งสถานีเช่น สถานีปลายทางหรือสถานีตัวเอง (TS)

  Ident_list รายการข้อมูลประจำเครื่องประกอบด้วยค่าดังนี้

  Vendor_name ชื่อของผู้ผลิตหรือผู้ขายอุปกรณ์

  Controller_type ชนิดของอุปกรณ์

  HW/SW_release รุ่นของอุปกรณ์ HW, SW

M_status

- OK การขอข้อมูล สามารถทำได้สำเร็จ
- LR ทรัพยากรของ FDL Controller ไม่พร้อมหรือไม่เพียงพอ
- NA ไม่มีการตอบกลับจากจากสถานีปลายทาง (สำหรับ Master)
- DS FDL/PHY Controller ไม่ได้ Token Ring (สำหรับ Master)
- NR เครื่องปลายทางมีการตอบกลับมาว่าไม่มีข้อมูลหรือไม่พร้อม
- IV พารามิเตอร์ที่ส่งมาผิด

5. LSAP Status FMA1/2

ผู้ใช้งาน FMA1/2 จะใช้ฟังก์ชันพื้นฐาน FMA1/2_LSAP_STATUS.request ในการขอข้อมูลการตั้งค่าให้กับ LSAP (configuration of a LSAP) สำหรับสถานีที่เป็น Master ที่จะขอข้อมูลการตั้งค่าให้กับ LSAP โดยใช้คำสั่ง FMA1/2_LSAP_STATUS.request (b7=1, Code No 15) ไปยังสถานีปลายทางที่เป็น slave จากนั้นสถานีปลายทางจะส่งค่าที่ต้องการกลับมาโดยผ่านฟังก์ชัน FDL_LSAP_STATUS.confirm (b7=0, Code No 1/3/8/9) ของ FMA1/2 ในกรณีที่เป็นการขอข้อมูลการตั้งค่าให้กับ LSAP ในตัวเอง (Local) ก็จะสั่งค่ากลับโดยผ่านฟังก์ชัน FDL_LSAP_STATUS.confirm เหมือนกัน

- FMA1/2_LSAP_STATUS.request
  (MSAP_0, LSAP, Rem/Loc_add)
  ฟังก์ชันในการขอข้อมูล Configuration LSAP

LSAP	จะกำหนดค่าของ LSAP ปลายทางหรือตัวเอง ที่ต้องการขอ Configuration LSAP จะมีค่า 0 ถึง 63 และ NIL ในกรณีที่ไม่มี DAE เฟรม request b7 = 0
Rem/Loc_add	กำหนดตำแหน่งสถานีเช่น สถานีปลายทางหรือสถานีตัวเอง (TS) สำหรับ Global address จะไม่อนุญาตให้ใช้งาน

- FMA1/2_LSAP_STATUS.confirm
  (MSAP_0, LSAP, Rem/Loc_add, Access, Service_type 1 to z, Role_in_service 1 to z, M_status)
  ฟังก์ชันในการส่งข้อมูล Configuration LSAP กลับมายังผู้ขอ

Access	มีค่าตั้งแต่ 0 ถึง 126 หรือทั้งหมด (All) เพื่อระบุตำแหน่งของผู้ที่มีสิทธิในการเข้าถึง
Service_type	กำหนดบริการของ FDL ที่จะทำงานซึ่งมีค่าดังนี้ SDA, SDN, SRD, CSRD
Role_in_service	ระบุค่าของ configuration สำหรับการทำงานของบริการ โดยมีค่าดังนี้
Initiator:	สถานีนี้เป็นผู้เริ่มการติดบริการ
Responder:	สถานีนี้เป็นผู้ตอบสนองบริการ

Both: สถานีนี้เป็นผู้เริ่มการติดบริการและเป็นผู้ตอบสนองบริการ

M_status

- OK การอ่านสถานะของข้อมูลสามารถทำได้
- LR ทรัพยากรของ FDL Controller ไม่พร้อมหรือไม่เพียงพอ
- LS Local LSAP ในตัวเองไม่ทำงาน
- RS Remote LSAP ในเครื่องปลายทางไม่ทำงาน
- NA ไม่มีการตอบกลับจากจากสถานีปลายทาง (สำหรับ Master)
- DS FDL/PHY Controller ไม่ได้ Token Ring หรือขาดการเชื่อมต่อ (Master)
- NR เครื่องปลายทางมีการตอบกลับมาว่าไม่มีข้อมูลหรือไม่พร้อม
- IV พารามิเตอร์ที่ส่งมาผิด

6. Live List FMA1/2

บริการนี้เป็นของ Master เท่านั้น ในที่นี้จะไม่กล่าวถึง

- FMA1/2_LIVE_LIST.request
  (MSAP_0)
- FMA1/2_LIVE_LIST.confirm
  (MSAP_0, Live_list, M_status)

7. (R)SAP Activate FMA1/2, SAP Deactivate FMA1/2

○ SAP Activate FMA1/2

บริการนี้จัดทำเพื่อผู้ใช้งาน FMA1/2 เพื่อสั่งงานและกำหนดค่าให้กับ LSAP ของตัวเอง สำหรับ บริการFDL แต่ละอัน สำหรับบริการผู้ตอบกลับของ SRD, CSRD จะอยู่ในบริการของ RSAP เมื่อมีการเรียกใช้บริการฟังก์ชัน FMA1/2_SAP_ACTIVATE.request จะเป็นการสั่งให้ทำงาน และกำหนดค่าการให้กับ LSAP และจะยืนยันการทำงานโดยฟังก์ชัน FMA1/2_SAP_ACTIVATE แจ้งให้ผู้ใช้งาน FMA1/2

- FMA1/2_SAP_ACTIVATE.request
  (MSAP_2, SSAP, Access, Service_list)

  MSAP_2 คือจุดให้บริการการจัดการ

  SSAP คือจุดให้เข้าถึงบริการที่ต้องการตั้งค่า

  Access มีค่าตั้งแต่ 0 ถึง 126 หรือทั้งหมด (All) เพื่อระบุตำแหน่งของผู้ที่มีสิทธิในการเข้าถึง จะใช้กับฟังก์ชันที่เป็นการตอบสนองเท่านั้น

  Service_list เป็นรายการบริการดังต่อไปนี้ (z มีค่ามากสุดคือ 4)

ลำดับ	ชื่อ	
1	Service_list_length ความยาวของรายการ	1 + 3*Z)
2	Service_activate	1
3	Role_in_service	1

4	L_sdu_length_list	1
5	Service_activate	2
6	Role_in_service	2
7	L_sdu_length_list	2
	. .....	
	. .....	
	. .....	
z	Service_activate	z
z	Role_in_service	z
z	L_sdu_length_list	z

Service_activate เก็บค่าของบริการ ของ FDL มีค่าคือ SDA, SDN, SRD, CSRD

Role_in_service กำหนดค่าบทบาทคือ Initiator, Responder, Both สำหรับ Responds ใช้กับผู้ที่ตอบสนองบริการเท่านั้น

L_sdu_length_list คือ ความยาวมากสุดของ L_sdu ที่จะเก็บ (Max_L_sdu) มีโครงสร้างรายการดังนี้

ลำดับ	ชื่อ
1	Max_L_sdu_length_req_low
2	Max_L_sdu_length_req_high
3	Max_L_sdu_length_ind/con_low
4	Max_L_sdu_length_ind/con_high

- FMA1/2_SAP_ACTIVATE.confirm
  (MSAP_2, SSAP, M_status)
  ฟังก์ชันยืนยันการกำหนดค่าให้กับ SSAP จะส่งค่าสถานะของการทำงานกลับมา
  M_status
  - OK ฟังก์ชันร้องขอกำหนดให้ SSAP ทำงาน
  - NO ฟังก์ชันร้องขอไม่สามารถกำหนดให้ SSAP ทำงาน อาจจะมีการทำงานอยู่แล้ว หรือทรัพยากรไม่เพียงพอหรือไม่พร้อม
  - IV พารามิเตอร์ที่ส่งมาผิด

○ RSAP Activate FMA1/2

ผู้ใช้งาน FMA1/2 เรียกใช้ฟังก์ชันพื้นฐาน FMA1/2_RSAP_ACTIVATE.Request เพื่อสั่งให้ทำงานหรือกำหนดค่า LSAP สำหรับตอบสนอง Reply services เช่น SRD, CSRD เมื่อมีการ

เรียกฟังก์ชันบริการ FDL_RSAP_ACTIVATE.request หลังจากนั้น FDL จะส่ง FMA1/2_RSAP_ACTIVATE.confirm เพื่อยืนยันการทำงาน

- FMA1/2_RSAP_ACTIVATE.request

  (MSAP_2, SSAP, Access, L_sdu_length_list, Indication_mode)

- FMA1/2_RSAP_ACTIVATE.confirm

  (MSAP_2, SSAP, M_status)

  M_status

  - OK SSAP ทำงานตอบสนอง
  - NO SSAP ไม่ทำงานหรือมีการสั่งให้ทำงานแล้วหรือทรัพยากรไม่พอหรือไม่พร้อม
  - IV พารามิเตอร์ที่ส่งมาผิด

○ SAP Deactivate FMA1/2

ผู้ใช้งาน FMA1/2 จะสามารถสั่งให้บริการ FDL หยุดทำงานซึ่งหมายถึง Local LSAP โดยใช้ฟังก์ชัน FMA1/2_SAP_DEACTIVATE.request และจะตอบสนองในสิ่งที่กำลังทำอยู่ หลังจากนั้นจะสั่งให้หยุดทำงานและจะส่งค่ากลับโดยผ่านฟังก์ชัน FDL_SAP_DEACTIVATE.confirm

- FMA1/2_SAP_DEACTIVATE.request

  (MSAP_2, SSAP)

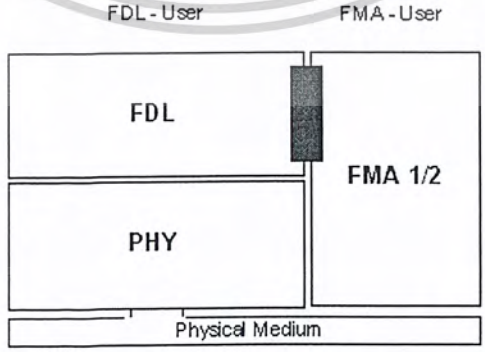
- FMA1/2_SAP_DEACTIVATE.confirm

  (MSAP_2, SSAP, M_status)

  M_status

  - OK SSAP หยุดทำงาน
  - NO SSAP ไม่สามารถหยุดทำงาน
  - IV พารามิเตอร์ที่ส่งมาผิด

3.2.3 FDL - FMA1/2 Interface



รูปที่ 3-23 การเชื่อมต่อของ FDL กับ FMA1/2 Interface

ฟังก์ชันที่ FDL จะต้องเตรียมไว้บริการกับ FMA1/2 มีดังนี้

1. Reset FDL

   FMA1/2 จะใช้บริการนี้เพื่อสั่งรีเซตการทำงานในระดับ FDL หลังจากที่ทำการ reset แล้วจะได้รับการยืนยัน เมื่อได้ทำการรีเซตเสร็จแล้ว

2. Set Value FDL

   บริการนี้จะกำหนดค่าให้กับบริการ FDL เพื่อให้ FMA1/2 กำหนดค่าให้กับ FMA1/2 ตามต้องการแล้วจะได้รับการยืนยัน เมื่อได้ทำการกำหนดค่าแล้วหรืออาจจะไม่สำเร็จ

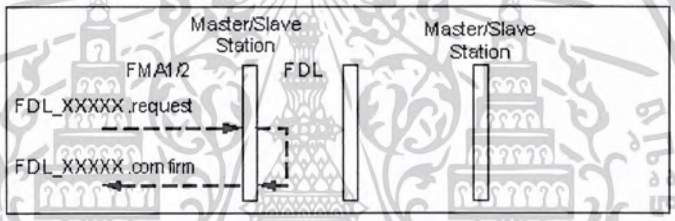
3. Read Value FDL

   บริการนี้จะอ่านค่าตัวแปรปัจจุบันให้ของบริการ FDL แล้วจะตอบส่งค่ากลับพร้อมค่าที่ต้องการอ่าน

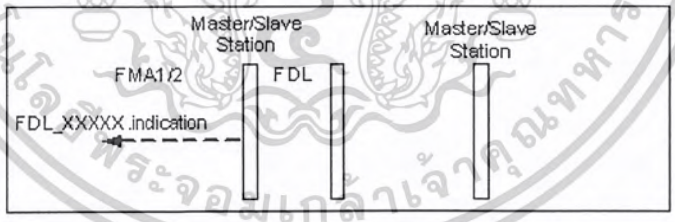
4. Fault FDL

   ในชั้น FDL จะใช้บริการนี้แจ้งความผิดพลาดหรือสิ่งที่เกิดขึ้นไปบอก FMA1/2

3.2.3.1 ลักษณะการทำงาน



รูปที่ 3-24 การทำงานของฟังก์ชันReset FDL, Set Value FDL, Read Value



รูปที่ 3-25 การทำงานของฟังก์ชัน Fault FDL

3.2.3.2 รายละเอียดการทำงาน

1. Reset FDL

   เมื่อต้องการรีเซตการทำงานของ FDL Layer จะใช้คำสั่งพื้นฐาน FDL_RESET.request จะเสมือนกับทำการเปิดเครื่องใหม่ สถานะจะเป็น Offline และค่าตัวแปรทุกตัวจะถูกลบค่า หลังจากที่ทำการ reset จะทำการ หลังจะทำการ reset การทำงาน FDL จะใช้ฟังก์ชัน FDL_RESET.confirm ส่งค่ากลับมา

   - FDL_RESET.request

- FDL_RESET.confirm

  ฟังก์ชันทั้ง 2 คำสั่งไม่มีพารามิเตอร์

2. Set Value FDL, Read Value FDL

   ○ Set Value FDL

   บริการกำหนดค่าตัวแปรให้กับ FDL จะเรียกใช้โดย FMA1/2 ผ่านของ FDL ฟังก์ชันดังต่อไปนี้

   - FDL_SET_VALUE.request

     (Variable_name, Desired_value)

     ฟังก์ชันกำหนดค่าให้กับตัวแปร FDL

     Variable_name ระบุตัวแปรของ FDL ที่ต้องการจะกำหนดค่า

     Desired_value ระบุค่าที่ต้องการกำหนดให้กับตัวแปร

   - FDL_SET_VALUE.confirm

     ฟังก์ชันแจ้งการกำหนดให้กลับตัวแปร

     Variable_name ชื่อตัวแปรที่มีใน FDL มีดังต่อไปนี้

     ตัวแปรสำหรับการปฏิบัติงานของระบบ (Operating Parameters)

TS	เป็นค่าตำแหน่งของสถานีนี้ (FDL Address)
Baud_rate	เป็นค่าความเร็วในการส่งข้อมูลของระบบ
Medium_red	มีการใช้การเข้าถึงตัวกลางหลายอัน
HW-Release	ชื่อและรุ่นของอุปกรณ์
SW-Release	ชื่อซอฟแวร์และรุ่นที่ใช้
T_{SL}	เวลาที่ใช้ในการอ่านอุปกรณ์นี้ (Slot Time)
min T_{SDR}	เวลาที่น้อยที่สุดที่อุปกรณ์นี้จะล่าช้า

     ตัวแปรสำหรับการนับการทำงาน

SD_count	จำนวนของข้อมูลตัวเริ่มต้นที่ถูกต้อง (Start Delimiters)
SD_error_count	จำนวนของข้อมูลตัวเริ่มต้นที่ผิดพลาด (Start Delimiters)

   ○ Read Value FDL

   การอ่านค่าตัวแปรที่ต้องการภายใน FDL สามารถทำได้โดยใช้ฟังก์ชัน FDL_READ_VALUE.request เพื่ออ่านค่าโดยจะนำค่าสถานะปัจจุบันส่งกับด้วยฟังก์ชันพื้นฐาน FDL_READ_VALUE.confirm ซึ่งมีการเรียกใช้งานดังนี้

   - FDL_READ_VALUE.request

     (Variable_name)

   - FDL_READ_VALUE.confirm

     (Current_value)

3. Fault FDL

เป็นบริการพื้นฐานที่แจ้งไปยัง FMA1/2 ด้วยฟังก์ชันของ FDL ชื่อ FDL_FAULT.indication เพื่อแจ้งความ ผิดพลาดที่เกิดขึ้น

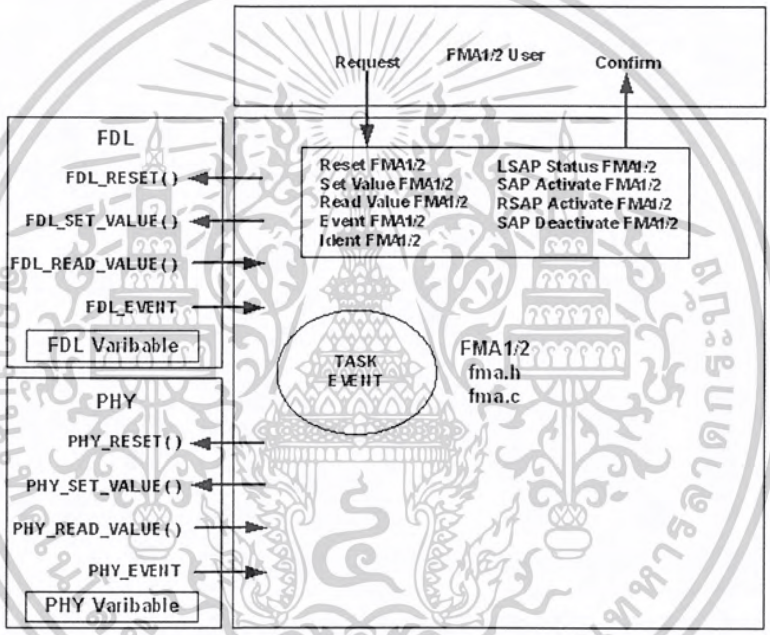
- FDL_FAULT.indication

(Fault_type)

