หุ่นจำลองระบบลิฟต์ด้วย PLC

ELEVATOR SIMULATOR USING PLC

นายชาคริต ยาโน

ว่าที่ ร.ต.ณรงค์ศักดิ์ ทองแก้ว

นายพิษณุ พันธ์อุบล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ELEVATOR SIMULATOR USING PLC

CHAKRIT YANO

NARONGSAK THONGKAEW

PISANU PUNOUBOL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2003

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ หุ่นจำลองระบบลิฟต์ด้วย PLC

ELEVATOR SIMULATOR USING PLC

นักศึกษาผู้จัดทำ	นายชาคริต	ยาโน	รหัสประจำตัว	44015512
	ว่าที่ร.ต.ณรงค์ศักดิ์	ทองแก้ว	รหัสประจำตัว	44015515
	นายพิษณุ	พันธ์อุบล	รหัสประจำตัว	44015524

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2546

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.วิริยะ กองรัตน์	

วัน/เดือน/ปี ที่สอบ วันอังคารที่ 23 มีนาคม พ.ศ. 2547

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(รศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	หุ่นจำลองระบบลิฟต์ด้วย PLC
	ELEVATOR SIMULATOR USING PLC
นักศึกษาผู้จัดทำ	นายชาคริต ยาโน
	ว่าที่ ร.ต.ณรงค์ศักดิ์ ทองแก้ว
	นายพิษณุ พันธ์อุบล
อาจารย์ที่ปรึกษา	รศ.วิริยะ กองรัตน์
ปีการศึกษา	2546

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นงานวิจัย หุ่นจำลองระบบลิฟต์ด้วยPLC ซึ่งเป็นการจำลองการทำงานของลิฟต์ในลักษณะของ (Warehouse)เพื่อแสดงให้ผู้ใช้เห็นสภาวะการทำงานของฮาร์ดแวร์ผ่านทางหน้าจอคอมพิวเตอร์และเพิ่มความสะดวกต่อการดูแลควบคุม โดยในการสร้างโครงงานนี้จะใช้ชุดควบคุมระยะไกลเข้ามาจำลองการเชื่อมต่อกับฮาร์ดแวร์ ซึ่งชุดควบคุมระยะไกลนี้จะประกอบด้วยชุดอินพุท (RTI16) และเอาท์พุท (RTO16)โมดูล เป็นอินพุท-เอาท์พุทใช้รับ-ส่งข้อมูลที่ต้องการในระยะไกล โดยสามารถตัดปัญหาข้อยุ่งยากทางด้านอินเตอร์เฟสของสัญญาณออกไป สามารถนำมาใช้ในการควบคุมระบบลิฟต์หรืออุปกรณ์ไฟฟ้าต่างๆในอาคารได้แบบอัตโนมัติ ซึ่งอาศัยการสื่อสารข้อมูลผ่านพอร์ตอนุกรม RS-232C และ RS485 มีคอมพิวเตอร์เป็นตัวประมวลผลกลางภายใต้ข้อตกลงในการสื่อสารข้อมูลเดียวกัน (PROTOCAL) ร่วมกับการพัฒนาโปรแกรมประยุกต์ (MICROSOFT VISUAL BASIC) ขึ้นมาสนับสนุนการทำงานในส่วนของการแสดงผลการเคลื่อนที่และการทำงานของลิฟต์ในระบบ ทำให้ง่ายต่อการควบคุมอีกทั้งมีความยืดหยุ่นสูง เพื่อแก้ปัญหาการสร้างวงจรอิเล็กทรอนิกส์ที่ยุ่งยาก การใช้สายไฟเป็นจำนวนมาก ง่ายต่อการติดตั้งและช่วยประหยัดค่าใช้จ่าย

Thesis Title Elevator Simulator Using PLC

Authors Mr. Chakrit Yano

Sub LT. Narongsak Thongkaew

Mr. Pisanu Panoubol

Thesis Advisor Assoc.Prof.Viriya Kongrat

Year 2003

ABSTRACT

In this Thesis Title, present the elevator simulator that using programmable logic controller (PLC) and remote I/O module is simulated status software display of the warehouse. The controller and the remote I/O module should have the same protocol are interfaced through RS485 with Microsoft Visual Basic software support. Each remote I/O module consists of 16 digital input and 16 digital outputs, and the maximum of 32 remote I/O modules can be linked to one controller. The remote I/O is linked for interrupt request to controller independently. Therefore, there is no affect to the scan time of the controller. Using this technique, the PLC can be efficiently applied to the several hundred meters, for differential control point and easy to installation more than logic circuits control.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีเพราะ ได้รับความกรุณาและความเมตตาจาก รองศาสตราจารย์ วิริยะ กองรัตน์ และท่านอาจารย์ทวีพล ซื่อสัตย์ ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์ในการทำปริญญานิพนธ์นี้ ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกๆท่านที่ได้ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และที่ลืมเสียมิได้คือ ขอกราบคุณพ่อคุณแม่ อันเป็นที่รักยิ่ง ที่สนับสนุนและเป็นแรงบันดาลใจในการทำ ปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ทางผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกๆท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

ในปัจจุบันการดำเนินชีวิตของมนุษย์เราเกี่ยวข้องกับการเคลื่อนที่ โดยมีรูปแบบต่างๆกัน อาจเป็นการวิ่ง การเดิน การใช้ยานพาหนะหรือวิธีการอื่นๆในการเดินทางซึ่งรวมถึงการใช้ลิฟต์ เพราะอาคารหรือตึกส่วนมากในปัจจุบันนั้นต้องทำหลายชั้นเพื่อการใช้งานพื้นที่การใช้งานให้คุ้มค่าที่สุดทำให้การเดินทางไปในชั้นต่างๆ ต้องพึ่งพาอาศัยอุปกรณ์อำนวยความสะดวกในการเดินทางสำหรับการเคลื่อนที่ไปในชั้นต่างๆดังนั้น ลิฟต์จึงมีบทบาทสำคัญในการเดินทางของมนุษย์ภายในอาคารและลิฟต์ยังสามารถนำมาใช้เคลื่อนย้ายสิ่งของไปยังชั้นต่างๆของอาคารได้ สิ่งสำคัญของลิฟต์ก็คือการควบคุมการเคลื่อนที่และการทำงานของลิฟต์

โดยอาคารหรือตึกที่มีความสูงมากกว่า 5 ชั้นขึ้นไปต้องทำการติดตั้งลิฟต์และภายในอาคารส่วนมากเราจะไม่เห็นการเคลื่อนที่ของลิฟต์จริงๆในระหว่างการเคลื่อนที่ไปในชั้นต่างๆเลย เราจะทราบการเคลื่อนที่ของลิฟต์ได้จากปุ่มไฟแสดงผลภายในและนอกลิฟต์เท่านั้น ยกเว้นลิฟต์ที่ทำเป็นแบบกระจกใส โครงการหุ่นจำลองระบบลิฟต์ด้วย PLC (Elevator Simulator Using PLC) นี้จะช่วยในการแสดงผลการเคลื่อนที่ของลิฟต์ทั้งตึกผ่านออกทางคอมพิวเตอร์ เพื่อให้เราทราบสภาวะการทำงานของลิฟต์ได้อย่างชัดเจน เพื่อที่ทำให้ผู้ที่ดูแลและทำการควบคุม สามารถทำการดูแลและตรวจสอบการทำงานของลิฟต์ได้ง่ายขึ้น แต่ในโครงงานจะใช้ชุดความคุมระยะไกลที่มีอินพุท (RTI16) และเอาท์พุท (RTO16) เป็นอินพุท-เอาท์พุทใช้รับ-ส่งข้อมูลที่ต้องการในระยะไกล โดยสามารถตัดปัญหาข้อยุ่งยากทางด้านอินเตอร์เฟสของสัญญาณออกไป สามารถนำมาใช้ในการควบคุมระบบลิฟต์หรืออุปกรณ์ไฟฟ้าต่างๆในอาคารแบบอัตโนมัติ ซึ่งอาศัยการสื่อสารข้อมูลผ่านพอร์ตอนุกรม RS-232C และ RS485 มีคอมพิวเตอร์เป็นตัวประมวลผลกลาง ภายใต้ข้อตกลงในการสื่อสารข้อมูลเดียวกัน (PROTOCAL) ร่วมกับ การพัฒนาโปรแกรมภาษาระดับสูงในการประยุกต์ (MICROSOFT VISUAL BASIC) ขึ้นมาสนับสนุนการทำงานในส่วนของการแสดงผลการเคลื่อนที่และการทำงานของลิฟต์ในระบบ ทำให้ง่ายต่อการควบคุมและความยืดหยุ่นสูง เพื่อแก้ปัญหาการสร้างวงจรอิเล็กทรอนิกส์ที่ยุ่งยาก การใช้สายไฟเป็นจำนวนมาก อีกทั้งง่ายต่อการติดตั้ง และช่วยประหยัดค่าใช้จ่าย

โครงการหุ่นจำลองระบบลิฟต์ด้วย PLC (Elevator Simulator Using PLC) เป็นการจำลองลิฟต์ของอาคาร 6 ชั้น โยการนำเอา I/O Module บวกกับชุดแผงสวิตซ์ควบคุมมาช่วยในการป้อนอินพุตพร้อมกับควบคุมและแสดงผลของโปรแกรม Visual Basic ซึ่งเป็นแบบจำลองลิฟต์ทั้งระบบรวมทั้งการนำค่าที่ได้ไปควบคุมอุปกรณ์ทางด้านเอาท์พุทรวมทั้งการแสดงผล ภาษาที่เลือกใช้ใน

การพัฒนาซอฟท์แวร์ในโครงการนี้ ได้แก่ Microsoft Visual Basic 6.0 เนื่องจากมีการติดต่อกับผู้ใช้เป็นแบบกราฟิก (Graphic User Interface) ทำให้สะดวกในการใช้งานมากกว่ารูปแบบการติดต่อแบบใช้ Text เพียงอย่างเดียว

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ช่วยเพิ่มประสิทธิภาพของการควบคุมให้มีระยะทางที่ไกลขึ้นกว่าเดิมที่ต้องควบคุมที่ตู้คอนโทรลหรือในระบบ อีกทั้งสามารถควบคุมได้ทั้งแบบอัตโนมัติและด้วยมือ

2. สามารถเขียนโปรแกรมภาษา VISUAL BASIC INTERFACE ระหว่าง PC กับ RTI16,RTO16 ได้

3. เพื่อพัฒนาการเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ควบคุมให้ทำงานได้ง่ายและมีประสิทธิภาพมากขึ้นโดยอาศัยการเขียนโปรแกรมภาษาบนเครื่องคอมพิวเตอร์

4. เพื่อให้เข้าใจเงื่อนไขในการติดต่อสื่อสารระหว่างอุปกรณ์ควบคุมกับคอมพิวเตอร์ผ่านเงื่อนไขการสื่อสาร (FAC-TALK PROTOCAL) เป็นสื่อกลางในการติดต่อสื่อสาร

5. สามารถสร้าง Graphic บนหน้าจอเพื่อดู Status ได้สวยงามและสามารถควบคุมบนหน้าจอได้

6. เพื่อเป็นแนวทางในการพัฒนาการควบคุมในระดับสูงที่มีการควบคุมเอาท์พุทภายนอกและการตรวจสอบอินพุทที่เข้ามาได้อย่างมีประสิทธิภาพ

1.3 ขอบเขตของปริญญานิพนธ์

1. สามารถควบคุมระบบการเคลื่อนที่และการทำงานของลิฟต์ภายในตัวอาคารทั้งแบบอัตโนมัติและด้วยมือได้ด้วยรีโมทเทอร์มินอล อินพุท เอาท์พุท (RTI16,RTO16) ผ่านทางพอร์ทอนุกรมได้

2. สามารถเขียนโปรแกรม Application Program (VB6) เพื่อเป็นตัวแสดงผลการเคลื่อนที่รวมทั้งระบบการทำงานต่างๆของลิฟต์ และเป็นตัวจัดการการทำงานของระบบ Building Automation ได้

3. ระบบสามารถนำเอาเอาท์พุทที่ได้ไปเป็นแนวทางในการควบคุมอุปกรณ์ไฟฟ้าต่างๆในระบบลิฟต์ได้

4. สามารถออกแบบระบบของลิฟต์ได้อย่างครอบคลุมทั้งการเคลื่อนที่ขึ้น-ลงการเปิดปิดประตู การแสดงสภาวะการทำงานรวมทั้งอุปกรณ์ป้องกันและตรวจจับต่างๆได้

5. สามารถนำโปรแกรมอื่นๆมาช่วยในการออกแบบกราฟฟิกและส่วนต่างๆของ Display ได้อย่างสวยงามและมีประสิทธิภาพ

1.4 ขั้นตอนการศึกษา

ในส่วนของขั้นตอนการศึกษานั้นสามารถแบ่งออกเป็น 3 ส่วนหลักๆ คือ

ส่วนแรกจะทำการวางแผนงาน โดยการกำหนดออกมาเป็นรูปแบบของตารางเวลาและหน้าที่ แล้วทำการค้นหาข้อมูลต่างๆที่เกี่ยวของกับการออกแบบระบบลิฟต์ อาทิเช่น ทฤษฎีการทำงานของลิฟต์เบื้องต้น ส่วนประกอบต่างๆของลิฟต์เพื่อให้เข้าใจถึงระบบการทำงานของลิฟต์ก่อนการสร้างเงื่อนไขเพื่อการออกแบบ โดยไปศึกษาจากสถานที่ที่มีลิฟต์ในหลายๆที่หลายๆแบบ และทำการเตรียมหาเอกสาร หนังสือที่เกี่ยวข้องกับการออกแบบทั้ง VB คู่มือของ I/O Module โปรแกรม Applications Display ต่างๆรวมทั้งตรวจสอบแผงอุปกรณ์รีโมทระยะไกล I/O Module ให้เรียบร้อย

ในส่วนที่สองเมื่อได้เก็บรวบรวมข้อมูลต่างๆที่ต้องการได้ในระดับหนึ่งก็ทำการศึกษาในส่วนของเงื่อนไขการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับ I/O Module โดยใช้โปรแกรม TERMINAL ใน WINDOWS โดยอาศัยการเขียน PROTOCAL ตามรูปแบบของ Block Command ของ Data Cheet ของ RTI16 และ RTO16 เมื่อทำการรับ-ส่งข้อมูลได้ ก็หันกลับมาศึกษา Apllication Program (VISUAL BASIC 6.0) โดยอาศัยศึกษาการเขียนข้อมูลรับ-ส่ง ออกSerial Port Com 1 โดยใช้ Component MS Comm.ที่มีมาให้ในตัวโปรแกรม

ส่วนที่สาม เป็นการศึกษาการ Application ว่าจะนำเอา Module ของ Remote Terminal Input & Output ไปใช้ควบคุมในระบบอะไรได้บ้าง เมื่อได้ข้อมูลและเข้าใจในตัว Hardware และ Soft Ware แล้ว ก็ทำการเขียนProgram ขึ้นมาสนับสนุน (Support) การทำงานของระบบที่ประยุกต์ (Application) ได้อย่างถูกต้อง

บทที่ 2
หลักการและทฤษฎีเบื้องต้น

2.1 โครงสร้างและการทำงานโดยทั่วไปของลิฟต์

ลิฟต์โดยทั่วไปจะมีโครงสร้างและองค์ประกอบที่มีลักษณะใกล้เคียงกัน กล่าวคือประกอบไปด้วยตัวลิฟต์ และน้ำหนักถ่วงแขวนติดด้วยกัน โดยลวดสลิงที่คล้องผ่านรอกซึ่งขับเคลื่อนรอกนี้ด้วยมอเตอร์ ซึ่งมีรายละเอียดดังนี้

1. ตัวลิฟต์ (Car) เป็นส่วนที่ใช้บรรทุกคนหรือสิ่งของซึ่งรวมทั้งพื้นที่ตัวลิฟต์ สาแหรก ห้องลิฟต์และประตูลิฟต์

2. ห้องลิฟต์ เป็นโครงสร้างและส่วนประกอบที่เป็นเพดานและผนังรอบๆ ตัวลิฟต์ซึ่งประกอบอยู่บนพื้นถังลิฟต์

3. สาแหรก (Car Frame) เป็นโครงสร้างซึ่งประกอบไปด้วยเหล็กคานบน เหล็กเสาข้างและเหล็กคานล่างยึดติดกันเป็นสาแหรกรองรับตัวลิฟต์ ที่สาแหรกจะมีตัวนำร่อง เครื่องนิรภัย ห่วง แขวนสลิงหรือโซ่และรองติดกันอยู่

4. ชุดควบคุม (Controller) เป็นอุปกรณ์หรือกลุ่มอุปกรณ์ที่ใช้บังคับการทำงานของอุปกรณ์สำเร็จต่างๆตามที่ได้กำหนดไว้

5. รางบังคับ (Guide Rails) คือรางที่บังคับการขึ้น-ลงของตัวลิฟต์หรือ น้ำหนักถ่วงตลอดแนวของตัวลิฟต์

6. ปล่องลิฟต์ (Hoist Way) เป็นส่วนของอาคารที่ออกแบบก่อสร้างไว้สำหรับติดตั้งลิฟต์ มีลักษณะเป็นปล่องทะลุติดกัน ระหว่างชั้นตลอดแนวความสูงที่ลิฟต์วิ่งขึ้น-ลง รวมทั้งส่วนที่เป็นลิฟต์ขึ้นไปจนถึงใต้พื้นห้องเครื่องหรือใต้พื้นหลังคา

7. บ่อลิฟต์ (Pit) หมายถึง ส่วนของปล่องลิฟต์ที่อยู่ใต้ระดับพื้นตัวลิฟต์จากชั้นล่างสุดไปจนถึงพื้นปล่องลิฟต์

8. น้ำหนักถ่วง (Counterweight) เป็นตุ้มน้ำหนักเพื่อถ่วงน้ำหนัก ของลิฟต์ในการวิ่งขึ้นลง

9. โครงสร้างบนปล่องลิฟต์ คือส่วนของโครงสร้างทั้งหมดรวมทั้งพื้นที่รองรับเครื่องลิฟต์และส่วนอื่นๆที่ติดบนปล่องลิฟต์

10. เครื่องกันปะทะ (Buffer) เป็นอุปกรณ์ที่ใช้ลดการกระแทก และหยุดการเคลื่อนที่ของลิฟต์เมื่อลิฟต์เคลื่อนที่เลยระดับต่ำสุด

2.1.1อุปกรณ์ของสัญญาณต่างๆที่ใช้ในลิฟต์ (Car Operating and Indicating Equipment) ประกอบไปด้วย

1.ปุ่มภายในห้องลิฟต์ (Car Operating Button) ใช้เลือกว่าจะให้ลิฟต์เคลื่อนที่ไปที่ชั้นใด จำนวนปุ่มนี้มีเท่ากับจำนวนชั้นที่มีความต้องการใช้ลิฟต์ ผู้ใช้สามารถกดปุ่มเรียกใช้ทีละปุ่มหรือหลายๆปุ่มพร้อมๆกันได้ ภายในปุ่มแต่ละปุ่มจะมีหลอดไฟแสดงการทำงาน ซึ่งหลอดไฟจะสว่างขึ้นเมื่อปุ่มถูกกด

2.ปุ่มหน้าชั้นลิฟต์ (Land Call Button) สำหรับแต่ละชั้น มีไว้กดเมื่อต้องการเรียกใช้ลิฟต์ที่หน้าลิฟต์ชั้นล่างสุดและชั้นบนสุดจะมีเพียงปุ่มเดียว คือปุ่มเรียกลิฟต์สำหรับทิศทางขึ้นสำหรับชั้นล่างสุด และปุ่มเรียกใช้ลิฟต์ ในทิศทางลงหน้าชั้นบนสุด ลักษณะของปุ่มจะเหมือนปุ่มที่ใช้ในลิฟต์

3.อุปกรณ์แสดงผลการเคลื่อนที่ของลิฟต์ (Direction Indicator) มี 3 กรณีคือ ขณะกำลังเคลื่อนที่ขึ้น ขณะกำลังเคลื่อนที่ลง หรือขณะกำลังจอดนิ่งอยู่กับที่ ในขณะที่ลิฟต์กำลังจอดนิ่งอยู่กับที่ และยังไม่มีการเรียกใช้อุปกรณ์แสดงผลจะไม่แสดงทิศทาง โดยลักษณะอุปกรณ์แสดงผลโดยมากจะเป็นหลอดไฟอยู่ภายในหน้าปัทม์ ที่มีสัญลักษณ์ลูกศรชี้ขึ้นและชี้ลงอย่างละตัว ซึ่งอุปกรณ์นี้จะติดตั้งไว้ที่หน้าลิฟต์ทุกชั้นและภายในลิฟต์ด้วย

4.อุปกรณ์แสดงสถานะของลิฟต์ (Position Indicator) เป็นอุปกรณ์แสดงสถานะของลิฟต์ว่าขณะนั้นลิฟต์กำลังจอดหรือเคลื่อนที่อยู่ที่ชั้นใด อุปกรณ์นี้จะติดตั้งอยู่ภายในตัวลิฟต์เพื่อแสดงสถานะของลิฟต์ให้ผู้โดยสารที่อยู่ภายในตัวลิฟต์ ทราบว่าลิฟต์กำลังอยู่ที่ชั้นใด

2.1.2 อุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่ (Speed Governor)

เป็นอุปกรณ์ที่ติดตั้งภายในห้องเครื่องลิฟต์มีหน้าที่ ในการตรวจจับความไวในการเคลื่อนที่ของลิฟต์ไม่ให้เคลื่อนที่ด้วยความเร็วเกินกว่าความเร็วที่ได้กำหนดไว้ โดยใช้ทฤษฎีของแรงเหวี่ยงหนีศูนย์กลาง ซึ่งหมุนได้โดยอาศัยสลิงที่ยึดติดกับตัวลิฟต์โดยตรงและมีลูกถ่วงอยู่ภายในบ่อลิฟต์คอยถ่วงให้สลิงตึงอยู่เสมอ

อุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่ประกอบไปด้วย สวิตซ์ไฟฟ้าและน้ำหนักกดสลิง (Catch Wight) สวิตซ์ไฟฟ้ามีหน้าที่ตัดกระแสไฟฟ้าที่ป้อนให้กับมอเตอร์ให้ผ่านระบบควบคุมลิฟต์ ส่วนน้ำหนักกดสลิงจะทำหน้าที่กดสลิงเพื่อให้สลิงกระชากเซฟตี้แคช (Safty catch) เพื่อบังคับไม่ให้ลิฟต์เคลื่อนที่ต่อไปได้ ซึ่งเป็นการป้องกันไม่ให้ลิฟต์ตกไปกระแทกบ่อลิฟต์

2.1.3 ประเภทของเครื่องขับลิฟต์

เครื่องขับลิฟต์ หมายถึง ตัวพลังงานที่ให้พลังงานในการขับเคลื่อนตัวลิฟต์โดยทั่วไปจะเป็นเครื่องขับลิฟต์ประเภทเครื่องขับไฟฟ้า (Electrical Driving Machine) ซึ่งมี มอเตอร์ ชุดเบรกและรอกขับเคลื่อน (Driving Sheave) หรือล้อแรง (Drum) พร้อมด้วยเฟือง เครื่องขับลิฟต์สามารถแบ่งเป็นประเภทได้ดังนี้

1. Direct – Drive Machine) หมายถึงเครื่องลิฟต์ขับเคลื่อนด้วยมอเตอร์ไฟฟ้า ที่ต่อโดยตรงกับรอกขับเคลื่อน ล้อแรงหรือเพลา โดยไม่ใช้สายหรือเฟืองกลาง ซึ่งสามารถแบ่งได้เป็น

1.1 เครื่องลิฟต์ขับเคลื่อนด้วยเฟือง (Gear Drive Machine) คือเครื่องลิฟต์ที่ขับเคลื่อนด้วยมอเตอร์ไฟฟ้าผ่านเฟืองไปหมุนรอกขับเคลื่อนล้อแรงหรือเพลา

1.2 เครื่องลิฟต์แรงความฝืด (Traction Machine) เป็นเครื่องลิฟต์ที่ขับเคลื่อนตัวลิฟต์โดยความฝืดระหว่างลวดแขวนกับรอก ซึ่งมีทั้งแบบใช้เฟืองและไม่ใช้เฟืองต่อระหว่างมอเตอร์และรอก

1.3 เครื่องลิฟต์รอกกว้าน (Winding Drum Machine) คือเครื่องลิฟต์ที่ใช้เฟืองในการขับเคลื่อนรอกกว้านลวดแขวนลิฟต์

1.4 ลิฟต์เฟืองหนอน (Worm Gear Machine) คือเครื่องลิฟต์ที่ใช้กำลังจากมอเตอร์-ไฟฟ้าโดยผ่านเฟืองหนอนไปหมุนรอกขับเคลื่อนล้อแรงหรือเพลา

2. เครื่องลิฟต์ขับเคลื่อนทางอ้อม (Indirect Drive Machine) หมายถึง เครื่องขับลิฟต์ด้วยมอเตอร์ไฟฟ้าผ่านสายพานหรือโซ่ไปยังเฟืองหมุนรอกขับเคลื่อนล้อแรงหรือเพลา

แบ่งออกได้ดังนี้

2.1 ขับเคลื่อนด้วยสายพาน

2.2 ขับเคลื่อนด้วยโซ่

2.3 เครื่องลิฟต์ขับเคลื่อนด้วยเกลียว (Screw Machine) คือเครื่องลิฟต์ที่ขับเคลื่อนด้วยมอเตอร์ไฟฟ้าไปหมุนแกนเดี่ยวที่เป็นเกลียวหรือลดแป้นเกลียวที่ยึดอยู่กับตัวลิฟต์

2.4 เครื่องลิฟต์ขับเคลื่อนด้วยไฮดรอลิก (Hydrolic Machine) คือ เครื่องลิฟต์ที่ถูกขับเคลื่อนด้วยไฟฟ้าโดยมอเตอร์จะไปควบคุมการเคลื่อนที่ของไฮดรอลิกให้ลิฟต์ขึ้น-ลง

