ระบบการติดต่อสื่อสาร และการควบคุมด้วยโปรโตคอล MODBUS
กรณีศึกษา บริษัท โนเวมเอนจิเนียริ่ง จำกัด

COMMUNICATION AND CONTROL SYSTEM
BASED ON MODBUS PROTOCOL
CASE STUDY: NOVEM ENGINEERING CO., LTD.

นางสาวรสวันต์ รองวงศ์

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีไทย – ญี่ปุ่น
พ.ศ. 2555

ระบบการติดต่อสื่อสาร และการควบคุมด้วยโปรโตคอล MODBUS

กรณีศึกษา บริษัท โนเวมเอนจิเนียริ่ง จำกัด

COMMUNICATION AND CONTROL SYSTEM BASED ON MODBUS PROTOCOL

CASE STUDY: NOVEM ENGINEERING CO., LTD.

นางสาวรสวันต์ รองวงศ์

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

พ.ศ. 2555

คณะกรรมการสอบ

.....ประธานกรรมการสอบ

(อาจารย์ ดร.วรากร ศรีเชวงทรัพย์)

.....กรรมการสอบ

(อาจารย์ประเวศน์ เอื้อตรงจิตต์)

.....อาจารย์ที่ปรึกษา

(อาจารย์ศิษฏ์ชฎา อ่ำเทศ)

.....ประธานสหกิจศึกษาสาขาวิชา

(อาจารย์ ดร.วรากร ศรีเชวงทรัพย์)

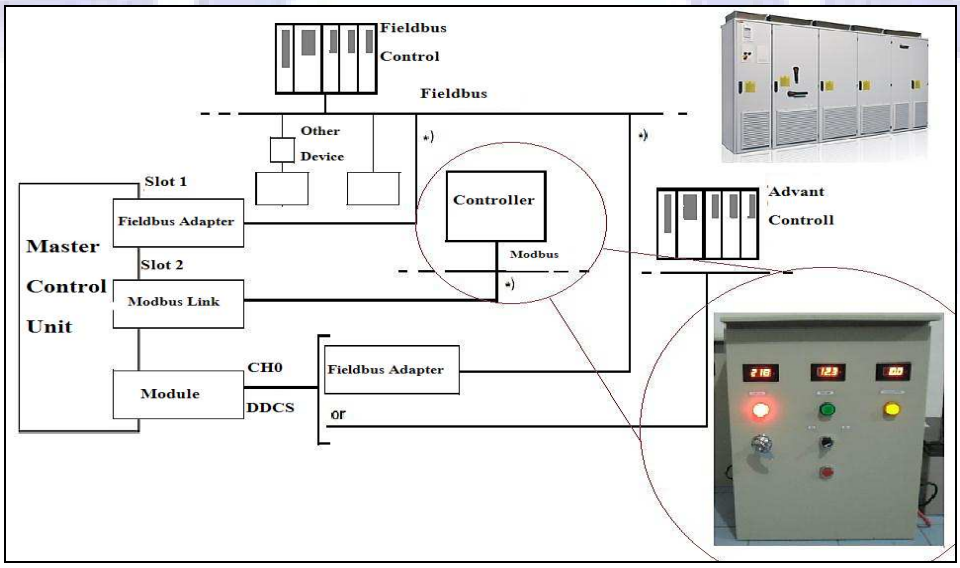
ชื่อโครงงาน	ระบบการติดต่อสื่อสาร และการควบคุมด้วยโปรโตคอล MODBUS กรณีศึกษา บริษัท โนเวมเอนจิเนียริ่ง จำกัด COMMUNICATION AND CONTROL SYSTEM BASED ON MODBUS PROTOCOL CASE STUDY: NOVEM ENGINEERING CO., LTD.
ผู้เขียน	นางสาวรสวันต์ รองวงศ์
คณะวิชา	วิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ศิษฏ์ชฎา อ่ำเทศ
พนักงานที่ปรึกษา	1. นายสุชาติ ศักดิ์ชัยกุล 2. นายสุชาติ ตันติดำรงค์
ชื่อบริษัท	NOVEM ENGINEERING CO., LTD.
ประเภทธุรกิจ/สินค้า	ผู้ผลิต และออกแบบเครื่องอินเวอร์เตอร์

บทสรุป

สืบเนื่องจากการทำโครงงานสหกิจศึกษา ณ บริษัท โนเวม เอนจิเนียริ่ง จำกัด ในช่วงเวลาอันมีค่านี้ ผู้จัดทำโครงงานได้เรียนรู้ และทำการศึกษาระบบการติดต่อสื่อสาร รวมถึงเรียนรู้วิธีการควบคุมการทำงานของคอนโทรลเลอร์ (Controller) ด้วยโปรโตคอล Modbus (Modbus Protocol) ซึ่งเป็นโปรโตคอลเกี่ยวกับงานด้านการติดต่อสื่อสาร (Communication Protocol) ที่มีการใช้งานอย่างกว้างขวาง โดยออกแบบมาให้สามารถใช้งานได้ดีใน 3 ระดับชั้นคือ ในระดับชั้นของการประยุกต์ใช้งานติดต่อได้โดยตรงกับผู้ใช้ (Application layer) ในระดับเครือข่าย (Network layer) และในระดับส่วนติดต่อทางกายภาพของฮาร์ดแวร์ (Physical layer) ในกระบวนการเริ่มต้นศึกษานั้น ผู้จัดทำโครงงานได้เริ่มด้วยการศึกษาระบบการติดต่อสื่อสารในงานภาคอุตสาหกรรม พบว่ามีความแตกต่างกับระบบการติดต่อสื่อสารบนเครือข่ายทั่ว ๆ ไป โดยการศึกษาครั้งนี้ได้เน้นหนักไปที่โครงสร้าง และหลักการทำงานของโปรโตคอล Modbus เพื่อนำไปประยุกต์ใช้งานกับเครื่องอินเวอร์เตอร์ จุดประสงค์หลักเพื่อให้ผู้จัดทำโครงงานสามารถวางโครงสร้างต่าง ๆ ในการส่งผ่านข้อมูลระหว่างผู้ใช้งานกับอุปกรณ์อิเล็กทรอนิกส์ได้ กล่าวคือเป็นส่วนที่ทำให้ผู้ใช้สามารถส่งคำสั่งให้อุปกรณ์ทำงานตามที่ต้องการได้นั่นเอง ในส่วนของการสั่งงานต่าง ๆ นั้นจะทำงานได้โดยผ่าน

การควบคุมจากคอนโทรลเลอร์ ซึ่งโครงงานนี้ได้เฉพาะเจาะจงไปที่ TMS320F2808 หรือ DSP2808 เป็นตัวควบคุมที่มีราคาถูก และมีคุณสมบัติเพียงพอต่อการทดสอบเพื่อควบคุมการทำงานเบื้องต้น

จากการได้ศึกษาการทำงาน และวิธีใช้งานเกี่ยวกับโปรโตคอล Modbus ผู้จัดทำโครงงานได้ทดสอบในเบื้องต้นด้วยการเขียนโปรแกรมภาษาซี เพื่อควบคุมคอนโทรลเลอร์ TMS320F2808 โดยคำสั่งต่าง ๆ จะอาศัยโปรโตคอล Modbus เป็นตัวกลางในการติดต่อสื่อสารกับอุปกรณ์ในระบบ ในที่นี้ใช้โปรแกรมตัวอย่าง SCI Echoback และเชื่อมต่ออุปกรณ์ผ่านพอร์ตอนุกรม (Serial Port) ชนิด EIA-232 หรือ RS232 จากนั้นทำการแสดงผลผ่านโปรแกรม Hyperterminal โดยลักษณะของโปรแกรมจะเป็นการสะท้อนข้อมูลที่ผู้ใช้ป้อนเข้าไปและแสดงผลกลับมายังจอแสดงผลข้อมูล ซึ่งการทดลองดังกล่าวได้มีการนำไปประยุกต์ใช้กับการเขียนโปรแกรมควบคุมการทำงานของอินเวอร์เตอร์ในส่วนของการแสดงผลข้อมูลได้เป็นอย่างดี อีกทั้งการวางโครงสร้างของระบบการติดต่อสื่อสารด้วยโปรโตคอล Modbus ทำให้ระบบการจัดการข้อมูลของเครื่องอินเวอร์เตอร์ทำได้ง่ายขึ้น เป็นเหตุผลที่สถานประกอบการนำโปรโตคอล Modbus มาประยุกต์ใช้งาน นอกจากนี้ข้อดีที่สามารถทำการแก้ไขได้ง่ายแล้วนั้น โปรโตคอล Modbus ยังเป็นโปรโตคอลที่ไม่มีค่าใช้จ่ายในการนำไปใช้งาน ทำให้ลดต้นทุนในส่วนของระบบการติดต่อสื่อสารบนผลิตภัณฑ์ได้อีกด้วย โดยที่โครงสร้างของโปรโตคอล Modbus และการแสดงผลข้อมูลบนเครื่องอินเวอร์เตอร์นำแสดงดังรูป



กิตติกรรมประกาศ

การที่ข้าพเจ้าได้เข้ามาสหกิจศึกษา ณ บริษัท โนเวมเอนจิเนียริ่ง จำกัด ตั้งแต่วันที่ 18 มิถุนายน 2555 จนถึงวันที่ 6 ตุลาคม 2555 ส่งผลให้ข้าพเจ้ามีความรู้และมีประสบการณ์การทำงานจากการทำงานจริงในสถานประกอบการ ซึ่งมีค่าเป็นอย่างมาก และส่งผลให้ข้าพเจ้าสามารถนำสิ่งต่างๆ เหล่านั้น มาใช้พัฒนาทักษะของตนเอง การเข้าสหกิจศึกษาในครั้งนี้จะสำเร็จลุล่วงไม่ได้หากขาดความช่วยเหลือจากบุคลากรในสถานประกอบการทุกท่านที่คอยให้คำปรึกษา และแนะนำการการทำงาน ในที่นี้ข้าพเจ้าขอขอบคุณผู้ควบคุมการทำงานคือ

1. คุณสุชาติ ศักดิ์ชัยกุล (ผู้จัดการทั่วไป) ที่เห็นความสำคัญของการสหกิจศึกษา และได้ให้โอกาสที่มีคุณค่ายิ่ง แก่ข้าพเจ้าในการเข้ามาสหกิจศึกษาที่บริษัทแห่งนี้
2. คุณสุชาติ ตันติดำรงค์ (ผู้จัดการฝ่ายวิศวกรรม) ที่ได้ให้ความรู้และคอยให้คำปรึกษาต่างๆ ในการปฏิบัติงาน

ตลอดจนบุคลากรท่านอื่นๆภายในแผนกที่ไม่ได้กล่าวนามทุกท่าน ที่ให้คอยให้คำแนะนำและช่วยเหลือข้าพเจ้าตลอดระยะเวลาในการสหกิจศึกษาทั้งสิ้น 16 สัปดาห์ จนทำให้การสหกิจศึกษาในครั้งนี้ประสบความสำเร็จไปได้ด้วยดี ข้าพเจ้าจึงขอขอบคุณไว้ ณ ที่นี้ด้วย

นางสาวรสวันต์ รองวงศ์
ผู้จัดทำโครงงาน
6 ตุลาคม 2555

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

บทที่

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูปประกอบ

รูปที่ **หน้า**

สารบัญรูปประกอบ(ต่อ)

รูปที่ **หน้า**

สารบัญรูปประกอบ(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ชื่อและที่ตั้งของสถานประกอบการ

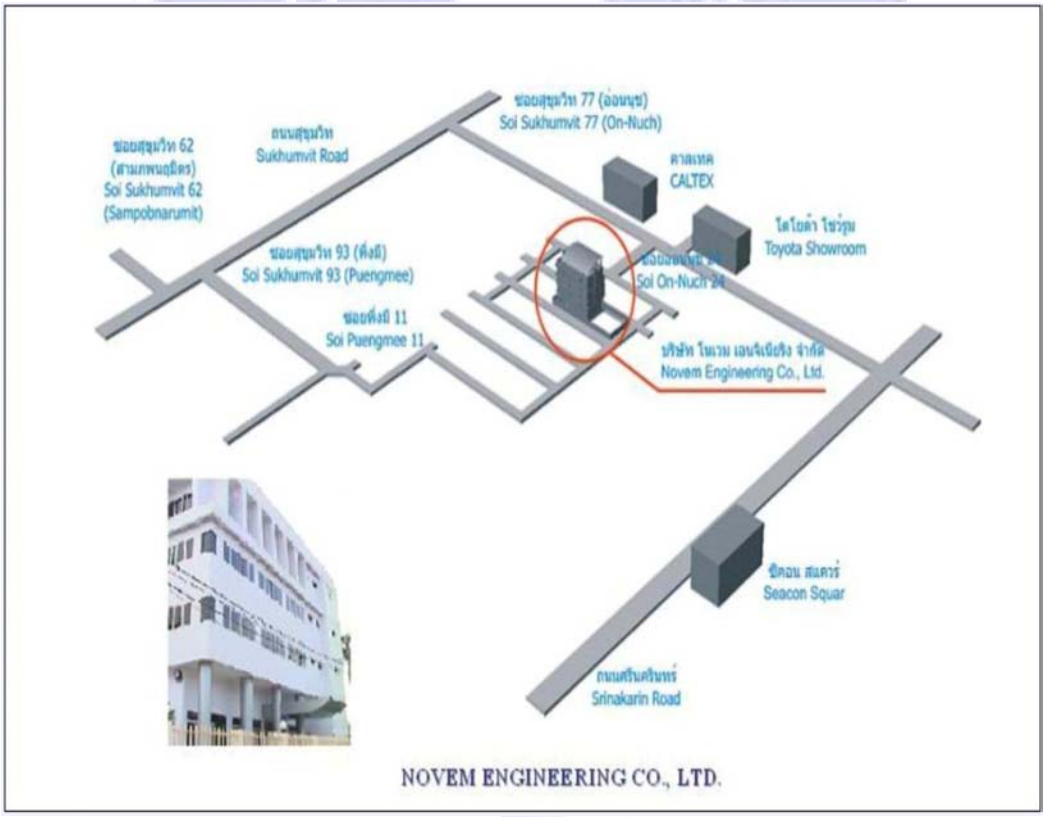
สถานประกอบการ : NOVEM ENGINEERING CO., LTD.

ที่ตั้ง: 227/1 ซอยพึ่งมี 11 ถนนสุขุมวิท แขวงบางจาก เขตพระโขนง กรุงเทพมหานคร 10260

โทร: 023349941

Home page: http://www.novemeng.com

แผนที่ของบริษัท:



รูปที่ 1.1 แผนที่ Novem Engineering CO., LTD.

1.2 ลักษณะธุรกิจของสถานประกอบการ

รูปที่ 1.2 เครื่อง Inverter NSI (ผลิตภัณฑ์ของบริษัท)

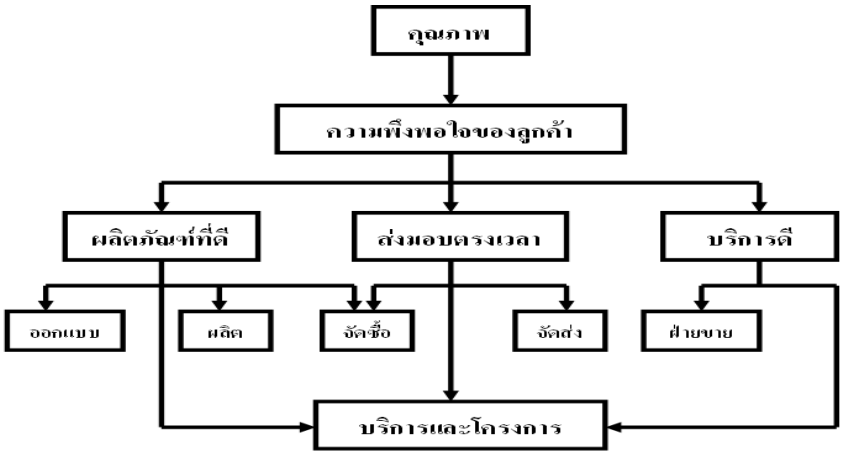
บริษัทโนเวมเอนจิเนียริ่ง จำกัด เป็นสถานประกอบการซึ่งเป็นผู้ผลิต จำหน่าย ออกแบบ ปรับปรุง ติดตั้ง ตรวจซ่อมและให้คำปรึกษาในเรื่องของการประหยัดพลังงานไฟฟ้าการผลิตไฟฟ้าและวงจรอิเล็กทรอนิกส์ในโรงงานอุตสาหกรรมทุกประเภทและอาคารขนาดใหญ่ โดยการบริการขององค์กรจะเน้นในเรื่องของการตรวจซ่อมวงจรอิเล็กทรอนิกส์และเครื่องปรับความเร็วรอบมอเตอร์ทุกยี่ห้อ อาทิเช่นอินเวอร์เตอร์ ดีซีไดรฟ์วีเอสคอนโทรล รวมไปถึงการออกแบบและประกอบตู้เมนท์ไฟฟ้า MCB ตู้ควบคุม Power Factor และรับเหมาเดินระบบไฟฟ้ากำลังภายในโรงงานอุตสาหกรรม ทั้งนี้ยังให้คำปรึกษาและจัดทำระบบการประหยัดพลังงานไฟฟ้าที่เกี่ยวข้องกับมอเตอร์เป็นหลักโดยจะมีบริการหลังการขายแก่กลุ่มลูกค้าอีกด้วย

1.3 รูปแบบการจัดองค์กรและบริหารองค์กร

ในการบริหารงานขององค์กรจะเน้นในเรื่องคุณภาพและความพึงพอใจของลูกค้าเป็นหลักทั้งในเรื่องของผลิตภัณฑ์และการบริการต่าง ๆ เพื่อให้เป็นที่ยอมรับของตลาดและกลุ่มลูกค้า ซึ่งองค์กรมีนโยบายทางด้านคุณภาพว่า "พันธกิจของเรา คือ จะมุ่งพัฒนาอย่างต่อเนื่องและเสริมสร้างความพึงพอใจบริการที่ดีให้ลูกค้ากับผลิตภัณฑ์แหล่งจ่ายไฟฟ้าเพื่อขับเคลื่อนมอเตอร์" โดยหลักการในการบริหารงานเป็นดังนี้

1.3.1 หลักการ "คุณภาพเกิดขึ้นได้อย่างไร"

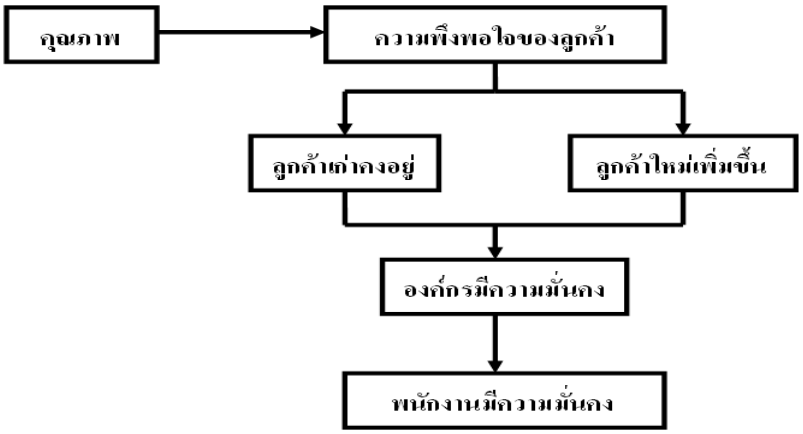
เมื่อสินค้ามีคุณภาพย่อมเป็นที่พึงพอใจของลูกค้า โดยการที่จะทำให้ลูกค้ามีความพึงพอใจ คือ ผลิตภัณฑ์ที่ดี ส่งมอบตรงเวลา และการบริการดี ทั้งในเรื่องของการออกแบบ ผลิต จัดซื้อ จัดส่ง และฝ่ายขาย ซึ่งนั่นคือหัวใจหลักของการบริการ



รูปที่ 1.3 แผนผัง "คุณภาพเกิดขึ้นได้อย่างไร"

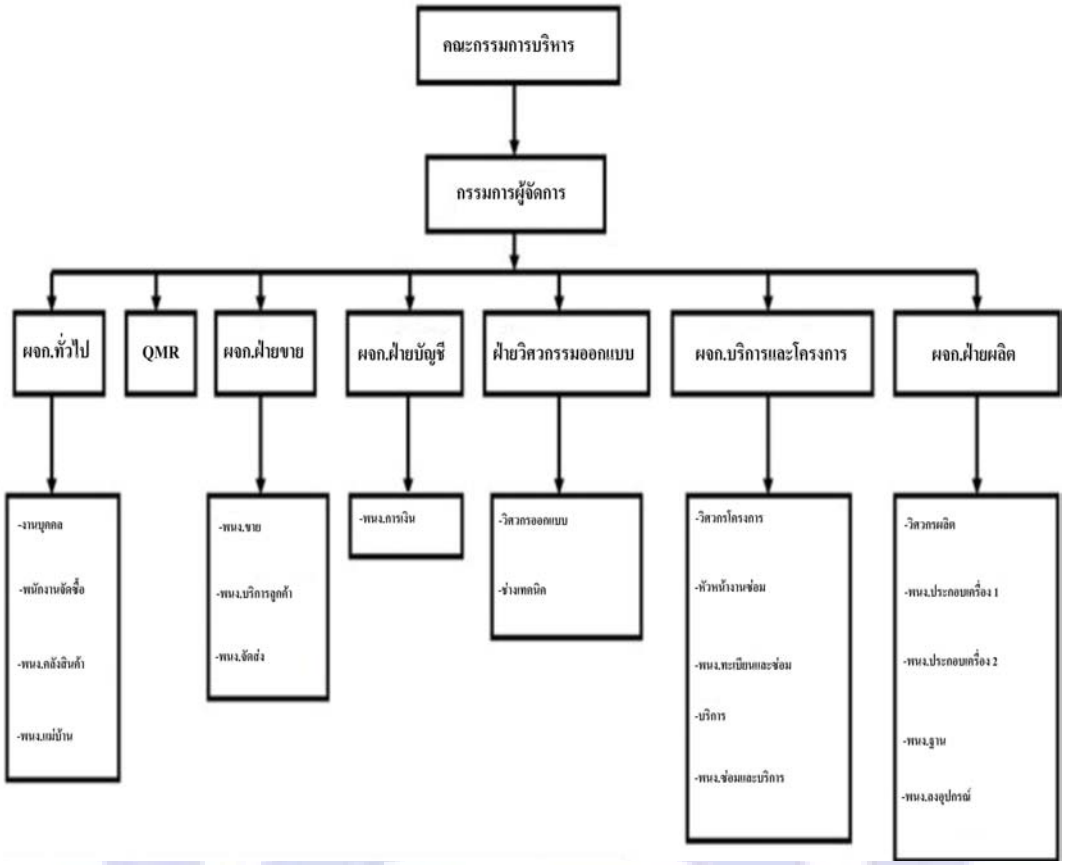
1.3.2 หลักการ "ลูกค้าสำคัญอย่างไร"

เมื่อสินค้ามีคุณภาพเป็นผลให้ลูกค้ามีความพึงพอใจ ลูกค้ารายเก่ายังคงอยู่ และลูกค้ารายใหม่เพิ่มขึ้นทำให้องค์กรมีความมั่นคง และพนักงานมีความมั่นคง



รูปที่ 1.4 แผนผัง "ลูกค้าสำคัญอย่างไร"

1.3.3 รูปแบบการจัดผังองค์กร



รูปที่ 1.5 แผนผัง "รูปแบบการจัดผังองค์กร"

1.4 ตำแหน่งและหน้าที่งานที่ได้รับมอบหมาย

ตำแหน่ง : นักศึกษาฝึกงาน

หน้าที่ที่ได้รับมอบหมาย : ศึกษาการทำงานของ Modbus Protocol และศึกษาการเขียนโปรแกรมควบคุมคอนโทรลเลอร์ชนิด TMS320F2808

1.5 พนักงานที่ปรึกษาและตำแหน่งของพนักงานที่ปรึกษา

ชื่อ-สกุล : นายสุชาติ ศักดิ์ชัยกุล ตำแหน่ง ผู้จัดการทั่วไป

ชื่อ-สกุล : นายสุชาติ สุชาติ ตันติดำรง ตำแหน่ง ผู้จัดการฝ่ายวิศวกรรม

1.6 ระยะเวลาที่ปฏิบัติงาน

เริ่มตั้งแต่ 18 มิถุนายน 2555 จนถึง 6 ตุลาคม 2555
รวมเป็นระยะเวลาในการสหกิจศึกษาทั้งสิ้น 4 เดือน หรือ 16 สัปดาห์

1.7 วัตถุประสงค์ของโครงงาน

1.7.1 เพื่อศึกษารูปแบบการติดต่อสื่อสารข้อมูลที่ใช้ในอุตสาหกรรม

1.7.2 เพื่อศึกษาระบบการติดต่อสื่อสาร และการทำงานด้วยโปรโตคอล Modbus

1.7.3 เพื่อศึกษาการทำงานของ โปรโตคอล Modbusในการติดต่อสื่อสารบนอุปกรณ์อิเล็กทรอนิกส์

1.7.4 เพื่อศึกษาอุปกรณ์สำหรับการเชื่อมต่อระหว่าง Device และ Terminal บนมาตรฐานของการติดต่อสื่อสาร และควบคุมอุปกรณ์อิเล็กทรอนิกส์ผ่านการเขียนโปรแกรมคำสั่งบนคอนโทรลเลอร์ชนิด TMS320F2808 หรือ DSP2808 ด้วยโปรแกรม Code Composer Studio V. 3.3

1.8 ผลที่ได้รับจากโครงงาน

1.8.1 มีความรู้ความเข้าใจเรื่องของรูปแบบการสื่อสารข้อมูลในอุตสาหกรรม และสามารถนำไปประยุกต์ใช้ได้การเชื่อมต่อระหว่างอินเวอร์เตอร์ (Device) และอุปกรณ์อิเล็กทรอนิกส์ในงานอุตสาหกรรม (Terminal) บนมาตรฐานการติดต่อสื่อสาร และการทำงานด้วยโปรโตคอล Modbus

1.8.2 สามารถสั่งงานคอนโทรลเลอร์ชนิด TMS320F2808 หรือ DSP2808 ผ่านชั้นการประยุกต์ (Application layer) ของแบบจำลอง OSI (Open Systems Interconnection model) ได้

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

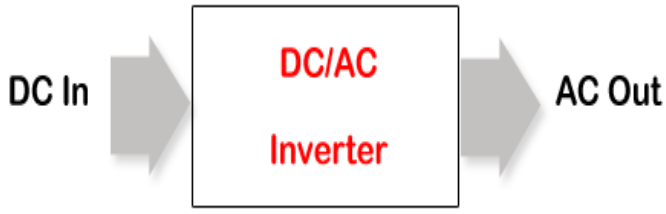
2.1 อินเวอร์เตอร์ (Inverter) คืออะไร

อินเวอร์เตอร์ (Inverter) คือ อุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการปรับเปลี่ยนความเร็วรอบของ 3-Phase Squirrel-Cage Induction Motor โดยวิธีการปรับแรงดันและความถี่ไฟฟ้าให้เหมาะสมกับมอเตอร์

2.1.1 หลักการทำงานของอินเวอร์เตอร์

อินเวอร์เตอร์ (Inverter) จะแปลงไฟกระแสสลับ (AC) จากแหล่งจ่ายไฟทั่วไปที่มีแรงดันและความถี่คงที่ ให้เป็นไฟกระแสตรง (DC) โดยวงจรคอนเวอร์เตอร์ (Converter Circuit) จากนั้นไฟกระแสตรงจะถูกแปลงเป็นไฟกระแสสลับที่สามารถปรับขนาดแรงดันและความถี่ได้โดยวงจรอินเวอร์เตอร์ (Inverter Circuit) วงจรทั้งสองนี้จะเป็นวงจรหลักที่ทำหน้าที่แปลงรูปคลื่น และผ่านพลังงานของอินเวอร์เตอร์

โดยทั่วไปแหล่งจ่ายไฟกระแสสลับมีรูปคลื่นซายน์ แต่เอาต์พุตของอินเวอร์เตอร์จะมีรูปคลื่นแตกต่างจากรูปซายน์ นอกจากนั้นยังมีชุดวงจรควบคุม (Control Circuit) ทำหน้าที่ควบคุมการทำงานของวงจรคอนเวอร์เตอร์และวงอินเวอร์เตอร์ให้เหมาะสมกับคุณสมบัติของ 3-phase Induction motor



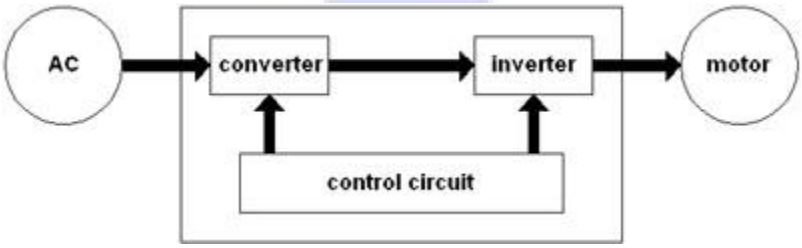
รูปที่ 2.1 หลักการทำงานของอินเวอร์เตอร์

2.1.2 โครงสร้างภายในของอินเวอร์เตอร์

2.1.2.1 ชุดคอนเวอร์เตอร์ (Converter Circuit) ซึ่งทำหน้าที่ แปลงไฟสลับจากแหล่งจ่ายไฟ AC. Power supply (50 Hz) ให้เป็นไฟตรง (DC Voltage)

2.1.2.2 ชุดอินเวอร์เตอร์ (Inverter Circuit) ซึ่งทำหน้าที่ แปลงไฟตรง (DC Voltage) ให้เป็นไฟสลับ (AC Voltage) ที่สามารถเปลี่ยนแปลงแรงดันและความถี่ได้

2.1.2.3 ชุดวงจรควบคุม (Control Circuit) ซึ่งทำหน้าที่ ควบคุมการทำงานของชุดคอนเวอร์เตอร์และชุดอินเวอร์เตอร์ ลักษณะการทำงานเปรียบเสมือนเครื่อง PLC



รูปที่ 2.2 กระบวนการทำงานของอินเวอร์เตอร์

2.2 การสื่อสารข้อมูลในอุตสาหกรรมไทย

การสื่อสารข้อมูลเกี่ยวข้องกับการถ่ายโอนข้อมูลข่าวสารระหว่างจุดหนึ่งไปยังอีกจุดหนึ่งในบทความนี้จะกล่าวถึงเนื้อหาเกี่ยวกับการสื่อสารข้อมูลแบบดิจิตอลซึ่งเป็นรูปแบบการสื่อสารหลักในปัจจุบันโดยข้อมูลที่รับส่งกันในบางครั้งอาจเรียกอีกอย่างว่าข่าวสาร (Information) ซึ่งถูกแทนด้วยแถวลำดับของค่าเท่ากับศูนย์และค่าเท่ากับหนึ่ง (Logic 0 1) ในการส่งค่าแบบดิจิตอล ซึ่งเป็นค่าที่สามารถจัดการได้ง่ายด้วยคอมพิวเตอร์หรือหน่วยประมวลผล (CPU) ส่วนการสื่อสารข้อมูลแบบอนาลอกนั้นสามารถนำไปใช้กับระบบต่าง ๆ ได้อีกหลายระบบ เช่น โทรศัพท์พื้นฐาน วิทยุโทรทัศน์ เป็นต้น

แต่ในงานวัดค่าสมัยใหม่เกือบทั้งหมดใช้การถ่ายโอนหรือส่งข้อมูลแบบดิจิตอล ระบบสื่อสารข้อมูลทุกระบบต้องมีตัวส่งข้อมูลหรือทรานสมิตเตอร์ (Transmitter) เพื่อส่งข้อมูลข่าวสาร และตัวรับ (Receiver) เพื่อตอบรับข้อมูลข่าวสารนั้นและจะต้องเชื่อมต่ออุปกรณ์ทั้ง 2 เข้าด้วยกัน ประเภทของสื่อสำหรับการเชื่อมต่ออาจจะใช้สายทองแดง สายใยแก้วนำแสง คลื่นวิทยุหรือคลื่นไมโครเวฟ และในการเชื่อมต่อระยะสั้น ๆ อาจใช้การเชื่อมต่อแบบขนาน (Parallel Connection) ซึ่งหมายความว่ามีสายสัญญาณหลาย ๆ เส้นในการนำพาข้อมูลหรือสัญญาณในเวลาเดียวกัน

บางครั้งข้อมูลแบบดิจิตอลถูกส่งบนระบบที่ออกแบบมาสำหรับสัญญาณอนาลอกซึ่งมีจุดประสงค์หลักคือการพยายามที่จะใช้โครงสร้างพื้นฐานเดิมที่ไม่ต้องลงทุนใหม่ ตัวอย่างก็คือโมเด็ม (Modem) ซึ่งทำงานโดยวิธีการผสมข้อมูลดิจิตอลกับสัญญาณพื้นฐานที่เป็นอนาลอก หรือการมอดูเลต (Modulate) ที่ส่งค่าไปบนสายโทรศัพท์ อีกด้านของสายโทรศัพท์จะมีโมเด็มอีกตัวหนึ่งทำการแยกสัญญาณหรือดีมอดูเลต (Demodulate) เพื่อให้ได้ข้อมูลดิจิตอลที่เหมือนข้อมูลที่ตัวส่งส่งมา คำว่า Modem ได้มาจาก Modulate and Demodulate อย่างไรก็ตามต้องมีการตกลงร่วมกันเกี่ยวกับการเข้ารหัสข้อมูลนั้น คือตัวรับต้องสามารถเข้าใจว่าตัวส่งส่งอะไรมา

โครงสร้างของข้อมูลหรือสัญญาณที่อุปกรณ์ตัวรับและตัวส่งใช้ในการส่งข้อมูลถูกเรียกว่าโปรโตคอล (Protocol) พูดง่าย ๆ ก็คือภาษาในการสื่อสารภาษาหนึ่งในอุปกรณ์สื่อสารก็ว่าได้ ในช่วง 10 ปีที่ผ่านมามีมาตรฐานและโปรโตคอลมากมายที่ถูกพัฒนาเพื่อให้เทคโนโลยีการสื่อสารสามารถนำมาประยุกต์ใช้ในงานอุตสาหกรรมได้อย่างมีประสิทธิภาพ ซึ่งผู้ออกแบบและผู้ใช้จะตระหนักถึงการประหยัดงบประมาณและค่าใช้จ่ายเป็นสำคัญ

การที่จะเพิ่มผลผลิตให้ได้มากที่สุดด้วยการรวบระบบหลาย ๆ ระบบให้เป็นระบบเดียวเพื่อให้ง่ายต่อการปฏิบัติงานและการวางแผนบำรุงรักษารวมทั้งการใช้งาน และลดจำนวนผู้ดูแลบำรุงรักษา ซึ่งโปรโตคอลคือโครงสร้างข้อมูลและระเบียบวิธีการที่ถูกใช้ในระบบสื่อสารข้อมูล ตัวอย่างเช่น คอมพิวเตอร์รับส่งข้อมูลกับเครื่องพิมพ์เอกสาร (Printer) เป็นต้น ในสมัยแรกเริ่มผู้พัฒนาซอฟต์แวร์และฮาร์ดแวร์ได้พัฒนาโปรโตคอลที่สามารถใช้ได้กับผลิตภัณฑ์ของตนเองเท่านั้นซึ่งเรียกอีกอย่างว่าระบบปิด

เพื่อที่จะเพิ่มความสามารถให้หลาย ๆ ระบบวัดและควบคุม สามารถทำงานร่วมกันได้ ลดการผูกขาดของผู้ผลิต ทำให้การพัฒนามาตรฐานของโปรโตคอลเป็นที่ต้องการของผู้ใช้ รวมทั้งผู้รับจ้างติดตั้งระบบงานหรืออินทีเกรเตอร์ (Integrator) โดยมาตรฐานอาจจะเริ่มมาจากการใช้อย่างแพร่หลายของโปรโตคอลของผู้ผลิตใดผู้ผลิตหนึ่งที่ไม่ได้ปกปิดข้อมูลของตนเอง (บางครั้งเรียกว่าดีแฟกโต้: De Facto) หรืออาจถูกพัฒนามาโดยเฉพาะจากกลุ่มนักพัฒนากลุ่มหนึ่งที่เป็นตัวแทนของกลุ่มอุตสาหกรรมใดอุตสาหกรรมหนึ่ง

ถ้ากล่าวถึงระบบเกี่ยวกับการสื่อสารข้อมูลในงานอุตสาหกรรม ก็สามารถกล่าวได้ว่ายังขาดมาตรฐานที่แข็งแกร่งในด้านการสื่อสารข้อมูลที่สามารถตอบสนองความต้องการได้ทุกประเภทงานอุตสาหกรรม แต่อย่างไรก็ตามก็ยังมีบางมาตรฐานที่โดดเด่นและใช้อย่างแพร่หลาย ตัวหนึ่งที่คาดว่ารู้จักกันดีก็คือ MODBUS ซึ่งเป็นมาตรฐานแบบดีแฟกโต้ ในระยะเวลา 20 ปีที่ผ่านมา MODBUS ได้ถูกนำมาใช้กับสื่อหรือมาตรฐานทางด้านฟิสิคอลแบบอนุกรม เช่น EIA-232 และ EIA-485 ได้อย่างแพร่หลายและกว้างขวาง

ในเรื่องเกี่ยวกับการสื่อสารข้อมูลมีประเด็นที่พูดได้ว่าน่าปวดหัวสำหรับผู้ที่ยังขาดประสบการณ์รวมทั้งอินทีเกรเตอร์มือใหม่นั้นคือโปรโตอลที่ใช้ในการสื่อสารระหว่างอุปกรณ์ต่างยี่ห้อกัน เช่น พีแอลซี เซนเซอร์ รวมแม้กระทั่งคอมพิวเตอร์โดยเฉพาะถ้าโปรโตคอลในแต่ละผู้ผลิตมีความเบี่ยงเบนอันเนื่องจากการตีความหมายของมาตรฐาน รวมทั้งตั้งใจที่จะทำให้แตกต่าง ซึ่งมักจะเป็นส่วนที่เรียกว่าฟังก์ชันเฉพาะของผลิตภัณฑ์ (Private Function) และนำฟังก์ชันมาใช้ในการติดตั้ง โดยโปรโตคอลที่มีความโดดเด่นในเรื่องการใช้งานร่วมกันของผลิตภัณฑ์นั้นมีอยู่ไม่มากนัก ซึ่งแต่ละตัวก็มีข้อดีข้อเสียหรือจุดแข็งจุดอ่อนต่างกัน เช่น PROFIBUS, Asi, DEVICENET หรือ DNP3 ที่ถูกใช้อย่างกว้างขวางในงานอุตสาหกรรมในแต่ละค่าย เช่น ค่ายยุโรป ค่ายอเมริกาเหนือและค่ายเอเชียที่มักอ้างอิงถึงผลิตภัณฑ์ของประเทศญี่ปุ่น

มาตรฐานที่ได้รับความสนใจอย่างมากในช่วงหลายปีที่ผ่านมานั้นคือ Ethernet ซึ่งในอดีตหรือยุคแรกเริ่มได้ถูกปฏิเสธในการนำมาใช้งาน เนื่องจาก Ethernet ไม่สามารถคาดการณ์ล่วงหน้าได้ หมายความว่าไม่สามารถรับประกันได้ว่าข้อมูลระดับวิกฤตที่สำคัญจะถูกส่งได้ภายในเวลาที่กำหนดหรือไม่ แต่ในปัจจุบันปัญหาเหล่านี้ได้ถูกแก้ไขและปรับปรุงโดยการพัฒนามาตรฐาน Ethernet และเทคโนโลยีสวิตชิ่งสมัยใหม่ อีกอย่างยังมีอีกโปรโตคอลที่ใช้ได้ดีมากกับ Ethernet และรู้จักกันดีนั้นคือ TCP/IP ซึ่งได้มาจากการพัฒนาอินเตอร์เน็ตและ TCP/IP ยังถือว่าเป็นโปรโตคอลที่นิยมอย่างกว้างขวางในทุกระบบงานมากที่สุด

2.2.1 ระบบวัดค่าและควบคุมสมัยใหม่ (Modern Instrumentation&Control System)

ในระบบวัดคุม ข้อมูลที่ต้องการคือข้อมูลที่มาจากอุปกรณ์วัดค่าหรือเซนเซอร์และค่านั้น ๆ จะถูกส่งไปยังอุปกรณ์ควบคุมหรือคอนโทรลเลอร์ (Controller) โดยส่วนใหญ่แล้วมักจะเป็นคอมพิวเตอร์หรือ PLC (Programmable Logic Controller) โดยอุปกรณ์ควบคุมจะส่งข้อมูลให้ชุดวงจรควบคุมเพื่อควบคุมอุปกรณ์หรือเครื่องจักรตามต้องการ

การรวบรวมหลาย ๆ ระบบเข้าด้วยกันโดยการใช้ความสามารถของระบบสื่อสารข้อมูลระหว่างแต่ละระบบที่มาจากผู้ผลิตแตกต่างกันในโรงงานจะสามารถลดจำนวนการลากสายคอนโทรลรวมทั้งจำนวนเทอร์มินอล (Cable Terminal) ซึ่งมีค่าติดตั้งราคาค่อนข้างสูงลงได้ ซึ่งความสามารถในการผลิตและคุณภาพของผลิตภัณฑ์คือเป้าหมายหลักในการบริหารจัดการที่ดีของแต่ละกระบวนการผลิตโดยการบริหารจัดการสามารถถูกปรับปรุงได้ด้วยข้อมูลกระบวนการผลิตที่แม่นยำที่สามารถหาได้ในระบบและภายในช่วงเวลาที่เหมาะสมกับความต้องการ ซึ่งสามารถพูดได้ว่าระบบวัดค่าและควบคุมที่ดีสามารถสนับสนุนคุณภาพและความสามารถในการผลิต ซึ่งเป้าหมายหลักของระบบวัดคุมในงานอุตสาหกรรมมีดังต่อไปนี้