Fault_type	หมายถึงชนิดความผิดพลาดที่เกิดขึ้น สำหรับ slave มีดังนี้
Time_out	ไม่มีกิจกรรมใด ๆ เกิดขึ้น
Not_syn	การ Synchronization ไม่พบภายในช่วงเวลา Syn-Interval-Time T_{SYNI}

3.2.4 สรุปการออกแบบส่วน Layer 2

สรุปการออกแบบการทำงานของส่วน Layer 2 ทั้งหมด ดังนี้

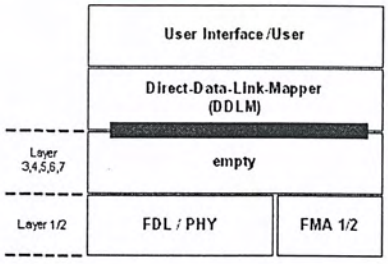


รูปที่ 3-26 การออกแบบโครงสร้างการทำงานของส่วน Layer 2

การออกแบบในระดับ Layer 2 แบ่งออกเป็น 2 ส่วน คือ FDL ทำในส่วนของการสร้าง และประกอบ Telegram และ FMA จะทำหน้าที่ในการจัดการเกี่ยวค่าของตัวแปร โดยจะส่งผ่านไปยัง PHY อีชั้นหนึ่ง โดยFDL จะมีการเรียกใช้งานส่วนของ FMA ทางด้านการจัดการตัวแปร

3.3 ส่วนติดต่อกับชั้น Data link โดยตรง DDLM (Direct Data Link Mapper)

3.3.1 การเชื่อมต่อระหว่าง DDLM และชั้นที่ 2 (Interface between DDLM and Layer 2)



รูปที่ 3-27 การเชื่อมต่อของ DDLM กับ Layer 2

DDLM จะทำการติดต่อกับ FDL โดยผู้ใช้งานส่วน FDL ก็คือส่วน DDLM และจะเป็นผู้ใช้งานส่วน FDL ในส่วนของ FMA ผู้ใช้งานส่วน FMA ก็คือ DDLM สำหรับ DDLM จะเป็นผู้ติดต่อระหว่าง User-Interface กับ Layer 2 โดยจะจัดการบริการบน Layer ให้กับ User-Interface ดังนั้นฟังก์ชันของ DDLM จะติดต่อกับ Layer 2 เพื่อเรียกใช้บริการดังต่อไปนี้

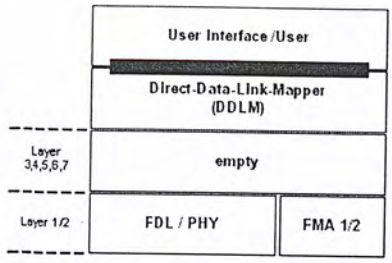
1. ใช้บริการของ FDL Service ได้แก่
   - Send and Request Data with Reply (SRD) บริการส่งและขอข้อมูลพร้อมด้วยการตอบกลับ
   - Send Data with No Acknowledge (SDN) บริการส่งข้อมูลโดยไม่มีการตอบรับ
2. ใช้บริการของ FMA1/2 services ได้แก่
   - Reset FMA1/2
   - Set Value FMA1/2
   - Read Value FMA1/2
   - Event FMA1/2
   - SAP Activate FMA1/2
   - RSAP Activate FMA1/2
   - SAP Deactivate FMA1/2

การติดระหว่างฟังก์ชันของ DDLM ไปยังชั้นที่ 2 เรียกใช้งานการบริการ (SSAP) ดังต่อไปนี้

คำสั่ง DDLM พื้นฐาน	SSAP	DSAP	Layer 2 Service	Serv_class
DDLM_Data_Exchange.req/.ind	NIL	NIL	SRD	high
DDLM_Data_Exchange.con				low/high
DDLM_Data_Exchange_Upd.req				low/high
DDLM_Chk_Cfg.req/.ind	62	62	SRD	high
DDLM_Chk_Cfg.con				
DDLM_Set_Prm.req/.ind	62	61	SRD	high
DDLM_Set_Prm.con				
DDLM_Slave_Diag.req/.ind	62	60	SRD	high
DDLM_Slave_Diag.con				low
DDLM_Slave_Diag_Upd.req				low
DDLM_Get_Cfg.req	62	59	SRD	high
DDLM_Get_Cfg.con				low
DDLM_Get_Cfg_Upd.req				low
DDLM_Global_Control.req/.ind	62	58	SDN	high
DDLM_RD_Outp.req	62	57	SRD	high
DDLM_RD_Outp.con				low
DDLM_RD_Outp_Upd.req				low
DDLM_RD_Inp.req	62	56	SRD	high
DDLM_RD_Inp.con				low
DDLM_RD_Inp_Upd.req				low
DDLM_Set_Slave_Add.req/.ind	62	55	SRD	high
DDLM_Set_Slave_Add.con				

ตารางที่ 3-1 การติดต่อระหว่างฟังก์ชันของ DDLM กับ Layer ผ่านจุด SSAP

3.3.2 การเชื่อมต่อระหว่าง DDLM และ User-Interface



รูปที่ 3-28 การเชื่อมต่อของ DDLM กับ User-Interface

อุปกรณ์ DP-Slave จะต้องตอบสนองฟังก์ชันที่เรียกโดย DP-Master ดังนั้นฟังก์ชันที่ต้องพัฒนาสำหรับ DP-Slave ประกอบด้วยดังต่อไปนี้

1. Data_Exchange (บังคับ)
2. RD_Inp (บังคับ)
3. RD_Outp (บังคับ)
4. Slave_Diag (บังคับ)
5. Set_Prm (บังคับ)
6. Chk_Cfg (บังคับ)
7. Get_Cfg (บังคับ)
8. Global_Control (บังคับ)
9. Set_Slave_Add (ตัวเลือก)

สถานะของฟังก์ชันเมื่อเรียกใช้งานมีดังต่อไปนี้

1. OK การตอบรับทราบเป็นบวก
2. IV พารามิเตอร์ที่ส่งมาร้องขอผิดพลาด
3. NO บริการในสถานะนี้ไม่สามารถทำได้หรือเป็นไม่ได้
4. DS ไม่ได้ token ring หรือขาดการเชื่อมต่อ (master)
5. NA การตอบรับทราบเป็นลบ หรือไม่มีการกระทำใดเกิดขึ้น
6. TO ฟังก์ชันนี้หมดเวลาทำงาน (time-out)

3.3.3 ฟังก์ชันการทำงานระหว่าง DP-Master กับ DP-Slave

3.3.3.1 การอ่านค่าข้อมูลวินิจฉัยของ DP-Slave จาก DP-Master

ข้อมูลวินิจฉัยจะเก็บภายใน Data_Unit ภายในข้อมูลนั้นข้อมูลถ้าBitใดมีการเซตหมายความว่ามีเหตุการณ์นั้นเกิดขึ้น ตามตำแหน่งบิตต่าง ๆ ข้อมูลวินิจฉัยประกอบด้วย 2 ส่วนคือ ข้อมูลวินิจฉัยมาตรฐาน (standard diagnostic information) เก็บในByte ที่ 1 ถึง Byte ที่ 6 และส่วนที่ 2 คือส่วนข้อมูลวินิจฉัยเพิ่มเติม extended diagnostic information (Ext_Diag_Data)

1. DDLM_Slave_Diag ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

   - DDLM_Slave_Diag.indication

     (Req_Add)

     เป็นฟังก์ชั่นที่รับมาจาก อุปกรณ์รองในการร้องขอข้อมูลวินิจฉัย

     Req_Add เป็นค่าที่อยู่ของอุปกรณ์หลักที่ทำการร้องขอ

   - DDLM_Slave_Diag_Upd.request

     (Diag_Data)

     เป็นฟังก์ชั่นที่ทำการเปลี่ยนแปลงข้อมูลวินิจฉัย

     Diag_Data ข้อมูลวินิจฉัยประกอบไปด้วยข้อมูลพื้นฐานขนาด 6 ไบต์

3.3.3.2 การส่งและรับข้อมูล Input และ Output

ฟังก์ชันเหล่านี้จะให้ Local user ของ DP-Master ส่งข้อมูล Output ไปยัง DP-Slave และให้ DP-Slave ส่งข้อมูล Input กลับมาหา DP-Master จำนวนของ Input และ output จะกำหนดไว้ตอนที่ทำการตั้งค่าข้อมูลในช่วงเฟสเริ่มทำงาน ถ้ามีข้อมูลวินิจฉัยหรือความผิดพลาดเกิดขึ้นที่ DP-Slave จะส่งข้อมูลตอบสนองก่อนที่จะเฟรมข้อมูล

1. DDLM_Data_Exchange ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงาน ดังต่อไปนี้

   - DDLM_Data_Exchange.indication

     (Outp_Data)

     เป็นฟังก์ชั่นที่รับข้อมูลมาจากอุปกรณ์หลัก

     Outp_Data ข้อมูลจากอุปกรณ์หลัก

   - DDLM_Data_Exchange_Upd.request

     (Diag_Flag, Input_Data)

     เป็นฟังก์ชั่นที่คอยส่งข้อมูลกลับไปยังอุปกรณ์หลัก

     Input_Data ข้อมูลที่จะส่งไปยังอุปกรณ์หลัก

     Diag_Flag เป็นค่าที่เกี่ยวข้องกับการตรวจสอบว่าข้อมูลวินิจฉัยได้ถูกอุปกรณ์นำเข้ามาค่าที่เป็นไปได้ คือ

     True ข้อมูลวินิจฉัยมีอยู่

     False ไม่มีข้อมูลวินิจฉัยมีอยู่

3.3.3.3 การอ่านข้อมูล Input และ Output ของ DP-Slave

เป็นฟังก์ชันของ DP-Master (class 2) ในโหมดที่ทำการแลกเปลี่ยนข้อมูล

1. DDLM_RD_Inp ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

   - DDLM_RD_Inp_Upd.request

     (Input_Data)

     เป็นการแสดงว่าอุปกรณ์รองพร้อมที่จะทำการแลกเปลี่ยนข้อมูล

2. DDLM_RD_Outp ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

- DDLM_RD_Oup_Upd.request

  (Outp_Data)

  เป็นการแสดงว่าอุปกรณ์รองพร้อมที่จะทำการแลกเปลี่ยนข้อมูล

3.3.3.4 การส่งข้อมูลเกี่ยวกับพารามิเตอร์

หมายความว่า DP-Slave จะได้รับข้อมูลพารามิเตอร์ การกำหนดพารามิเตอร์กับสถานี passive จะทำในเฟสเริ่มทำงานหรือเฟสแลกเปลี่ยนข้อมูล

1. DDLM_Set_Prm ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

   - DDLM_Set_Prm.indication

     (Req_Add,Prm_Data)

     ปืนฟังก์ชั่นตั้งค่าตัวแปร

     rm_Data เป็นข้อมูลการตั้งค่าตัวแปร โดยอุปกรณ์หลักจะส่งมาให้อุปกรณ์รอง เพื่อทำการกำหนดค่าต่างๆ ให้ตรงกัน โดยพื้นฐานจะประกอบไปด้วยข้อมูลขนาด 7 ไบต์

3.3.3.5 การตรวจสอบข้อมูลที่ตั้งไว้ (Configuration Data)

ฟังก์ชันนี้จะอนุญาตให้ DP-Master ส่งข้อมูล configuration ไปยัง DP-Slave เพื่อตรวจสอบภายในข้อมูลนั้นจะเก็บช่วงของ input และ output เพื่อตรวจสอบความถูกต้องของข้อมูล

DP-Slave จะทำการเปรียบเทียบข้อมูลที่ตั้งไว้จริง (Real_Cfg_Data) กับค่าข้อมูลที่ได้ที่ตั้งไว้ (Cfg_Data) ที่ที่รับมาจากสถานีมี่เป็น DP-Master การทำการตรวจสอบค่าที่ตั้งไว้ (Configuration) จะดูรูปแบบและความยามของข้อมูล input/output ถ้าการตรวจสอบผิดพลาดจะทำการแจ้งโดยการเซตบิต Diag.Cfg_Fault ของข้อมูลวินิจฉัย มีกรณีที่เป็นไปได้ดังนี้

- DP-Slave บอกว่าบริเวณของข้อมูลนั้นถูกต้องแต่ DP-Master บอกว่าไม่ถูกต้อง
- DP-Slave ไม่สามารถหาความถูกต้องในบริเวณข้อมูลนั้นได้แต่ DP-Master ต้องการความถูกต้องที่ข้อมูลนั้น

1. DDLM_Chk_Cfg ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

   - DDLM_Chk_Cfg.indication

     (Req_Add, Cfg_Data)

     เป็นฟังก์ชั่นที่ทำหน้าที่ในการตรวจสอบข้อมูลองค์ประกอบที่ใช้ในกำหนดข้อตกลงในการรับ-ส่งข้อมูล

     Cfg_Data ข้อมูลองค์ประกอบที่รับมาจากอุปกรณ์รอง

3.3.3.6 การอ่านค่าข้องข้อมูลที่ตั้งไว้ (Configuration Data)

ฟังก์ชันนี้อนุญาตให้ผู้ใช้งานอ่านข้อมูล Configuration ของ DP-Slave โดยจะได้รับข้อมูลที่ตั้งไว้จริง ๆ ที่ DP-Slave (Real_Cfg_Data)

1. DDLM_Get_Cfg ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงานของอุปกรณ์รอง ดังต่อไปนี้

   - DDLM_Get_Cfg_Upd.request

(Real_Cfg_Data)

เป็นการตั้งค่าข้อมูลองค์ประอบของอุปกรณ์รอง

Real_Cfg_Data ข้อมูลองค์ประกอบของอุปกรณ์รอง

3.3.3.7 คำสั่งควบคุมไปยัง DP-Slave

ฟังก์ชันนี้อนุญาตให้ส่งคำสั่งพิเศษจาก DP-Master ไปหา DP-Slave สถานีเดียวหรือหลาย ๆ สถานี ใช้สำหรับการสั่งงาน DP-Slave ให้ทำงานเป็นจังหวะพร้อม ๆ กัน (synchronization) โดย DP-Slave จะทำงานกับ DP-Master ที่ได้มีการกำหนดไว้เท่านั้น โดยกำหนดไว้ตอนที่กำหนดข้อมูลพารามิเตอร์และข้อมูลที่ตั้งไว้ (configuration) โดย DP-Master จะใช้คำสั่งควบคุมกำหนดโหมดการทำงานของ slave

1. DDLM_Global_Control ประกอบด้วยคำสั่งที่เกี่ยวข้องกับการทำงาน ดังต่อไปนี้

   - DDLM_Global_Control.indication

   (Req_Add,Control_Command,Group_Select)

   เป็นฟังก์ชั่นที่รับคำสั่งการทำงานพิเศษมาจากอุปกรณ์หลัก

   Control_Command ชุดคำสั่งพิเศษที่ต้องการให้ทำงานเฉพาะ

   Group_Select กลุ่มของอุปกรณ์ที่ต้องทำคำสั่งพิเศษ

3.3.3.8 คำสั่งกำหนดหมายเลขตำแหน่ง (Address) ให้กับ DP-Slave

ฟังก์ชันนี้จะให้ DP-Master (class 2) เป็นผู้เปลี่ยนหมายเลขตำแหน่ง (Address) ให้กับ DP-Slave ในกรณีที่ DP-Slave ไม่สามารถเก็บข้อมูลในหน่วยความจำ EEPROM หรือ FLASH สำหรับ DP-Slave ที่กำหนดด้วย switch ฟังก์ชันนี้จะถูกปฏิเสธด้วย RS Error message

สำหรับการเปลี่ยนหมายเลขตำแหน่ง (Address) จะใช้วิธีการตรวจสอบ Ident_Number ด้วยเพื่อตรวจสอบสถานีว่าตรงตามที่ต้องการหรือไม่โดยจะส่ง Ident_Number ไปในฟังก์ชัน

1. DDLM_Set_Slave_Add เป็นคำสั่งเพิ่มเติมประกอบด้วยคำสั่งดังต่อไปนี้

   - DDLM_Set_Slave_Add.indication

   (New_Slave_Add, Ident_Number, No_Add_chg)

   เป็นฟังก์ชั่นรับค่าการตั้งค่าที่อยู่ใหม่จากอุปกรณ์รอง

   New_Slave_Add ที่อยู่ใหม่ของอุปกรณ์รอง

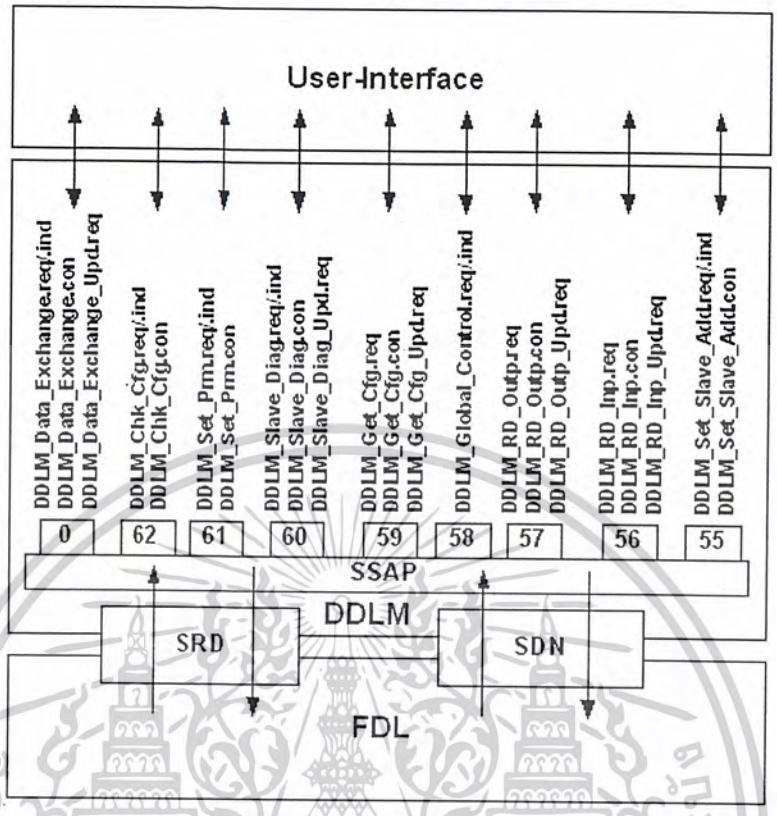
   Ident_Number หมายเลขประจำตัวของอุปกรณ์รอง