การทำงานของเครื่องตรวจจับความเร็วในการเคลื่อนที่นั้นจะทำงาน 2 ขั้นตอน คือ

ขั้นตอนที่ 1 เมื่อความเร็วของมอเตอร์เพิ่มขึ้นถึง 125% ของความเร็วสูงสุดของลิฟต์นั้นๆอุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่จะไปชนสวิตซ์ไฟฟ้า ทำให้วงจาของสวิตซ์ไฟฟ้าเปิดออก ทำให้ระบบควบคุมตัดกระแสไฟฟ้าที่จะป้อนเข้าที่มอเตอร์และจ่ายกระแสไฟฟ้าที่เข้าที่ Magnetic Break ทำให้ลิฟต์หยุดการเคลื่อนที่

ขั้นตอนที่ 2 เมื่อความเร็วของลิฟต์เพิ่มขึ้นถึง 135% ของความเร็วสูงสุดของลิฟต์นั้นๆ อุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่ จะบังคับให้น้ำหนักกดสลิงตกจากที่ยึดด้ามจับสลิงแล้วสลิงก็จะไปกระชากเซฟตี้แคชให้จับยึดรางทำให้ลิฟต์หยุดการเคลื่อนที่ทันที

ดังนั้นจะเห็นได้ว่าอุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่จะมีหน้าที่ บังคับให้ลิฟต์หยุดโดยทางไฟฟ้าก่อน แต่ถ้าไม่สามารถบังคับให้ลิฟต์หยุดโดยทางไฟฟ้าได้แล้ว จึงจะบังคับให้ลิฟต์หยุดการเคลื่อนที่โดยทางกล

2.1.4 อุปกรณ์เพื่อความปลอดภัย (Safety Device)

อุปกรณ์เพื่อความปลอดภัยเป็นอุปกรณ์ที่ติดตั้งอยู่ทางด้านล่างของตัวลิฟต์ ทำงานด้วยระบบทางกลเพื่อให้ลิฟต์หยุดการเคลื่อนที่ ในกรณีที่ลิฟต์เคลื่อนที่ด้วยความเร็วที่เกินกว่า 135% ของความเร็วสูงสุด

การทำงานของเซฟตี้แคช จะทำงานเมื่อสลิงของอุปกรณ์ตรวจจับความเร็วในการเคลื่อนที่ถูกจับยึดโดยน้ำหนักกดสลิงทำให้สลิงหยุดการเคลื่อนที่ทันทีทันใด ขณะที่ลิฟต์กำลังเคลื่นที่อยู่ซึ่งทำให้เกิดการกระชากขึ้น และจากการกระชากนี้ส่งผลให้มีการดึงก้านเหล็ก (Push Rod) ทำให้แท่งลิ่มเคลื่อนที่บีบกดรางเข้ากับรางลิฟต์ทำให้ลิฟต์หยุดได้

อุปกรณ์เพื่อความปลอดภัยแบ่งออกได้เป็น 2 ชนิดคือ

1. อุปกรณ์เพื่อความปลอดภัยแบบอาร์ (R-Type Safety Device) ใช้สำหรับลิฟต์ที่มีความเร็วไม่เกิน 45 เมตร/นาที ใช้ลูกล้อ (Catch Roller) ในการกดราง

2. อุปกรณ์เพื่อความปลอดภัยแบบดับเบิล (W-Type Safety Device) ใช้สำหรับลิฟต์ที่มีความเร็วเกิน 45 เมตร/นาที ใช้ลักษณะในการกดทับราง

2.1.5 ประเภทของกลไกประตู (Door Operation Machine)

แบ่งออกเป็น 2 ประเภทคือ

1. แบบเปิดออกทางด้านแนวนอน (Side Sliding)

2. แบบเปิดออกทางด้านแนวดิ่ง (Up-Sliding)

การทำงานของระบบเปิดปิดประตูนั้น จะทำงานโดยอาศัยบานประตูของตัวลิฟต์เป็นตัวพาให้ประตูนอก (Hatch Door) เปิดออก ดังนั้นการที่ประตูนอกจะเปิดออกนั้นจะอยู่ในช่วงระยะเวลาหนึ่งเท่านั้น ซึ่งเรียกว่า (Door Zone) ต้นกำลังของการเปิด-ปิดประตูนั้นจะมาจากมอเตอร์ที่ ติดอยู่บนหลังคาลิฟต์นั่นเอง

สำหรับประตูนอกนั่นไม่สามารถเปิดออกได้โดยคนที่อยู่ภายนอก โดยจะมีกลไกทางกลเป็นตัวเกี่ยวติดตั้งไว้ ซึ่งอุปกรณ์นี้เรียกว่า Door Lock แต่ยังสามารถเปิดประตูนี้ได้โดยการใช้กุญแจสำหรับเปิดประตูลิฟต์ ซึ่งออกแบบโดยเฉพาะแล้วแต่ตัวลิฟต์

นอกจากนี้ประตูนอกยังมีอุปกรณ์ที่สามารถปิดตัวเองได้ด้วย ซึ่งอุปกรณ์ที่บังคับให้ประตูปิดตัวเองนี้เรียกว่า Door Close ปัจจุบันประตูลิฟต์จะใช้ติดตั้งอยู่ที่รางแขวนประตู Header เป็นตัวปิดประตู

2.1.6 อุปกรณ์ตรวจสอบชั้นของลิฟต์ (Landing Field)

ใช้ในการตรวจสอบระดับชั้นของลิฟต์ เพื่อให้ลิฟต์หยุดตรงระดับชั้นจริงของอาคาร ภายหลังจากที่ลิฟต์ได้รับคำสั่งให้ลดความเร็วลง เพื่อให้เข้าจอดในชั้นที่ได้รับคำสั่งแล้ว อุปกรณ์ที่ใช้ตรวจสอบจะใช้หลักการสนามแม่เหล็ก (Magnetic Field)

1. Position Detector (Positector) ตั้งอยู่บนหลังคาลิฟต์

2. Shielding Plant เป็นแผ่นเหล็กที่ติดตั้งอยู่ภายในช่องลิฟต์

2.1.7 หลักการทำงานของระบบการเคลื่อนที่ของลิฟต์

คือ ลิฟต์สามารถเคลื่อนที่ได้โดยอาศัยการควบคุมจากหน่วยประมวลผลกลาง โดยหน่วยประมวลผลกลางจะทำหน้าที่รับอินพุทจากภายนอกซึ่งก็คือ การระบุการเคลื่อนที่จองผู้ใช้ในแต่ละชั้นเข้ามาเพื่อใช้ในการประมวลผล สั่งการให้ส่วนของวงจรขับเคลื่อนของมอเตอร์ทำงานเพื่อให้ลิฟต์เคลื่อนไปยังตำแหน่งที่ต้องการ

หลักการทำงานของภาคตรวจจับวิธีการตรวจจับที่เลือกใช้ คือ วิธีการตรวจวัดระยะทางโดยรับสัญญาณจากเอนโค้ดเดอร์หรืออุปกรณ์เข้ารหัส ซึ่งเป็นอุปกรณ์ซึ่งเปลี่ยนลักษณะทางเชิงกล (ในที่นี้คือ การหมุนของแกนมอเตอร์) เข้ามาเป็นสัญญาณดิจิตอลนั่นคือ เมื่อเกิดการหมุนของแกนมอเตอร์ โดยเอนโค้ดเดอร์จะสร้างจำนวนพัลส์ออกมา 500 พัลส์ต่อการหมุน 1 รอบหรือมีความ

ละเอียด 1.39 องศาต่อสเต็ปในที่นี้จะใช้สัญญาณเอาท์พุท 2 เส้นที่มีความถี่เท่ากันแต่มีเฟสต่างกัน 90 องศา ถ้าเอนโค้ดเดอร์หมุนไปในทิศทางตามเข็มนาฬิกา A นำ B แต่ถ้าหมุนตามทิศทางทวนเข็มนาฬิกา B นำ A

2.1.8 หลักการทำงานของภาคขับเคลื่อน

ภาคขับเคลื่อนของระบบจำลองถูกแบ่งออกเป็น 2 ส่วน คือ

1. มอเตอร์ทำหน้าที่ขับเคลื่อนตัวลิฟต์
2. มอเตอร์ทำหน้าที่เปิด-ปิดประตูลิฟต์

แรงดันไฟฟ้าที่จ่ายให้กับมอเตอร์จะถูกควบคุมด้วยวิธีการ Pulse Width Modulation การบังคับทิศทางทำได้โดยส่งสัญญาณบอกทิศทางให้วงจรขับมอเตอร์ซึ่ง

1. ในกรณีของมอเตอร์ขับเคลื่อนลิฟต์ สัญญาณบอกทิศทางจะประมวลมาจากอินพุทเลือกชั้นจากภายนอก โดยไมโครคอนโทรลเลอร์จะตัดสินใจจากอินพุทที่ได้รับว่าลิฟต์จะเคลื่อนที่ขึ้นหรือลง

2. ส่วนมอเตอร์ที่ทำหน้าที่ เปิด – ปิด ประตูลิฟต์นั้น ไมโครคอนโทรลเลอร์จะสั่งให้มอเตอร์ทำการเปิดประตูเมื่อลิฟต์เคลื่อนที่ถึงชั้นที่ต้องการ หลังจากนั้นทำการหน่วง

2.2 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association: EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3 ถึง –12 V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12 V แสดงว่าเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุดถึง 20 เมตร

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้น เช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอก ประกอบด้วยการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ และ การเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละ ขาของพอร์ตอนุกรม RS-232 มีดังนี้

- Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูลเช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก
- Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์ บัฟเฟอร์
- Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป
- Data Terminal Ready : DTR เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทางและขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ Null Modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์
- Signal Ground : GND ขากราวด์ของระบบ
- Data Set Ready : DSR ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

- Request To Send : RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null Modem 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- Clear To Send : CTS ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่
- Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

2.2.1 UART

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์ทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (บอดเรต) , รูปแบบการส่งข้อมูล , ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้,เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

ภายใน UART จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (programmabie buadrate generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิต ดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1-0.5,535 UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (half duplex) และฟูลดูเพล็กซ์ (full duplex) โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน

2.2.2 ชนิดของ UART

ในเครื่องคอมพิวเตอร์ทั่วไปมี UART ที่ใช้งานกันอยู่ 2 เบอร์คือ 8250 ซึ่งเป็น UART มาตรฐานที่มีใช้กันมายาวนาน UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ก็ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุกๆรุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้

UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 115,200 บิตต่อวินาทีและเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UARTนอกจากนั้นยังเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO (First In First Out) ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบัตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5V และ +3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาที เมื่อใช้สัญญาณนาฬิกา 16 MHz

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ UART เบอร์ใหม่ๆก็ไม่ได้ช่วยให้การรับส่งข้อมูลของคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่ของสัญญาณนาฬิกาในการแปลงข้อมูลเพียง 1.8432 MHz เท่านั้น

2.2.3 วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะมีชื่อเรียกเป็น COM1,COM2,COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน

การทำงานภายในของพอร์ตอนุกรม ซึ่งจะประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิต 8 ตัวที่ใช้งานร่วมกับ UART แอดเดรสของรีจิสเตอร์ภายในพอร์ตอนุกรมสามารถคำนวณได้จากค่ารีจิสเตอร์พื้นฐานของพอร์ตอนุกรม ยกตัวอย่าง พอร์ตอนุกรม COM1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจิสเตอร์ต่างๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า3F8Hโดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

00H เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูล ก่อนที่จะส่งออกไป

01H รีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์ ใช้ในการเซตโหมดการอินเตอร์รัปต์ของ พอร์ตอนุกรม

02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ใช้เพื่อตรวจสอบโหมดของการอินเตอร์รัปต์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

03H รีจิสเตอร์กำหนดรูปแบบของข้อมูล

04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR

05H รีจิสเตอร์แสดงสถานะการรับและการส่งข้อมูลแบบอนุกรม

06H รีจิสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขา DCD,RI,DSR และ CTS

07H รีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราว

2.2.4 ลักษณะสัญญาณอินพุตและเอาท์พุตของพอร์ต RS-232

สัญญาณเอาท์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTS, DSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART จะให้ระดับสัญญาณเอาท์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จึงต้องเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์ สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้ คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000: 0400H และ 0000: 0401H ส่วนตำแหน่งอื่นๆมีรายละเอียดดังนี้

ตารางที่ 2.1 แสดงข้อมูลในแอดเดรส 0000: 0411H ที่ใช้แสดงจำนวนพอร์ต

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

COM2 = 0000 : 0402H – 0000 : 0403H

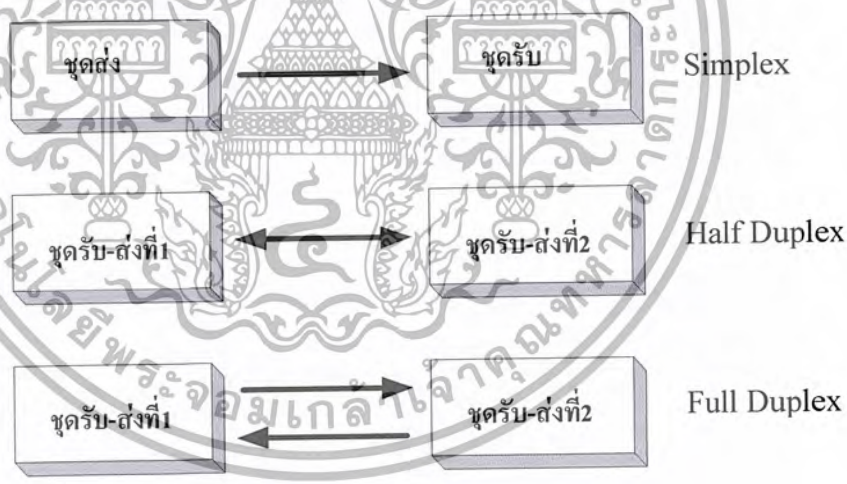
COM3 = 0000 : 0404H – 0000 : 0405H

COM4 = 0000 : 0406H – 0000 : 0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส 0000 : 0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วย โดยมีรายละเอียดดังแสดงในตารางที่ 2.1

2.2.5 การส่งข้อมูลแบบซิมเพ็ลกซ์(Simplex)และดูเพ็ลกซ์(Duplex)

ในการสื่อสารไม่ว่าจะเป็นการสื่อสารข้อมูลหรือการสื่อสารทั่วไปนั้นย่อมจะต้องประกอบด้วยผู้รับและผู้ส่ง ผู้รับในขณะนี้สามารถเป็นผู้ส่งในอนาคตได้ แต่มีบางกรณีที่เป็นผู้รับและผู้ส่งแน่นอนตายตัวอยู่ตลอดเวลา เช่น การสื่อสารข้อมูลระหว่างเครื่องคอมพิวเตอร์กับเครื่องพิมพ์ เป็นต้น การสื่อสารของอุปกรณ์ที่มีผู้รับและผู้ส่งตายตัวนั้น เราเรียกว่าการสื่อสารแบบซิมเพ็ลกซ์ กล่าวคือ การสื่อสารเป็นไปในลักษณะทิศทางเดียวตลอดเวลา ซึ่งจะมีที่ใช้ไม่มากนัก การสื่อสารทั่วไปนั้นจะเป็นแบบ ดูเพ็ลกซ์ คือมีทิศทางการสื่อสาร 2 ทิศทางทั้งไปและกลับ การสื่อสารในลักษณะ ดูเพ็ลกซ์ นี้ยังแบ่งออกได้เป็น 2 ชนิด คือ แบบ ฮาร์ฟดูเพ็ลกซ์ นิยมเขียนย่อว่า Haft Duplex ซึ่งจะมีทิศทางการสื่อสารในลักษณะกันที่ผลัดกันเป็นผู้ส่ง และผู้รับพร้อมกันไป แบบฟูลดูเพ็ลกซ์ (Full Duplex)นิยมเขียนย่อ HDX จะมีทิศทางการสื่อสารในลักษณะสัญญาณรับทิศทางหนึ่ง สัญญาณส่งอีกทิศทางหนึ่งหรือกล่าวได้อีกนัยหนึ่งว่าสัญญาณรับและส่งจะมีสายตัวนำสัญญาณแยกกันจากกัน โดยเด็ดขาดดังแสดงในรูปที่ 2.1

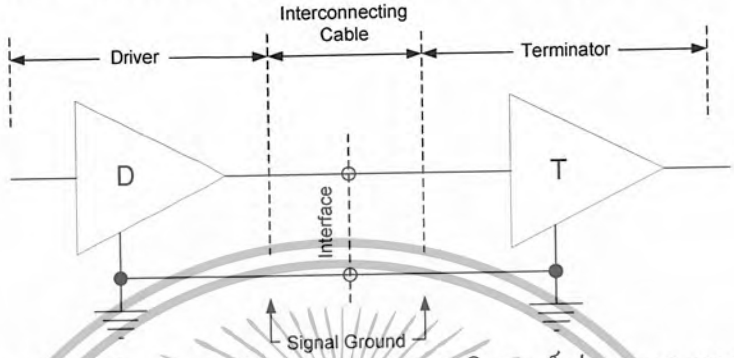


รูปที่ 2.1 แสดงการสื่อสารข้อมูลอนุกรมแบบต่างๆ

2.2.6 เปรียบเทียบมาตรฐานสัญญาณอนุกรมแบบต่าง ๆ

2.2.6.1 มาตรฐานสัญญาณอนุกรมแบบ RS232C

มาตรฐานการสื่อสารข้อมูลแบบอนุกรมที่กำหนดโดย EIA (Electronics Industries Association) มาตรฐาน RS232C ได้ถูกตีพิมพ์ในปี ค.ศ. 1969 ตัวอักษร RS แทน "Recomm Standard" 232 แทนหมายเลขของมาตรฐาน ส่วนอักษร C แสดงให้รู้ว่ามาตรฐานได้รับการแก้ไขกี่ครั้ง



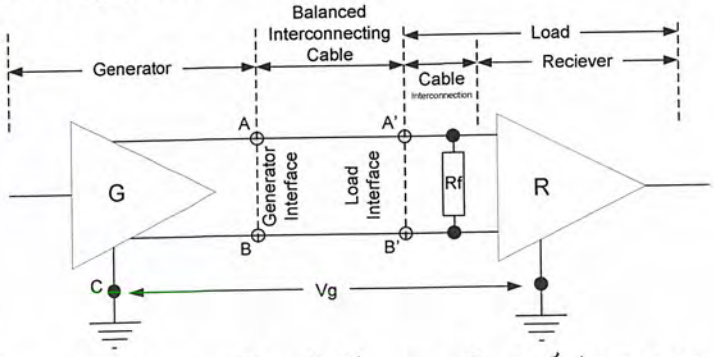
รูปที่ 2.2 แสดงลักษณะสมบัติทางไฟฟ้าของการอินเทอร์เฟสแบบ RS232C

คุณลักษณะสมบัติทางไฟฟ้าของการอินเทอร์เฟสแบบ RS232C

- ถูกออกแบบให้ใช้กับอุปกรณ์พวกสัญญาณ Discrete
- ใช้การอินเทอร์เฟสแบบ Unbalanced
- ในแต่ละวงจรใช้ลวดนำในการนำสัญญาณ 1 เส้น และมีสายกราวน์รวมของทุกวงจรอีกหนึ่งเส้น
- อัตราเร็วในการส่งข้อมูลมีค่า < 20 กิโลบิทต่อวินาที (Kbps)
- ระยะทางสูงสุดที่ใช้ในการส่งข้อมูลมีค่า < 15 เมตร
- ทำให้เกิด Crosstalk ที่มีค่ามาก

2.1.6.2 มาตรฐานสัญญาณอนุกรมแบบ RS422

มาตรฐาน RS422 เป็นชุดขับแบบขยายความแตกต่างในรูปของกระแส ทำให้อัตราเร็วในการส่งข้อมูลมีสูงขึ้น และระยะทางที่ใช้ส่งข้อมูลระหว่างชุดส่งและชุดรับมีระยะไกลขึ้น เพราะเป็นการส่งชนิด ฟูลดูเพล็กซ์



รูปที่2.3 แสดงลักษณะสมบัติทางไฟฟ้าของการอินเทอร์เฟสแบบ RS422

ลักษณะสมบัติทางไฟฟ้าของการอินเทอร์เฟสแบบ RS422

- ชุดส่งสัญญาณแบบ Balanced
- ชุดรับข้อมูลเป็นแบบขยายความแตกต่าง
- ในแต่ละวงจรใช้ลวดตัวนำในการส่งสัญญาณจำนวน 2 เส้น
- อัตราเร็วในการส่งข้อมูลสูงถึง 10 เมกกะบิทต่อวินาที(Mbps)
- ระยะทางที่ใช้ในการส่งข้อมูลได้ไกลถึง 4000 ฟุต

มาตรฐาน RS422 ใช้การส่งข้อมูลในลักษณะของ One-Way Balanced-Line ซึ่งมีชุดส่งข้อมูลบนระบบอยู่ 1 ชุดและชุดรับข้อมูล 32 ชุด โดยอัตราเร็วในการส่งข้อมูลมีค่าสูงถึง 10 mbps ที่ระยะทางเท่ากับ 40 ฟุต ในกรณีที่ข้อในอัตราเร็วที่ต่ำกว่า 10 mbps ระยะทางที่ใช้ในการส่งข้อมูลสามารถขยายได้ถึง 4,000 ฟุต

2.1.6.3 มาตรฐานสัญญาณอนุกรมแบบ RS485

มาตรฐาน RS485 นี้ก็มีพัฒนามาจาก RS422 คือผู้ผลิตบางบริษัทได้ทำวงจรขับสัญญาณ(ชุดส่ง)เป็นแบบ Tri-State ทำให้เราสามารถส่งข้อมูล ได้สองทิศทางบนสายคู่เดียว (Single Pair) คุณสมบัติข้อนี้จึงทำให้ระบบส่งข้อมูลมีโครงสร้างเป็นแบบ Multidrop ซึ่งอุปกรณ์หลาย ๆ ประเภทสามารถรับและส่งข้อมูลแบบ ฮาร์ฟดูเพล็กซ์ บนสายเดียวได้

รูปที่ 2.4 แสดงลักษณะสมบัติทางไฟฟ้าของการอินเทอร์เฟสแบบ RS485

มาตรฐาน RS485 นี้ ทางบริษัท ผู้ผลิตได้ออกแบบชุดส่งและชุดรับให้สามารถต่อร่วมอยู่บนบัสได้ถึงอย่างละ 32 ชุด แต่การส่งข้อมูลในขณะเวลาหนึ่ง ๆ นั้นจะส่งได้ทีละชุด ฉนั้นจะต้องมีกรรมวิธีตรวจสอบบัสก่อนว่าว่างและพร้อมที่จะให้ข้อมูลผ่านไปได้ ทั้งนี้เพื่อป้องการสับสนของข้อมูลนั้นเองการรับส่งข้อมูลสามารถที่จะรับส่งทั้งหมดได้ถึง 32 ชุด แต่ละชุดต้องมีการตรวจหัวข้อมูล(Header)หรือ Identify Code ก่อนเพื่อเป็นรหัสให้รับทราบว่าต้องการสื่อสารกับชุดใด

ส่วนประกอบของมาตรฐาน RS485 ประกอบด้วย ชุดรับส่ง สายเคเบิล และ Resistor Terminating ซึ่งค่าความต้านทาน RT มีค่าประมาณ 220 โอห์ม โดยทั่วไปจะต้องศึกษาข้อมูลจาก Data Sheet ของ IC เบอร์นั้น ๆ ประกอบควบคู่ไปด้วย

ตารางที่ 2.2 แสดงการเปรียบเทียบการสื่อสารแบบ

Specification	**RS232C**	**RS422**	**RS485**
Mode Of Operated	Single Ended	Difference	Difference
No of Driver & Receivers	1 Driver	1 Driver	32 Drivers
Allowed On One Line	1 Receiver	32 Receivers	32 Receivers
Max Cable Length	50 feet	4000 feet	4000 feet
Max Data Length	20 Kb/s	100Kb/s	10 Mb/s
Driver Output Max Voltage	+/- 5Vdc.	-0.25 to 6Vdc.	-7 to 12Vdc.
Driver Out (Load)	+/- 5Vdc.	+/- 2Vdc.	+/- 1.5Vdc.
Signal Level (Unload)	+/- 15Vdc.	+/- 5Vdc.	+/- 5Vdc.
Driver Load Impedance	3 to 7 KΩ	100Ω	54Ω
Driver Output Current (ON)	-	-	+/- 100μA
High Impedance State (OFF)	Vmax/300 Ω	+/- 100 μA	+/- 100 μA
Receiver Input Volt Rang	+/- 15Vdc.	+/- 7Vdc	-7 to 12Vdc
Receiver Input Sensitivity	+/- 3Vdc.	+/- 200mVdc.	+/- 200mVdc.
Receiver Input Resistance	3 to 7 Ω	4 KΩ(Min)	12 KΩ(Min)

2.3 รูปแบบของโปรโตคอล (Protocol Type)

ในการสื่อสารข้อมูลจะต้องมีกฎหรือข้อกำหนดในการสื่อสารข้อมูล หรือที่นิยมเรียกว่า โปรโตคอล (Protocol) ซึ่งเป็นส่วนที่จะกำหนดมาตรฐานในการควบคุม และจะจัดระบบในการสื่อสารข้อมูลสำหรับรายละเอียดที่จะกล่าวในที่นี้จะเกี่ยวข้องเฉพาะในส่วนของโปรโตคอลการควบคุมการเชื่อมโยงข้อมูล (Data Link Control Protocol หรือ DLCP) ซึ่งจัดการในส่วนของขั้นตอนและหลักการต่าง ๆ คือโครงสร้างและรายละเอียดของข้อมูล วิธีในการสื่อสารข้อมูล การตรวจสอบแก้ไขความผิดพลาดของข้อมูล และขบวนการในการควบคุมการสื่อสาร โดย DLCP แบ่งออกได้ตามโครงสร้างของข้อมูล 2 แบบ คือ Byte – Oriented Protocol และ Bit Oriented Protocol

2.3.1 ไบท์โอเรียนโปรโตคอล (Byte-Oriented Protocol)

โปรโตคอลแบบนี้เป็นโปรโตคอลที่การสื่อสารข้อมูล และควบคุมการทำงานจะทำโดยใช้ลักษณะข้อมูลเป็นตัว (Character) หรือ Byte หรืออาจเรียก Character Oriented Protocol ซึ่งแบ่งออกเป็น

1. อะซิงโครนัสโปรโตคอล (Asynchronous Protocol)

