# การประยุกต์การใช้งานโรงงานอัตโนมัติ
# FACTORY AUTOMATION APPLICATION



โดย

| | | |
|---|---|---|
| นายณัฐวุฒิ | สุระเสถียร | 40012088 |
| นางสาวรติรส | ศรีธีระวิโรจน์ | 40012099 |
| นายวิริยะ | สดสิริสวัสดิ์ | 40012101 |
| นายสุทธิพงษ์ | กุลศิริ | 40012109 |


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขา วิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

ปริญญานิพนธ์ ปีการศึกษา 2542

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์การใช้งานโรงงานอัตโนมัติ

( FACTORY AUTOMATION APPLICATION )


ผู้จัดทำ

1. นายณัฐวุฒิ สุระเสถียร 40012088
2. นางสาวรติรส ศรีธีระวิโรจน์ 40012099
3. นายวิริยะ สดสิริสวัสดิ์ 40012101
4. นายสุทธิพงษ์ กุลศิริ 40012109

อาจารย์ที่ปรึกษา

.....

( รศ. สุพรรณ กุลพานิชย์ )

.....

( รศ. วิริยะ กองรัตน์ )

# การประยุกต์ใช้งานโรงงานอัตโนมัติ
# FACTORY AUTOMATION APPLICATION


| | | | |
|---|---|---|---|
| ผู้จัดทำ | นายณัฐวุฒิ | สุระเสถียร | 40012088 |
| | นางสาวรติรส | ศรีธีระวิโรจน์ | 40012099 |
| | นายวิริยะ | สดสิริสวัสดิ์ | 40012101 |
| | นายสุทธิพงษ์ | กุลศิริ | 40012109 |

อาจารย์ที่ปรึกษา รศ. สุพรรณ กุลพานิชย์
รศ. วิริยะ กองรัตน์


## บทคัดย่อ


เนื่องจากโครงสร้างของระบบควบคุมโรงงานหรืออาคารอัตโนมัติจำเป็นที่จะต้องใช้ "ชุดอินเพุท/เอาท์พุท เพื่อการควบคุมระยะไกล" (Remote Terminal Input/output Module) ดังนั้นโครงงานนี้จึงเป็นการพัฒนาชุดเอาท์พุท เพื่อการควบคุมระยะไกล ให้สามารถควบคุมการทำงานของหน่วยแสดงผลแบบตัวอักษร (Character Display Unit) และประยุกต์การใช้งานชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล ในการควบคุมอุปกรณ์ไฟฟ้าภายในโรงงานหรืออาคาร เพื่อให้มีการควบคุมการใช้พลังงานได้อย่างมีประสิทธิภาพ โดยให้ชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกลให้เป็นสื่อกลางในการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์และอุปกรณ์ไฟฟ้าภายในอาคาร ซึ่งใช้การเชื่อมต่อระบบโครงข่ายแบบบัสโดยอาศัยการอินเตอร์เฟสของสัญญาณมาตราฐานRS-485 ในการสื่อสารข้อมูลถึงกันจะอาศัยข้อตกลงการสื่อสารข้อมูลของ "ชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล" (Protocol)

# FACTORY AUTOMATION APPLICATION

**STAFF** Nattawut Surasatian 40012088
Ratiros Sriterawiroaj 40012099
Wiriya Sodsirisawad 40012101
Suttipong Kulsiri 40012109

**ADVISOR** Mr.Suphan Gulpanich
Mr.Wiriya Kongrat

**Abstract**

Due to the structure of the factory or building automation control system need to use the remote terminal input/output module. So that this project is development of the remote terminal output module to control the character display unit. And apply to use the remote terminal input/output module to control electrical input/output device within the building and factory in order to control efficiently power using. Through the remote terminal input/output module control for being media of the connection between computer or electrial input/output device within the building. The connection in each module is the bus topology network. This network use interface's standard sign. The data communication use defined protocal. Refer to the positions (Addressing) on the bus.

# กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงด้วยดี เพราะได้รับความเมตตาจากท่าน รศ. สุพรรณ กุลพาณิชย์ และท่าน รศ. วิริยะ กองรัตน์ ที่ได้ให้คำปรึกษาและแนะนำคณะผู้จัดทำมาตลอด คณะผู้จัดทำรู้สึกทราบซึ้งอย่างยิ่งในความอนุเคราะห์จากท่าน รวมทั้งอาจารย์ทุกๆท่านที่ไม่ได้เอ่ยนามในที่นี้ที่ได้ประสิทธิ์ประสาทวิชาความรู้แก่ศิษย์ในการศึกษาระดับต่างๆผู้จัดทำขอกราบขอบพระคุณอย่างสูง

ขอกราบขอบพระคุณ บิดา มารดา และพี่น้อง ที่ให้โอกาสกับลูกสำหรับการศึกษาเล่าเรียน รวมทั้งให้กำลังใจต่างๆในการศึกษาตลอดมา

ขอขอบคุณ บริษัท แฟคทอรี่ คอนเซ้าท์แตนท์ จำกัด ที่ให้ความสนับสนุนอุปกรณ์และข้อมูลในการทำปริญญานิพนธ์และทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ทุกท่าน ที่เป็นกำลังใจและช่วยเหลือในการทำปริญญานิพนธ์ฉบับนี้มาโดยตลอด