1). ควบคุมกระบวนการผลิตและส่งสัญญาณเตือน (Control of Processes and Alarm)

โดยทั่วไปการควบคุมกระบวนการต้องวัดค่าที่มีผลต่อกระบวนการ เช่น อุณหภูมิ หรืออัตราการไหล ทำได้โดยใช้ตัววัดค่าแบบอนาลอกซึ่งทำงานโดยให้ค่าเอาต์พุตมาตรฐานที่ 4-20 mA มาตรฐาน 4-20 mA ได้ถูกนำไปใช้ในอุปกรณ์วัดค่าอย่างแพร่หลายในหลาย ๆ ผู้ผลิตนั้นหมายความว่าเราสามารถใช้อุปกรณ์วัดค่าแบบอนาลอกจากผู้ผลิตหลาย ๆ รายที่มีฟังก์ชันการทำงานเหมือนกันได้

ในปัจจุบันระบบวัดค่าแบบฟังก์ชันเดี่ยวหรือทำงานโดยไม่เกี่ยวข้องกับระบบอื่นได้ถูกแทนที่ระบบแบบรวมหลาย ๆ ระบบเข้าด้วยกัน เช่น ระบบ DCS (Distributed Control System)

2). การควบคุมลำดับการทำงาน อินเตอร์ล็อกกิ้ง และสัญญาณเตือน

โดยทั่วไประบบควบคุมเหล่านี้จะใช้รีเลย์ (Relay) ตัวตั้งเวลาหรือไทเมอร์ (Timer) และการลากสายควบคุมไปยังแผงควบคุม (Control Panel) แต่ในระบบขนาดใหญ่จะใช้ PLC เข้ามาช่วยการทำงานเนื่องจากสามารถลดจำนวนสายและอุปกรณ์ดังที่กล่าวมา อีกทั้ง PLC สามารถทำงานที่มีความซับซ้อนสูงได้ คิดค่าใช้จ่ายสำหรับงานที่มีความซับซ้อนสูง การใช้ PLC จะประหยัดค่าใช้จ่ายในการติดตั้งได้มาก

3). HMI (Human Machine Interface)

ในสมัยแรกกระบวนการผลิตในโรงงานจะถูกสั่งทำงานจากแผงควบคุมหน้าเครื่องจักรโดยผู้ปฏิบัติงานหลาย ๆ คนซึ่งแต่ละคนจะมีความรับผิดชอบในแต่ละส่วนงานของกระบวนการผลิตแตกต่างกันไป ระบบควบคุมสมัยใหม่มีแนวโน้มใช้ห้องควบคุมที่ศูนย์กลางสำหรับมอนิเตอร์โรงงานทั้งโรงงาน โดยห้องควบคุมจะถูกติดตั้งด้วยชุดคอมพิวเตอร์ที่รวบรวมข้อมูลจากเซนเซอร์ณ จุดต่าง ๆ และแสดงสถานะของกระบวนการทำงานโดยภาพกราฟฟิกที่เข้าใจง่าย รวมทั้งการมอนิเตอร์สัญญาณเตือน ควบคุมลำดับการผลิตแบบอัตโนมัติ และที่ขาดไม่ได้เช่นเดียวกันคือการทำการอินเตอร์ล็อก (Interlocking) เพื่อป้องกันอันตรายที่อาจเกิดขึ้นกับเครื่องจักรและผู้ปฏิบัติงาน

4). สารสนเทศเพื่อการจัดการ (Management Information)

ข้อมูลข่าวสารเพื่อการจัดการแบบสมัยแรกคือการอ่านค่าจากมิเตอร์ ตัวบันทึกหรือเร็คคอร์ดเดอร์ที่เป็นกราฟ ตัวนับและทรานสดิวเซอร์ และการสุ่มตัวอย่างผลิตภัณฑ์จากกระบวนการทำงาน ข้อมูลจะถูกใช้เพื่อมอนิเตอร์ตรวจสอบประสิทธิภาพการทำงานโดยรวมทั้งระบบของโรงงานเพื่อจะได้ข้อมูลที่ใช้ในการจัดการปรับปรุงกระบวนการทำงานและการผลิต ปัจจุบันการดึงข้อมูลได้ถูกผนึกรวมเข้ากับระบบควบคุมเพื่อลดวิธีการดึงข้อมูลที่ซ้ำซ้อน และเวลาการดึงข้อมูลที่เกี่ยวเนื่องกัน

2.2.1.1 ปัจจัยที่มีผลต่อการทำงานและการควบคุม

การลดเวลาหรือปัญหาการเกิดคอขวดของระบบหลายระบบ การบริหารจัดการที่ดีสามารถเข้าถึงแก่นของความสามารถในการผลิตและความสามารถของอุปกรณ์ที่ใช้ในการควบคุม ความสามารถหรือฟังก์ชันสามารถถูกเพิ่มเติมตามความต้องการโดยขึ้นอยู่กับเทคโนโลยีและประสิทธิภาพของส่วนประกอบอุปกรณ์ที่ใช้ เช่น IC, CPU และระบบสื่อสารข้อมูล สื่อ เครื่องมือ ตัวอย่างปัจจัยที่มีผลต่อการทำงานและการควบคุมโรงงานคือ

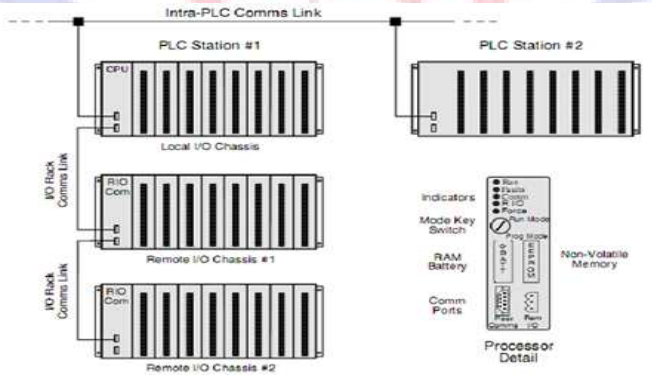
1). Distributed Control System (DCS)

DCS คือการควบคุมและการเข้าถึงข้อมูลกระบวนการผลิตโดยประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์แบบดิจิตอล การทำงานของ DCS อยู่บนพื้นฐานการสื่อสารข้อมูล และการออกแบบโมดูลเป็นส่วน ๆ หรือกระจายหน้าที่การทำงานแต่อยู่บนพื้นฐานสถาปัตยกรรมที่ทำงานร่วมกัน แต่ละโมดูลจะมีหน้าที่เฉพาะของมัน เช่น HMI การดึงค่าอนาลอกและดิจิตอล โดยปกติจะมีอุปกรณ์ที่ไว้สำหรับการเชื่อมต่ออุปกรณ์ควบคุมและเซนเซอร์เข้าด้วยกัน

2). Programmable Logic Controller (PLC)

PLC ได้ถูกพัฒนาประมาณปลายปี 1960 เพื่อแทนที่ชุดของรีเลย์โดยเฉพาะในอุตสาหกรรมการผลิตรถยนต์ PLC ถูกใช้ควบคุมลำดับการทำงานและอินเตอร์ล็อกกิ้งด้วยการควบคุมเปิด/ปิดวงจรหรือเรียกอีกอย่างว่า Digital Input/Output (DI/DO) ในตัว PLC จะมีหน่วยประมวลผลหรือ CPU ที่สามารถเขียนโปรแกรมสั่งทำงานด้วยภาษาค่อนข้างเข้าใจได้ง่ายเรียกว่าภาษาแลดเดอร์ (Ladder Logic)

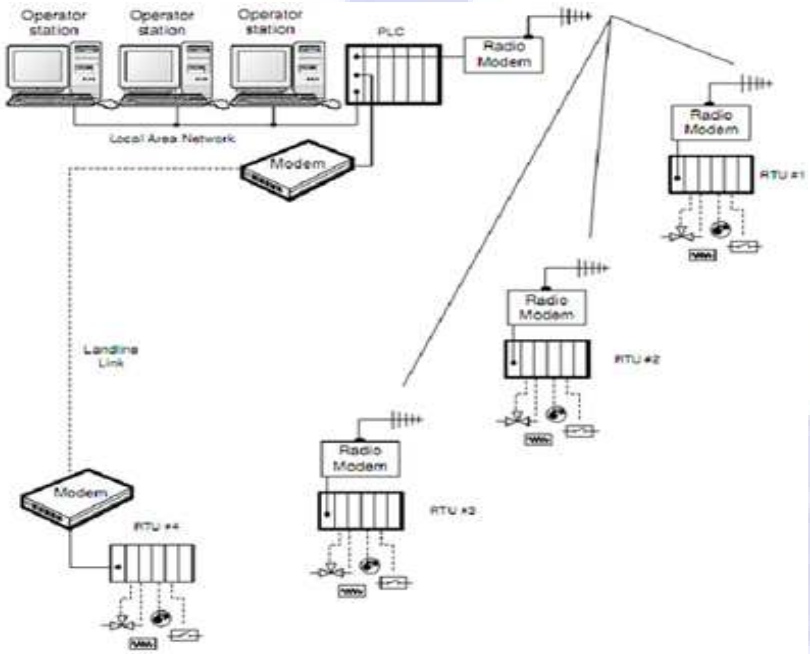
PLC สมัยใหม่จะมีชุด Analog Input (AI) ด้วย และสามารถรองรับการโปรแกรมที่ซับซ้อนและขนาดใหญ่คล้ายกับระบบ DCS เช่น PID Loop การเชื่อมต่อ PLC สามารถเชื่อมต่อด้วยความเร็วสูงในระดับ 10/100 mbps หรือโดยทางอินเตอร์เน็ตได้ ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 รูปแบบการเชื่อมต่อของ PLC

3). Supervisory Control and Data Acquisition (SCADA)

SCADA หมายถึงระบบที่ประกอบด้วย RTU (Remote Terminal Unit) จำนวนหลาย ๆ ตัว ที่เก็บข้อมูลจากภาคสนามและส่งข้อมูลกลับมายังมาสเตอร์ (Master) ที่ศูนย์กลางด้วยการสื่อสารข้อมูล SCADA สามารถควบคุมอุปกรณ์ระยะไกลผ่านชุดดิจิตอลเอาต์พุตของ RTU ดังรูปที่ 2.4 ซึ่งแสดงตัวอย่างของ SCADA



รูปที่ 2.4 รูปแบบการเชื่อมต่อของ SCADA

4). ระบบวัดค่าอัจฉริยะ (Smart Instrument Systems)

ในปี 1960 อุปกรณ์ที่วัดค่าใช้ค่า 4-20 mA ในการส่งค่าและเป็นที่นิยมเป็นอย่างมากหรือเรียกอีกอย่างว่าเป็นมาตรฐานดีแฟกโต้สำหรับเทคโนโลยีการวัดค่าในสมัยนั้นก็ว่าได้ ผลก็คือผู้ผลิตอุปกรณ์วัดค่าในสมัยนั้นใช้การสื่อสารโดยส่งค่าอนาลอก 4-20 mA ในผลิตภัณฑ์ของพวกเขา ทำให้ผู้ใช้มีตัวเลือกสำหรับเครื่องวัดและเซนเซอร์จากผู้ผลิตมากมายโดยไม่ต้องมีการแก้ไขปรับปรุงให้ทำงานเข้ากันได้

ด้วยการพัฒนาอย่างรวดเร็วของโปรเซสเซอร์และเทคโนโลยีทางดิจิตอล สถานการณ์หรือผลได้เปลี่ยนไปมาก ผู้ใช้เริ่มพึงพอใจกับประสิทธิภาพและประโยชน์เครื่องวัดค่าแบบดิจิตอลด้วยคุณสมบัติของดิจิตอลข้อมูลหรือค่าสามารถถูกแสดงโดยหน้าจอแสดงผลหลาย ๆ แห่งพร้อม

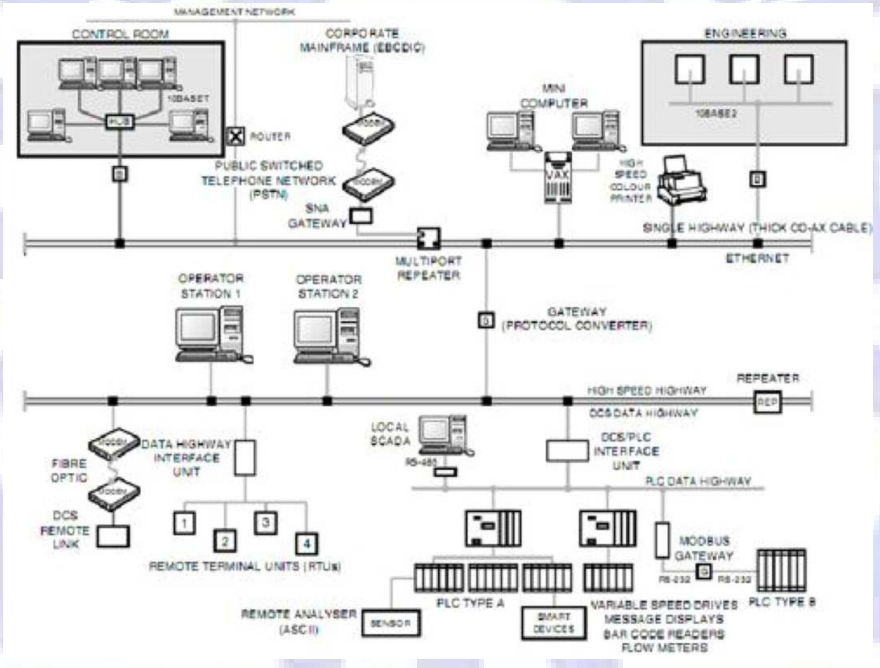
กัน มีความน่าเชื่อถือได้สูง ประหยัด สามารถปรับจูนตนเองได้โดยอัตโนมัติ และสามารถถูกตรวจสอบความผิดพลาดได้โดยง่าย

การเปลี่ยนแปลงมีแนวโน้มค่อย ๆ เปลี่ยนจากแบบอนาลอกไปสู่แบบดิจิตอลทั้งหมด ในปัจจุบันเซนเซอร์แบบดิจิตอลมีจำนวนมากที่มีความสามารถในการสื่อสารข้อมูลแบบดิจิตอล รวมทั้งเซนเซอร์สำหรับทำการวัดอุณหภูมิ ความดัน ระดับของเหลว การไหล มวลหรือน้ำหนัก ความเข้มข้น รวมทั้งวัดค่าทางไฟฟ้า เหล่านี้รู้จักในนามตัววัดค่าอัจฉริยะ (Smart Instrument)

<u>คุณสมบัติหลักในการนิยามตัววัดค่าอัจฉริยะคือ</u>

- เป็นเซนเซอร์ประมวลผลแบบดิจิตอล
- มีความสามารถในการส่งข้อมูลแบบดิจิตอล
- มีความสามารถในการต่อพ่วงกับอุปกรณ์ตัวอื่น

มีอุปกรณ์หลายรูปแบบที่ประกอบด้วยการประมวลการสื่อสารแบบดิจิตอลที่ถูกเรียกว่าแอกชูเอเตอร์ (Actuator) อัจฉริยะ ตัวอย่างเช่น ตัวขับความเร็วแบบปรับเปลี่ยนค่าได้ ซอฟต์สตาร์ทเตอร์ รีเลย์ป้องกัน ตัวควบคุมสวิตช์เกียร์แบบดิจิตอล



รูปที่ 2.5 รูปแบบระบบสื่อสารข้อมูลในงานอุตสาหกรรม

2.2.2 โมเดลของ OSI (OSI Model)

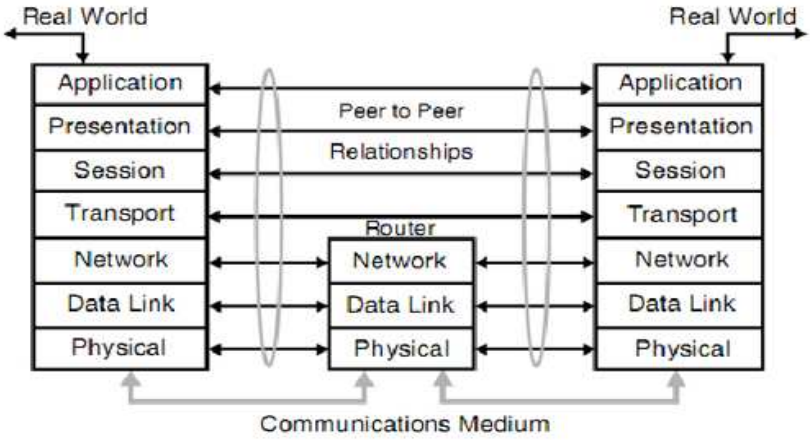
โมเดลของ OSI เป็นโมเดลที่ถูกอ้างอิงค่อนข้างมากซึ่งถูกพัฒนาโดยหน่วยงานสากลชื่อ International Organization for Standardization โดยได้รับการสนับสนุนจากผู้ผลิตในงานอุตสาหกรรมเป็นจำนวนมาก โมเดลหรือต้นแบบของ OSI จะลดขั้นตอนในการออกแบบระบบสื่อสารแทบทุกขั้นตอนรวมทั้งลดปัญหาในการสื่อสารในแต่ละชั้นหรือเลเยอร์ (Layer) ดังรูปที่ 2.6 มาตรฐานในการเชื่อมต่อทางฟิสิคอล เช่น EIA-232 จะถูกจัดลงในเลเยอร์ที่ 1

ขณะเดียวกันเลเยอร์อื่น ๆ มักจะเกี่ยวข้องกับการทำงานของซอฟต์แวร์ข้อความหรือข้อมูลจะถูกจัดลงในรูปของแพ็คเก็ต (Packet) ที่เป็นแถวลำดับข้อมูลในหน่วยไบต์ (Byte) โปรโตคอลจะกำหนดขนาดหรือความยาวของแพ็คเก็ตในแต่ละแพ็คเก็ตต้องการที่อยู่ต้นทางหรือแอดเดรสต้นทาง (Source Address) และที่อยู่ปลายทางหรือแอดเดรสปลายทาง (Destination Address) ที่ตัวส่งหรือระบบสื่อสารรู้ว่าที่ไหนที่ข้อมูลจะถูกส่งไปถึงและตัวรับจะสามารถรู้ว่าที่ไหนส่งข้อมูลมาให้

แพ็คเก็ตจะเริ่มถูกสร้างที่ชั้นบนสุดของโปรโตคอลปกติคือแอพพลิเคชันเลเยอร์ (Application Layer) และจะถูกส่งลงมาทีละลำดับชั้นผ่านการถูกประมวลผลโดยซอฟต์แวร์ในแต่ละชั้นจนกระทั่งมาถึงชั้นล่างสุดหรือฟิสิคอลเลเยอร์ (Physical Layer) และจะถูกส่งไปในสื่อสัญญาณหรือตัวกลางในช่วงเวลาที่แพ็คเก็ตลงมาทีละชั้นนั้นแพ็คเก็ตจะถูกเพิ่มข้อมูลส่วนหัวหรือเฮดเดอร์ (Header) เฮดเดอร์ที่เพิ่มในแต่ละชั้นจะบอกเลเยอร์อีกฝั่งหนึ่งว่าจะต้องทำอย่างไรกับแพ็คเก็ตที่มันได้รับโดยที่ฝั่งรับแพ็คเก็ตจะถูกส่งขึ้นทีละชั้นเช่นกันและส่วนของเฮดเดอร์จะถูกถอดออกและนำไปประมวลผลในแต่ละเลเยอร์

ดังตัวอย่างที่แอพพลิเคชันเลเยอร์จะได้รับข้อมูลส่วนหัวที่แอพลิเคชั่นเลเยอร์อีกฝั่งหนึ่งสร้างลูกศรระหว่างเลเยอร์ว่าแต่ละเลเยอร์อ่านแพ็คเก็ตที่มาจากเลเยอร์ลำดับเดียวกันอีกฝั่งหนึ่งซึ่งวิธีนี้อาจถูกเรียกว่าการสื่อสารแบบเพียร์ทูเพียร์ (Peer-to-Peer) ถึงแม้ตัวแพ็คเก็ตจริง ๆ แล้วจะถูกเคลื่อนย้ายผ่านสื่อทางกายภาพจริง ๆ เช่นสายทองแดงชั้นสแต็กของเลเยอร์ที่อยู่คั่นระหว่างกลางในรูปที่ 2.6 แสดงถึงเราเตอร์ที่ถูกใช้เพื่อแก้เงื่อนไขในการส่งข้อมูลระหว่าง 2 ฝั่งในกรณีพิเศษ เช่น กรณีอุปกรณ์ต่างระบบหรือต่างแพลตฟอร์ม (Platform) โมเดล OSI มีประโยชน์ในการกำหนดกรอบการออกแบบสำหรับระบบสื่อสารข้อมูลทุกระบบ

อย่างไรก็ตามมันไม่ได้นิยามถึงตัวโปรโตคอลจริง ๆ หรือภาษาสำหรับการสื่อสารข้อมูลที่ถูกใช้ในแต่ละเลเยอร์ผู้ใช้ยังคงคาดหวังว่ากลุ่มผู้ผลิตจะทำงานร่วมกันเพื่อนิยามมาตรฐานซอฟต์แวร์หรือฮาร์ดแวร์ให้เหมาะสมกับงานอุตสาหกรรมในแต่ละประเภทดังนั้นการออกแบบระบบสื่อสารข้อมูลต้องออกแบบให้สอดคล้องกับโมเดล OSI และใช้มันเป็นข้อมูลพื้นฐานในการออกแบบเพื่อให้มันสามารถทำงานเข้ากันกับอุปกรณ์ตัวอื่น ๆ ได้



รูปที่ 2.6 การสื่อสารของโมเดล OSI

2.2.3 มาตรฐาน

ในบทความตอนนี้จะอธิบายเกี่ยวกับมาตรฐานที่สำคัญที่ใช้ในงานอุตสาหกรรมซึ่งมีดังต่อไปนี้

2.2.3.1 มาตรฐาน RS-232

RS-232 หรืออีกชื่อ EIA-232 คือมาตรฐานการเชื่อมต่อที่ถูกพัฒนาขึ้นในสหรัฐอเมริกาในปี 1969 เพื่อนิยามรายละเอียดเกี่ยวกับสัญญาณไฟฟ้าและการเชื่อมต่อทางกายภาพระหว่างอุปกรณ์ต้นทางและปลายทางหรือ DTE (Data Terminal Equipment) และอุปกรณ์ในการสื่อสารข้อมูล (DCE: Data Communication Equipment) ที่ทำการเปลี่ยนข้อมูลให้เป็นข้อมูลไบนารีแบบอนุกรม การสื่อสารแบบอนุกรมระบบสื่อสารอาจจะประกอบด้วยองค์ประกอบดังนี้

- DTE อุปกรณ์ที่สร้างข้อมูลหรือประมวลผลข้อมูลและส่งข้อมูล เช่น คอมพิวเตอร์
- DCE คือตัวแปลงข้อมูล เช่น โมเด็มที่แปลงสัญญาณให้อยู่ในรูปที่เหมาะสมกับสื่อสัญญาณ เช่น สัญญาณอนาลอกสำหรับระบบโทรศัพท์
- สื่อสัญญาณ เช่น ระบบสายโทรศัพท์ และสายใยแก้วนำแสง
- ตัวรับที่เหมาะสมเช่น โมเด็ม หรือ DCE อีกด้านที่แปลงสัญญาณอนาลอกกลับไปยังระดับสัญญาณที่ตัวเทอร์มินอลรับปลายทางสามารถนำไปใช้งานได้
- เทอร์มินอลรับข้อมูล เช่น เครื่องพิมพ์ที่รับสัญญาณพัลส์ดิจิตอลและถอดรหัสกลับเป็นตัวอักษร

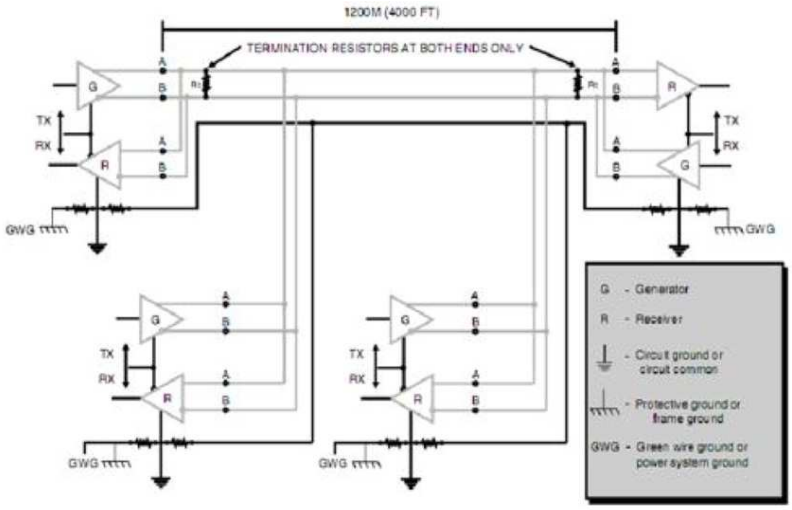
รูปที่ 2.7 รูปแบบการส่งข้อมูลโดย RS-232 ผ่านโมเด็มอนาลอก

มาตรฐาน EIA-232 อธิบายการเชื่อมต่อระหว่างเทอร์มินอล (DTE) และโมเด็ม (DCE) ที่ทำการส่งข้อมูลดิจิตอลแบบอนุกรม และมันยังเปิดช่องให้ผู้ออกแบบฮาร์ดแวร์และโปรโตคอลสามารถพัฒนาผลิตภัณฑ์ได้อย่างยืดหยุ่น ในระยะเวลาที่ผ่านมามาตรฐานได้ถูกดัดแปลงประยุกต์ใช้ในอุปกรณ์มากมายเช่น คอมพิวเตอร์ส่วนบุคคล เครื่องพิมพ์ PLC เครื่องมือวัด และอุปกรณ์อื่น ๆ

ปัจจุบันเวอร์ชันล่าสุดของ EIA-232 คือ EIA-232E ที่ได้ขยายเปลี่ยนนิยามของ DCE จาก "Data Communication Equipment" เป็นนิยามที่มีความหมายกว้างกว่าคือ "Data Circuit-Terminating Equipment" EIA-232 มีจุดอ่อนหลายจุดที่ไม่เหมาะในระบบสื่อสารสำหรับการควบคุมและวัดค่าในสภาพแวดล้อมอุตสาหกรรม ผลต่อเนื่องจากจุดอ่อนทำให้เกิดมาตรฐาน EIA อื่น ๆ ที่ถูกพัฒนาเพื่อแก้ข้อด้อยหรือข้อจำกัดเหล่านั้น มาตรฐานอื่น ๆ ที่ถูกพัฒนาเพื่อระบบวัดคุมที่ใช้อย่างแพร่หลายคือ EIA-423, EIA-422 และ EIA-485

2.2.3.2 มาตรฐาน EIA-485

EIA-485 เป็นระบบสื่อสารแบบสมดุลซึ่งใช้ระดับสัญญาณระดับเดียวกับมาตรฐาน EIA-422 แต่เพิ่มอัตราการส่งข้อมูล และจำนวนตัวรับส่งในสื่อเดียวกันสามารถมีถึง 32 ตัวตามที่มาตรฐานกำหนด มาตรฐาน EIA-485 ถูกใช้อย่างแพร่หลายในงานระบบวัดคุม ตัววัดค่าและตัวควบคุมสามารถถูกต่อเข้าด้วยกันบนเครือข่ายเดียวกัน รูประบบ RS-485 แบบง่าย ๆ แสดงในรูปที่ 2.8 EIA-485 สามารถใช้ระดับสัญญาณ TTL คือ 0-5 โวลต์ ในการส่งสัญญาณได้ การส่งข้อมูลจะเป็นแบบ Half Duplex ในกรณีต้องการส่งข้อมูลแบบ Full Duplex จะต้องใช้วงจร RS-485 2 ชุดเรียกอีกอย่างว่า RS-422



รูปที่ 2.8 รูปแบบการส่งข้อมูลโดย RS-485

2.2.3.3 สายใยแก้วนำแสง

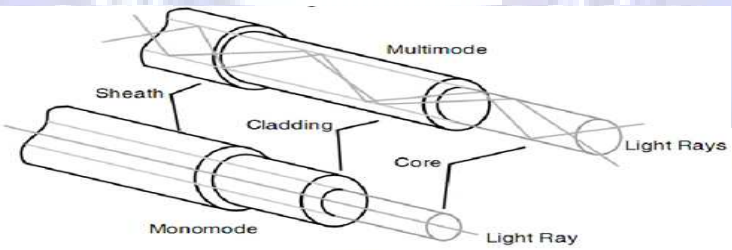
มีสายใยแก้วนำแสง 2 ประเภทหลัก ๆ ที่ใช้ในงานอุตสาหกรรมดังนี้

- ซิงเกิลโหมด หรือ โมโนโหมด (Single Mode/Mono Mode) โดยความยาวคลื่นแสงที่ใช้มักอยู่ที่ 1300 nm
- มัลติโหมด (Multimode) โดยความยาวคลื่นแสงที่ใช้มักอยู่ที่ 850 nm หรือ 820 nm

<u>เหตุผลหลักที่ใช้สายใยแก้วนำแสงในระบบสื่อสารข้อมูลในงานอุตสาหกรรมคือ</u>

- ทนต่อสัญญาณรบกวนทางไฟฟ้า
- แสงไม่ได้รับผลกระทบจากเสิร์จและทรานเซี้ยนต์

สรุปได้ว่าแนวโน้มการใช้สายใยแก้วนำแสงค่อนข้างโดดเด่นมากโดยเฉพาะในงานติดตั้งใหม่ที่ต้องการส่งข้อมูลความเร็วสูงและปริมาณมาก อีกอย่างประเภทหัวของสายใยแก้วที่ได้รับความนิยมในประเทศไทยคือหัว ST, SC และ FC



รูปที่ 2.9 รูปแบบการเคลื่อนที่ของแสงในสายใยแก้วนำแสง

2.2.3.4 MODBUS

MODBUS เป็นโปรโตคอลที่ถูกพัฒนาโดยบริษัท Modicon ประเทศสหรัฐอเมริกาซึ่งปัจจุบันได้เป็นส่วนหนึ่งของบริษัท Schneider Electric MODBUS ถูกออกแบบสำหรับใช้งานในระบบควบคุมการผลิตโดย PLC มาตรฐาน MODBUS นี้อ้างอิงกฎหรือขั้นตอนของชั้นดาต้าลิ้งค์เลเยอร์ (Data-Link Layer) และแอพพลิเคชันเลเยอร์ (Application Layer) ของโมเดล OSI เท่านั้น หมายความว่าขั้นตอนการทำงานในระดับชั้นฟิสิคอลเลเยอร์ (Physical Layer) สามารถใช้มาตรฐานใดก็ได้

MODBUS โปรโตคอลเป็นที่นิยมใช้อย่างกว้างขวางเพราะมีผลการสำรวจแล้วพบว่ามากกว่า 40% ของระบบสื่อสารข้อมูลในงานอุตสาหกรรมใช้ MODBUS ในหนึ่งเครือข่ายหรือระบบที่ใช้โปรโตคอล MODBUS สามารถมีจำนวนโหนดมากถึง 247 โหนดตามที่มาตรฐานกำหนดโดยใช้หลักการสื่อสารแบบ Master-Slave โดยมีโครงสร้างของเฟรมดังรูปที่ 2.10 ซึ่งในส่วนของ Address Field บ่งบอกถึงอุปกรณ์ที่กำลังถูกดึงข้อมูลในส่วนของ Function Field บ่งบอกถึงคำสั่งที่กำลังถูกสั่งให้ทำงาน ตัวอย่างเช่น การอ่านการส่งค่าอนาลอกและดิจิตอลในตัว Slave ในส่วนของ Data Field คือข้อมูลที่ถูกส่งจากอุปกรณ์ Slave ไปยัง Master หรือจาก Master ไปยัง Slave (กรณีคำสั่งเขียนข้อมูล) สุดท้ายคือส่วน Error Field ใช้เพื่อสร้างความมั่นใจว่าตัวรับสามารถตรวจสอบความถูกต้องของข้อมูลได้

รูปที่ 2.10 รูปแบบเฟรมของ MODBUS

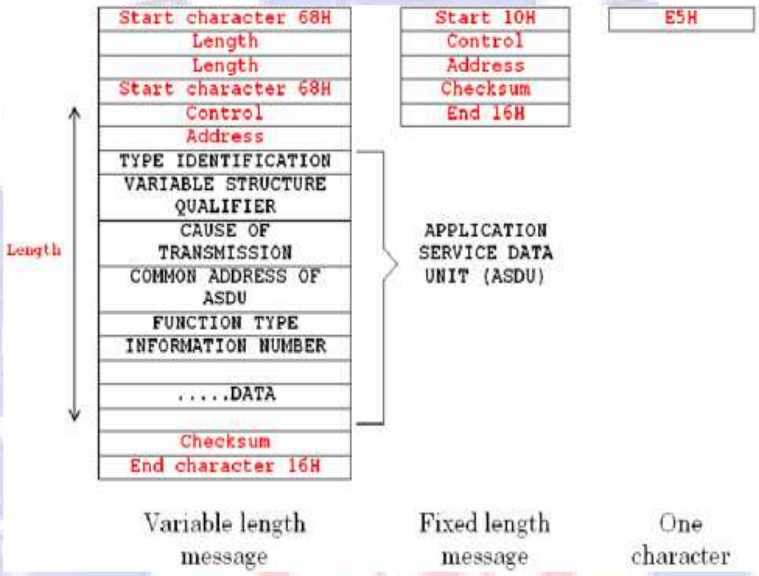
2.2.3.5 MODBUS PLUS

MODBUS PLUS ถูกสร้างโดยใช้เทคนิคการส่งข้อมูลแบบ Token Passing ถูกออกแบบให้ใช้งานบนอุปกรณ์ Modicon PLC เท่านั้น ส่งผลทำให้ขาดการเปิดของโปรโตคอลหรือพูดได้อีกอย่างว่า MODBUS PLUS คือโปรโตคอลปิด ซึ่งมีน้อยอุปกรณ์ที่สนับสนุน MODBUS PLUS

2.2.3.6 IEC60870-5-103

IEC60870-5-103 เดิมทีถูกพัฒนาโดยบริษัท SIEMENS บริษัทยักษ์ใหญ่ของเยอรมันในชื่อ VDEW และได้ถูกเสนอเข้ามาเป็นมาตรฐาน IEC ในชื่อ IEC870-5-103 และสุดท้ายในชื่อ IEC60870-5-103, IEC60870-5-103 เป็นที่นิยมในระบบสื่อสารข้อมูลของระบบป้องกันทางไฟฟ้า โดยเฉพาะในรีเลย์ป้องกัน IEC60870-5-103 ใช้ได้ดีเฉพาะระบบป้องกันเท่านั้นเพราะได้ถูกออกแบบมาเฉพาะงานดังกล่าว

จำนวนโหนดที่มีได้ในหนึ่งเครือข่ายคือ 254 โหนด แต่ที่ต่อได้จริงไม่ควรเกิน 20 โหนด เพราะจะทำให้ใช้เวลานานในการดึงทั้งหมด ประเภทเฟรมของ IEC60850-5-103 มี 3 ประเภทคือ เฟรมที่ความยาวเปลี่ยนแปลงตามขนาดข้อมูล เฟรมที่มีความยาวคงที่ส่วนใหญ่ใช้ในการทำแฮนด์เช็คกิ้ง และเฟรมขนาดหนึ่งไบต์ส่วนใหญ่ใช้สำหรับการทำการยืนยัน

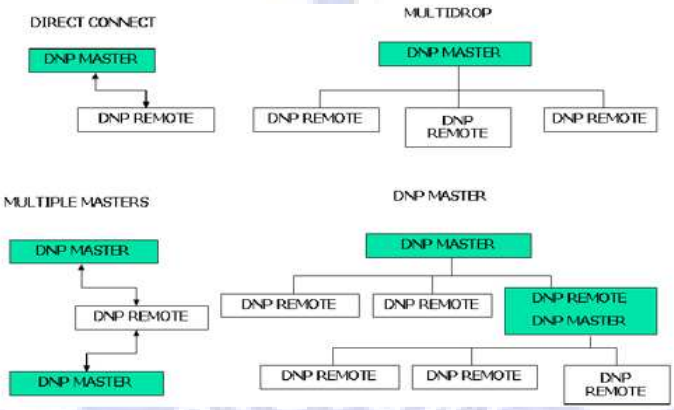


รูปที่ 2.11 รูปแบบเฟรมของ IEC60850-5-103

2.2.3.7 DNP3 (Distributed Network Protocol 3)

DNP3 ถูกพัฒนาโดยบริษัท GE-Harris โดยจุดประสงค์ที่สามารถสนับสนุนทุกประเภทงานในอุตสาหกรรม DNP3 เป็นโปรโตคอลที่มีความยืดหยุ่นค่อนข้างสูงและมี 3 ระดับตามขนาดและประสิทธิภาพของตัวอุปกรณ์เช่น ระดับหนึ่งสำหรับดิจิตอลมิเตอร์ ระดับสองสำหรับรีเลย์ป้องกัน และระดับสามสำหรับศูนย์สั่งการ DNP3 สามารถทำงานบน TCP/IP เรียกว่า DNP3 over TCP/IP

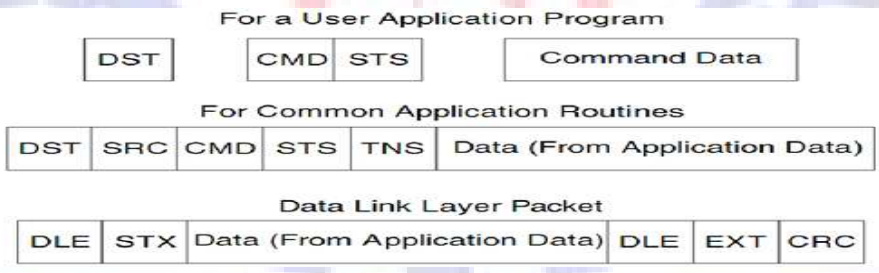
ข้อดีข้อหนึ่งของ DNP3 คือสามารถเปลี่ยนแปลงรูปแบบข้อมูลได้ในระหว่างการใช้งานและสนับสนุนงานที่ต้องการหลายมาสเตอร์ได้ DNP3 ยังสนับสนุนการทำงานแบบ Unsolicited Response หมายความว่าอุปกรณ์ประเภทสเลฟสามารถส่งข้อมูลได้เองโดยไม่ต้องรอคำถามจากมาสเตอร์



รูปที่ 2.12 รูปแบบการเชื่อมต่อของอุปกรณ์ที่ใช้ DNP3

2.2.3.8 Data High Plus /DH485

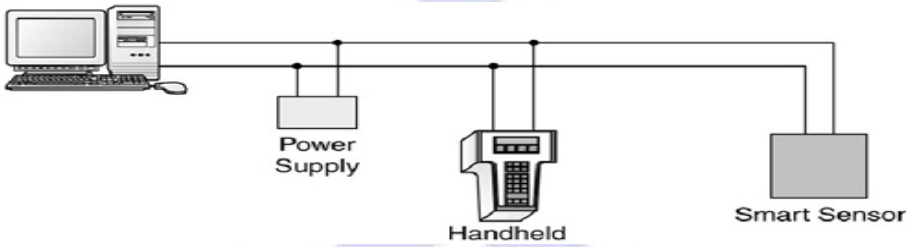
โปรโตคอลนี้ถูกสร้างเพื่อเป็นแบ็คโบนของอุปกรณ์ PLC ของ Allen Bradley (ปัจจุบันเป็นส่วนหนึ่งของ Rockwell Automation) Data High Plus เป็นโปรโตคอลที่ใช้ 3 เลเยอร์ของ OSI คือ ฟิสิคอล ดาต้าลิงค์ และแอพพลิเคชัน โดยมีโครงสร้างดังรูปที่ 2.13 มี 2 Address คือ DST และ SRC ใน Protocol Message ที่บอกที่อยู่ต้นทางและปลายทาง Data High Plus ใช้วิธีโทเคนพาสซิงที่แต่ละสถานีในเครือข่ายสามารถเป็นมาสเตอร์ในช่วงเวลาสั้น ๆ ตามที่กำหนดหรือตั้งค่าไว้



รูปที่ 2.13 รูปแบบโครงสร้างของโปรโตคอล Data High Plus

2.2.3.9 HART

โปรโตคอล HART (Highway Addressable Route Transducer) คือโปรโตคอลที่ใช้สำหรับอุปกรณ์วัดค่าโดยสามารถทำงานบนค่าอนาลอกที่ 4-20 mA ในรูปแบบการส่งค่าแบบดิจิตอล นั้นหมายความว่าเราสามารถใช้สายเส้นเดิมของทรานส์ดิวเซอร์แบบดั้งเดิมทำการส่งข้อมูลโดยโปรโตคอล HART