โปรโตคอลในการสื่อสารข้อมูลนี้จะใช้สื่อสารข้อมูลแบบ Half-Duplex ที่มีลักษณะการสื่อสารข้อมูลแบบอะซิงโครนัส ซึ่งเป็นการสื่อสารข้อมูลแบบพื้นฐานที่ใช้งานมาเป็นเวลานานแล้ว จึงมีรายละเอียดและขั้นตอนในการสื่อสารข้อมูลที่ทำให้โอกาสเกิดความผิดพลาดได้น้อยยังมีข้อดีที่การสื่อสารข้อมูลแบบนี้มีในโครงสร้างการทำงานที่ง่าย อุปกรณ์ที่ใช้ในการสื่อสารข้อมูลก็ไม่สลับซับซ้อนและมีราคาถูก โปรโตคอลแบบนี้จึงเหมาะสำหรับใช้ในระบบขนาดเล็ก

2. ไบนารี่ซิงโครนัสโปรโตคอล (Binary Synchronous Protocol)

โปรโตคอลแบบนี้จะมีลักษณะการทำงานที่ใช้งานข้อมูลเป็นลักษณะไบท์ และยังคงใช้การสื่อสารข้อมูลแบบซิงโครนัส มีรายละเอียดและขั้นตอนในการสื่อสารข้อมูลที่ให้ความน่าเชื่อถือมากกว่า อีกทั้งยังสามารถใช้อัตราเร็วในการสื่อสารข้อมูลที่สูงกว่า โดยตัวอย่างการสื่อสารข้อมูลแบบนี้ที่ได้กำหนดเป็นมาตรฐานแล้ว คือการสื่อสารข้อมูลตามมาตรฐาน BSC(Binary Synchronous Communications) ซึ่งเป็นโปรโตคอลที่ลักษณะของข้อมูลแบบไบท์ที่นิยมนำไปใช้งาน

2.3.2 บิทโอเรียนท์โปรโตคอล (Bit Oriented Protocol)

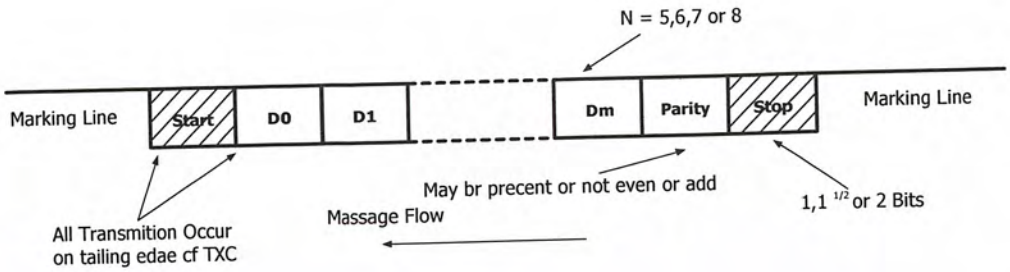
โปรโตคอลแบบนี้เป็นโปรโตคอลที่การสื่อสารข้อมูลและการควบคุมการทำงานจะทำ โดยใช้ลักษณะข้อมูลที่เป็นบิทโดยมีตัวอย่างของการสื่อสารข้อมูลในลักษณะนี้ที่มีการกำหนดขึ้น เป็นมาตรฐานแล้ว คือ HDLC (High – Level Data Link Control) โดยมีโครงสร้างของข้อมูลแบบซิงโครนัสเช่นเดียวกันกับ BSC แต่ต่างกันที่มีลักษณธของข้อมูลเป็นบิท ซึ่งโปรโตคอลแบบนี้ ข้อดี

ที่สามารถสื่อสารข้อมูลแบบ Full Duplex ได้ทำให้การสื่อสารข้อมูลได้รวดเร็วกว่าแต่โปรโตคอลแบบนี้ก็จะมีรายละเอียดและโครงสร้างในการสื่อสารข้อมูลที่สลับซับซ้อนมากทำให้การควบคุมการทำงานทำได้ยากและต้องใช้อุปกรณ์ที่มีราคาสูง จึงไม่เหมาะที่จะนำมาใช้กับงานที่มีขนาดเล็ก ๆ

2.3.3 โปรโตคอลของการสื่อสารแบบอนุกรม

เมื่อพิจารณาการส่งข้อมูลในแบบอนุกรมให้ดีจะพบว่า ปัญหาหนึ่งที่ควรจะเกิดขึ้นอยู่เสมอก็คือการตัดสินใจว่าข้อมูลที่ได้รับนั้นมีจุดเริ่มต้นที่ใด ดังนั้นจึงมีการกำหนดข้อตกลงในการสื่อสารขึ้นเพื่อแก้ปัญหานี้ข้อตกลงดังกล่าวเราเรียกว่า โปรโตคอล (Protocol) ของการสื่อสารข้อมูลแบบอนุกรมสามารถแบ่งออกเป็น2ประเทใหญ่ๆคือ โปรโตคอลสำหรับการสื่อสารข้อูลแบบซิงโครนัส (Symchronous) และโปรโตคอลสำหรับการสื่อสารข้อมูลแบบอะซิงโครนัส (Asymchronous) การสื่อสารข้อมูลแบบซิงโครนัสนั้น ข้อมูลจะถูกส่งออกไปอย่างสม่ำเสมอ ช่วงเวลาระหว่างบิตและระหว่างเวิร์ดจะเท่ากันเสมอ ดังนั้นในการสื่อสารข้อมูลอนุกรมในแบบซิงโครนัสจึงต้องมีสายสัญญาณเพิ่มเติมคำกับการส่งว่าควรจะส่งเมื่อใด และควรจะหยุดเมื่อใด ระบบที่เป็นซิงโครนัสจะเป็นระบบที่มีความเร็วสูง แต่ก็ยังต่ำกว่าการสื่อสารแบบขนาน

การสื่อสารแบบอะซิงโครนัสนี้ เป็นหัวใจของการสื่อสารข้อมูลผ่านทางสายโทรศัพท์ ในปัจจุบัน การสื่อสารแบบนี้ ช่วงระยะเวลาระหว่างบิทจะมีค่าเท่ากันเช่นเดียวกับซิงโครนัส แต่จะมีระยะห่างระหว่างเวิร์ดนั้นแตกต่างกันออกไปเป็นกี่วินาที นาที ชั่วโมง หรือวันเป็นต้น ได้ทั้งสิ้นขึ้นอยู่กับทางฝ่ายรับสามารถรอได้หรือไม่เท่านั้น เมื่อไม่มีข้อกำหนดทางด้านระยะเวลาระหว่างเวิร์ดแล้ว ทางผู้ส่งและผู้รับจะเข้าใจตรงกันได้อย่างไรว่าที่ใดคือจุดเริ่มต้นและจุดสิ้นสุดของแต่ละเวิร์ด เพื่อแก้ปัญหานี้ จึงมีการกำหนดข้อตกลงเกี่ยวกับรูปแบบของข้อมูลที่จะส่งให้ทางผู้รับสามารถเข้าใจว่าจุดใดเป็น จุดเริ่มต้นของเวิร์ด ข้อกำหนดดังกล่าวกำหนดให้แต่ละเวิร์ดจะต้องขึ้นต้นด้วยบิทที่เรียกว่า บิทเริ่มต้น(Start Bit) ซึ่งจะต้องมีข้อมูลเป็นลอจิก 0 เสมอ จากนั้นตามด้วยบิทข้อมูลที่ต้องการส่ง ซึ่งมีความยาว 5-8 บิท ถัดจากบิทข้อมูลก็จะเป็นพาริตีบิท ซึ่งทำหน้าที่เป็นบิทสำหรับตรวจสอบความถูกต้องของข้อมูลที่ได้รับว่ามีความถูกต้องหรือไม่ บิทพาริตี บิทนี้มีสองประเภทคืออีเวนพาริตี (Even Parity) ซึ่งจะกำหนดจำนวนบิทที่เป็นลอกจิก 1 ในบิททที่เป็นข้อมูลมีจำนวนเป็นจำนวนคู่ และออดพาริตี (Add Parity)ซึ่งจะกำหนดจำนวนบิทที่เป็นลอกจิก 1 ในบิททที่เป็นข้อมูลมีจำนวนเป็นจำนวนคี่ ในการส่งข้อมูลบางครั้งอาจจะไม่มีการใช้บิทพาริตีก็ได้ ถ้าหากการสื่อสารในครั้งนั้นมีความน่าเชื่อถือสูง คือ มีสัญญาณรบกวนต่ำ เป็นการเพิ่มความเร็วในการสื่อสารได้ด้วย บิทสุดท้ายในรูปแบบก็คือ บิทสุดท้าย (Stop Bit)ทำหน้าที่บอกทางผู้รับว่า ขณะนี้ข้อมูลที่ทางผู้รับได้รับนั้น ครบเวิร์ดแล้ว ขอให้เตรียมชุดรับเวิร์ดต่อไปได้ บิทสุดท้าย นี้อาจมีความยาวเป็น 1 บิทหรือสองบิทก็ได้



รูปที่ 2.5 แสดงแบบมาตราฐานการสื่อสารข้อมูลแบบอะซิงโครนัส

จากรูปแบบดังกล่าว จะเห็นว่า เรามีรูปแบบสำหรับการสื่อสารมากมายไปหมด เช่น 5E1 (5Data bit,Even Parity, 1 Stop bit), 7E1 (7Data bit,Even Parity, 1 Stop bit), 8N1 (8Data bit,No Parity, 1 Stop bit) เป็นต้น ในการใช้งานทั่วไป เรานิยมใช้กันอยู่เพียงสองรูปแบบคือ 7E1และ 8N1 จะเลือกใช้รูปแบบใดขึ้นอยู่กับสภาพของสายส่งสัญญาณ ว่ามีสัญญาณรบกวนมากเพียงใด ถ้าหากสายส่งมีสัญญาณรบกวนมาก ก็ควรจะใช้ 7E1 แต่ถ้าสายส่งสัญญาณมีสภาพดี สัญญาณรบกวนต่ำ การใช้ 8N1 จะเร็วกว่า เป็นต้น ทั้งนี้จะต้องมีการตกลงกันล่วงหน้า ระหว่างผู้รับและผู้ส่ง ว่าจะใช้รูปแบบใดในการสื่อสาร ลักษณะของข้อมูลที่ถูกส่งออกไปจะมีลักษณะดังรูป

บทที่ 3
หลักการและทฤษฎีที่ใช้ในการออกแบบ

3.1 รีโมทเทอร์มินอล อินพุท และ เอาท์พุท (RTI16, RTO16)

รีโมทเทอร์มินัล (RTI16) รีโมทเทอร์มินัลเอาท์พุท (RTO16) เป็นโมดูลอินพุท-เอาท์พุทเพื่อการควบคุมระยะไกล

รูปที่ 3.1 รีโมทเทอร์มินัล (RTI16)

รูปที่ 3.2 รีโมทเทอร์มินัลเอาท์พุท (RTO16)

ตารางที่ 3.1 แสดงคุณสมบัติ

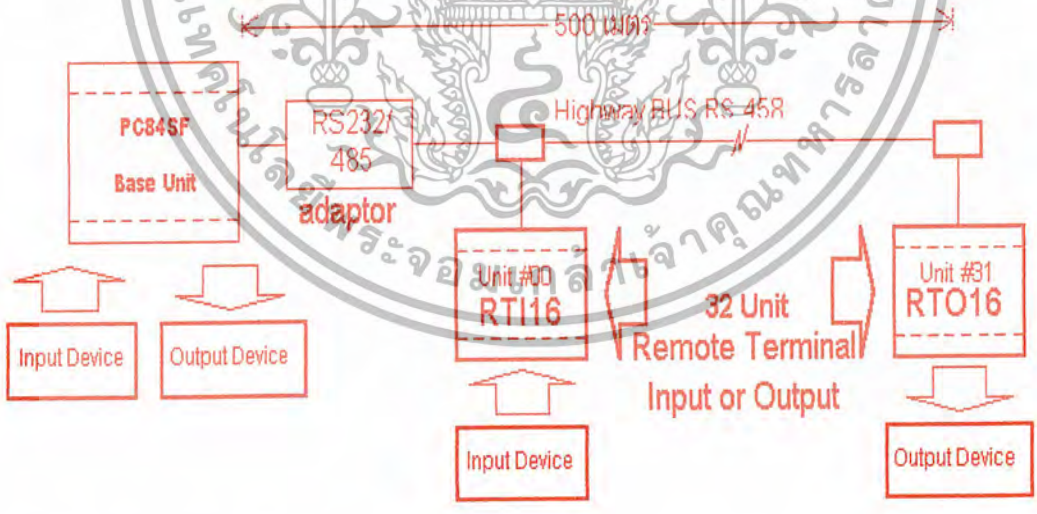
Model	RTI16	RTO16	RTAD
Points / Module	16 Points Digital Input	16 Points Digital Output	4 Point Analog Input, 2 Point Analog Output
Digital Input Signal	Dc. Voltage Source ,active Low	–	–
Digital Output Signal	–	Transistor, Open Collector NPN Type	–
Analog Input & Output Signal	–	–	0 to 10 Vdc. OR 4 to 20 mA. 8 Bit resolution
Isolated Circuit	Opto isolators	Opto isolators	Opto isolators
Relay Output	–	Option (RLY16)	–
Communication	RS232c/RS485	RS232c/RS485	RS232c/RS485
Maximum Module in the system	16 Module	16 Module	16 Module
Fac-talk Protocol	Yes	Yes	Yes
Installation	Rail rack	Rail rack	Rail rack
Dimension (W*L*H) mm.	125*136*29	125*136*29	125*136*29
Power Supply	24 Vcd. , 1A.	24 Vdc. , 1A.	24 Vdc. , 1A.

3.1.1 ผู้ใช้งานสามารถประยุกต์ใช้งาน RTI16 และ RTO16 ตามลักษณะการควบคุมได้เป็น 2 วิธี

1. ใช้เครื่องควบคุม PLC เป็นเครื่องควบคุมหลัก
2. ใช้เครื่องคอมพิวเตอร์เป็นชุดประมวลผล

➢ ใช้เครื่องควบคุม PLC (PC84SF Series)เป็นเครื่องควบคุมหลัก

เพื่อติดต่อสื่อสารกับส่วนอินพุทโมดูล (RTI16) และเอาท์พุทโมดูล (RTO16) ผ่านทางพอร์ตอนุกรม RS485 โปรแกรมบนเครื่องควบคุม PLC (PC84SF Series) จะมีคำสั่งสนับสนุนเฉพาะที่ง่ายต่อการใช้งาน คำสั่งสนับสนุนนี้ได้แก่คำสั่ง RDI(Read Input Module) FUN70 , WRO (Write Output Module) FUN71 เพื่อใช้อ่านข้อมูลจาก RTI16 หรือเขียนข้อมูลให้กับ RTO16 ตามลำดับ การประยุกต์การใช้งานในรูปแบบนี้สามารถใช้งานได้จำนวนสูงสุด 32 โมดูล จะเป็นอินพุทโมดูล เอาท์พุทโมดูล หรือ อินพุท-เอาท์พุทโมดูลร่วมกันก็ได้แต่ต้องไม่เกินจำนวนสูงสุด จะเห็นได้ว่าระบบทางด้านฮาร์ดแวร์นั้นมีความยืดหยุ่นมากพอเพราะไม่ถูกจำกัดด้วยขนาดของอินพุทหรือเอาท์พุทที่แน่นอนตายตัวลงไป จึงทำให้ผู้ใช้งานสามารถทำการออกแบบวงจรควบคุมให้กับเครื่องจักรกลที่หลากหลายโดยไม่ต้องกังวล เหมือนกับการใช้เครื่องควบคุม PLC อื่นๆที่ค่อนข้างจะจำกัดด้วยจำนวนอินพุทเอาท์พุทที่แน่นอนตายตัว



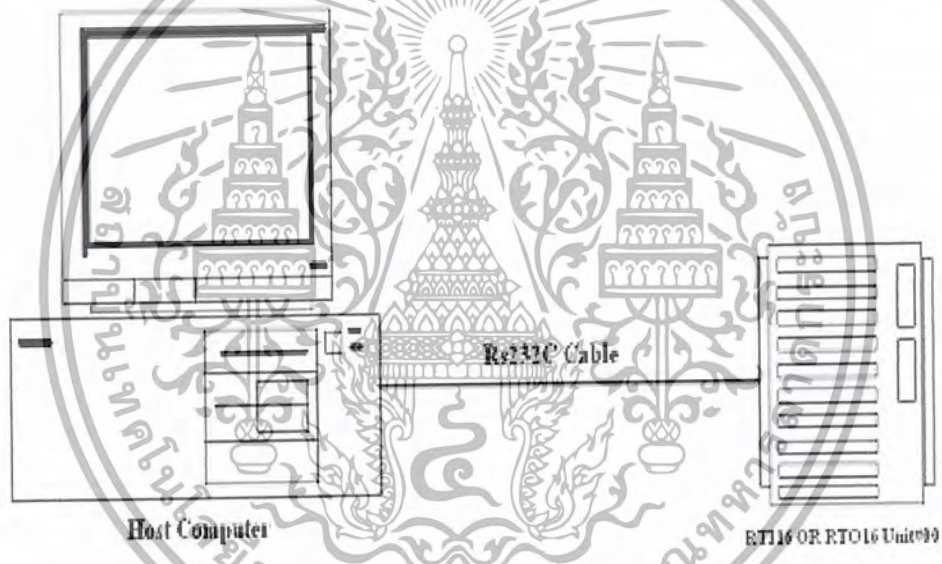
รูปที่ 3.3 แสดงการใช้ PC84SF Series เป็นเครื่องควบคุมหลักต่อร่วมกับ RTI16 และ RTIO16

ข้อดีอีกประการหนึ่งก็คือเป็นการเพิ่มระยะทางของการควบคุมหรือควบคุมระยะไกล ได้ดี เพราะส่วนของการสื่อสารข้อมูลระหว่างเครื่องควบคุม PC84SF กับโมดูลรีโมทเทอร์มินอล RTI16

และ RTIO16 เป็นแบบ RS485 นี่แหละซึ่งถือว่าเป็นการกระจายจุดต่อเทอร์มินอล อินพุท เอาท์พุท บางครั้งก็เรียกว่าอินพุทเอาท์พุทแบบมัลติดร๊อป (Multi drop Input-Output) เพราะผู้ออกแบบไม่ต้องการพะวงเรื่องระยะทางของการควบคุมของอุปกรณ์อินพุทเอาท์พุท การควบคุมวิธีการนี้จึงมีข้อได้เปรียบกว่าเครื่องควบคุม PLC ตระกูลอื่นๆ ที่มีจำหน่ายทั่วไป

➢ ใช้เครื่องคอมพิวเตอร์เป็นชุดประมวลผล

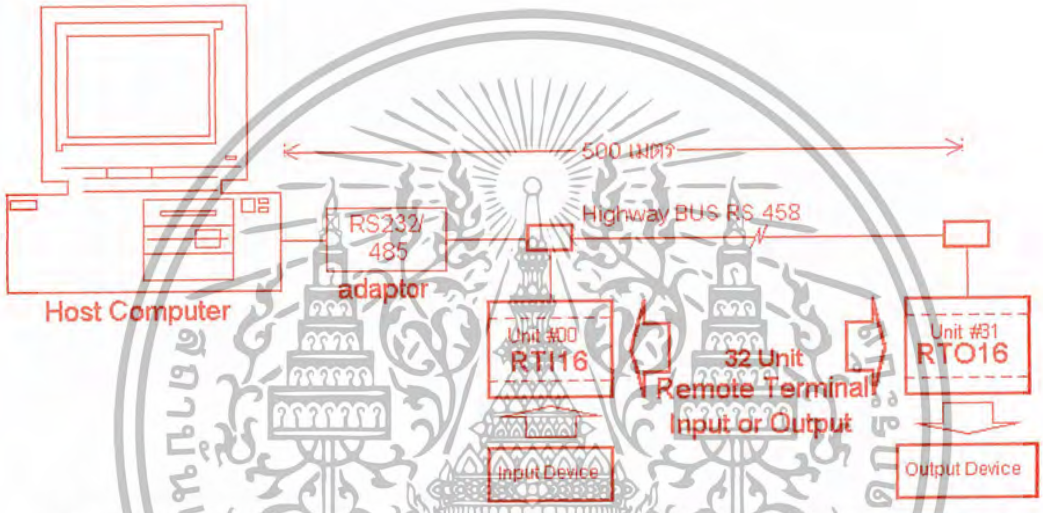
โดยเครื่องคอมพิวเตอร์จะทำหน้าที่เป็นชุดประมวลผล ส่วน RTI16 และ RTO16 ในส่วนของพอร์ตอนุกรมของโมดูลทั้งสองนี้ ตามมาตรฐานสินค้าแล้วจะประกอบด้วย ชุดขับสัญญาณในแบบแรงดัน RS232c และชุดขับสัญญาณแบบกระแส RS485 ผู้ใช้สามารถเลือกใช้งานได้ความต้องการ กรณีที่ในรูปแบบของ RS232c บางครั้งเรียกการต่อใช้งานแบบนี้ว่า การต่อจุดต่อจุด (Point to Point) สามารถเลือกใช้ RTI16 และ RTO16 อย่างใดอย่างหนึ่งเท่านั้น



รูปที่ 3.4 การเชื่อมต่อกันแบบ Point to Point ด้วย RS232c

ในการประยุกต์ด้วยวิธีนี้มีข้อจำกัดอยู่มาก และไม่ค่อยนิยมจะใช้กันเพราะได้ระยะทางการควบคุมที่ไม่ไกลมากนักและได้เพียงจุดต่อเดียว แต่ถึงอย่างไรก็ตามจะช่วยอำนวยการตรวจสอบความสะดวกในการทำงานของอุปกรณ์ อินพุท เอาท์พุท เพราะผู้ใช้งานสามารถใช้โปรแกรมช่วยประเภทเทอร์มินอล เช่น Windows Terminal, ProComm และอื่นๆอ่านข้อมูลจากอินพุทผ่าน RTI16 หรือเขียนข้อมูลโดยการ Force ให้กับอุปกรณ์เอาท์พุทผ่าน RTO16 ในรูปแบบ Factalk Protocol ที่ทางบริษัทได้ออกแบบไว้ ผู้ใช้งานจะทราบได้ทันทีว่าชุดอุปกรณ์ใดบ้างที่ใช้งานได้หรือกล่าวอีกนัยหนึ่งว่าเป็นการทดสอบแบบ Manual ที่ใช้คอมพิวเตอร์เป็นเครื่องมือช่วยทดสอบนั่นเอง

สำหรับการติดตั้งเพื่อกระจายไปในเครือข่ายของ RTI16 และ RTO16 จะสามารถทำได้สูงสุดถึง 32 โมดูลพอร์ทอนุกรม RS485 ดั้งนั้นทางด้านฮาร์ดแวร์ผู้ใช้จำเป็นต้องเพิ่มชุดแปลงสัญญาณจาก RS232C เป็น RS485 ให้กับคอมพิวเตอร์อีกชุดหนึ่งจึงจะใช้งานได้ส่วนการขยายจุดต่อให้กับ RTI16 และ RTO16 สามารถทำได้ไม่ต่างกับวิธีการใช้งาน PC84SF จะแตกต่างก็เพียงโปรแกรมควบคุมการทำงาน วิธีที่กล่าวถึงนี้ใช้คอมพิวเตอร์ควบคุมดังนั้นโปรแกรมที่พัฒนาสะดวกถ้าใช้โปรแกรมภาษาระดับสูง (High Level Langue) เช่น Visual Basic เป็นต้น แต่สำหรับวิธีของ PC84CF โปรแกรมจะเป็นภาษาเฉพาะที่เรียกว่า Ladder Program ผู้ใช้งานจึงสามารถเลือกใช้งานได้ตามแบบที่ตนเองมีความถนัด

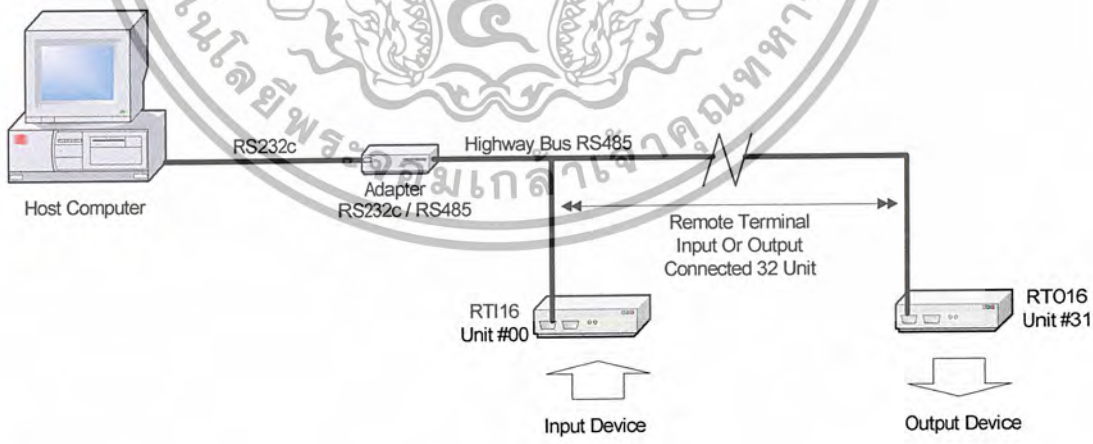


รูปที่ 3.5 เครื่องคอมพิวเตอร์เป็นชุดประมวลผลร่วมกับ RTI16 และ RTO16 ทางพอร์ท RS485

โดยเครื่องคอมพิวเตอร์จะทำหน้าที่เป็นชุดประมวลผล ส่วน RTI16 และ RTO16 ในที่ส่วนของพอร์ตอนุกรมของโมดูลทั้งสองชุดนี้ ตามมาตรฐานสินค้าแล้วจะประกอบด้วย ชุดขับสัญญาณในแบบแรงดัน RS232C และชุดขับสัญญาณแบบกระแส RS485 ผู้ใช้สามารถเลือกใช้ได้ตามต้องการ กรณีเลือกใช้ ในรูปของ RS232C บางครั้งเรียกการต่อใช้งานแบบนี้ว่า การต่อจุดต่อจุด (Point to Point) สามารถใช้เลือก RTI16 หรือ RTO16 อย่างใดอย่างหนึ่งเท่านั้น ในการประยุกต์ด้วยวิธีนี้มีข้อจำกัดอยู่มากและไม่ค่อยจะอำนวยความสะดวกในการตรวจสอบการทำงานของอุปกรณ์อินพุท เอาท์พุท เพราะผู้ใช้งานสามารถใช้โปรแกรมช่วยประเภทเทอร์มินอล เช่น Windows Terminal ,Procomm และอื่น ๆ อ่านข้อมูลจากอุปกรณ์อินพุทผ่าน RTI16 หรือเขียนข้อมูลโดยการ Force ให้กับอุปกรณ์เอาท์พุท RTO16 ในรูปแบบ FAC Talk Protocol ที่ทางบริษัทได้ออกแบบไว้ผู้ใช้งานจะทราบได้เลยทันทีว่าอุปกรณ์ชุดได้บ้างที่ใช้งานได้หรือกล่าวได้อีกนัยหนึ่งว่าเป็นการทดสอบแบบ Manual ที่ใช้คอมพิวเตอร์เป็นเครื่องมือ (Tools) ช่วยทดสอบนั่นเอง