ขออำนาจคุณพระศรีรัตนตรัยและสิ่งศักดิ์สิทธิ์ทั้งหลาย ช่วยดลบันดาลให้ทุกท่านที่กล่าวถึงนั้น ประสบความสำเร็จ มีความสุขความเจริญ ในหน้าที่การงานทุกประการ

คณะผู้จัดทำ

| | |
|---|---|
| นาย ณัฐวุฒิ | สุระเสถียร |
| นางสาว รติรส | ศรีธีระวิโรจน์ |
| นาย วิริยะ | สดสิริสวัสดิ์ |
| นาย สุทธิพงษ์ | กุลศิริ |

# สารบัญ

# สารบัญ (ต่อ)

หน้า

# สารบัญภาพ

# สารบัญภาพ(ต่อ)

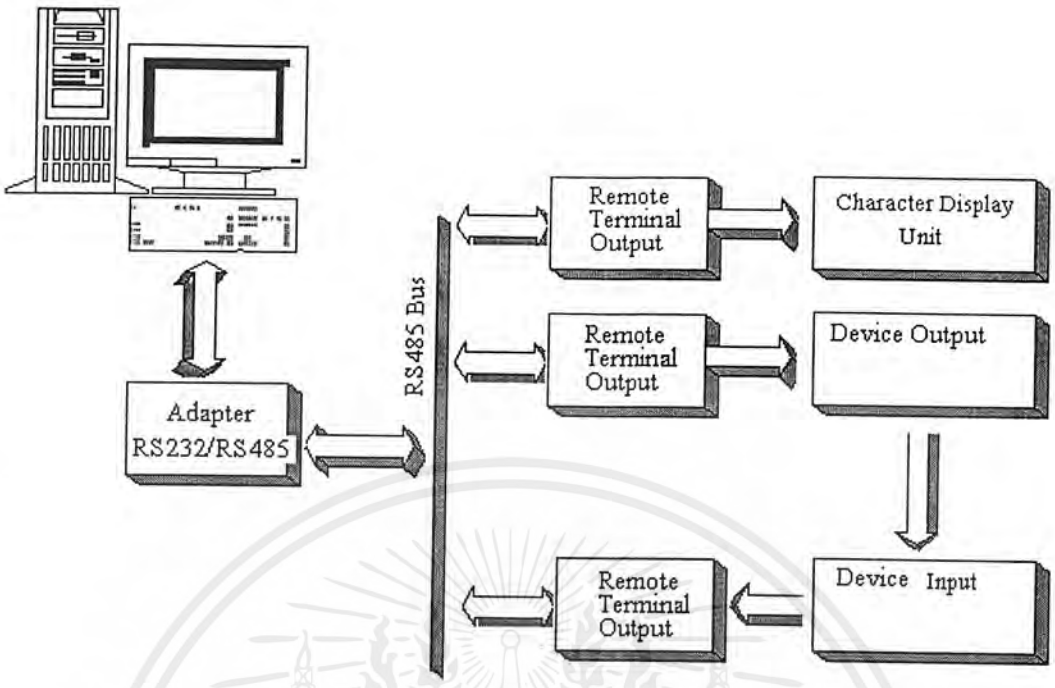
# สารบัญตาราง

# บทที่ 1
# บทนำ

สภาวการณ์ปัจจุบันและแนวโน้มในอนาคต จะเห็นได้ว่ามีอาคาร/โรงงานเกิดขึ้นอย่างมากมายไม่ว่าจะเป็นอาคารสำนักงาน อาคารพานิชย์ โรงแรม โรงงานอุตสาหกรรมปิโตรเคมี หรือแม้กระทั่งอาคารที่อยู่อาศัย ในอาคาร/โรงงานเหล่านี้มักมีความต้องการใช้พลังงานในอัตราสูงมาก ซึ่งหากได้รับการดูแลเอาใจใส่จากเจ้าของอาคาร/โรงงาน หรือผู้ที่เกี่ยวข้องแล้ว บางอาคาร/โรงงานอาจจะมีศักยภาพในการประหยัดพลังงานได้มากถึง 50% ช่วยลดค่าใช้จ่ายทางด้านนี้ได้มากทีเดียว ระบบควบคุมอาคารหรือโรงงานอัตโนมัติเพื่อการจัดการพลังงาน เป็นระบบซึ่งนำเอาเทคโนโลยีคอมพิวเตอร์ เข้ามาใช้ในการควบคุมสภาพการทำงานของอุปกรณ์เครื่องจักร ที่มีการใช้พลังงานในปริมาณสูง โดยคุณสมบัติของคอมพิวเตอร์ดังกล่าวนี้จะมีหน้าที่คอยตรวจสอบและบันทึกข้อมูลการทำงานของอุปกรณ์ที่ใช้งานในอาคาร/โรงงาน ตลอดจนมีการรายงานผลข้อมูลหรือทำการตัดสินใจด้วยตัวเองอย่างรวดเร็ว โดยสั่งการตามโปรแกรมที่กำหนดไว้ล่วงหน้า การนำเอาระบบอาคารอัตโนมัติเข้ามาในด้านการจัดการพลังงานมีวัตถุประสงค์หลักดังต่อไปนี้คือ