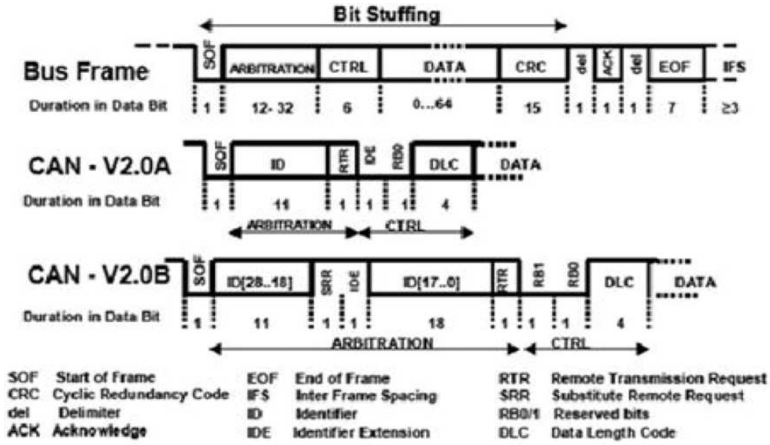
รูปที่ 2.14 รูปแบบการเชื่อมต่อของโปรโตคอล HART

2.2.3.10 ASi

ASi เป็นโปรโตคอลที่ใช้ในการควบคุมระบบงานที่ไม่ซับซ้อน โดยรูปแบบการสื่อสารที่โปรโตคอล ASi ใช้คือรูปแบบมาสเตอร์สสเลฟ ตัว ASi สามารถทำความเร็วสูงสุดได้ที่ 167 kbps หมายความว่าถ้ามี 31 โหนด โดยแต่ละโหนดมี 124 I/O จะใช้เวลา 5 ms ในการดึงข้อมูลที่ต้องการได้ทั้งหมด

2.2.3.11 CANBUS (Control Area Network)

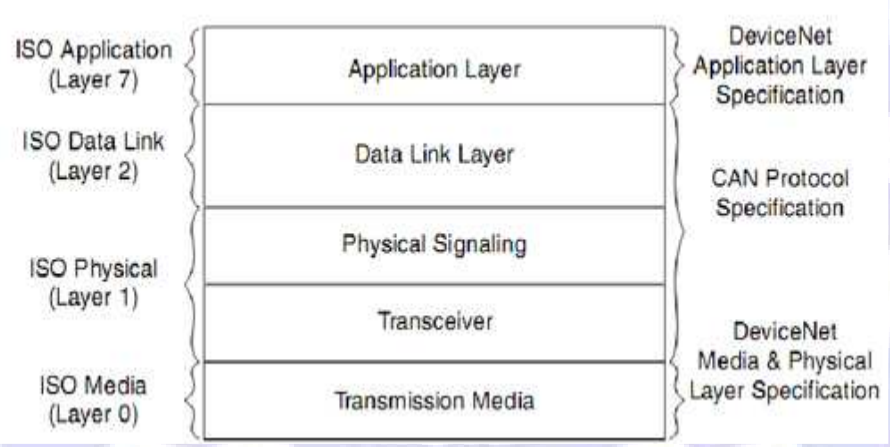
CANBUS ถูกพัฒนาโดยบริษัท Bosch ที่เรารู้จักกันดีในเรื่องเครื่องมือช่างที่คุณภาพดีจากประเทศเยอรมัน โดยจุดประสงค์แรกเริ่มของ CANBUS ใช้ในอุตสาหกรรมรถยนต์โดยเฉพาะระบบอิเล็กทรอนิกส์ภายในรถ เช่น แอร์ ประตู ระบบเบรก รวมทั้งกล่อง ECU (Electronic Control Unit) CANBUS ยังถูกนำมาใช้เป็น I/O Bus ใน RTU หลาย ๆ ยี่ห้อเช่น ABB หรือกระทั่งระบบอัตโนมัติจากประเทศจีนเช่น NARI



รูปที่ 2.15 รูปแบบของเฟรมโปรโตคอล CAN

2.2.3.12 DEVICENET

DEVICENET ถูกพัฒนาโดยบริษัท Allen-Bradley (ปัจจุบัน Rockwell) เป็นโปรโตคอลระดับล่างที่เน้นในการส่งข้อมูลประเภท DI/DO โดยที่ DEVICENET ใช้ 3 เลเยอร์ของโมเดล OSI ตัว DEVICENET สามารถสนับสนุนจำนวนโหนดตั้งแต่ 64 โหนดจนถึง 2048 โหนด และถูกออกแบบให้สามารถส่งไฟเลี้ยงไปยังอุปกรณ์วัดค่าได้ด้วย ซึ่ง OMROM PLC จากประเทศญี่ปุ่นที่เรารู้จักกันดีใช้ DEVICENET เป็นโปรโตคอลหลักในการสื่อสาร

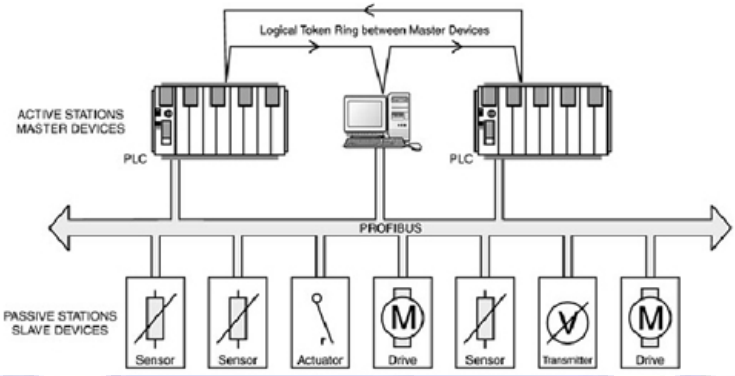


รูปที่ 2.16 รูปแบบการอ้างอิงโมเดล OSI ของโปรโตคอล DEVICENET

2.2.3.13 PROFIBUS

ถึงแม้จะถูกพัฒนาโดยองค์กร German Standard Association โดยมาตรฐานที่ทำงานจะเป็นแบบ EIA-485 เรียกว่า PROFIBUS DP และที่ทำงานบนมาตรฐาน IEC61158 เรียกว่า PROFIBUS PA ตัว PROFIBUS เป็นโปรโตคอลที่ได้รับความนิยมค่อนข้างมากโดยเฉพาะในงานอุตสาหกรรม

โดยมีรูปแบบการเชื่อมต่อดังรูปที่ 2.17 PROFIBUS ใช้วิธีการสื่อสารข้อมูลแบบผสมผสานระหว่างวิธีโทเคนพาสชิงและมาสเตอร์สเลฟเพื่อให้ได้ประสิทธิภาพในการสื่อสารดีที่สุด PROFIBUS ใช้ 3 เลเยอร์ตามมาตรฐาน OSI คือ ฟิสิคอล ดาต้าลิงค์ และแอพพลิเคชัน และมีการเพิ่มเลเยอร์ที่ 8 เรียกว่า ยูสเซอร์เลเยอร์ (User Layer) สำหรับอุปกรณ์ที่ใช้ PROFIBUS เป็นโปรโตคอลหลักคือ PLC



รูปที่ 2.17 รูปแบบการเชื่อมต่อของโปรโตคอล PROFIBUS

2.2.3.14 Foundation Field bus

Foundation Fieldbus ปัจจุบันอาจจะถือว่าเป็นมาตรฐานใหม่ล่าสุดที่ใช้ในการเชื่อมต่อระหว่างอุปกรณ์วัดค่า และ PLC รวมไปถึง DCS ตัว Foundation Fieldbus ใช้ 3 เลเยอร์ของโมเดล OSI และมีใช้ 2 เวอร์ชันอ้างอิงตามความเร็วคือความเร็วทั่วไปในชื่อ H1 และเวอร์ชันความเร็วสูงเรียกว่า HSE

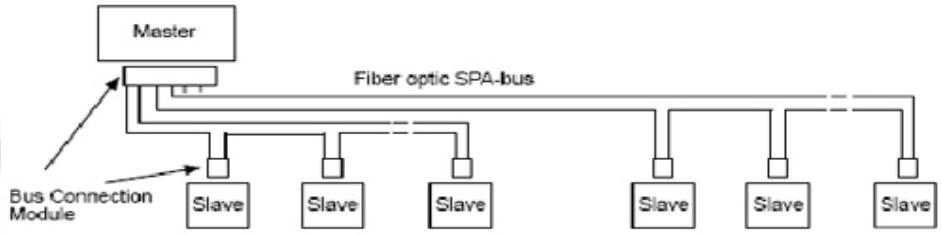
Foundation Fieldbus อ้างอิงใช้งานเลเยอร์คล้ายกับ PROFIBUS โดยประกอบด้วยเลเยอร์ ฟิสิคอล ดาต้าลิงค์ แอพพลิเคชัน และยูสเซอร์ที่ใช้นิยามฟังก์ชันบล็อก

2.2.3.15 IEC60850-5-101

IEC60850-5-101 ถูกออกแบบสำหรับการสื่อสารระหว่าง SCADA และ RTU โดยเฉพาะระบบไฟฟ้า โครงสร้างระดับดาต้าลิงค์มีความคล้ายคลึงกับ IEC60850-5-103 ตัว IEC60850-5-101 สามารถทำงานบน TCP/IP เรียกอีกอย่างว่า IEC60850-5-104 IEC60850-5-101 มีข้อด้อยอย่างหนึ่งคือไม่สามารถมีจำนวนมาสเตอร์หลายมาสเตอร์ได้เนื่องจากไม่มีฟิลด์สำหรับระบุมาสเตอร์ อย่างไรก็ตาม IEC60850-5-101 นิยมใช้ในระบบ SCADA โดยเฉพาะในผลิตภัณฑ์จากประเทศยุโรป

2.2.3.16 SPABUS

SPABUS ถูกพัฒนาโดยบริษัท ABB ในประเทศฟินแลนด์ โดยออกแบบมาเพื่อใช้ในระบบควบคุมและป้องกันในอุตสาหกรรมการผลิตและจำหน่ายไฟฟ้า SPABUS เป็นโปรโตคอลที่ไม่ซับซ้อนทำงานโดยใช้อักขระ ASCII ทำให้เราสามารถอ่านแพ็คเก็ตบนเครื่องมือพื้นฐานได้โดยง่าย เช่นโปรแกรมไฮเปอร์เทอร์มินอล (Hyperterminal) อุปกรณ์ที่ใช้ SPABUS มักจะเป็นรีเลย์ป้องกันโดยรีเลย์ของ ABB ในตระกูล SPA



รูปที่ 2.18 รูปแบบการเชื่อมต่อของโปรโตคอล SPABUS โดยใช้สายใยแก้วนำแสง

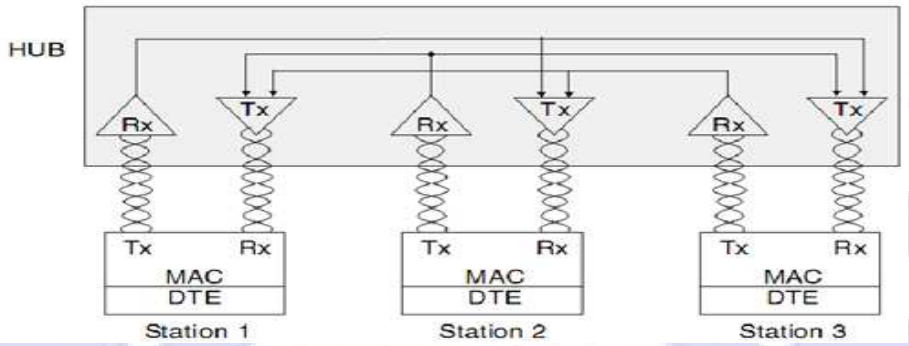
2.2.3.17 LONBUS

LONBUS ถูกพัฒนาโดยบริษัท Echelon มีชื่อเดิมว่า LonTalk โดยได้ถูกใช้อย่างแพร่หลายในระบบอาคารอัตโนมัติ แต่อย่างไรก็ตาม LONBUS ได้ถูกนำมาประยุกต์ใช้ในอีกหลายระบบงานเช่น ระบบรถไฟฟ้า ระบบจำหน่ายไฟฟ้า จนได้รับเสนอเป็นมาตรฐานในชื่อ ANSI/CEA-709.1-B ตัว LONBUS ยังถูกสนับสนุนโดยซอฟต์แวร์ประเภทบำรุงรักษา เช่น IFS ของสวีเดนนั้นหมายความว่าซอฟต์แวร์สามารถดึงค่าจากอุปกรณ์หรือเครื่องจักรเพื่อวางแผนบำรุงรักษาได้โดยไม่ต้องใช้คนป้อนข้อมูล

2.2.3.18 Industrial Ethernet

Industrial Ethernet ได้ถูกพัฒนาเร็วมากและได้ก้าวข้ามปัญหาเกี่ยวกับความเชื่อถือได้ที่ต่ำไม่เพียงพอต่องานระบบอุตสาหกรรม หนึ่งในเหตุผลสำหรับความสำเร็จของ Industrial Ethernet คือการใช้งานที่ง่ายและราคาติดตั้งที่ไม่สูงเนื่องจากมีผลิตภัณฑ์ในตลาดเป็นจำนวนมาก ในยุคเริ่มแรก Ethernet ใช้แค่เพียงวิธีการที่เรียกว่า CSMA/CD (Carrier Sense Multiple Access/Collision Detect) เท่านั้น ซึ่งมีข้อด้อยในเรื่องการไม่สามารถคาดการณ์เวลาที่จะได้รับข้อมูล จึงไม่เหมาะสมกับงานควบคุมกระบวนการผลิต

แต่ในปัจจุบัน Ethernet มีความเร็วในการส่งข้อมูลสูงมากกว่า 100 mbp และยังทำงานแบบ Full Duplex ตามมาตรฐาน IEEE 802.3 ด้วยความสามารถของ Ethernet Switch ปัจจุบันสามารถจัดระดับความสำคัญของแพ็คเก็ตส่งผลให้ระบบ Ethernet สามารถคาดการณ์การได้รับข้อมูลแม่นยำดีขึ้นเป็นอย่างมาก Ethernet Switch ยังจัดการและติดตั้งได้ง่ายกว่าวิธีการแบบโทเคนพาสซึ่งเป็นอย่างมาก โดยรูปที่ 2.19 แสดงตัวอย่างทั่วไปของ 100 BASE TX



รูปที่ 2.19 รูปแบบการเชื่อมต่อของโปรโตคอล Ethernet

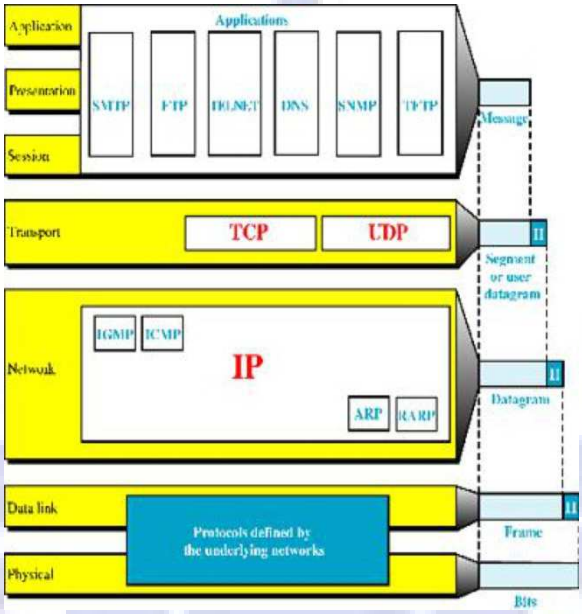
2.2.3.19 TCP/IP (Transmission Control Protocol/Internet Protocol)

แพร่หลายเนื่องจากเติบโตพร้อมกับระบบอินเตอร์เน็ต TCP/IP ยังทำงานได้ดีมากกับ Ethernet จริงจริงแล้ว TCP/IP สามารถถูกจัดได้ 3 เลเยอร์ดังนี้

- Process/Application Layer (เทียบเท่ากับ 3 เลเยอร์บนสุดของ OSI)
- Service/Host-to-Host Layer (เทียบเคียงกับ Transport ของ OSI)
- Internetwork Layer (เทียบเคียงกับ Network ของ OSI)

และที่จริงแล้ว TCP/IP คือชุดของโปรโตคอลหลาย ๆ ตัวที่มีคุณสมบัติเฉพาะตัวแตกต่างกันไปเช่น HTTP สำหรับการแสดงผลบนเว็บ FTP สำหรับการถ่ายโอนไฟล์ เป็นต้น TCP/IP ถือว่าเป็น

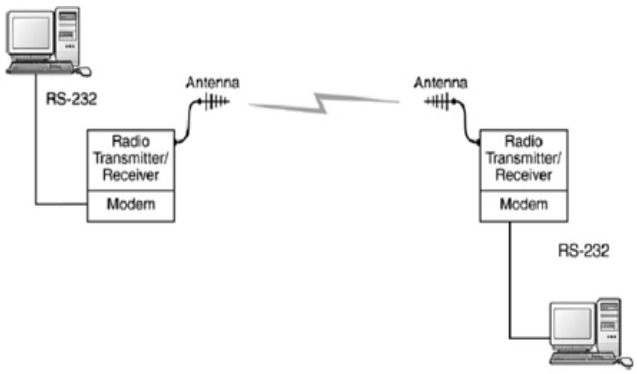
โปรโตคอลที่มีค่าใช้จ่ายค่อนข้างต่ำและถูกสนับสนุนอย่างแพร่หลายในทุกระบบงาน อาจจะกล่าวได้ว่าการเกิดของ TCP/IP ทำให้หลาย ๆ โปรโตคอลได้ตายหรือหายไปจากระบบสื่อสารก็ว่าได้ ปัจจุบันโปรโตคอล TCP/I ได้มีความสำคัญต่อชีวิตประจำวันของมนุษย์มากไม่ว่าจะเป็นอินเตอร์เน็ต โทรศัพท์ คอมพิวเตอร์ซึ่งเป็นอุปกรณ์หลักในการให้บริการเราเช่น ระบบธนาคาร



รูปที่ 2.20 ชุดโปรโตคอล TCP/IP เทียบเคียงกับโมเดล OSI

2.2.3.20 Radio (Wireless) Communication

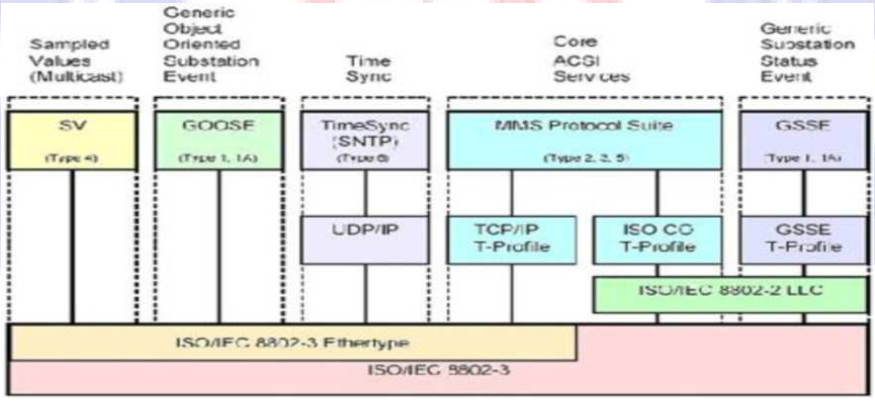
การใช้ Radio ในงานอุตสาหกรรมเกี่ยวข้องกับการใช้โมเด็มในการส่งข้อมูล ดังแสดงในตัวอย่างรูปที่ 2.21 MODBUS สามารถถูกส่งบนดาต้าลิงค์ของโมเด็มไร้สายได้ การมีมาตรฐานใหม่ของไวร์เลส (Wireless) เช่น IEEE 802.11b หรือ IEEE 802.11a และ IEEE 802.15 หรือ Bluetooth ทำให้มีความเชื่อถือในระบบไร้สายสูงขึ้นและราคาติดตั้งค่อนข้างต่ำ และในปัจจุบันเริ่มมีหลายโรงงานอุตสาหกรรมหันมาเริ่มใช้ไวร์เลสในกระบวนการผลิตเนื่องจากประหยัดต้นทุนค่าสายสัญญาณและการติดตั้ง



รูปที่ 2.21 รูปแบบการใช้สัญญาณวิทยุในการส่งข้อมูล

2.2.3.21 IEC61850

IEC61850 ถือว่าเป็นโปรโตคอลตัวล่าสุดที่ออกแบบมาใช้ในระบบสถานีไฟฟ้าโดยตัวตั้งตัวดีคือสามบริษัทยักษ์ใหญ่ทางด้านระบบไฟฟ้าในยุโรปนั้นคือ Siemens, ABB และ AREVA จุดมุ่งหมายคือมาแทนที่โปรโตคอลทุกตัวไม่ว่าจะเป็นระบบเวลาจริง (Real-time System) หรือแบบไคลเอนท์เซิร์ฟเวอร์ (Client-Server) การโอนหรือบันทึกค่าการติดตั้งจะใช้เทคโนโลยีทาง XML (Extension Mark-up Language) ในชื่อของภาษา SCL (Substation Configuration Language) นั้นหมายความว่าถ้าแต่ละผู้ผลิตทำตามมาตรฐาน IEC61850 ไฟล์คอนฟิกกูเรชันและพารามิเตอร์ก็สามารถใช้งานร่วมกันได้ รูปแบบการสื่อสาร IEC61850 จะใช้ Ethernet อย่างเดียวสำหรับการส่งข้อมูลแบบวิกฤตเช่นระบบป้องกัน เรียกว่า GOOSE (Generic Object Oriented System Event) ในส่วนระบบควบคุมจะใช้ IEC61850 วิ่งบน TCP/IP เรียกว่า MMS (Manufacturing Message Specification)



รูปที่ 2.22 รูปแบบโครงสร้างของ IEC61850

2.2.4 โปรโตคอล

ดังที่กล่าวมาแล้วว่าโมเดล OSI ได้กำหนดกรอบการออกแบบของแต่ละโปรโตคอลให้ปฏิบัติตาม ในทางกลับกันโปรโตคอลนิยามรูปแบบเฟรมฟอร์แมตที่อาจจะประกอบส่วนต่าง ๆ ดังนี้

ไบต์แรกสามารถเป็นชุดของหนึ่งและศูนย์สลับกันเพื่อการเข้าจังหวะกับตัวรับหรือเป็นแฟลก (Flag) หรือตัวบ่งบอกว่าจุดเริ่มต้นของเฟรมอยู่ที่ไหน (ใช้สำหรับตัวรับ) ไบต์ที่สองอาจจะบรรจุที่อยู่ปลายทางที่บอกว่าข้อมูลจะส่งไปที่ไหน ไบต์ที่สามบรรจุที่อยู่ต้นทางเพื่อบอกว่าข้อมูลมาจากไหน และหลาย ๆ ไบต์ที่อยู่ตรงกลางเป็นข้อมูลจริงที่ต้องถูกส่งจากตัวส่งไปยังตัวรับ และไบต์สุดท้าย คือตัวชี้บ่งบอกว่าเป็นจุดจบของเฟรมหรือเป็นรหัสสำหรับไว้ตรวจสอบความผิดพลาด เช่น CRC, LRC, หรือ Checksum โปรโตคอลมีหลากหลายจากง่าย ๆ (เป็นแบบ ASCII) หรือเป็นแบบไบนารีที่ซับซ้อนมาก (เช่น TCP/IP) ที่ทำงานที่ความเร็วในการส่งระดับ mbps อีกอย่างเราสามารถพูดได้ว่าไม่มีโปรโตคอลใดถูกหรือผิด แต่ขึ้นอยู่กับจุดประสงค์และความเหมาะสมในการนำไปใช้งานนั้น ๆ

รูปที่ 2.23 รูปแบบเฟรมทั่วไปของโปรโตคอล

2.3 โปรโตคอล Modbus

2.3.1 ข้อมูลทั่วไปของ Modbus Protocol

Modbus Protocol เป็นโปรโตคอลที่อยู่ในชั้น Application Layer ของ OSI Model ซึ่งถูกออกแบบและพัฒนาโดย Modicon Inc. (ปัจจุบันเป็นบริษัท Schneider Electric) ในปี 1997 โดย Modbus Protocol เป็นโปรโตคอลที่ถูกออกแบบมาเพื่อการติดต่อสื่อสารข้อมูล Input/Output และ Register ภายใน PLC ทำหน้าที่ในการรับและส่งข้อมูลบนระบบเครือข่าย และเนื่องจาก Modbus Protocol เป็นโปรโตคอลระบบเปิด มีการเชื่อมต่อและพัฒนาได้ง่ายและไม่มีค่าใช้จ่าย (Royalty-Free) ด้วยเหตุนี้ส่งผลให้ Modbus Protocol ได้รับความนิยมอย่างแพร่หลายในวงการอุตสาหกรรม และการนำไปใช้งานบนระบบ RTU (Remote terminal Unit), RTU I/O, Digital Power Meter, Flow computer เป็นต้น นอกจากนี้ยังรองรับและใช้งานร่วมกับแอพพลิเคชันจำพวก SCADA และ HMI Software ได้อีกด้วย

Modbus Protocol ในช่วงแรกได้ถูกออกแบบมาเพื่อใช้งานกับ Serial Port ซึ่งเป็นการเชื่อมต่อโดยตรงระหว่าง device และ terminal โดย Serial Port จะประกอบไปด้วย RS232, RS422 และ RS485 จนการใช้งานเป็นไปอย่างแพร่หลาย ทำให้มีการพัฒนาให้อุปกรณ์สามารถติดต่อสื่อสารกับอุปกรณ์ที่อยู่บนเครือข่าย Ethernet โดยพัฒนาไปเป็น Modbus Plus และ Modbus TCP/IP อย่างไรก็ตาม Modbus จำเป็นจะต้องมีอุปกรณ์จำพวก Gateway หรือ Bridge เพื่อใช้ในการเชื่อมต่อระหว่าง Serial line กับ Ethernet ทั้งนี้ทั้งนั้น Modbus Protocol ที่ทำงานบน Serial Port ยังคงเป็นหลักพื้นฐานและได้รับความนิยมสูงสุด

การทำงานของ Modbus Protocol นั้นในขณะที่มีการเชื่อมต่อสื่อสารอยู่บนเครือข่าย โปรโตคอลจะเป็นตัวกำหนดว่าทำอย่างไรให้คอนโทรลเลอร์สามารถทราบถึง Device Address และจำที่อยู่ของ Message ที่ส่งถึง Device ดังกล่าวได้ อีกทั้งยังสามารถกำหนดการกระทำและสกัดกั้นข้อมูลต่าง ๆ ที่ต้องการบรรจุไว้ภายใน Message ได้อีกด้วย

บนเครือข่ายข้อความที่เป็นส่วนหนึ่งในการสื่อสารแบบ Modbus Protocol จะถูกฝังไว้ภายในเฟรมหรือแพคเกตที่ใช้งานอยู่บนเครือข่ายเพื่อให้สามารถรับส่งข้อมูลระหว่างโหนดต่าง ๆ ของ Device บนการสื่อสารโดยการใช้ Modbus Protocol ได้

จากข้อมูลข้างต้นสามารถสรุปได้ว่า Modbus Protocol เป็นโปรโตคอลเพื่อใช้ในการติดต่อสื่อสารทั้งการเชื่อมต่อโดยตรงภายใน PLC และระบบเครือข่าย ซึ่งใช้ในการรับและการส่งข้อมูลระหว่าง Device และ Terminal ผ่านทาง Serial Port หรือ Gateway และ Bridge

2.3.2 วัตถุประสงค์ในการใช้ Modbus Protocol

Modbus Protocol เป็น Open Protocol ซึ่งก็คือโปรโตคอลที่สามารถให้บุคคลทั่วไปใช้งานได้ อีกทั้งยังสามารถพัฒนาอุปกรณ์ที่ใช้การสื่อสารแบบ Modbus ได้โดยไม่ต้องเสียค่าใช้จ่ายใด ๆ จึงทำให้ Modbus Protocol เป็นโปรโตคอลพื้นฐานและได้รับความนิยมอย่างแพร่หลายในทางอุตสาหกรรมโดย Modbus Protocol จะใช้การรับและการส่งข้อมูลผ่านการควบคุมของคอนโทรลเลอร์หรืออุปกรณ์ที่ใช้ในการประมวลผลต่าง ๆ

2.3.3 ลักษณะการทำงานของ Modbus Protocol

Modbus Protocol เป็นการสื่อสารโดยการส่งข้อมูลไปตามสายสัญญานที่เชื่อมต่อระหว่างอุปกรณ์ โดยวิธีการที่ง่ายที่สุดคือการต่อสายสัญญานระหว่าง Master หนึ่งตัวและ Slave อีกหนึ่งตัว

ข้อมูลจะถูกส่งต่อเนื่องกันด้วยสัญญาน 0 หรือ 1 ซึ่งเราเรียกกันว่าบิต โดยแต่ละบิตจะอยู่ในรูปแบบของแรงดัน (Voltage) โดยค่า 0 จะแทนด้วยแรงดันด้านบวกและค่า 1 จะแทนด้วยแรงดันด้านลบ ทำให้บิตถูกส่งออกไปด้วยความเร็ว ซึ่งความเร็วที่ใช้งานโดยทั่วไปบน Modbus Protocol คือ 9600 bps. (bit per sec)

2.3.4 ลักษณะการติดต่อสื่อสารบน Modbus Protocol

ลักษณะการติดต่อสื่อสารบน Modbus Protocol ใช้รูปแบบของ Master/Slave ในการรับและการส่งข้อมูลต่าง ๆ

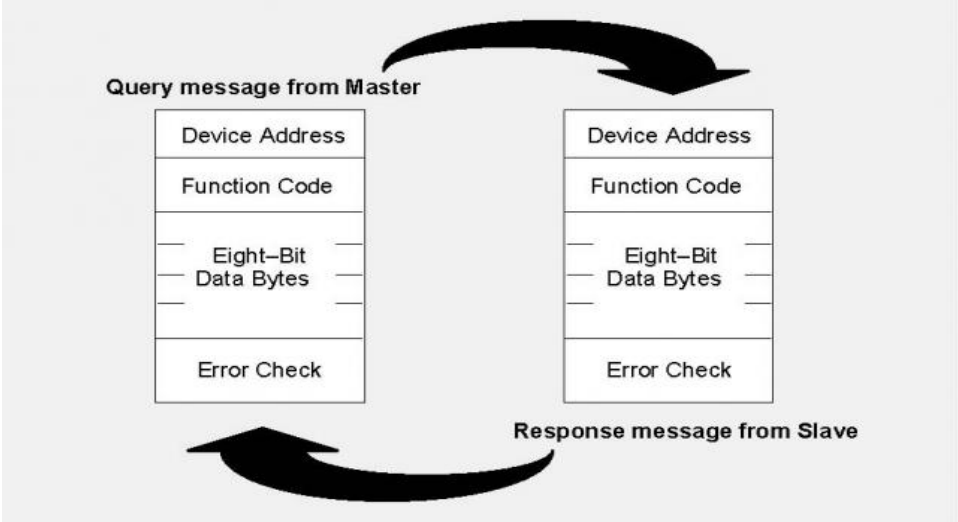
2.3.4.1 Master/Slave

Master/Slave คือการสื่อสารจากอุปกรณ์แม่ (Master) ที่มีเพียงเครื่องเดียว ส่วนใหญ่มักเป็น Software หรืออุปกรณ์ HMI โดยจะทำการส่งข้อมูลไปยังอุปกรณ์ลูก (Slave) ได้ครั้งละหลาย ๆ เครื่อง แต่จะกำหนดให้มี Slave ได้สูงสุด 255 เครื่องเท่านั้น

Master สามารถสร้างที่อยู่เฉพาะให้กับ Slave หรือสามารถกระจายข้อมูลไปยัง Slave ทุก ๆ ตัวได้ และ Slave สามารถส่งข้อความตอบกลับมายัง Master ได้เช่นกัน โดยการทำงานดังกล่าวจะถูกควบคุมโดยคอนโทรลเลอร์ ซึ่งเป็นตัวกำหนดการกระทำต่าง ๆ ให้แก่ Master และ Slave

2.3.5 รูปแบบการส่งข้อมูลภายใน Modbus Protocol

Modbus Protocol มีรูปแบบในการส่งเป็นแบบ Query-Response Cycle คือการส่งคำร้องจาก Master ไปยัง Slave นั้น ข้อมูลทั้งหมดจะถูกบรรจุไว้ในส่วนของ Query Message และการส่งข้อความตอบกลับจาก Slave เพื่อแจ้งให้ Master ทราบว่า Slave ได้รับการร้องขอดังกล่าวแล้วนั้นจะถูกบรรจุไว้ในส่วนของ Response Message



รูปที่ 2.24 วงจรของ Query-Response

2.3.5.1 Query Message from Master

Function Code ภายใน Query จะเป็นตัวบอกไปยังที่อยู่ของ Slave ให้ทราบถึงการกระทำที่ Master ร้องขอการดำเนินการไปยัง Slave และ Data Byte จะบรรจุข้อมูลเริ่มต้นที่ต้องการดำเนินการบน Slave ตัวอย่างเช่น Function Code 03 จะสอบถามไปยัง Slave ให้อ่านหน่วยความจำที่อยู่ใน Holing Register และตอบกลับมายัง Master เกี่ยวกับเนื้อหานั้น ในส่วนของ Data Field จะบรรจุข้อมูลที่บอกให้ Slave ทราบถึงตำแหน่งเริ่มต้นและจำนวนของรีจิสเตอร์ที่ต้องใช้ในการอ่านข้อมูลนั้นว่ามีเท่าไหร่ และ Error Checking Field จะคอยตรวจสอบความผิดพลาด เพื่อให้การทำงานของ Slave เป็นไปอย่างสมบูรณ์

2.3.5.2 Response Message from Device

ถ้า Slave ต้องการที่จะตอบกลับมายัง Master นั้น Function Code ภายใน Response จะเป็นการสะท้อนกลับมาของ Function Code บน Query ซึ่ง Data Byte จะบรรจุการเลือก Register และ

สถานะของข้อมูลใน Slave และหากเกิดความผิดพลาด Function Code จะถูกดัดแปลงเพื่อบ่งบอกถึงความผิดพลาดของข้อมูลที่เกิดขึ้นซึ่งก็คือ ErrorResponse รวมไปถึงการบรรจุรหัสสำหรับอธิบายถึงความผิดพลาดดังกล่าว จากนั้น Error Checing Field จึงจะยอมให้ Master ยืนยันเนื้อหาของข้อมูลที่ถูกต้องสมบูรณ์

2.3.6 The two serial transmittion Mode

คอนโทรลเลอร์สามารถตั้งค่าการสื่อสารบนมาตรฐานของ Modbus Networks โดยใช้โหมดการส่งผ่านสองรูปแบบคือโหมด ASCII และโหมด RTU ซึ่งผู้ใช้จะเลือกใช้โหมดตาม Serial Port Communication Parameters (baud rate, parity, etc.) ในระหว่างการกำหนดของคอนโทรลเลอร์ ซึ่ง Mode และ Serial Parameter จะต้องเหมือนกันทุก ๆ Device บนเครือข่าย Modbus

การเลือกโหมด ASCII หรือ RTU บนมาตรฐานของ Modbus Network เป็นการนิยามถึงบิตของ Message Field ในลำดับการส่งบนเครือข่าย และจะกำหนดข้อมูลที่จะจัดเก็บลงใน Message Field ว่าควรเป็นอย่างไรแล้วทำการถอดรหัสออกมา

บนเครือข่ายต่าง ๆ เช่น MAP และ Modbus Plus นั้น Modbus Message จะถูกวางไว้ในเฟรมที่ไม่มีความสัมพันธ์กับลำดับการส่งผ่านข้อมูล ตัวอย่างเช่นการอ่านการร้องขอที่อยู่ในตำแหน่ง Holding Register จะสามารถจัดการคอนโทรลเลอร์สองตัวบน ModbusPlus ที่นอกเหนือจากการพิจารณาการตั้งค่าปัจจุบันของทั้ง Controller's serial Modbus port ได้

2.3.6.1 RTU Mode

เมื่อคอนโทรลเลอร์ตั้งค่าการสื่อสารบน Modbus Network โดยใช้โหมด RTU (Remote Terminal Unit) ซึ่งมีลักษณะของข้อมูลในการส่งเป็น 8-bit byte ที่บรรจุเลขฐานสอง ในลักษณะข้อมูลแบบ 4-bit จำนวน 2 byte ประโยชน์หลักของโหมดนี้คือมันเป็นโหมดที่ดีที่สุดที่ยอมให้ส่งตัวอักษรได้จำนวนมาก และเป็นโหมดที่ Data Throughput ดีกว่าโหมด ASCII สำหรับช่วง Baud Rate ที่เหมือนกัน

รูปแบบของไบต์ข้อมูลในโหมด RTU

Coding System: ตัวเลขฐานสอง 8-bit

Bits per Byte: 1 Start bit

8 Data bit บิตที่สำคัญน้อยสุดจะถูกส่งก่อน

1 Sit สำหรับ Even/Odd Parity และไม่มีบิตสำหรับ No Narity

1 Stop bit หากมีการใช้ Parity และเป็น 2 stop bit หากเป็น No Parity

Error Check Field: Cyclical Redundancy Check (CRC)

2.3.6.2 ASCII Mode

เมื่อคอนโทรลเลอร์ตั้งค่าการสื่อสารบน Modbus Network โดยใช้โหมด ASCII (American Standard Codes for Infomation Interchange) ซึ่งมีลักษณะของข้อมูลในการส่งเป็น 8-bit byte และจะส่งเป็นรหัสแอสกีจำนวน 2 ตัว ประโยชน์หลักของโหมดนี้คือมันจะยอมให้ช่วงเวลาของการเพิ่มขึ้นที่ 1 วินาทีเกิดขึ้นระหว่างตัวอักษรที่อยู่นอกเหนือจากการเกิดความผิดพลาด

รูปแบบของไบต์ข้อมูลในโหมด ASCII

Coding System: ตัวเลขฐาน 16และตัวอักษรแอสกี 0-9, A-F

ตัวเลขฐาน 16 หนึ่งตัวจะบรรจุใน ASCII Character ของข้อความ

Bits per Byte: 1 start bit

7 Data bit บิตที่สำคัญน้อยสุดจะถูกส่งก่อน

1 Bit สำหรับ Even/Odd Parity และไม่มีบิตสำหรับ No Parity

1 Stop bit หากมีการใช้ Parity และเป็น 2 stop bit หากเป็น No Parity

Error Check Field: Longitudinal Redundancy Check (LRC)

2.3.7 Modbus Message Framing

Modbus Message Framing คือโหมดในการรับส่งข้อมูลของ Modbus Protocol ซึ่งController สามารถส่งผ่านข้อมูลบน Standard Modbus Network ที่เป็นแบบ Serial Line ได้ 2 โหมดคือ โหมด ASCII และ โหมด RTU ซึ่งผู้ใช้สามารถตั้งค่าของ Parameter ต่าง ๆ ของ Serial Port (Baud Rate, Parity Mode, etc.) ได้ ซึ่งค่า Serial Parameter ของอุปกรณ์ทุกตัวบน Modbus Network จะมีค่า Parameter เหมือนกันทุกตัวแม้ว่าจะอยู่ในโหมดใดก็ตาม

2.3.7.1 Modbus RTU Framing

โหมด RTU นั้นเป็นการส่งข้อมูลในลักษณะของเลขฐานสอง (Binary) ซึ่งใช้หลักการของ Remote Terminal Unit บิตของข้อมูลที่ถูกส่งแบบ RTU จะถูกส่งออกไปในรูปของ 8-bit binary

เฟรมข้อมูลในโหมด RTU ประกอบด้วยข้อมูลแสดงตำแหล่งแอดเดรสของ slave 1ไบต์ หมายเลขฟังก์ชัน 1 ไบต์ ข้อมูลที่ทำการส่งมากสุดไม่เกิน 252 ไบต์ และรหัสตรวจสอบความถูกต้องของข้อมูลแบบ CRC (Cyclical Redundancy Checking) ขนาด 2 ไบต์

ค่า CRCที่กล่าวข้างต้นนั้นเป็นค่าที่ได้จากการคำนวณทุก ๆ ไบต์ ซึ่งไม่รวม Start bit, Stop bit และ Parity Check โดยที่ Slave ตัวที่ส่งข้อมูลออกมาจะสร้างรหัส CRC แล้วส่งรหัสดังกล่าวตามท้ายไบต์ของข้อมูลออกมา หลังจากที่ Master ได้รับเฟรมข้อมูล และถอดข้อมูลอออกจากเฟรมแล้ว Master จะทำการคำนวณค่า CRC สูตรเดียวกับ Slave เพื่อทำการเปรียบเทียบค่า CRC ทั้ง 2 ค่าว่าตรงกันหรือไม่หากไม่ตรงกันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูล

ในโหมด RTU การส่งข้อมูล 1 ไบต์ ไม่ว่าจะเป็นข้อมูลส่วนใดภายในเฟรม จะต้องทำการส่งข้อมูลรวม 11 บิต คือ Start bit 1 บิต Data bit 8 บิต (1 byte) Parity check 1 บิต และ Stop bit 1 บิต หรือหากเลือกแบบไม่มี Parity bit ก็สามารถตั้งค่าเป็นแบบ Stop bit แทน 2 บิตก็ได้ และสำหรับการเลือกให้มี Parity bit นั้น สามารถเลือกเป็นแบบคู่ (Even Parity) หรือแบบคี่ (Odd Parity) ก็ได้ หากต้องการออกแบบให้สอดคล้องกับอุปกรณ์ทั่วไปได้มากที่สุด ควรเลือกออกแบบเป็นแบบคู่ที่สามารถปรับเปลี่ยนเป็นแบบคี่ได้หรือสามารถปรับเปลี่ยนไปเป็นไม่มีการตรวจสอบ Parity (No Parity) ได้ด้วย