สำหรับการติดตั้งเพื่อกระจายไปในเครือข่าย RTI16 และ RTO16 จะสามารถทำได้สูงสุดถึง 32 โมดูลทางพอร์ตอนุกรม RS485 ดังนั้นทางด้านฮาร์ดแวร์ผู้ใช้จำเป็นต้องเพิ่มชุดแปลงสัญญาณ RS232C เป็น RS485 (Adapter) ให้กับคอมพิวเตอร์อีกชุดหนึ่งจึงจะใช้งานด้านได้ส่วนการขยายจุดต่อให้กับ RTI16 และ RTO16 สามารถทำได้ไม่ต่างจากการต่อให้งานแบบใช้ PLC เป็นตัวประมวลผล จะแตกต่างก็เพียงโปรแกรมควบคุมการทำงาน วิธีที่กล่าวถึงนี้ใช้คอมพิวเตอร์ควบคุมดังนั้นโปรแกรมที่พัฒนาจะสะดวกถ้าใช้โปรแกรมภาษาสูง (High Level Language) เช่น VB6,Turbo C++ หรือ Delphi เป็นต้น แต่ในกรณีที่ใช้ PLC เป็นตัวประมวลผลจะใช้เพียงโปรแกรม Ladder เท่านั้นเอง

รูปที่ 3.6 แสดงการใช้เครื่องคอมพิวเตอร์เป็นชุดประมวลผลร่วมกับ RTI16 หรือ RTO16 ทางพอร์ต RS232C (การต่อแบบ Point To Point)



รูปที่ 3.7 แสดง การใช้เครื่องคอมพิวเตอร์เป็นชุดประมูลผลร่วมกับ RTI16 และ RTO16 ทางพอร์ต RS485

3.1.2 คุณสมบัติทั่วไปของ RTI16

เป็นโมดูลอินพุท ที่ออกแบบมาเพื่อรับค่าสภาวะต่าง ๆ จากอุปกรณ์ตรวจจับสัญญาณในงานอุตสาหกรรม หรือ ในเครื่องจักรกล เช่น Poximity Switch ,Limit Switch ,Photo Switch เป็นต้นที่มีเอาท์พุทของอุปกรณ์ตรวจจับสัญญาณเป็นแบบ Contact หรือ Transistor ชนิด NPN รับค่าสภาวะได้จำนวน 16 จุด วงจรภายในเป็นแบบ Opto-Isorator ส่วนแสดงผลการรับสภาวะด้วย LED ทีละจุด และแสดงผลเป็นเลขฐานสิบหกได้โดยมี 7 Segment LED 4 หลักทำให้สะดวกต่อการสังเกตค่าสภาวะที่มาจาก การสื่อสารข้อมูลกระทำได้โดยตรงทั้งทาง พอร์ตอนุกรม แบบ RS232C (Point To Point) และ RS485(Multi Drop)ใช้กับแหล่งจ่ายไฟกระแสตรง 24 โวลต์ 1 แอมป์

ตารางที่ 3.2 คุณสมบัติทางเทคนิคของ RTI16

Input Opto-Isolator	16 Points
Input Sensor Contact or Transistor(NPN Type)	24 VDC. Active Low 7 mA./Point
Power Supply	24 VDC/ 1 A.
Binary Display	16 Points
Hexadecimal Display 7 Segment LED	4 Digits
Operation Under CPU 89C51 at 11 MHz.	
Serial Port RS232C And RS485	Standard
Baudrate 9600,4800,2400,1200 Bits/S	DIP Switch No. 1,2
Communication	FAC-Talk Protocol
Unit Number 0-31	DIP Switch No 4,5,6,7,8

การกำหนดและเลือก DIP Switch ของ RTI16 ให้เป็น Unit Number ต่าง ๆ

ข้อกำหนดของข้อมูลแบบอนุกรม [8 N 1] ความเร็วในการรับส่งข้อมูลสามารถเลือกได้โดยปรับ DIP Switch No. 1,2 ดังนี้

1 บิทเริ่ม	8 บิทข้อมูล	N พาริตี้	1 บิทหยุด

SW1	SW2	Baud Rate
0	0	1200
0	1	2400
1	0	4800
1	1	9600

DIP SWITCH 3

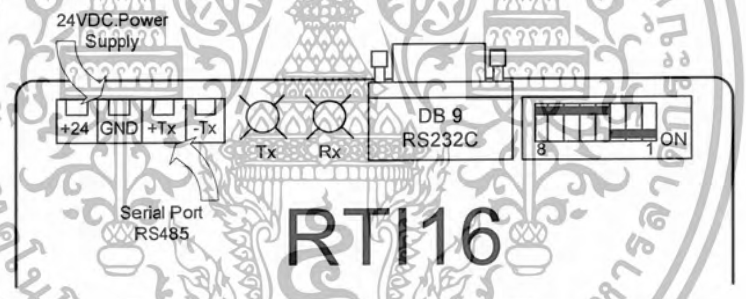
ถ้าเป็น RTI16 จะต้อง ON เสมอ

ON = 1 OFF = 0

การกำหนดตำแหน่ง Unit Number กำหนดได้โดยตั้ง DIP Switch No. 4,5,6,7,8 ดังนี้

SW4	SW5	SW6	SW7	SW8	Unit Number
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
↓	↓	↓	↓	↓	↓
1	1	1	1	1	31

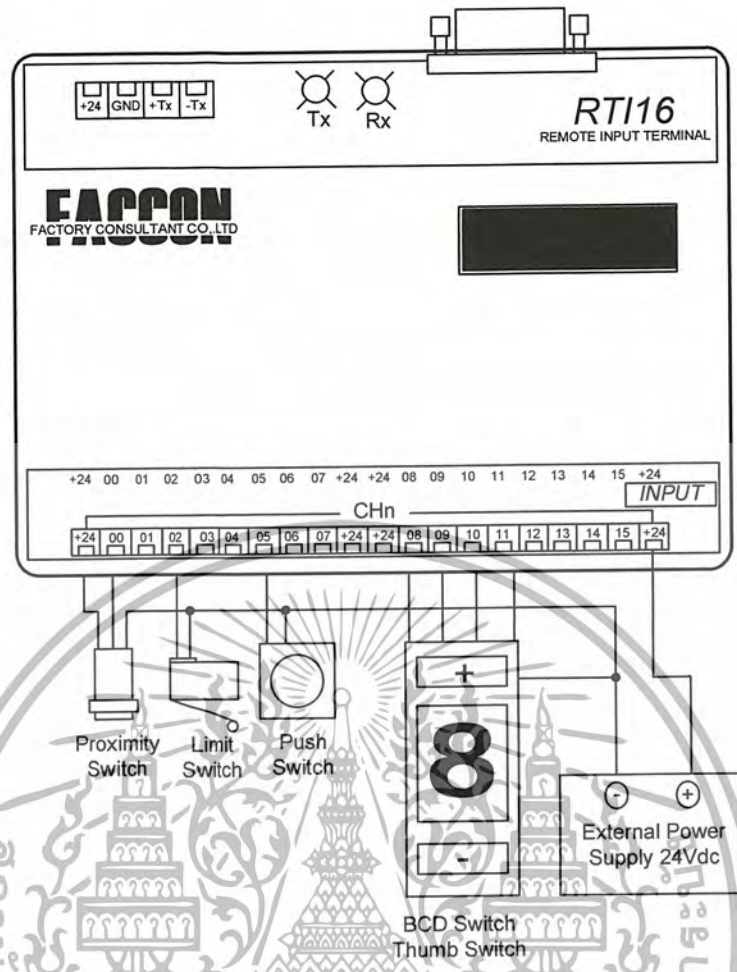
ขั้วต่อสายต่าง ๆ ของ RTI16



รูปที่ 3.8 แสดงขั้วต่อสายบนโมดูล

การนำไปใช้งานร่วมกับอุปกรณ์ตรวจจับต่าง ๆ สำหรับ RTI16

สัญญาณอินพุทเป็นสัญญาณที่ต้องการไฟแรงดัน 24 VDC. ชนิด Active Low หรือ Source-Current ที่กินกระแสต่อจุดโดยประมาณที่ 7 มิลลิแอมป์ ถ้าผู้ใช้งานจะใช้ตัวตรวจจับ แบบ Proximity Switch ,Photo Switch หรือเซ็นเซอร์แบบอื่น ๆ ที่มีเอาท์พุทเป็นทรานซิสเตอร์ แบบ NO. ชนิด NPN Type ก็สามารถต่อใช้งานได้ทันที สำหรับตัวตรวจจับชนิดที่มีเอาท์พุทเป็น Contact หรือสวิทซ์กดต่าง ๆ ทางด้านจุด Common ของอุปกรณ์จะเป็นสัญญาณไฟฟ้าไฟ GND และข้อต่อ Common ทางด้านอินพุทของชุด RTI16 จะเป็นไฟแรงดัน +24 VDC. ดังรูปที่แสดงข้างล่าง



รูปที่ 3.9 แสดงการต่อร่วมกับอุปกรณ์ตรวจจับต่าง ๆ

3.1.3 คุณสมบัติทั่วไปของ RTO16

เป็นโมดูลที่ออกแบบมาเพื่อส่งสภาวะจาก PLC รุ่น PC84SF หรือคอมพิวเตอร์ไปควบคุมอุปกรณ์ไฟฟ้าในโรงงานอุตสาหกรรม หรือในเครื่องจักรกล เช่น Motor ,Relay ,Solenoid Valve ,Lamp เป็นต้น ส่งค่าสภาวะได้จำนวน 16 จุด แบบ Opto- Isolator และวงจรขับสัญญาณเป็น Transistor ชนิด NPN Type ดังนั้นถ้าต้องการขับอุปกรณ์ที่ได้กล่าวมาแล้วข้างต้นจำเป็นจะต้องขับผ่าน Relay Board (RLY16) แสดงผลการเปิดปิดด้วย LED ทีละจุดและแสดงผลเป็นเลขฐานสิบหกได้โดยมี 7 Segment LED 4 หลัก ทำให้สะดวกต่อการสังเกตค่าสภาวะที่กำลังเป็นอยู่ การสื่อสารข้อมูลกระทำได้โดยตรงทั้งทาง พอร์ตอนุกรมแบบ RS232C. (Point To Point) และ RS485 (Multi Drop) ใช้กับแหล่งจ่ายไฟกระแสตรง 24 VDC 1 แอมป์

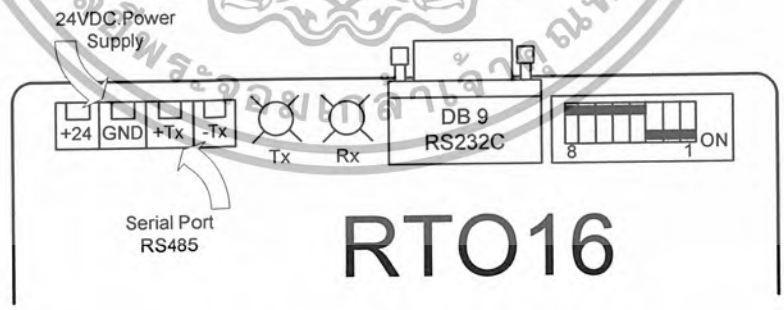
ตารางที่ 3.3 คุณสมบัติทางเทคนิคของ RTO16

Output Opto – Isolator	16 Points
Transistor NPN Type (Sink Current)	24 VDC.
Relay Driver (RLY16) Contact 125 VA	Option
Power Supply	24 VDC
Binary Display	16 Points
Hexadecimal Display 7 Segment LED	4 Digits
Operation Under CPU 89C51 at 11 Mhz	
Serial Port RS232C And RS485	Standard
Baud Rate 9600,4800,2400,1200 Bit/Sec	DIP Switch No.12
Communication	FAC –Talk Protocol
Unit Number 0-31	DIP Switch NO. 4,5,6,7,8

การกำหนดและเลือก DIP Switch ของ RTO16 ให้เป็น Unit Number ต่าง ๆ

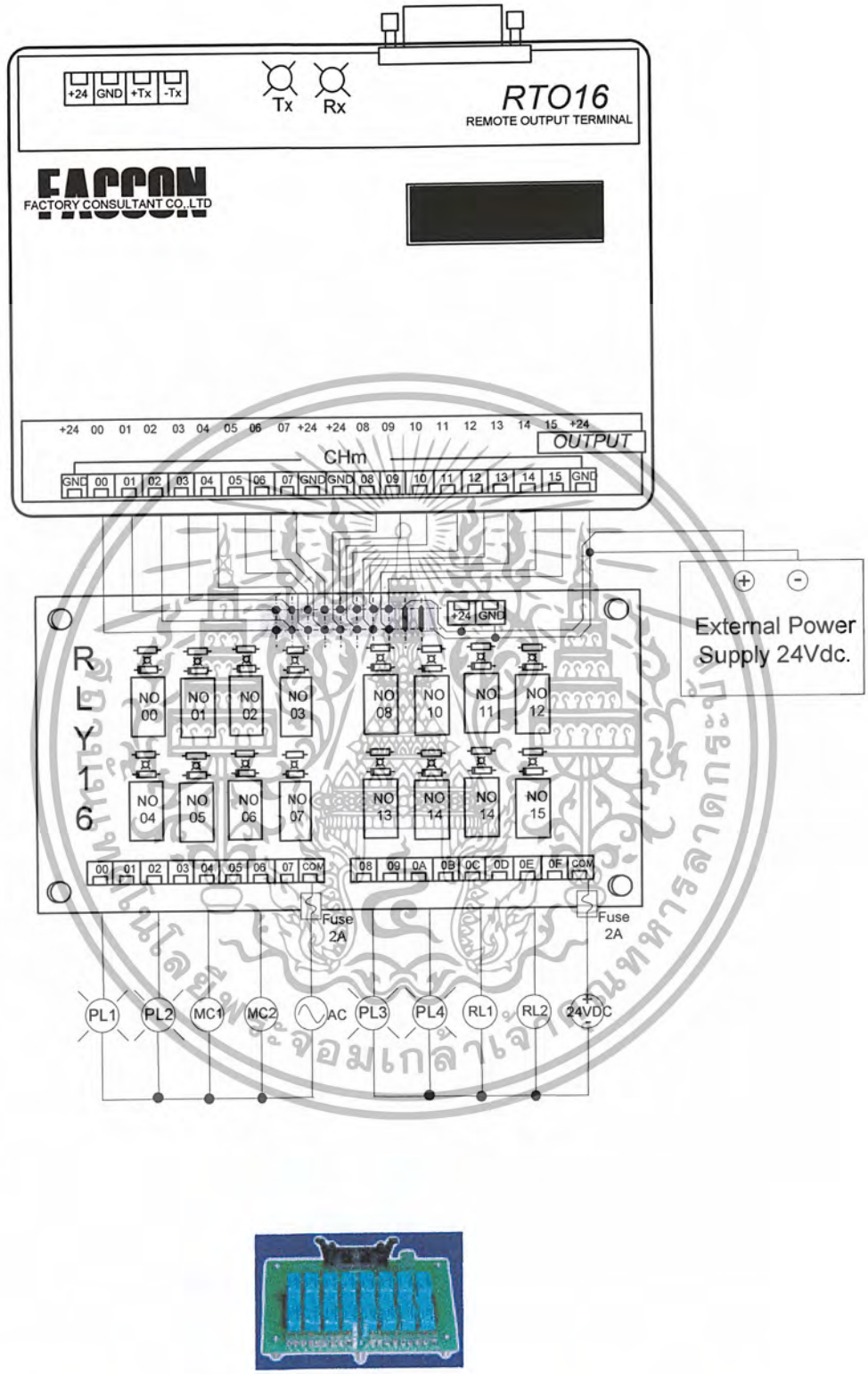
- ข้อกำหนดของข้อมูลแบบอนุกรม [8 N 1] ความเร็วในการรับส่งข้อมูลสามารถเลือกได้ โดยปรับ DIP Switch No. 1, 2 จะเหมือนกับ การปรับตั้งค่าที่ โมดูล RTI16
- การกำหนดตำแหน่ง Unit Number กำหนดได้โดยตั้ง DIP Switch No. 4,5,6,7,8 ก็เช่นเดียวกับโมดูล RTI16

ขั้วต่อสายต่าง ๆ ของ RTO16



รูปที่ 3.10 แสดงขั้วต่อสายบนโมดูล RTO16

การต่อใช้งานร่วมกันระหว่าง RTO16 และ RLY16 ควบคุมอุปกรณ์ไฟฟ้าแบบต่าง ๆ



รูปที่ 3.11 แสดงการต่อร่วมกันของ RTO16 และ RLY16 กับอุปกรณ์ไฟฟ้า

3.1.4 ข้อตกลงในการสื่อสาร (FAC-Talk Protocol)

ข้อตกลงในการสื่อสารถูกออกแบบให้อยู่ในรูปของรหัส ASCII จัดเป็นบล็อก โดยแบ่งเป็น

- บล็อกคำสั่ง (Command Block)
- บล็อกตอบสนอง (Response Block)

ในแต่ละบล็อกประกอบด้วยสาระสำคัญดังนี้

ตารางที่ 3.4 BlockCommand

@	Unit Number	Header	Data	*	BCS หรือ XX	[CR]

3.1.4.1 รูปแบบบล็อกคำสั่ง

ตารางที่ 3.5 แสดงรูปแบบบล็อกคำสั่ง

@	เป็นอักขระเริ่มต้นของบล็อก ค่าฐานสิบหกเท่ากับ 40 (Hex)
Unit Number	ตำแหน่งของโมดูลที่คอมพิวเตอร์อ้างอิง (0-31) เลือกจากการกำหนดที่ DIP Switch No. 4,5,6,7,8
Header	คำสั่ง / รหัสให้ปฏิบัติเป็น ASCII 2 อักขระ อันได้แก่ RD : หมายถึงคำสั่งในการอ่านข้อมูลจากโมดูล RTI16 WR : หมายถึงคำสั่งในการเขียนข้อมูลให้กับโมดูล RTO16 MD : หมายถึงคำสั่งในการอ่านชนิดของโมดูลว่าเป็น RTI16 หรือ RTO16
Data	ข้อมูลประกอบคำสั่ง / รหัสเป็น ASCII 2-4 อักขระ RD : ไม่ต้องมีข้อมูลประกอบ WR : ข้อมูลที่เป็นรหัส ASCII แทนค่าฐานสิบหก เช่น ข้อมูล 12AB(Hex) MD : ไม่ต้องมีข้อมูลประกอบ
*	เป็นอักขระปิดท้ายข้อมูล ค่าฐานสิบหกเท่ากับ 2A(Hex)
BCS หรือ XX	รหัสกำกับบล็อกเป็น ASCII 2 อักขระได้จากการ XOR ข้อมูล ASCII ทุก ๆ อักขระในแต่ละบล็อก ถ้าเป็น XX โมดูลจะรู้ว่า ไม่มีการตรวจสอบ BCS
[CR]	รหัส ASCII (Return Code) 1 อักขระ

3.1.4.2 รูปแบบบล็อกตอบสนอง

ตารางที่ 3.6 แสดงรูปแบบบล็อก

@	เป็นอักขระเริ่มต้นของบล็อก ค่าฐานสิบหกเท่ากับ 40(Hex) Unit Number ตำแหน่งของโมดูลที่คอมพิวเตอร์อ้างอิง (0-31) เลือกจากการกำหนดที่ DIP Switch No. 4,5,6,7,8
Header	คำสั่ง / รหัสให้ปฏิบัติเป็น ASCII 2 อักขระ อันได้แก่ RD : หมายถึงคำสั่งที่โมดูลตอบกลับเพื่อส่งข้อมูลให้ PC84SF หรือคอมพิวเตอร์จาก RTI16 MD : หมายถึงคำสั่งที่โมดูลตอบกลับเพื่อบอกชนิดของโมดูลว่าเป็น RTI16 หรือ RTO16 ER : เมื่อการปฏิบัติคำสั่งใด ๆ เกิดข้อผิดพลาดโมดูลจะตอบกลับ Data ข้อมูลประกอบคำสั่ง / รหัสเป็น ASCII 2-4 อักขระ RD : ข้อมูลที่ตอบกลับมาเป็นค่าสภาวะปัจจุบันขณะอ่าน เป็นข้อมูลที่เป็นรหัสASCII แทนค่าฐานสิบหก เช่น ข้อมูล 12 AB (Hex) ชุดข้อมูล ASCII จะเป็น 31324142 เป็นต้น MD : 00 เป็น RTO16 [ASCII 2 อักขระ] 01 เป็น TRI16 ER : 00 : Over Range อักขระในบล๊อค ไม่อยู่ในพิกัด 01 : Format Error รูปแบบผิด 02 : Block C Check Sum Error รหัสกำกับ บล็อคไม่ถูกต้อง 03 : Unknown Command ไม่รู้จักคำสั่งนี้ 04: Execute Error ปฏิบัติการไม่ได้ หรือมีข้อผิดพลาดอื่นๆ
*	เป็นอักขระปิดท้ายข้อมูล ค่าฐานสิบหกเท่ากับ 2A(Hex)
BCS หรือ XX	รหัสกำกับบล็อกเป็นASCII 2 อักขระได้จากการ XORข้อมูล ASCII ทุก ๆ อักขระในแต่ละบล็อก ถ้าเป็น XX โมดูลจะรู้ว่า ไม่มีการตรวจสอบ BCS
[CR]	รหัส ASCII (Return Code) 1 อักขระ

การคำนวณรหัสกำกับบล็อค (Block Check Sum Calculation) BCS โดยโปรแกรม Pascal

```
Program Block_Check_Sequence_Calculate ;
Var     j,k,l,m,n       :Byte ;
        Chksum          :Byte ;
        BlockCommand : String[128] ;
Begin
      Clrscr ;
         BlockCommand := '@02WR90AF*' ;
          Chksum := ord(BlockCommand[1] ;
            For j := 2 LENGTH(BlockCommand) do
              Begin
                Chksum := Chksum XOR ord(BlockCommand[j]);
                Writeln(BlockCommand[j],' ',Chksum) ;
              End ;
End.
```

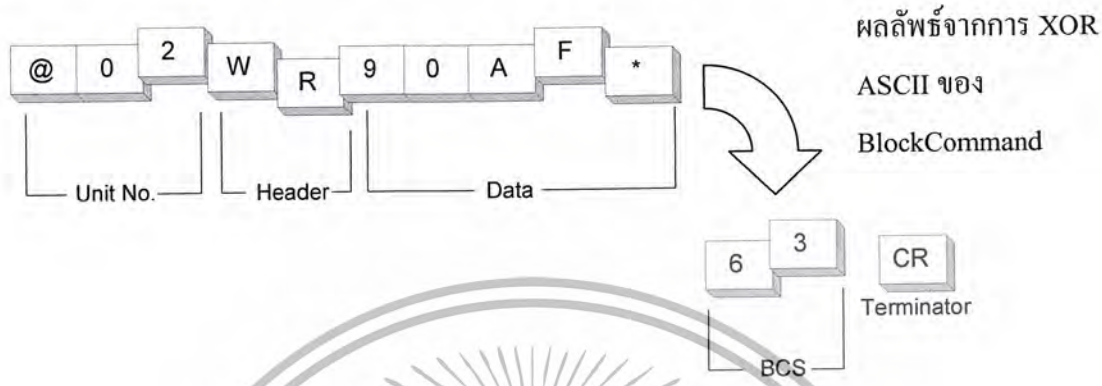
การคำนวณ BCS ด้วยโปรแกรม Visual Basic6

วิธีการเขียนจะมีลักษณะที่ค่อนข้างจะคล้ายกันกับ ภาษา Pascal(ใช้เป็นพื้นฐาน) ก็คือมุ่งเน้นเพื่อ นำเอาค่ารหัส ASCII ที่ อยู่ใน Command Block มาคำนวณ XOR กันทุกตัวเพื่อนำไปใช้กำกับที่ตำแหน่ง BCS ภายใน Block เป็นต้น ดังต่อไปนี้

```
Private Sub T_RTI1_Timer()              'โปรแกรมย่อย Sub Timer1
Dim Check As String, Resum As Integer, I As Integer     'ประกาศตัวแปร
Dim Str As String
    If Setting.MSComm1.PortOpen = False Then        'ตรวจสอบ Port ว่าเปิด/ปิด
       Setting.MSComm1.PortOpen = True              'เปิด Port
    End If
    Check = Setting.MSComm1.Input                   'รับข้อมูลจาก Buffer
    If (Mid(Check, 2, 2) = "01") And (Len(Check) = 13) Then     'ตัดเอาข้อมูล
        For I = 1 To 10
        Resum = Resum Xor Asc(Mid(Check, I, 1))     'XOR ค่า ASCII ทุกค่า
        Next I
        Str = Hex(Resum)                            'แปลงเป็น Hex
    End Sub
```

3.1.4.3 Block Checksum (BCS) Calculation

BCS เป็นข้อมูลขนาด 8 บิท และถูกแปลงเป็น ASCII จำนวน 2 อักขระ ข้อมูล 8 บิทนี้ได้จากผลลัพธ์ของการ Exclusive OR ตามลำดับเรื่อยมาตั้งแต่อักขระเริ่มต้นไปจนถึงอักขระสุดท้ายในแต่ละ Block แสดงการคำนวณดังนี้



รูปที่ 3.12 การคำนวณหาค่า BCS

- การเขียนข้อมูลไปยัง RTI16 โดย O/P Module ถูกตั้งค่าตำแหน่งไว้ที่ 02 และต้องการให้ข้อมูลเป็น 90AF (Hex) จะต้องต้องทำการจัดรูปแบบบล็อคคำสั่งเพื่อส่งให้ RTO16 ดังนั้น "@02WR90AF*63,[CR] การคำนวณ BCS จะได้ค่า 63 หรือ "@02WR90AF*XX,[CR] ก็ได้ และเมื่อปฏิบัติเสร็จสิ้นไม่มีข้อผิดพลาดใด ๆ จะไม่มีข้อมูลใดตอบกลับออกมา