   No_Add_chg ค่ากำหนดการเปลี่ยนแปลงที่อยู่ค่าที่เป็นไปได้ คือ

   True ต้องเปลี่ยนจากการเริ่มต้นทำงานเท่านั้น

   False เปลี่ยนได้ตลอด

3.3.4 การเชื่อมต่อระหว่าง DDLM และ User-Interface และ FDL



รูปที่ 3-29 การออกแบบการเชื่อมต่อระหว่าง DDLM และ User-Interface และ FDL

การออกแบบการทำงานของ DDLM จะเป็นลักษณะของการ Call Function จาก User Interface โดยจะมี Function ที่เรียกใช้ต่างกันตามลักษณะของการการบริการ คือ SRD และ SDN ก่อนที่จะส่งผ่านการทำงานยังชั้นของ FDL

3.3.5 ฟังก์ชันภายในของ DP-Slave

การเชื่อมต่อระหว่างส่วนติดต่อผู้ใช้งาน (User-Interface) ของ DP-Slave จะมีการสร้างฟังก์ชันสำหรับติดต่อภายในระหว่างส่วนติดต่อผู้ใช้งาน (User-Interface) กับ the DDLM

DDLM จะทำการวางฟังก์ชันไปยังบริการของ FDL และ FMA1/2 และDDLM จะกระทำการจัดการบริการภายในตัวเองโดยเป็นอิสระ (independent) จาก FDL และ FMA1/2 เมื่อเกิดความผิดพลาดหรือมีเหตุการณ์เกิดขึ้นจากบริการของ FDL และ FMA1/2 จะมีการจัดการโดย DDLM_Fault.ind สำหรับ DDLM เอง

ฟังก์ชันภายในสำหรับ DP-Slave มีดังต่อไปนี้

1. DDLM_Slave_Init

ฟังก์ชันนี้จะจัดการกับ FDL และ FMA1/2 ภายในตัวเอง โดยจะทำการกำหนดค่าเริ่มต้นต่าง ๆ ในการสื่อสารหรือส่งข้อมูลและหมายเลขสถานี โดยจะทำการเรียกฟังก์ชันที่เป็นบริการของ FMA1/2 ดังนี้

- Reset FMA1/2
- Set Value FMA1/2
- RSAP Activate FMA1/2

2. DDLM_Set_minTsdr

ฟังก์ชันกำหนดค่าพารามิเตอร์ minTsdr กำหนดค่าเวลาที่น้อยที่สุดที่ station นี้จะล่าช้า (delay time) เมื่อทำการตอบสนอง โดย DDLM จะมีการเรียกใช้บริการของ FMA1/2 ฟังก์ชันดังต่อไปนี้

- Set Value FMA1/2

3. DDLM_Enter

ฟังก์ชันนี้จะใช้ฟังก์ชันของ FDL และ FMA1/2 เพื่อกำหนดค่าเริ่มต้นของ DP-Slave เพื่อที่จะเข้าสู่โหมดแลกเปลี่ยนข้อมูลของ DDLM จะทำการกำหนดค่า SAP สำหรับข้อมูล output และฟังก์ชันDDLM_Global_Control จะทำงานและกำหนดค่าพารามิเตอร์ในชั้นที่ 2 จะถูกป้องกัน ซึ่งจะเรียกใช้บริการของ FMA1/2 ฟังก์ชันดังต่อไปนี้

- RSAP Activate FMA1/2
- SAP Activate FMA1/2

4. DDLM_Leave

ฟังก์ชันนี้จะใช้ฟังก์ชันของ FDL และ FMA1/2 เพื่อแก้ไขค่าใหม่อีกครั้ง โหมดการแลกเปลี่ยนข้อมูลของ DDLM จะหยุดทำงาน ในขณะเดียวกันการเข้าถึง Layer 2 จะถูกป้องกัน การกำหนดค่า SAP สำหรับ Output data และ ฟังก์ชัน DDLM_Global_Control จะหยุดทำงาน สำหรับฟังก์ชันจะไม่มี Input พารามิเตอร์ ซึ่งจะเรียกใช้บริการของ FMA1/2 ฟังก์ชันดังต่อไปนี้

- SAP Deactivate FMA1/2

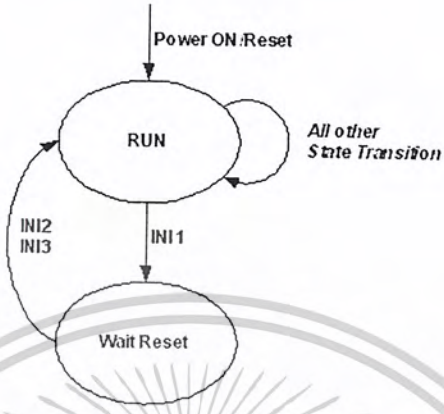
5. DDLM_Fault

ฟังก์ชันนี้จะเป็นการแจ้งความผิดพลาดที่เกิดขึ้นใน DP-Slave จากส่วน DDLM ไปยังส่วนติดต่อผู้ใช้งาน (User-Interface) ในขณะที่มีการเรียกใช้บริการ FDL หรือ FMA1/2 เมื่อมีการเกิดความผิดพลาดดังต่อไปนี้

- FMA1/2_RESET.con (NO/IV)
- FMA1/2_SET_VALUE.con (NO/IV)
- FMA1/2_RSAP_ACTIVATE.con (NO/IV)
- FMA1/2_SAP_ACTIVATE.con (NO/IV)
- FMA1/2_SAP_DEACTIVATE.con (NO/IV)
- FDL_REPLY_UPDATE.con (LS/LR/IV)
- FDL_DATA_REPLY.ind (DSAP ไม่สามารถรับได้)

- FDL_DATA.ind (DSAP ไม่สามารถรับได้)
- ไม่รู้จักคำสั่งพื้นฐาน FDL หรือ FMA1/2

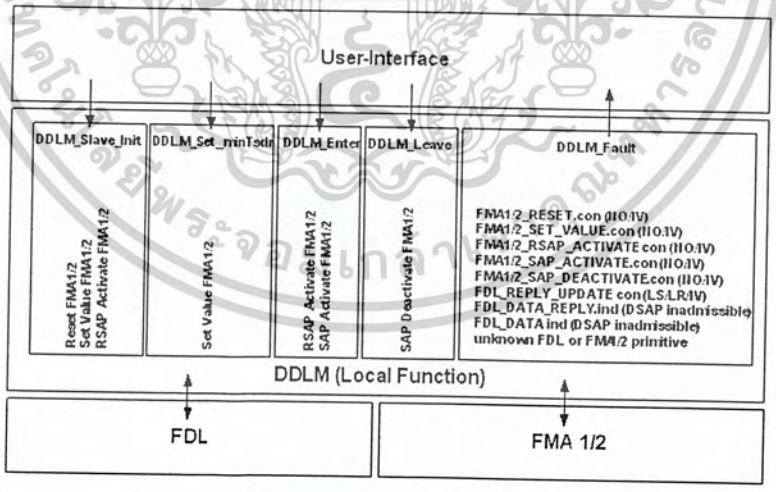
3.3.6 สถานะการทำงานของ DDLM



รูปที่ 3-30 State Diagram ของ DDLM ใน DP-Slave

โดยปกติ DDLM ของ DP-Slave จะอยู่ในสถานะ "RUN" ขณะที่ทำการกำหนดค่าเริ่มต้นของ DDLM และ FDL โดยฟังก์ชัน DDLM_Slave_Init.req (Station_Address) สถานะจะเปลี่ยนเป็นสถานะ "WAIT RESET" หลังจากนั้นจะกลับมาอยู่ในสถานะ "RUN" เมื่อมีการกำหนดค่าครั้งถัด ๆ ไป

3.3.7 การออกแบบการเชื่อมต่อระหว่างฟังก์ชั่นของ DDLM และ User-Interface และ FDL

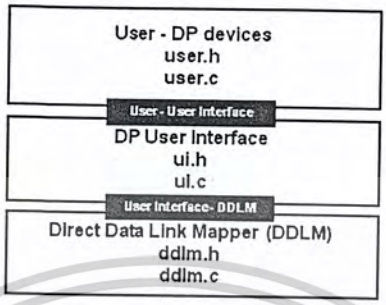


รูปที่ 3-31 การเชื่อมต่อระหว่างฟังก์ชั่นของ DDLM และ User-Interface และ FDL

การ Call Function ระหว่าง Layer ของ DDLM มีการทำงานแยกออกเป็นส่วนคือ DDLM_Slave_Init, DDLM_Set_MinTSRD, DDLM_Enter, DDLM_leave และ DDLM_Fault

3.4 ส่วนติดต่อผู้ใช้งาน (User-Interface)

ส่วนติดต่อผู้ใช้งานจะติดต่อกับ DDLM ผ่านทางฟังก์ชันของ DDLM ที่ได้ที่ได้กล่าวมาแล้ว การใช้งานจริง ๆ จะเหมือนกับการเชื่อมต่อผ่านข้อมูล สถานะการทำงานของ DP-slave จะปฏิบัติงานโดย ดูจากข้อมูลที่ส่งผ่านเข้ามายัง DP-slave สำหรับ DP-Master จะรับประกันช่วงเวลาระหว่างการส่งข้อมูลกับ Slave ภายในรายการของตัวเอง



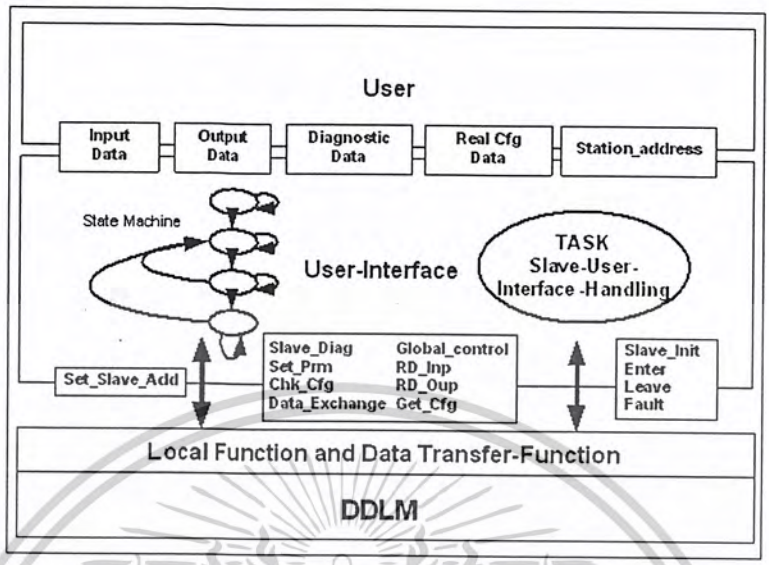
รูปที่ 3-32 การเชื่อมต่อของ User-Interface และ DDLM

3.4.1 ส่วน User-Interface DDLM

เป็นส่วนที่ใช้ติดต่อระว่าง User และ DDLM โดยจะเป็นการทำงานลักษณะการ call Function ผ่านค่าจากที่ User เรียก ใช้ แล้วส่งต่อ ให้ Function ใน ระดับ DDLM ในการ เป็น Function ที่ใช้สำหรับการ set ค่าตัวแปรต่างๆ ที่เกี่ยวกับอุปกรณ์รอง ให้เป็นไปตามสถานะการทำงานของ slave device โดยจะมี function ที่เกี่ยวข้องดังตัวอย่างต่อไปนี้

1. UI_Set_Prm_ind (Req_Add, Prm_Data)
   เป็น Function ที่ถูกใช้งานขณะที่อุปกรณ์อยู่ในสถานะ Set Parameter โดยจะทำการ set ค่า Prm_Data ให้ตรงกับ Mater ซึ่งก็คือ Req_Add
2. UI_Chk_Cfg_ind (Req_Add, Cfg_Data)
   เป็น Function ที่ถูกใช้งานขณะที่มีการตรวจสอบข้อมูล Configuration Data จาก Master เพื่อที่จะมีรูปแบบการแลกเปลี่ยนข้อมูลที่ตรงกันกับอุปกรณ์
3. UI_Data_Exchange_ind (Outp_Data)
   เป็นFunction ที่ทำงานในสถานการณ์แลกเปลี่ยนข้อมูล โดยที่เป็นส่วนที่รับข้อมูลมากจาก Master เพื่อที่จะนำข้อมูลไปส่งให้กับ Module อุปกรณ์ต่างๆ ที่เชื่อมต่ออยู่
4. UI_Slave_Diag_ind (Req_Add)
   เป็น Function ที่รับมาจาก Master เพื่อตรวจสอบสถานะของอุปกรณ์ต่าง ๆเพื่อตรวจสอบความพร้อม ในการสื่อสาร
5. UI_Global_Control_ind (Req_Add, Control_Command, Group_Select)
   เป็น คำสั่งจาก Master ในการสั่งการทำงาน พิเศษเฉพาะอย่าง ใน Control_Command โดยอุปกรณ์ที่มีค่า Group_Select ตรงกับที่กำหนด จะต้องทำงานตามคำสั่งพิเศษนี้
6. UI_Set_Slave_ind (New_Slave_Add, Ident_Number, No_Add_Chg)

เป็นคำสั่งที่ใช้ในการ Set ที่อยู่ของอุปกรณ์รองให้อยู่ ระหว่าง 0-255 เพื่อที่จะสามารถ ข้ามสถานะของการทำงานไปยัง สถานะรอการ set Parameter ได้ โดยการ Set New Address จะกระทำผ่าน Master Class 2



รูปที่ 3-33 การออกแบบการเชื่อมต่อของ DDLM และ User-Interface และ User

การออกแบบการทำงานในส่วนของ User Interface จะมีการใช้ task ควบคุมการทำงานให้เป็นไปตามสถานะของการทำงาน Slave Device และส่งผ่านการทำงานไปยัง Layer ของ DDLM

ความผิดพลาดที่เกิดขึ้นจาก FDL และ DDLM จะวิเคราะห์ด้วยดูจาก DDLM_Fault.ind การพัฒนาส่วนเชื่อมต่อกับผู้ใช้งานและการนำไปประยุกต์ใช้ในกระบวนการจริง จำเป็นต้องมีการติดต่อระหว่างกันและทำงานร่วมกันหรือการ synchronization การพัฒนานี้จึงไม่เป็นอิสระต่อกัน สำหรับเหตุการณ์ที่เกิดขึ้นภายในตัวเองอาจจะเกิดจาก

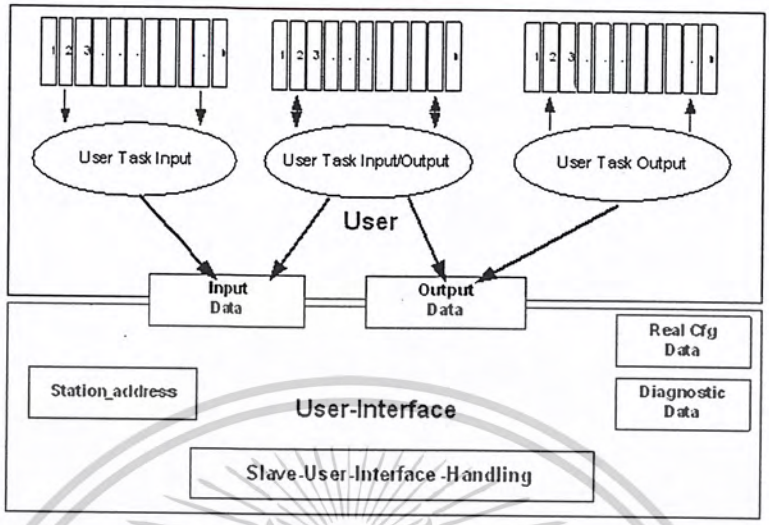
1. มีข้อมูลเข้ามาใหม่
2. มีการเปลี่ยนแปลงข้อมูลวินิจฉัย
3. มีการเปลี่ยนแปลงการกำหนดตั้งค่าต่าง ๆ

3.4.2 ส่วน User กับ User-Interface

เป็น Function ที่เรียกใช้ สำหรับกำหนดการทำงานให้เป็นไปตามสถานการนะการทำงานของอุปกรณ์รอง โดยจะมี function ที่เกี่ยวข้องดังต่อไปนี้

1. Send_Diag_Data ()
2. Send_Cfg_Data ()
3. Send_Set_Slave_Add ()
4. Send_Set_Prm ()
5. Send_Data_Exch ()
6. Send_Global_Control ()

โดยการเชื่อต่อข้อมูลจะใช้การเชื่อมต่อ ผ่าน Data Interface คือ Input Data, Output Data, Diagnostic Data, Parameter Data, Real Configuration Data และ Station Address

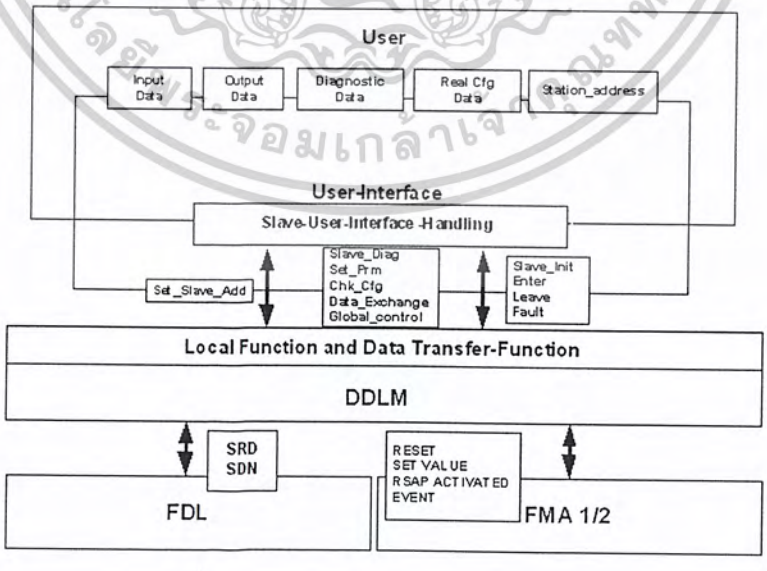


รูปที่ 3-34 การออกแบบการทำงานของระดับ User

ในตัวอย่าง Simple Slave Device จะมี task ที่ควบคุมการส่ง-รับ Input และ Output ของแต่ละ module โดยจะมีส่วนเก็บข้อมูล Input และ Output ก่อนที่จะทำการส่งไปยัง Master และ กระจายให้แต่ละ Module