ใน RTU Mode เฟรมข้อมูลที่จะทำการส่งข้อมูลออกไปภายในบัสจะต้องรอระยะเวลาว่างของข้อมูลอย่างน้อย 3.5 Character จึงจะส่งเฟรมข้อมูลถัดไปได้ และภายในเฟรมแต่ละเฟรมประกอบไปด้วยชุดบิตข้อมูลจำนวนหลาย ๆ ชุดก็จะอยู่ห่างกันไม่เกิน 1.5 bit ซึ่งวัตถุประสงค์ในการกำหนดช่วงเวลาว่างระหว่างเฟรมข้อมูลและชุดบิตข้อมูลภายในเฟรม ก็เพื่อให้อุปกรณ์ต่าง ๆ ไม่ว่าจะเป็น Master หรือ Slave สามารถรับรู้ถึงจุดเริ่มต้นและจุดสิ้นสุดของเฟรมข้อมูลแต่ละเฟรมและสามารถตรวจสอบได้ว่าการรับส่งข้อมูลขณะนั้น เกิดความผิดพลาดหรือไม่ โดยตรวจสอบกับช่วงระยะห่างของเวลาที่ควรจะเป็นกับค่าที่วัดได้จริง

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4

รูปที่ 2.25 เฟรมการส่งข้อมูลแบบ RTU

ข้อดีของการส่งข้อมูลแบบ RTU คือสามารถส่งข้อมูลที่มีขนาดใหญ่ได้ดีกว่า เมื่อมีอัตราส่ง (Bauad Rate) ที่เท่ากัน และแต่ละ Message จะมีการไหลของข้อมูลแบบต่อเนื่อง

2.3.7.2 Modbus ASCII Framing

โหมด ASCII นั้นเป็นการส่งข้อมูลในลักษณะของเลขฐาน 16 (Hexadecimal) ซึ่งใช้หลักการของการส่งข้อมูลแบบ ASCII (American Standard Code for Information Interchange) บิตของข้อมูลที่ถูกส่งแบบ ASCII ข้อมูลขนาด 1 ไบต์นั้น จะถูกมองเป็นตัวอักษร 2 ตัว ซึ่งเป็นเลขฐาน16 ตัวอย่างเช่น 0x5B ก็จะถูกมองเป็นตัวอักษร "5" และตัวอักษร "B"จากนั้นก็นำค่าดังกล่าวไปค้นหารหัส ASCII ของตัวอักษรทั้งสองตัว จะได้เป็น 0x35 สำหรับ "5" และ 0x42 สำหรับ "B" แล้วทำการส่งรหัส ASCII สองค่านี้ออกไป ซึ่งผลที่ได้จะเท่ากับการส่งค่า 0x5B เป็นข้อมูลขขนาด 1 ไบต์ในโหมด RTU

จากการส่งข้อมูลในโหมด ASCII เฟรมข้อมูลที่จะทำการส่งจะเริ่มต้นด้วยเครื่องหมาย ":" (Colon) และตามด้วย Character ที่เป็นอักษรASCII (ASCII 0x3A) ปิดท้ายเฟรมด้วย "CRLF" (carriage return-line feed) และรหัส ASCII อีกสองตัว (ASCII 0x0D and 0x0A) ซึ่งหมายถึงจุดสิ้นสุดของเฟรมข้อมูลนั้น ๆ และในส่วนอื่น ๆ ของเฟรมข้อมูลจะถูกส่งไปเป็นข้อมูลเลขฐาน 16 (0-9,A-F) โดยขณะที่บัสข้อมูลว่างจากการรับส่งข้อมูล อุปกรณ์ทุกตัวจะคอยตรวจสอบข้อมูลในบัสว่ามีการส่งรหัส ASCII ของ ":" ออกมาหรือไม่ถ้ามีก็จะรับรู้ว่าขณะนี้ได้มีการเริ่มต้นส่งเฟรมข้อมูลออกมาแล้ว และจะเข้าสู่กระบวนการรับข้อมูลต่อไป

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

รูปที่ 2.26 เฟรมการส่งข้อมูลแบบ ASCII

การรับส่งข้อมูลในโหมด ASCII นั้น เป็นการส่งข้อมูลในรูปแบบรหัส ASCII ของตัวอักษร ซึ่งสามารถกำหนดได้ด้วยบิตข้อมูลจำนวน 7 บิต ไม่ต้องใช้ถึง 8 บิต ดังนั้น บิตที่จำเป็นต้องใช้ในการส่งรหัส ASCII 1 ตัว จะต้องประกอบด้วย Start bit 1 บิต Data bit ของรหัส ASCII 7 บิต Parity check 1 บิต และ Stop bit 1 บิต รวมทั้งหมด 10 บิต และเช่นเดียวกับโหมด RTU ที่สามารถเลือกประเภทของ Parity Check ได้ว่าจะให้เป็นแบบคู่หรือแบบคี่ หรือไม่มี Parity Check ซึ่งจะต้องเปลี่ยนบิตดังกล่าวเป็น Stop bit แทน

ถึงแม้โหมด ASCII จะไม่ต้องกำหนดช่วงระยะห่างทางเวลาของเฟรมข้อมูลแต่ละเฟรม แต่อุปกรณ์ยังต้องสามารถตรวจจับช่วงระยะห่างทางเวลาในขณะที่มีการส่งข้อมูลรหัส ASCII แต่ละ

ตัวได้ ซึ่งหากเว้นช่วงนานเกินไป แสดงว่าเกิดความผิดพลาดในการสื่อสาร โดยปกติจะกำหนดค่าเวลานี้ไว้ที่ 1 วินาที เรียกเวลานี้ว่า Time Out Period หากเปรียบเทียบระหว่างเฟรมข้อมูลในโหมด ASCII กับโหมด RTU จะพบว่าการส่งข้อมูลในโหมด ASCII นั้นหากต้องการส่งไบต์ข้อมูลให้เท่ากับโหมด RTU จะต้องส่งข้อมูลรหัส ASCII ออกไปเป็นจำนวนสองเท่าของจำนวนไบต์ข้อมูลในโหมด RTU ซึ่งเฟรมข้อมูล 1 เฟรมสามารถส่งข้อมูลได้มากสุด 252 ไบต์ หากเป็นโหมดASCII จะต้องส่งข้อมูลตัวอักษรออกไปทั้งหมด 2x252 ซึ่งเท่ากับ 504 ตัวอักษร และเพื่อเป็นมาตรฐานให้เฟรมข้อมูลของทั้งสองโหมดมีขนาดเท่ากัน จึงกำหนดให้ค่า 504 เป็นค่าจำนวนตัวอักษรที่มากที่สุดในการส่งเฟรมข้อมูลด้วยโหมด ASCII

ส่วนข้อดีของการส่งข้อมูลแบบ ASCII คือ การส่งข้อมูลในโหมด ASCII นั้นเป็นไปอย่างรวดเร็ว โดยที่มี Error เกิดขึ้นน้อยมาก

2.3.8 How Character are Transmitted Serial (ลำดับการส่งตัวอักษรเป็นอย่างไร)

เมื่อ Message ถูกส่งไปบน Standard Modbus Serial Network ตัวอักษรต่าง ๆ หรือไบต์ข้อมูลจะถูกส่ง (ขยับจากซ้ายไปขวา) ไปในคำสั่งด้วย Least Significant Bit (LSB) ไปเรื่อย ๆ จนถึง Most Significant Bit (MSB)

2.3.8.1 RTU character framing มีลำดับของบิตข้อมูลดังนี้

With Parity Checking

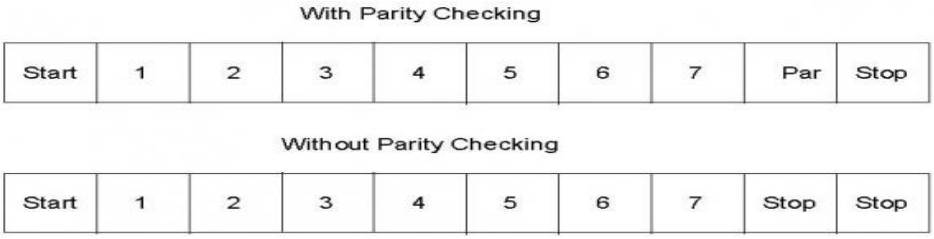
Start	1	2	3	4	5	6	7	8	Par	Stop

Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

รูปที่ 2.27 ลำดับบิตข้อมูล RTU

2.3.8.2 ASCII character framing มีลำดับของบิตข้อมูลดังนี้



รูปที่ 2.28 ลำดับบิตข้อมูล ASCII)

2.3.9 Field of Message Frame

2.3.9.1 Description about Field of Message Freme

1). Address Field

Address ของ Slave ที่สามารถกำหนดได้จริงจะอยู่ในช่วง 0-247(ฐานสิบ) แต่ Slave แต่ละตัวจะถูกกำหนดที่อยู่ภายในช่วง 1-247 เนื่องจากตำแหน่งที่ 0 นั้น จะถูกใช้สำหรับBroadcast Query ซึ่ง Master สามารถระบุที่อยู่ของ Slave ที่ต้องการสื่อสารได้ โดยการใส่ที่อยู่ของ Slave ตัวนั้น ๆ ไปใน Address Field ของข้อความ

เมื่อ Slave ส่งResponse กลับมา มันจะทำการใส่ที่อยู่ของตัวมันเองลงใน Address Field ของ Response เพื่อให้ Master รู้ว่า Response ที่ได้รับนั้นเป็นของ Slave ตัวใด

2). Function Field

Function Field Code สามารถกำหนดได้ในช่วง 1-255 (ฐานสิบ) ซึ่งเป็นตัวที่บอกให้ Slave ทราบว่าคำสั่งที่ส่งมานั้นเป็นชนิดใดและต้องการให้ทำอะไร เมื่อ Slave ทำการตอบสนองต่อคำสั่งจาก Master แล้ว มันจะใช้ Function Code Field เป็นตัวบอกสถานะการทำงานว่าเป็นปกติหรือไม่ โดยที่หากการทำงานเป็นปกติมันจะส่ง Function Code ตัวเดิมกลับมา แต่ถ้าเกิดความผิดพลาดขึ้นมันจะส่ง Function Code ตัวเดิมกลับมาพร้อมกับ Most-Significant Bit ที่ถูกตั้งค่าให้เป็นลอจิก 1 กลับมา ซึ่งจะทำให้ Master ทราบว่าเกิดความผิดพลาดขึ้น

3). Data Field

Data Field คือข้อมูลที่จำเป็นสำหรับ Slave ซึ่งต้องใช้ในการทำงานตามคำสั่งของ Master ที่กำหนดมาใน Function Code ซึ่ง Data Field นี้อาจมีความยาวมากหรือไม่มีเลยก็ได้

2.3.9.2 How the Address Field in Handle (จัดการกับ Address Field ได้อย่างไร)

Address Field ของเฟรมข้อมูลจะบรรจุตัวอักษร 2 ตัว (ASCII) หรือตัวอักษร 8 บิต (RTU) และที่อยู่ของ Slave Device ที่สมบูรณ์จะอยู่ในตำแหน่งที่ 0-247 ซึ่งเป็น Slave Addressเฉพาะที่อยู่ในตำแหน่งที่ 1-247 ของ Master ที่ถูกวางไว้ใน Slave Address ภายใน Address Field ของข้อมูลนั้น และเมื่อ Slave ต้องการตอบกลับ มันจะถูกวางลงบนที่อยู่ที่มันเป็นเจ้าของภายใน Address Field ของการตอบกลับ เพื่อแจ้งให้ Master รู้ว่า Slave กำลังมีการตอบกลับของข้อมูล

ที่อยู่ตำแหน่งที่ 0 คือที่อยู่ที่ถูกจองไว้สำหรับการส่งข้อมูลในลักษณะการกระจายข้อมูล (Broadcast) เรียกพื้นที่นั้นว่า Broadcast Address ซึ่งจะเป็นที่รู้กันของ Slave ทุก ๆ ตัวเมื่อ Modbus Protocolมีการใช้งานบนเครือข่ายชั้นสูง ซึ่งการส่งข้อมูลแบบกระจายข้อมูลนี้อาจไม่ได้รับการยอมรับหรืออาจถูกแทนด้วยวิธีการอื่น ๆ เช่น Modbus Plus ใช้การแบ่งของ Global Databast ซึ่งวิธีการนี้เป็นวิธีการที่สามารถทำให้การอัพเดตข้อมูลเกิดขึ้นได้โดยโดยลักษณะเฉพาะของการสับเปลี่ยนหมุนเวียนของข้อมูลนั้น ๆ

2.3.9.3 How the Function Field in Handle (จัดการกับ Function Field ได้อย่างไร)

Function Code ของ Message Frame จะบรรจุตัวอักษรในโหมดแอสกี 2 ตัว หรือโหมด RTU จำนวน 8 บิต ซึ่งรหัสที่สมบูรณ์จะอยู่ในช่วง 1-255 และรหัสบางรหัสเท่านั้นที่สามารถใช้งานได้กับคอนโทรลเลอร์ขณะที่บางรหัสจะนำไปประยุกต์ใช้ได้กับโมเดลเท่านั้น และอาจจะมีโมเดลที่ถูกสำรองไว้ให้มีการใช้งานในอนาคตเมื่อข้อมูลถูกส่งจาก Master ไปยัง Slave นั้น Function Code Field จะบอกให้ Slave ทราบถึงประเภทของการกระทำที่ Master ต้องการร้องขอให้ Slave ดำเนินการ ตัวอย่างเช่น การอ่านค่า สถานะ On/Off ของกลุ่มรีจิสเตอร์ การวินิจฉัยการอ่านสถานะของ Slave ที่เขียนถึง Designated Coils หรือ Register รวมไปถึงการยอมให้ทำการ Loading หรือการตรวจสอบโปแกรมภายใน Slave และเมื่อ Slave มีการตอบกลับไปยัง Master มันจะใช้ Function Code Field เพื่อบ่งบอกให้ Master ทราบถึงการตอบกลับที่เป็นปกติ (Error Free) หรืออาจมีความผิดปกติบางประเภทเกิดขึ้น (Exception Response) ซึ่งหากการตอบกลับเป็นปกติแล้วนั้น Slave จะสามารถสะท้อนค่า Function Code เดิมกลับไปได้อย่างง่ายดาย แต่หากเกิดความผิดพลาดหรือเกิด Exception Response ขึ้น Slave จะส่งรหัสที่เทียบเท่ากับ Function Code เดิมกลับไป และจะทำการตั้งค่าMost-Significant ในลอจิกบิตให้เป็น1

<u>ตัวอย่าง</u>

1). ข้อความจาก Master ถึง Slave ที่ทำการอ่านกลุ่มของ Holding Register ควรจะเป็นไปตาม Function Code ต่อไปนี้ : 0000 0011(Hexadecimal 03)

2). ถ้าหาก Slave Device นำการร้องขอการกระทำที่อยู่นอกเหนือจากการร้องขอจาก Master การตอบกลับของ Slave จะเป็นรหัสที่แสดงถึงความผิดพลาดของการร้องขอนั้น และหากเกิด Exception ขึ้นค่าที่จะถูกส่งกลับมาคือ : 1000 0011 (Hexadecimal 83)

นอกจากนี้การดัดแปลง Function Code สำหรับการเกิด Exception Response ซึ่ง Slave จะวางเอกลักษณ์เฉพาะของรหัสไว้ใน DataField ของ Response Message มันจะทำหน้าที่บอกให้ Master ทราบว่าประเภทของการเกิดความเป็นผิดพลาดเป็นประเภทใดหรือบอกถึงเหตุผลของการเกิด Exception Response และ Master Device's Application Program จะทำการตอบรับการจัดการ Exception Response ที่เกิดขึ้น โดยจะแสดง Process อีกครั้งหลังจากที่มีการวินิจฉัยข้อมูลที่ถูกส่งไปยัง Slave และ Notify Operators

2.3.9.4 Content of the Data Field (เนื้อหาของ Data Field)

Data Field เป็นการสร้างโดยใช้การตั้งค่าของเลขฐานสิบหกจำนวน 2 ค่า ซึ่งอยู่ในช่วง 00 ถึง FF โดยเป็นการจับคู่กันของรหัสแอสกี หรือจากบิตข้อมูลในระบบ RTU จำนวน 1 ตัวอักษร เพื่อให้สอดคล้องกับความต่อเนื่องของเครือข่ายในโหมดของการส่งผ่านข้อมูล ซึ่ง Data Field ของข้อมูลจะถูกส่งจาก Master ไปยัง Slave Device ที่บรรจุข้อมูลไว้ โดย Slave ต้องนำ Function Code มาใช้ในส่วนที่เป็นการกำหนดการกระทำต่าง ๆ ภายใน Device และสามารถทำให้ส่วนประกอบนั้นคล้ายคลึงกับ Register Address ที่มีการจัดการและนับปริมาณของไบต์ข้อมูลที่เกิดขึ้นจริงภายในฟิลด์

<u>ตัวอย่าง</u>

1). ถ้าหาก Master ทำการร้องขอให้ Slave อ่านกลุ่มของ Holding Register (Function Code 03) Data Field จะระบุจุดเริ่มต้นของ Register และระบุว่ามี Register จำนวนเท่าใดที่จะทำการอ่าน

2). ถ้าหาก Master เขียนคำร้องขอไปยังกลุ่มของ Register ภายใน Slave (Function Code 10) Data Field จะระบุจุดเริ่มต้นของ Register และระบุว่ามี Register จำนวนเท่าใดที่จะทำการเขียนและนับไบต์ของข้อมูลภายใน Data Field รวมไปถึงข้อมูลที่จะเขียนใน Register

3). หากไม่มีความผิดพลาดเกิดขึ้น Data Field จะส่งข้อมูลจาก Slave ไปยัง Master ที่บรรจุการร้องขอข้อมูลนั้นไว้และหากมีความผิดพลาดเกิดขึ้น ฟิลด์ที่บรรจุ Exception Code ของ Master Application ไว้จะสามารถกำหนดการกระทำที่จะนำมาใช้ถัดไปได้

4). Data Field สามารถที่จะไม่มี Address (ความยาวเป็น 0) ในการบรรจุประเภทของข้อมูลได้ ตัวอย่างเช่น ในการร้องขอจาก Master Device เพื่อให้ Slave ทำการตอบกลับการ

ติดต่อสื่อสารแม้ว่าจะถูกล็อคไว้ก็ตาม (Function Code 0B) Slave จะไม่ต้องการข้อมูลนอกเหนือจากนี้ โดยจะมี Function Code เพียงอย่างเดียวเท่านั้นที่จะระบุถึงการกระทำต่าง ๆ

2.3.9.5 Content of Error Checking Field (เนื้อหาของ Error Checking Field)

ประเภทของการตรวจสอบความผิดพลาดมีด้วยกัน 2 ประเภท สำหรับการใช้งานตามมาตรฐานของ Mosbus Network ซึ่งเนื้อหาของฟิลด์ที่ใช้ในการตรวจสอบความผิดพลาดจะขึ้นอยู่กับวิธีการใดวิธีการหนึ่งที่นำมาใช้ ซึ่งในที่นี้มีอยู่ 2 ประเภทดังนี้

1). RTU

เมื่อโหมดRTU ถูกนำมาใช้ใน Character Framing ฟิลด์ในการตรวจสอบความผิดพลาดจะบรรจุค่า 16-bit ที่ดำเนินการโดยการแบ่งเป็นข้อมูล 8-bit จำนวน 2-byte ค่าของการตรวจสอบความผิดพลาดเป็นผลที่ได้จากการคำนวณด้วยวิธีของ Cyclic Redundancy Check (CRC) ที่ดำเนินการบนเนื้อหาของข้อมูล ฟิลด์ของ CRC จะถูกผนวกกับข้อความเช่นเดียวกับฟิลด์สุดท้ายใน Message และเมื่อเสร็จสิ้น Low-Order Byte ของฟิลด์ มันจะผนวกเข้ามาเป็นอันดับแรก ตามมาด้วย High-Order Byte ซึ่ง CRC ที่เป็น High-Order Byte จะเป็นไบต์สุดท้ายที่ถูกส่งไปใน Message

2). ASCII

เมื่อโหมดแอสกีถูกนำมาใช้ใน Character Framing ฟิลด์ที่ใช้ในการตรวจสอบความผิดพลาดของข้อมูลละบรรจุตัวอักษรแอสกีไว้สองตัวด้วยกัน ซึ่งการตรวจสอบดังกล่าวเป็นผลที่ได้จากการคำนวณด้วยวิธี Longitudinal Redundancy Check (LRC) ซึ่งเป็นการคำนวณที่ดำเนินการบนเนื้อหาของข้อมูล ส่วนที่สำคัญที่สุดคือที่จุดเริ่มต้นนั้นจะเริ่มด้วยเครื่องหมาย ":" (Colon) และสิ้นสุดด้วยตัวอักษร CRLF

วิธีการ LRC นั้นจะเป็นการผนวกเข้ากับ Message ที่อยู่ก่อนหน้าฟิลด์สุดท้ายของตัวอักษร CRLF หรือคือฟิลด์สุดท้ายก่อนที่จะถึงฟิลด์ของตัวอักษา CRLF นั่นเอง

2.3.10 Error Checking Method

ในการติดต่อสื่อสารแบบ Modbus Protocal นั้น เมื่อมีการรับส่งข้อมูลเกิดขึ้นระหว่าง Master และ Slave จะต้องมีการตรวจสอบความผิดพลาดของข้อมูล ซึ่งวิธีที่ใช้ในการตรวจสอบความผิดพลาดมีอยู่ 2 ประเภทด้วยกัน แบ่งเป็น Parity Checking (Even หรือ Odd) สามารถเลือกรูปแบบการประยุกต์ของตัวอักษรได้ และอีกประเภทคือ Frame Checking (LRC และ CRC) จะประยุกต์กับข้อความทั้งหมดทั้งการตรวจสอบแบบตัวอักษรและแบบเฟรมข้อมูล ซึ่งเป็นการสร้าง

ใน Master Device และมีการประยุกต์เนื้อหาของข้อมูลก่อนที่จะส่งออกไป และ Slave Device จะตรวจสอบทั้งตัวอักษรและเฟรมข้อมูลทั้งหมดในระหว่างที่มีการรับ โดยวิธีการตรวจสอบความผิดพลาดมีดังต่อไปนี้

2.3.10.1 Parity Checking

ผู้ใช้สามารถกำหนดค่าบนคอนโทรลเลอร์ให้เป็นแบบ Even หรือ Odd Parity Checking หรืออาจจะกำหนดเป็นแบบ No Parity Checking ก็ได้ ซึ่งการกำหนดบิตของ Parity สามารถจะตั้งค่าต่าง ๆ ได้ ถ้าทั้ง Even หรือ Odd Parity เป็นการระบุปริมาณตัวอักษรโดยตัวอักษร 1 ตัวจะเท่ากับ 1 บิตของข้อมูล (7 บิต ข้อมูลสำหรับโหมด ASCII และ 8 บิตข้อมูลสำหรับโหมด RTU) ซึ่งParity Bit จะถูกตั้งค่าเป็น 0 หรือ 1 เพื่อเป็นผลรวมใน Even หรือ Odd ของ 1 บิต

ตัวอย่างเช่น บิตข้อมูลจำนวน 8 บิต ที่บรรจุใน RTU Character Frame เป็น 1100 0101

ปริมาณผลรวมของ 1 บิตในเฟรมข้อมูลจะเท่ากับ 4 ถ้าหาก Even Parity ถูกใช้งาน เฟรมข้อมูลของ Parity บิตข้อมูลจะเป็น 0 ทำให้ปริมาณผลรวมของ 1 บิตข้อมูลเป็นเลขคู่ (4) และถ้า Odd Parity ถูกใช้งาน เฟรมข้อมูลของ Parity บิตข้อมูลจะเป็น 1 ทำให้ปริมาณผลรวมของ 1 บิตข้อมูลเป็นเลขคี่ (5) เมื่อข้อมูลถูกส่งออกไปParity Bit จะทำการคำนวณและประยุกต์กับเฟรมข้อมูล ซึ่ง Device ที่ใช้ในการรับจะนับปริมาณของ 1 บิตและตั้งค่าเป็น Error ถ้าหากผลที่คำนวณออกมาไม่ตรงกับค่าที่ได้กำหนดไว้สำหรับ Device (ทุก ๆ Device บน Modbus Networkจะต้องกำหนดให้ตรงกับวิธีการของ Parity Check)

Parity Checking สามารถค้นพบได้เฉพาะความผิดพลาดที่เกิดจาก Odd Number ของบิตที่ถูกเลือกหรือวางใน Character Frame ระหว่างการส่งข้อมูล ตัวอย่างเช่น ถ้า Odd Parity Checking ทำงานและไบต์ข้อมูลลดเหลือ 2 ไบต์ใน 1 บิตข้อมูลซึ่งจากเดิมบรรจุไว้ 3 ไบต์ใน 1 บิต ผลที่ตามมาคือจะนับเลขคี่เป็น 1 บิต และถ้ามีการระบุให้ใช้ No Parity Checking และใช้ No Parity Bit ในการส่งข้อมูลและสามารถใช้งาน Parity Check ได้ จะต้องมีการเพิ่ม Stop Bit ในการส่งข้อมูลไปยัง Message Frame

2.3.10.2 Cyclical Redundancy Checking (CRC)

การตรวจสอบความผิดพลาดแบบ Cyclical Redundancy Checking นี้ เป็นการตรวจสอบความผิดพลาดที่ใช้ในโหมด RTU ซึ่งในโหมด RTU ข้อมูลจะประกอบไปด้วย Error-Checking Field โดยวิธีการดังกล่าวจะไม่คำนึงถึงวิธีการของ Parity Check เลย

ใน CRC Field จะบรรจุข้อมูลเลขฐานสองไว้ 2 ไบต์ซึ่งมีข้อมูลทั้งหมด 16 บิต ค่าของ CRC จะเป็นการคำนวณโดยการส่ง Device ซึ่ง CRC จะเข้าไปผนวกกับข้อมูล และ Device ที่เป็นตัวรับข้อมูลนั้นจะทำการคำนวณ CRC อีกครั้งในระหว่างที่มีการรับข้อมูลและเปรียบเทียบค่าที่ได้จากการคำนวณกับค่าที่เกิดขึ้นจริงที่ได้รับใน CRC Field หากค่าทั้งสองไม่ตรงกัน แสดงว่ามี Error เกิดขึ้น

การตรวจสอบความผิดพลาดของข้อมูลแบบ CRC นั้น จะใช้วิธีการของอัลกอรึทึม โดยการนำข้อมูลในแต่ละไลต์มาหาร ซึ่งอยู่ในรูป X^2 $+X^2$ $+X^2+1$ ในรูปสมการโพลิโนเมียลของเลขฐานสอง

<u>หลักการของ CRC คือ</u>

1). บิตข้อมูล M(x) มีความยาวของข้อมูล m บิต ถูกหารโดย G(x) มีความยาว R บิตใช้การลบแบบ XOR

2). m > r

3). นำเศษที่ได้จากการหาร r-1 บิต มาปะที่ท้ายข้อมูล M(x) จะได้เฟรมข้อมูลใหม่ซึ่งจะถูกส่งไปให้ผู้รับ

4). ผู้รับจะนำเอาข้อมูลมาหารด้วย G(x) ซึ่งหากหารไม่ลงตัวแสดงว่าข้อมูลเกิดความผิดพลาด

<u>ตัวอย่างการคำนวณ CRC</u>

Frame ข้อมูล M(x) คือ 1 1 0 1 0 1 1 0 1 1 ค่า m = 10 บิต

G(x) คือ 1 0 0 1 1 ค่า r = 5 บิต

m > r

ตั้งหารสั้น 10011 <u>| 1101011011000</u>

10011 <u>| 111011000</u> เศษ 1001

10011 <u>| 11000</u> เศษ 1110

เฟรมที่ส่ง 1 1 0 1 0 1 1 0 1 1 <- 1 1 1 0 CRC

2.3.10.3 Longitudinal Redundancy Checking (LRC)

การตรวจสอบความผิดพลาดแบบ Longitudinal Redundancy Checking นี้ เป็นการตรวจสอบความผิดพลาดที่ใช้ในโหมด ASCII ซึ่งจะประกอบไปด้วย Error-Checking Field ของ Message และที่พิเศษคือการเริ่มต้นฟิลด์ด้วยเครื่องหมาย ":" (Colon) และจะสิ้นสุดด้วยตัวอักษร CRLF ซึ่งวิธีการนี้ไม่จำเป็นต้องคำนึงถึงการตรวจสอบแบบ Parity เช่นเดียวกับการตรวจสอบแบบ CRC

ภายใน LRC Field จะมีลักษณะเป็นไบต์ข้อมูล 1 ไบต์ที่บรรจุตัวเลขฐานสองไว้จำนวน 8 บิต ซึ่งค่าของ LRC เป็นการคำนวณโดยใช้ Transmitting Device ซึ่งวิธีการคือการนำ LRC ไปผนวกกับ Message และในส่วนของ Receiving Device จะทำการคำนวณ LRC ระหว่างที่มีการรับข้อมูล และนำค่าที่ได้จากการคำนวณและค่าที่เกิดขึ้นจริงภายใน LRC Field มาเปรียบเทียบกัน ถ้าหากค่าทั้งสองไม่เท่ากัน ผลที่ตามมาคือ Error ที่เกิดขึ้น

การตรวจสอบความผิดพลาดในโหมด LRC นั้น จะใช้วิธีการของอัลกอริทึมที่นำค่าข้อมูลของตัวอักษร ASCII ที่ส่งออกไปตั้งแต่แรก นำมาผ่านลอจิก Exclusive-OR (XOR) ไปเรื่อย ๆ จนถึงตัวสุดท้ายที่ค่าตัวอักษร BCC (Block Check Character)

<u>หลักการของ LRC คือ</u>

หา Parity ของแต่ละบิตของทุกตัวอักษรข้อมูล

ตารางที่ 2.1 การหา Parity ของ LRC

b8 Parity														
b7-b0 data	A	B	C	D	E	F	G	H	I	J	K	L	M	N

P	7	6	5	4	3	2	1	Character
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	1	0	2
0	1	0	0	0	0	1	1	3
1	1	0	0	0	1	0	0	4
0	1	0	0	0	1	0	1	5
0	1	0	0	0	1	1	0	6
1	1	0	0	0	1	1	1	7
1	1	0	0	1	0	0	0	8
0	1	0	0	1	0	0	1	9
0	1	0	0	1	0	1	0	10
1	1	0	0	1	0	1	1	11
0	1	0	0	1	1	0	0	12
1	1	0	0	1	1	0	1	13
1	1	0	0	1	1	1	0	14
1	0	0	0	1	1	1	1	LRC

2.3.11 การเปรียบเทียบ Message Field ระหว่าง RTU และ ASCII

ตารางที่ 2.2 เปรียบเทียบความแตกต่างของ Message Field ระหว่าง RTU และ ASCII

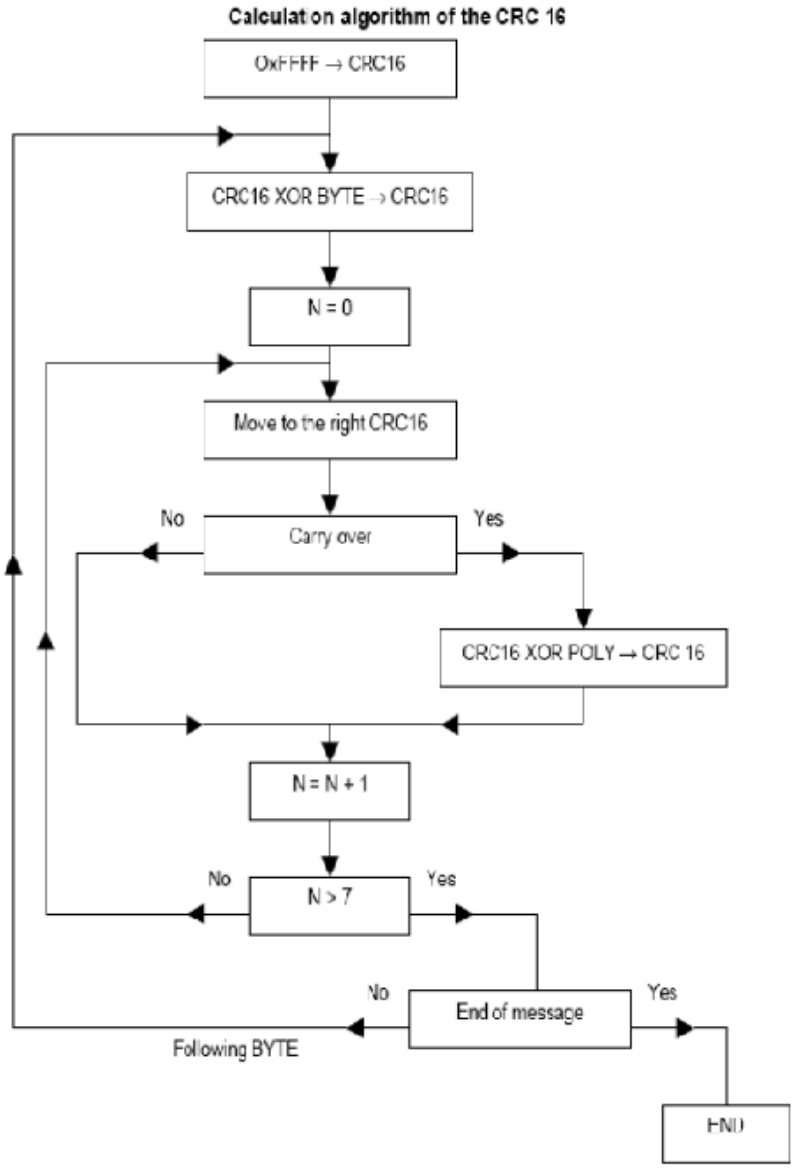
		Slave address		Function code		Relay address				Number of data to be read				Error check			
RTU		0	1	0	1	0	4	6	0	0	0	0	C	'nn' 'nn'			
ASCII	:	30	30	30	31	30	34	36	30	30	30	30	43	38	45	CR	LF

จะเห็นว่าลักษณะของเฟรมข้อมูลจะเหมือนกัน แตกต่างกันตรงส่วนของข้อมูลที่อยู่ภายในเฟรมข้อมูล นั่นคือ ในโหมด RTU ข้อมูลในเฟรมจะเป็นเลขฐานสอง แต่ข้อมูลในโหมด ASCII ข้อมูลภายในเฟรมจะเป็นตัวอักษรที่เป็นรหัส ASCII ในค่าข้อมูลนั้น ๆ ดังนั้นค่าที่เท่ากับ 0 ในเฟรมข้อมูลแบบ RTU จึงต้องเป็นค่าที่เท่ากับ 30 ในโหมด ASCII และเช่นกันกับเฟรมข้อมูลการตอบกลับจากเครื่อง Slave ในโหมด ASCII และ RTU แสดงตัวอย่างเปรียบทียบได้ดังนี้

ตารางที่ 2.3 เปรียบเทียบความแตกต่างของ Data Frame ระหว่าง RTU และ ASCII

		Slave address		Function code		Number of data		Data				Error check			
RTU		0	1	0	1	0	2	0	8	0	0	'nn' 'nn'			
ASCII	:	30	31	30	31	30	32	30	30	30	30	46	34	CR	LF

ความแตกต่างอีกอย่างในโหมด ASCII กับ RTU คือส่วนของการตรวจสอบความผิดพลาดท้ายเฟรมข้อมูลนั้น ในโหมด RTU จะใช้วิธีการ Cyclical Redundancy Check แต่ในโหมด ASCII จะใช้วิธีการ Longitudinal Redundancy Check และจะปิดท้ายข้อมูลด้วยรหัสแอสกี Character Return (CR) และ Line Feed (LF)



XOR = exclusive or

N = number of information bits

POLY = calculation polynomial of the CRC 16 = 1010 0000 0000 0001

(Generating polynomial = $1 + x_2 + x_{15} + x_{16}$)

In the CRC 16, the 1st byte transmitted is the least significant one.