- การอ่านข้อมูลจาก RTI16 โดยถูกตั้งค่าตำแหน่งไว้ที่ 10 จะต้องทำการจัดรูปแบบบล็อกคำสั่งเพื่อไปถามค่าของข้อมูลปัจจุบันดังนี้ "@0ARD*0D",[CR] คำนวณ BCS ได้ 0D หรือ "@0ARD*XX",[CR] สมมุติว่าข้อมูลปัจจุบันของ RTI16 นั้นเป็น 55AA(Hex) Module จะตอบกลับออกมาดังนี้ "@0ARD55AA*0D",[CR]

3.2 การสื่อสารผ่านพอร์ตอนุกรมด้วย Application Program (Visual Basic6)

3.2.1 คัสตอมคอนโทรล (MSComm คอนโทรล)

สำหรับการใช้งาน Visual Basic ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual Basic จะมีคัสตอมคอนโทรล สำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยใน Visual Basic เวอร์ชัน 2 และ เวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCMM16.OCX สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ MSCMM32.OCX สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับใน Visual Basic เวอร์ชัน 5 จะมีเพียง MSCMM32.OCX เท่านั้นเพราะถูกออกแบบมาใช้งานกับระบบปฏิบัติการ 32 บิตMSComm

จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือ การสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (Even-Driven Communications) เป็นการจัดรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใดเช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรม หรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier Detect (DCD) หรือขา Request To Send เหตุการณ์ ONCOMM ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียด ในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่ เกิดขึ้นด้วยการดูค่าการเปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชั่นต่างๆไปเรียบร้อยแล้วซึ่งวิธีนี้ ใช้งานได้ดีในกรณี โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ล่ะพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติ (property) มากมายหลากหลายตัว แต่สามารถทำความเข้าใจได้ไม่ยากดังนี้

CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM 1, COM 2, COM 3, COM 4)

รูปแบบการใช้งาน

Object.CommPort [= value]

โดย Value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1 – 16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึง อุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรม ก่อนที่ใช้คำสั่ง OpenPort

Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด, พาริตี้, จำนวนบิตของข้อมูล, จำนวนของบิตปิดท้าย

รูปแบบการใช้งาน

Object.setting [= value]

ค่า Value มีชนิดของข้อมูลเป็นแบบ String มีรูปแบบเป็น “BBBB,P,D,S,” โดย BBBB เป็นค่าอัตราบอด , P เป็นค่าพาริตี้บิต, D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “3600, N,8,1”

ค่าบอดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที
300 บิตต่อวินาที
600 บิตต่อวินาที
1,200 บิตต่อวินาที
2,400 บิตต่อวินาที
9,600 บิตต่อวินาที (ค่าปกติ)
14,400 บิตต่อวินาที
19,200 บิตต่อวินาที
28,800 บิตต่อวินาที
38,400 บิตต่อวินาที (สงวน)
56,000 บิตต่อวินาที (สงวน)
128,000 บิตต่อวินาที (สงวน)
256,000 บิตต่อวินาที (สงวน)

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้มีดังนี้

สัญลักษณ์	รายละเอียด
E	พาริตี้คู่ (EVEN)
M	ลอจิก "1" (MARK)
N	ไม่ใช้ (ค่าปกติ)
O	พาริตี้คี่ (Odd)
S	ลอจิก "0" (Space)

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่า คือ 4,5,6,7 และ 8 (เป็นค่าปกติ)
ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่า คือ 1 (เป็นค่าปกติ), 1.5 และ 2

ตัวอย่าง การใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

```
MSComm1.Setting = "9600,N,8,1"
```

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด "" เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

Object.PortOpen [= Value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงผิดพลาด error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้วโปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบลัติของ DTREnable หรือ RTSEnable จะถูกเซตเป็น False หลังจากปิดพอรืตแต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

ตัวอย่าง การใช้คำสั่งเปิดพอร์ต เพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM 1 และมีบอดเรต 9,600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต มีดังนี้

```
MSComm1.Setting = "9600,n,8,1"
MSComm1.CommPort = 1
MSComm1.PortOpen = True
```

Input

อ่านค่าและลบค่าบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบ Variant ถ้าInputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรม แสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

```
Private Sub Command_Click()
```

```
Dim InString as String
    MSComm1.InputLen = 0              ' Retrieve all available data.
    If MSComm1.InBufferCount Then     ' Check for data.
    Instring = MSComm1.Input          ' Read data.
  End If
End Sub
```

InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InBufferCount [= Value]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

หมายเหตุ อย่าสับสนระหว่างคำสั่ง InBufferSize และ InbufferCount คำสั่ง InBufferSize นั้นใช้เพื่อการกำหนดขนาดของบัฟเฟอร์ภาครับ

InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

Object.InBufferSize [= Value]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้ หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่น ๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่า บัฟเฟอร์ภาครับที่น้อยเกินไปจะทำให้เกิดการโอเวอร์โฟลงหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่มีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลงแล้วจึงค่อยปรับเพิ่มค่าขนาดของบัฟเฟอร์โฟลงให้มีค่ามากขึ้น

ImputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InputLen [= Value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ "0" จะทำให้ คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับ

จำนวน InputLen คำสั่ง Input จะส่งค่าว่าง (“”) กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรม การอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

```
Private Command_Click()
Dim CommData as String
MSComm1.InputLen = 10          'Specify a 10 charater block of data.
CommData = MSComm1.Input   'Read data.
End Sub
```

InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

Object.InputMode [= Value]

คุณสมบัติ InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

comInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น “0” และค่าเริ่มต้นของการรับค่าข้อมูลจะเป็นค่านี้

comInputModeBinary สำหรับข้อมูลอื่น ๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านค่าข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาเรย์ ชนิดข้อมูลเป็นแบบไบต์

```
Private Sub Command_Click()
Dim Buffer as Variant
Dim Arr  () as Byte
MSComm1.CommPort = 1    ' Set and open port
MSComm1.PortOpen = True
MSComm1.InputMode = comInputModeBinary    ' Set InputMode to read binary data
Do Until MSComm1.InBufferCount < 10  'Wait until 10 bytes are in the
input buffer
      DoEvants
Loop
```

```
Buffer = MSComm1.Input          'Store binary data in buffer
Arr = Buffer                    'Assign to byte array for processing
End Sub
```

OutPut

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

Object. Output [= value]

ค่า Value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ OutPut สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับข้อมูลไบนารี่จะต้องกำหนดชนิดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte ตัวอย่างโปรแกรม เป็นค่าที่ป้อนจากคีย์บอร์ดทุกๆตัวไปยังพอร์ตอนุกรม โปรแกรม

```
Private Sub Command1_KeyPress(KeyAscii As Integer)
Dim Buffer As Variant
'Set and open port
MSComm1.Comport = 1
MSComm.PortOpen = True
Buffer = Chr$(KeyaAscii)
MSComm1.Output = Buffer
End Sub
```

OutBufferCount

คืนค่าจำวนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่ง และสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งานคำสั่ง

Object.OutBufferCount [= value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่าOutBufferCountเท่ากับ “0”

OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์

รูปแบบการใช้งานคำสั่ง

Object.OutBufferSize [= object]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

Parity Replace

Object.ParityReplace [= value]

บิตพาริตี้ เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูล เพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบวกบิตทุกบิตที่มีค่าลอจิก “1” ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก “1” เป็นเลขคู่หรือเลขคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการรับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ Parity Replace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า Parity Replace ให้เป็นค่าว่าง (“”) จะเป็นการยกเลิกการใช้งาน Parity Replace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อมูลผิดพลาด

Parity Replace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะกำหนด ได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใด ๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0 -255

DTREnable

ใช้ในการอีนาเบิลขา Data Terminal Ready (DTR) โดยที่สัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object. DTREnable [= value]

ค่า Value เป็นค่าที่แสดงสถานะ True หรือ False เพื่อที่จะอีนาเบิลขา DTR โดย

True หมายถึงการอีนาเบิลขา DTR

False หมายถึงการดิสอีนาเบิลขา DTR (เป็นปกติ)

เมื่อขา DTR ถูกกำหนดสถานะให้เป็นTrue ที่ขา DTR จะมีสถานะเป็นลอจิก “1” เมื่อกำหนดการเปิดพอร์ตและจะมีสถานะเป็น “0” เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดเป็น False ที่ขา DTR จะมีสถานะลอจิก “0” ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” จะเป็นยกเลิกการติดต่อ (วางหู)

RTSEnable

ใช้เพื่ออีนาเบิลขา Request To Send (RTS)โดยที่ขาRTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object. RTSEnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อที่อีนาเบิลหรือดิสเอเบิลขา RTS โดย

True หมายถึงการอีนาเบิลขา RTS

False หมายถึงการดิสอีนาเบิลขา RTS (เป็นปกติ)

เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1” เมื่อเปิดพอร์ตและจะมีสถานะลอจิก “0” เมื่อปิดพอร์ต

EOFEnable

เป็นการกำหนดให้ MSComm รอสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file: EOF) ระหว่างการรับอินพุทเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุทจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ CommEvEOF

รูปแบบการใช้งาน

Object. EOFEnable [= value]

โดย Value เป็นค่าสถานะ True หรือ False เพื่ออีนาเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีสถานะ ลอจิก “0” หรือ “1” โดยที่ค่าที่อ่านจะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น “1” ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น “0”

รูปแบบการใช้งาน

Object. CTSHolding

เมื่อขา CTS เป็นลอจิก “0” (CTSHolding = False) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น ComEventCTSTO (Clear To Send Timeont) และกระตุ้นให้เกิดเหตุการณ์ OnComm

CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิก “0” หรือ “1” โดยที่ค่าที่อ่านจะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิกเป็น “1” ถ้าค่า CDHolding เป็น False ขา DCD จะมีสถานะลอจิกเป็น “0”

รูปแบบการใช้งาน

Object. CDHolding

เมื่อขา DCD เป็นลอจิก “1” (CDHolding = True) และเกิดไทม์เอาต์คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น ComEventCDTO (Data Carrier Detect Timeont Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR (DSR) ได้ว่ามีสถานะลอจิก “0” หรือ “1” โดยที่ค่าที่อ่านจะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิกเป็น “1” ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิกเป็น “0”

รูปแบบการใช้งาน

Object. DSRHolding

เมื่อขา DSR เป็นลอจิก “1” (DSRHolding = True) และเกิดไทม์เอาต์คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น ComEventDSRTO (Data Set Relay Timeont) และกระตุ้นให้เกิดเหตุการณ์ OnComm

Handshanking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบการใช้คำสั่ง

Object.Hanshaking [= value]

ค่าตัวแปร Valve ที่ใช้กำหนดค่ากำหนดได้ 4 รูปแบบด้วยกัน คือ

1. comNone ค่าที่กำหนด คือ 0 เป็นการกำหนดให้ไม่มีแฮนด์เช็ก(เป็นค่าเริ่มต้น)
2. comXonXoff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้มี แฮนด์เช็กแบบ XON/XOFF
3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request To Send/Clear-To Send)
4. comRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send - XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายในระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับโปรแกรมที่ใช้งานจะทำการอ่านข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติของ Handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่รับได้มานั้นไม่สูญหาย เมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือ Overflow โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งเข้ามาให้

Break

ใช้ในการกำหนดการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean รูปแบบการใช้งาน

Object.Break [= value]

โดยที่ Value เป็นค่า Boolean ถ้า Value = True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า Value = False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False ตัวอย่างโปรแกรม เป็นวิธีการส่งสัญญาณ Break ออกเป็นช่วงเวลาสั้นๆที่ 1/10 ของวินาที

โปรแกรม

```
MSComm1.Break = True                    ' Set the Break condition.
Duration! = Timer + 0.1                 ' Set duration to 1/10 second.
Do Until Timer > Duration!              ' Wait for the duration to pass.
  Dummy = DoEvents()
Loop
MSComm1.Break = False                   ' Clear the Break condition.
```

เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลง เพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้น ตัวอย่างโปรแกรม เป็นโปรแกรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดง

```
Private Sub MSComm_OnComm()
        Select Case MSComm1.CommEvent
        ' Handle each event or error by placing
        ' code below each case statement
        ' Errors
                Case commEvenBreak          'A Break was received.
                Case commEvenCDTO           'CD (RLSD) Timeout
                Case commEvenCTSTO 'CTR Timeout
                Case commEvenDSRTO 'DSR Timeout
```

```
        Case commEvenFrame          'Framing Error
        Case commEvenOverrun 'Data Lost.
        Case commEvenRxOver 'Receive buffer overflow
        Case commEvenRxParity       'Parity Error.
Case comEvenTxFull
        ' Events
        Case comEvCD        'Chang in the CD line
        Case comEvCTS               'Chang in the CTS line
        Case comEvDSR               'Chang in the DSR line
        Case comEvRing              'Chang in the Ring Indicator
        Case comEvReceive           'Received Rthreshold # of chars
        Case comEvSend          'Sthreshold number in the 'transmit buffer.
Case comEvEof           'An EOF character was found in the input stream
        End Select
End Sub
```

ค่าคงที่คุณสมบัติของคอนโทรล MsComm

ค่าคงที่สำหรับคุณสมบัติ Hanshanke

ค่าคงที่	ค่า	รายละเอียด
comNone	0	ไม่ใช้การตรวจสอบแฮนด์เช็ก
comXonXoff	1	ใช้การตรวจสอบแฮนด์เช็กแบบ XonXoff
comRTS	2	ใช้การสอบแฮนด์เช็กผ่านทางขา RTS และCTS
comRTSXonXOff	3	กำหนดการตรวจสอบแฮนด์เช็กทั้งแบบ RTS,CTS และXon/Xoff

ค่าคงที่สำหรับคุณสมบัติ OnComm

ค่าคงที่	ค่า	รายละเอียด
comEvSend	1	ส่งค่าเหตุการณ์(send event)
comEvReceive	2	รับค่าเหตุการณ์(receive event)
comEvCTS	3	มีการเปลี่ยนแปลงที่ขา CTS
comEvDSR	4	มีการเปลี่ยนแปลงที่ขา DSR
comEvCD	5	มีการเปลี่ยนแปลงที่ DCD
comEvRing	6	ตรวจจับสัญญาณกระดิ่งของโทรศัพท์

comEvEOF 7 ตรวจพบตำแหน่งท้ายสุดของไฟล์(End Of file)

ค่าคงที่สำหรับคุณสมบัติ Error

ค่าคงที่	ค่า	รายละเอียด
comEventBreak	1001	ได้รับสัญญาณ Break
comEventCTSTO	1002	ขา CTS เกิดไทม์เอาท์
comEventDSRTO	1003	ขา DSR เกิดไทม์เอาท์
comEventFrame	1004	เกิดข้อผิดพลาดที่เฟรมข้อมูล
comEventOverrun	1006	พอร์ตอนุกรมเกิดโอเวอร์รัน
comEventCDTO	1007	ขา DCD เกิดไทม์เอาท์
comEventRxOver	1008	บัฟเฟอร์รับข้อมูลเกิดโอเวอร์โฟลว
comEventRxParity	1009	เกิดข้อผิดพลาดที่พาริตี้ (Parity error)
comEventTxFull	1010	บัฟเฟอร์ส่งข้อมูลเต็ม

การใช้ MSComm เพื่อการติดต่อฮาร์ดแวร์

จากรายละเอียดของ MSComm ที่กล่าวไปในตอนต้นนั้น จะเห็นได้ว่าวิธีการที่จะอ่านค่าหรือ เขียนค่าไปยังขาสถานะและขาควบคุมของพอร์ตอนุกรมสามารถทำได้ง่ายดายมาก โดยใช้คำสั่งเหล่านี้

DTREnable สำหรับสั่งให้ขา DTR มีลอจิก “0” หรือ “1”

RTSEnable สำหรับสั่งให้ขา RTS มีลอจิก “0” หรือ “1”

CTSHolding สำหรับอ่านค่าสถานะจากขา CTS ว่ามีลอจิก “0” หรือ “1”

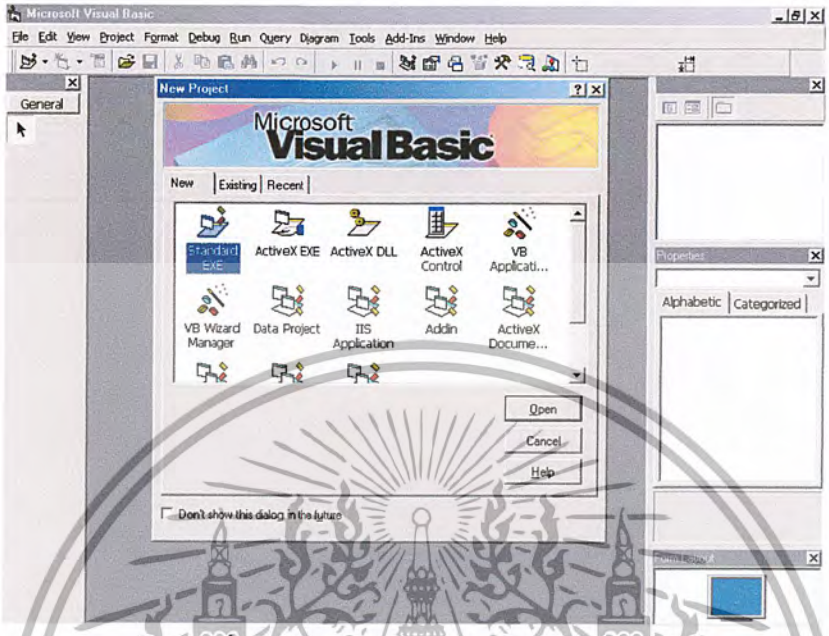
CDHolding สำหรับอ่านค่าสถานะจากขา DCD ว่ามีลอจิก “0” หรือ “1”

DSRHolding สำหรับอ่านค่าสถานะจากขา DSR ว่ามีลอจิก “0” หรือ “1”

Break สำหรับการสั่งให้ขา Txd มีลอจิก “0” หรือ “1”

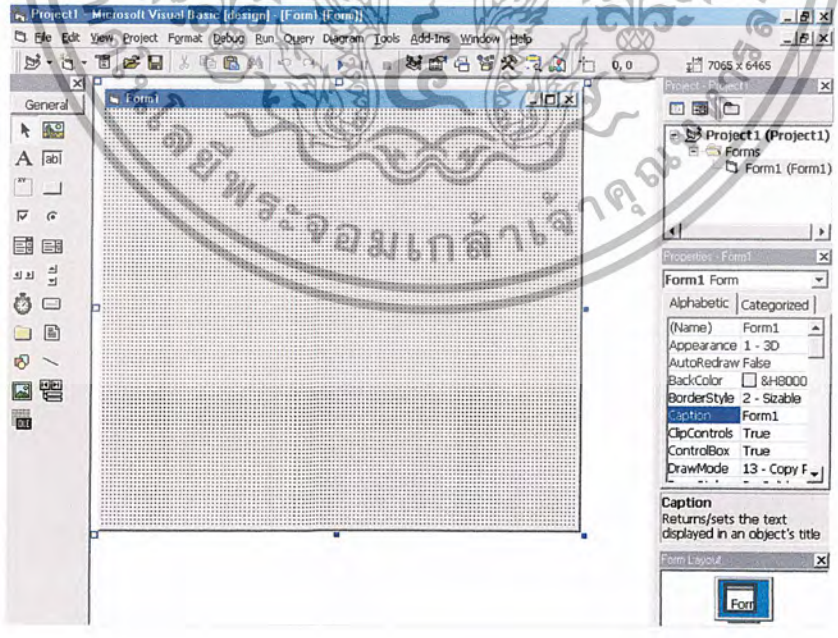
3.3 พื้นฐานการเขียนโปรแกรมด้วย Visual Basic

3.3.1 การเขียนโปรแกรมด้วย Visual Basic ขั้นพื้นฐาน



รูปที่ 3.13แสดงหน้าจอของ Visual Basic 6.0

โดยจะเลือก Standart.EXE ในการเขียนโปรแกรมที่จะ Run บนวินโดวส์ทั่วไป จะปรากฏหน้าจอดังรูปที่ 3.15



รูปที่ 3.14 แสดงส่วนต่างๆของหน้าจอการทำงาน

3.3.2 รายละเอียดของส่วนประกอบต่างๆของหน้าจอ

1. MenuBar เก็บคำสั่งที่สามารถใช้งานได้ทั้งหมดใน Visual Basic 6.0 ประกอบด้วยเมนูการทำงาน File, View และ Windows

2. ToolBar ประกอบด้วยปุ่มคำสั่งต่างๆ ที่ช่วยให้งานคำสั่งของ Visual Basic 6.0 ได้รวดเร็วขึ้น

3. ToolBox เป็นที่แสดงเครื่องมือต่างๆ ที่เรียกว่าคอนโทรล ซึ่งเป็นเครื่องมือที่สามารถนำไปวางบนฟอร์มได้ เพื่อออกแบบหน้าจอของโปรแกรม (เรียกว่าส่วนติดต่อกับผู้ใช้ หรือ User Interface)

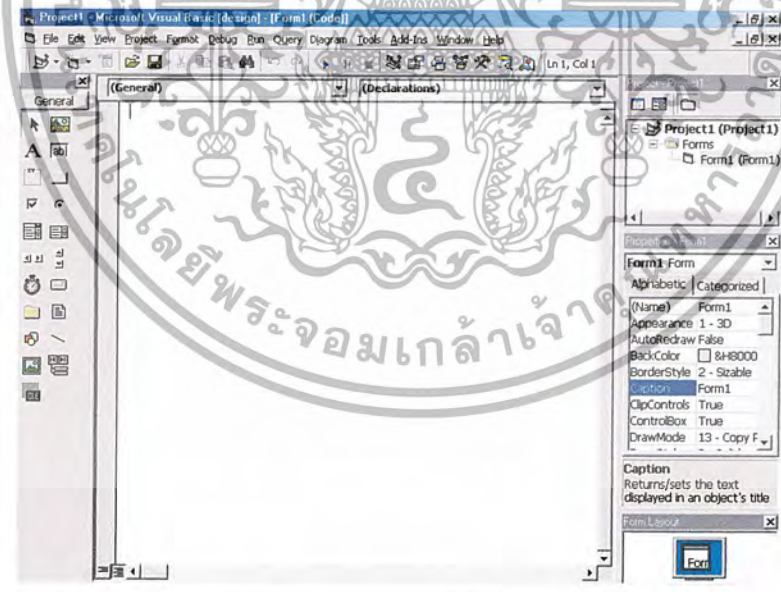
4. Form ฟอร์มที่ใช้ในการออกแบบเป็นหน้าต่างที่ใช้ในการออกแบบหน้าจอโปรแกรม

5. Project Explorer เป็นหน้าต่างที่แสดงโมดูล (Modules) ต่างๆที่มีอยู่ในโปรเจคทั้งหมด

6. Properties windows เป็นหน้าต่างที่แสดงคุณสมบัติของคอนโทรลที่เลือกอยู่ในขณะนั้น

7. From Layout เป็นหน้าต่างที่แสดง ฟอร์มบนจอภาพ ทำให้จัดตำแหน่งของฟอร์มได้สะดวกขึ้น

สำหรับส่วนประกอบที่สำคัญอีกส่วนหนึ่ง คือ หน้าต่าง Code Editor ซึ่งเป็นหน้าต่างที่ใช้ในการพิมพ์คำสั่งเพื่อควบคุมการทำงานของโปรแกรม สามารถเรียกหน้าต่าง Code Editor ขึ้นมาตามขั้นตอนดังรูปที่ 3.16



รูปที่ 3.15 แสดงหน้าต่าง Code Editor

3.3.3 การทำงานกับโปรเจค

เนื่องจาก Visual Basic 6.0 จำเป็นจะต้องทำงานเกี่ยวข้องกับไฟล์โปรเจคทุกครั้งที่สร้างโปรแกรมขึ้นมา จึงจำเป็นที่จะต้องทำความเข้าใจในคำสั่งต่างๆที่ Visual Basic 6.0 grnเพื่อช่วยในการทำงานต่างๆได้รวดเร็วยิ่งขึ้น

ไฟล์ประเภทต่างๆที่มีในโปรเจคของ Visaul Basic 6.0

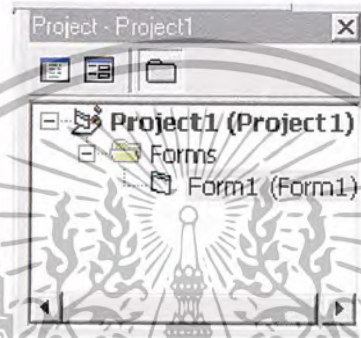
โปรเจคเป็นไฟล์ที่เก็บฟอร์ม และโมดูลต่างๆ ของโปรเจคที่สร้างขึ้นมา จะมีไฟล์ในรูปแบบต่างๆที่เป็นไปได้ดังตารางที่ 3.7

ตารางที่ 3.7 แสดงประเภทของไฟล์ในโปรเจคของ Visual Basic 6.0

ชนิดของไฟล์	คำอธิบาย	ส่วนขยายไฟล์
ไฟล์กลุ่มโปรเจค	เป็นไฟล์ที่ใช้เก็บว่ามีโปรเจคอะไรอยู่บ้าง (ต้องมากกว่า1โปรเจคขึ้นไป)	.vbg
ไฟล์โปรเจค	เป็นไฟล์หลักโปรเจคต่างๆของแอปพลิเคชั่น	.vbp
ไฟล์ของฟอร์ม	เป็นไฟล์ที่เก็บข้อมูลเกี่ยวกับฟอร์มและคำสั่งจัดการอีเวนต์สำหรับฟอร์มนั้นๆ	.frm
ไฟล์ไบนารี่ฟอร์ม	เป็นไฟล์ที่เก็บคุณสมบัติที่เป็นไบนารี่ของฟอร์ม	.frx
ไฟล์โมดูลมาตรฐาน	ส่วนใหญ่จะใช้เก็บค่าคงที่ ตัวแปร โปรแกรมย่อย ที่ให้โมดูลอื่นสามารถใช้งานได้	.bas
ไฟล์คลาสโมดูล	ในการสร้างออบเจคที่มีลักษณะต่างๆตามที่ต้องการ	.cls
ไฟล์ Active Control	จะเป็นไฟล์ของคอนโทรล ActiveX ซึ่งเป็นคอนโทรลที่สร้างขึ้นมาเอง และสามารถนำไปใช้ในแอบพลิเคชั่นทั่วไปที่สร้างขึ้นมาใหม่ได้	.ctl
ไฟล์ของ ActiveX document	จำเป็นไฟล์ของแอบพลิเคชั่น ที่สามารถนำไปแสดงในโปรแกรม Web Browser	.dob
ไฟล์ของ Property Page	จะเป็นไฟล์ของ Property Pageที่ใช้แสดงคุณสมบัติของคอนโทรล	.pag