1. เพื่อมีการใช้อุปกรณ์เครื่องจักรได้ตามตารางเวลาที่กำหนดอย่างเหมาะสม
2. เพื่อให้มีการควบคุมการใช้พลังงานได้อย่างมีประสิทธิภาพ
3. เพื่อช่วยลดค่าใช้จ่ายด้านพลังงาน

โครงงานนี้จึงเป็นการพัฒนาชุดเอาท์พุท เพื่อการควบคุมระยะไกล ให้สามารถควบคุมการทำงานของหน่วยแสดงแบบตัวอักษร (Character Display Unit) และประยุกต์การใช้งานชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล ในการควบคุมอุปกรณ์ไฟฟ้าภายโรงงานหรืออาคาร เพื่อให้มีการควบคุมการใช้พลังงานได้อย่างมีประสิทธิภาพ ซึ่งโครงงานนี้จะมีโครงสร้างการทำงานดังรูปที่ 1.1 แสดงโครงสร้างการทำงานของโครงการ




รูปที่ 1.1แสดงโครงสร้างการทำงานของโครงการ

โครงงานนี้มีส่วนประกอบที่สำคัญสองส่วน คือ ฮาร์ดแวร์(Hardware) และซอฟต์แวร์(Software)

ฮาร์ดแวร์(Hardware)

ฮาร์ดแวร์ของระบบจะประกอบด้วยอุปกรณ์ต่างๆดังนี้

1. ชุดอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกล(Remote Terminal Unit)
2. แผงแสดงผลแบบตัวอักษร(Character Display Board)
3. ชุดแปลงพอร์ตอนุกรมแบบ RS-232 ให้เป็นแบบ RS-485(Adapter RS-232/RS-485)
4. แบบจำลองอาคารภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม(Device Input/Output)

ซอฟต์แวร์(Software)

โปรแกรมหรือชุดคำสั่งที่เขียนขึ้น เพื่อให้คอมพิวเตอร์สามารถเข้าใจและทำงานได้ ประกอบด้วยส่วนต่างๆดังนี้

1. ชุดของโปรแกรมซึ่งควบคุมการทำงานของหน่วยแสดงผลแบบตัวอักษร
2. ชุดของโปรแกรมประยุกต์เพื่อการใช้งาน

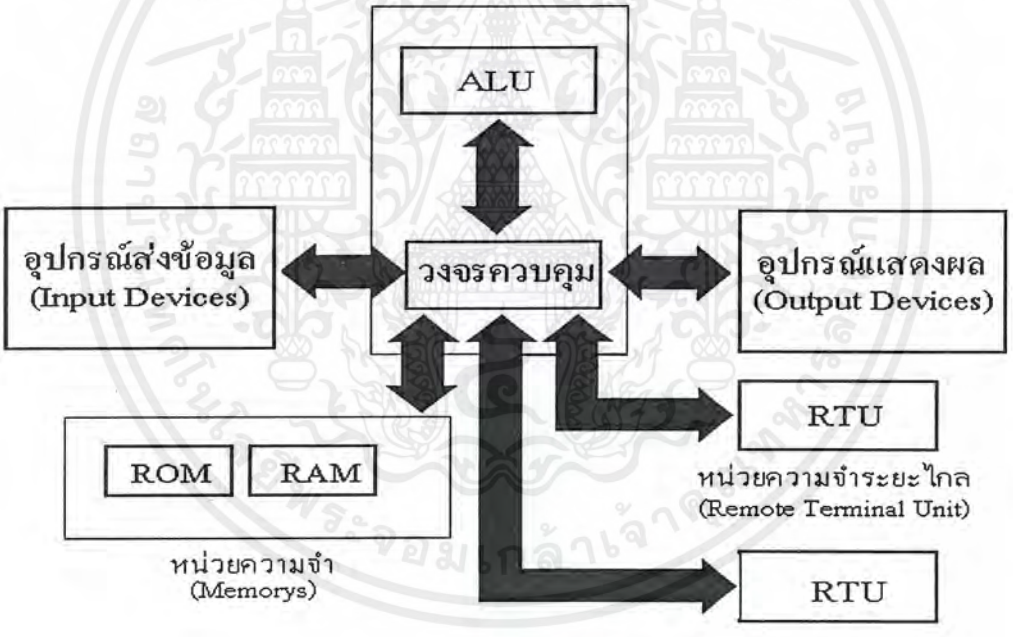
# บทที่ 2
# ทฤษฎีและหลักการ

2.1 โครงสร้างของระบบการจัดการพลังงานโดยใช้ไมโครคอมพิวเตอร์

โดยทั่วไปแล้วโครงสร้างพื้นฐานของระบบการจัดการพลังงาน มีลักษณะเดียวกันกับโครงสร้างของระบบคอมพิวเตอร์ ซึ่งมีส่วนประกอบที่สำคัญสองส่วน คือ ฮาร์ดแวร์(Hardware) และซอฟต์แวร์(Software)