3.5. ผู้ใช้งาน (User)

ผู้ใช้งานสำหรับกรณีที่เป็นอุปกรณ์จะแทนกระบวนการของการประยุกต์ใช้งานจริงในขึ้นกับการนำไปประยุกต์ใช้งาน แต่การทำงานหลักโดยทั่วไปจำเป็นต้องมีโครงสร้างการทำงานดังรูป



รูปที่ 3-35 โครงสร้างของอุปกรณ์ DP-Slave

3.6 โครงสร้างข้อมูล

โครงสร้างของข้อมูลที่ใช้ในระดับ User ของ Simple Slave Device ประกอบไปด้วย Input Data, Output Data, Diagnostic Data, Real Cfg Data และ Station Address โดยแต่ละส่วนมีโครงสร้างในการเก็บข้อมูลดังนี้

- Input Data และ Output Data

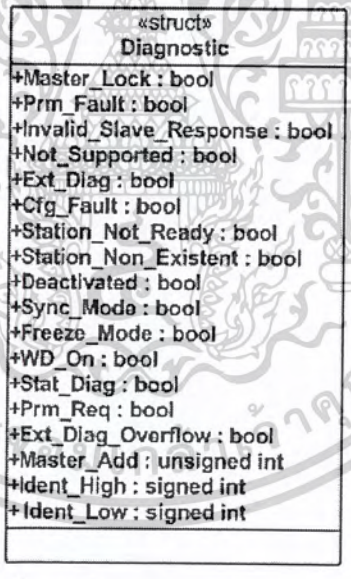
  เนื่องจากใน Simple Slave Device ได้สร้าง Module มา 4 Module เป็น Input 2 Module และ Output 2 Module ดังนั้นจึงเก็บข้อมูลโดยใช้ Array เนื่องจากว่า ได้กำหนดค่า Configuration ของข้อมูล Input และ Output ไว้ 1 Byte จึงสามารถใช้ INT8U ได้ โดยมีการประกาศข้อมูลดังนี้

  - INT8U Input[Input_Module] ; Input_Module = 2
  - INT8U Output[Output _Module] ; Output _Module = 2

- Station Address เป็นหมายเลขประจำที่อยู่ของแต่ละ Slave Device ดังนั้น จึงประกาศเป็นลักษณะของตัวแปรประเภท Global ดังนี้

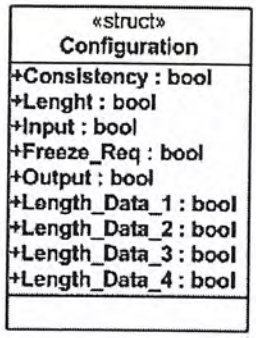
  - INT8U Station_Address

- Diagnostic Data มีลักษณะการเก็บข้อมูลเป็น Struct โดยมีประกอบด้วยตัวแปรที่เกี่ยวข้อง และชนิดของข้อมูลดังนี้



รูปที่ 3-36 โครงสร้าง Diagnostic Data

- Real Cfg Data มีลักษณะการเก็บข้อมูลเป็น Struct โดยมีประกอบด้วยตัวแปรที่เกี่ยวข้อง และชนิดของข้อมูลดังนี้

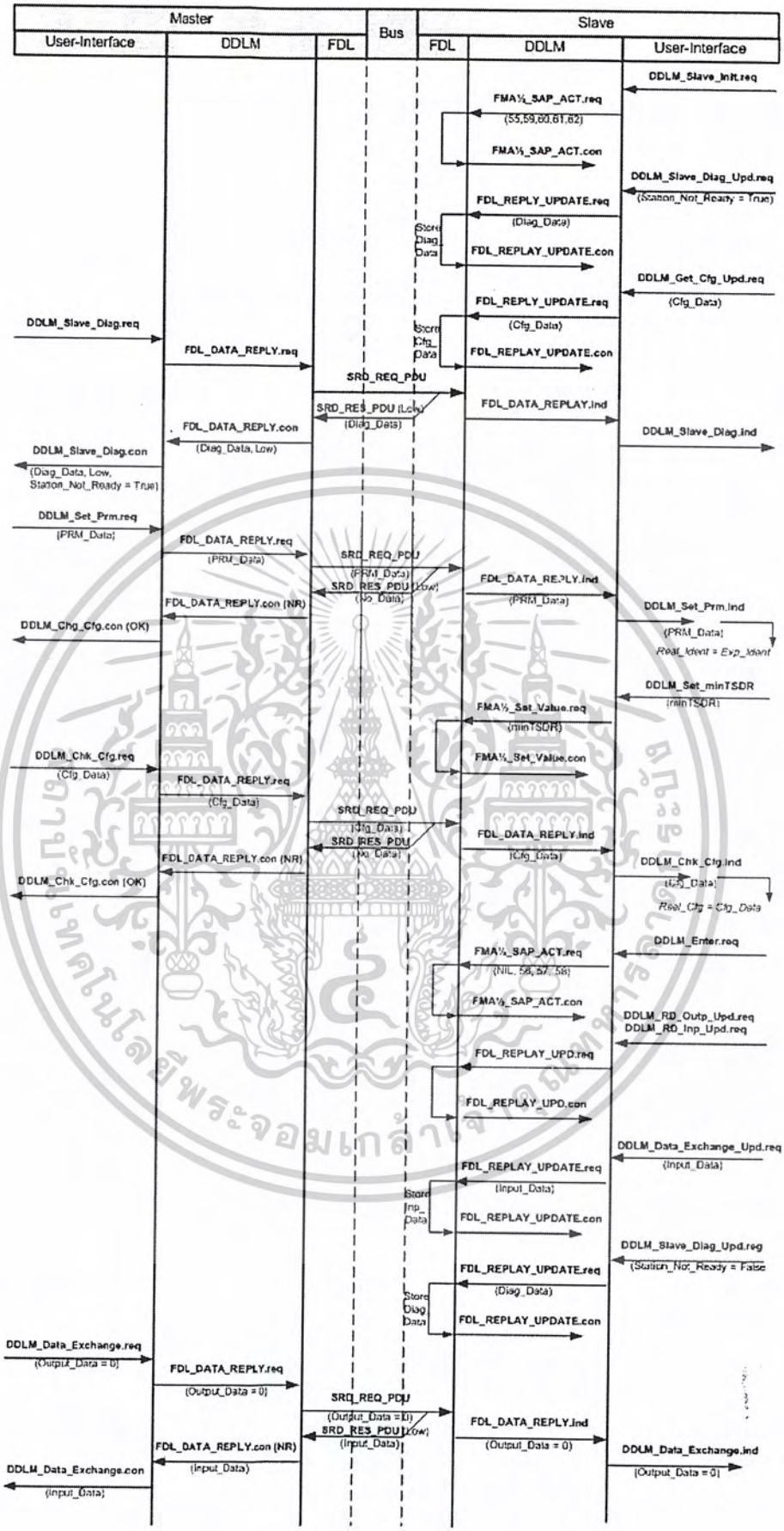


รูปที่ 3-37 โครงสร้าง Configuration Data

3.7 Sequence Diagram

Sequence Diagram ของ Simple Slave Device มีดังรูปที่ 3-38 โดยมีลำดับในการทำงานดังนี้

- เริ่มต้น Simple Slave Device จะทำการ Reset ค่าต่าง ๆ ของตัวแปรภายใน User Interface จะทำงานผ่านการเรียก Function DDLM_Slave_Init และ ในระดับ DDL จะเรียก Function ของ FMA ในระดับ Data Link Layer
- ขั้นต่อไปเป็นการ Set ค่าตัวแปรต่างๆ เพื่อที่รอจากตรวจสอบจาก Master โดยเรียกฟังก์ชั่น DDLM_Slave_Diag และส่งผ่านให้ FDL จัดการต่อไป นอกจากนี้ยังต้องมีการ Set ค่าของข้อมูล Configuration โดยจะใช้ DDLM_Get_Cfg ในการตั้งรูปแบบการแลกเปลี่ยนข้อมูล โดยจะส่งผ่านให้ FDL จัดการเช่นกัน
- ขั้นตอนต่อไป Master จะทำการตรวจสอบความพร้อมของ Slave Device โดยจะได้รับเป็นการ Call Function จาก FDL ส่งมายัง DDLM เพื่อแจ้งว่ามีกาตรวจสอบข้อมูล และส่งข้อมูลเพื่อให้ Master ทำการตรวจสอบ
- ต่อไปจะเป็นการ Set ข้อมูล Parameter เพื่อเป็นการตกลงกันก่อนทำการแลกเปลี่ยนข้อมูล เมื่อ FDL ได้รับข้อมูลจะทำการเรียก Function ของ DDLM เพื่อแจ้งให้มีการ Set และ ตรวจสอบรูปแบบการแลกเปลี่ยนข้อมูลว่าตรงกัน หลังจากนั้นจะทำการ Set ค่าเวลาในการรอข้อมูลจาก Master โดย Function DDLM_Set_minTSRD
- หลังจากตรวจสอบรูปแบบในการแลกเปลี่ยนข้อมูลที่ตรงกันแล้ว Slave Device จะทำการอ่านค่าข้อมูลจาก Module แล้วมาเก็บไว้ในตัวแปร Input รอการส่งไปยัง Master ในขณะที่มีการแลกเปลี่ยนข้อมูลเกิดขึ้น
- เมื่อถึงการแลกเปลี่ยนข้อมูล Master จะส่งOutput Data มายัง Slave Device และ รับ Input Data จาก Slave Device กลับไป FDL จะทำการเรียก Function DDLM_Data_Exchange ของ DDLM เพื่อแจ้งว่า มีการส่งข้อมูลจาก Master หลังจากนั้น User Interface จะทำการนำค่าที่ได้ไปเก็บใน Output Data เพื่อรอ User นำไปใช้ต่อไป

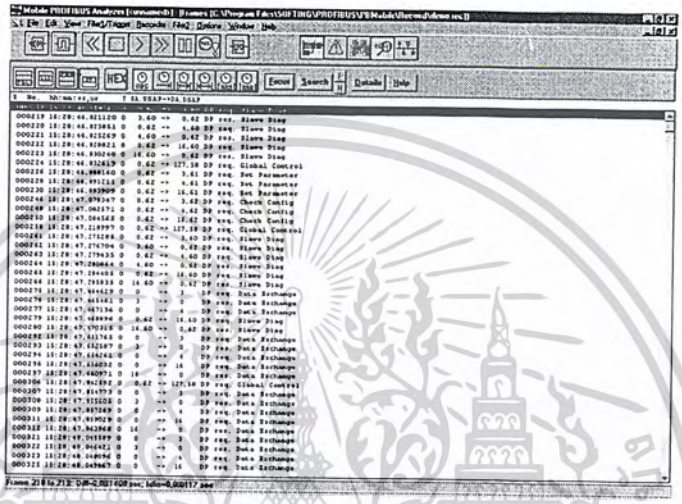


รูปที่ 3-38 Sequence Diagram

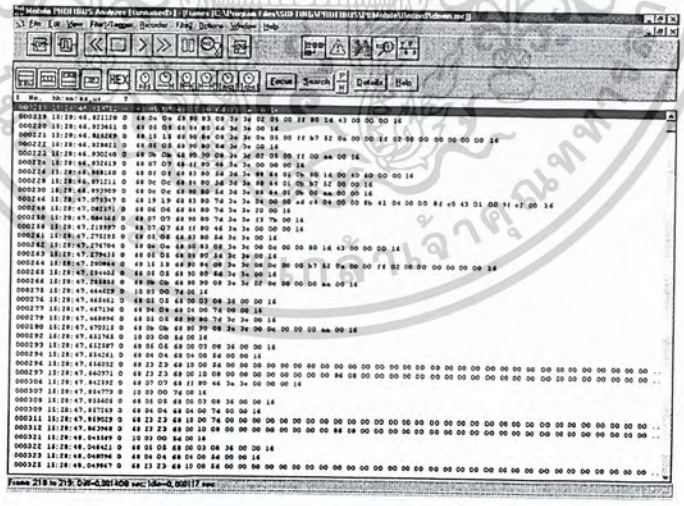
บทที่ 4
การทดลองการทำงานอุปกรณ์รองโปรฟิบัสดี-พี

4.1 การวิเคราะห์การส่งข้อมูลโปรฟิบัส

ตัวอย่างโปรแกรมที่นำมาวิเคราะห์การส่งข้อมูลมีชื่อว่า Mobile PROFIBUS Analyzer เป็นโปรแกรมที่คอยดักจับข้อมูลที่สื่อสารภายในเครือข่ายโปรฟิบัส



รูปที่ 4-1 ตัวอย่างโปรแกรมวิเคราะห์การสื่อสารโปรฟิบัสแบบคำสั่ง



รูปที่ 4-2 ตัวอย่างโปรแกรมวิเคราะห์การสื่อสารโปรฟิบัสแบบเทเลแกรม

การทำงานของโปรแกรมจะแบ่งชนิดของข้อมูลออกเป็นประเภทต่างๆ ทำให้สามารถเข้าใจการลำดับการทำงานได้มาขึ้น นอกจากนี้แล้วยังแสดงข้อมูลในรูปแบบของเลขฐาน 16 ทำให้สามารถเข้าใจ

รายละเอียดของข้อมูลในการสื่อสารมากยิ่งขึ้น โดยสามารถแสดงตัวอย่างการวิเคราะห์ข้อมูลจากโปรแกรมดังกล่าวได้ดังต่อไปนี้

4.1.1 การรับ-ส่งข้อมูลวินิจฉัย (Diag_Data)

```
000222 15:28:46.828821 0   0.62 ->  16.60 DP req. Slave Diag
000223 15:28:46.830240 0  16.60 ->   0.62 DP res. Slave Diag
```

รูปที่ 4-3 วิเคราะห์การรับ-ส่งข้อมูลวินิจฉัย

จากรูปลำดับที่ 222 เป็นการร้องขอข้อมูลวินิจฉัยจากอุปกรณ์หลักหมายเลข 0 ไปยัง อุปกรณ์รองหมายเลข 16 และลำดับที่ 223 เป็นการส่งข้อมูลวินิจฉัยจากอุปกรณ์รองกลับไปยังอุปกรณ์หลัก

```
000222 15:28:46.828821 0  68 05 05 68 90 80 6d 3c 3e 00 16
000223 15:28:46.830240 0  68 0b 0b 68 80 90 08 3e 3c 02 05 00 ff 00 aa 00 16
```

รูปที่ 4-4 วิเคราะห์เทเลแกรมการรับ-ส่งข้อมูลวินิจฉัย

เมื่อพิจารณาถึงข้อมูลภายในเทเลแกรม สามารถแสดงรายละเอียดส่วนของข้อมูลต่างๆ คือ ข้อมูลที่ส่งจากอุปกรณ์หลักจะเรียกใช้การบริการต้นทางเป็น 62 และ ปลายทางเป็น 60 เป็นการเรียกใช้งานการร้องขอข้อมูลวินิจฉัยจากอุปกรณ์รอง ดังนั้นเมื่ออุปกรณ์รองตอบกลับข้อมูลจะมีการส่งข้อมูลวินิจฉัยกลับไป ซึ่งมาตรฐานของข้อมูลวินิจฉัยทั่วไปมีขนาด 6 ไบต์

SD2	LE	LEr	SD	DA	SA	FC	DASP	SSAP	FCS	ED
68	05	05	68	90	80	6d	3c	3e	00	16

รูปที่ 4-5 วิเคราะห์เทเลแกรมการ ส่งข้อมูลวินิจฉัยของอุปกรณ์หลัก

SD2	LE	LEr	SD	DS	SA	FC	DSAP	SSAP	D1	D2	D3	D4	D5	D6	FCS	ED
68	0	0	68	80	90	08	3c	3c	02	05	00	Ff	00	aa	00	16

รูปที่ 4-6 วิเคราะห์เทเลแกรมการ ส่งข้อมูลวินิจฉัยของอุปกรณ์รอง

4.1.2 การรับ-ส่งข้อมูลพารามิเตอร์ (Set_Prm)

จากรูปเป็นการส่งข้อมูลพารามิเตอร์จากอุปกรณ์หลักหมายเลข 0 ไปยังอุปกรณ์รอง หมายเลข 16 โดยข้อมูลจากอุปกรณ์หลักจะประกอบไปด้วย ข้อมูลการตั้งค่าต่างที่ต้องการให้อุปกรณ์รองทำการตั้งค่าดังกล่าว ให้ตรงกัน โดยทั่วไปแล้วข้อมูลพารามิเตอร์จะมีขนาด 7 ไบต์ โดยจะเรียกใช้หมายเลขบริการต้นทางเป็น 62 และปลายทางเป็น 61

```
000230 15:28:46.893909 0    0.62 ->  16.61 DP req. Set Parameter

000230 15:28:46.893909 0  68 0c 0c 68 90 80 5d 3d 3e 88 64 01 0b 00 aa 00 00 16
```

รูปที่ 4-7 วิเคราะห์การรับ-ส่งข้อมูลพารามิเตอร์

SD2	LE	LEr	SD	DS	SA	FC	DSAP	SSAP	D1	D2	D3	D4	D5	D6	D7	FCS	ED
68	0c	0c	68	90	80	5d	3d	3e	88	64	01	0b	00	aa	00	00	16