3.3.4 การทำงานกับ Project Explorer

หน้าต่าง Project Explorer เป็นหน้าต่างที่แสดงให้เห็นโครงสร้างของโปรเจคที่เราทำงานด้วย โดยจะแสดงไฟล์โมดูล รวมทั้งคอมโพแนนซ์ต่างๆที่ใช้ในโปรเจค ดังนั้นทุครั้งที่มีการเพิ่มและลบไฟล์ต่างๆในโปรเจค Visual Basic 6.0 จะเปลี่ยนข้อมูลเกี่ยวกัยโปรเจคที่แสดงในหน้าต่าง Project Explorer ให้สอดคล้องตามไปด้วย และเมื่อมีการบันทึกโปรเจค Visual Basic 6.0 จะบันทึกไฟล์ต่างๆตามที่แสดงใน Project Explorer สำหรับ Project Explorer จะมีหน้าต่างแสดงดังรูปที่ 3.17



รูปที่ 3.16 แสดงหน้าต่าง Project Explorer

การออกแบบหน้าจอใน Visual Basic 6.0

การสร้างโปรแกรมด้วย Visual Basic 6.0 ในขั้นตอนแรกจะเป็นการออกแบบจอที่ติดต่อกับผู้ใช้ ซึ่งเป็นการนำคอนโทรลที่มีอยู่มาออกแบบฟอร์มให้เหมาะสม จะเห็นได้ว่าทั้งฟอร์มและคอนโทรลนับเป็นเครื่องมือพื้นฐานในการเขียนโปรแกรมด้วย Visaul Basic 6.0

3.3.4.1 ความรู้เกี่ยวกับฟอร์ม และคอนโทรลพื้นฐานต่างๆ

ฟอร์ม(Form)

ฟอร์มจะเป็นเครื่องมือที่เราจะต้องทำงานด้วยบ่อยมาก ในการสร้างโปรแกรมด้วย Visual Basic โดยที่ฟอร์มจะเป็นหน้าต่างที่ผู้ใช้ติดต่อทำงานด้วยผ่านมาคอนโทรลต่างๆ ที่วางบนฟอร์ม

รูปที่ 3.17 แสดงหน้าต่างฟอร์มของ Visual Basic

ลาเบล (Label)

ลาเบลเป็นคอนโทรที่ใช้ในการแสดงข้อความตามที่ต้องการบนฟอร์ม ซึ่งอธิบายข้อมูลบางอย่างแก่ผู้ใช้ โดยที่ผู้ใช้ไม่สามารถแก้ไขในลาเบลได้ในตอน Run โปรแกรม แต่สามารถแก้ไขได้โดยใช้คำสั่งโปรแกรมตอน Run และตอนออกแบบโปรแกรมเท่านั้น ส่วนใหญ่จะใช้ลาเบลในการแสดงข้อความอีบายการทำงานรวมทั้งแสดงสถานะบางอย่าง

รูปที่ 3.18 แสดงคอนโทรลลาเบล

เท็กบ็อกซ์ (Textbox)

เท็กบ็อกซ์ เป็นคอนโทรลที่ใช้ในการรับข้อมูลจากผู้ใช้ที่เข้ามาในโปรแกรม รวมทั้งความสามารถแสดงผล และให้ผู้ใช้แก้ไขข้อมูลได้ด้วย

รูปที่ 3.19 แสดงคอนโทรลเท็กบ็อกส์

ปุ่มคำสั่ง (Command Button)

ปุ่มคำสั่งมีหน้าที่หลักคือ ตอบสนองต่อการคลิกเม้าส์ของผู้ใช้ที่สั่งงานมายังโปรแกรมว่าต้องการให้โปรแกรมทำอะไรตอบกลับไป

รูปที่ 3.20 แสดงคอนโทรลปุ่มคำสั่ง

เช็กบ็อกส์(Cheak Box)

คอนโทรลเช็กบ็อกส์ใช้สำหรับเป็นตัวเลือกที่ผู้ใช้สามารถเลือกได้ 2 สถานะ คือ จะเช็ก(ไม่ให้มีเครื่องหมายถูกแสดง)หรือไม่เช็ค(ไม่มีเครื่องถูก)โดยสามารถเช็คได้หลายเช็กบ็อกส์พร้อมกันซึ่งต่างจากการใช้ออบชั่นบัตทอน ที่จะเลือกได้เพียง 1 ตัวเลือกเท่านั้น

รูปที่ 3.21 แสดงคอนโทรลเช็กบ็อกส์

ออบชั้นบัททอน(Option Button)

คอนโทรลออบชั่นบัททอน ทำหน้าที่คล้ายกับเช็กบ็อกส์ แต่คอนโทรลนี้จะสามารถเลือกได้เพียงตัวเลือกเดียวเท่านั้น ในกลุ่มตัวเลือกที่เลือกไว้ก่อนจะไม่ถูกเลือกอัตโนมัติ สำหรับในการกำหนดกลุ่มของออบชั่นบัททอนนั้นจะใช้คอนโทรลเฟรมในการแบ่งกลุ่ม โดยที่แต่ละกลุ่มจะเป็นอิสระไม่เกี่ยวกัน

รูปที่ 3.22 แสดงคอนโทรลออบชั่นบัททอน

เฟรม (Frame)

เฟรม เป็นคอนโทรลที่มีจุดประสงค์ เพื่อใช้ในการจัดกลุ่มคอนโทรล ที่ต้องใช้ร่วมกันเพื่อทำงานอย่างเดียวกันไว้ด้วยกัน เช่น ใช้ในการแบ่งกลุ่มออบชั่นบัททอนออกเป็นพวกๆ ซึ่งโดยทั่วไปแล้วคอนโทรลเฟรมเป็นคอนโทรลที่ช่วยเพิ่มความเรียบร้อย และความสวยงามของโปรแกรม

รูปที่ 3.23 แสดงคอนโทรลเฟรม

ลิสต์บ็อกซ์ (ListBox)

ลิสต์บ็อกส์เป็นคอนโทรล ที่มีลักษณะเป็นทางเลือกเช่นเดียวกันกับออบชั่นบัททอนแต่ว่าจะมีทางเลือกที่ไม่จำกัดเนื่องจากสามารถเพิ่มเติมได้ และ สามารถเลือกได้มากกว่าหนึ่งทางเลือกซึ่งแตกต่างจากออบชั่นบัททอนที่มีทางเลือกตายตัว และมีทางเลือกได้เพียงทางเลือกเดียว

รูปที่ 3.24 แสดงคอนโทรลลิสต์บ็อกซ์

คอมโบบ็อกซ์ (Combo Box)

คอมโบบ็อกส์เป็นคอนโทรลที่ใช้มีการแสดงรายการ ที่ต้องการให้ผู้ใช้เลือกรายการ หรือสามารถแก้ไขรายการที่เลือก คอนโทรลนี้จะใช้พื้นที่ในการวางคอนโทรลน้อยกว่าลิสต์บ็อกซ์ คอนโทรลนี้จะเหมือนกับคอนโทรลเท็กบ็อกซ์ และลิสต์บ็อกซ์ร่วมกัน

รูปที่ 3.25 แสดงคอนโทรลคอมโบบ็อกซ์

ไทเมอร์ (Timer)

จุดประสงค์ของการใช้ไทเมอร์ คือให้มีการทำงานบางอย่างทุกช่วงเวลาที่กำหนดซึ่งจะช่วยให้การทำงานบางอย่างถูกทำแบบฉากหลังพร้อมกับมีโปรแกรมอื่นทำงานอยู่ด้วย ซึ่งคอนโทรลนี้จะไม่สามารถมองเห็นได้เมื่อรันโปรแกรม

รูปที่ 3.26 แสดงคอนโทรลไทเมอร์

ความรู้เกี่ยวกับคุณสมบัติ เมถอด และอีเวนต์

ในการทำงานกับฟอร์ม หรือคอนโทรล จะต้องเกี่ยวข้องกับการกำหนดคุณสมบัติ (Properties) ของคอนโทรลต่างๆ การสั่งงานคอนโทรลตามที่ต้องการผ่านทางเมถอด (Method) รวมทั้งคอนโทรลตอบสนองต่ออีเวนต์

คุณสมบัติ (Properties)

คุณสมบัติ เป็นสิ่งที่ใช้บรรยายลักษณะต่างๆของคอนโทรล เช่นคุณสมบัติ Text ของคอนโทรลเท็กซ์บ็อกส์ จะเป็นคุณสมบัติที่บอกว่า ข้อความในเท็กซ์บ็อกซ์เป็นอย่างไรเป็นต้น คุณสมบัติต่างๆของคอนโทรลจะทำให้คอนโทรลสามารถใช้งานได้หลายรูปแบบมากขึ้น เช่น คอนโทรลคอมโบบ็อกส์ที่เปลี่ยนลักษณะของคอนโทรลได้ถึง 3 แบบด้วยคุณสมบัติ Style เป็นต้น

เมถอด (Method)

เป็นโปรแกรมย่อยประเภทหนึ่ง ซึ่งเป็นสมาชิกของคอนโทรลประเภทนั้นๆ เหมือนกับคุณสมบัติของคอนโทรล เมื่อเรียกใช้เมถอดจะเป็นการสั่งให้คอนโทรลทำงานให้ซึ่งนอกจากจะมีการเปลี่ยนแปลงค่าคุณสมบัติต่างๆของคอนโทรลนั้นๆ ในการเรียกใช้เมถอดจะเหมือนกับการเรียกใช้โปรแกรมย่อยต่างๆ เช่น ต้องมีการส่งค่าพารามิเตอร์ และการส่งค่ากลับคืนมา เป็นต้น

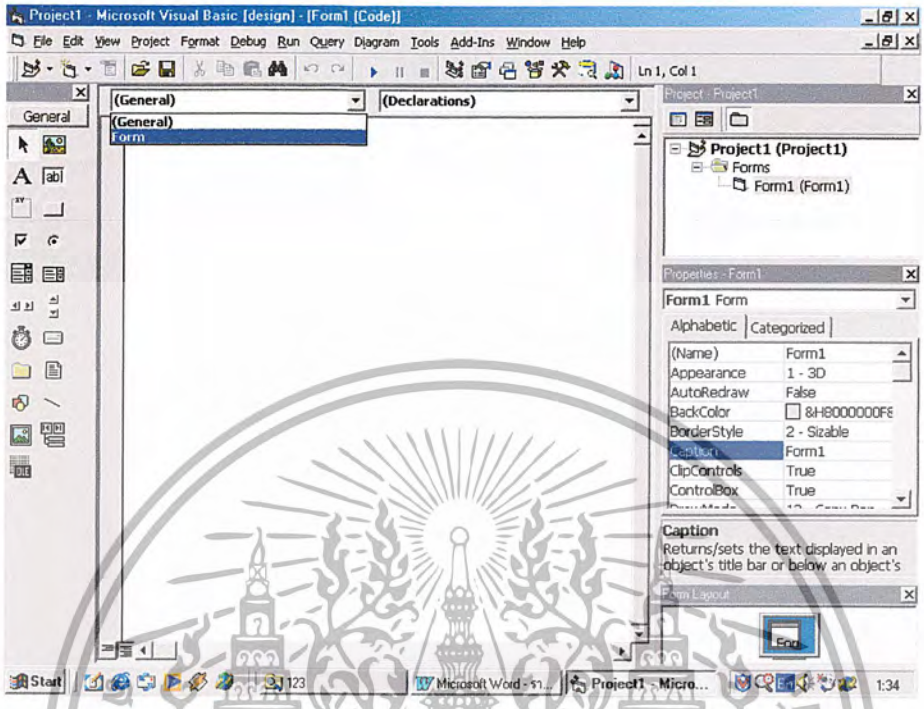
อีเวนต์ (Event)

อีเวนต์ เป็นการตอบสนองต่อเหตุการณ์ภายนอกคอนโทรล เช่น ปุ่มคำสั่งจะมีอีเวนต์Click ที่คอยตอบสนองต่อเหตุการณ์ที่ผู้ใช้งานคลิกเมาส์บนปุ่มคำสั่งนั้น เป็นต้น

ใน Visual Basic 6.0 อีเวนต์จะเป็นโปรแกรมย่อยที่จะทำงานทันทีที่เกิดเหตุการณ์นั้นขึ้นมาบางอีเวนต์อาจจะมีพารามิเตอร์ส่งเข้ามา เพื่อใช้เป็นข้อมูลบางอย่างที่จำเป็นต่อการเขียนคำสั่งตอบสนองต่ออีเวนต์นั้น

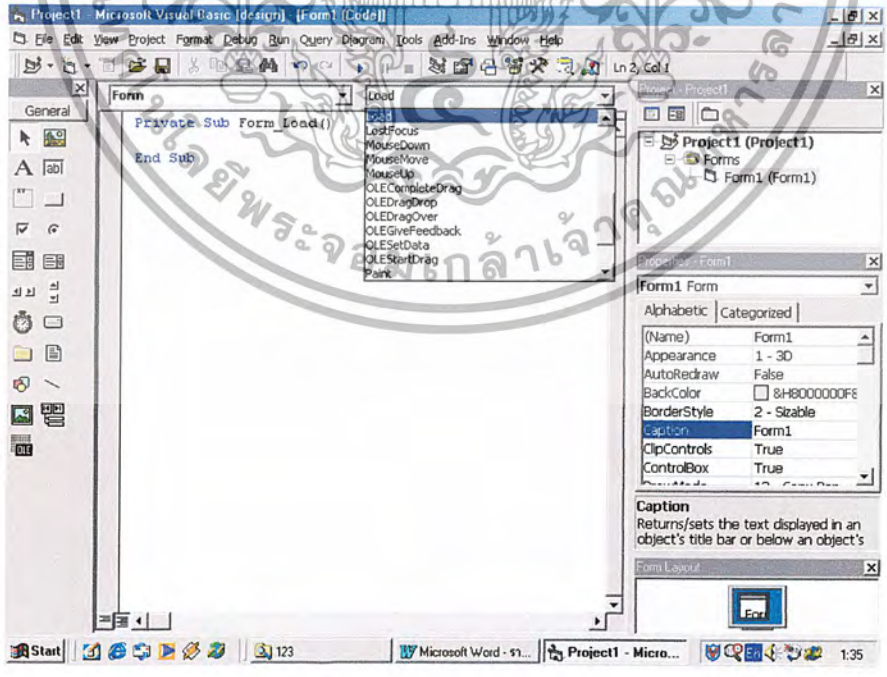
ถ้าต้องการดูคำสั่งอีเวนต์ของคอนโทรล ให้ทำตามขั้นตอนต่อไปนี้

1.เลือกคอนโทรลที่ต้องการจาก Object List Box ของหน้าต่าง Code Editor ดังรูปที่ 3.28



รูปที่ 3.27 แสดงการเลือกคอนโทรล

2.เลือกอีเวนต์ที่ต้องการดูจาก Procedure List Box ในหน้าต่าง Code Editor ดังรูป



รูปที่ 3.28 แสดงการเลือกอีเวนต์

บทที่ 4
การออกแบบและการสร้าง

4.1 การวางแผนการปฏิบัติงานในการสร้างโครงงานมีดังนี้

ทำการวางแผนการทำงานโดยกำหนดเป็นตารางวางแผนการทำงานและปฏิบัติงาน

1. ทำการรวบรวมข้อมูลที่ได้เพื่อเขียนเงื่อนไขการทำงานของลิฟต์ โดยการแสดงอยู่ในรูปของ Flow Chart เพื่อเป็นการกำหนดลำดับการทำงานในการสร้างส่วนแสดงผลโดยการใช้โปรแกรมภาษา Visual Basic

2. ทำการวางขั้นตอนการเขียนเงื่อนไขการสื่อสารของระบบระหว่างคอมพิวเตอร์กับ I/O Module ก็คือการเขียน PROTOCOL

3. ทำการปฏิบัติงานตามที่วางแผนไว้

การเขียนโปรแกรมภาษา Visual Basic เพื่อการควบคุมการเคลื่อนที่และแสดงผลการเคลื่อนที่ของลิฟต์แบ่งเป็นการทำงานออกเป็นส่วนๆ โดยทุกส่วนจะสามารถควบคุมผ่านทางหน้าจอคอมพิวเตอร์และทำงานสอดคล้องกันซึ่งมีการเขียนคำสั่งดังนี้

1. คำสั่งให้ลิฟต์ทำการเคลื่อนที่ ขึ้น – ลง

2. คำสั่งให้ลิฟต์เปิด – ปิด ประตูเมื่อมีการกดปุ่ม เปิด – ปิด ประตูหรือเมื่อลิฟต์ถึงชั้นที่ทำการกดปุ่ม ลิฟต์ก็จะทำการเคลื่อนที่ไปหา

3. คำสั่งให้ลิฟต์ทำการหน่วงเวลาในการ เปิด – ปิดประตูเมื่อลิฟต์ถึงชั้นที่ต้องการแล้ว

4. คำสั่งในการควบคุมให้ประตูต้องปิดสนิทก่อนลิฟต์จะเคลื่อนที่ขึ้น–ลงและต้องหยุดจอดสนิทถึงจะให้เปิดประตู

5. คำสั่งในการควบคุมไฟแสดงสถานการณ์เคลื่อนที่ของลิฟต์แสดง โดยใช้ไฟลูกศรขึ้น-ลง และไฟแสดงชั้นภายในลิฟต์

4.2 การออกแบบทางด้านซอร์ฟแวร์

4.2.1 สำหรับส่วนของการใช้ภาษาทางคอมพิวเตอร์แสดงสภาวะการทำงานของลิฟต์มีดังนี้

1.การออกแบบระบบการแสดงผลเป็นลักษณะโครงสร้างและตัวลิฟต์ในมุมมองต่างโดยใช้ภาษา Microsoft Visual Basic 6.0

2.การออกแบบระบบแสดงผลการทำงานในส่วนต่างๆ และการสั่งงานให้โปรแกรมทำงานผ่านทางคอมพิวเตอร์ เช่น การเปิด-ปิดประตู การขับเคลื่อนให้ลิฟต์ขึ้นลง ปุ่มและไฟแสดงผลต่างๆรวมทั้ง Option พิเศษต่างๆของลิฟต์ เป็นต้น

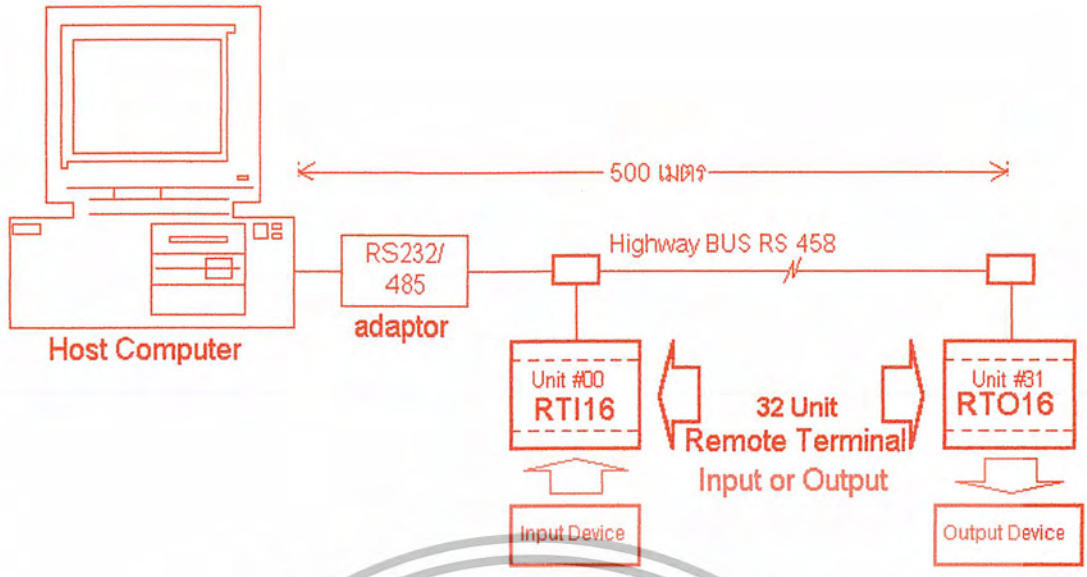
3.การปรับปรุงพัฒนาระบบการแสดงผลการทำงานให้สมบูรณ์โดยการใช้โปรแกรมกราฟิกอื่นๆเข้ามาประยุกต์ใช้

4.2.2 สำหรับส่วนของการใช้งาน I/O Module มีดังนี้

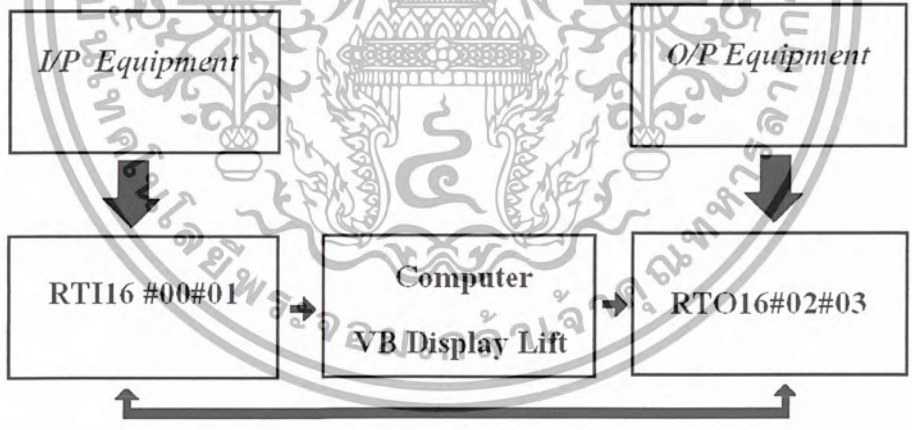
1.I/P Module ใช้ในการรับค่าอินพุทจากชุดอุปกรณ์อินพุทจากภายนอก ในการสร้างโครงงานนี้ใช้แผงสวิตซ์ควบคุมเป็นอุปกรณ์ในการป้อนค่าอินพุทให้กับ I/P Module แล้วกำหนดการแสดงผลให้กับ VB Display Lift โดยผ่านทาง Computer

2.O/P Module ใช้ในการแสดงผลของการเคลื่อนที่ของลิฟต์เพื่อจะนำค่าที่ได้จากโปรแกรมคอมพิวเตอร์ส่งออกไปควบคุมเอาท์พุทภายนอก

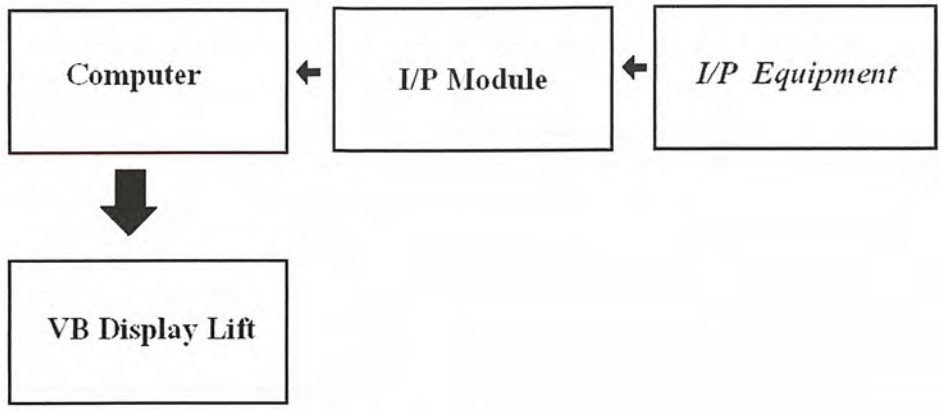
3.I/P Module ทั้งสองส่วนจะต้องทำงานสอดคล้องกันกับโปรแกรม VB Display Lift และทั้งสองส่วนของ Module ซึ่ง I/P Module จะคอยรับข้อมูลที่ส่งเข้ามาให้ VB แสดงผลและส่งข้อมูลไปที่ O/P Module เพื่อส่งออกไปแสดงผลและนำไปควบคุมการทำงานของอุปกรณ์เอาท์พุทภายนอกอีกที



รูปที่ 4.1 การเชื่อมต่อระหว่างคอมพิวเตอร์กับ I/O Module

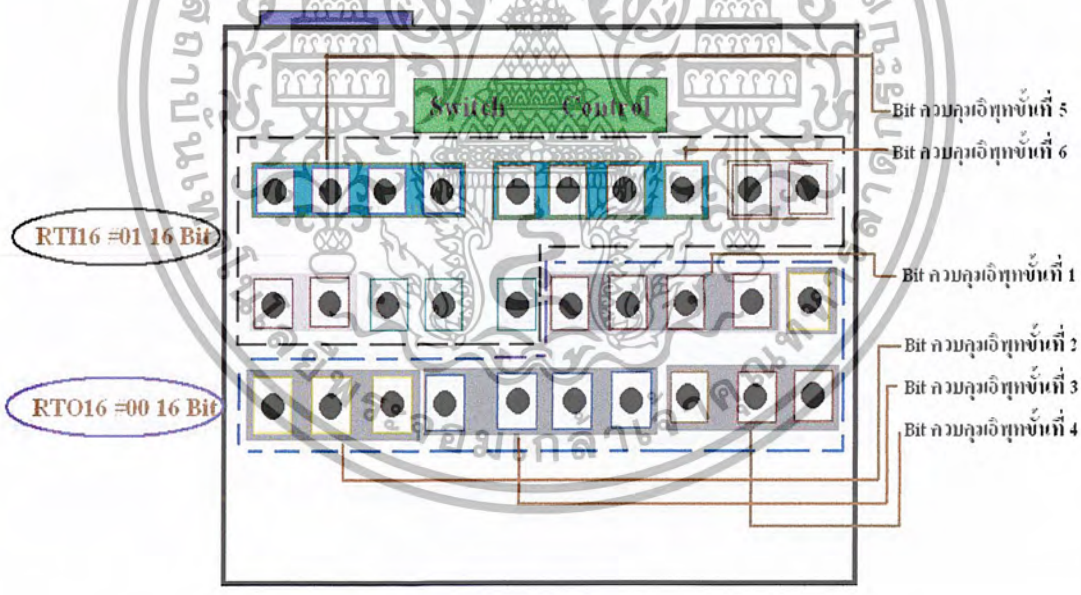


รูปที่ 4.2 การทำงานทั้งระบบของโครงงาน



รูปที่ 4.3 การทำงานของส่วนควบคุมทางด้านอินพุท

4.2.3 การกำหนดอินพุทที่ป้อนให้กับ ชุด INPUT MODULE RTI16



รูปที่ 4.4 รูปแผงสวิตซ์คอนโทรล

การกำหนดส่วนของอินพุตที่จะป้อน เพื่อทดสอบสภาวะการทำงานของลิฟต์ก็เป็นการจำลองการถูกกดเพื่อเลือกใช้ลิฟต์โดยจะกำหนดเป็น 2 ชุดส่วนของ โมดูล 00 เป็นชุดที่ 1 และโมดูล 01 เป็นชุดที่ 2 โดยการกำหนดจะเป็น 4 ดิจิต 16 บิตส่งไปที่อินพุตโมดูล และยังมีการกำหนดการป้อนน้ำหนักเพิ่มเพื่อทดสอบการทำงานของชุด Sensor กับ Safety