2.1.1 ฮาร์ดแวร์ (Hardware)

ฮาร์ดแวร์ของระบบคอมพิวเตอร์ ประกอบด้วยอุปกรณ์ต่างๆดังแสดงในรูปที่ 2.1 มีรายละเอียดดังนี้




รูปที่ 2.1 โครงสร้างของระบบการจัดการพลังงานโดยใช้ไมโครคอมพิวเตอร์

2.1.1.1 หน่วยประมวลผลกลาง (Central Processing Unit) ทำหน้าที่ควบคุมการทำงานตามที่ได้โปรแกรมไว้ โดยจะใช้ในการรับส่งข้อมูล คำนวณ วิเคราะห์ ตัดสิน และสั่งงานตามโปรแกรมที่ตั้งเอาไว้เป็นส่วนสำคัญที่สุดของระบบ

2.1.1.2 หน่วยความจำ (Memory Unit) ทำหน้าที่เก็บข้อมูลและโปรแกรมต่างๆไว้ดำเนินการ หน่วยความจำนี้หากมีขนาดใหญ่จะทำงานได้มาก แต่มีข้อเสียคือ เมื่อไฟฟ้าดับสิ่งที่มีอยู่ในหน่วยความจำจะสูญหายไป

2.1.1.3 อุปกรณ์รับส่งข้อมูล (Input and Output Devices) ทำหน้าที่ป้อนข้อมูลหรือโปรแกรมให้เครื่องคอมพิวเตอร์และใช้ในการแสดงผลการทำงานของเครื่องคอมพิวเตอร์ ซึ่งประกอบด้วย คอนโซลสำหรับพนักงานควบคุมหมายถึง แป้นพิมพ์ (Keyboard), เครื่องพิมพ์ (Printer และจอภาพ (Monitor)

2.1.1.4 หน่วยความจำระยะไกล (Remote Terminal Unit, RTU) ทำหน้าที่ในการควบคุมอุปกรณ์ต่างๆ ตามคำสั่งของ CPU หน่วยควบคุมระยะไกลแต่ละหน่วยสามารถควบคุมได้หลายจุด โดยทั่วไป RTU จะรับข้อมูลจากตัววัด (Sensor) ต่างๆ เช่น หม้อแปลงกระแส(Current Transformer,CT) เพื่อวัดกระแสไฟฟ้า, เทอร์โมคัปเปิลสำหรับวัดค่าอุณภูมิ, แผ่นออริฟิสสำหรับวัดการไหล, ไดอะแฟรมสำหรับวัดความดัน เป็นต้น จากนั้นจะส่งข้อมูลไปยัง CPU และปฏิบัติงานตามคำสั่งของ CPU ในการควบคุมอุปกรณ์ต่างๆ เช่น รีเลย์ (Relay), คอนแทรกเตอร์ (Contactor), โซลินอยส์วาล์ว (Solinoids Valve) เป็นต้น

โดยทั่วไปใน RTU จะประกอบด้วยแผงควบคุมภายใน (Control Card) ซึ่งมีหน้าที่ต่างๆดังต่อไปนี้

- ใช้วัดค่าปริมาณต่อเนื่องต่างๆ เช่น กระแส, แรงดัน, อุณภูมิ, ความชื้น เป็นต้น แล้วส่งข้อมูลให้ CPU
- ใช้สั่งการอุปกรณ์ควบคุมให้ทำงานตามลักษณะของสัญญาณต่อเนื่อง เช่น ให้วาล์วหรือแดมเปอร์ (Damper)

  ทำการเปิดหรือปิดได้มากน้อยตามต้องการ ตามคำสั่งจาก CPU
- ใช้ตรวจสอบสภาวะการทำงานของหน้าสัมผัส (Contactor) ต่างๆว่าอยู่ในสภาพเปิดหรือปิด แล้วส่งข้อมูลสภาวะการทำงานให้ CPU ทราบ
- ใช้สั่งการอุปกรณ์ควบคุมให้ทำงานในลักษณะเปิดหรือปิด (ON – OFF) เช่น รีเลย์ หรือคอนแทรกเตอร์
- ใช้สะสมข้อมูลที่ต้องการเช่นผลรวมของพลังงานไฟฟ้า,ปริมาณการไหลของน้ำเพื่อนำข้อมูลส่งให้กับ CPU สำหรับคำนวณค่าอื่นๆ ต่อไป

นอกจากนี้ยังมีคุณสมบติบางประการที่ระบบการจัดการพลังงานควรมีเพิ่มเติมดังต่อไปนี้

1. ในกรณีที่ระบบเกิดการขัดข้อง อุปกรณ์หรือโหลดต่างๆ ต้องสามารถทำงานต่อไปได้ตามปกติ (Fail Safe Mode)

2. มีอุปกรณ์แสดงอาการขัดข้อง (Indicator) เพื่อแสดงให้พนักงานควบคุมทราบ ในกรณีที่ RTU ตัวใดตัวหนึ่งขัด