รูปที่ 2.29 แผนผังของการตรวจสอบความผิดพลาดแบบ CRC

Example of CRC calculation (frame 02 07)

CRC register initialization		1111	1111	1111	1111
XOR 1st character		0000	0000	0000	0010
		1111	1111	1111	1101
	Move 1	0111	1111	1111	1110|1
Flag to 1, XOR polynomial		1010	0000	0000	0001
		1101	1111	1111	1111
	Move 2	0110	1111	1111	1111|1
Flag to 1, XOR polynomial		1010	0000	0000	0001
		1100	1111	1111	1110
	Move 3	0110	0111	1111	1111|0
	Move 4	0011	0011	1111	1111|1
		1010	0000	0000	0001
		1001	0011	1111	1110
	Move 5	0100	1001	1111	1111|0
	Move 6	0010	0100	1111	1111|1
		1010	0000	0000	0001
		1000	0100	1111	1110
	Move 7	0100	0010	0111	1111|0
	Move 8	0010	0001	0011	1111|1
		1010	0000	0000	0001
		1000	0001	0011	1110
XOR 2nd character		0000	0000	0000	0111
		1000	0001	0011	1001
	Move 1	0100	0000	1001	1100|1
		1010	0000	0000	0001
		1110	0000	1001	1101
	Move 2	0111	0000	0100	1110|1
		1010	0000	0000	0001
		1101	0000	0100	1111
	Move 3	0110	1000	0010	0111|1
		1010	0000	0000	0001
		1100	1000	0010	0110
	Move 4	0110	0100	0001	0011|0
	Move 5	0011	0010	0000	1001|1
		1010	0000	0000	0001
		1001	0010	0000	1000
	Move 6	0100	1001	0000	0100|0
	Move 7	0010	0100	1000	0010|0
	Move 8	0001	0010	0100	0001|0
			Most significant		least significant

The CRC 16 of the frame is then: 4112

รูปที่ 2.30 ตัวอย่างของการคำนวณ CRC (frame 02 07)

ตารางที่ 2.4 เปรียบเทียบความแตกต่างระหว่างโหมด RTU และ ASCII

	Modbus/ASCII		Modbus/RTU	
Characters	ASCII **0...9** and **A..F**		Binary **0...255**	
Error check	**LRC** Longitudinal Redundancy Check		**CRC** Cyclic Redundancy Check	
Frame start	character **':'**		3.5 chars silence	
Frame end	characters **CR/LF**		3.5 chars silence	
Gaps in message	1 sec		1.5 times char length	
Start bit	**1**		**1**	
Data bits	**7**		**8**	
Parity	even/odd	none	even/odd	none
Stop bits	**1**	**2**	**1**	**2**

2.3.12 การส่งข้อมูลระหว่าง Modbus Protocol และ Terminal Device

การส่งข้อมูลระหว่าง Modbus Protocol กับ Terminal Device จะต้องใช้ Modbus Field Bus Plug (MRP21-FBP) ซึ่งก็คือสายเชื่อมต่อระหว่าง Modbus Protocol กับ Terminal โดยจะมี Input และ Output อยู่ภายใน และการส่งข้อมูลผ่านทาง Modbus Field Bus Plug สามารถทำได้สองวิธีคือ

2.3.12.1 Parallel Communication

การแลกเปลี่ยนสัญญาณแบบ Parallel Communication นั้น จะใช้การเชื่อมต่อแบบ Field Bus-Neutral Interface ซึ่งจะต้องมีการกำหนดขอบเขตข้อมูลที่มากที่สุด โดยให้ 1Digital Output (สัญญาณที่ใช้ควบคุมไปยัง Terminal) รวมกับ 2 Digital Input (สัญญาณตอบกลับจาก Terminal Device) ซึ่งถ้า Field Bus Plug ไม่ได้รับ Telegram จาก Termianl Deviceขณะที่เครื่องกำลังทำงานอยู่นั้น จะต้องทำการตั้งค่าการแลกเปลี่ยนข้อมูลใหม่ทันที

2.3.12.2 Serial Communication

การแลกเปลี่ยนสัญญาณแบบ Serial Communication จะได้รับความช่วยเหลือจาก Serial Data Protocol ซึ่งจะใช้การเชื่อมต่อด้วยรูปแบบ Field Bus-Neutral Interface ระบบเลขฐานสองระบบอนาลอก และ Parameter &Diagnostic Data ในการรับและการส่งข้อมูล และเมื่อ Field Bus Plug กำลังจะได้รับ Valid Telegram จาก Terminal Device โหมดดังกล่าวจะทำการเปลี่ยนแปลงข้อมูลและหลังจากนั้นข้อมูลดังกล่าวจะถูกตั้งค่าไว้ไม่ให้มีการเปลี่ยนแปลงอีก

2.3.13 ลักษณะของข้อมูลภายใต้รูปแบบการทำงานของ Modbus Protocol

ข้อมูลต่าง ๆ ของอุปกรณ์ที่เป็น Salve จะเก็บอยู่ในตารางที่มีคุณลักษณะต่างกันทั้งหมด 4 ตาราง โดยแบ่งสองตารางแรกเป็นค่า On/Off (coil) ซึ่งจะเก็บข้อมูลแบบ Discrete ส่วนอีกสองตารางที่เหลือจะเก็บข้อมูลที่เป็นค่าตัวเลข (Register) ในส่วนของ Coil และ Register ต่างก็มีตารางแบบ Read-Only คือเขียนอย่างเดียว และแบบ Read-Write คือทั้งอ่านและเขียนข้อมูลลงไปได้ ซึ่งแต่ละตารางจะกำหนดให้มีข้อมูลทั้งหมด 9,999 ค่า ในส่วนของ Coil หรือ Contact ซึ่งเป็นการจัดเก็บข้อมูลแบบ Discrete ข้อมูลแต่ละตัวจะถูกระบุตำแหน่งด้วย Address ตั้งแต่ 0000 ถึง 270E ซึ่งเป็นเลขฐานสิบหก (แปลงเป็นเลขฐานสิบคือ 0 ถึง 9998)

Register แต่ละตัวใช้พื้นที่ 16 bits = 2 bytes = 1 word และมี address ตั้งแต่ 0000 ถึง 270E เช่นกัน

Data Addresses	Coil/Register Numbers	Type	Table Name
0000 to 270E	1-9999	Read-Write	Discrete Output Coils
0000 to 270E	10001-19999	Read-Only	Discrete Input Contacts
0000 to 270E	30001-39999	Read-Only	Analog Input Registers
0000 to 270E	40001-49999	Read-Write	Analog Output Holding Registers

รูปที่ 2.31 การเก็บข้อมูล

Coil/Register Number จะเป็นเพียงแค่ Location Name ซึ่งจะไม่ปรากฏอยู่ในข้อมูลที่ใช้ในการรับและการส่ง แต่ Data Address จะถูกระบุอยู่ในข้อมูล ซึ่งเปรียบเสมือนการทำงานของบุรุษไปรษณีที่ส่งจดหมายได้จะต้องมีบ้านเลขที่ปรากฏอยู่หน้าซองจดหมาย

2.3.14 Modbus Function Format

2.3.14.1 How Numerical Values are Expressed

การระบุค่าของ Numerical Values (เช่น Address, Code or Data) ต่าง ๆ นั้น จะแสดงออกมาเป็นเลขฐานสิบภายในส่วนข้อมูล ซึ่งจะแปลงเป็นเลขฐานสิบหกในไบต์ข้อมูล โดยจะถูกบรรจุไว้ภายใน Message Field

2.3.14.2 Data Address in Modbus Message

Data Address ทั้งหมดภายใน Modbus Message เป็นการอ้างอิงข้อมูลไปยังค่าศูนย์ ซึ่งสิ่งที่เกิดขึ้นเป็นอันดับแรกของ Data Item คือที่อยู่ที่อ้างอิงในที่อยู่ตำแหน่งที่ศูนย์ ดังตัวอย่างต่อไปนี้

1). Coil จะรู้ว่า Coil ที่เป็น Coil ตำแหน่งที่ 1 ในโปรแกรมควบคุมคือที่อยู่ของ 'Coil 0000' ภายใน Data Address Field ของ Modbus Message

2). Coil 127(ฐาน10) คือที่อยู่ของ Coil 0x7E (126 ในระบบเลขฐานสิบ)

3). Holding Register 40001 คือที่อยู่ของ Register 0000 ใน Data Address Field ของ Message ซึ่ง Function Code Field ได้ระบุ Holding Register ไว้เรียบร้อยแล้ว นั่นคือการอ้างอิงถึง '4xxxx' ไปโดยปริยาย

4). Holding Register 40108 คือที่อยู่ของ Register 006B (หรือ 107 ในระบบเลขฐานสิบ)

2.3.14.3 Field Content in Modbus Message

รูปที่ 2.32 แสดงตัวอย่างของ Modbus Query Message และรูปที่ 2.33 เป็นตัวอย่างของ Normal Response ซึ่งทั้งตัวอย่างจะแสดงเนื้อหาของ Field ในตัวเลขฐานสิบหก และยังแสดงว่า Message จะเลือกเฟรมในโหมด ASCII หรือ RTU อย่างไร

Master Query เป็นการอ่าน Holding Register ที่ส่งคำร้องไปยัง Slave Device Address ที่ 06 และ Message จะร้องขอข้อมูลจาก Holding Register 3 ตัวคือ 40108 ไปจนถึง 40110 ซึ่งการระบุตำแหน่งของ Register จะเริ่มต้นที่ 0107 (006B ในระบบเลขฐานสิบหก) เสมอ และ Slave Response จะสะท้อน Function Code ที่บ่งบอกถึง Normal Response ซึ่ง 'Byte Count'Field เป็นการระบุว่า 8-bit Data Item จะถูกส่งกลับมาเท่าไหร่

'Byte Count' Field จะแสดงการนับของ 8-bit byte ที่ตามมาในข้อมูล สำหรับ ASCII และ RTU ซึ่ง ASCII ค่าดังกล่าวจะเป็นการนับแบบ One-Half ของตัวอักษร ASCII ที่แท้จริงภายในข้อมูล และในโหมด ASCII ค่าที่เป็น 4-bit hexadecimal จะต้องการตัวอักษรASCII 1ตัว ดังนั้นจะต้องมีตัวอักษรASCII 2ตัวในข้อความที่บรรจุข้อมูลแบบ 8-bit

ตัวอย่างเช่น ค่า 63 ที่เป็นเลขฐานสิบหกเป็นการส่งข้อมูล 8-Bit จำนวน 1 ไบต์ภายในโหมด RTU (01100011) ซึ่งเหมือนกับค่าที่ส่งในโหมด ASCII ที่ต้องการ 2 ไบต์ สำหรับ ASCII '6'(0110110) และ '3' (0110011) 'Byte Count'Field จะนับข้อมูลที่เป็น 8-bit โดยจะไม่คำนึงถึงวิธีการของ Character Framing (ASCII หรือ RTU)

How to Use the Byte Count Field

เมื่อต้องการสร้าง Response ภายในบัฟเฟอร์ที่ใช้คาดการการนับไบต์ที่เท่ากับการนับไบต์แบบ 8-bit ใน Message Data และค่าที่เป็นส่วนสำคัญของรายละเอียดภายในฟิลด์ทั้งหมดจะประกอบด้วย Byte Count Field

จากรูปที่ 2.32 แสดง Byte count field ดำเนินการใน typical response ได้อย่างไร

QUERY

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Starting Address Hi	00	0 0	0000 0000
Starting Address Lo	6B	6 B	0110 1011
No. of Registers Hi	00	0 0	0000 0000
No. of Registers Lo	03	0 3	0000 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	17	8

รูปที่ 2.32 เฟรมข้อมูลของ Master Query with ASCII/RTU

RESPONSE

Field Name	Example (Hex)	ASCII Characters	RTU 8–Bit Field
Header		: (colon)	None
Slave Address	06	0 6	0000 0110
Function	03	0 3	0000 0011
Byte Count	06	0 6	0000 0110
Data Hi	02	0 2	0000 0010
Data Lo	2B	2 B	0010 1011
Data Hi	00	0 0	0000 0000
Data Lo	00	0 0	0000 0000
Data Hi	00	0 0	0000 0000
Data Lo	63	6 3	0110 0011
Error Check		LRC (2 chars.)	CRC (16 bits)
Trailer		CR LF	None
	Total Bytes:	23	11

รูปที่ 2.33 เฟรมข้อมูลของ Slave Response with ASCII/RTU

2.3.15 Function Code Categories

หมวดหมู่ทั้ง 3 ของ Modbus Function code นั่นคือ

2.3.15.1 Public Function Codes

1). เป็นนิยามที่ดีของ Function codes

2). การันตีได้ถึงความเป็นเอกลักษณ์

3). เป็นเอกสารสาธารณะ

4). มีการใช้การทดสอบความสอดคล้อง

5). ประกอบด้วยทั้งนิยามที่เป็นสาธารณะที่ Function code ได้รับมอบหมายตลอดจน Function code ที่ไม่ได้รับมอบหมายให้จองพื้นที่สำหรับการใช้ในอนาคต

2.3.15.2 User-Defined Function Codes

1). จะมีสองระดับของ User-defined function code เช่น 65 ถึง 72 และจาก 100-110 decimal

2). ผู้ใช้สามารถเลือกการดำเนินการ Function codeที่ไม่รองรับโดยเฉพาะเจาะจงได้

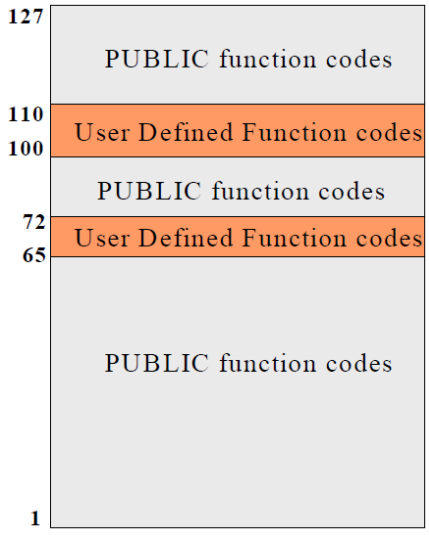
3). ไม่มีการการันตีการใช้ของการเลือก Function code ที่เป็นเอกลักษณ์

4). ถ้าผู้ใช้ต้องการที่จะทำการตำแหน่งใหม่ใน Functionallity ให้เป็นไปตาม Public Function Code ผู้ใช้จะต้องเริ่มต้นโดย RFC แนะนำให้เปลี่ยนไปเป็นหมวดหมู่ที่เป็นสาธารณะและมีการมอบหมายให้กับ Public function code อันใหม่

5). การจัด Modbus เพื่อการจองอย่างรวดเร็วทางด้านขวาเพื่อที่จะพัฒนาการเสนอ RFC

2.3.15.3 Reserved Function Codes

1). Function code ที่เป็นปัจจุบันถูกใช้โดยบางบริษัทสำหรับผลิตภัฒฑ์ที่เป็นมรดกและไม่ได้ใช้งานสำหรับการใช้สาธารณะ



รูปที่ 2.34 หมวดหมู่ Function Code ของ Modbus

ตารางที่ 2.5 นิยามของ Public Function Code

				Function Codes			
				code	Sub code	(hex)	Section
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02	6.2
		Internal Bits Or Physical coils	Read Coils	01		01	6.1
			Write Single Coil	05		05	6.5
			Write Multiple Coils	15		0F	6.11
	16 bits access	Physical Input Registers	Read Input Register	04		04	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03	6.3
			Write Single Register	06		06	6.6
			Write Multiple Registers	16		10	6.12
			Read/Write Multiple Registers	23		17	6.17
			Mask Write Register	22		16	6.16
			Read FIFO queue	24		18	6.18
	File record access		Read File record	20		14	6.14
			Write File record	21		15	6.15
Diagnostics			Read Exception status	07		07	6.7
			Diagnostic	08	00-18,20	08	6.8
			Get Com event counter	11		0B	6.9
			Get Com Event Log	12		0C	6.10
			Report Slave ID	17		11	6.13
			Read device Identification	43	14	2B	6.21
Other			Encapsulated Interface Transport	43	13,14	2B	6.19
			CANopen General Reference		13		6.20

2.3.16 Function Code

อุปกรณ์ Slave แต่ละตัวภายใน Network นั้น จะมี Address ไม่ซ้ำกัน ซึ่งจะมีตั้งแต่ 1-247 เมื่อ Master ร้องขอข้อมูลไป ข้อมูล Byteแรกที่ถูกส่งไปจะต้องเป็นค่า Slave Address เพื่อระบุว่าจะส่งไปที่ Slave ตัวใด ส่วนใน Byte ต่อไปจะเป็น Function Code เพื่อระบุว่าต้องการติดต่อกับตารางใด เป็นการบอกว่าต้องการที่จะอ่านข้อมูลอย่างเดียวหรือทั้งอ่านและเขียนข้อมูล ซึ่ง Function Code สามารถแบ่งได้เป็น 4 ลักษณะดังต่อไปนี้

ตารางที่ 2.6 แสดง Function Code Definition

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

2.3.16.1 Discrete output coils หรือ Read Coil Status (0x01)

Discrete Output Coils เป็นการอ่านสถานะ On/Off ของ Discrete Outputs ใน Slave ซึ่งวิธีการนี้จะไม่รองรับการส่งข้อมูลแบบ Broadcast

Query: Query Message ได้ระบุจุดเริ่มต้นของ Coil และปริมาณของ Coil ที่จะถูกนำมาอ่าน ซึ่ง Coil ตัวที่ 1 จะมีจุดเริ่มต้นของที่อยู่ที่ตำแหน่งที่ 0 โดย Coil ที่ 1-16 จะอยู่ในตำแหน่งที่อยู่ที่ 0-15

จากรูปที่ 2.35 แสดงตัวอย่างของการร้องขอการอ่าน Coil ที่ 20-56 จาก Slave Device ตัวที่ 17

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Points Lo	25
Error Check (LRC or CRC)	—

รูปที่ 2.35 การอ่านสถานะของ Coil Status-Query

Response : สถานะของ Coil ใน Response Message เป็นการบรรจุ Coil 1 ตัว ต่อ Data Field 1 บิต ซึ่งจะเป็นการบ่งบอกสถานะของบิต โดยบิตที่เป็น 1 จะมีสถานะเป็น On และ 0 จะมีสถานะเป็น Off โดย LSB ของข้อมูลไบต์แรกจะบรรจุที่อยู่ของ Coil ไว้ใน Query และ Coil อื่น ๆ ก็จะตามมาไปจนถึงคำสั่งที่เป็นจุดสิ้นสุดของไบต์ข้อมูล และจาก Low Order ไปจนถึง High Order ใน Subsequent Byte

ถ้าหากเรากลับปริมาณ Output โดยกลับให้ไม่เป็น Multipoint of Eight และให้บิตที่ยังเหลืออยู่ในไบต์ข้อมูลชุดสุดท้ายถูกอัดไว้กับที่อยู่ตำแหน่งที่ 0(จนกระทั่งถึง High Order ซึ่งเป็นจุดสิ้นสุดของไบต์) โดย Byte CountField จะบรรจุปริมาณของไบต์ข้อมูลที่ดำเนินการสำเร็จแล้ว

จากรูปที่ 2.36 แสดงตัวอย่างของการตอบกลับไปยัง Query บน Opposite Page

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27–20)	CD
Data (Coils 35–28)	6B
Data (Coils 43–36)	B2
Data (Coils 51–44)	0E
Data (Coils 56–52)	1B
Error Check (LRC or CRC)	—

รูปที่ 2.36 การอ่านสถานะของ Coil Status-Response

สถานะของ Coil ที่ 27-20 เป็นการแสดงค่าไบต์ข้อมูล โดยค่าที่แสดงคือ CD ซึ่งเป็นเลขฐานสิบหก และเมื่อแปลงเป็นเลขฐานสองจะได้ 1100 1101 ซึ่ง Coil ตำแหน่งที่ 27 เป็น MSB

ของไบต์ข้อมูลนี้ และ Coil ที่ 20 จะเป็น LSB ซึ่งการขยับของ Coil จากซ้ายไปขวาจะมีสถานะเป็น ON-ON-OFF-OFF-ON-ON-OFF-ON

บิตภายในของไบต์จะเป็นการแสดงถึงการขยับข้อมูลจากไปทางซ้ายของ MSB และขยับไปทางขวาของ LSB ซึ่ง Coil ในไบต์แรกจะเป็นตำแหน่งที่ 27 และจะขยับไปยังตำแหน่งที่ 20 โดยการขยับดังกล่าวเป็นการขยับจากซ้ายไปขวา และในไบต์ถัดไปจะเป็น Coil ตำแหน่งที่ 35 ไปจนถึง Coil ตำแหน่งที่ 28 ขยับจากซ้ายไปขวาเช่นเดียวกัน เมื่อมีการส่งลำดับบิตของข้อมูล มันจะไหลจาก LSB ไปยัง MSB ดังนี้ 20....27, 28....35 และต่อ ๆ ไป

ในไบต์ข้อมูลชุดสุดท้าย สถานะของ Coil ตำแหน่งที่ 56-52 จะเป็นไบต์ข้อมูลที่บรรจุค่า 1B ไว้ ซึ่งเป็นเลขฐานสิบหก หรือเมื่อเปลี่ยนเป็นเลขฐานสองจะได้ 0001 1011 ซึ่งตำแหน่งที่ 56 คือตำแหน่งบิตที่สี่จากทางด้านซ้ายและตำแหน่งที่ 25 จะเป็น LSB ของไบต์ข้อมูลนี้ โดยสถานะของ Coil ที่ 56 ไปจนถึงตำแหน่งที่ 52 จะมีสถานะเป็น ON-ON-OFF-ON-ON

2.3.16.2 Discrete input Contact หรือ Read Input Status (0x02)

Discrete Input Contact เป็นการอ่านสถานะ On/Off ของ Discrete Input ใน Slave ซึ่งวิธีการนี้จะไม่รองรับการส่งข้อมูลแบบ Broadcast

Query: Query Message จะระบุจุดเริ่มต้นของ Input และปริมาณของ Input ที่จะอ่าน ซึ่งจะเริ่มต้นที่ที่อยู่ตำแหน่งที่ 0 โดย Input ตำแหน่งที่ 1-16 จะเป็นที่อยู่ที่เป็นตำแหน่งที่ 0-15

จากรูปที่ 2.37 แสดงตัวอย่างของการร้องขอการอ่าน Input 10197-10218 จาก Slave Device

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	02
Starting Address Hi	00
Starting Address Lo	C4
No. of Points Hi	00
No. of Points Lo	16
Error Check (LRC or CRC)	—

รูปที่ 2.37 การอ่านสถานะของ input Status-Query

Response: สถานะ Input ใน Response Message เป็นการบรรจุ Input 1 ตัวต่อ Data Field 1บิต ซึ่งจะเป็นการบ่งบอกว่าสถานะของบิตโดยบิตที่เป็น 1 จะมีสถานะเป็นOn และ 0 จะมีสถานะเป็น Off

โดย LSB ของข้อมูลไบต์แรกจะบรรจุที่อยู่ของ Input ไว้ใน Query และ Input อื่น ๆ ก็จะตามมาไปจนถึงคำสั่งที่เป็นจุดสิ้นสุดของไบต์ข้อมูล และจาก Low Order ไปจนถึง High Order ใน Subsequent Byte

ถ้าหากเรากลับปริมาณ Iutput โดยกลับให้ไม่เป็น Multipoint of Eight และให้บิตที่ยังเหลืออยู่ในไบต์ข้อมูลชุดสุดท้ายถูกอัดไว้กับที่อยู่ตำแหน่งที่ 0 (จนกระทั่งถึง High Order ซึ่งเป็นจุดสิ้นสุดของไบต์) โดย Byte Count Field จะบรรจุปริมาณของไบต์ข้อมูลที่ดำเนินการสำเร็จแล้ว

จากรูปที่ 2.38 แสดงตัวอย่างของการส่งกลับไปยัง Query บน Opposite Page

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	02
Byte Count	03
Data (Inputs 10204–10197)	AC
Data (Inputs 10212–10205)	DB
Data (Inputs 10218–10213)	35
Error Check (LRC or CRC)	—

รูปที่ 2.38 การอ่านสถานะของ Input Status-Response

สถานะของ Input ตำแหน่งที่ 10204-10197 เป็นการแสดงค่าไบต์ข้อมูล โดยค่าที่แสดงคือ AC ซึ่งเป็นเลขฐานสิบหก และเมื่อแปลงเป็นเลขฐานสองจะได้ 1010 1100ซึ่ง Input ตำแหน่งที่ 10204จะเป็น MSB ของไบต์นี้และตำแหน่งที่ 10197 จะขยับจากด้านซ้ายไปขวา โดยสถานะของ Input ที่ 10204 ไปจนถึงตำแหน่งที่ 10197 จะมีสถานะเป็น ON-OFF-ON-OFF-ON-ON-OFF-OFF

สถานะของ Input ตำแหน่งที่ 10218-10213 เป็นการแสดงค่าไบต์ข้อมูล โดยค่าที่แสดงคือ 35 ซึ่งเป็นเลขฐานสิบหก และเมื่อแปลงเป็นเลขฐานสองจะได้ 0011 0101 ซึ่ง Input ตำแหน่งที่ 10218 เป็นตำแหน่งที่ 3 นับจากด้านซ้ายของไบต์ข้อมูล และ Input ตำแหน่งที่ 10213 เป็น LSB โดยสถานะของ Inputที่ 10218 ไปจนถึงตำแหน่งที่ 10213 จะมีสถานะเป็น ON-ON-OFF-ON-OFF-ON

2.3.16.3 Analog output holding register หรือ Read Holding Registers (0x03)

Analog Outputs Holding Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสองของ Holding Register ใน Slave แต่จะไม่รองรับการส่งข้อมูลประเภท Broadcast

Analog Outputs Holding Register ถูกใช้งานกับค่า Analog Output (AO) ใน Fieldหรือในการตั้งค่าข้อมูลของ Slave ข้อมูลของ Holding Register จะมีความยาวว 16 บิต ซึ่งมีความสามารถทั้งการเขียนและการอ่านข้อมูล จะใช้ Address ตำแหน่งที่ 40001-49999 โดยข้อมูลชนิด Floating หรือ Double Floating จะเป็นตัวควบคุมเมื่อมีการกำหนด Address

Query: Query Message จะบรรจุจุดเริ่มต้นของ Register และปริมาณของ Register ที่จะทำการอ่าน โดยที่อยู่เริ่มต้นของ Register คือตำแหน่งที่ 0 จะทำให้ Register ที่ 1-16 จะถูกเก็บไว้ในที่อยู่ตำแหน่งที่ 0-15

จากรูปที่ 2.39 แสดงตัวอย่างของการร้องขอการอ่าน Register ที่ 40108-40110จาก Slave Device ตัวที่ 17

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	03
Starting Address Hi	00
Starting Address Lo	6B
No. of Points Hi	00
No. of Points Lo	03
Error Check (LRC or CRC)	—

รูปที่ 2.39 การอ่านสถานะของ Holding Register-Query

Response: Register Data ใน Response Message เป็นการบรรจุไบต์ข้อมูล 2 ไบต์ต่อ 1 Register กับรายละเอียดที่เป็นเลขฐานสองของข้อมูล โดยต้องจัดให้ชิดขอบด้านซ้ายของไบต์ดังกล่าวด้วยและสำหรับ Register นี้ เนื้อหาของไบต์แรกจะเป็น High Order Bits และไบต์ที่สองจะเป็น Low Order Bits

ข้อมูลจะถูกสแกนใน Slave ที่ Register ตำแหน่งที่ 125 โดยการสแกนสำหรับ 984-X0X Controller (984-685, etc) และ Register ตำแหน่งที่ 32 สำหรับการสแกน Controller ทั้งหมด การตอบกลับของข้อมูลจะถูกส่งกลับมาเมื่อมีการรวบรวมข้อมูลเรียบร้อยแล้ว

จากรูปที่ 2.40 แสดงตัวอย่างของการตอบกลับไปยัง Query บน Opposite Page

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	03
Byte Count	06
Data Hi (Register 40108)	02
Data Lo (Register 40108)	2B
Data Hi (Register 40109)	00
Data Lo (Register 40109)	00
Data Hi (Register 40110)	00
Data Lo (Register 40110)	64
Error Check (LRC or CRC)	—

รูปที่ 2.40 การอ่านสถานะของ Holding Register-Response

รายละเอียดของ Register ตำแหน่งที่ 40108 เป็นการแสดงค่าไบต์ข้อมูล 2 ไบต์ โดยค่าที่แสดงคือ 02 และ 2B ซึ่งเป็นเลขฐานสิบหกหรือ 555ในระบบเลขฐานสิบ และรายละเอียดของ Register ตำแหน่งที่ 40109-40110 จะเป็น 00 00 และ 00 64 ในระบบเลขฐานสิบหก หรือ 0 และ 1 ในระบบเลขฐานสิบ

2.3.16.4 Analog Input Register หรือ Read Input Register (0x04)

Analog Input Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสองของ Input Register ใน Slave แต่จะไม่รองรับการส่งข้อมูลประเภท Broadcast

Analog Iutputs Register ถูกใช้งานกับค่า Analog Iutput (AO) ใน Fieldหรือในการตั้งค่าข้อมูลของ Slaveข้อมูลของ Input Register จะมีควมยาวว 16 บิต โดยมีความสามารถในการอ่านได้เพียงอย่างเดียวเท่านั้น จะใช้ Address ตำแหนงที่ 30001-39999 โดยข้อมูลชนิด Floating หรือ Double Floating จะเป็นตัวควบคุมเมื่อมีการกำหนด Address

Query: Query Message จะบรรจุจุดเริ่มต้นของ Register และปริมาณของ Register ที่จะทำการอ่านโดยที่อยู่เริ่มต้นของ Register คือตำแหน่งที่ 0 จะทำให้ Register ที่ 1-16 จะถูกเก็บไว้ในที่อยู่ตำแหน่งที่ 0-15

จากรูปที่ 2.41 แสดงตัวอย่างของการร้องขอการอ่าน Register ที่ 3009 จาก Slave Device ตัวที่ 17

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	04
Starting Address Hi	00
Starting Address Lo	08
No. of Points Hi	00
No. of Points Lo	01
Error Check (LRC or CRC)	—

รูปที่ 2.41 การอ่านสถานะของ Input Register-Query

Response: Register Data ใน Response Message เป็นการบรรจุไบต์ข้อมูล 2 ไบต์ต่อ 1 Registerกับรายละเอียดที่เป็นเลขฐานสองของข้อมูล โดยต้องจัดให้ชิดขอบด้านขวาของไบต์ดังกล่าวด้วย และสำหรับ Register ตัวอื่น ๆ รายละเอียดของไบต์แรกจะเป็น High Order Bits และไบต์ที่สองจะเป็น Low Order Bits

ข้อมูลจะถูกสแกนใน Slave ที่ Register ตำแหน่งที่ 125 โดยการสแกนสำหรับ 984-X0X Controller (984-685, etc.) และ Register ตำแหน่งที่ 32 สำหรับการสแกน Controller ทั้งหมด การตอบกลับของข้อมูลจะถูกส่งกลับมาเมื่อมีการรวบรวมข้อมูลเรียบร้อยแล้ว

จากรูปที่ 2.42 แสดงตัวอย่างของการตอบกลับไปยัง Query บน Opposite Page

RESPONSE

Field Name	Example (Hex)
Slave Address	11
Function	04
Byte Count	02
Data Hi (Register 30009)	00
Data Lo (Register 30009)	0A
Error Check (LRC or CRC)	—

รูปที่ 2.42 การอ่านสถานะของ Input Register-Response

รายละเอียดของ Register ตำแหน่งที่ 30009 เป็นการแสดงค่าไบต์ข้อมูล 2 ไบต์ โดยค่าที่แสดงคือ 00 และ 0A ซึ่งเป็นเลขฐานสิบหก หรือ 10 ในระบบเลขฐานสิบ

2.3.17 ข้อดีและข้อจำกัดของ Modbus Protocol

2.3.17.1 ข้อดีของ Modbus Protocol

1). เป็นโปรโตคอลที่ง่ายต่อการเรียนรู้และพัฒนา

2). มีการใช้งานอย่างแพร่หลายเช่น ระบบ Process Control, Power Meter, Chillers, Boilers, ควบคุมการจ่ายกระแสไฟฟ้า เป็นต้น

3). การรวมกลุ่มของผู้ใช้งานและผู้พัฒนาทำให้อุปกรณ์ที่ผลิตจากหลายผู้ผลิตสามารถใช้งานร่วมกันได้

2.3.17.2 ข้อจำกัดของ Modbus Protocal

1). เป็นโปรโตคอลที่มีการส่งข้อมูลง่าย ๆ ไม่ซับซ้อนอาจทำให้เกิดข้อจำกัดในการส่งข้อมูลได้

2). ไม่ใช่โปรแกรมระดับสูง Standardization

3). มีข้อมูลการใช้งานที่จำกัด

4). ผู้ขายอาจจะไม่สามารถกำหนดความสามารถของ Modbus ที่แน่นอนได้

2.3.18 มาตรฐานการเชื่อมต่อบน Modbus Protocol

มาตรฐานการเชื่อมต่อบน Modbus Protocol เป็นการอ้างอิงถึงการเชื่อมต่อระหว่าง Modbus Protocol และ Terminal เพื่อทำการส่งผ่านข้อมูลหากัน โดยการใช้สายเชื่อมต่อหรือ Serial Port เป็นตัวกลางในการส่งผ่านข้อมูล

Serial Port มีอยู่หลายชนิดด้วยกัน แต่ที่นิยมใช้บน Modbus Protocol จะมีอยู่ 2 ชนิดคือ

1). EIA-232 หรือ RS232

EIA-232 เป็นมาตรฐานการเชื่อมต่อที่ถูกพัฒนาขึ้นในสหรัฐอเมริกาในปี 1969 เพื่อนิยามรายละเอียดเกี่ยวกับสัญญาณไฟฟ้าและการเชื่อมต่อทางกายภาพระหว่างอุปกรณ์ต้นทางและปลายทาง ซึ่งทำการเปลี่ยนข้อมูลให้เป็นข้อมูลไบนารีแบบอนุกรม และยังเปิดช่องให้ผู้ออกแบบฮาร์ดแวร์และโปรโตคอลสามารถพัฒนาผลิตภัณฑ์ได้อย่างยืดหยุ่น แต่ EIA-232 ยังมีจุดอ่อนอยู่จึงมีการพัฒนามาตรฐานอื่น ๆ ตามมาในภายหลัง เพื่อให้การทำงานมีประสิทธิภาพมากยิ่งขึ้น

2). EIA-485 หรือ RS485

EIA- 485 เป็นระบบสื่อสารแบบสมดุลซึ่งใช้ระดับสัญญาณระดับเดียวกับมาตรฐาน EIA-422 แต่เพิ่มอัตราการส่งข้อมูล และมีจำนวนตัวรับส่งในสื่อเดียวกันสามารถมีถึง 32 ตัวตามที่มาตรฐานกำหนด ซึ่งเมื่อต่อ RS-485 2 ชุดจะเรียกอีกอย่างว่า RS-422 นั่นเอง

2.4 คอนโทรลเลอร์ TMS320F2808

2.4.1 ลักษณะเฉพาะของ TMS320F2808

1). High-Performance Static CMOS Technology

- 100 MHz (10nS. Cycle Time)
- 60 MHz (16.67 nS Cycle Time)
- Low-Power (1.8 V Core, 3.3 V I/O)

2). JTAG Boundary Scan Support

3). High-Performance 32-Bit CPU

-16x16 and 32x32 MAC Operations

-16x16 Dual MAC

- Harvard Bus Architecture
- Atomic Operation
- Fast Interrupt Response and Processing
- Unified Memory Programming Model
- Code-Efficient (in C/C++ and Assembly)

4). On Chip Memory

- F2808:128 Kx16 Flash, 18 K-16 SARAM
- 1Kx16 OTP ROM (Flash Device Only)

5). Boot ROM (4 Kx16)

- With Software Boot Modes (via SCI, SPI, CAN, I2C, and Parrallel I/O)
- Standard Math Tables

6). Clock and System Control

- Dynamic PLL Ratio Changes Supported
- On-Chip Oscillator
- Watchdog Timer Module

7). Any GPIO A Pin Can Be Connected to One of the Three External Core Interrupt

8). Peripheral Interrupt Expansion (PIE) Block That Supports All 43 Peripheral Interrupt

9). Endiannes: Little Endian

10). 128-Bit Security Key/Lock

- Protects Flash/OTP/L0/L1 Blocks
- Prevent Firmware Reverse Engineering

11). 32-Bit CPU-Timers

12). Enhanced Control Peripherals

- Up to 16 PWM Outputs
- Up to 16 HRPWM Outputs With 150 ps MEP Resolution
- Up to Four Capture Inputs
- Up to Two Quadrature Encoder Interface
- Up to Six 32-Bit/Six 16-Bit Timers

13). Serial Port Peripheral

- Up to 4 SPI Modules
- Up to 2 SCI (UART) Modules
- Up to 2 CAN Modules
- One Inter-Integrated-Circuit (I2C) Bus

14). 12-Bit ADC, 16 Chanels

- 2x8 Chanel Input Multiplexer
- Two Sample-and-Hold
- Single/Simultaneous Conversions
- Fast Convertion Rate: 160 nS - 6.25 MSPS (280X)
- Internal or External Reference

15). Up to 35 Individually Programmable, Multiplexed GPIO Pins With Input Filtering

16). Advanced Emulation Features

- Analysis and Breakpoint Function
- Real-Time Debug via Hardware

17). Development Support Includes

- ANSI C/C++ Complier/Assembly, Linker
- Code Composer Studio IDE
- DSP/BIOS
- Digital Motor Control and Digital Power Software Libraries

18). Low-Power Mode and Power Saving

- IDLE, STANDBY, HALT Modes Supported
- Disagble Individual Peripheral Clocks

ตามมาตรฐาน IEEE Standard 1149-1990 Standard Test Access Port and Boundary Scan Architecture

2.4.2 เริ่มต้นใช้งาน DSP2808

ขั้นตอนแรกของการเริ่มต้นใช้งาน DSP2808

ขั้นที่ 1 Acquire the Appropriate development tools (พัฒนาการของเครื่องมือที่เหมาะสม)

วิธีที่รวดเร็วสำหรับการเริ่มใช้งาน DSP2808 เป็นการใช้ eZdsp kit ในการเริ่มต้นพัฒนาซึ่งใน 1 แพ็คเกต ประกอบด้วย

- On-Board JTAG emulation via USB or Parallel port (บอร์ด JTAG emulation เชื่อมต่อโดย USB หรือ พอร์ตแบบขนาน)
- Appropriate emulation driver (emulation driver ที่เหมาะสม)
- Code Composer Studio IDE for eZdsp

ขั้นแรกเมื่อมีตระกูลของ Device พร้อมแล้ว และต้องการเริ่มใช้งานกับฮาร์ดแวร์ จะต้องมีการซื้อโปรแกรม Code Composer Studio IDE เพื่อใช้แยกการพัฒนาซอฟแวร์ และจะใช้เครื่อง JTAG emulation ในการเริ่มต้นการทำงาน

ขั้นที่ 2 Download Starter Software (โหลดซอฟแวร์เริ่มต้น)

เพื่อลดความซับซ้อนของโปรแกรมมิ่งสำหรับ DSP2808 แนะนำให้ผู้ใช้งานทำการดาวน์โหลดและใช้ Header Files ของ C/C++ และตัวอย่าง เพื่อการพัฒนาซอฟแวร์และอุปกรณ์ต่อพ่วงต่าง ๆ

หลังจากที่ดาวน์โหลด Header files ที่เหมาะสมแล้ว จะต้องอ้างถึงทรัพยการที่ตามมาทีละขั้นตอน โดยแนะนำว่าจะทำการรันอุปกรณ์ต่อพ่วงอย่างไร และใช้ Header Files อย่างไรในโครงสร้างของซอฟแวร์

- อ่านคู่มือการใช้งานเพื่อมี่จะเริ่มทำงานแอพพลิเคชั่นแรก

- เขียนโปรแกรมของ DSP2808 และอุปกรณ์ต่อพ่วงใน C/C++

ขั้นที่ 3 Download Flash programming software (โหลดซอฟแวร์สำหรับการเขียนโปรแกรม Flash)

DSP2808 ประกอบด้วย on-chip flash memory และเครื่องมือที่จะยอมให้มีการโปรแกรมแฟรชกับซอฟแวร์

- Flash Tools: C2808 Flash Tools

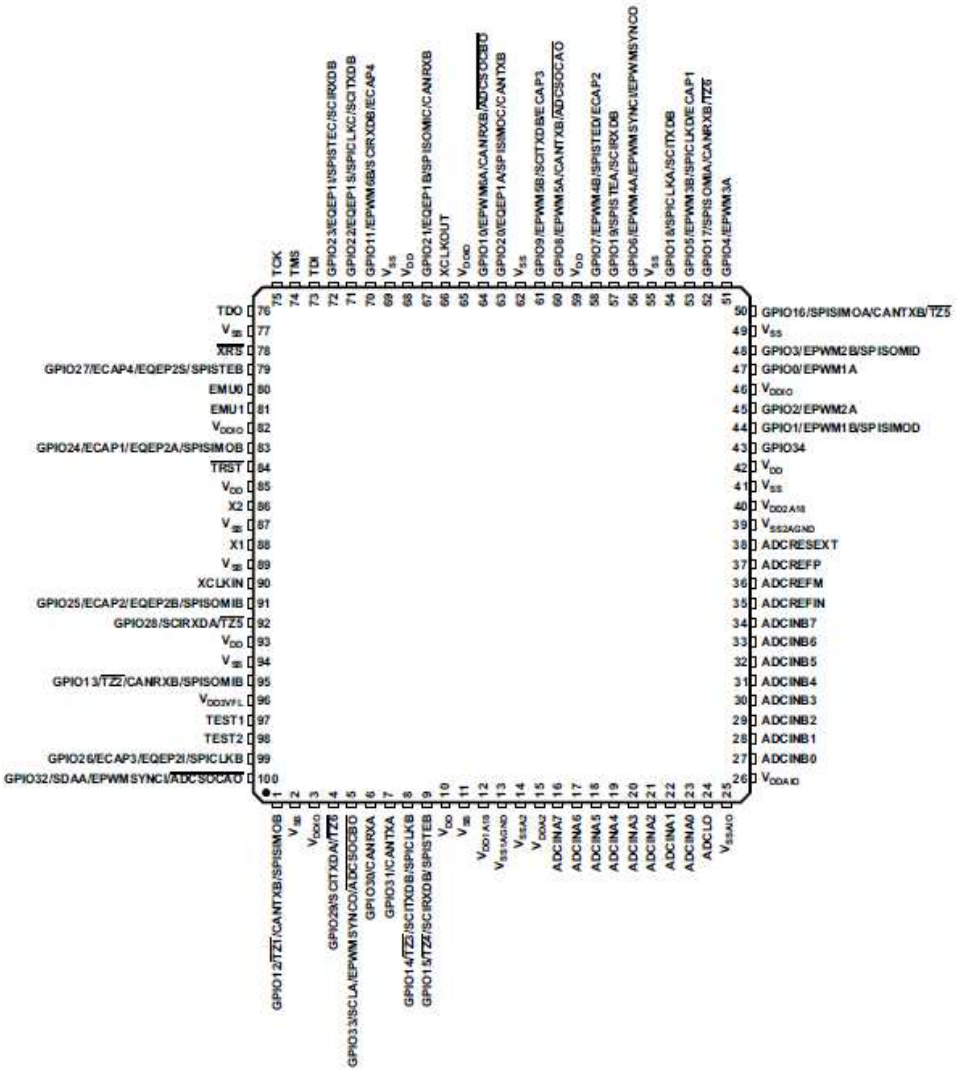
- DSP2808 Flash Programming Solution

- รันแอพพลิเคชั่นจากเมมโมรี่ภายในบน DSP2808

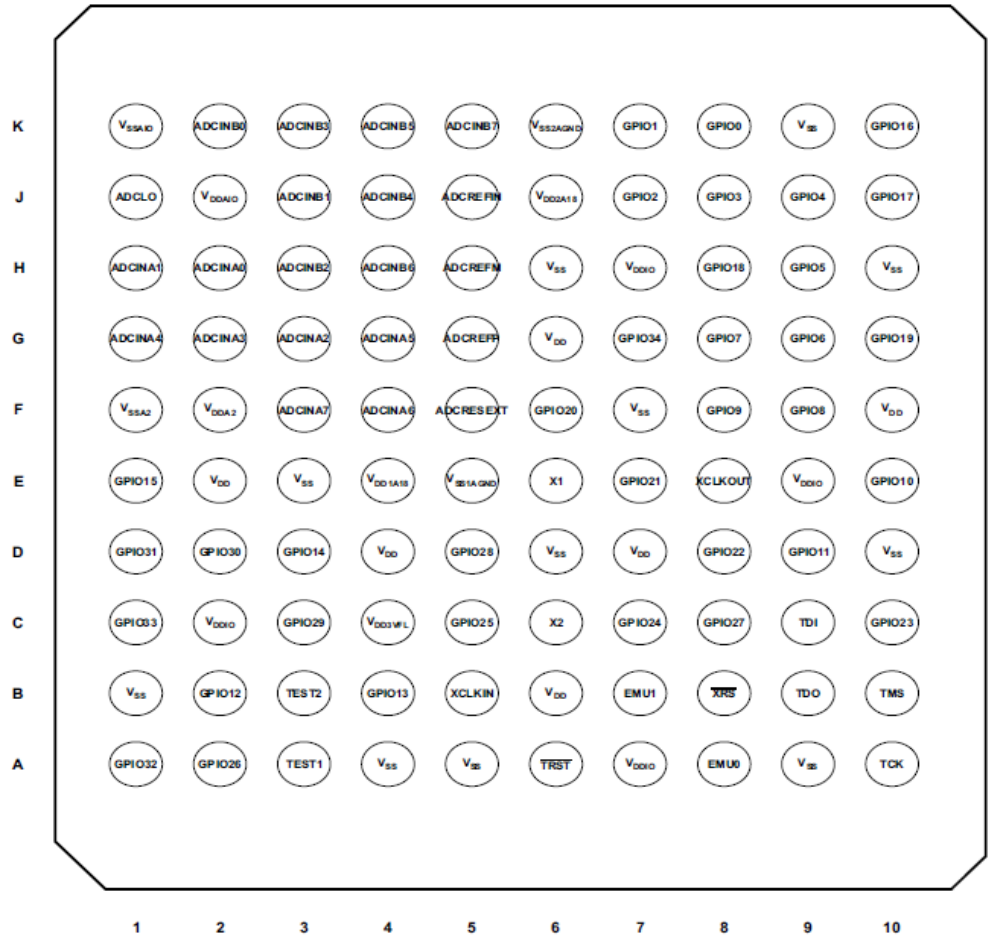
ขั้นที่ 4 Move on to more advanced topics (ศึกษาในหัวข้อที่ยากขึ้น)

2.4.3 Pin Assignment

TMS320F2808 หรือ DSP2808 มีจำนวนพินทั้งหมด 100 พิน ดังนี้

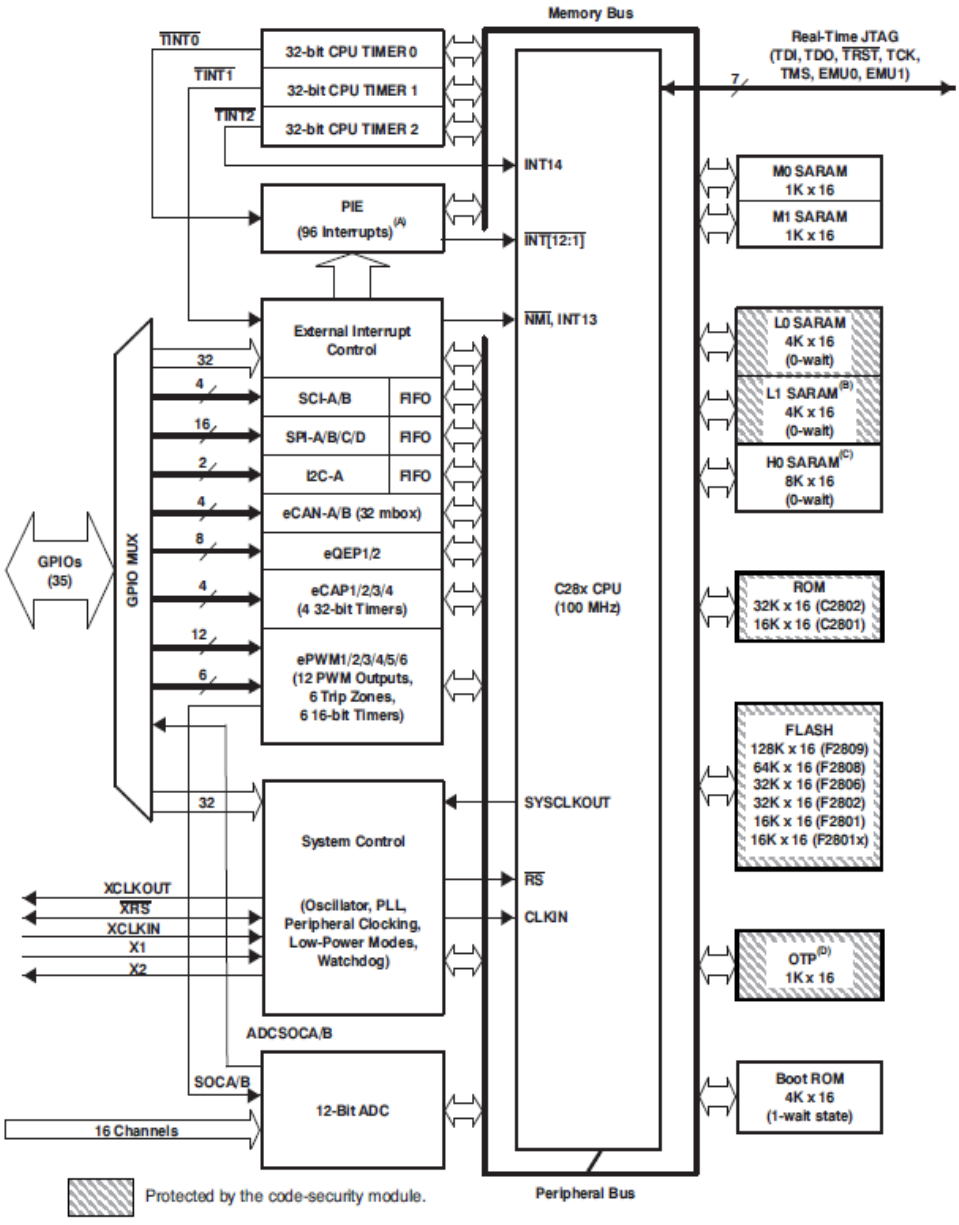


รูปที่ 2.43 คอนโทรลเลอร์ TMS320F2808 (มุมบน)



รูปที่ 2.44 คอนโทรลเลอร์ TMS320F2808 (มุมล่าง)

2.4.4 Functional Overview



รูปที่ 2.45 แผนภาพ Function block

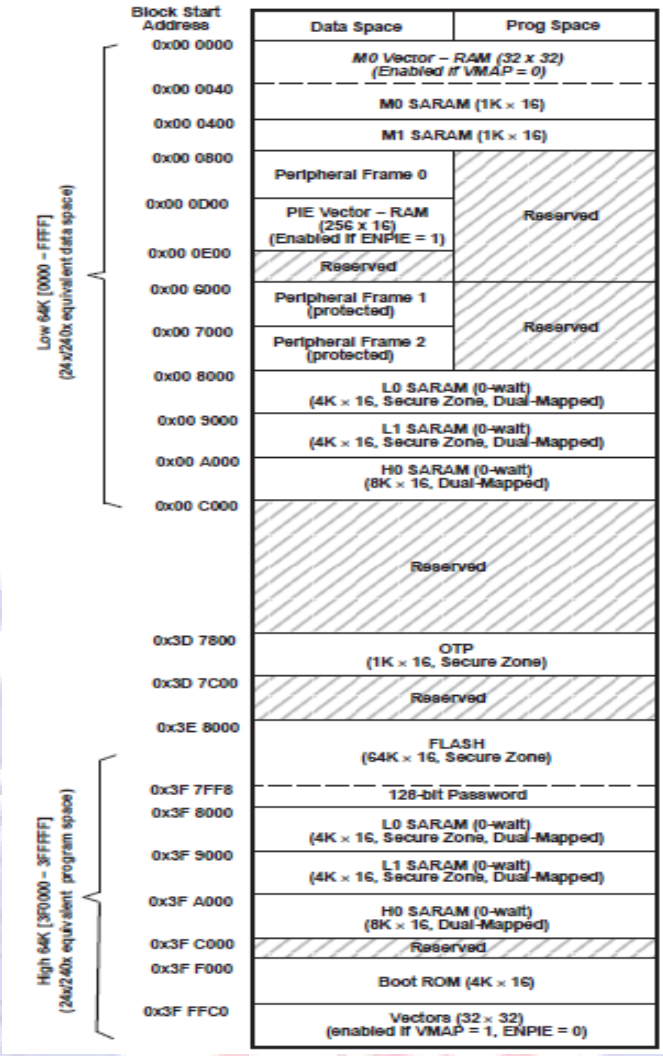
A. 43 of the possible 96 interrupt are used on the devices

B. Not available in F2802, F2801, C2808 and C2801

C. Not available in F2806, F2802, F2801, C2802 and C2801

D. The 1K *16 OTP has been replaces with 1K*16 ROM for C280X device

2.4.5 Memory Map



รูปที่ 2.46 ที่อยู่ของหน่วยความจำบน F2808

A. Memory blocks are not to scale

B. Peripheral Frame 0, Peripheral Frame 1 and Peripheral Frame 2 memory map are restriced to data memory only.

User program cannot access these memory maps in program space.