ตารางที่ 4.1 การแสดงผลที่ LED RTI16 เป็น Digit 4 Bit 4 ชุด 16 บิต

ชั้น LIFT	1	2	3	4	5	6
RTI16 #01 (ตำแหน่ง LED)	00	05	08,09	14		
RTI16 #00 (ตำแหน่ง LED)					00,02	05,06

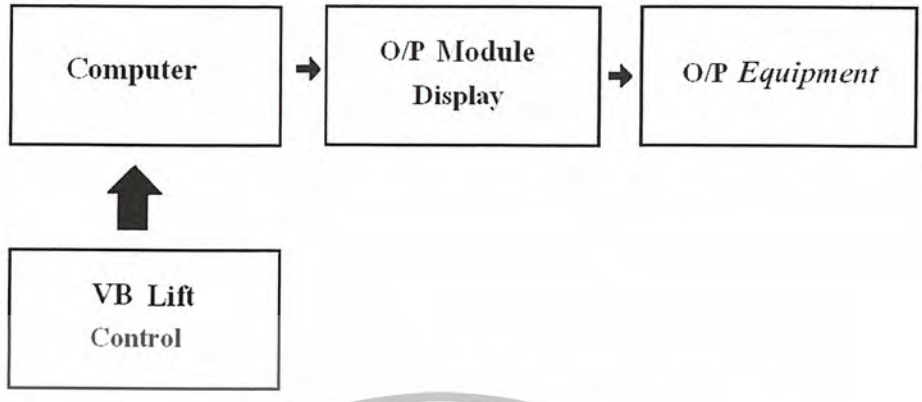
ตารางที่ 4.2 การเขียน Protocol จาก RTI16 เพื่อนำไปควบคุมการแสดงผล

เงื่อนไขการทำงาน	RTI16 Unit	PROTOCOL
อ่านค่าอินพุทที่ป้อนให้ RTI ไปแสดงผล	#00	@00RD*7C
	#01	@00RD*7D

ตารางที่ 4.3 แสดงตำแหน่งและค่าประจำตำแหน่ง LED ของ RTI16 และ RTO16

ตำแหน่ง / ตัวเลข	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0	0												
2					0	1	0	0								
3									1	1	0	0				
4													0	0	1	0
5	1	0	1	0												
6					0	1	1	0								

การทำงานทางด้านเอาท์พุท



รูปที่ 4.5 บล็อกการทำงานของส่วนเอาท์พุท

ตารางที่ 4.4 การเขียน Protocol เพื่อสื่อสารระหว่าง VB Display Lift

เงื่อนไขการทำงาน	RTO16 Unit	PROTOCOL
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 1	#03	@03WR001*6D
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 2	#03	@03WR020*6F
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 3	#03	@03WR300*6F
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 4	#03	@03WR400*68
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 5	#02	@02WR005*68
ให้แสดงผลการเคลื่อนที่ของลิฟต์อยู่ที่ชั้นที่ 6	#02	@02WR060*6B

ตารางที่ 4.5 การแสดงผลที่ LED RTO16 เป็น Digit 4 Bit 4 ชุด 16 บิต

ชั้น LIFT	1	2	3	4	5	6
RTI16 #01 (ตำแหน่ง LED)	00	05	08,09	14		
RTI16 #00 (ตำแหน่ง LED)					00,02	05,06

รูปที่ 4.6 Flowchart การทำงานการเปิด-ปิดประตูลิฟต์

การออกแบบทางด้านซอร์ฟแวร์ หรือการออกแบบโปรแกรมระบบจัดการข้อตกลงในการติดต่อสื่อสารข้อมูล (Protocol) ข้อมูลที่ได้รับเข้ามาทางพอร์ตอนุกรมจะประกอบไปด้วยบล็อกคำสั่งจะประกอบไปด้วย 2 คำสั่งหลัก เช่น คำสั่งการอ่าน (RD) และคำสั่งการเขียน (WR) ข้อมูลที่ส่ง มาจะถูกแปลงจากอนุกรมให้เป็นแบบขนานมาเก็บไว้ยังพื้นที่หน่วยความจำที่เตรียมไว้ (Buffer) สำหรับ บล็อกคำสั่งมีขนาดไม่เกิน 256 ไบท์ ทันทีที่มีการส่งบล็อกคำสั่งเข้า โปรแกรมที่ถูกสร้าง Application บนคอมพิวเตอร์ จะถูกอินเตอร์รัพให้มาตรวจสอบข้อมูลทีละ 1 ไบท์ โดย 3 ไบท์แรกจะเป็น ASCII ที่บ่งบอกถึงหมายเลขประจำเครื่อง ถ้าตรงกันก็จะรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์ต่อไปแต่ถ้าไม่ตรงกันบัฟเฟอร์จะถูกลบออก และรอรับข้อมูลต่อไป ซึ่งถ้าข้อมูลตรงกันตามข้อมูลตรงกันตามข้อตกลงในการติดต่อสื่อสาร โดยจะมีการตรวจสอบแล้ว จากนั้นคอมพิวเตอร์จำทำการถอดรหัสคำสั่ง และประมวลผลตามบล็อกคำสั่ง จากนั้นจะเริ่มส่งบล็อกคำสั่ง จากนั้นจะเริ่มส่งบล็อกตอบสนองกลับไป เมื่อได้รับสัญญาณ Carrier Return (CR) เป็นการบอกให้ทราบว่าจบบล็อกคำสั่ง

4.3 ข้อตกลงในการสื่อสารข้อมูลที่ทำการออกแบบ

ชุดของข้อมูลในการสื่อสารจะถูกเรียกว่าบล็อก บล็อกของข้อมูลจะถูกส่งจากเครื่องคอมพิวเตอร์ ไปในระบบการเชื่อมต่อ ซึ่งจะเรียกว่า บล็อกคำสั่ง (COMMAND BLOCK) และบล็อกของข้อมูลที่ถูกส่งจากระบบ การเชื่อมต่อไปสู่ HOST จะเรียกว่า บล็อกตอบสนอง (RESPONSE BLOCK) ในระบบการเชื่อมต่อสื่อสารแบบหลายจุด แต่ละบล็อก ไม่ว่าจะเป็นบล็อกคำสั่ง หรือ บล็อกตอบสนองก็ตาม จะเริ่มต้นด้วย อักขระ "@" ตามด้วยตำแหน่งเฉพาะ (UNIT NUMBER) ตามด้วยคำสั่ง (HEADER) ข้อมูล (DATA) และอักขระ "*" สิ้นสุดด้วยรหัสกำกับบล็อก (FRAME CHECK SEQUENCE CODE : FCS) และรหัสปิดท้ายบล็อกที่เป็นอักขระ [CR]

รูปที่ 4.7 แสดงรูปแบบของบล็อกตอบสนอง

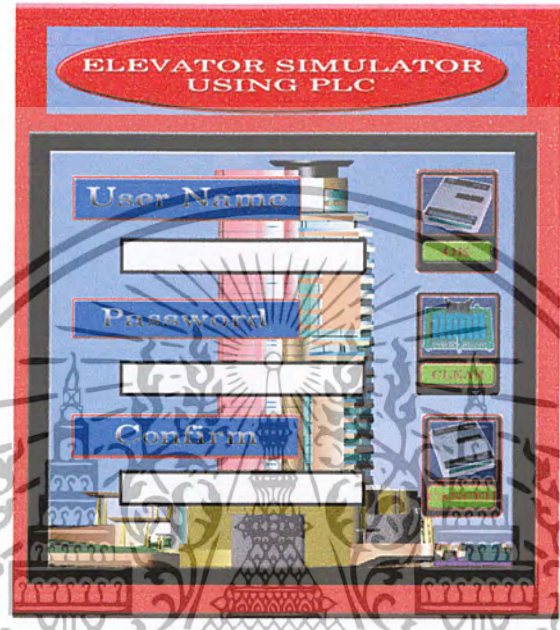
จำนวนอักขระในแต่ละบล็อกทั้งหมดจะต้องไม่เกิน 128 ตัว และช่วงของการคำนวณเพื่อหารหัสกำกับบล็อกจะอยู่ระหว่างอักขระเริ่มต้น (@) ไปจนถึงสิ้นสุด DATA ที่เป็น "*"

4.4 คำสั่งและคำตอบสนอง

ในขณะที่ทำการเชื่อมต่อเพื่อสื่อสารข้อมูล HOST สามารถที่จะทำการเฝ้ามองการดำเนินการและสามารถที่จะทำการควบคุมการทำงานของเครื่องควบคุม ถ้าอยู่ในสถานการณ์เฝ้ามองโฮสคอมพิวเตอร์จะต้องส่งคำสั่งไปถามตามชนิดของข้อมูลที่ต้องการ แต่ละเครื่องควบคุมหรือถ้าในสถานการณ์ควบคุม ก็สามารถที่จะส่งคำสั่งไปทำการเปลี่ยนแปลงค่าของข้อมูลที่อยู่ในหน่วยความจำโดยตรง เช่น ข้อมูลของอินพุท / เอาท์พุท เป็นต้น เวลาของการตอบสนองจะแปรไปขึ้นอยู่กับความเร็วในการส่งผ่านข้อมูล จำนวนของข้อมูล และเวลาในการสแกนของเครื่องควบคุม และ ถ้าเวลาในการสื่อสารมากขึ้นก็เป็นผลให้เวลาในการสแกนมากขึ้นตามไปด้วย ต่อไปจะเป็นคำสั่งต่างๆที่เป็นข้อกำหนด ที่ใช้ในการสื่อสาร

4.5 การใช้งานโปรแกรม

เริ่มต้นเมื่อทำการ Run โปรแกรม จะมีหน้าต่างให้ใส่รหัสก่อนเข้าสู่โปรแกรมซึ่งจะมีการตรวจสอบค่าของ Password ที่มี Permission ต่างกันแล้วแต่ Administrator จะเป็นผู้กำหนดและ Add เก็บไว้ใน Data Base ตามความสามารถของผู้ใช้โปรแกรม ซึ่งการใส่ค่า Password ผิดเกิน 3 ครั้งผู้ที่ใช้จะไม่สามารถเข้าไปใช้โปรแกรมได้ ต้องให้ Administrator ทำการ Add ข้อมูลเข้าให้ก่อน



รูปที่ 4.8 แสดงหน้าจอใส่ Password

รูปที่ 4.9 แสดงหน้าจอเมนูการใช้งานหลัก

รูปที่ 4.10 แสดงหน้าจอ HELP

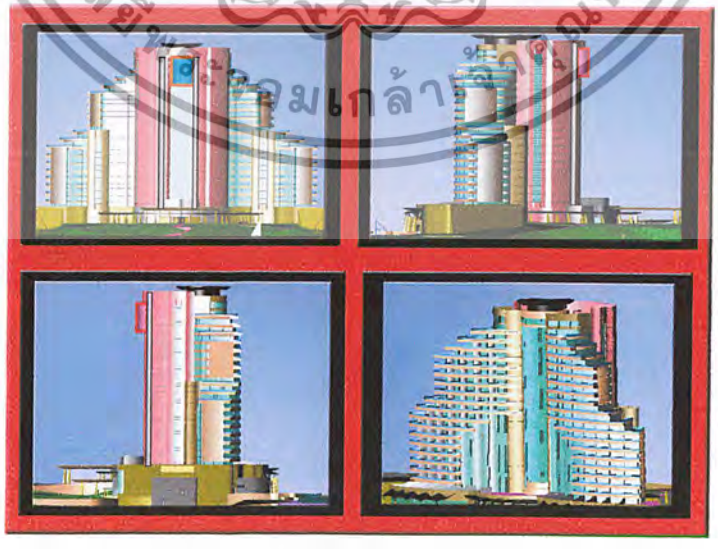
ในหน้าจอส่วนนี้จะแสดงปุ่มการใช้งาน 4 ปุ่มหลักๆ โดยปุ่ม Option จะเป็นการเลือกใช้โปรแกรมพิเศษต่างๆแล้วแต่ที่ผู้จัดทำได้ใส่ไว้เพื่ออำนวยความสะดวกในการใช้โปรแกรม เช่น โปรแกรมแว่นขยายและ Help จะเป็นข้อมูลที่ผู้สร้างได้นำมาศึกษาอ้างอิง

เมื่อต้องการเข้าไปตรวจสอบระบบการทำงานหลักของระบบ ให้ทำการเลือกที่ปุ่ม Manu จากนั้นโปรแกรมจะเข้าไปยังส่วนของหน้าต่างการแสดงสภาวะการทำงานของลิฟต์โดยรวม เป็นหน้าจอในการแสดงสภาวะการทำงานของโปรแกรมและฮาร์ดแวร์ที่ต่อร่วมโดยรวมนั่นเอง ดังรูป

รูปที่ 4.11 แสดงหน้าจอสภาวะการทำงานหลักของโปรแกรม

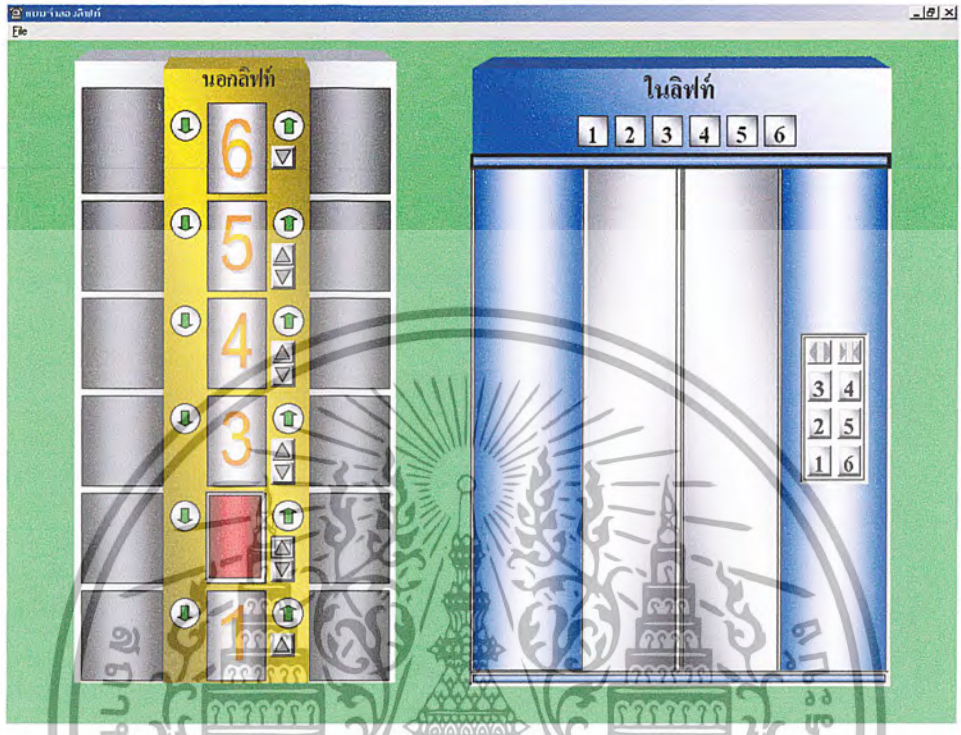
โดยที่เมื่อหน้าจอนี้ปรากฏ เราก็สามารถทราบการทำงานของระบบลิฟต์ที่ใช้งานได้ว่าอยู่ที่ชั้นใดและสามารถควบคุมการทำงาน พร้อมกับตรวจสอบสภาวะผ่านหน้าจอได้ทันทีอีกทั้งยังสามารถเลือกฟังก์ชั่นการทำงานต่างๆผ่านหน้าจอส่วนนี้ได้ทันทีประกอบด้วยปุ่ม Manu เพื่อกลับไปที่เมนูหลัก ปุ่มEdit เพื่อดูสภาวะการทำงานโดยรวม ปุ่มInput Control โดยถ้าเลือกจะแสดงภาพการกำหนดการตั้งค่า Input ที่ป้อนเข้ามาในที่นี้เป็นแผงสวิตซ์คอนโทรล ปุ่ม กดเลือกชั้นต่างๆ พร้อมปุ่ม เปิด-ปิดประตูปุ่มเหล่านี้จะใช้ได้ก็ต่อเมื่อต้องการทดสอบโปรแกรม และเลือกการทำงานเป็นแบบ Manual เท่านั้น ปุ่มDisplay Output เมื่อเลือกจะแสดงภาพการกำหนดการแดงผลที่จะต้องแสดงออกที่ Led ของเอาท์พุตโมดูล และไฟแสดงผลการขึ้นลงและไฟการเปิด-ปิดประตู ส่วนแสดงผล Sensor Weight เมื่อเลือกโหมด Auto จะแสดงสภาวะ On พร้อมแสดงตัวเลขน้ำหนักของผู้โดยสารในลิฟต์พร้อมนำค่าที่ได้ไปเปรียบเทียบกับค่าที่ตั้งไว้ เพื่อควบคุมน้ำหนักและจำนวนผู้โดยสารเมื่อเกินค่าที่ตั้งไว้ก็จะแสดง Aram เตือนทันที ปุ่ม Sensor Speed ก็เช่นกันจะแสดงสภาวะการทำงานและตรวจจับความเร็วพร้อมนำค่าไปเปรียบเทียบกับที่ตั้งค่าไว้ เพื่อควบคุมความเร็วของลิฟต์โดยถ้าค่าเกินก็จะแสดง Aram เตือนทันทีส่วนปุ่น Safety System จะ On ได้ก็ต่อเมื่อเกิด Aram ของการตรวจจับน้ำหนักกับความเร็วขึ้นเท่านั้น ไฟแสดงสภาวะเหล่านี้จะแสดงก็ต่อเมื่อเลือกการทำงานเป็นแบบ Auto เท่านั้นโดยภายในหน้าจอแสดงสภาวะการทำงานหลักนี้จะประกอบด้วยปุ่มในการเลือกใช้งานได้ดังนี้

ปุ่มในการเลือกแสดงผลการทำงานของลิฟต์ประกอบด้วยOutside, Rightside, Left side, back side ปุ่มเหล่านี้เมื่อทำการกดเลือกจะแสดงสภาวะการทำงานของลิฟต์ในมุมมองต่างๆให้ดูซึ่งจะทำให้เราทราบว่าลิฟต์กำลังอยู่ที่ชั้นไหนเคลื่อนที่อยู่หรือไม่ แต่จะเป็นมุมมองในมุมกว้าง ดังรูป



รูปที่ 4.12 แสดงหน้าจอสภาวะการทำงานในมุมมองต่างๆของโปรแกรม

และในส่วนของปุ่ม In side ของลิฟต์นั้นจะเป็นส่วนที่แสดงผลสภาวะการทำงานของลิฟต์ได้อย่างละเอียดและชัดเจนที่สุด โดยแสดงได้ทั้งการแสดงผลการเคลื่อนที่ การเปิด-ปิดประตู ไฟแสดงผลการขึ้นลงการเลือกการทำงานแบบ Manual Auto เพื่อทดสอบการทำงานของลิฟต์และตัวโปรแกรมหน้าจอ Inside แสดงดังรูป



รูปที่ 4.13 แสดงหน้าจอส่วน inside ของลิฟต์

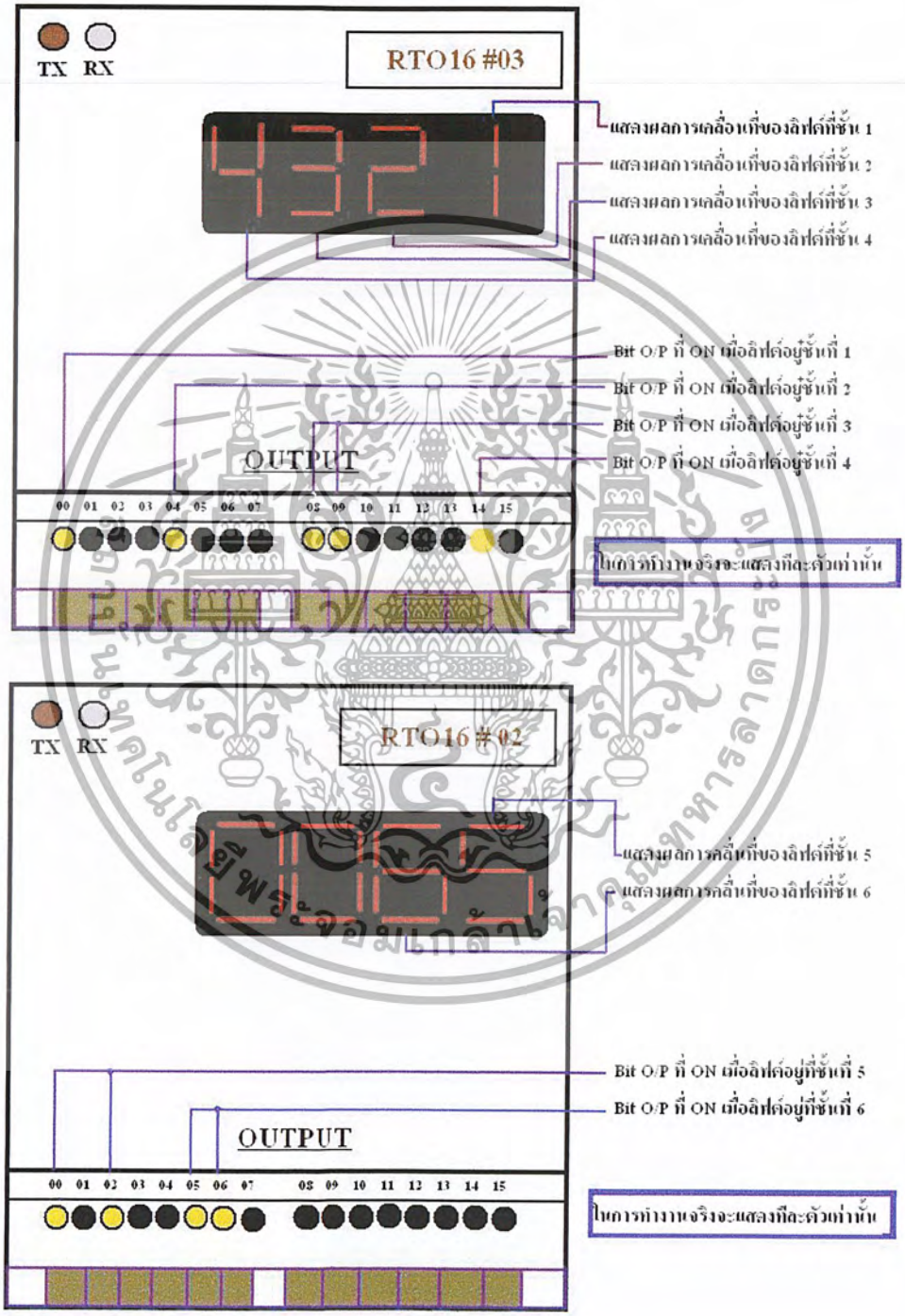
จากรูปโปรแกรมจริงจะมีบล็อกแสดงการรับ-ส่ง โปรโตคอลแสดงออกที่บล็อกแสดงผล

รูปที่ 4.14 การแสดงหน้าจอของ Credits

บทที่ 5

ผลการทดลอง

5.1 การแสดงผลของการทดลอง



รูปที่ 5.1 ส่วนของการแสดงผล

5.1.1 การกำหนดอินพุทที่ป้อนให้กับ ชุด INPUT MODULE RTI16

ในการป้อนค่าอินพุทเพื่อกำหนดการทำงานให้ชุด VB Display Lift เปรียบเสมือนการจำลองขณะที่ลิฟต์ได้รับการกดปุ่มเรียกใช้จากทั้งภายในและภายนอกตัวลิฟต์ แล้วลิฟต์จะเคลื่อนที่ ซึ่งการกำหนดอินพุทให้นั้นเมื่อ RTI16 รับค่าจากแผงสวิตซ์ที่มีการป้อนเป็นลักษณะเลข 4 Digit เข้ามา ชุด RTI16 จะรับค่าและแสดงผลเป็นเลขฐาน 16 โดยมี 7 Segment LED 4 ทำให้สังเกตค่าสภาวะที่เป็นอยู่ได้

โดยการกำหนดอินพุทนั้นใช้ชุดแผงสวิตซ์คอนโทรลจัดป้อนให้กับ RTI16 ทั้ง CH#00 และ CH#01 และในการใช้หารทำการตั้งเป็นแบบ Manual จะใช้แผงสวิตซ์ในการป้อนและไม่สามารถใช้การควบคุมผ่านหน้าจอได้แต่ถ้าใช้แบบ Auto เราจะสามารทดลองควบคุมโปรแกรมการทำงานผ่านหน้าจอคอมพิวเตอร์ได้เลยและเป็นการล็อกไม่ให้การใช้งานภายนอกเข้ามายุ่งเกี่ยวด้วย