รูปที่ 4-8 วิเคราะห์เทเลแกรมการรับ-ส่งข้อมูลพารามิเตอร์

4.1.3 การตรวจสอบข้อมูลองค์ประกอบ (Chk_Cfg)

เป็นการตรวจสอบข้อมูลที่ตั้งไว้โดยอุปกรณ์หลักหมายเลข 0 จะส่งค่าองค์ประกอบไปตรวจสอบที่อุปกรณ์รองหมายเลข 16 โดยจะมีข้อมูลที่ใช้ตรวจสอบส่งไปด้วย เรียกใช้ประเภทบริการ 62

```
000250 15:28:47.084565 0    0.62 ->  16.62 DP req. Check Config

000250 15:28:47.084565 0  68 07 07 68 90 80 7d 3e 3e f3 7b 00 16
```

รูปที่ 4-9 วิเคราะห์การตรวจสอบข้อมูลองค์ประกอบ

SD2	LE	LEr	SD	DS	SA	FC	DSAP	SSAP	D1	D2	FCS	ED
68	07	07	68	90	80	7d	3e	3e	F3	7b	00	16

รูปที่ 4-10 วิเคราะห์เทเลแกรมการตรวจสอบข้อมูลองค์ประกอบ

4.1.4 การส่งข้อมูลควบคุม (Global_Control)

การส่งข้อมูลควบคุมจะส่งจากอุปกรณ์หลักหมายเลข 0 ส่งไปยังอุปกรณ์รองทุกๆ ตัว โดยให้ค่าหมายเลขของปลายทางเป็น 255 เมื่ออุปกรณ์รองได้รับข้อมูลจะกระทำการบางอย่างตามข้อมูลที่กำหนดไว้ โดยจะใช้บริการต้นทางประเภท 62 และปลายทางเป็น 58 พื้นฐานทั่วไปจะประกอบด้วยข้อมูล 2 ไบต์

```
000258 15:28:47.218997 0    0.62 -> 127.58 DP req. Global Control

000258 15:28:47.218997 0  68 07 07 68 ff 80 46 3a 3e 00 00 00 16
```

รูปที่ 4-11 วิเคราะห์การส่งข้อมูลควบคุม

SD2	LE	LEr	SD	DS	SA	FC	DSAP	SSAP	D1	D2	FCS	ED
68	07	07	68	Ff	90	46	3a	3e	00	00	00	16

รูปที่ 4-12 วิเคราะห์เทเลแกรมการส่งข้อมูลควบคุม

4.1.5 การแลกเปลี่ยนข้อมูล (Data_Exchange)

การแลกเปลี่ยนข้อมูลจากอุปกรณ์หลักหมายเลข 0 ไปอุปกรณ์รองหมายเลข 16 โดยสมารถส่งข้อมูลได้มีขนาดมากสุดถึง 32 ไบต์ โดยจะไม่มีการเรียกใช้ประเภทการบริการ

```
000296 15:28:47.656032 0   0    ->  16    DP req. Data Exchange
000297 15:28:47.660971 0  16    ->   0    DP res. Data Exchange


000296 15:28:47.656032 0  68 23 23 68 10 00 5d 00 00 00 00 00
000297 15:28:47.660971 0  68 23 23 68 00 10 08 00 00 00 00 00
```

รูปที่ 4-13 วิเคราะห์การแลกเปลี่ยนข้อมูล

SD2	LE	LEr	SD	DS	SA	FC	D1	D2	Dx	Dx	Dx	D32	FCS	ED
68	23	23	65	10	00	5d	00	05	00	00	00	00	00	16

รูปที่ 4-14 วิเคราะห์เทเลแกรมการแลกเปลี่ยนข้อมูลของอุปกรณ์หลัก

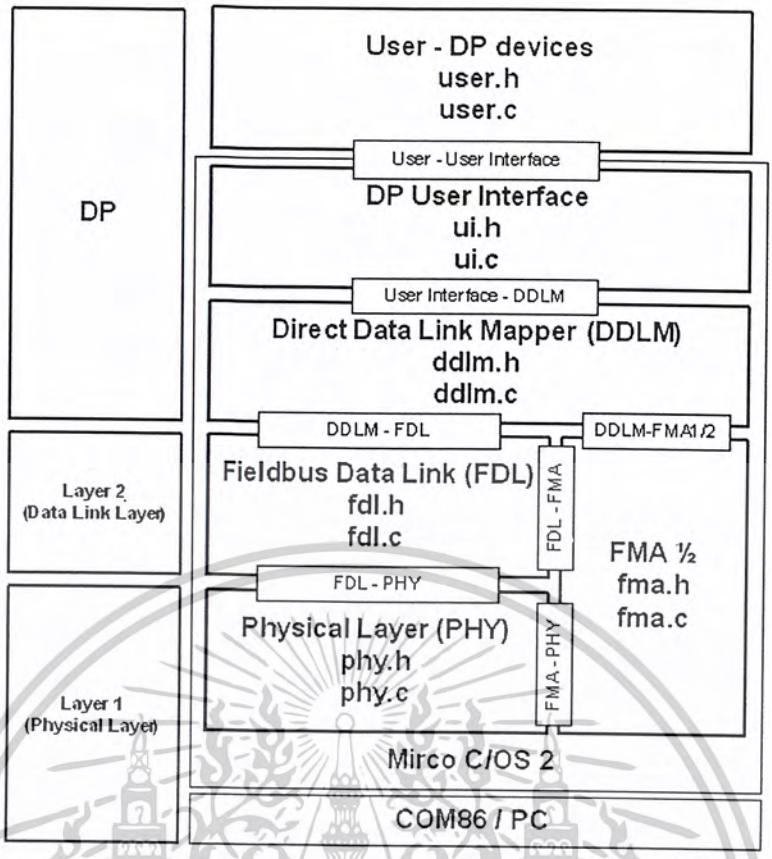
SD2	LE	LEr	SD	DS	SA	FC	D1	D2	Dx	Dx	Dx	D32	FCS	ED
68	23	23	65	00	10	08	00	00	00	00	00	00	00	16

รูปที่ 4-15 วิเคราะห์เทเลแกรมการส่งข้อมูลควบคุมของอุปกรณ์รอง

4.2 การทดสอบการทำงาน

การทดสอบการทำงานจะทดสอบความถูกต้องของแต่ละชั้น แต่ละโมดูลของในแต่ละชั้น การเชื่อมต่อระหว่างโมดูล รวมถึงการเชื่อมต่อระหว่างแต่ละชั้น ว่าสามารถทำงานได้ร่วมกันหรือไม่ ถูกต้องอย่างไร ตรวจสอบว่าสามารถรับและส่งข้อมูลได้ทันกันหรือไม่

สำหรับการพัฒนาโปรโตคอลสำหรับอุปกรณ์ Slave นี้ ได้พัฒนาออกเป็นส่วนต่าง ๆ ตามลักษณะการเชื่อมต่อของแต่ละ Layer ที่ได้ออกแบบไว้ ดังนี้



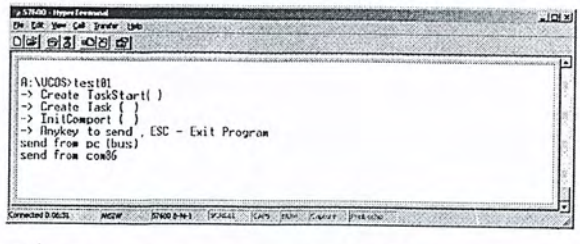
รูปที่ 4-16 โครงสร้างของโปรโตคอลสำหรับอุปกรณ์ Slave

4.2.1 การทดสอบในระดับ Physical layer (PHY)

ในชั้นนี้จะทำการทดลองข้อมูลรับมาจากทางพอร์ตอนุกรม และทดลองส่งข้อมูลออกจากพอร์ตอนุกรมไปยังบัส จึงได้ทดสอบการทำงานกับเครื่องคอมพิวเตอร์โดยการใช้โปรแกรม Hyper Terminal หรือ SercureCRT เพื่อติดต่อผ่านทางพอร์ตอนุกรมที่ความ 57600 บิตต่อวินาที ขนาดข้อมูล 8 บิตและบิตหยุด 1 บิต พาริตี้บิตไม่มี

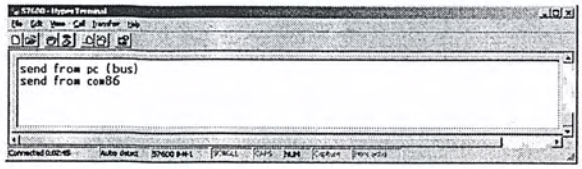
สำหรับรับข้อมูล การทำงานของในจะใช้กระบวนการ Interrupt ของพอร์ตอนุกรมนำข้อมูลมาเก็บไว้ใน Buffer ก่อนจากนั้นให้ระดับ PHY ทำการดึงข้อมูลออกมาทีละตัวอักษร โดยมีวิธีการดังนี้

1. ทำการเขียนโปรแกรมสำหรับทดสอบการทำงานและเรียกโปรแกรมดังกล่าว



รูปที่ 4-17 การทำงานบน Com86 กับโปรแกรม test01.exe

2. ทดลองส่งข้อมูลจากเครื่องคอมพิวเตอร์ไปหา Com86 โดยทำการส่งข้อมูลโดยพิมพ์ว่า "Send from pc (bus)" ดังรูปด้านล่างนี้ข้อมูลที่พิมพ์จะไปปรากฏดังหน้าจอของ Com86 ดังรูปด้านบน



รูปที่ 4-18 การทำงานบน PC

3. ทดลองส่งข้อมูลจาก Com86 มาหาเครื่อง PC โดยทำการส่งข้อมูลโดยพิมพ์ว่า "Send from com86" ดังรูปข้อมูลที่พิมพ์จะไปปรากฏดังหน้าจอของ PC

ตัวอย่าง การรับข้อมูลของโปรแกรม test01.exe

```
while(1)
{
        if (CheckInterrupt() )
        putchar(int_getch());
}
```

ตัวอย่างการส่งข้อมูลของโปรแกรม test01.exe

```
putchar ( int_putch(key));
```

ผลการทดสอบ

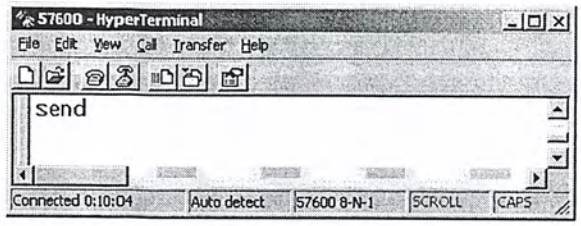
สามารถรับและส่งข้อมูลได้ถูกต้อง

4.2.2 การทดสอบการรับและส่งข้อมูลระหว่างส่วน PHY กับ FDL

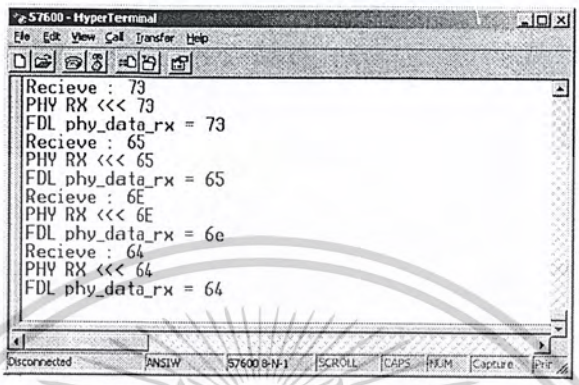
4.2.2.1 การทดสอบการส่งข้อมูลจาก PHY ไปหา FDL

การรับข้อมูลมากจากพอร์ตอนุกรมและส่งข้อมูลระหว่างส่วน PHY กับ FDL เมื่อได้รับข้อมูลเข้ามาจะใช้วิธีการส่งสัญญาณจาก PHY ไปหา FDL โดย sem_rx_phy_to_fdl และทันทีที่ FDL ได้รับสัญญาณมาจาก PHY จะทำการอ่านตัวแปรที่ได้กำหนดไว้ชื่อว่า PHY_RX เมื่อได้ทำการอ่านและเก็บค่าไว้แล้วจะทำการส่งสัญญาณจาก FDL ไปบอก PHY ว่าได้อ่านข้อมูลได้เรียบร้อยแล้ว sem_rx_fdl_to_phy

การทดสอบสามารถทำได้โดยรับข้อมูลจากบัสหรือเครื่อง PC แล้ว ดังรูปต่อไปนี้



รูปที่ 4-19 การส่งข้อมูลจาก PC



รูปที่ 4-20 การรับข้อมูลบน Com86 กับโปรแกรม test02.exe

สำหรับ PHY จะทำงานดังนี้

1. รอรับข้อมูลจากพอร์ตอนุกรม และรับข้อมูลเมื่อได้รับข้อมูลเข้ามาเก็บที่ PHY_RX
2. OSSemPost (sem_rx_phy_to_fdl) ส่งสัญญาณไปบอก FDL
3. OSSemPend (sem_rx_fdl_to_phy, 1000, &err) จากนั้นรอสัญญาณตอบกลับจาก FDL
4. ทำข้อ 1 ต่อ

สำหรับ FDL จะทำงานดังนี้

1. OSSemPend (sem_rx_phy_to_fdl, 1000, &err) รอสัญญาณจาก PHY
2. เมื่อได้รับสัญญาณจะอ่านข้อมูลจาก PHY_RX มาเก็บไว้
3. FDL_parse_telegram(phy_data_rx) นำข้อมูลที่เก็บไว้ไปประมวลผล
4. OSSemPost (sem_rx_fdl_to_phy) ส่งสัญญาณไปบอก PHY ว่าได้อ่านไปแล้ว
5. ทำข้อ 1 ต่อ

กระบวนการทดสอบ

1. ส่งข้อมูลจาก PC โดยพิมพ์ว่า “send”
2. โปรแกรม test02.exe จะทำการรับข้อมูลจาก Interrupt ได้รับข้อมูล “send” และแสดงผลดังนี้

Receive: 73	รับข้อมูล‘s’ จาก Interrupt
PHY RX <<< 73	เก็บ‘s’ ในชั้น PHY
FDL phy_data_rx = 73	เก็บ‘s’ ในชั้น FDL

Receive: 65	รับข้อมูล ‘e’ จาก Interrupt
PHY RX <<< 65	เก็บ ‘e’ ในชั้น PHY
FDL phy_data_rx = 65	เก็บ ‘e’ ในชั้น FDL

Receive: 6E	รับข้อมูล ‘n’ จาก Interrupt
PHY RX <<< 6E	เก็บ ‘n’ ในชั้น PHY
FDL phy_data_rx = 6e	เก็บ ‘n’ ในชั้น FDL

Receive: 64	รับข้อมูล‘d’ จาก Interrupt
PHY RX <<< 64	เก็บ‘d’ ในชั้น PHY
FDL phy_data_rx = 64	เก็บ‘d’ ในชั้น FDL

ผลการทดสอบ

สามารถรับข้อมูลได้ถูกต้อง โดยการใช้ Semaphore ช่วยในการรับข้อมูล

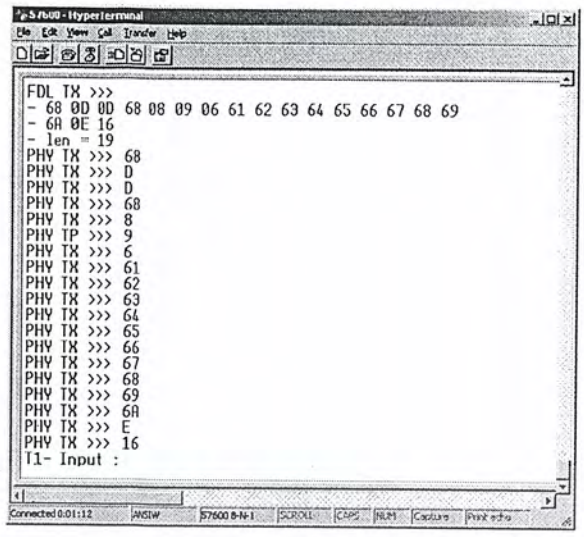
4.2.2.2 การทดสอบการส่งข้อมูลจาก FDL ไปหา PHY

การส่งข้อมูลจาก FDL ไปหา PHY จะใช้เมื่อต้องการส่งข้อมูลไปยังพอร์ตอนุกรม ในส่วนนี้จัดการโดย FDL จะใช้วิธีการเรียกฟังก์ชัน PHY_DATA_PUT (INT8U) เพื่อส่งข้อมูลลงมา ภายในฟังก์ชันนี้จะจึงมาใช้ Semaphore ควบคุมการส่งข้อมูลออกทางพอร์ตอนุกรม เพราะการส่งข้อมูลนั้นสามารถทำได้ที่ละครั้งหรือทีละ 1 ตัวอักษร โดยการทำงานดังนี้

1. OSSemPend(PHY_PUT, 0, &err) ขอเข้าไปฟังก์ชันส่งข้อมูล
2. int_putch(temp) ส่งข้อมูล
3. OSTimeDly(1)
4. OSSemPost(PHY_PUT) ออกจาการใช้ไปฟังก์ชันส่งข้อมูล

ผลการทดสอบ

ส่งข้อมูล 68 0D 0D 68 08 09 06 61 62 63 64 65 66 67 68 69 6A 0E 16 ทั้งหมด 19 ตัวอักษรจาก FDL ออกมาด้วยฟังก์ชัน PHY_DATA_PUT จากนั้น PHY ได้ทำการส่งข้อมูลออกไปทีละตัวอักษร ดังรูปด้านล่างนี้ และ ผลการทดสอบที่ได้ สามารถทำงานได้อย่างถูกต้อง