C. Protected means the order of Write Followed by Read Operations is preserved rather than the pipeline order.

D. Certain memory ranges are EALLOW protected against write after configuration

ตารางที่ 2.7 ที่อยู่ของ Flash Sectors ใน F2808

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3E 8000 – 0x3E BFFF	Sector D (16K x 16)
0x3E C000 – 0x3E FFFF	Sector C (16K x 16)
0x3F 0000 – 0x3F 3FFF	Sector B (16K x 16)
0x3F 4000 – 0x3F 7F7F	Sector A (16K x 16)
0x3F 7F80 – 0x3F 7FF5	Program to 0x0000 when using the Code Security Module
0x3F 7FF6 – 0x3F 7FF7	Boot-to-Flash Entry Point (program branch instruction here)
0x3F 7FF8 – 0x3F 7FFF	Security Password (128-Bit) (Do not program to all zeros)

บทที่ 3
แผนการปฏิบัติงานและขั้นตอนการดำเนินงาน

3.1 แผนการปฏิบัติงาน

การเข้าฝึกปฏิบัติงานของนักศึกษาในบริษัท โนเวมเอนจิเนียริ่ง จำกัด ได้รับมอบหมายให้ทำการศึกษาการติดต่อสื่อสารแบบ Modbus Protocol ซึ่งศึกษาร่วมกับการทำงานบนอุปกรณ์อิเล็กทรอนิกส์ โดยการใช้โปรแกรมคำสั่งในการควบคุมคอนโทรลเลอร์ คอนโทรลเลอร์ที่ใช้คือ DSP2808 ใช้ภาษาซีในการเขียนโปรแกรมดังกล่าว และเชื่อมต่ออุปกรณ์ผ่าน RS-232 ซึ่งเป็น Serial Port สำหรับการเชื่อมต่ออุปกรณ์ Device และ Terminal ดังนั้นจึงจำเป็นต้องมีการวางแผนการเรียนรู้สิ่งต่าง ๆ ที่เกี่ยวข้องไว้อย่างรอบคอบ เพื่อป้องกันความผิดพลาดของข้อมูล ซึ่งอาจนำไปสู่ความเข้าใจที่ผิดในการประกอบหน้าที่และอาจทำให้เกิดข้อผิดพลาดในการปฏิบัติงานได้

เนื่องจากการเข้าฝึกปฏิบัติงานในครั้งนี้มีระยะเวลาที่จำกัด จึงจำเป็นต้องมีการวางแผนการปฏิบัติงานไว้เป็นอย่างดี เพื่อให้สามารถทำการศึกษาและปฏิบัติหน้าที่ให้เป็นไปตามขั้นตอนและเสร็จสิ้นตามกำหนดการที่ได้วางแผนไว้

ตารางที่ 3.1 แผนการปฏิบัติงาน

หัวข้องาน	เดือนที่ 1				เดือนที่ 2				เดือนที่ 3				เดือนที่ 4			
ศึกษาการทำงานละการใช้งานของ TMS320F2808 และอื่นๆ																
ปรึกษาที่ปรึกษาเกี่ยวกับหัวข้อโครงงาน																
ศึกษารายละเอียดเกี่ยวกับหัวข้อโครงงาน เรื่องโปรโตคอล Modbus																
ฝึกใช้งานโปรแกรม Code Composer Studio V.3.3																
ทดลองใช้งานโปรแกรมด้วยโปรแกรมคำสั่งSCI EchoBack																
ฝึกการเขียนโปรแกรมโดยใช้ภาษาซี																
ค้นคว้าข้อมูลเกี่ยวกับโปรโตคอล Modbus																
เริ่มทำรายงานเกี่ยวกับโปรโตคอล Modbus และการควบคุมด้วย TMS320F2808																
ตรวจสอบความถูกต้องเรียบร้อยของโครงงาน																
เตรียมแผนการนำเสนอโครงงาน																
ตรวจสอบความผิดพลาดพร้อมนำเสนอโครงงาน																
ปฏิบัติงานตามที่สถานประกอบการมอบหมาย																

3.2 รายละเอียดโครงงานที่ได้รับมอบหมาย

การฝึกปฏิบัติในครั้งนี้ นักศึกษาได้รับมอบหมายจากทางสถานประกอบการให้ทำการศึกษาและทำความเข้าใจในเรื่องของ Modbus Protocol ซึ่งเป็นรูปแบบในการติดต่อสื่อสารบนอุปกรณ์อิเล็กทรอนิกส์ประเภท PLC หรือเครื่องควบคุมเชิงตรรกะที่สามารถโปรแกรมได้ โปรโตคอลดังกล่าวจะทำงานภายใต้การควบคุมของคอนโทรลเลอร์ผ่านทางพอร์ตที่ใช้ในการติดต่อสื่อสาร โดยรายละเอียดของโครงงานมีดังต่อไปนี้

3.2.1 การสื่อสารข้อมูลในงานอุตสาหกรรม

ศึกษาการสื่อสารข้อมูลในงานอุตสาหกรรมต่าง ๆ เช่น รูปแบบการติดต่อสื่อสาร และมาตรฐานสำหรับการติดต่อสื่อสารต่าง ๆ เป็นต้น

3.2.2 ความรู้พื้นฐานเกี่ยวกับอินเวอร์เตอร์

เนื่องจากสถานประกอบการเป็นผู้ผลิตและออกแบบเครื่องอินเวอร์เตอร์ จึงต้องมีการศึกษาหลักการทำงานและโครงสร้างต่าง ๆ ของเครื่องอินเวอร์เตอร์ ทั้งนี้เพื่อนำไปเป็นความรู้พื้นฐานในการวางโครงสร้างระบบการติดต่อสื่อสารของเครื่องอินเวอร์เตอร์

3.2.3 ระบบการสื่อสารแบบ Modbus Protocol

ศึกษาเกี่ยวกับการสื่อสารแบบ Modbus Protocol อย่างละเอียด

3.2.4 อุปกรณ์สำหรับการเชื่อมต่อระหว่าง Device และ Terminal บนมาตรฐานการติดต่อสื่อสาร

ศึกษาข้อมูลเกี่ยวกับสายสัญญาณที่ใช้ในการเชื่อมต่อทางกายภาพระหว่างอุปกรณ์ต้นทางและอุปกรณ์ปลายทาง เพื่อการสื่อสารข้อมูล

3.2.6 การเขียนโปรแกรมคำสั่งบนคอนโทรลเลอร์ชนิด TMS320F2808 หรือ DSP2808

ฝึกการเขียนโปรแกรมควบคุมคอนโทรลเลอร์ที่ใช้ประกอบการทำงานชนิด DSP2808

3.3 ขั้นตอนการดำเนินงานในการทำโครงงาน

3.3.1 ศึกษาความรู้ขั้นพื้นฐาน

การศึกษาความรู้ขั้นพื้นฐานเกี่ยวกับข้อมูลในทำโครงงานนั้นมีความจำเป็นเป็นอย่างมาก ไม่ว่าจะเป็นข้อมูลที่เกี่ยวกับคอนโทรลเลอร์ที่ใช้ในการทำโครงงาน รวมไปถึงข้อมูลของโปรแกรมที่ใช้เขียนคำสั่งและอุปกรณ์ฮาร์ดแวร์ต่าง ๆ และสิ่งสำคัญที่สุดคือความรู้ขั้นพื้นฐานของการติดต่อสื่อสารข้อมูลในงานอุตสาหกรรม

3.3.2 ศึกษาระบบการสื่อสารแบบต่างๆ

ศึกษาระบบการติดต่อสื่อสารแต่ละชนิดที่ใช้ในงานอุตสาหกรรม รวมไปถึงข้อดีข้อเสียของแต่ละโปรโตคอล เพื่อนำมาเปรียบเทียบคุณสมบัติของโปรโตคอลแต่ละชนิด และนำความรู้ที่ได้มาใช้ในการวิเคราะห์ข้อมูลต่อไป

3.3.3 วิเคราะห์ระบบการติดต่อสื่อสาร

วิเคราะห์โปรโตคอลเพื่อการสื่อสารรูปแบบต่าง ๆ ว่ามีความแตกต่างกันอย่างไร รวมไปถึงจุดเด่น-จุดด้อยของแต่ละโปรโตคอล และชี้แจงถึงเหตุผลที่เลือกใช้ Modbus Protocol เป็นโปรโตคอลเพื่อการติดต่อสื่อสารบนอุปกรณ์อิเล็กทรอนิกส์ที่เป็นการผลิตโดยสถานประกอบการ

3.3.4 ศึกษาระบบการสื่อสารแบบ Modbus Protocol

ศึกษารายละเอียดต่าง ๆ ของ Modbus Protocol ไม่ว่าจะเป็นเรื่องประวัติความเป็นมา ข้อมูลทั่วไปที่เกี่ยวกับ Modbus Protocol วัตถุประสงค์ในการใช้งาน ลักษณะการติดต่อสื่อสารและรูปแบบการติดต่อสื่อสารแบบต่าง ๆ บน Modbus Protocol อย่างละเอียด

3.3.5 ศึกษาการเชื่อมต่ออุปกรณ์ต้นทางและอุปกรณ์ปลายทาง

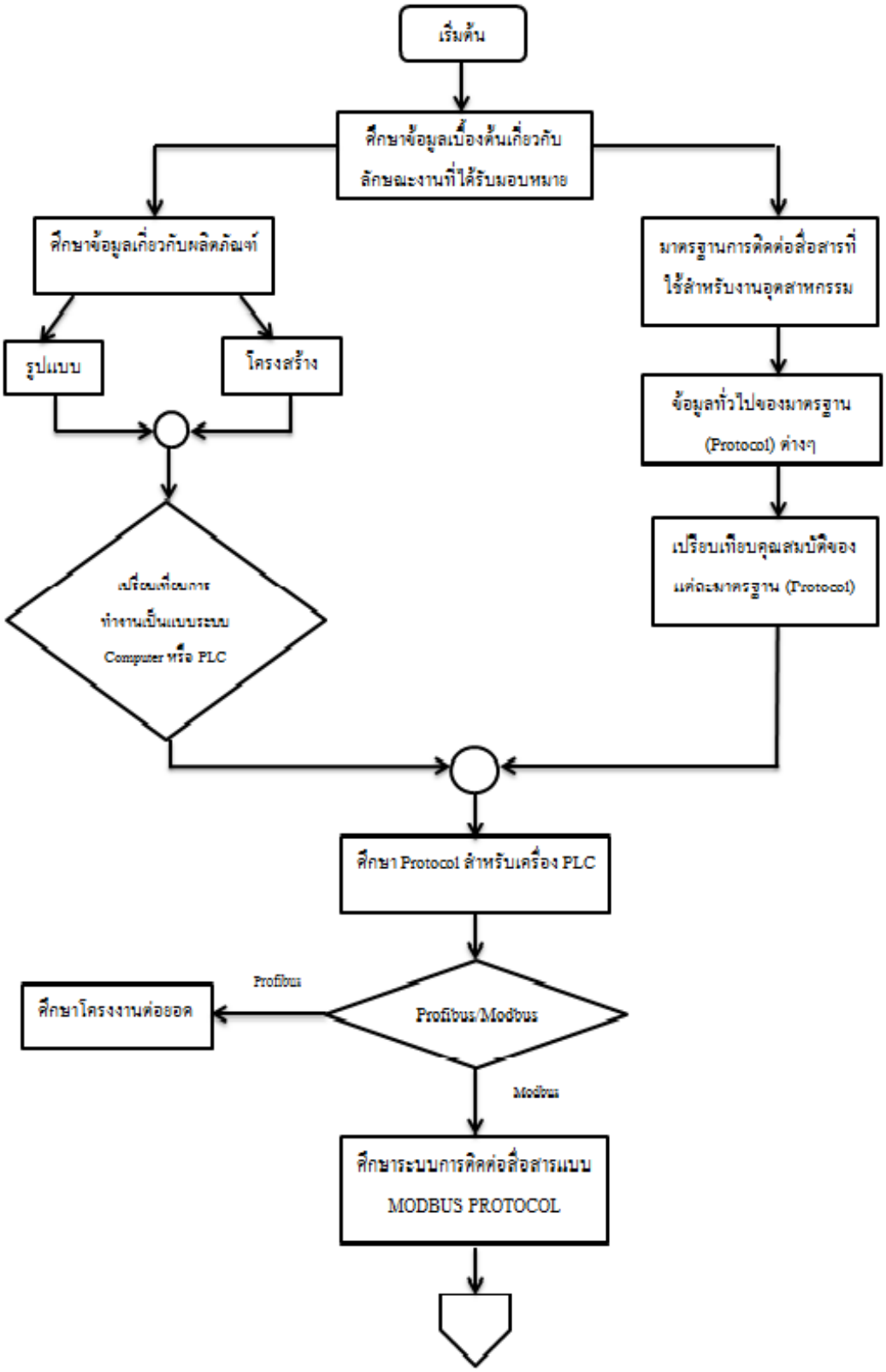
ในการรับและการส่งข้อมูลระหว่างอุปกรณ์ต้นทางและปลายทางจะมีอุปกรณ์สำหรับการเชื่อมต่อ หรือที่เรียกกันว่าสายสัญญาณ ดังนั้นการศึกษาข้อมูลในส่วนนี้เป็นการศึกษาถึงสายสัญญาณชนิดต่างๆที่ใช้ในการเชื่อมต่อ รวมไปถึงคุณสมบัติและข้อดีข้อเสียต่าง ๆ

3.3.6 ศึกษาคอนโทรลเลอร์ที่ใช้ในการเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์อิเล็กทรอนิกส์

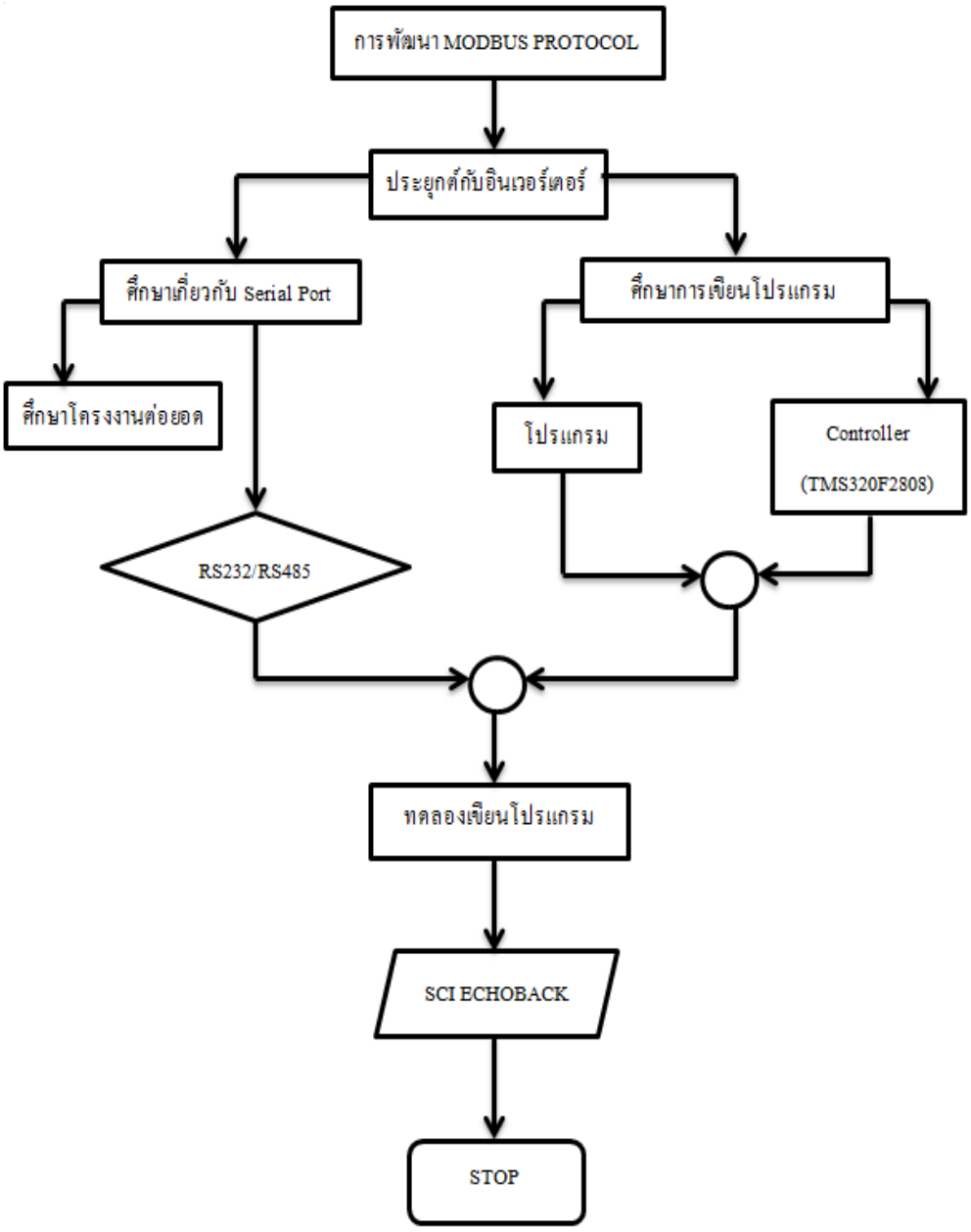
การทำงานของ Modbus Protocol จะมีการควบคุมคำสั่งต่างๆโดยคอนโทรลเลอร์ ซึ่งเป็นอุปกรณ์ที่ใช้ในการควบคุมการทำงานตามคำสั่งที่ผู้ใช้ต้องการ ดังนั้นจึงต้องศึกษาถึงข้อมูลทั่วไปเกี่ยวกับคอนโทรลเลอร์ที่ใช้ด้วย

3.3.7 ศึกษาการเขียนโปรแกรมควบคุมการทำงานของคอนโทรลเลอร์ชนิด DSP2808

เมื่อทราบถึงชนิดของคอนโทรลเลอร์แล้วคือ DSP2808 จำเป็นที่จะต้องมีการศึกษาการเขียนโปรแกรมคำสั่งที่ใช้ควบคุมการทำงานของคอนโทรลเลอร์ดังกล่าว ไม่ว่าจะเป็นคำสั่งเฉพาะหรือฟังก์ชันต่าง ๆ ที่ทำให้คอนโทรลเลอร์ทำงานได้ โดยการเขียนโปแกรมจะใช้รูปแบบของภาษาซี



รูปที่ 3.1 แผนผังการทำงาน



รูปที่ 3.2 แผนผังการทำงาน (ต่อ)

บทที่ 4
ผลการดำเนินงาน การวิเคราะห์และสรุปผลต่าง ๆ

4.1 ขั้นตอนและผลการดำเนินงาน

ในการดำเนินงานการทำโครงงานในครั้งนี้ ได้รับมอบหมายให้ทำการศึกษาเรื่องการติดต่อสื่อสารโดยการใช้ Modbus Protocol ร่วมกับการทำงานบนอุปกรณ์อิเล็กทรอนิกส์ ซึ่งในการทำโครงงานนั้นได้มีการวางแผนการทำงานไว้อย่างละเอียดและรอบคอบ ทั้งนี้เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้

ในการดำเนินการตามขั้นตอนในหัวข้อที่ 3.3 ว่าด้วยเรื่องขั้นตอนการดำเนินงานในการทำโครงงาน มีผลของการดำเนินงานดังต่อไปนี้

4.1.1 ความรู้ที่ได้รับจากการศึกษาการสื่อสารข้อมูลในงานอุตสาหกรรม

การสื่อสารข้อมูลเป็นการโอนถ่ายข้อมูลข่าวสารจากจุดหนึ่งไปยังอีกจุดหนึ่ง ซึ่งระบบสื่อสารข้อมูลทุกระบบต้องมีตัวส่งข้อมูลหรือทรานสมิตเตอร์ (Transmitter) เพื่อส่งข้อมูลข่าวสาร และตัวรับ (Receiver) เพื่อตอบรับข้อมูลข่าวสารนั้น และจะต้องเชื่อมต่ออุปกรณ์ทั้ง 2 เข้าด้วยกัน โดยการสื่อสารข้อมูลเป็นการส่งข้อมูลแบบดิจิตอลและอนาลอก ซึ่งการสื่อสารข้อมูลแบบดิจิตอลกลายเป็นรูปแบบหลักในปัจจุบัน ข้อมูลจะถูกส่งออกและรับด้วยค่าทางลอจิกคือ 0 และ 1 โดยค่าดังกล่าวเป็นค่าที่คอมพิวเตอร์สามารถเข้าใจได้ ส่วนการสื่อสารข้อมูลแบบอนาลอกนั้นจะนำไปใช้กับระบบโทรศัพท์พื้นฐานหรือวิทยุและโทรทัศน์

ในการสื่อสารข้อมูลมักจะมีตัวกลางที่ใช้สำหรับดำเนินการเพื่อให้การจัดการข้อมูลเป็นระเบียบและมีแบบแผน เราเรียกสิ่งนี้ว่าโปรโตคอล ซึ่งมาตรฐานของโปรโตคอลต่างๆได้พัฒนาขึ้นมาเพื่อการลดการผูกขาดของผู้ผลิตกับผลิตภัณฑ์ที่ขึ้นกับองค์กรใดองค์กรหนึ่งเพียงอย่างเดียว และสามารถทำให้อุปกรณ์ต่างๆทำงานร่วมกันได้ ปัจจุบันได้มีการกำหนดมาตรฐานของการสื่อสารข้อมูลต่างๆหลายมาตตรฐานด้วยกันซึ่งจะแบ่งออกเป็น 2 ประเภทคือ

4.1.1.1 มาตรฐานที่เกี่ยวข้องกับโปรโตคอล

1). Modbus Protocol

โปรโตคอล MODBUS เป็นโปรโตคอลเพื่อการรับส่งข้อมูลบน PLC ซึ่งสามารถมีจำนวนโหนดมากถึง 247 โหนดตามที่มาตรฐานกำหนดโดยใช้หลักการสื่อสารแบบ Master-Slave ซึ่ง

Modbus Protocol เป็นโปรโตคอลระบบเปิด หมายความว่าสามารถเป็นโปรโตคอลที่ไม่มีค่าใช้จ่ายเชื่อมต่อและสามารถพัฒนาได้ง่าย ซึ่งสามารถเลือกอุปกรณ์เชื่อมต่อได้อย่างอิสระ โดยสามารถเลือกโหมดในการทำงานได้สองโหมดคือ RTU Mode และ ASCII Mode

2). Modbus Plus Protocol

Modbus Plus ถูกสร้างโดยใช้เทคนิคการส่งข้อมูลแบบ Token Passing โดยออกแบบให้ใช้งานบนอุปกรณ์ Modicon PLC เท่านั้น ส่งผลทำให้ขาดการเปิดของโปรโตคอลหรือพูดได้อีกอย่างว่า MODBUS PLUS คือโปรโตคอลปิด ซึ่งมีน้อยอุปกรณ์ที่สนับสนุน MODBUS PLUS

3). IEC60870-5-103

นิยมใช้ในระบบสื่อสารข้อมูลของระบบป้องกันทางไฟฟ้าโดยเฉพาะในรีเลย์ป้องกัน IEC60870-5-103 ใช้ได้ดีเฉพาะระบบป้องกันเท่านั้นเพราะได้ถูกออกแบบมาเฉพาะงานดังกล่าว

4). DNP3 (Distributed Network Protocol 3)

DNP3 เป็นโปรโตคอลที่มีความยืดหยุ่นค่อนข้างสูงและมี 3 ระดับตามขนาดและประสิทธิภาพของตัวอุปกรณ์เช่น ระดับหนึ่งสำหรับดิจิตอลมิเตอร์ ระดับสองสำหรับรีเลย์ป้องกันและระดับสามสำหรับศูนย์สั่งการ DNP3 สามารถทำงานบน TCP/IP เรียกว่า DNP3 over TCP/IP ซึ่งข้อดีของ DNP3 คือสามารถเปลี่ยนแปลงรูปแบบข้อมูลได้ในระหว่างการใช้งานและสนับสนุนงานที่ต้องการหลายมาสเตอร์ได้ และสนับสนุนการทำงานแบบ Unsolicited Response หมายความว่าอุปกรณ์ประเภทสแลฟสามารถส่งข้อมูลได้เองโดยไม่ต้องรอคำถามจากมาสเตอร์

5). Data High Plus /DH485

โปรโตคอลนี้ถูกสร้างเพื่อเป็นแบ็คโบนของอุปกรณ์ PLC ของ Allen Bradley (ปัจจุบันเป็นส่วนหนึ่งของ Rockwell Automation)

6). HART

โปรโตคอล HART (Highway Addressable Route Transducer) คือโปรโตคอลที่ใช้สำหรับอุปกรณ์วัดค่าโดยสามารถทำงานบนค่าอนาล็อกที่ 4-20 mA

7). ASi

ASi เป็นโปรโตคอลที่ใช้ในการควบคุมระบบงานที่ไม่ซับซ้อน เพื่อนำมาแก้ไขปัญหาเรื่องการติดตั้งแบบเก่า (Traditional Cable Tree) ซึ่งจะใช้สายไฟฟ้าเพียงแค่ 2 เส้นเท่านั้น โดยสามารถป้อนสัญญาณควบคุม (Signal) และพลังงานไฟฟ้า (Power) ในเวลาเดียวกันได้ จึงสามารถลดจำนวนสายไฟได้มาก

8). CANBUS (Control Area Network)

จุดประสงค์แรกเริ่มของ CANBUS คือใช้ในอุตสาหกรรมรถยนต์ โดยเฉพาะระบบอิเล็กทรอนิกส์ภายในรถ เช่น แอร์ ประตู ระบบเบรก รวมทั้งกล่อง ECU (Electronic Control Unit) CANBUS ยังถูกนำมาใช้เป็น I/O Bus ใน RTU หลาย ๆ ยี่ห้อเช่น ABB หรือกระทั่งระบบอัตโนมัติจากประเทศจีนเช่น NARI

9). DEVICENET

เป็นโปรโตคอลระดับล่างที่เน้นในการส่งข้อมูลประเภท DI/DO โดยที่ DEVICENET ใช้ 3 เลเยอร์ของโมเดล OSI ตัว DEVICENET สามารถสนับสนุนจำนวนโหนดตั้งแต่ 64 โหนดจนถึง 2048 โหนด และถูกออกแบบให้สามารถส่งไฟเลี้ยงไปยังอุปกรณ์วัดค่าได้ด้วย

10). IEC60850-5-101

IEC60850-5-101 ถูกออกแบบสำหรับการสื่อสารระหว่าง SCADA และ RTU โดยเฉพาะระบบไฟฟ้า โครงสร้างระดับดาต้าลิงค์มีความคล้ายคลึงกับ IEC60850-5-103 ตัว IEC60850-5-101 สามารถทำงานบน TCP/IP เรียกอีกอย่างว่า IEC60850-5-104 IEC60850-5-101 มีข้อด้อยอย่างหนึ่งคือไม่สามารถมีจำนวนมาสเตอร์หลายมาสเตอร์ได้เนื่องจากไม่มีฟิลด์สำหรับระบุมาสเตอร์

11). PROFIBUS

PROFIBUS เป็นโปรโตคอลที่ใช้ควบคุมการทำงานของ PLC ใช้วิธีการสื่อสารข้อมูลแบบผสมผสานระหว่างวิธีโทเคนพาสซิงและมาสเตอร์สแลฟเพื่อให้ได้ประสิทธิภาพในการสื่อสารดีที่สุด โดยจะทำงานบนเชื่อมต่อแบบ EIA-485 เรียกว่า PROFIBUS DP และที่ทำงานบนการเชื่อมต่อแบบ IEC61158 เรียกว่า PROFIBUS PA

12). SPABUS

SPABUS ออกแบบมาเพื่อใช้ในระบบควบคุมและป้องกันในอุตสาหกรรมการผลิตและจำหน่ายไฟฟ้า เป็นโปรโตคอลที่ไม่ซับซ้อนทำงานโดยใช้อักขระ ASCII ทำให้เราสามารถอ่านแพ็คเก็ตบนเครื่องมือพื้นฐานได้โดยง่ายเช่นโปรแกรมไฮเปอร์เทอร์มินอล (Hyperterminal)

13). LONBUS

LONBUS ถูกนำมาใช้ในระบบอาคารอัตโนมัติ และจากนั้นได้นำมาประยุกต์ใช้ในอีกหลายระบบงานเช่น ระบบรถไฟฟ้า ระบบจำหน่ายไฟฟ้า จนได้รับเสนอเป็นมาตรฐานในชื่อ ANSI/CEA-709.1-B ตัว LONBUS ยังถูกสนับสนุนโดยซอฟต์แวร์ประเภทบำรุงรักษา เช่น IFS ของสวีเดนนั้นหมายความว่าซอฟต์แวร์สามารถดึงค่าจากอุปกรณ์หรือเครื่องจักรเพื่อวางแผนบำรุงรักษาได้โดยไม่ต้องใช้คนป้อนข้อมูล

14). Industrial Ethernet

Industrial Ethernet มีการพัฒนาที่เร็วมากและได้ก้าวข้ามปัญหาเกี่ยวกับความเชื่อถือได้ที่ต่ำไม่เพียงพอต่องานระบบอุตสาหกรรม หนึ่งในเหตุผลสำหรับความสำเร็จของ Industrial Ethernet คือการใช้งานที่ง่ายและราคาติดตั้งที่ไม่สูงเนื่องจากมีผลิตภัณฑ์ในตลาดเป็นจำนวนมาก

15). TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP คือชุดของโปรโตคอลหลาย ๆ ตัวที่มีคุณสมบัติเฉพาะตัวแตกต่างกันไปเช่น HTTP สำหรับการแสดงผลบนเว็บ FTP สำหรับการถ่ายโอนไฟล์ เป็นต้น TCP/IP ถือว่าเป็นโปรโตคอลที่มีค่าใช้จ่ายค่อนข้างต่ำและถูกสนับสนุนอย่างแพร่หลายในทุกระบบงาน อาจจะกล่าวได้ว่าการเกิดของ TCP/IP ทำให้หลาย ๆ โปรโตคอลได้ตายหรือหายไปจากระบบสื่อสารก็ว่าได้ ปัจจุบันโปรโตคอล TCP/IP ได้มีความสำคัญต่อชีวิตประจำวันของมนุษย์มากไม่ว่าจะเป็นอินเตอร์เน็ต โทรศัพท์ คอมพิวเตอร์ซึ่งเป็นอุปกรณ์หลักในการให้บริการเราเช่น ระบบธนาคาร

16). Radio (Wireless) Communication

การใช้ Radio ในงานอุตสาหกรรมเกี่ยวข้องกับการใช้โมเด็มในการส่งข้อมูล ซึ่งสามารถส่งบนดาต้าลิงค์ของโมเด็มไร้สายได้

17). IEC61850

IEC61850 ถือว่าเป็นโปรโตคอลตัวล่าสุดที่ออกแบบมาใช้ในระบบสถานีไฟฟ้า จุดมุ่งหมายคือมาแทนที่โปรโตคอลทุกตัวไม่ว่าจะเป็นระบบเวลาจริง (Real-time System) หรือแบบไคลเอนท์เซิร์ฟเวอร์ (Client-Server) การโอนหรือบันทึกค่าการติดตั้งจะใช้เทคโนโลยีทาง XML (Extension Mark-up Language)

4.1.1.2 มาตรฐานที่เกี่ยวข้องกับการอินเตอร์เฟส

1). RS-232

RS-232 หรืออีกชื่อ EIA-232 คือมาตรฐานการเชื่อมต่อที่ถูกพัฒนาขึ้นในสหรัฐอเมริกาในปี 1969 เพื่อนิยามรายละเอียดเกี่ยวกับสัญญาณไฟฟ้าและการเชื่อมต่อทางกายภาพระหว่างอุปกรณ์ต้นทางและปลายทางหรือ DTE (Data Terminal Equipment) และอุปกรณ์ในการสื่อสารข้อมูล (DCE: Data Communication Equipment) ที่ทำการเปลี่ยนข้อมูลให้เป็นข้อมูลไบนารีแบบอนุกรม

มาตรฐาน EIA-232 อธิบายการเชื่อมต่อระหว่างเทอร์มินอล (DTE) และโมเด็ม (DCE) ที่ทำการส่งข้อมูลดิจิตอลแบบอนุกรม และมันยังเปิดช่องให้ผู้ออกแบบฮาร์ดแวร์และโปรโตคอลสามารถพัฒนาผลิตภัณฑ์ได้อย่างยืดหยุ่น

2). EIA-485

EIA-485 เป็นระบบสื่อสารแบบสมดุลซึ่งใช้ระดับสัญญาณระดับเดียวกับมาตรฐาน EIA-422 แต่เพิ่มอัตราการส่งข้อมูล และจำนวนตัวรับส่งในสื่อเดียวกัน สามารถมีถึง 32 ตัวตามที่มาตรฐานกำหนด EIA-485 สามารถใช้ระดับสัญญาณ TTL คือ 0-5 โวลต์ ในการส่งสัญญาณได้ โดยการส่งข้อมูลจะเป็นแบบ Half Duplex ในกรณีต้องการส่งข้อมูลแบบ Full Duplex จะต้องใช้วงจร RS-485 2 ชุดเรียกอีกอย่างว่า RS-422

3). สายใยแก้วนำแสง

คุณสมบัติที่โดดเด่นของสายใยแก้วนำแสงคือทนต่อสัญญาณรบกวนทางไฟฟ้าและแสงไม่ได้รับผลกระทบจากเสิร์จและทรานเชี้ยนต์ ซึ่งโดดเด่นมากโดยเฉพาะในงานติดตั้งใหม่ที่ต้องการส่งข้อมูลความเร็วสูงและปริมาณมาก

4). Foundation Fieldbus

Foundation Fieldbus ปัจจุบันอาจจะถือว่าเป็นมาตรฐานใหม่ล่าสุดที่ใช้ในการเชื่อมต่อระหว่างอุปกรณ์วัดค่า และ PLC รวมไปถึง DCS ตัว Foundation Fieldbus

4.1.2 การวิเคราะห์ข้อมูลจากความรู้ที่ได้รับในการศึกษา

เมื่อได้พิจารณาถึงคุณสมบัติและเอกลักษณ์ของแต่ละโปรโตคอลแล้วนั้น จะเห็นได้ว่ามีเพียงไม่กี่โปรโตคอลที่นำมาประยุกต์ใช้กับผลิตภัณฑ์ของสถานประกอบการณ์ได้ ซึ่งสถานประกอบการณ์เป็นผู้ผลิตและจัดจำหน่ายอินเวอร์เตอร์ โดยอินเวอร์เตอร์นั้นเป็นอุปกรณ์อิเล็กทรอนิกส์จำพวก PLC หรือเครื่องควบคุมเชิงตรรกที่สามารถโปรแกรมได้

โปรโตคอลที่สามารถใช้งานบน PLC ได้มีอยู่ด้วยกัน 4 ประเภทคือ Modbus Protocol, Profibus Protocol, Modbus Plus Protocol และ Data Hight plus หรือ DH485 ซึ่งเมื่อได้พิจารณาคุณสมบัติต่างๆของโปรโตคอลทั้งสี่แล้ว ทำให้ทราบว่า Modbus Plus Protocol เป็นโปรโตคอลที่ออกแบบมาเพื่อการใช้งานสำหรับ PLC ของ Modicon เท่านั้น และ Data Hight Plus/DH485 เป็นโปรโตคอลที่ใช้งานบน PLC ของ Allen Bradlley เท่านั้น จึงทำให้ทั้งสองโปรโตคอลไม่มีคุณสมบัติเพียงพอที่จะนำมาใช้ในการติดต่อสื่อสารของเครื่องอินเวอร์เตอร์ ซึ่งในกระบวนการผลิตย่อมต้องการโปรโตคอลที่มีความยืดหยุ่นในเรื่องของการใช้งานร่วมกับอุปกรณ์ที่มาจากต่างผู้พัฒนาและเป็นโปรโตคอลที่เป็นระบบเปิด คือโปรโคคอลที่ไม่เสียค่าใช้จ่ายและสามารถพัฒนาได้ง่าย ดังนั้นจึงเหลือเพียงสองโปรโตคอลที่เป็นที่น่าสนใจคือ