ข้องก็จะยังคงสามารถทำงานตามโปรแกรมที่วางไว้ต่อไปได้

3. มีแบตเตอรี่สำรองภายในระบบ ที่มีความจุเพียงพอในการจ่ายกระแสไฟฟ้าให้อุปกรณ์ในระบบได้ไม่น้อยกว่า 6-8 ชั่วโมง หลังจากไฟฟ้ากระแสสลับดับลง

4. เมื่อไฟฟ้ากระแสสลับดับนานเกินกว่าความสามารถของแบตเตอรี่สำรอง โปรแกรมที่ใช้งานต้องสามารถเก็บไว้ได้อย่างถาวรและสามารถโหลดกลับมายังอุปกรณ์ของหน่วยประมวลผลกลางได้อย่างอัตโนมัติ

ตามที่กล่าวมานี้เป็นโครงสร้างทางด้านฮาร์ดแวร์ ของระบบการจัดการพลังงานโดยใช้ไมโครคอมพิวเตอร์ ระบบดังกล่าวจะสามารถทำงานตามต้องการได้จำเป็นต้องมีโปรแกรมหรือชุดคำสั่งที่เขียนขึ้น (Software) เพื่อให้เครื่องคอมพิวเตอร์ทำงาน

2.1.2 ซอฟแวร์ (Software)

โปรแกรมหรือชุดคำสั่งที่เขียนขึ้น เพื่อให้คอมพิวเตอร์สามารถเข้าใจและทำงานได้ ประกอบด้วยส่วนต่างๆ ดังแสดงในรูปที่ 1.2 รายละเอียดของซอฟต์แวร์มีดังต่อไปนี้คือ


รูปที่ 2.2 โครงสร้างทางด้านซอฟต์แวร์

2.1.2.1 โปรแกรมจัดระบบงาน (Operating System) คือชุดของโปรแกรมซึ่งควบคุมการทำงานของระบบการจัดการพลังงานทั้งหมด เช่น CPU, RTU, อุปกรณ์ Input และ Output รวมทั้งซอฟต์แวร์ประยุกต์ ตลอดจนเป็นตัวเชื่อมโยงการติดต่อระหว่างผู้ใช้คอมพิวเตอร์กับระบบฮาร์ดแวร์และซอฟต์แวร์ด้วย

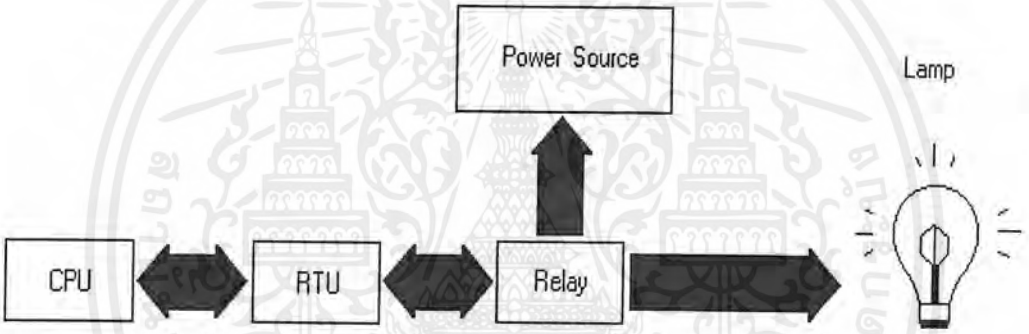
2.1.2.2 โปรแกรมประยุกต์ (Application Program) คือโปรแกรมที่เขียนขึ้นโดยมีวัตถุประสงค์เพื่อการใช้งานเฉพาะอย่างตามที่ต้องการ ซึ่งมีโปรแกรมที่สำคัญต่างๆดังนี้