รูปที่ 4-21 การส่งข้อมูลระหว่าง FDL กับ PHY บน Com86 ด้วยโปรแกรม test03.exe

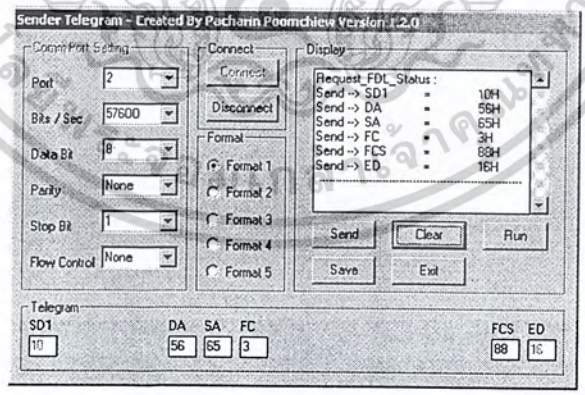
4.2.3 การทดสอบในระดับ Fieldbus Data link layer (FDL)

4.2.3.1 การทดสอบการตอบ-รับ Telegram ของโปรฟิบัส

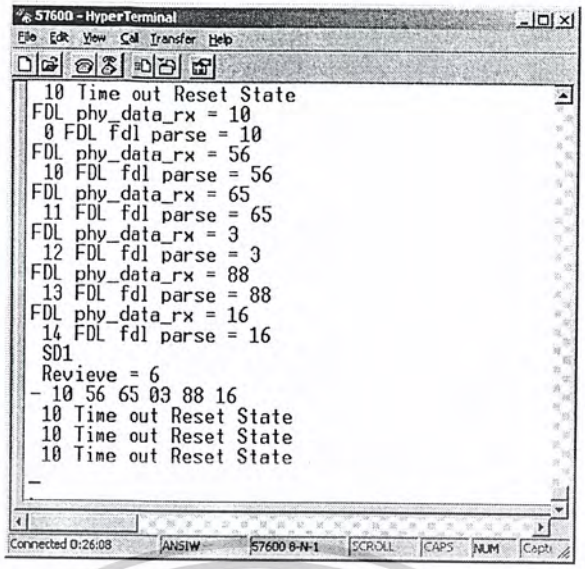
การรับข้อมูลและถอด Telegram โดยใช้ฟังก์ชันที่ได้สร้างขึ้นมาชื่อว่า FDL_parse_telegram ฟังก์ชันนี้จะรับข้อมูลที่ละ 1 ไบต์เข้ามา แล้วทำการแยกแต่ละไบต์ที่เข้ามาว่าเป็นอะไรบ้างและตรวจสอบความถูกต้องคือ ส่วนหัวและหาง ประเภท ความยาว การตรวจสอบผลรวม (checksum) และเพื่อป้องกันรูปแบบ telegram ที่ผิดหรือไม่มีฟังก์ชันนี้จะมีการ reset สถานะทุกครั้งที่เจอความผิดพลาดหรือหมดเวลา

การทดสอบจะใช้โปรแกรมที่สร้างขึ้นมาเพื่อสร้าง Telegram รูปแบบต่าง ๆ

1. การทดสอบการรับข้อมูลแบบ SD1



รูปที่ 4-22 การส่งข้อมูลแบบที่ 1 (SD1)



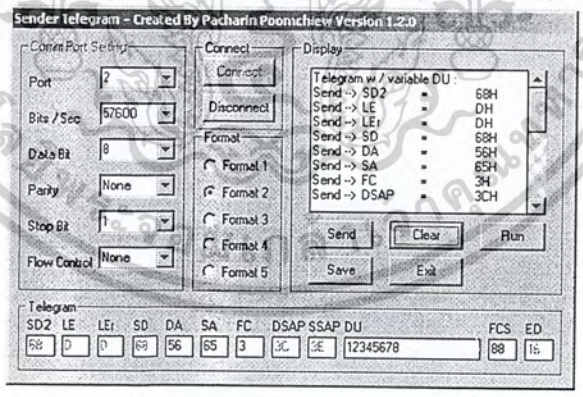
รูปที่ 4-23 ผลการรับข้อมูลแบบที่ 1 (SD1)

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SD1 เป็นคำสั่งประเภท Request_FDL_Status ผลลัพธ์ที่ได้สามารถรับข้อมูลได้ถูกต้อง คือสามารถที่ส่งกับที่รับเหมือนกัน

SD1	DA	SA	FC	FCS	ED
10H	56H	65H	3H	88H	16H

2. การรับข้อมูลแบบ SD2



รูปที่ 4-24 การส่งข้อมูลแบบที่ 2 (SD2)

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SD2 เป็นคำสั่งประเภท Telegram w / variable DU ผลลัพธ์ที่ได้สามารถรับข้อมูลได้ถูกต้อง

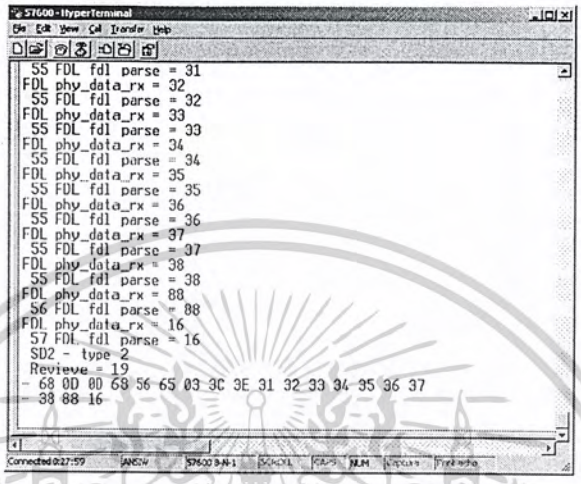
SD2	LE	LEr	SD	DA	SA	FC	DSAP	SSAP

68H	DH	DH	68H	56H	65H	3H	3CH	3EH

DU (Data Unit)

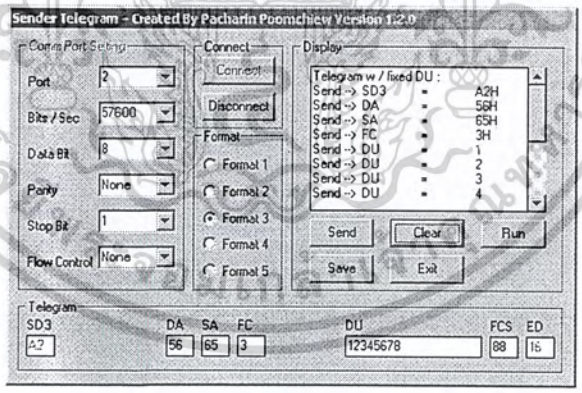
1	2	3	4	5	6	7	8

FCS	ED
88H	16H



รูปที่ 4-25 ผลการรับข้อมูลแบบที่ 2 (SD2)

3. การรับข้อมูลแบบ SD3



รูปที่ 4-26 การส่งข้อมูลแบบที่ 3 (SD3)

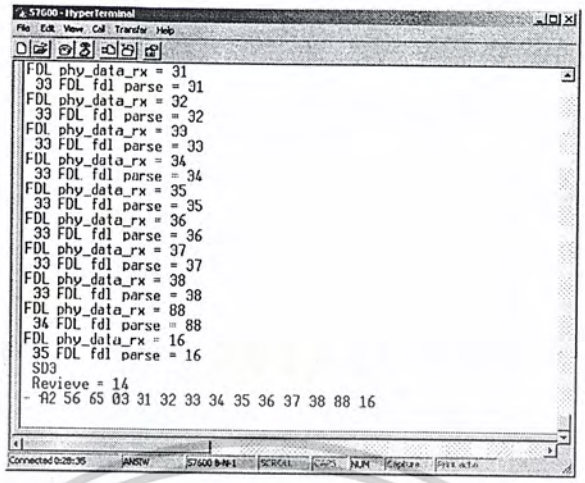
ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SD2 เป็นคำสั่งประเภท Telegram w / variable DU ผลลัพธ์ที่ได้สามารถรับข้อมูลได้ถูกต้อง

SD2	DA	SA	FC	FCS	ED
56H	56H	65H	3H	88H	16H

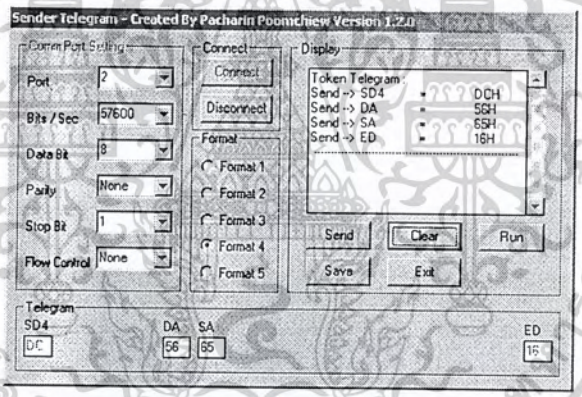
DU (Data Unit)

1 2 3 4 5 6 7 8

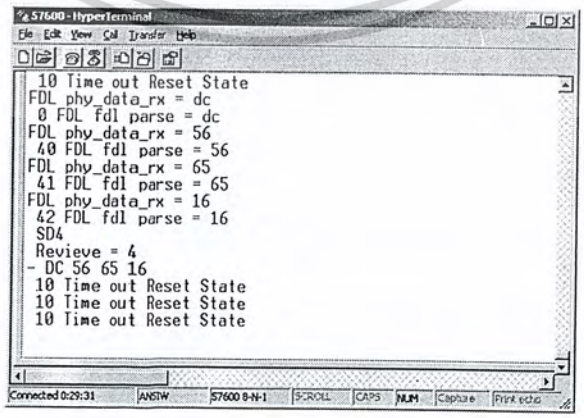


รูปที่ 4-27 ผลการรับข้อมูลแบบที่ 3 (SD3)

4. การรับข้อมูลแบบ SD4



รูปที่ 4-28 การส่งข้อมูลแบบที่ 4 (SD4)



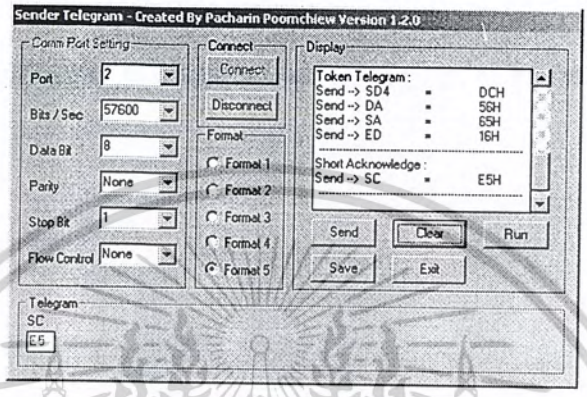
รูปที่ 4-29 ผลการรับข้อมูลแบบที่ 4 (SD4)

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SD2 เป็นคำสั่งประเภท Telegram w / variable DU ผลลัพธ์ที่ได้สามารถรับข้อมูลได้ถูกต้อง

SD2	DA	SA	ED
56H	56H	65H	16H

5. การรับข้อมูลแบบ SC



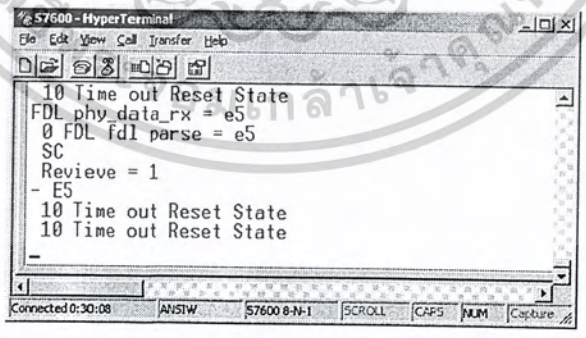
รูปที่ 4-30 การส่งข้อมูลแบบที่ 5 (SC)

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SD2 เป็นคำสั่งประเภท Telegram w / variable DU ผลลัพธ์ที่ได้สามารถรับข้อมูลได้ถูกต้อง

SC

E5H



รูปที่ 4-31 การรับข้อมูลแบบที่ 5(SC)

4.3 การทดสอบการตอบ-รับบริการ SDN และ SRD ของ FDL PROFIBUS-DP

4.3.1 การส่งข้อมูลแบบ SDN

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SDN ผลลัพธ์ที่ได้สามารถแยกบริการตามที่กำหนดไว้ คือ SSAP = 58 จะมีการเรียกฟังก์ชัน FDL_SDN_indication () เพื่อบอกว่ามีบริการ SDN

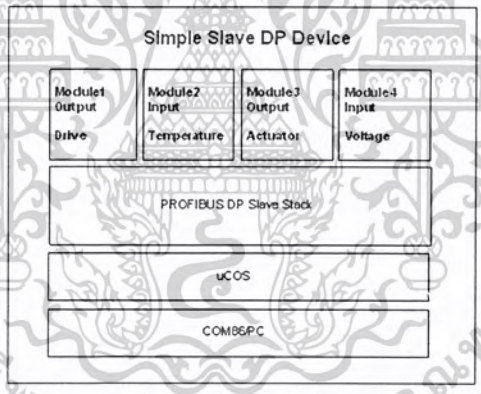
4.3.2 การส่งข้อมูลแบบ SRD

ผลการทดสอบ

ทดสอบการรับข้อมูลแบบ SDN ผลลัพธ์ที่ได้สามารถแยกบริการตามที่กำหนดไว้ คือ SSAP มีค่า คือ 54, 55, 56, 57, 59 ,60 ,61 , 62 จะมีการเรียกฟังก์ชัน FDL_SRN_indication () เพื่อบอกว่ามีบริการSRNและจะมีการตอบส่งข้อมูลกลับด้วยฟังก์ชัน FDL_SRD_REPLY_UPDATE_respond โดยก่อนหน้านี้จะต้องการนำข้อมูลที่ต้องการส่งกลับมาใส่ใน FDL_SRD_REPLY_UPDATE ()

4.4 การทดสอบสถานะทำงาน

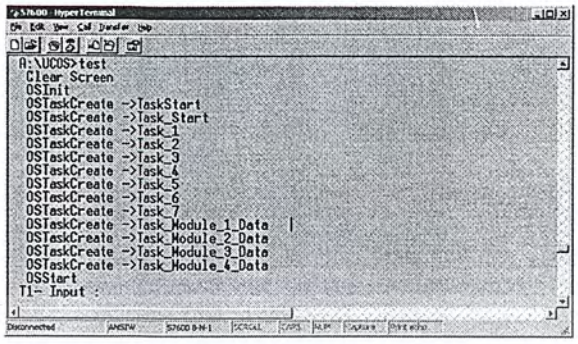
ในการทดสอบสถานะของการทำงานของอุปกรณ์ จะทำการจำลองอุปกรณ์รองขึ้นมา โดยสมมุติให้มี Module ในการทำงานตามรูปที่ 4-32 และคุณสมบัติของอุปกรณ์ตาม GSD File ที่ได้ออกแบบไว้



รูปที่ 4-32 ตัวอย่างโครงสร้างอุปกรณ์รองอย่างง่าย

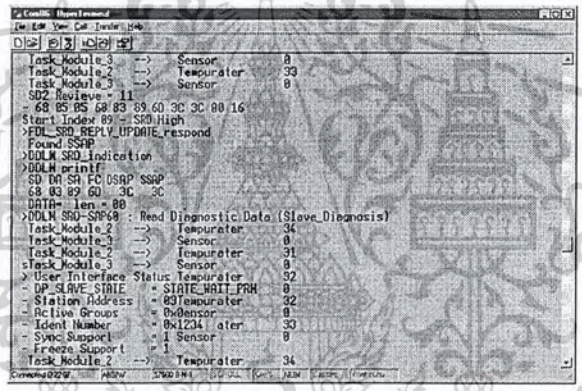
การทดสอบการทำงานจะเป็นไปตามสถานะของการทำงานของอุปกรณ์รอง โดยมีขั้นตอนการทดสอบเป็นลำดับดังต่อไปนี้

1. เริ่มทำงาน สร้าง Task และ Initial ค่าเริ่มต้น



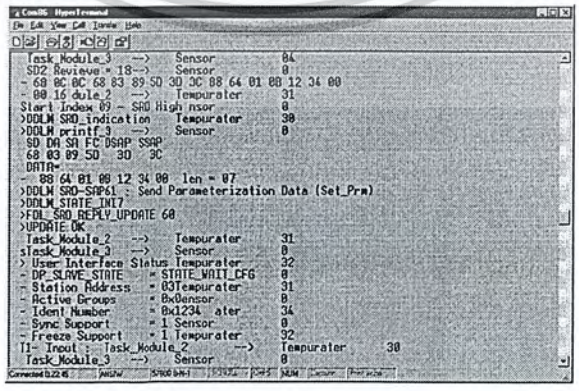
รูปที่ 4-33 ตัวอย่างสถานะการทำงานเริ่มต้น

2. Master ตรวจสอบ Slave_Diag เพื่อสอบถามสถานะ



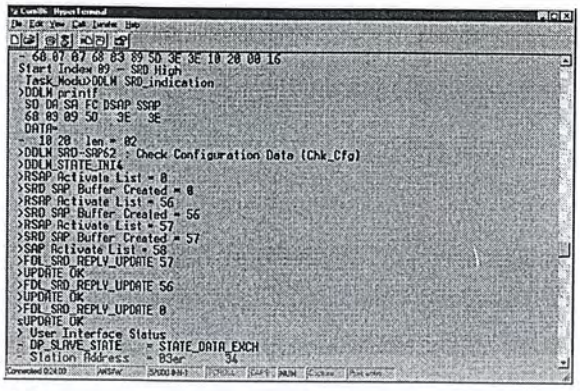
รูปที่ 4-34 ตัวอย่างสถานะการทำงานของการตรวจสอบ