1). Profibus Protocol

Profibus Protocol เป็นโปรโตคอลที่ถูกพัฒนาขึ้นมาเพื่อใช้งานในอุตสาหกรรมที่ควบคุมโดยระบบ PLC ในสมัยก่อนนั้นพบว่ามีความยุ่งยากในการต่อสายสัญญาณซึ่งหนึ่งสายต่อหนึ่งอินพุตหรือเอาท์พุต ทำให้ต้องใช้สายจำนวนมากในการเชื่อมต่อ Profibus Protocol ได้ถูกออกแบบมาเพื่อการติดต่อสื่อสารแบบอนุกรมภายในโรงงาน โดยการใช้บัสเพียงเส้นเดียวในการเชื่อมต่อจึงทำให้สามารถลดจำนวนสายสัญญาณลงได้และยังสามารถเพิ่มความเร็วในการส่งข้อมูลได้อีกด้วย ซึ่ง Profibus Protocol เป็นรูปแบบการสื่อสารที่ไม่มีข้อผูกมัดกับหน่วยงานหรือองค์กรใดๆ พูดง่ายๆคือเป็นโปรโตคอลระบบเปิดนั่นเอง ด้วยเหตุผลนี้จึงทำให้ Profibus Protocol ได้รับความนิยมอย่างแพร่หลายในวงการอุตสาหกรรม

โปรฟิบัสสามารถแบ่งได้เป็น 3 ตระกูลคือ โปรฟิบัสดีพี (PROFIBUS DP – Decentralized Peripherals) ใช้สื่อสารระหว่างส่วนควบคุมกลางกับอุปกรณ์อินพุท-เอาท์พุทที่ระดับฟิลด์ เชื่อมต่อโดยใช้ EIA-485 โปรฟิบัสเอฟเอ็มเอส (PROFIBUS FMS – Fieldbus Message System) ใช้สื่อสารระหว่างพีเอลซี (PLCs – Programmable Logic Controllers) กับ PC (Personal Computer) และแลกเปลี่ยนข้อมูลที่ระดับเซลล์ และโปรฟิบัสพีเอ (PROFIBUS PA – Process Automation) เป็นส่วนขยายของโปรฟิบัสดีพี โดยสามารถรวมอุปกรณ์ของโปรฟิบัสพีเอ และโปรฟิบัสดีพีเข้าด้วยกันได้โดยการใช้อุปกรณ์แยกส่วน (Segment Coupler) ใช้ในการสื่อสารที่มีความเร็วสูงและระบบอัตโนมัติ และต้องการความน่าเชื่อถือ เชื่อมต่อโดยใช้ IEC61158

2). Modbus Protocol

Modbus Protocol เป็นโปรโตคอลที่ถูกพัฒนาโดย Modicon Inc. เป็นโปรโตคอลเพื่อการสื่อสารข้อมูล I/O และ Register ของ PLC ซึ่ง Modbus Protocol จะอ้างอิงกฎหรือขั้นตอนของชั้นดาต้าลิ้งค์เลเยอร์ (Data-Link Layer) และแอพพลิเคชั่นเลเยอร์ (Application Layer) ของโมเดล OSI เท่านั้น หมายความว่าขั้นตอนการทำงานในระดับชั้นฟิสิคอลเลเยอร์ (Physical Layer) สามารถใช้มาตรฐานใดก็ได้ Modbus Protocol เป็นที่ยอมรับกันอย่างกว้างขวางในการติดต่อสื่อสารที่เป็นแบบ Network Protocol อันเนื่องมาจาก MODBUS เป็นระบบเปิด, ไม่มีค่าใช้จ่าย, เชื่อมต่อและพัฒนาง่าย พร้อมทั้งยังสามารถนำโปรโตรคอลนี้ไปใช้งานในอุปกรณ์อื่นๆ เช่น Digital Power Meter, RTU (Remote Terminal Unit), Remote I/O, PLC เป็นต้น

การทำงานของ Modbus Protocol เป็นการรับและส่งข้อมูลโดยใช้รูปแบบของ Master/Slave ซึ่งเป็นการสื่อสารจากอุปกรณ์แม่ (Master) เครื่องเดียว ส่วนใหญ่มักเป็นซอฟต์แวร์คอมพิวเตอร์หรืออุปกรณ์แสดงผล HMI ไปยังอุปกรณ์ลูก (Slave) ได้หลายๆเครื่อง โดยสามารถกำหนดหมายเลขอุปกรณ์ได้สูงสุด 255 เครื่อง และสามารถกำหนดลักษณะการส่งข้อมูลได้ 2 แบบ

คือ ข้อมูลแบบแอสกี (ASCII) และข้อมูลแบบเลขฐานสอง (Binary) ในโปรโตคอล MODBUS ที่สื่อสารข้อมูลแบบ ASCII จะเรียก MODBUS ASCII และโปรโตคอล MODBUS ที่สื่อสารข้อมูลแบบเลขฐานสอง จะเรียก MODBUS RTU ทำให้มีความแตกต่างในการกำหนดค่าพอร์ตสื่อสาร ซึ่งอุปกรณ์ทุกตัวที่ต่อร่วมกันอยู่ในบัสหรือเครือข่ายเดียวกัน จะต้องตั้งให้เลือกใช้โหมดเดียวกันทั้งหมด

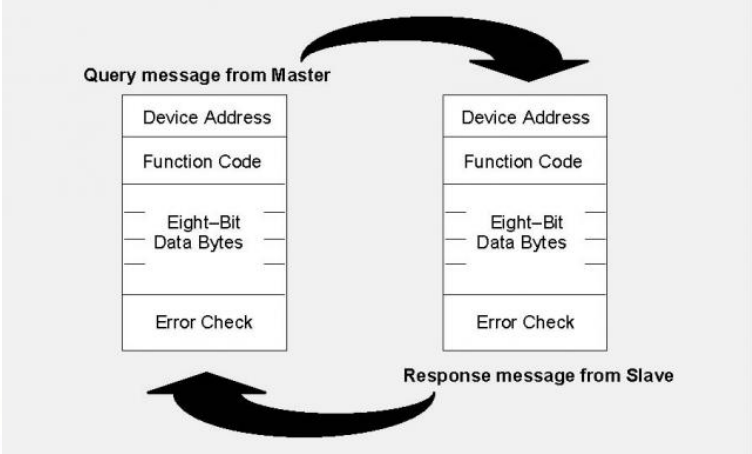
เมื่อได้ศึกษาข้อมูลเบื้องต้นของทั้งสองโปรโตคอลและได้วิเคราะห์ข้อมูลต่างๆแล้วนั้น สามารถสรุปได้ว่า Modbus Protocol มีคุณสมบัติที่เหมาะสมที่สุดที่จะนำมาใช้เป็นโปรโตคอลเพื่อการสื่อสารข้อมูลของผลิตภัณฑ์ของสถานประกอบการ โดยเมื่อเปรียบเทียบระหว่าง Profibus และ Modbus จะเห็นได้ว่า Profibus Protocol ยังมีข้อจำกัดในเรื่องของการเชื่อมต่อ ซึ่งหากใช้ Modbus Protocol เราสามารถเลือกใช้มาตรฐานสำหรับการเชื่อมต่อแบบใดก็ได้ โดยดูตามความเหมาะสมของสภาพการใช้งาน และลักษณะการนำไปใช้งานที่มีคุณสมบัติแตกต่างกันของทั้งสองโปรโตคอล ทำให้เห็นว่า Modbus Protocol เป็นโปรโตคอลที่มีความเหมาะสมมากที่สุดสำหรับการนำมาใช้ในการออกแบบการสื่อสารของเครื่องอินเวอร์เตอร์ อีกทั้ง Modbus Protocol ยังเป็นโปรโตคอลระบบเปิดที่ไม่มีค่าใช้จ่ายใดๆ และยังมีการเชื่อมต่อรวมไปถึงสามารถพัฒนาได้ง่ายอีกด้วย

4.1.3 ความรู้ที่ได้รับจากการศึกษาระบบการสื่อสารแบบ Modbus Protocol

Modbus Protocol เป็นโปรโตคอลที่ถูกออกแบบและพัฒนาโดย Modicon Inc. ในปี ค.ศ. 1997 เป็นระบบเพื่อการติดต่อสื่อสารข้อมูล Input Output และ Register ภายใน PLC ซึ่ง Modbus Protocol อยู่ในชั้น Application Layer ของ OSI Model และเนื่องจากเป็นโปรโตคอลที่สามารถเชื่อมต่อและพัฒนาได้ง่ายโดยไม่มีค่าใช้จ่ายใด ๆ จึงทำให้ Modbus Protocol ได้รับความนิยมอย่างแพร่หลายในวงการอุตสาหกรรมทั้งในอดีตและปัจจุบัน

ลักษณะการทำงานของ Modbus Protocol จะเป็นการรับและการส่งข้อมูลไปตามสายสัญญาณที่เรียกว่า Serial Port โดยวิธีการที่ง่ายที่สุดคือการต่อสายสัญญาณระหว่าง Master 1 ตัว ทำหน้าที่เป็นอุปรณ์ในการส่งคำร้องขอไปยัง Slave ที่มี 1 ตัวหรือหลายๆตัวก็ได้ (มากสุดได้ 255 ตัว) ซึ่ง Slave เป็นอุปกรณ์ที่คอยทำหน้าที่ในการรับคำร้องขอจาก Master และทำการส่งค่ากลับไปยัง Master อีกครั้งเพื่อเป็นการยืนยันการกระทำที่ได้ร้องขอมา และในการดำเนินการของ Modbus Protocol จะมีรูปแบบในการรับและการส่งข้อมูลเป็นแบบ Query-Response Cycle คือ ข้อมูลที่เป็นการร้องขอการกระทำทั้งหมดจะถูกเก็บไว้ในส่วนของ Query Message ซึ่งจะประกอบไปด้วย

Device Address, Function Code, Eight Bit-Data Byte และ Error Check และข้อมูลที่เป็นการตอบกลับจะถูกเก็บไว้ในส่วน Response Message ประกอบไปด้วย Device Address, Function Code, Eight Bit-Data Byte และ Error Check เช่นเดียวกับส่วนของ Query Message แต่มีหน้าที่ในการทำงานที่แตกต่างกัน



รูปที่ 4.1 วงจรของ Query-Response

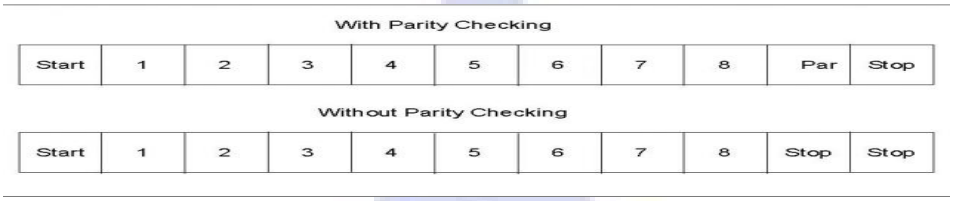
ภายใน Modbus Protocol มีโหมดสำหรับการส่งผ่านข้อมูลด้วยกัน 2 รูปแบบ โดยการเลือกใช้งานขึ้นอยู่กับการตั้งค่าบนคอนโทรลเลอร์ ซึ่งโหมดดังกล่าวมีดังนี้

1). RTU Mode

เป็นโหมดที่มีลักษณะการส่งข้อมูลแบบ 8 บิต จำนวน 1 ไบต์ ซึ่งบรรจุเลขฐานสอง โดยจะส่งไปในรูปของเฟรมข้อมูล ซึ่งภายในเฟรมจะบรรจุข้อมูลต่างๆไว้ 11 บิต ประกอบไปด้วย Start Bit จำนวน 1 บิต Data Bit จำนวน 8 บิต โดยบิตที่สำคัญน้อยที่สุด (LSB) จะถูกส่งออกไปก่อน Parity Check 1 บิต ประกอบไปด้วย Even, Odd, No Parity และ Stop Bit 1 บิต แต่หากมีการกำหนด Parity Check เป็น No Parity จะต้องมีการเพิ่ม Stop Bit เป็น 2 บิต ในการตรวจสอบความผิดพลาดของเฟรมข้อมูลในโหมด RTU นั้นจะใช้วิธีการ CRC หรือ Cyclical Redundancy Check โดยการสร้างรหัส CRC เข้าไปต่อที่ส่วนท้ายของเฟรมข้อมูล ประโยชน์หลักของโหมดนี้คือเป็นโหมดที่ยอมให้ส่งตัวอักษรได้จำนวนมาก และ Data Throughput ดีกว่าโหมด ASCII ในช่วงที่ Baud Rate ที่เหมือนกัน

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4

รูปที่ 4.2 ลักษณะเฟรมข้อมูลของ Modbus RTU



รูปที่ 4.3 ลักษณะข้อมูลแต่ละไบต์ของ Modbus RTU

2). ASCII Mode

เป็นโหมดที่มีลักษณะการส่งข้อมูลแบบ 8 บิต จำนวน 1ไบต์ โดยจะส่งเป็นรหัส ASCII จำนวน 2 ตัวอักษร โดยจะส่งไปในรูปของเฟรมข้อมูล ภายในเฟรมจะบรรจุข้อมูลต่างๆไว้ 10 บิต ประกอบไปด้วย Start Bit จำนวน 1 บิต Data Bit จำนวน 8 บิต โดยบิตที่สำคัญน้อยที่สุด (LSB) จะถูกส่งออกไปก่อน Parity Check 1 บิต ประกอบไปด้วย Even, Odd, No Parity และ Stop Bit 1 บิต แต่หากมีการกำหนด Parity Check เป็น No Parity จะต้องมีการเพิ่ม Stop Bit เป็น 2 บิต ซึ่งลักษณะพิเศษของเฟรมข้อมูลในโหมด ASCII คือจุดเริ่มต้นของเฟรมจะเป็นเครื่องหมาย ":" (Colon) และจะสิ้นสุดด้วยตัวอักษร "CRLF" เสมอ ในการตรวจสอบความผิดพลาดของเฟรมข้อมูลในโหมด ASCII นั้นจะใช้วิธีการ LRC หรือ Longitudinal Redundancy Check โดยวิธีการของอัลกอริทึมที่นำค่าข้อมูลของตัวอักษร ASCII ที่ส่งออกไปตั้งแต่แรก นำมาผ่านลอจิก Exclusive-OR (XOR) ไปเรื่อยๆจนถึงตัวสุดท้ายที่ค่าตัวอักษร BCC (Block Check Character) และข้อดีของโหมด ASCII คือการส่งข้อมูลในโหมด ASCII นั้นเป็นไปอย่างรวดเร็ว โดยที่มี Error เกิดขึ้นน้อยมาก

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

รูปที่ 4.4 ลักษณะเฟรมข้อมูลของ Modbus ASCII

With Parity Checking

Start	1	2	3	4	5	6	7	8	Par	Stop

Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

รูปที่ 4.5 ลักษณะข้อมูลแต่ละไบต์ของ Modbus ASCII

เมื่อทราบถึงโหมดในการทำงานของ Modbus Protocol แล้วนั้น ส่วนที่สำคัญต่อไปคือการออกคำสั่งให้ Slave กระทำการใดๆตามคำร้องขอของ Master นั้น Slave จะต้องได้รับคำสั่งจาก Master โดยจะถูกเก็บไว้ในส่วนของ Function Code ซึ่ง Function Code แต่ละตัวมีคุณสมบัติที่แตกต่างกันออก ขึ้นอยู่กับว่าจะต้องการอ่านข้อมูลเพียงอย่างเดียว หรือทั้งอ่านและเขียนข้อมูล เราสามารถแบ่ง Function Code ได้ 4 ลักษณะดังนี้

1). Discrete Output Coils หรือ Read Coil Status (0x01)

Discrete Output Coils เป็นการอ่านสถานะ On/Off ของ Discrete Outputs ใน Slave ซึ่ง วิธีการนี้จะไม่รองรับการส่งข้อมูลแบบ Broadcast

2). Discrete Input Contact หรือ Read Input Status (0x02)

Discrete Input Contact เป็นการอ่านสถานะ On/Off ของ Discrete Input ใน Slave ซึ่ง วิธีการนี้จะไม่รองรับการส่งข้อมูลแบบ Broadcast

3). Analog Output Holding Register หรือ Read Holding Registers (0x03)

Analog Outputs Holding Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสอง ของ Holding Register ใน Slave แต่จะไม่รองรับการส่งข้อมูลประเภท Broadcast

4). Analog Input Register หรือ Read Input Register (0x04)

Analog Input Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสองของ Input Register ใน Slave แต่จะไม่รองรับการส่งข้อมูลประเภท Broadcast

ซึ่งเมื่อได้ทำการศึกษา Modbus Protocol อย่างละเอียดแล้วนั้น ทำให้เห็นว่า Modbus Protocol มีทั้งข้อดีและข้อเสียต่างๆมากมาย โดยสามารถสรุปได้ดังนี้

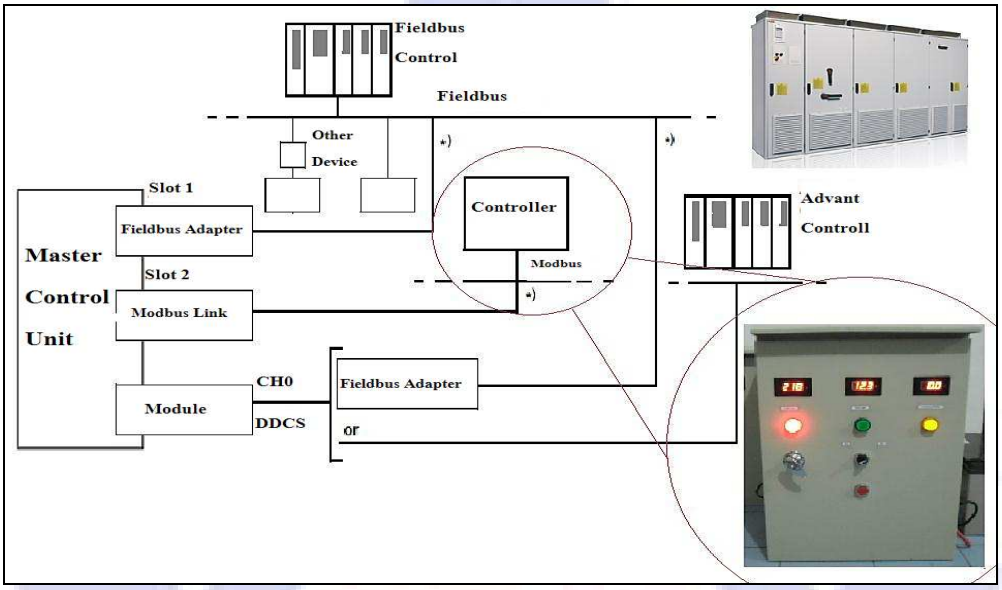
1). ข้อดีของ Modbus Protocol

- เป็นโปรโตคอลที่ง่ายต่อการเรียนรู้และพัฒนา
- มีการใช้งานอย่างแพร่หลายเช่น ระบบ Process Control, Power Meter, Chillers, Boilers, ควบคุมการจ่ายกระแสไฟฟ้า เป็นต้น

- การรวมกลุ่มของผู้ใช้งานและผู้พัฒนาทำให้อุปกรณ์ที่ผลิตจากหลายผู้ผลิตสามารถใช้งานร่วมกันได้

2). ข้อเสียของ Modbus Protocol

- เป็นโปรโตคอลที่มีการส่งข้อมูลง่ายๆไม่ซับซ้อนอาจทำให้เกิดข้อจำกัดในการส่งข้อมูลได้
- ไม่ใช่โปรแกรมระดับสูง Standardization
- มีข้อมูลการใช้งานที่จำกัด
- ผู้ขายอาจจะไม่สามารถกำหนดความสามารถของ Modbus ที่แน่นอนได้



รูปที่ 4.6 โครงสร้างของ Modbus Protocol กับการนำไปใช้งานบนอินเวอร์เตอร์

4.1.4 Modbus Protocol ภายใต้การควบคุมของคอนโทรลเลอร์

Modbus Protocol จะสามารถทำงานได้นั้นต้องมีคำสั่งที่ใช้ควบคุมการทำงาน โดยจะเขียนคำสั่งผ่านคอนโทรลเลอร์ และในการติดต่อสื่อสารจะต้องมี Serial Communication Interface (SCI) ซึ่งเป็นการแลกเปลี่ยนข้อมูลแบบ Two-wire หรือที่รู้จักกันในนาม UART โดย SCI จะรองรับการเชื่อมต่อของ CPU กับ Device อื่นๆที่ต้องการจัดรูปแบบของการติดต่อสื่อสาร ซึ่งคอนโทรลเลอร์จะเป็นตัวกำหนดคำสั่งและฟังก์ชั่นต่างๆของ SCI

ในที่นี้จะใช้คอนโทรลเลอร์ชนิด DSP2808 ในการเขียนโปรแกรมคำสั่ง และใช้ RS-232 เป็น Serial Port Interface สำหรับการติดต่อสื่อสาร ซึ่ง RS-232 เป็นสายที่ใช้เชื่อมต่อระหว่าง

คอนโทรลเลอร์และ CPU สำหรับการประมวลผล โดยคำสั่งควบคุมที่เขียนไว้บนคอนโทรลเลอร์จะถูกส่งผ่านไปทาง RS-232 เมื่อมีการเรียกใช้งานโปรแกรมดังกล่าวหรือขณะที่มีการเปิดเครื่องขึ้น

4.1.5 ผลจากการศึกษาคอนโทรลเลอร์ชนิด DSP2808

DSP2808 เป็นคอนโทรลเลอร์ที่ทำงานบน CMOS Technology ที่ช่วงความถี่ 100 MHz หรือมีการหมุนของเวลาเท่ากับ 10 nS. ในหนึ่งรอบการหมุน ใช้กำลังไฟฟ้าต่ำที่ช่วงแรงดันไฟฟ้า 3.3 V. สำหรับการป้อนอินพุตและเอาท์พุต เป็นคอนโทรลเลอร์แบบ 32-CPU Timer มีเมมโมรี่เป็นลักษณะ On chip Memory ควบคุมการทำงานด้วยระบบ Boot ROM (4 K x 16), Clock and System Control, 128-Bit Security Key/Lock, 16 PWM Outputs, 16 HRPWM Outputs With 150 ps MEP, 12-Bit ADC, 16 Chanels,IDLE and STANDBY and HALT Modes Supported และมี Serial Port Peripheral เป็น Up to 4 SPI Modules, 2 SCI (UART) Module, 2 CAN Modules, One Inter-Integrated-Circuit (I2C) Bus

เมื่อศึกษาการเริ่มต้นใช้งานของ DSP2808 ในขั้นแรก จะต้องมีการจัดเตรียมอุปกรณ์สำหรับการพัฒนาซอฟแวร์ ซึ่งประกอบด้วย JTAG emulation เป็นอุปกรณ์สำหรับการอินเตอร์เฟสรองรับการทำงานแบบ Real-Time Mode บนรายละเอียดการทำงานของเมมโมรี่อุปกรณ์ต่อพ่วงและรีจิสเตอร์ ซึ่งสามารถดัดแปลงได้ในขณะที่โปรเซสเซอร์กำลังทำงานและประมวลผลข้อมูลอยู่ และจะต้องมีไดรฟ์เวอร์สำหรับ Emulation ที่เหมาะสม รวมไปถึงโปรแกรมซอฟแวร์ที่ใช้สำหรับการเขียนคำสั่งควบคุมคอนโทรลเลอร์

การเขียนโปรแกรมคำสั่งควบคุมการทำงานของ DSP2808 จะใช้การเขียนโปรแกรมโดยใช้ภาษาทางคอมพิวเตอร์อย่างง่ายและไม่มีความซับซ้อนมากนัก ในที่นี้จะใช้ภาษาซี (C Language) ในการเขียนโปรแกรม ซึ่งจะใช้ Code Composer Studio V.3.3 ในการเขียนโปรแกรมควบคุมคำสั่ง

4.1.6 การเขียนโปรแกรมควบคุมการทำงานของคอนโทรลเลอร์ชนิด DSP2808

ในการเขียนโปรแกรมควบคุมการทำงานของ DSP2808 ในที่นี้ได้ยกตัวอย่างโปรแกรม SCI Echoback มาใช้ประกอบการศึกษา โดยเป็นการทดลองประมวลผลคำสั่งผ่านโปรแกรม Hyper terminal ซึ่งใช้ RS232 เป็น Serial Port ระหว่างการติดต่อสื่อสารของเครื่องคอมพิวเตอร์และคอนโทรลเลอร์ ลักษณะการทำงานของโปรแกรมคือ ผู้ใช้จะทำการป้อนข้อมูลผ่านทางเครื่องคอมพิวเตอร์ และเมื่อคอนโทรลเลอร์ประมวลผล มันจะส่งค่าที่ผู้ใช้ป้อนเข้าไปกลับมา โดยจะแสดงการรับค่าและส่งค่ากลับบนโปรแกรม Hyper Terminal

4.1.6.1 ตัวอย่างโปรแกรม SCI Echoback

```
// FILE:    Example_280xSci_Echoback.c
// TITLE:   DSP280x Device SCI Echoback.
// ASSUMPTIONS:
//    This program requires the DSP280x header files. As supplied, this project is configured
//for "boot to SARAM" operation. Connect the SCI-A port to a PC via a transciever and
//cable. The PC application 'hypterterminal' can be //used to view the data from the SCI and
//to send information to the SCI.  Characters recieved  by the SCI port are sent back to the
//host.
//    As supplied, this project is configured for "boot to SARAM" operation.  The 280x //Boot
Mode table is shown below. For information on configuring the boot mode of an //eZdsp,
please refer to the documentation included with the eZdsp,
//      Boot              GPIO18              GPIO29              GPIO34
//      Mode              SPICLKA             SCITXDA             SCITXB
//      ------------------------------------------------------------------------------------
//      Flash             1                   1                   1
//      SCI-A             1                   1                   0
//      SPI-A             1                   0                   1
//      I2C-A             1                   0                   0
//      ECAN-A            0                   1                   1
//      SARAM             0                   1                   0  <- "boot to
SARAM"
//      OTP               0                   0                   1
//      I/0               0                   0                   0
//
//    DESCRIPTION:
```

```
//   This test recieves and echo-backs data through the SCI-A port.
//   1) Configure hyperterminal:Use the included hyperterminal configuration file SCI_96.ht.
//      To load this configuration in hyperterminal: file->open
//      and then select the SCI_96.ht file.
//   2) Check the COM port.The configuration file is currently setup for COM1. If this is not
correct, disconnect Call->Disconnect  Open the File-Properties dialog and select the correct
COM port.
//   3) Connect hyperterminal Call->Call  and then start the 280x SCI echoback program
execution.
//   4) The program will print out a greeting and then ask you to  enter a character which it
will echo back to hyperterminal.
//
//   As is, the program configures SCI-A for 9600 baud with
//   SYSCLKOUT = 100MHz and LSPCLK = 25Mhz.
//   ----------------------------------------------------------
//    Watch Variables:
//     LoopCount for the number of characters sent
//     ErrorCount
//###########################################################################
// $TI Release: DSP280x C/C++ Header Files V1.70 $
// $Release Date: July 27, 2009 $
//###########################################################################
#include "DSP280x_Device.h"     // DSP280x Headerfile Include File
#include "DSP280x_Examples.h"   // DSP280x Examples Include File
// Prototype statements for functions found within this file.
void scia_echoback_init(void);
void scia_fifo_init(void);
void scia_xmit(int a);
void scia_msg(char *msg);
// Global counts used in this example
```

```
Uint16 LoopCount;
Uint16 ErrorCount;


void main(void)
{
   Uint16 ReceivedChar;
   char *msg;
// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP280x_SysCtrl.c file.
  InitSysCtrl();
// Step 2. Initalize GPIO:
// This example function is found in the DSP280x_Gpio.c file and
// illustrates how to set the GPIO to it's default state.
// InitGpio(); Skipped for this example
// For this example, only init the pins for the SCI-A port.
// This function is found in the DSP280x_Sci.c file.
  InitSciaGpio();
// Step 3. Clear all interrupts and initialize PIE vector table:
// Disable CPU interrupts
  DINT;
// Initialize PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP280x_PieCtrl.c file.
  InitPieCtrl();
// Disable CPU interrupts and clear all CPU interrupt flags:
  IER = 0x0000;
  IFR = 0x0000;
```

```
// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP280x_DefaultIsr.c.
// This function is found in DSP280x_PieVect.c.
InitPieVectTable();
// Step 4. Initialize all the Device Peripherals:
// This function is found in DSP280x_InitPeripherals.c
// InitPeripherals(); // Not required for this example
// Step 5. User specific code:
LoopCount = 0;
ErrorCount = 0;
scia_fifo_init();              // Initialize the SCI FIFO
scia_echoback_init();          // Initalize SCI for echoback
msg = "\r\n\n\nHello World!\0";
scia_msg(msg);

msg = "\r\nYou will enter a character, and the DSP will echo it back! \n\0";
scia_msg(msg);
        for(;;)
{
msg = "\r\nEnter a character: \0";
scia_msg(msg);
// Wait for inc character
while(SciaRegs.SCIFFRX.bit.RXFFST !=1) { } // wait for XRDY =1 for empty state
// Get character
ReceivedChar = SciaRegs.SCIRXBUF.all;
// Echo character back
```

```
     msg = "  You sent: \0";
      scia_msg(msg);
      scia_xmit(ReceivedChar);
      LoopCount++;
   }
}
// Test 1,SCIA  DLB, 8-bit word, baud rate 0x000F, default, 1 STOP bit, no parity
void scia_echoback_init()
{
   // Note: Clocks were turned on to the SCIA peripheral
   // in the InitSysCtrl() function
          SciaRegs.SCICCR.all =0x0007;          // 1 stop bit,  No loopback
                                                // No parity,8 char bits,
                                                // async mode, idle-line protocol
          SciaRegs.SCICTL1.all =0x0003;         // enable TX, RX, internal SCICLK,
                                                // Disable RX ERR, SLEEP, TXWAKE
          SciaRegs.SCICTL2.all =0x0003;
          SciaRegs.SCICTL2.bit.TXINTENA =1;
          SciaRegs.SCICTL2.bit.RXBKINTENA =1;
 #if (CPU_FRQ_100MHZ)
          SciaRegs.SCIHBAUD    =0x0001;         // 9600 baud @LSPCLK = 20MHz.
          SciaRegs.SCILBAUD    =0x0004;
#endif
#if (CPU_FRQ_60MHZ)
          SciaRegs.SCIHBAUD    =0x0000;         // 9600 baud @LSPCLK = 15MHz.
          SciaRegs.SCILBAUD    =0x00C2;
#endif
          SciaRegs.SCICTL1.all =0x0023;         // Relinquish SCI from Reset
}
```

```
// Transmit a character from the SCI
void scia_xmit(int a)
{
   while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
   SciaRegs.SCITXBUF=a;
}
void scia_msg(char * msg)
{
   int i;
   i = 0;
   while(msg[i] != '\0')
   {
      scia_xmit(msg[i]);
      i++;
   }
}
// Initalize the SCI FIFO
void scia_fifo_init()
{
   SciaRegs.SCIFFTX.all=0xE040;
   SciaRegs.SCIFFRX.all=0x204f;
   SciaRegs.SCIFFCT.all=0x0;
}
//===========================================================================
// No more.
//===========================================================================
```

4.1.6.2 ผลของการประมวลผลโปรแกรม

เมื่อได้ตั้งค่าต่างๆภายในโปแกรมเป็นที่เรียบร้อยแล้ว หลังจากที่มีการประมวลผลข้อมูล ผลที่ออกมาบนโปรแกรม Hyper Terminal เป็นดังนี้

```
Hello World!
You will enter a character, and the DSP will echo it back!

Enter a character: 1  You sent: 1
Enter a character: 4  You sent: 4
Enter a character: 2  You sent: 2
Enter a character: 5  You sent: 5
Enter a character: 3  You sent: 3
Enter a character: 6  You sent: 6
Enter a character:
```

รูปที่ 4.7 ผลที่ได้จากการประมวลผลโปรแกรม

4.1.6.3 การกำหนดค่า

การกำหนดค่าต่าง ๆ ของ SCI echoback ที่สำคัญของโปรแกรมมีดังต่อไปนี้

1). ค่า SCICCR เป็นการกำหนด Stop Bit, Even or Odd Parity, Parity Enable, Loopback Enable, ADDR or Idle Mode และ SCI Character อีก 3 บิต รวมทั้งสิ้น 8 บิต

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น 1 Stop Bit, No Loopback, No Parity, 8 Char Bits, Async Mode, Idle-line Protocol

ดังนั้น ค่าSCICCR จึงเป็น 00000111 หรือ0x0007

2).ค่า SCICTL1 เป็นการกำหนด SCI Receive Error Interrupt Enable, SCI Software Reset, SCI Transmitter Wake-up Method Select, SCI Sleep, SCI Receiver Enable โดยบิตที่ 4 และ 7 เป็นบิตที่ไม่มีผลในการประมวลผล ซึ่งส่วนใหญ่จะตั้งค่าให้เป็น 0 รวมทั้งสิ้น 8 บิต

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น TX & RX, Internal SCICLK ให้เป็น Enable และ RX ERR, SLEEP, TXWAKE ให้เป็น Disable

เมื่อมีการตั้งค่า SCIHBAUD & SCILBAUD แล้วนั้นจะต้องตั้งค่า SCICTL1 อีกครั้ง เพื่อให้มีการตรวจสอบความผิดพลาดในการประมวลผล โดยจะต้องตั้งค่า RX Error Interrupt อยู่ในสถานะ Enable

ดังนั้น ค่า SCICTL1 จึงเป็น 00100011 หรือ 0x0023

3).ค่า SCICTL2 เป็นการกำหนด Transmitter Buffer Register Ready Flag, Transmitter Empty Flag, Receiver Buffer/Break Interrupt Enable, SCITXBUF-Register Enable โดยบิตที่ 2-5 เป็นบิตที่ไม่มีผลในการประมวลผล ซึ่งส่วนใหญ่จะตั้งค่าให้เป็น 0 รวมทั้งสิ้น 8 บิต

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น TX Interrupt & RX/BK Interrupt โดยจะต้องกำหนดค่า TXINTENA และ RXBKINTENA ให้เป็น 1

ดังนั้น ค่า SCICTL2 จึงเป็น 00000011 หรือ 0x0003

4). ค่า SCIHBAUD & SCILBAUD เป็นการกำหนดค่า SCI 16-bit Baud Selection Register SCIHBAUD (MSB) and SCILBAUD (LSB) โดยข้อมูลทั้งหมดจะมี 16 บิต ซึ่งบิตที่ 8-15 เป็นค่า SCIHBAUD และบิตที่ 0-7 เป็นค่า SCILHAUD

ค่า SCIHBAUD & SCILBAUD เป็นค่าที่ได้จากการคำนวณ โดยมีสูตรในการคำนวณดังนี้

SCI Asynchronous Baud = $\frac{LSPCLK}{(BRR+1)x8}$

BRR = $\left(\frac{LSPCLK}{SCI\ Asynchronous\ Baud x8}\right) - 1$

เมื่อ 1 = <BRR = < 65535 แต่ถ้าค่า BRR = 0 จะใช้สูตรต่อไปนี้ในการคำนวณ

SCI Asynchronous Baud = $\frac{LSPCLK}{16}$

โดยค่า BRR ที่ได้จะเท่ากับค่า 16 บิตซึ่งออกมาเป็นฐานสิบ (ต้องแปลงจากฐานสิบเป็นฐานสิบหก) ซึ่งค่า BRR เป็นค่าที่ใช้ Baud-Select Register

จากโปรแกรมตัวอย่างที่ช่วงความถี่ 100 MHz. กำหนดให้ค่า Baud Rate เท่ากับ 9600 bps. และ LSPCLK เท่ากับ 20 MHz. คำนวณได้ดังนี้

จากสูตร SCI Asynchronous Baud = $\frac{LSPCLK}{(BRR+1)x\ 8}$

แทนค่า 9600 = $20x\ \frac{10^6}{(BRR+1)x\ 8}$

จะได้ BRR = 259.4167 ปัดเป็น 260

เมื่อค่า 260 มาแปลงเป็นเลขฐานสอง 16 บิตจะได้ 0000 0001 0000 0010

ดังนั้น ค่า SCIHBAUD จะเป็น 0000 0001 หรือ 0x0001 และค่า SCILBAUD จะเป็น 0000 0010 หรือ 0x0004

และจากโปรแกรมตัวอย่างที่ช่วงความถี่ 60 MHz. กำหนดให้ค่า Baud Rate เท่ากับ 9600 bps. และ LSPCLK เท่ากับ 15 MHz. คำนวณได้ดังนี้

จากสูตร SCI Asynchronous Baud = $\frac{LSPCLK}{(BRR+1)x8}$

แทนค่า 9600 = $15\ X\ \frac{10^6}{(BRR+1)x8}$

จะได้ BRR = 194.3125 ปัดเป็น 194

เมื่อค่า 260 มาแปลงเป็นเลขฐานสอง 16 บิตจะได้ 0000 0000 1100 0010

ดังนั้น ค่า SCIHBAUD จะเป็น 0000 0001 หรือ 0x0000 และค่า SCILBAUD จะเป็น 0000 0010 หรือ 0x00C2

5). ค่า SCIFFTX เป็นการกำหนด SCI Reset, SCI FIFO Reset, Transmit FIFO Reset, TXFFST4-0, Transmit FIFO Interrupt, Transmit FIFO Clear, Transmit FIFO Interrupt Enable, TXFFIL4-0 Transmit FIFO Interrupt Level Bit รวมทั้งหมด 16 บิต โดยบิตที่ 8-12 เป็นการกำหนดค่า TXFFST4-0

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น SCI FIFO can Resume Transmit or Receive, SCI FIFO are Enable, Re-Enable Transmit FIFO Operation, Transmit FIFO is Empty, TXFIFO Interrupt has not Occured, Clear TXFFINT Flag in Bit 7, TXFFIENA are Disable

ดังนั้น ค่า SCIFFTX จึงเป็น 1110 0000 0100 0000 หรือ 0xE040

6).ค่า SCIFFRX เป็นการกำหนด Receive FIFO Overflow, RXFFOVF Clear, Receive FIFO Reset, RXFFST4-0, Receive FIFO Interrupt, Receive FIFO Interrupt Clear, Receive FIFO Interrupt Enable, Receive FIFO Interrupt Level Bit รวมทั้งหมด 16 บิต โดยบิตที่ 8-12 เป็นการกำหนดค่า RXFFST4-0

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น Receive FIFO has not Overflow, No Effect on RXFFOVF, Re-Enable Receive FIFO Operation, Receive FIFO is Empty, RXFIFO Interrupt has not Occured, Clear RXFFINT Flag in Bit 7, Receive FIFO Interrupt Clear are Disable

ดังนั้น ค่า SCIFFRX จึงเป็น 0010 0000 0100 1111 หรือ 0x204F

7). ค่า SCIFFCT เป็นการกำหนด Auto-Baud Detect (ABD) bit, ABD Clear bit, CDC Calibrate A-Detect bit, FIFO Transfer Delay

จากโปรแกรมตัวอย่างมีการตั้งค่าเป็น Auto-Baud Detection is not Complete, No Effect on ABD, Disable Auto-Baud Alignment

ดังนั้น ค่า SCIFFCT จึงเป็น 0000 0000 0000 0000 หรือ 0x0

4.1.6.4 Header Files สำหรับโปรแกรม Sci Echoback

ไฟล์ข้อมูลที่ใช้ประกอบโปรแกรม Sci Echoback มีดังต่อไปนี้

1). Device.h

เป็นการเขียนคำสั่งที่ใช้สำหรับตั้งค่าต่างๆของคอนโทรลเลอร์ ไม่ว่าจะเป็นชนิดของคอนโทรลเลอร์ ชนิดข้อมูลต่างๆ รวมไปถึงการเรียกใช้งานไฟล์ต่าง ๆ ที่เกี่ยวข้องกับการตั้งค่าของคอนโทรลเลอร์ทั้งหมด

2). Example.h

เป็นการเขียนคำสั่งที่ใช้สำหรับตั้งค่า PLL control register (PLLCR) clock in divide (CLKINDIV) ค่า Clock Rate ต่าง ๆ รวมไปถึงการตั้งค่าความถี่ที่จะใช้สำหรับการประมวลผลและการคำนวณ Us_Delay ของคอนโทรลเลอร์

4.1.6.5 Initialize Files

โปรแกรมคำสั่งที่มีการเรียกใช้งานในโปรแกรม Sci Echoback มีดังต่อไปนี้

1). SysCtrl.c

ฟังก์ชั่น InitSysCtrl(); เป็นการเรียกใช้งาน System Control มีชื่อไฟล์ว่า SysCtrl.c โดยโปรแกรมคำสั่งดังลกล่าวเป็นการตั้งค่าของ PLL, WatchDog และ Peripheral Clocks ซึ่งเป็นการเขียนคำสั่งสำหรับการโหลดค่าและรันที่อยู่โดยการใช้ linker cmd file.