- โปรแกรมออกคำสั่งหรือหยุดการทำงานของอุปกรณ์ต่างๆ เป็นจังหวะ (Duty Cycling) เช่น คำสั่งให้มีการเดินพัดลมของเครื่องส่งลมเย็น (Fan Coil Unit, FCU) เป็นเวลา 50 นาที และให้หยุดเดินเป็นเวลา 10 นาที ซึ่งค่าที่กำหนดสามารถปรับตั้งได้อัตโนมัติ โดยคำนึงถึงอุณภูมิของอากาศภายในอาคาร, ขีดจำกัดของอุณภูมิเพื่อความสบายของผู้ใช้อาคาร, ช่วงเวลาที่น้อยที่สุดในการหยุดเดินพัดลม ซึ่งจะไม่ให้เกิดการสะสมความร้อนภายในตัวมอเตอร์, จำนวนคนในอาคารและอุณภูมิภายนอกอาคาร
- โปรแกรมควบคุมความต้องการกำลังไฟฟ้า (Demand Control) ด้วยการตัดโหลดของอุปกรณ์ต่างๆ ที่มีความจำเป็นน้อยออกชั่วขณะหรือในช่วงที่มีค่าความต้องการกำลังไฟฟ้าสูงเกินกว่าที่ตั้งไว้
- โปรแกรมออกคำสั่งเริ่มหรือหยุดการทำงานของโหลด ตรงตามเวลาที่ได้รับข้อมูลไว้ในส่วนของหน่วยความจำ ซึ่งสามารถปรับแต่งช่วงเวลาในตารางเวลาดังกล่าวให้สั้นที่สุดอย่างเหมาะสม (Optimum Start/Stop Programming) โดยพิจารณาจากสภาวะแวดล้อมต่างๆ เช่น ยืดระยะเวลาการเปิดเครื่องปรับอากาศ ถ้าอุณภูมิของอากาศภายในและภายนอกอาคารมีค่าต่ำ และปิดเครื่องปรับอากาศเร็วกว่าปกติในกรณีเดียวกัน แต่ยังคงไว้ซึ่งสภาวะสบาย (Comfort Zone) ของผู้ใช้อาคาร
- โปรแกรมออกคำสั่งเปิดและปิดไฟฟ้าแสงสว่างตามตารางเวลาที่กำหนด (Programmable Lighting Control)
- โปรแกรมออกคำสั่งเริ่มหรือหยุดการทำงานของโหลดต่างๆ โดยอัตโนมัติหากเกิดกรณีฉุกเฉิน (Emergency Operation)
- โปรแกรมตรวจสอบขีดจำกัดบนและล่างของค่าต่างๆ เช่น กิโลวัตต์, ความดัน, อุณภูมิ ฯลฯ ของอุปกรณ์เข้าไว้ในหน่วยความจำหากค่าดังกล่าวอยู่นอกขอบเขตจะมีการรายงานเหตุฉุกเฉินไปที่คอนโซลของพนักงานควบคุม
- โปรแกรมปรับแต่งค่าต่างๆ เช่น กิโลวัตต์, อุณภูมิ ฯลฯ เพื่อให้โหลดต่างๆทำงานด้วยเวลาน้อยที่สุดและทำงานด้วยค่าที่เหมาะสม
- โปรแกรมคำนวณค่าต่างๆ เช่น คำนวณค่าประสิทธิภาพของชิลเล่อร์ (Chiller) จะใช้ข้อมูลจากตัววัดอุณภูมิ ความดัน และกิโลวัตต์มิเตอร์ เป็นต้น

2.2 หลักการทำงานของระบบการจัดการพลังงาน

หลักการทำงานของระบบจัดการพลังงานสรุปได้ดังนี้คือเมื่อ CPU ส่งสัญญาณไปยัง RTU ที่ติดตั้งอยู่ในบริเวณต่างๆ แล้ว RTU จะทำการวัดค่าปริมาณต่างๆ จากตัววัดแต่ละชนิดของแผงควบ

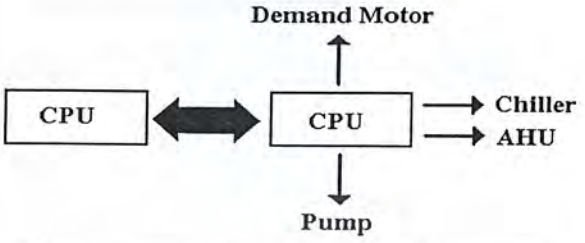
คุมภายในแล้วส่งข้อมูลกลับมาที่ CPU เมื่อ CPU ได้รับข้อมูลตามที่ต้องการก็จะทำการตรวจสอบลักษณะและสภาวะของข้อมูลที่รับมาหลังจากนั้นซอฟต์แวร์ที่มีหน้าที่รับผิดชอบก็เริ่มทำการคำนวณและให้ผลลัพธ์ออกมา โดยจะแสดงผลลัพธ์ในรูปของคำสั่งที่ส่งจาก CPU ไปยัง RTU ให้ทำงานตามโปรแกรมหรือซอฟต์แวร์ที่ได้เขียนไว้ เช่น การเปิด – ปิดไฟฟ้าแสงสว่าง เมื่อ CPU ได้รับคำสั่งให้ทำงานโปรแกรมควบคุมการเปิด – ปิด (Start/Stop Program) ของระบบจะส่งคำสั่งไปยัง RTU ตัวที่ต้องการ ซึ่ง RTU ดังกล่าวจะส่งสัญญาณไปยังแผงควบคุมภายใน เพื่อไปขับรีเลย์ ทำให้หลอดไฟติดและเมื่อรีเลย์ทำงานแล้วจะมีหน้าสัมผัส (Auxiliary Relay Contrac) อีกชุดหนึ่งส่งสัญญาณกลับไปที่ RTU โดยผ่านทางแผงควบคุมภายในเช่นกันและส่งสัญญาณกลับไปยัง CPU เพื่อให้ CPU ทราบว่าได้เปิดไฟตามคำสั่งแล้ว ดังแสดงในรูปที่ 2.3