3. Master กำหนดค่า Parameter Data



รูปที่ 4-35 ตัวอย่างสถานะการทำงานตั้งค่า Parameter

4. Master ตรวจสอบการกำหนดค่า Chk_cfg

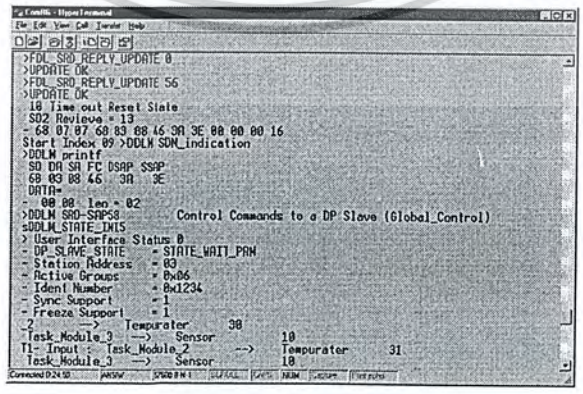


รูปที่ 4-36 ตัวอย่างสถานะการตรวจสอบ Configuration Data

5. Master แลกเปลี่ยนข้อมูลกับอุปกรณ์ (Data Exchange)

รูปที่ 4-37 ตัวอย่างสถานะการแลกเปลี่ยนข้อมูล

6. Master สั่งงานแบบ global control



รูปที่ 4-38 ตัวอย่างสถานะคำสั่งพิเศษ

สำหรับการทดสอบส่วนอื่น ๆ ขึ้นกับผู้ที่นำไปใช้งานจะพัฒนานำไปพัฒนาต่อ ในโครงการนี้ได้ค้นคว้า ข้อมูลและกระบวนการต่าง ๆ ในการทำงานของมาตรฐาน PROFIBUS DP ซึ่งมีรายละเอียดมาก ไม่สามารถจะพัฒนาทั้งได้หมด ซึ่งการใช้งานจริง ๆ แล้วขึ้นกับผู้ใช้งานนำไปประยุกต์ใช้กับงานประเภทใด แต่ส่วนที่เหมือนกันคือฟังก์ชันพื้นฐานที่เรียกใช้งาน PROFIBUS DP

4.5 GSD File Simple Slave Device

```
#Profibus_DP
GSD_Revision    =    21
Vendor_Name     =    "ESL KMITL"
Model_Name      =    "DP-TEST01"
Revision        =    "Version 1"
Ident_Number    =    0x1234
Protocol_Ident  =    0        ; DP protocol
Station_Type    =    0        ; Slave device
FMS_supp        =    0        ; FMS not supported
Hardware_Release =   "Version 1"
Software_Release =   "Version 1"
9.6_supp        =    1
MaxTsdr_9.6     =    15
Redundancy      =    0    ; not supported
Implementation_Type  =       "COM86"
Freeze_Mode_supp     =       1    ; supported
Sync_Mode_supp       =       1    ; supported
Auto_Baud_supp       =       0    ; Not supported
Set_Slave_Add_supp   =       0    ; not supported
Min_Slave_Intervall  =       1    ; 100 us
Modular_Station      =       0    ; modular
Max_Module           =       4
Max_Input_Len        =       2
Max_Output_Len       =       2
Max_Data_Len         =       4
Modul_Offset         =       0
Slave_Family         =       0
Max_Diag_Data_Len    =       6
```

Module = "INPUT: 1 Byte" 0x01

EndModule

Module = "OUTPUT: 1 Byte" 0x02

EndModule

Module = "INPUT: 1 Byte " 0x03

EndModule

Module = "OUTPUT: 1 Byte" 0x04

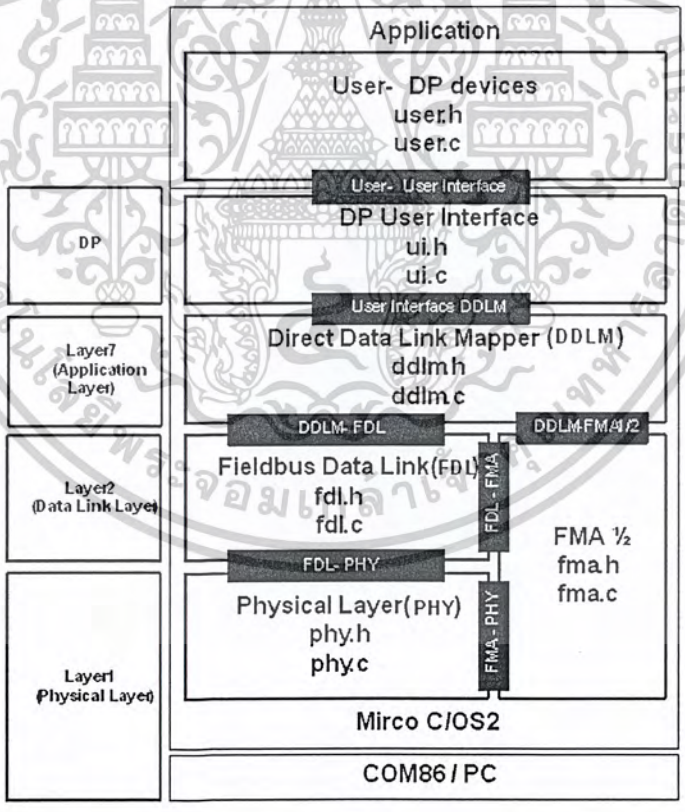
EndModule

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการดำเนินงาน

การดำเนินของงาวิจัยชิ้นนี้สำเร็จลุล่วงไปได้เป็นอย่างดี เนื่องด้วยมาจากมีการศึกษา ค้นคว้า วิเคราะห์รายละเอียดของมาตรฐานโปรฟิบัสอย่างแท้จริง มีการนำตัวอย่างโปรแกรมดักจับข้อมูลโปรฟิบัส มาวิเคราะห์ถึงลักษณะการส่ง-รับข้อมูลภายในเครือข่าย มาตีความหมายตามโครงสร้างของทฤษฎีที่มีอยู่ ทำให้เข้าใจการทำงานของมาตรฐานนี้ได้อย่างแท้จริง

ในขั้นตอนการออกแบบ มีการออกแบบการทำงานเป็นไปตามโครงสร้างของสถาปัตยกรรมโพรโตคอลโปรฟิบัส โดยแยกออกแบระดับชั้นการสื่อสารต่างๆ ซึ่งแต่ละชั้นการสื่อสารจะมีหน้าที่ และการทำงานที่แตกต่างกันออกไป นอกจากนี้ ยังมีการออกแบบในส่วนของการสื่อสารระหว่างระดับชั้นการสื่อสารในส่วนของการเรียกใช้งานข้ามระหว่างชั้นการสื่อสารอีกด้วย



รูปที่ 5-1 ภาพรวมการออกแบบ PRFIBUS-DP Slave

5.1.2 สรุปขั้นตอนการดำเนินงาน

สามารถสรุปขั้นตอนการดำเนินงานของโครงงานได้ดังต่อไปนี้

1. ศึกษาข้อมูลการทำงานและเปิดโลกทางด้าน PROFIBUS DP
2. พัฒนาแนวทางสำหรับอุปกรณ์ Slave Device
3. พัฒนา PROFIBUS DP Protocol Library สำหรับอุปกรณ์ Slave Device ที่สามารถทำงานได้ในระดับพื้นฐานที่สามารถทำงานได้
4. พัฒนาต้นแบบของอุปกรณ์ PROFIBUS DP - Slave Device

การทดลองการใช้งานจะทำการส่งสร้างรูปแบบข้อมูลที่ได้จาก ตัวอย่างโปรแกรมดักจับการสื่อสารโปรฟิบัส แล้วนำค่าที่ได้จากการสนองตอบการทำงานของอุปกรณ์รองมาเปรียบเทียบกับตัวอย่างที่ได้จากโปรแกรม โดยจะทำการทดสอบในรูปแบบข้องข้อมูลที่แตกต่างกันออกไป ในแต่ละคำสั่ง และได้ทำการทดลองส่งข้อมูลที่ผิดแปลกไปจากรูปแบบมาตรฐานการสื่อสาร โดยผลการทดลองที่ได้ อุปกรณ์รองสามารถรับส่งข้อมูลจากอุปกรณ์หลัก สามารถตีความหมายของคำสั่งที่ส่งมา และ ตอบกลับข้อมูลได้ตรงตามตัวอย่างการทำงาน รวมถึงการปฏิเสธรูปแบบข้อมูลที่แปลกไปจากโครงสร้างการสื่อสารของมาตรฐานโปรฟิบัสดี-พีด้วย

5.1.3 ความถูกต้อง

ความถูกต้องในการทำงานของอุปกรณ์มีดังต่อไปนี้

1. สามารถรับ SD1 SD2 SD3 SD4 SC
2. สามารถรับ FC SSAP DSAP ที่รู้จักและเข้าใจ
3. สามารถตรวจจับ Error เช่น LE != LEr , LE >>> , LE <<<
4. มีการตรวจสอบ Address SA, DA Broadcast
5. SAP ที่ไม่ได้เปิดบริการ SDN,SRD จะไม่รับบริการ,ตอบสนองบริการ และตอบกลับด้วยการบอกว่าไม่มีบริการ
6. สามารถรับข้อมูลได้สูงสุด 44 ตัวอักษร (เป็นการส่งข้อมูลตามที่ใช้ในปัจจุบัน)

5.14 ข้อจำกัด

ข้อจำกัดในการทำงานโครงงานได้แก่

1. ไม่มีอุปกรณ์จริงที่สามารถนำมาถอดรหัสได้
2. มีบางส่วนที่สามารถตีความได้และไม่ได้โดยเฉพาะคำสั่งพิเศษ นอกจากมาตรฐานหรือข้อมูลไม่เพียงพอ
3. ในการใช้งานอุปกรณ์ Slave จะต้องใช้ร่วมกับ Master แต่ในระบบนี้จะใช้วิธีสร้าง Telegram ที่ Master ส่งแทน ตามแต่ละ State ที่ควรจะเป็น

5.1.5 ปัญหาในการดำเนินงาน

ปัญหาที่พบระหว่างการดำเนินงานสามารถสรุปได้ดังต่อไปนี้

1. ข้อมูลเกี่ยวกับมาตรฐานโปรฟิบัส ในส่วนที่เกี่ยวกับการสร้างการทำงานของอุปกรณ์ในระบบ มีน้อยมากทำให้ต้องใช้เวลานานในการค้นคว้า หาแหล่งข้อมูลเพื่อศึกษา

2. ผู้เชี่ยวชาญทางด้านมาตรฐานโปรฟิบัสภายในประเทศมีน้อย ซึ่งทำให้เสียโอกาสการได้รับคำแนะนำ หรือชี้แนวทางจากผู้รู้รายละเอียดการทำงานจริงๆ
3. ในการเลือกชุดพัฒนาคอม 86 มาใช้ในการพัฒนา ทำให้สามารถสร้างโปรแกรมที่มีขนาดจำกัด เนื่องจากเนื้อที่ของหน่อยความจำในชุดพัฒนามีขอบเขตที่จำกัด
4. การใช้งานของ uCOS-II ประสบปัญหากับรูปแบบการเรียกใช้คำสั่งในการทำงาน เนื่องจากมีข้อจำกัดความสามารถในการใช้งานของบางคำสั่ง
5. โครงสร้างการทำงานตามมาตรฐานโปรฟิบัสมีความซับซ้อนและขั้นตอนการทำงานที่ละเอียดมาก รวมถึงมีฟังก์ชั่นการทำงานที่หลากหลาย ทำให้การสร้างโปรแกรมให้สามารถทำงานสมจริงและรองรับความสามารถพิเศษหลายๆ ด้าน ทำได้ลำบาก
6. การ Monitor ของข้อมูลที่ใช้ระหว่างการรับ-ส่งข้อมูลตามมาตรฐานโปรฟิบัส ทำได้ยาก เนื่องจากความเร็วที่ใช้ในการสื่อสารตามมาตรฐานทำงานที่ความเร็วสูงมาก

5.2 แนวทางการพัฒนาต่อ

แนวทางการพัฒนางานวิจัยต่อไปในอนาคต สามารถนำไปประยุกต์ให้เข้ากับการใช้งานต่างๆ ภายในเครือข่ายมาตรฐานโปรฟิบัส ได้ดังนี้

1. สร้างอุปกรณ์เชื่อมต่อกับการ์ดฟิวบัสที่สร้างขึ้น เนื่องจากใช้ชุดคอม 86 ในการพัฒนาดังนั้นสามารถนำอุปกรณ์ภาคสนามของระบบอุตสาหกรรมอัตโนมัติ เช่น Sensor มาต่อเข้ากับ ชุดคอม 86 ได้หลายทาง เช่น Ethernet หรือ ISA
2. สร้างการ์ดฟิวบัสของอุปกรณ์หลัก เนื่องจากว่าโครงสร้างพื้นฐานการทำงานจะคล้ายๆ กันระหว่างอุปกรณ์หลัก และอุปกรณ์รอง แต่อาจจะมีการทำงานที่ละเอียดและซับซ้อนกว่า
3. สร้าง Library สำหรับทำงานเฉพาะอย่างของอุปกรณ์รอง โดยเรียกใช้ Library พื้นฐานที่สร้างไว้สำหรับใช้งาน
4. สร้าง Option การทำงานของอุปกรณ์รองเพิ่มเติม จากพื้นฐานการทำงานที่มีให้

5.3 บทวิจารณ์

ในปัจจุบันแนวโน้มของระบบอุตสาหกรรมภายในประเทศมีความตื่นตัวมากขึ้น แต่เนื่องจากความพร้อมทางด้านอุปกรณ์ และเครื่องมือที่รองรับการเจริญเติบโตยังมีน้อยอยู่ ทำให้โรงงานส่วนใหญ่ต้องนำเข้าเครื่องจักรหรืออุปกรณ์จากต่างประเทศ ทำให้ประเทศชาติสูญเสียรายได้เป็นอย่างมาก

แนวคิดในการสร้างอุปกรณ์เพื่อทดแทนการนำเข้าจากต่างประเทศ และสามารถใช้งานได้กับมาตรฐานการสื่อสารต่างๆที่มีใช้อยู่ในระบบอุตสาหกรรมทั่วไปจึงเกิดขึ้น ผลงานวิจัยชิ้นนี้เป็นส่วนหนึ่งของการสร้างนาวทางในการพัฒนาอุปกรณ์ตามมาตรฐาน ที่ได้รับความนิยมใช้งานภายในอุตสาหกรรมแบบอัตโนมัติซึ่งก็คือ มาตรฐานโปรฟิบัส

ถึงแม้ว่าการค้นคว้าหาข้อมูลของมาตรฐานโปรฟิบัสภายในประเทศนั้นจะทำได้ยาก แต่คณะผู้วิจัยอาศัยความพยายามเต็มที่ ในการค้นหาแหล่งข้อมูลจากภายนอก แม้ว่าจะสามารถค้นหาได้

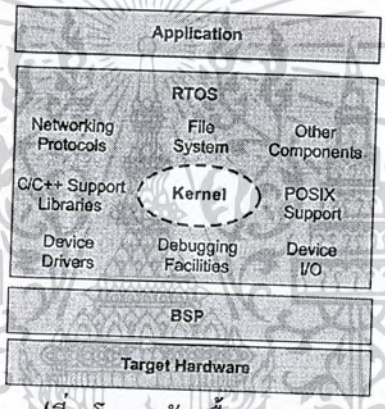
ยากลำบาก เนื่องมาจากว่า การที่จะเข้าไปศึกษาหรือค้นหาข้อมูลจากแหล่งต่างๆได้นั้น จะกระทำได้เฉพาะผู้ที่มีสิทธิ์ ซึ่งหมายถึงต้องเป็นสมาชิกขององค์กร หรือเวปไซค์เท่านั้น และในบางครั้งต้องเสียค่าใช้จ่ายในการเข้าสมัครเป็นสมาชิกดังกล่าวด้วย

ทั้งนี้ต้องขอขอบคุณอาจารย์ที่ปรึกษางานวิจัยที่คอยค้นหาแหล่งข้อมูล หรือแนะนำบุคคลที่มีความรู้เกี่ยวกับมาตรฐานโปรฟีบัสภายในประเทศ เพื่อช่วยเป็นที่ปรึกษา หรือชี้แนะแนวทางในการพัฒนา รวมทั้งเอื้อเฟื้อแหล่งข้อมูลที่เป็นประโยชน์ต่องานวิจัยในครั้งนี้

ภาคผนวก ก.
ระบบปฏิบัติการแบบเวลาจริง

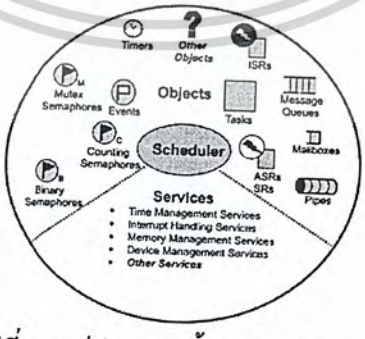
ในปัจจุบัน ระบบปฏิบัติการแบบเวลาจริง (RTOS-Real Time Operating System) เป็นหัวใจหลักในการจัดการของระบบฝังตัว เนื่องจากรูปแบบซอฟแวร์จะขึ้นอยู่กับการสร้างแอพลิเคชั่น และ แอพลิเคชั่นส่วนใหญ่ต้องการการจัดลำดับการทำงาน (Scheduling) ดังนั้นจึงจำเป็นต้องใช้ RTOS

RTOS เป็นโปรแกรมที่จัดการเกี่ยวกับลำดับการทำงานให้เหมาะสมกับช่วงเวลา เกี่ยวกับการจัดการทรัพยากร และการกระทำพื้นฐานสำหรับการพัฒนาแอพลิเคชั่น ทุก ๆ RTOS จะมีส่วนทำงานหลัก (Kernel) และสามารถมีหลายๆ ส่วนประกอบรวมกัน เช่น ระบบไฟล์ (File system), โครงสร้างโพรโตคอลเครือข่าย (Network Protocol Stack) และ องค์ประกอบที่สำคัญในการความต้องการที่เฉพาะของแต่ละแอพพลิเคชั่น



รูปที่ 1 โครงสร้างพื้นฐาน RTOS

ถึงแม้ว่า RTOS ส่วนใหญ่จะขยาย หรือลดความสามารถในการทำงานได้ตามความต้องการ แต่การทำงานหลักที่สำคัญที่ทุก RTOS จำเป็นต้องมี ได้แก่