2). Sci.c

ฟังก์ชั่น InitSciaGpio(); เป็นการเรียกใช้งานไฟล์ Sci.c ซึ่งเป็นโปรแกรมสำหรับเรียกใช้พินสำหรับ SCI-A Port

3). PieCtrl.c

ฟังก์ชั่น InitPieCtrl(); เป็นการเรียกใช้งาน PIE control registers เพื่อตั้งค่าสถานะเป็น Default ซึ่ง Default State เป็นการปิดการใช้งาน Peripheral และ Flag ทุกชนิด โดยไฟล์ที่ถูกเรียกใช้งานมีชื่อว่า PieCtrl.c

4). PieVect.c

ฟังก์ชั่น InitPieVectTable(); เป็นการเรียกใช้งาน PIE vector table กับ pointers ที่ใช้ชี้ shell Interrupt Service Routines (ISR) มีชื่อไฟล์ว่า PieVect.c

*<u>หมายเหตุ</u> ตัวอย่างไฟล์โปรแกรมต่าง ๆ จะนำไปแสดงในส่วนภาคผนวกหัวข้อที่ 1 และ 2 บนแผ่นซีดี

4.2 ผลการวิเคราะห์ข้อมูล

จากการศึกษาลักษณะการติดต่อสื่อสารบนเครือข่ายและอุปกรณ์อิเล็กทรอนิกส์ต่างๆ จะเห็นได้ว่า Modbus Protocol เป็นโปรโตคอลที่มีความเหมาะสมมากที่สุด เนื่องจากเป็นโปรโตคอลที่สามารถแก้ไขและพัฒนาโดยผู้ใช้ได้ง่ายและไม่เสียค่าใช้จ่าย จึงทำให้ Modbus Protocol เป็นโปรโตคอลที่ได้รับความนิยมสูงในวงการอุตสาหกรรม หน้าที่ของ Modbus Protocol คือทำการติดต่อสื่อสารกันระหว่าง Device และ Terminal ในการรับและการส่งข้อมูลภายใต้การควบคุมของคอนโทรลเลอร์ มีรูปแบบ การทำงานเป็นแบบ Master/Slave คือจะมีเครื่อง Master แค่ 1 เครื่องเท่านั้น แต่จะมีเครื่อง Slave เท่าใดก็ได้แต่ไม่เกิน 255 เครื่อง และมีการเชื่อมต่อผ่าน Field Bus Plug ซึ่งมีหลายประเภทด้วยกัน แต่ที่ได้รับความนิยมคือ RS232 และ RS485 เป็น Truck Cable ที่ใช้ในการเชื่อมต่อแบบ Point-to-Point โดย RS232 ใช้กับการเชื่อมต่อในลักษณะที่มีการรับและการส่งที่มีจำนวนข้อมูลไม่มากและไม่ซับซ้อน ส่วน RS485 ใช้กับการเชื่อมต่อที่ต้องการส่งข้อมูลเป็นจำนวนมาก การทำงานของ Modbus Protocol แบ่งเป็น 2 โหมดด้วยกันคือโหมด RTU และโหมด ASCII ซึ่งทั้งสองโหมดมีลักษณะต่างกัน โดยโหมด RTU จะส่งข้อมูลเป็นเฟรมข้อมูลที่บรรจุเลขฐานสองขนาด 8 บิต และมีการตรวจสอบความผิดพลาดแบบ Cyclical Redundancy Check มีข้อดีคือยอมให้ส่งตัวอักษรได้จำนวนมาก ส่วนโหมด ASCII นั้น เฟรมข้อมูลจะบรรจุเลขฐานสิบหกขนาด 8 บิต จำนวน 2 ไบต์ และจะส่งออกมาในลักษณะของตัวอักษรแอสกี 2 ตัว ซึ่งในโหมด ASCII นี้มีการตรวจสอบความผิดพลาดแบบ Logitudinal Redundancy Check มีข้อดีคือสามารถส่งข้อมูลที่มีความซับซ้อนได้มากกว่าโหมด RTU และเกิดความผิดพลาดขึ้นน้อย แต่จะส่งตัวอักษรครั้งละมากๆไม่ได้ การออกคำสั่งให้ Slave ดำเนินการตามคำร้องขอของ Master ในคำสั่งของเฟรม

ข้อมูลนั้นมีทั้งคำสั่งที่เป็นเพียงการอ่านเพียงอย่างเดียวหรือทั้งอ่านทั้งเขียนข้อมูล จะใช้ Function Code ในการกำหนดค่าดังกล่าว โดยในที่นี้ได้ทำการศึกษา 4 รูปแบบ คือ Discrete Output Coils เป็นการอ่านสถานะ On/Off ของ Discrete Outputs ใน Slave ต่อมาคือ Discrete Input Contact เป็นการอ่านสถานะ On/Off ของ Discrete Input ใน Slave และ Analog Outputs Holding Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสองของ Holding Register ใน Slave ซึ่งมีความสามารถทั้งการเขียนและการอ่านข้อมูล ส่วน Analog Input Register เป็นการอ่านรายละเอียดของข้อมูลที่เป็นเลขฐานสองของ Input Register ใน Slave โดยมีความสามารถในการอ่านได้เพียงอย่างเดียวเท่านั้น ซึ่งทุกรูปแบบไม่รองรับการส่งข้อมูลแบบ Broadcast

ในการศึกษาการทำงานของ Modbus Protocol ร่วมกับอุปกรณ์อิเล็กทรอนิกส์ ในที่นี้ได้ใช้คอนโทรลเลอร์ชนิด DSP2808 เป็นคอนโทรลเลอร์ที่ใช้เขียนคำสั่งควบคุมการทำงานของอุปกรณ์ โดยใช้โปรแกรมตัวอย่าง SCI Echoback ในการศึกษา และใช้ RS232 ในการเชื่อมต่อระหว่างคอมพิวเตอร์และคอลโทรลเลอร์ ซึ่งเขียนโปรแกรมควบคุมบน Code Composer Studio V. 3.3 ผลที่ได้จะแสดงค่าบนโปรแกรม Hyper Terminal ลักษณะของโปรแกรมเมื่อประมวลผลบน DSP2808 แล้วนั้น จะเป็นการสะท้อนกลับของข้อมูลที่ผู้ใช้ป้อนเข้าไปกลับมา โดยโปรแกรมจะมีคำสั่งให้ผู้ใช้ป้อนข้อมูลเข้าไป และคอนโทรลเลอร์จะทำการประมลผลและส่งค่าดังกล่าวกลับมาพร้อมทั้งแสดงผลให้ผู้ใช้งานเห็น แต่ข้อจำกัดของโปรแกรมดังกล่าวคือ สามารถรับและส่งข้อมูลได้เพียงตัวอักษรเดียวเท่านั้น

จากการวิเคราะห์ข้อมูลข้างต้นทำให้เห็นว่า Modbus Protocol มีความเหมาะสมสำหรับการติดต่อสื่อสารระหว่างผู้ใช้งานและอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ โดยการใช้คอนโทรลเลอร์ชนิด DSP2808 และ Truck Cacle แบบ RS2232 นั้น สามารถทำได้เพียงการติดต่อสื่อสารระหว่างอุปกรณ์อิเล็กทรอนิกส์ที่มีการเชื่อมต่อแบบจุดต่อจุดในระยะใกล้ๆและมีจำนวนข้อมูลไม่มากเท่านั้น ไม่สามารถควบคุมการสื่อสารที่มีความซับซ้อนและอยู่ในระยะไกลได้ ดังนั้นในการทำงานที่ต้องการความสามารถทางการสื่อสารข้อมูลที่เพิ่มขึ้น อาจจะมีการพัฒนาทางด้านโปรแกรมหรือการเชื่อมรูปแบบต่าง ๆ เพื่อให้ได้งานที่ตรงตามจุดประสงค์มากที่สุด

4.3 วิเคราะห์และวิจารณ์ข้อมูลจากการทำโครงงาน

ในการดำเนินงานได้มีการกำหนดวัตถุประสงค์และจุดมุ่งหมายในการทำโครงงานไว้ แต่เนื่องจากมีระยะเวลาในการดำเนินการที่จำกัด จึงอาจทำให้ข้อมูลในบางส่วนของโครงงานไม่เป็นไปตามวัตถุประสงค์ที่ได้ตั้งไว้ หรืออาจไม่ครบถ้วนตามหลักการทั้งหมดของเรื่องที่ทำการศึกษา โดยในโครงงานเป็นการศึกษาเกี่ยวกับการติดต่อสื่อสารของ Modbus Protocol บนอุปกรณ์อิเล็กทรอนิกส์ ซึ่งการติดต่อสื่อสารในรูปแบบของ Modbus Protocol ยังสามารถทำได้อีกหลายวิธี และโปรโตคอลได้มีการพัฒนาไปมากจากเดิม ทำให้มีวิวัฒนาการใหม่ๆออกมา

ภายในโครงงานมีการศึกษาข้อมูลของ Modbus Protocol ไว้อย่างละเอียด แต่ในการทดลองใช้งานได้ทำเพียงหนึ่งวิธีเท่านั้น คือการทำงานของ Modbus Protocol ภายใต้การควบคุมของคอนโทรลเลอร์ชนิด DSP2808 โดยใช้โปรแกรมตัวอย่าง SCI Echoback ซึ่งในการใช้งานจริงอาจจะมีโปรแกรมต่างๆอีกมากมายที่สามารถทำการประมวลผลบน DSP2808 ได้ ด้วยเหตุนี้อาจจะเป็นผลให้การวิเคราะห์ข้อมูลต่าง ๆ คลาดเคลื่อนไปจากความเป็นจริง และในการใช้อุปกรณ์เชื่อมต่อระหว่าง Device และ Terminal ได้ทำการทดลองเพียง RS232 Interface เท่านั้น ยังมีอุปกรณ์เชื่อมต่ออีกหลายชนิดบน Modbus Protocol ที่สามารถใช้งานได้ แต่เนื่องด้วยข้อจำกัดของเวลาจึงทำให้ไม่สามารถศึกษาประเด็นอื่น ๆ ได้ทัน ทั้งนี้ทั้งนั้นผลที่ได้รับและข้อสรุปต่างๆภายในโครงงานเป็นการใช้ข้อมูลที่ได้จากการศึกษาในครั้งนี้เท่านั้น

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินการ

กระบวนการทำงานในเรื่องของการศึกษาระบบการติดต่อสื่อสารและการควบคุมด้วย Modbus Protocol ทำให้ทราบว่า

- เป็นโปรโตคอลสำหรับการส่งผ่านข้อมูลอินพุต/เอาท์พุต
- เป็นโปรโตคอลระบบเปิด (ไม่มีค่าใช้จ่าย เชื่อมต่อและพัฒนาง่าย)
- ลักษณะการสื่อสารเป็นแบบ Master/Slave
- รูปแบบการส่งแบบ Query-Response Cycle
- โหมดสำหรับการส่งผ่านข้อมูล RTU/ASCII
- ควบคุมการทำงานโดยการเขียนโปรแกรมลงบนคอนโทรลเลอร์

เมื่อถึงขั้นตอนการเขียนโปรแกรมควบคุมคอนโทรลเลอร์ โดยคอนโทรลเลอร์ที่ใช้เป็น TMS320F2808 ทดลองโดยใช้การติดต่อระหว่าง Hyperterminal และคอนโทรลเลอร์ ผ่านการ Serial Port ชนิด RS232 ประมวลผลโปรแกรมที่มีชื่อว่า SCI Echoback ผลจาการประมวลผลโปรแกรมคือ คอนโทรลเลอร์จะทำการสะท้อนข้อมูลที่ผู้ใช้งานป้อนเข้าไปกลับมา ซึ่งการทดลองเขียนโปรแกรมดังกล่าวจะนำไปประยุกต์ใช้กับการเขียนโปรแกรมควบคุมการทำงานของอินเวอร์เตอร์

5.2 แนวทางการแก้ไขปัญหา

- เนื่องจากการเขียนโปรแกรมด้วย TMS320F2808 เป็นความรู้ใหม่ จึงต้องทำการศึกษาหาความรู้เพิ่มเติมที่มีความจำเป็นในการใช้คอนโทรลเลอร์ TMS320F2808 เช่น การศึกษาจาก Header หรือโปรแกรมตัวอย่าง
- ควรแบ่งหัวข้อสำหรับการศึกษาข้อมูลที่มีความจำเป็นในการทำโครงงาน รายละเอียดเพิ่มเติมอื่นๆ อาจมีการศึกษาเพิ่มเติมหลังจากที่การทำโครงงานเสร็จสมบูรณ์แล้ว
- อาจมีการนำโครงงานไปศึกษาต่อยอดในรุ่นต่อๆไป เพื่อทำให้โครงงานมีการศึกษาที่ครอบคลุมเนื้อหาและรายละเอียดมากยิ่งขึ้น

5.3 ข้อเสนอแนะจากการดำเนินงาน

จากการศึกษาระบบการสื่อสารในงานอุตสาหกรรม ได้เล็งเห็นแล้วว่า MODBUS PROTOCOL เป็น PROTOCOL ที่มีความเหมาะสมในการทำงานร่วมกับเครื่องอินเวอร์เตอร์มากที่สุด เดิมทีนั้นสถานประกอบการได้ใช้ทั้ง MODBUS และ PROFIBUS เป็นโปรโตคอลเพื่อการผลิตอินเวอร์เตอร์ ซึ่งมีข้อเสียคือ

- สิ้นเปลืองต้นทุนการผลิต
- การใช้งานร่วมกันเป็นไปได้ยาก
- การพัฒนาเป็นไปได้ยาก
- เสียเวลาในการศึกษาข้อมูล

ทั้งนี้ทั้งนั้น จากการศึกษาข้อมูลทั้งหมด สถานประกอบการควรเลือกใช้แค่โปรโตคอลใดโปรโตคอลหนึ่ง เพื่อเป็นการลดต้นทุนการผลิต และทำให้การพัฒนาผลงานต่างๆเป็นไปได้โดยง่าย โดยที่ไม่ต้องเสียเวลาในการศึกษาข้อมูลของโปรโตคอลต่างๆหลายโปรโตคอล

และจากข้อมูลที่ได้ศึกษาทั้งหมดทำให้ทราบว่า Modbus Protocol ได้มีการพัฒนาไปเป็นโปรโตคอลที่ความสามารถมากขึ้นอีกหลายโปรโตคอลด้วยกัน ยกตัวอย่างเช่น MODBUS ASI เป็นต้น ข้อดีของ MODBUS ASI คือ

- ลดจำนวนสายเชื่อมต่อ
- สามารถทำการเชื่อมต่อได้ง่ายขึ้น
- ลดต้นทุนในการผลิต

ซึ่งหากสถานประกอบการนำ MODBUS ASI ไปใช้ในการผลิตด้วยแล้วนั้น จะเป็นการลดความยุ่งยากซับซ้อนในการต่อสายเชื่อมต่อ และเป็นการลดต้นทุนการผลิตในส่วนของสายเชื่อมต่อได้อีกด้วย ทั้งนี้สถานประกอบการควรมีการศึกษาเทคโนโลยีใหม่ ๆ อย่างสม่ำเสมอ เนื่องจากในปัจจุบันสิ่งต่าง ๆ ได้พัฒนาไปอย่างรวดเร็ว

เอกสารอ้างอิง

1. Texas Instrument Incorporated, **TMS320F2808 Digital Signal Processor** [Online], Available: http://focus.ti.com/docs/prod/folders/print/tms320f2808.html [2012, June 22]

2. Texas Instrument Incorporated, **DSP System Control and Interrupt**[Online], Available: http://DSP.ti.com [2012, June 22]

3. Texas Instrument Incorporated, **TRS 232**[Online], Available: http://focus.ti.com/docs/prod/folders/print/trs232.html [2012, June 22]

4. Modicon Incorporated, 1996, **Modicon Protocol Reference Guide:PI-MBUS-300 Rev J.** [Online], Available: http://www.scribd.com/docs/3910148/PI-MBUS-300 [2012, July 23]

5. นายวินิต โวกสูงเนิน, **Protocol MODBUS**, Available: http://www.kmilt.ac.th/~kstaweep/fa/file/fas/intro-modbus.pdf [2012 ,July 23]

6. **MODBUS Protocol**, Available: http://www.kmilt.ac.th/~kstaweep/fa/file/fas/modbus.pdf [2012 ,July 23]

7. **MODBUS Protocol_4A**, Available: http://eu.lip.kmutt.ac.th/elearning/lms/filemanager/uploadfile/handout/338/Modbus_4A.pdf [2012 ,July 23]

เอกสารอ้างอิง(ต่อ)

8. **MODBUS Protocol_4B**, Available: http://eu.lip.kmutt.ac.th/elearning/lms/filemanager/uploadfile/handout/338/Modbus_4B.pdf [2012 ,July 23]

9. นายพิชิต จินตโกศลวิทย์, **การสื่อสารข้อมูลในงานอุตสาหกรรมไทย** [Online], Available: http://www.thailandindustry.com/guru/view.php?id=7318&section=9&rcount=Y [2012, Sep 3]

10. **PLC คืออะไร,** Available: http://www.rtafshooting.com/arm/index.php?option=com_content&view=article&id=83:pk-&catid=38:28Itemid=41 [2012, Sep 3]

11. Pongsakriverplus, 2011**, PLC Protocol : การสื่อสารแบบ Profibus** [Online], Available: http://reverplusblog.com/2011/08/18/plc-protocol-การสื่อสารแบบ-profibus-protocol/ [2012, Sep 3]

12. pongsakriverplus, **PLC Protocol : การสื่อสารแบบ Modbus** [Online], Available: http://reverplusblog.com/2011/08/18/plc-protocol-การสื่อสารแบบ-Modbus-protocol/ [2012, Sep 3]

ภาคผนวก ก

รูปแบบของ Exception Response

รูปแบบของ Exception Response

ข้อยกเว้นสำหรับการร้องขอข้อมูลแบบ Broadcast Message เมื่อ Master Device ส่ง Query ไปยัง Slave Device ซึ่งคาดว่าเป็นการร้องขอแบบปกติ เหตุการณ์ที่อาจเกิดขึ้นจาก Master's Query มีด้วยกัน 4 เหตุการณ์ ดังต่อไปนี้

- หาก Slave Device ไม่ได้รับ Query ที่เป็นการติดต่อสื่อสารที่ผิดพลาด Slave Device จะสามารถจัดการกับ Query ได้อย่างปกติ มันจะส่งค่า Response กลับไปเป็นปกติ
- หาก Slave Device ได้รับ Query ที่ถูกตรวจพบการสื่อสารที่ผิดพลาด โดยมันสามารถจัดการกับ Query ได้อย่างปกติ Master Program จะประมวลผล Timeout ที่มีเงื่อนไขสำหรับ Query ดังกล่าว
- หาก Slave Device ได้รับ Query แต่ถูกตรวจพบการสื่อสารที่ผิดพลาด (จากการตรวจสอบแบบ Parity, LRC, CRC) Slave Device จะไม่มีการส่งค่าใดๆกลับ และ Master Program จะทำการประมวลผล Timeout ที่มีเงื่อนไขสำหรับ Query ดังกล่าว
- หาก Slave Device รับ Query ที่ไม่ตรวจพบการสื่อสารที่ผิดพลาด แต่ไม่สามารถจัดการกับมันได้ (เช่น ถ้าการร้องขอเป็นการอ่าน Non-Existent Coil หรือ Register) Slave Device จะกลับไปอยู่ในสถานะ Exception Response และจะแจ้งไปยัง Master Device เกี่ยวกับลักษณะของข้อผิดพลาดดังกล่าว

Exception Response Message จะมีด้วยกัน 2 Field ที่แตกต่างไปจาก Normal Response คือ

1). Function Code Field

ใน Normal Response การสะท้อนข้อมูลกลับของ Slave นั้น Function Code ของ Query เดิมที่อยู่ภายใน Function Code Field ของ Response ซึ่งทุก ๆ Function Code จะมี Most-Significant Bit (MSB) ของ 0 (เป็นค่าที่อยู่ภายใต้จำนวน 80 ในระบบเลขฐานสิบหก) ใน Exception Response นั้น Slave จะตั้งค่า MSB ของ Function Code ไว้เป็น 1 ซึ่งจะทำให้ค่าของ Function Code ใน Exception Response เป็นจำนวน 80 ในระบบเลขฐานสิบหกอย่างแน่นอน โดยค่าดังกล่าวจะสูงกว่าค่าที่มีไว้สำหรับ Normal Response

จากการตั้งค่า MSB ของ Function Code นั้น Application Program ของ Master สามารถรับรู้ Exception Response และสามารถตรวจสอบ Data Field สำหรับ Exception Code ได้

2). Data Field

ใน Normal Response นั้น Slave อาจจะกลับข้อมูลหรือสถิติใน Data Field (ข้อมูลต่าง ๆ ที่ได้ร้องขอไปใน Query) และใน Exception Code Response นั้น Slave จะกลับ Exception Code ใน Data Field ที่เป็นการระบุ Slave อย่างมีเงื่อนไขอันเนื่องมาจากผลของ Exception

รูปที่ ก.1 แสดงตัวอย่างของ Master Query ที่เป็นการระบุ Slave Exception Response ซึ่งตัวอย่างของ Field จะแสดงในรูปของเลขฐานสิบหก

QUERY

Byte	Contents	Example
1	Slave Address	0A
2	Function	01
3	Starting Address Hi	04
4	Starting Address Lo	A1
5	No. of Coils Hi	00
6	No. of Coils Lo	01
7	LRC	4F

EXCEPTION RESPONSE

Byte	Contents	Example
1	Slave Address	0A
2	Function	81
3	Exception Code	02
4	LRC	73

รูปที่ ก.1 การร้องขอและการตอบกลับของ Master และ Slave

ในตัวอย่างข้างต้นนี้ Master Address จะสอบถามไปยัง Slave Device 10 (0A Hex) และ Function code 01 สำหรับสถานะการทำงานของ Read Coil มันจะส่งสถานะการร้องขอของ Coil ที่ Address 1245 (04A1 Hex) และจะมีเพียง 1 Coil เท่านั้นที่จะถูกอ่าน อย่างเช่นการระบุโดยตัวเลขของ Coil Field

หาก Coil Address เป็น Non-Existent ใน Slave Device แล้ว Slave จะกลับไปเป็น Exception Response และ Exception Code จะแสดงค่า 02 ซึ่งเป็นการระบุที่ผิดกฎของ Data Address สำหรับ Slave ตัวอย่างเช่น หาก Slave คือ 984-385 กับ 512 Coil รหัสดังกล่าวจะเป็นค่าที่ถูกส่งกลับไปยัง Master

Exception Code

ตาราง ก.1 ความหมายของ Exception code

รหัส	ชื่อ	ความหมาย
01	ILLEGAL FUNCTION	Function Code ที่รับใน Query ซึ่งไม่ได้รับการอนุญาตให้มีการกระทำบน Slave ถ้าหาก Poll Program ดำเนินการตามคำสั่งสำเร็จ โค้ดดังกล่าวจะบ่งบอกว่าไม่มีโปรแกรมใดๆจะสำคัญกว่ามัน
02	ILLEGAL DATA ADDRESS	จะรับ Data Address ใน Query ที่ไม่ยอมให้ที่อยู่สำหรับ Slave
03	ILLEGAL DATA VALUE	ค่าที่บรรจุใน query data field ที่ไม่อนุญาตให้ค่าสำหรับ slave
04	SLAVE DEVICE FAILURE	ค่าความผิดพลาดที่เหิดขึ้นไม่สามารถกู้คืนได้ในขณะที่ Slave พยายามที่จะดำเนินการร้องขอการกระทำ
05	ACKNOWLEDGE	Slave มีการยอมรับการร้องขอและประมวลผลการร้องขอดังกล่าว แต่ต้องการระยะเวลาที่ยาวนานในการทำตามคำสั่งดังกล่าว ซึ่ง response นี้เป็นการกลับไปขัดขวาง timeout error ที่กำลังเกิดขึ้นใน Master และ master สามารถตัดสินใจที่จะดำเนินการโปรแกรมถัดไปได้สำเร็จถ้าหากการประมวลผลสำเร็จ
06	SLAVE DEVICE BUSY	Slave ถูกจองในช่วงเวลาที่ยาวนานในการประมวลผลโปรแกรมคำสั่ง ซึ่ง master ควรจะทำการส่งใหม่อีกครั้งในภายหลังเมื่อ slaveเป็นอิสระ
07	NEGATIVE ACKNOWLEDGE	Slave ไม่สามารถดำเนินการรับ program function ใน query ได้ โดยโค้ดนี้จะกลับไปเพื่อร้องขอโปรแกรมที่ยังไม่สำเร็จโดยใช้ function code 13 หรือ 14 ซึ่งเป็นเลขฐานสิบ และ master ควรวินิจฉัยการส่งคำร้องหรือความผิดพลาดของข้อมูลจาก Slave

ตาราง ก.1 ความหมายของ Exception code (ต่อ)

รหัส	ชื่อ	ความหมาย
08	MEMORY PARITY ERROR	Slave มีความพยามที่จะขยายหน่วยความจำในการอ่าน และ master สามารถทำการร้องขอได้อีกครั้ง แต่ service อาจจะต้องการเพียง slave device

ภาคผนวก ข

การกำหนดค่า

SCI Communication Control Register (SCICCR).

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	LOOPBACK ENA	ADDR/IDLE MODE	SCICHAR2	SCICHAR1	SCICHAR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

รูปที่ ข.1 รูปแบบของ SCI Communication Control Register (SCICCR) -Address 7050h

ตาราง ข.1 การกำหนดค่าของ SCI Communication Control Register (SCICCR) Field

Bit	Field	Value	Description
7	Reserved		Reads return zero; writes have no effect.
6	RX ERR INT ENA		SCI receive error interrupt enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring.
		0	Receive error interrupt disabled
		1	Receive error interrupt enabled
5	SW RESET		SCI software reset (active low). Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition. The SW RESET bit does not affect any of the configuration bits. All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, re-enable the SCI by writing a 1 to this bit. Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST, bit 5). SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Once SW RESET is asserted, the flags are frozen until the bit is deasserted. The affected flags are as follows:
			Value After SW RESET / SCI Flag / Register Bit 1 / TXRDY / SCICTL2, bit 7 1 / TX EMPTY / SCICTL2, bit 6 0 / RXWAKE / SCIRXST, bit 1 0 / PE / SCIRXST, bit 2 0 / OE / SCIRXST, bit 3 0 / FE / SCIRXST, bit 4 0 / BRKDT / SCIRXST, bit 5 0 / RXRDY / SCIRXST, bit 6 0 / RX ERROR / SCIRXST, bit 7
		0	Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition.
		1	After a system reset, re-enable the SCI by writing a 1 to this bit.
4	Reserved		Reads return zero; writes have no effect.
3	TXWAKE		SCI transmitter wake-up method select. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3)
		0	Transmit feature is not selected. In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In address-bit mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1
		1	Transmit feature selected is dependent on the mode, idle-line or address-bit: TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5); it is cleared by a system reset or the transfer of TXWAKE to the WUT flag.

ตาราง ข.1 การกำหนดค่าของ SCI Communication Control Register (SCICCR) Field (ต่อ)

Bit	Field	Value	Description
2	SLEEP		SCI sleep. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3). In a multiprocessor configuration, this bit controls the receiver sleep function. Clearing this bit brings the SCI out of the sleep mode. The receiver still operates when the SLEEP bit is set; however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5–2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.
		0	Sleep mode disabled
		1	Sleep mode enabled
1	TXENA		SCI transmitter enable. Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent.
		0	Transmitter disabled0
		1	Transmitter enabled
0	RXENA		SCI receiver enable. Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers). Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.
		0	Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers
		1	Send received characters to SCIRXEMU and SCIRXBUF

SCI Control Register1 (SCICTL1)

7	6	5	4	3	2	1	0
Reserved	RX ERR INT ENA	SW RESET	Reserved	TXWAKE	SLEEP	TXENA	RXENA
R-0	R/W-0	R/W-0	R-0	R/S-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.2 รูปแบบของ SCI Control Register 1 (SCICTL1) -Address 7051h

ตาราง ข.2 การกำหนดค่าของ SCI Control Register 1 (SCICTL1) Field

Bit	Field	Value	Description
7	Reserved		Reads return zero; writes have no effect.
6	RX ERR INT ENA		SCI receive error interrupt enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring.
		0	Receive error interrupt disabled
		1	Receive error interrupt enabled
5	SW RESET		SCI software reset (active low). Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition. The SW RESET bit does not affect any of the configuration bits. All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, re-enable the SCI by writing a 1 to this bit. Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST, bit 5). SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Once SW RESET is asserted, the flags are frozen until the bit is deasserted. The affected flags are as follows: Value After SW RESET / SCI Flag / Register Bit 1 / TXRDY / SCICTL2, bit 7 1 / TX EMPTY / SCICTL2, bit 6 0 / RXWAKE / SCIRXST, bit 1 0 / PE / SCIRXST, bit 2 0 / OE / SCIRXST, bit 3 0 / FE / SCIRXST, bit 4 0 / BRKDT / SCIRXST, bit 5 0 / RXRDY / SCIRXST, bit 6 0 / RX ERROR / SCIRXST, bit 7
		0	Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition.
		1	After a system reset, re-enable the SCI by writing a 1 to this bit.
4	Reserved		Reads return zero; writes have no effect.
3	TXWAKE		SCI transmitter wake-up method select. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3)
		0	Transmit feature is not selected. In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In address-bit mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1
		1	Transmit feature selected is dependent on the mode, idle-line or address-bit: TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5); it is cleared by a system reset or the transfer of TXWAKE to the WUT flag.

ตาราง ข.2 การกำหนดค่าของ SCI Control Register 1 (SCICTL1) Field (ต่อ)

Bit	Field	Value	Description
2	SLEEP		SCI sleep. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3). In a multiprocessor configuration, this bit controls the receiver sleep function. Clearing this bit brings the SCI out of the sleep mode. The receiver still operates when the SLEEP bit is set; however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5–2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.
		0	Sleep mode disabled
		1	Sleep mode enabled
1	TXENA		SCI transmitter enable. Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent.
		0	Transmitter disabled0
		1	Transmitter enabled
0	RXENA		SCI receiver enable. Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers). Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.
		0	Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers
		1	Send received characters to SCIRXEMU and SCIRXBUF

SCI Baud-Select Registers (SCIHBAUD, SCILBAUD)

15	14	13	12	11	10	9	8
BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.3 รูปแบบของ Baud-Select MSbyte Registers (SCIHBAUD) -Address 7052h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD10 (LSB)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.4 รูปแบบของ Baud-Select LSbyte Registers (SCILBAUD) -Address 7053h

ตาราง ข.4 การกำหนดค่าของ SCI Baud-Select Registers (SCIHBAUD, SCILBAUD) Field

Bit	Field	Value	Description
15-0	BAUD15– BAUD0		SCI 16-bit baud selection Registers SCIHBAUD (MSbyte) and SCILBAUD (LSbyte) are concatenated to form a 16-bit baud value, BRR. The internally-generated serial clock is determined by the low speed peripheral clock (LSPCLK) signal and the two baud-select registers. The SCI uses the 16-bit value of these registers to select one of 64K serial clock rates for the communication modes. The SCI baud rate is calculated using the following equation: $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{(\text{BRR} + 1) \times 8}$ (2-1) Alternatively, $\text{BRR} = \frac{\text{LSPCLK}}{\text{SCI Asynchronous Baud} \times 8} - 1$ (2-2) Note that the above formulas are applicable only when $1 \le \text{BRR} \le 65535$. If BRR = 0, then $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{16}$ (2-3) Where: BRR = the 16-bit value (in decimal) in the baud-select registers.

SCI Control Register2 (SCICTL2)

7	6	5 ... 2	1	0
TXRDY	TX EMPTY	Reserved	RX/BK INT ENA	TX INT ENA
R-1	R-1	R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.5 รูปแบบของ SCI Control Register2 (SCICTL2) -Address 7054h

ตาราง ข.4 การกำหนดค่าของ SCI Control Register2 (SCICTL2) Field

Bit	Field	Value	Description
7	TXRDY		Transmitter buffer register ready flag. When set, this bit indicates that the transmit data buffer register, SCITXBUF, is ready to receive another character. Writing data to the SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit, TX INT ENA (SCICTL2.0), is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL1.5) or by a system reset.
		0	SCITXBUF is full
		1	SCITXBUF is ready to receive the next character
6	TX EMPTY		Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.5), or a system reset, sets this bit. This bit does not cause an interrupt request.
		0	Transmitter buffer or shift register or both are loaded with data
		1	Transmitter buffer and shift registers are both empty
5-2	Reserved		
1	RX/BK INT ENA		Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by either the RXRDY flag or the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BK INT ENA does not prevent the setting of these flags.
		0	Disable RXRDY/BRKDT interrupt
		1	Enable RXRDY/BRKDT interrupt
0	TX INT ENA		SCITXBUF-register interrupt enable.This bit controls the interrupt request caused by setting the TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set (being set indicates that register SCITXBUF is ready to receive another character).
		0	Disable TXRDY interrupt
		1	Enable TXRDY interrupt

SCI Receiver Status Register (SCIRXST)

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.7 รูปแบบของ SCI Receiver Status Register (SCIRXST) -Address 7055h

ตาราง ข.5 การกำหนดค่าของ SCI Receiver Status Register (SCIRXST) Field

Bit	Field	Value	Description
7	RX ERROR		SCI receiver error flag. The RX ERROR flag indicates that one of the error flags in the receiver status register is set. RX ERROR is a logical OR of the break detect, framing error, overrun, and parity error enable flags (bits 5–2: BRKDT, FE, OE, and PE). A 1 on this bit will cause an interrupt if the RX ERR INT ENA bit (SCICTL1.6) is set. This bit can be used for fast error-condition checking during the interrupt service routine. This error flag cannot be cleared directly; it is cleared by an active SW RESET or by a system reset.
		0	No error flags set
		1	Error flag(s) set
6	RXRDY		SCI receiver-ready flag. When a new character is ready to be read from the SCIRXBUF register, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by a reading of the SCIRXBUF register, by an active SW RESET, or by a system reset.
		0	No new character in SCIRXBUF
		1	Character ready to be read from SCIRXBUF
5	BRKDT		SCI break-detect flag. The SCI sets this bit when a break condition occurs. A break condition occurs when the SCI receiver data line (SCIRXD) remains continuously low for at least ten bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset.
		0	No break condition
		1	Break condition occurred
4	FE		SCI framing-error flag. The SCI sets this bit when an expected stop bit is not found. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. The FE bit is reset by a clearing of the SW RESET bit or by a system reset.
		0	No framing error detected
		1	Framing error detected
3	OE		SCI overrun-error flag. The SCI sets this bit when a character is transferred into registers SCIRXEMU and SCIRXBUF before the previous character is fully read by the CPU or DMAC. The previous character is overwritten and lost. The OE flag bit is reset by an active SW RESET or by a system reset.
		0	No overrun error detected
		1	Overrun error detected

ตาราง ข.5 การกำหนดค่าของ SCI Receiver Status Register (SCIRXST) Field (ต่อ)

Bit	Field	Value	Description
2	PE		SCI parity-error flag. This flag bit is set when a character is received with a mismatch between the number of 1s and its parity bit. The address bit is included in the calculation. If parity generation and detection is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an active SW RESET or a system reset.!
		0	No parity error or parity is disabled
		1	Parity error is detected
1	RXWAKE		Receiver wake-up-detect flag
		0	No detection of a receiver wake-up condition
		1	A value of 1 in this bit indicates detection of a receiver wake-up condition. In the address-bit multiprocessor mode (SCICCR.3 = 1), RXWAKE reflects the value of the address bit for the character contained in SCIRXBUF. In the idle-line multiprocessor mode, RXWAKE is set if the SCIRXD data line is detected as idle. RXWAKE is a read-only flag, cleared by one of the following: • The transfer of the first byte after the address byte to SCIRXBUF (only in non-FIFO mode) • The reading of SCIRXBUF • An active SW RESET • A system reset
0	Reserved		Reads return zero; writes have no effect.

SCI Receive Data Buffer Register (SCIRXBUF)

15	14	13	8
SCIFFFE[1]	SCIFFPE[1]	Reserved	
R–0	R–0	R–0	

7	6	5	4	3	2	1	0
RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

(1) Applicable only if the FIFO is enabled.

รูปที่ ข.7 รูปแบบของ SCI Receive Data Buffer Register (SCIRXBUF) -Address 7057h

ตาราง ข.6 การกำหนดค่าของ SCI Receive Data Buffer Register (SCIRXBUF) Field

Bit	Field	Value	Description
15	SCIFFFE		SCIFFFE. SCI FIFO Framing error flag bit (applicable only if the FIFO is enabled)
		0	No frame error occurred while receiving the character, in bits 7–0. This bit is associated with the character on the top of the FIFO.
		1	A frame error occurred while receiving the character in bits 7–0. This bit is associated with the character on the top of the FIFO.
14	SCIFFPE		SCIFFPE. SCI FIFO parity error flag bit (applicable only if the FIFO is enabled)
		0	No parity error occurred while receiving the character, in bits 7–0. This bit is associated with the character on the top of the FIFO.
		1	A parity error occurred while receiving the character in bits 7–0. This bit is associated with the character on the top of the FIFO.
13-8	Reserved		
7-0	RXDT7–0		Receive Character bits

ข.7 SCI FIFO Transmit (SCIFFTX)

15	14	13	12	11	10	9	8
SCIRST	SCIFFENA	TXFIFO Reset	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W-0	R/W-1	R–0	R–0	R–0	R–0	R–0

7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R–0	W–0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

รูปที่ ข.8 รูปแบบของ SCI FIFO Transmit (SCIFFTX) -Address 705Ah

ตาราง ข.7 การกำหนดค่าของ SCI FIFO Transmit (SCIFFTX) Field

Bit	Field	Value	Description
15	SCIRST		SCI Reset
		0	Write 0 to reset the SCI transmit and receive channels. SCI FIFO register configuration bits will be left as is.
		1	SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work.
14	SCIFFENA		SCI FIFO enable
		0	SCI FIFO enhancements are disabled
		1	SCI FIFO enhancements are enabled
13	TXFIFO Reset		Transmit FIFO reset
		0	Reset the FIFO pointer to zero and hold in reset
		1	Re-enable transmit FIFO operation
12-8	TXFFST4-0	00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 words
		00010	Transmit FIFO has 2 words
		00011	Transmit FIFO has 3 words
		0xxxx	Transmit FIFO has x words
		10000	Transmit FIFO has 16 words
7	TXFFINT Flag		Transmit FIFO interrupt
		0	TXFIFO interrupt has not occurred, read-only bit
		1	TXFIFO interrupt has occurred, read-only bit
6	TXFFINT CLR		Transmit FIFO clear
		0	Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero
		1	Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA		Transmit FIFO interrupt enable
		0	TX FIFO interrupt based on TXFFIVL match (less than or equal to) is disabled
		1	TX FIFO interrupt based on TXFFIVL match (less than or equal to) is enabled.
4-0	TXFFIL4-0		TXFFIL4–0 Transmit FIFO interrupt level bits. Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4–0) and FIFO level bits (TXFFIL4–0) match (less than or equal to).
			Default value should be 0x00000.

SCI FIFO Receive (SCIFFRX)

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVR CLR	RXFIFO Reset	RXFIFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W–0	R/W–1	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W–0	R/W–0	R/W–1	R/W–1	R/W–1	R/W–1	R/W–1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

รูปที่ ข.9 รูปแบบของ SCI FIFO Receive (SCIFFRX) -Address 705Bh

ตาราง ข.8 การกำหนดค่าของ SCI FIFO Receive (SCIFFRX) Field

Bit	Field	Value	Description
15	RXFFOVF		Receive FIFO overflow. This will function as flag, but cannot generate interrupt by itself. This condition will occur while receive interrupt is active. Receive interrupts should service this flag condition.
		0	Receive FIFO has not overflowed, read-only bit
		1	Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost
14	RXFFOVF CLR		RXFFOVF clear
		0	Write 0 has no effect on RXFFOVF flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset		Receive FIFO reset
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable receive FIFO operation
12-8	RXFFST4–0	00000	Receive FIFO is empty
		00001	Receive FIFO has 1 word
		00010	Receive FIFO has 2 words
		00011	Receive FIFO has 3 words
		0xxxx	Receive FIFO has x words
			Receive FIFO has 16 words 10000
7	RXFFINT		Receive FIFO interrupt
		0	RXFIFO interrupt has not occurred, read-only bit
		1	RXFIFO interrupt has occurred, read-only bit
6	RXFFINT CLR		Receive FIFO interrupt clear
		0	Write 0 has no effect on RXFIFINT flag bit. Bit reads back a zero.
		1	Write 1 to clear RXFFINT flag in bit 7
5	RXFFIENA		Receive FIFO interrupt enable
		0	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled
		1	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled.
4-0	RXFFIL4–0		Receive FIFO interrupt level bits
		11111	Receive FIFO generates interrupt when the FIFO status bits (RXFFST4–0) and FIFO level bits (RXFFIL4–0) match (i.e., are greater than or equal to). Default value of these bits after reset – 11111. This will avoid frequent interrupts, after reset, as the receive FIFO will be empty most of the time.

ประวัติผู้จัดทำโครงงาน

ชื่อ-สกุล	นางสาวรสวันต์ รองวงศ์
วัน เดือน ปีเกิด	12 มีนาคม 2533
ประวัติการศึกษา	
ระดับประถมศึกษา	ประถมศึกษาตอนปลาย พ.ศ. 2540 โรงเรียนวัดแม่เปียะ
ระดับมัธยมศึกษา	มัธยมศึกษาตอนต้น-มัธยมศึกษาตอนปลาย พ.ศ. 2546 โรงเรียนจุฬาภรณราชวิทยาลัย ตรัง
ระดับอุดมศึกษา	อุดมศึกษาชั้นปีที่ 1-4 พ.ศ. 2552-ปัจจุบัน คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ทุนการศึกษา	-ไม่มี-
ประวัติการฝึกอบรม	Modbus Protocol ณ บริษัท โนเวมเอนจิเนียริ่ง จำกัด
ผลงานที่ได้รับการตีพิมพ์	-ไม่มี-