รูปที่ 2.3 แสดงการทำงานของระบบเปิด – ปิด ไฟฟ้าแสงสว่าง

กรณีต่อไปคือตัวอย่างของการควบคุมค่าความต้องการกำลังไฟฟ้าสูงสุด ซึ่งค่านี้จะคิดจากพลังงานไฟฟ้าสูงสุดในรูปของค่าเฉลี่ย โดยใช้ช่วงเวลา 15 นาที ภายในเดือนนั้นๆเป็นช่วงเวลาในการวัด หน่วยเป็นกิโลวัตต์(kW) เมื่อ RTU อ่านค่าความต้องการกำลังไฟฟ้าจาก Kilowatt Meter และส่งข้อมูลไปยัง CPU แล้วโปรแกรมควบคุมค่าความต้องการกำลังไฟฟ้า (Demand Controler) จะทำหน้าที่ตรวจสอบและคำนวณอัตราเพิ่มของค่าความต้องการกำลังไฟฟ้าเฉลี่ยใน 15 นาทีว่ามีค่าเกินกว่าที่ได้กำหนดไว้แต่แรกหรือไม่ถ้าค่าดังกล่าวต่ำกว่าที่ตั้งไว้ก็ไม่มีอะไรเกิดขึ้น แต่จะเก็บข้อมูลที่ได้ไว้ แต่ถ้า CPU ตรวจสอบแล้วพบว่าอัตราการเพิ่มของค่าความต้องการกำลังไฟฟ้าเฉลี่ยใน 15 นาทีสูงเกินกว่าค่าที่กำหนดไว้ CPU จะส่งสัญญาณควบคุมไปยัง RTU เพื่อควบคุมให้อุปกรณ์ไฟฟ้าที่ถูกกำหนดลำดับความสำคัญที่ต่ำสุด (Lowest Priority) หยุดการทำงานไปทีละชุดตามลำดับ จนกว่าอัตราการเพิ่มของค่าความต้องการกำลังไฟฟ้าสูงสุดจะมีค่าลดลงต่ำกว่าที่กำหนดไว้แล้วอย่าแน่นอน จึงจะเริ่มสั่งอุปกรณ์ต่างๆ ให้กลับเข้าสู่สภาวะการทำงานตามปกติ อุปกรณ์ที่มัก

จะถูกใช้ให้มีการสั่งให้หยุดการทำงานเป็นช่วงดังกล่าวได้แก่ ชิลเลอร์ (Chiller) เครื่องส่งลมเย็น (Air Handling Unit, AHU), มอเตอร์และปั๊มน้ำต่างๆ เป็นต้น การทำงานแสดงดังรูปที่ 2.4



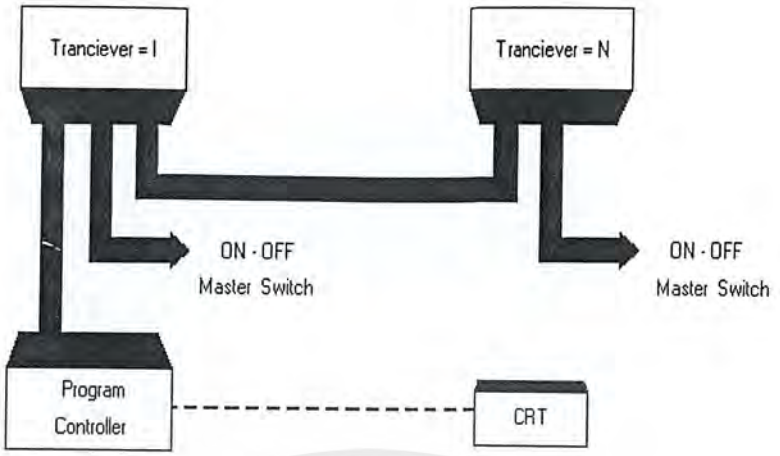

รูปที่ 2.4 แสดงการทำงานของ Demand Controller

2.3 ประเภทของระบบอาคารอัตโนมัติ

ระบบอาคารอัตโนมัติเพื่อการจัดการด้านพลังงานสามารถแบ่งออกได้เป็น 2 ประเภทคือ

2.3.1 ระบบควบคุมแบบการตั้งโปรแกรม (Programmable Control)

เป็นระบบที่ใช้ควบคุมการทำงานของอุปกรณ์เครื่องจักรที่ใช้งานในอาคารให้ได้ตามเวลาที่กำหนดอย่างแน่นอนและเหมาะสม เช่น ในส่วนการทำงานที่แตกต่างกันไปของอาคาร จะควบคุมให้มีการเปิด – ปิดระบบไฟฟ้าแสงสว่าง (Programmable Lighting Control, PLC) การควบคุมการเปิด – ปิดเครื่องปรับอากาศ ตามเวลาการทำงานหรือความต้องการของพื้นที่ อาจนำไปใช้ควบคุมโหลดอื่นๆได้เช่นกัน การทำงานของระบบจะใช้คอมพิวเตอร์และซอฟต์แวร์ที่เขียนขึ้นในรูปของการกำหนดตารางเวลา ไปสั่งการให้เซนเซอร์ทำการเปิดและปิดแมกเนติกส์อุปกรณ์เครื่องจักรเป็นจุดๆไปซึ่งมีข้อเสียคือทำให้ไม่เกิดความยืดหยุ่น (Flexbility) และความคล่องตัวในการควบคุมมากนักอย่างไรก็ตามการใช้ระบบดังกล่าวช่วยให้บรรลุวัตถุประสงค์ของการจัดการด้านพลังงานได้โดยการให้คอมพิวเตอร์ตรวจสอบและควบคุมให้ลดค่าใช้จ่ายด้านพลังงานไฟฟ้าลงได้ จากการกำหนดเวลาการใช้งานอุปกรณ์เครื่องจักรอย่างแน่นอน ไม่เกิดความพลั้งเผลอในการเปิดอุปกรณ์ในระบบไฟฟ้าทิ้งไว้โดยเปล่าประโยชน์ โครงสร้างของระบบควบคุมอัตโนมัติโดยการตั้งโปรแกรม แสดงดังรูปที่ 2.5