รูปที่ 2 องค์ประกอบพื้นฐาน ของ RTOS

1. Scheduler

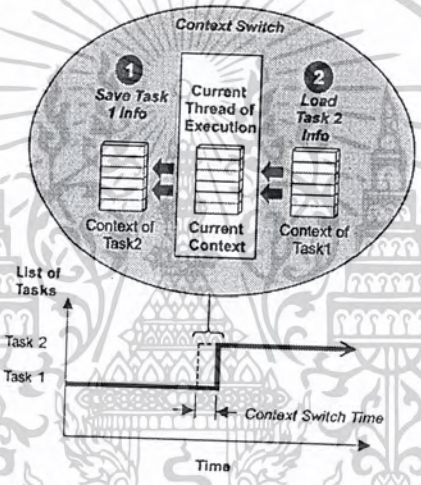
เป็นหัวใจหลักของทุก ๆ Kernel และจัดการเกี่ยวกับการทำงานของแทส (Task) ให้เป็นไปตามอัลกอลิทึม (Algorithm) ที่ตั้งไว้ ตัวอย่าง Algorithm คือ Preemptive schedule and Round-robin เมื่อ Task ทำงานจะพิจารณา algorithm ที่เหมาะสมในการจัดการ Task มีส่วนประกอบที่สำคัญ ได้แก่

1.1 Schedulable Entity

เป็นองค์ประกอบของ Kernel ที่มีการแย่งยิงช่วงเวลาการทำงานบนระบบ ตาม algorithm ที่กำหนด โดยที่จะมีการสื่อสารกันระหว่าง Task

1.2 Multitasking

เป็นความสามารถของระบบปฏิบัติการที่สามารถจัดการกับการกระทำหลายๆ อย่างพร้อมๆ กัน ในช่วงเวลาที่กำหนด RTOS มีหลาย task ที่จำเป็นต้องจัดลำดับในการทำงาน โดยจะมีการสลับการทำงานของแต่ละ Task ตาม algorithm



รูปที่ 3 การทำงานของ Context Switching และ Multitasking

1.3 The Context Switch

แต่ละ Task จะมีส่วนข้อมูล (Context) เป็นของตนเอง context switch จะเกิดขึ้นเมื่อตัวจัดการ (Scheduler) จะเปลี่ยนการทำงานหนึ่งไปเป็นอีกงานหนึ่ง

1.4 The Dispatcher

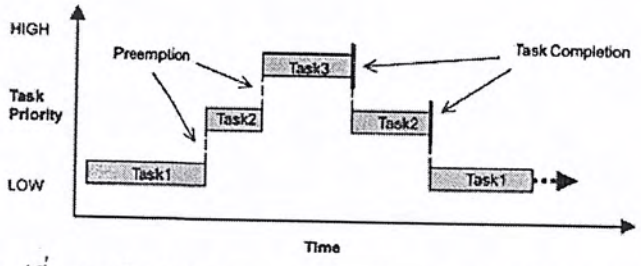
เป็นส่วนหนึ่งของ Scheduler โดยเป็นผู้จัดการเกี่ยวกับการทำ context switching และเปลี่ยนแปลงลำดับขั้นตอนการทำงาน ทำหน้าที่ส่งผ่านส่วนควบคุมการทำงานของ task

1.5. Scheduling Algorithms Kernel

ส่วนใหญ่จะมี Algorithm พื้นฐานอยู่ 2 อย่างคือ

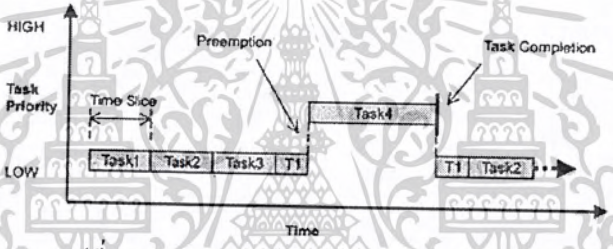
1. Preemptive Priority-Based Scheduling โดยแต่ละ Task จะมี ค่าลำดับความสำคัญ (Priority) ในการทำงานเป็นของตัวเอง Task ที่มีค่าลำดับการทำงานสูงกว่า จะได้ทำงานก่อน ถ้าในขณะที่ Task ใดทำงานอยู่ แล้วมี Task ที่มีค่าลำดับความสำคัญสูงกว่า เข้ามา Task ที่

ทำงานอยู่ จะถูกให้หยุดทำงานชั่วคราว เพื่อที่จะให้ Task มีค่าสูกว่าทำงานก่อน เมื่อ Task ดังกล่าวทำงานเสร็จ Task เดิมจะกลับมาทำงานต่อ โดยที่ลำดับความสำคัญของ task จะถูกเปลี่ยนแปลงเสมอ



รูปที่ 4 การทำงานของ Preemptive Priority-Based Scheduling

2. Round-Robin Scheduling แต่ละ Task จะสามารถทำงานได้เท่าๆ กัน โดยในการทำงานของแต่ละ Task จะมีการกำหนดช่วงเวลาไว้



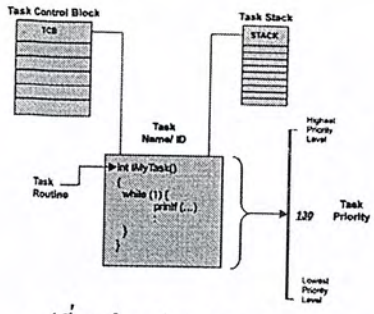
รูปที่ 5 การทำงานของ Round-Robin Scheduling

2. Objects

เป็นโครงสร้าง Kernel ที่มีลักษณะพิเศษที่ช่วยสำหรับการพัฒนาในการสร้างแอพลิเคชั่นสำหรับระบบฝังตัวที่ทำงานแบบเวลาจริง ส่วนประกอบที่สำคัญ ได้แก่

2.1 Tasks

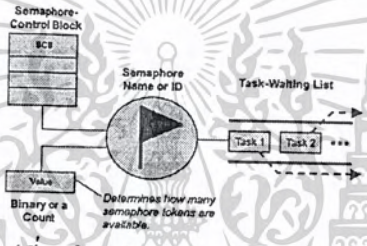
แต่ละ Task อิสระจากการทำงานระหว่างกัน มีการแย่งชิงการใช้ CPU ในการทำงานระหว่างกัน ในแต่ละ Task จะถูกกำหนดค่าตัวแปรต่าง ๆที่แตกต่างกัน และจะมีโครงสร้างข้อมูลเป็นของตัวเอง เมื่อมีการสร้าง Task ขึ้นจะมีส่วนที่เกี่ยวข้อง คือ ชื่อ, หมายเลขประจำของ Task, ค่าลำดับความสำคัญ, ส่วนควบคุม Task (TCB - task Control Block), Stack Task Routine



รูปที่ 6 ส่วนประกอบของ Task

2.2 Semaphores

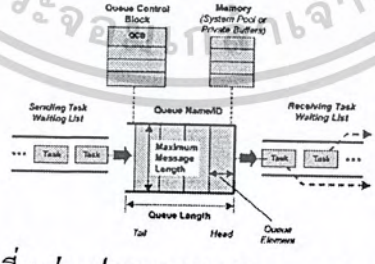
เมื่อ Semaphore ถูกสร้างขึ้น จะมีค่าที่เกี่ยวข้อง ได้แก่ ส่วนควบคุม (SCB-semaphore control Block) หมายเลขประจำที่ไม่ซ้ำกัน ค่าโทเคนของ semaphore และ รายการของ Task ที่รอใช้ เมื่อมี Task ได้สิทธิ์เข้าใช้ semaphore ค่าโทเคนที่เก็บจะเพิ่มขึ้น เมื่อ Task ออกจากการใช้ ค่า โทเคนจะลดลง เมื่อค่าโทเคนเป็น 0 จะเกิดการ block จน กว่าค่าจะมากกว่า 0



รูปที่ 7 ส่วนประกอบของ Semaphore

2.3 Message Queues

เป็น ตัวกันชน (Buffer) ที่ใช้ในการแลกเปลี่ยนข้อมูลโดยการส่งข้อความระหว่าง Task ลักษณะคล้ายท่อ โดยจะถือครองข้อความ จนกว่าจะมีการอ่านข้อความจากผู้รับโดยแต่ละ message queues จะถือครองได้เพียงหนึ่งข้อความเท่านั้น



รูปที่ 8 ส่วนประกอบของ Message Queues

3. Service

เป็นการทำงานที่ Kernel กระทำกับองค์ประกอบอื่นๆ หรือ การทำงานโดยทั่วๆ ไป เช่น Timing, Interrupt, การจัดการทรัพยากร (Resource)

ภาคผนวก ข.

Micro Controller Operating System Version 2

การพัฒนางานวิจัยชิ้นนี้ คณะนักวิจัยเลือก uCOS-II มาใช้งานในส่วนของ RTOS เนื่องจาก uCOS-II สร้างมาเพื่อการพัฒนากับระบบฝังตัว มีขนาดเล็ก พัฒนาโดยใช้ภาษา Borland C/C++ และมีการใช้งานที่ง่าย และสะดวก สามารถใช้งานกับสถาปัตยกรรมที่หลากหลาย

1. คุณสมบัติพื้นฐาน µC/OS –II

คุณสมบัติพื้นฐานของ µC/OS –II ที่สำคัญได้แก่ การพอร์ต ที่สามารถใช้งานกับสถาปัตยกรรม 8, 16, 32, 64 บิต มีการทำงานแบบ Preemptive คือ Task ที่มีค่าความสำคัญสูงสุดจะถูกทำงานก่อน สามารถทำ Multitasking ได้ถึง 64 task จองไว้ให้กับระบบ 8 Task และ สำหรับแอพลิเคชั่น 56 Task แต่ละ Task จะมี Stack เป็นของตนเอง นอกจากนี้ยังสามารถบอกเวลาที่ใช้ในการทำงานของฟังก์ชั่น (Deterministic) และสามารถจัดการกับ Interrupt ได้ซ้อนกันถึง 255 ระดับ

2. คำสั่งพื้นฐาน

คำสั่งที่ใช้ใน uCOS-II มีมากหมาย ในส่วนนี้ผู้วิจัยจะขอยกตัวอย่างของบางคำสั่งที่เป็นคำสั่งพื้นฐานในการใช้งาน uCOS-II เพื่อเป็นพื้นฐานให้กับผู้สนใจ และต้องการนำไปใช้งาน

2.1 ชนิดของตัวแปร

ชนิดของตัวแปรภายใน uCOS-II กำหนดให้ใช้ได้ดังรูป 2-39 โดยจะมีชนิดของตัวแปรที่สามารถใช้งานได้ทุกสถาปัตยกรรม

```
typedef unsigned char   BOOLEAN;
typedef unsigned char   INT8U;          /* Unsigned  8 bit quantity */
typedef signed   char   INT8S;          /* Signed    8 bit quantity */
typedef unsigned int    INT16U;         /* Unsigned 16 bit quantity */
typedef signed   int    INT16S;         /* Signed   16 bit quantity */
typedef unsigned long   INT32U;         /* Unsigned 32 bit quantity */
typedef signed   long   INT32S;         /* Signed   32 bit quantity */
typedef float           FP32;           /* Single precision floating point */
typedef double          FP64;           /* Double precision floating point */

typedef unsigned int    OS_STK;         /* Each stack entry is 16-bit wide */

#define BYTE            INT8S
#define UBYTE           INT8U
#define WORD            INT16S
#define UWORD           INT16U
#define LONG            INT32S
#define ULONG           INT32U
```

รูปที่ 1 ชนิดของตัวแปร

2.2 การสร้าง Task

uC/OS-II สามารถจัดการได้ถึง 64 Task โดยจะมีค่าลำดับความสำคัญแบบสูง 4 ค่า และแบบต่ำ 4 ค่า ในการสร้าง Task จะมีค่าตัวแปรที่เกี่ยวข้องดังแสดงในรูปที่ 2-40

```
OSTaskCreate(*task, *pdata, *ptos, prio)
/*
        task คือ pointer ที่ชี้ไปยัง ตำแหน่ง Code
        pdata คือ pointer ของตัวแปรที่ส่งมายัง Task
        ptos คือ pointer ที่จุด stack ของ Task
        prio คือ ค่าความสำคัญของ Task
*/
```

รูปที่ 2 การสร้าง Task

2.3 การกำหนดการทำงานภายใน Task

หลังจากที่สร้าง Task จำเป็นต้องกำหนดการทำงานให้กับ Task นั้น ๆ โดยการทำงานของ task จะทำงานไปเรื่อยๆ ไม่มีการสิ้นสุด เว้นแต่จะกำหนดเงื่อนไขในการหยุดการทำงานไว้

```
Void myTask(void *pdata)
{
        for (;;)
        {
            /* User Code */
        }
}
```

รูปที่ 3 การกำหนดการทำงานภายใน Task

2.4 การเริ่มใช้งาน uCOS-II

ในการเริ่มต้นใช้งานส่วนของ uC/OS-II จำเป็นต้องสร้าง Task เริ่มต้นในการใช้งานขึ้นมา 1 Task เพื่อทำหน้าที่เป็น Task หลักในการควบคุมการทำงานของ Task ทั้งหมด

```
Void main(void)
{
        OSInit;
        // create your startup task
        OSStart();
}
Void TaskStart(void *pdata)
{
        OSStatInit();
        for (;;)
        {
        //code for TaskStart() goes here
        }
}
```

รูปที่ 4 การเริ่มใช้งาน uCOS-II

2.5 การใช้ Semaphore

การใช้งาน Semaphore ขั้นแรกต้องมีการประกาศค่าตัวแปรให้เป็นชนิดของ semaphore หลังจากนั้น จึงทำการสร้าง semaphore ขึ้นมาโดยจะมีการใช้งาน 2 แบบ คือ การเข้าถือครอง semaphore และ การปล่อยการถือครอง semaphore

```
OS_EVENT *DispSem      <-ประกาศตัวแปร
Void main (void)
{
   OSInit();

   DispSem = OSemCreate(1);  <-สร้าง semaphore

   OSStart();

}

Void task1 (void *pdata)
{
   INT8U err;
   for (;;)
   {
    OSSemPend(DispSem,0,&err);  <-คืนค่า semaphore

}

Void task2 (void *pdata)
{
   INT8U err;
   for (;;)
   {
    err = OSSemPost(DispSem);  <-ปิดครอง semaphore

}
```

รูปที่ 5 การใช้ Semaphore

ภาคผนวก ค.

Com86

อุปกรณ์ฝังตัวที่คณะวิจัยเลือกมาใช้ในการพัฒนา คือ ชุด Com86 เนื่องจากสามารถทำงานในระบบเวลาจริงได้ และมีคู่มือการใช้งานที่ง่ายต่อการพัฒนา มีคอมไพล์ที่สามารถใช้งานคู่กับ uCOS-II

1. คุณสมบัติพื้นฐาน

คุณสมบัติพื้นฐานของชุด Com86 ที่สำคัญมีดังต่อไปนี้

1. สถาปัตยกรรมไมโครโปรเซสเซอร ▪ ตระกูล x86 บิต ทำงานที่ความเร็ว 24 เมกกะเฮิร ▪ ต (MHz)
2. ดีแรม 1 เมกกะไบต์1 MBytes DRAM)
3. แฟลตไบออส 64 กิโลไบต์(64 kBytes Flash BIOS)

2. จุดเด่น Com86

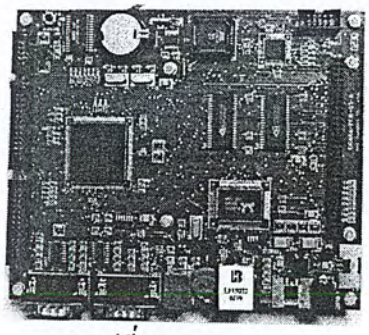
ชุด Com86 มีจุดเด่นที่สำคัญอยู่หลายด้าน ได้แก่

1. ไมโครโปรเซสเซอร์ที่มีความสามารถสูง ทำให้มีความสามารถในการประมวลผลต่างๆ นอกจากนี้ ภายในตัวไมโครโปรเซสเซอรเองยังได้ รวมเอาอุปกรณ์พื้นฐานต่างๆที่จำเป็นในการใช้งานรวมไว้ภายใน ทำให้ง่ายต่อการพัฒนา
2. มีเครื่องมือการพัฒนาโปรแกรมที่เหมือนกับการใช้งานเครื่องคอมพิวเตอร์ทั่วไป ทำให้ลดเวลาในการพัฒนา และสามารถพัฒนาได้ง่ายขึ้น
3. ความสามารถในการเพิ่มเติมส่วนขยายได้ทั้งส่วนฮาร์ดแวร์และซอฟแวร์

3. การประยุกต์ใช้งาน

ด้วยความสามารถที่หลากหลายของชุด Com86 จึสามารถนำไปประยุกต์ใช้งานได้มากมาย เช่น

1. ประยุกต์ใช้ในระบบอุตสาหกรรม
2. การพัฒนาเวปเซิรฟเวอร์
3. การทำหุ่นยนต์
4. ระบบควบคุมอุปกรณ์



รูปที่ 1 ชุด Com86

บรรณานุกรม

[1] European Standard EN50170 : *"PROFIBUS Specificatio"* Volume2. Edition 1.0

[2] European Standard EN50170 : *"Normative Parts of PROFIBUS-FMS,-DP,-PA"* Volume2. Edition 1.0

[3] PROFIBUS utzerorganisation e.V : *"GSD-Specification for PROFIBUS-DP"*, Karlsruhe. 1998.

[4] PROFIBUS utzerorganisation e.V : *"GSD Revision 2 Version 1.0"*, Karlsruhe. 1998

[5] Josef Weigmann, Gerhard Kilian : *"Decentralization with PROFIBUS-DP"*, Germany. 2000.

[6] Michael Barr : *"Programming Embedded Systems in C and C++"*, O'Reilly. 1999.

[7] Qing Li and Carolyn Yao : *"Real-Time Concepts for Embedded Systems"*, CMP Book. 2003.

[8] Jeremy BenThem : *"TCP/IP Lean Web Server For Embedded Systems"*, CMP Book Lawrence. 2000.

[9] Jean J. Labrosse : *"MicroC/OS-II The Real-Time Kernal"*, RD Book. 1999.

[10] Acromag : "Introduction to PROFIBUS DP", USA. 2002.