5.1.2 ส่วนของภาคเอาท์พุท

เมื่อทำการควบคุมและป้อนค่าอินพุทจากคอมพิวเตอร์ในส่วนของโปรแกรม VB โดยการกดปุ่มลิฟต์ในโหมดของ Auto คำสั่งเงื่อนไขใน VB จะทำการส่งค่าที่เขียนเป็นคำสั่ง PROTOCOL ออกไปทางเอาท์พุทเพื่อสั่งให้ RTO CH#02, CH#03 แสดงผลเป็นตัวเลขชั้นการเคลื่อนที่ของลิฟต์ตามลำดับชั้นต่างๆ โดยอัตโนมัติตามที่เราเรียกสั่งผ่านคอมพิวเตอร์ตามการกำหนดผ่านคอมพิวเตอร์ โดยจะทำการแสดงลำดับชั้นวิ่งไปเรื่อยๆแต่ละลำดับชั้นถ้าลิฟต์อยู่ที่ชั้นไหน ชุด RTO16 ที่กำหนดการทำงานก็แสดงเป็นเลขหลักเดียวในการบอกชั้นที่อยู่ของลิฟต์ โดยชั้นที่ไม่ได้กำหนดจะแสดงค่าเป็นศูนย์ และเมื่อ RTO16 Unit ใดก็ตามแสดงผล อีกUnit จะเซ็ทตัวเองเป็นศูนย์ และจะใช้ตำแหน่งบิตแสดงไม่ซ้ำกันวิ่งบอกตัวเลขขึ้นหรือลงแล้วแต่การเคลื่อนที่ของลิฟต์ จากการสั่งผ่านคอมพิวเตอร์ รวมทั้งแสดงผลการทำงานของลิฟต์ในมุมมองต่างๆทั้งการเคลื่อนที่ขึ้น-ลง การเปิด-ปิดประตู ไฟแสดงสภาวะการทำงาน และตัวเลขการทำงานของระบบเซ็นเซอร์และแสดงสัญญาณเตือนเมื่อเกิดความผิดพลาดซึ่งการทำงานทั้งหมดจะสอดคล้องกันระหว่างคอมพิวเตอร์กับฮาร์ดแวร์

5.1.3 ส่วนของภาคอินพุท

เมื่อทำการป้อนค่าอินพุทจากแผงสวิตซ์ในการเลือกใช้เป็นแบบ Manual ที่ได้กำหนดค่าเป็นชั้นต่างๆของลิฟต์เสมือนว่าเรากดเลือกชั้นที่ปุ่มลิฟต์ ค่าอินพุทที่ได้จะไปแสดงอยู่ที่ส่วนแสดงผลที่เป็น LED และ 7 Segment ของ RTI16 #00, #01 เป็นตัวเลขหลักเดียวตามการเลือกชั้นให้ลิฟต์ทำงานแล้วค่าที่ได้จะถูกอ่านจากเครื่องคอมพิวเตอร์โดยการใช้ PROTOCOL อ่านค่าและนำมาเก็บไว้ และในโปรแกรม VB จะทำการเขียนคำสั่งการทำงานตามเงื่อนไขการทำงานของข้อมูลที่อ่านเข้ามาและเปรียบเทียบแล้วนำไปใช้แสดงผลการเคลื่อนที่ของลิฟต์ผ่านออกทาง Display หน้าจออีกที

โดยทั้งสองส่วนของภาคการทำงานจะทำงานสอดคล้องพร้อมๆกันกับการเคลื่อนที่ของลิฟต์ไปทำให้เราทราบสภาวะการทำงานของลิฟต์ได้โดยไม่ต้องไปดูของจริงอีกทั้งเห็นได้ในหลายมุมมองและหลากหลายสภาวะมากกว่าและสามารถสั่งการควบคุมแก้ไขผ่านทางหน้าจอคอมพิวเตอร์โดยที่ไม่ต้องไปเสี่ยงอันตรายจากของจริงอีกทั้งช่วยให้ทราบสภาวะอุปกรณ์ที่ต่อร่วมว่ามีการทำงานอย่างไร On หรือ Off ได้อย่างมีประสิทธิภาพ

บทที่ 6
สรุปและวิจารณ์

รีโมทเทอร์มินอล อินพุท(Remote Terminal Input :RTI16) และ รีโมทเทอร์มินอล เอาท์พุท (Remote Terminal Output : RTO16) เป็นโมดูลอินพุท เอาท์พุท เพื่อการควบคุมระยะไกล ซึ่งได้พัฒนาขึ้นมาเพื่อแก้ปัญหาให้กับผู้ที่ต้องการควบคุมอุปกรณ์ไฟฟ้า ด้วยเครื่องคอมพิวเตอร์ โดยตัดข้อยุ่งยากทางด้านอินเตอร์เฟส สามารถที่ใช้กับควบคุมอุปกรณ์ไฟฟ้า โดยการเขียนโปรแกรมขึ้นมาสนับสนุน(Support) การทำงานบนคอมพิวเตอร์ ตัวอย่างเช่น โปรแกรมภาษา C , Pascal , Basic ,Visual Basic 6 และ Delphi เป็นต้น การควบคุมผ่านพอร์ตอนุกรมแบบ RS232C หรือ RS485 ภายใต้ข้อตกลงเดียวกันในการสื่อสารข้อมูล (Fac-Talk Protocol) เพราะฉะนั้นการพัฒนาตัวโปรแกรมขึ้นมาควบคุมการทำงานนั้นจะขึ้นอยู่ตัวผู้ใช้ว่ามีประสบการณ์มากน้อยแค่ไหน

นอกจากนี้ผู้ใช้ไม่จำเป็นต้องสร้างวงจรอิเลคทรอนิกส์อื่น ๆ เพิ่มเติมขึ้นอีก เท่ากับเป็นการลดขั้นตอนการทำงานที่ยุ่งยากลง และส่งเสียหายให้กับเครื่องควบคุม หรือ คอมพิวเตอร์อีกด้วยเมื่อเกิดปัญหาลัดวงจรทางอุปกรณ์อินพุทหรืออุปกรณ์เอาท์พุท นอกจากนี้ยังสามารถขยายระบบให้มีจำนวนอินพุท-เอาท์พุทมากขึ้น ในรูปแบบของการกระจายออกไปให้เครือข่ายนับได้ว่าเป็นระบบที่มีความยืดหยุ่นสูง และได้รับความสะดวกมากขึ้น ง่ายต่อการติดตั้ง และบำรุงรักษา ช่วยประหยัดค่าใช้จ่ายในการเดินสายไฟฟ้าควบคุมเมื่อเทียบกับแบบอื่นๆ

โดยการต่อใช้งาน RTI16 และ RTO16 จะสามารถต่อใช้ได้ทั้งแบบ จุด ต่อ จุด(Point to-Point) คือมีคอมพิวเตอร์เป็นตัวประมวลผล 1 ตัว ต่อกับ RTI16 หรือ RTO16 ก็ได้ 1 ตัว และแบบกระจายออกไปหลายจุด(Multi Drop หรือ Multi Point) สื่อสารบน High Way Bus(RS485) ซึ่งสามารถต่อขยายโมดูลอินพุท เอาท์พุท รวมกันทั้งสิ้นได้ 32 โมดูล แต่ละโมดูลสามารถรับหรือส่งข้อมูลได้ 16 Point

ในส่วนของการเขียนโปรแกรมขึ้นมาสนับสนุนการทำงาน ในระบบอาคารอัตโนมัติจำเป็นต้องนำเอาทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับการทำงาน เช่น การจัดการข้อมูลที่สำคัญลงบนฐานข้อมูลโดยใช้โปรแกรมVisual Basic ร่วมกับ Microsoft Access การสร้างกราฟแสดงการเปลี่ยนแปลงค่าพลังงาน แสดงสถานะการใช้ไฟฟ้าในอาคาร การใช้Componentของเสียงเข้ามาจัดการเรื่องสัญญาณ Alarm ทำให้การพัฒนาโปรแกรมขึ้นมาสนับสนุน RTI16 และ RTO16 ในงาน Application มี ประสิทธิภาพสูงขึ้น

แต่ถึงอย่างไรก็ตามการเขียนโปรแกรมควบคุมการรับ-ส่ง ข้อมูลกับ RTI16 และ RTO16 ก็ยังมีปัญหาเรื่อง Real Time เนื่องมาจากอัตราความเร็วในการรับส่งข้อมูล(Baud Rate)และตัวโปรแกรมที่เขียนขึ้นมาเนื่องจากการรับข้อมูลบางครั้งเราจะต้องหน่วงเวลาให้สามารถรับข้อมูลได้

ทันแต่การส่งข้อมูลไม่ต้องหน่วงเวลา(Delay) เมื่อรับ-ส่งข้อมูลได้แล้วเราต้องนำข้อมูลมาตรวจสอบว่าข้อมูลที่รับ-ส่งนั้นต้นทางกับปลายทางเป็นข้อมูลเดียวกันโดยเขียนโปรแกรม BCS ใช้ตรวจสอบข้อมูลให้ถูกต้อง จะตรวจสอบจนกว่าจะส่งและรับให้ถูกต้องเสียก่อนจึงจะทำงานขั้นตอนต่อไป เวลาที่ใช้ในการตรวจสอบข้อมูลจึงผลในเรื่อง Real Time ด้วยเช่นกัน การแก้ปัญหาในเรื่องนี้จึงต้องกำหนดค่าของTimer ที่ใช้หน่วงการรับข้อมูลนั้นมีช่วงเวลาที่ทำงานเหมาะสมกับ Baud Rate ให้มากที่สุดก็จะช่วยลดปัญหาลงได้ และในการ Wiring สายสัญญาณ RS485 นั้นต้องระมัดระวังเพราะเป็นสายสัญญาณไม่ควร Wiring ร่วมกับสายไฟกำลัง เนื่องจากจะทำให้เกิดสัญญาณรบกวนทำให้ข้อมูลที่รับ-ส่ง อาจสูญหายหรือเป็นข้อมูลที่ผิดเพี้ยนไป ควรตรวจสอบจุดต่อแหล่งจ่ายด้วยว่าต่อถูกต้อง เพราะจะทำให้อุปกรณ์ได้รับความเสียหายได้

บรรณานุกรม

นายพรเพชร กิจสิริสินชัย, นายสิทธิชัย ประสิทธิเวชชากูร, นายสุรเชษฎ์ ศิริลาภพานิช, 2543; โปรแกรมสนับสนุนการเขียนโปรแกรมควบคุมPLC, ปริญญานิพนธ์,สถาบันเทคโนโลยี-พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ธาริน สิทธรรมชารี, สุรสิทธิ์ คิวประสพศักดิ์; การเขียนโปรแกรมด้วย Advanced Visual Basic Version 6.0 ฉบับเพื่อการประยุกต์ใช้งาน บริษัท Success Media Co., Ltd

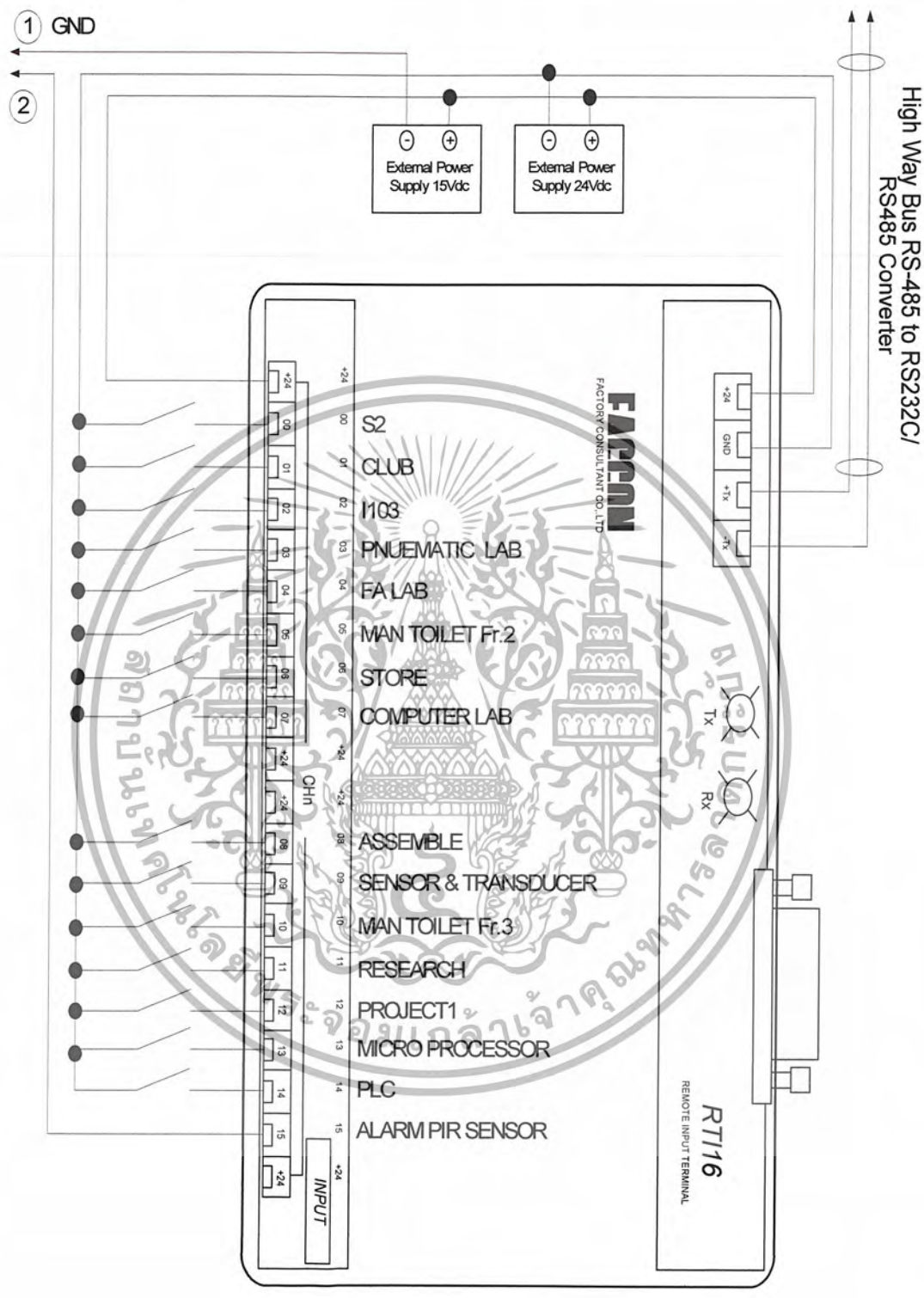
บริษัท แฟคทอรีคอนซัลแตนท์ จำกัด ; คู่มือRTI, RTO Remote terminal Input & Output, รศ. สุพรรรณ กุลพานิชย์

รศ.สุพรรณ กุลพานิชย์ ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ ; การใช้งานพีซีลิงค์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

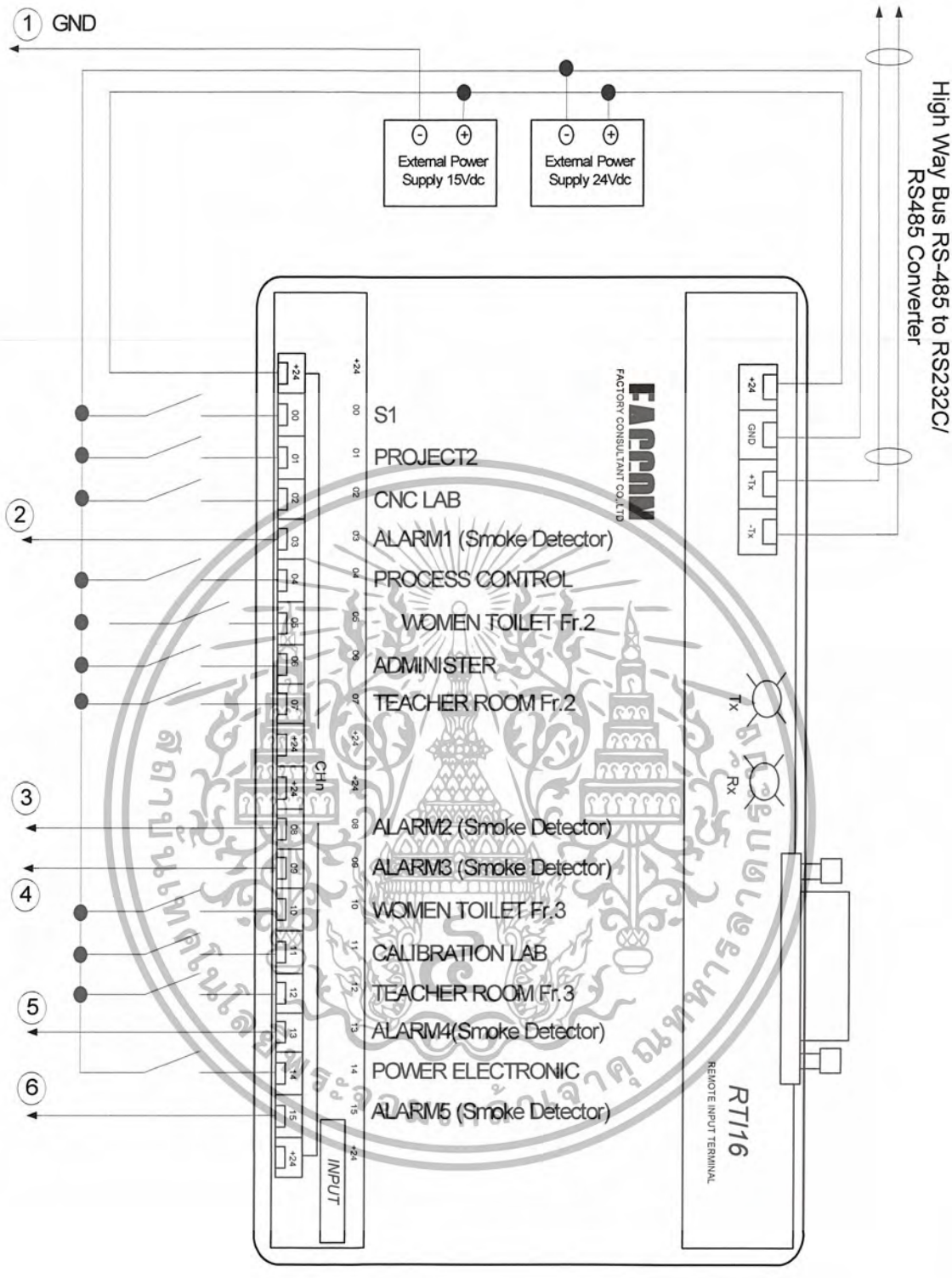
กิตติ ภักดีวัฒนะกุลมจำลอง ครูอุตสาหะ , 2543 ; Visual Basic 6

สัจจะ จรัสรุ่งรวีวร ; คู่มือการเขียนสร้างแอพพิเคชั่นMicrosoft Visual Basic 6 Basic & Advance พิมพ์ครั้งที่3 โดยบริษัท Infro press co.,ltd

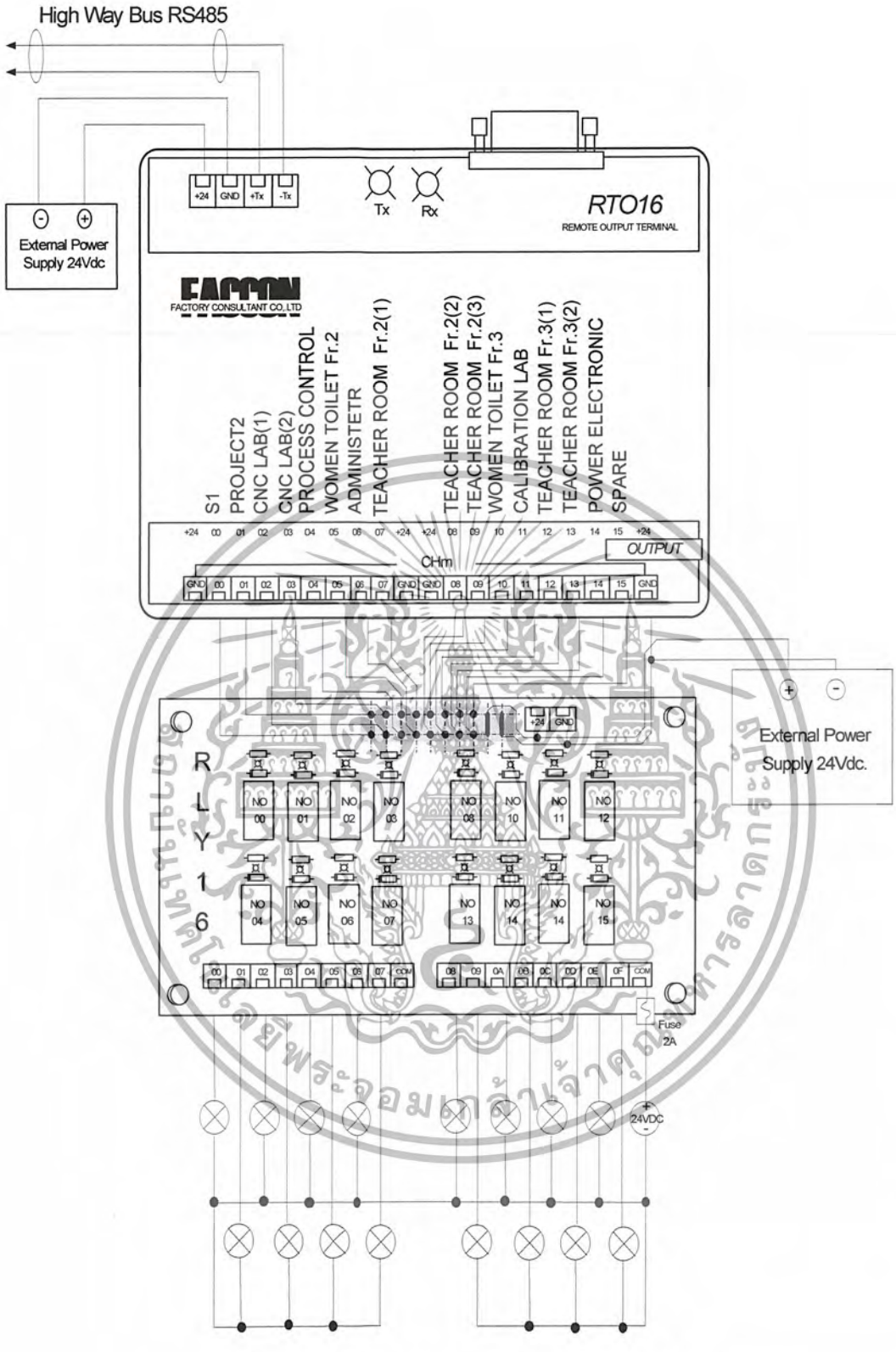
ภาคผนวก



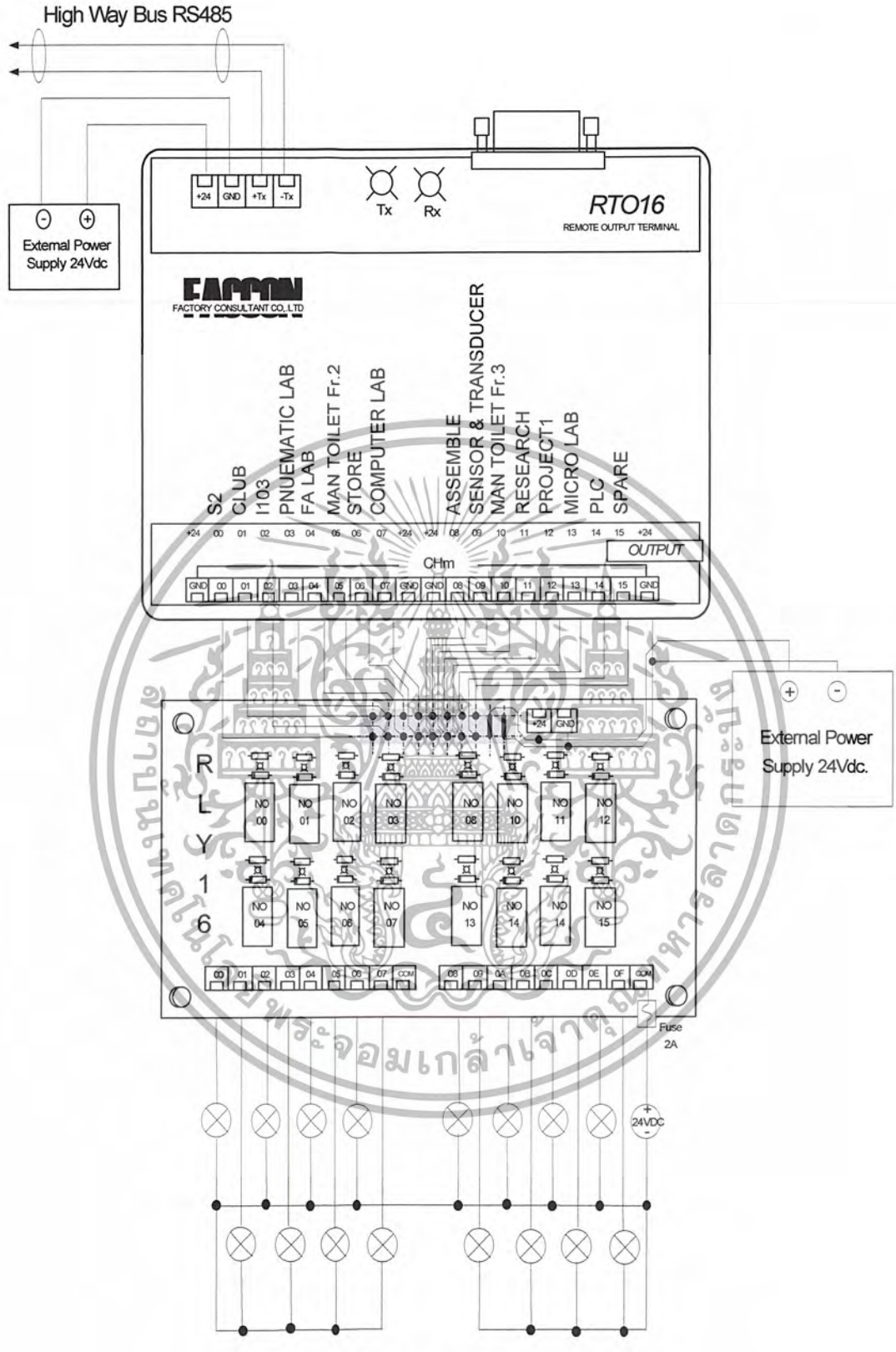
INSTALLATION RTI Unit01
Switch and Sensor



INSTALLATION RTI Unit00
Switch and Sensor



INSTALLATION RTO Unit02
Lighting system



INSTALLATION RTO Unit03
Lighting system