รูปที่ 2.5 แสดงโครงสร้างระบบควบคุมอัตโนมัติโดยการตั้งโปรแกรม

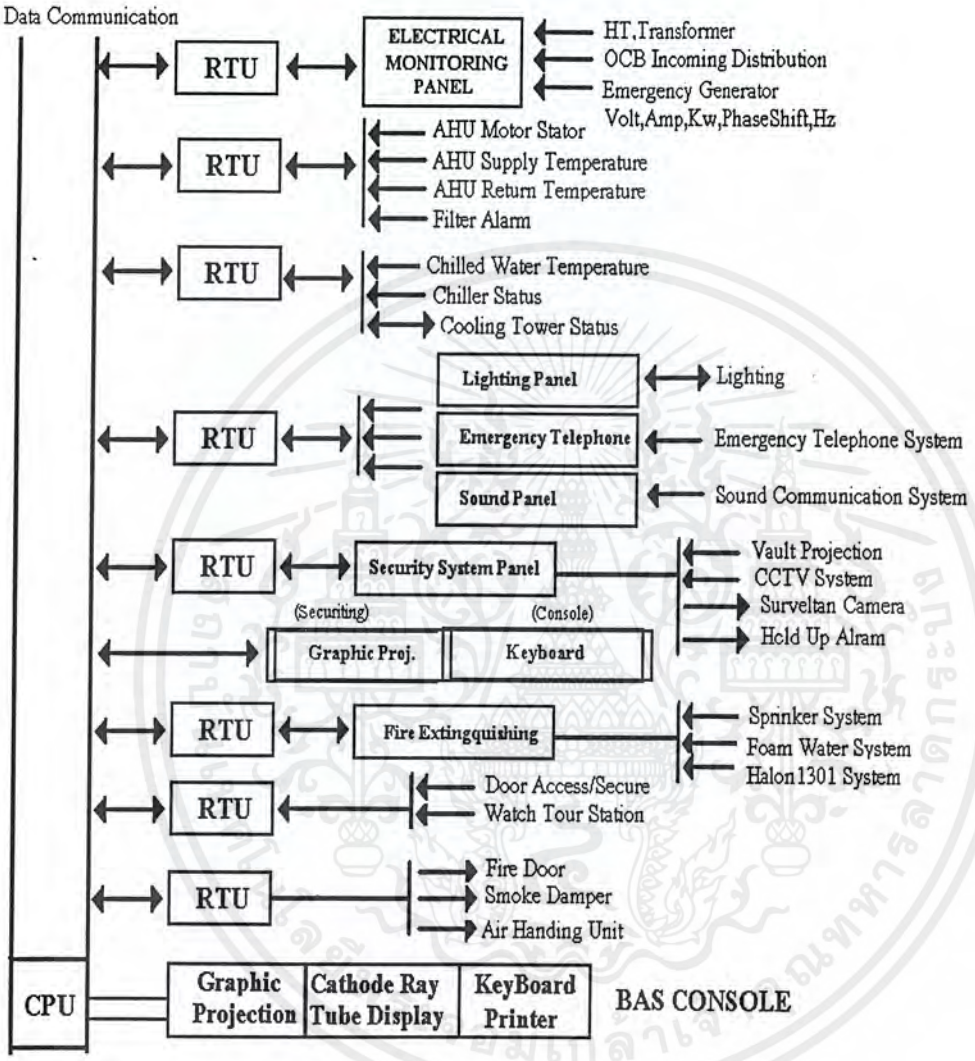
2.3.2 ระบบคอมพิวเตอร์ควบคุมอัตโนมัติสำหรับอาคาร

(Computer Based Building Automation System,BAS)

เป็นระบบที่ใช้ในการควบคุมการทำงานของอุปกรณ์เครื่องจักรที่ใช้งานในอาคารได้อย่างมีประสิทธิภาพมากขึ้น นอกเหนือจากการควบคุมการเปิดปิดอุปกรณ์เครื่องจักรให้ได้ตามเวลาที่กำหนดแล้วยังสามารถนำไปใช้ในระบบอำนวยความสะดวกอื่นๆ และระบบการรักษาความปลอดภัย เช่น ระบบสัญญาณแจ้งเหตุเกิดเพลิงไหม้, ระบบระบายอากาศ, ระบบสุขาภิบาล เป็นต้น เนื่องมาจากการใช้ระบบคอมพิวเตอร์และซอฟต์แวร์ที่มีขีดความสามารถสูงขึ้น ตลอดจนมีความยืดหยุ่น (Flexible) สูง ซึ่งการนำระบบ Artificial Inteligence (AI) เข้ามาร่วมใช้งานกับระบบทำให้มีการจัดการทางด้านพลังงานในอาคารอย่างคล่องตัว เนื่องมาจากคอมพิวเตอร์สามารถทำการตรวจสอบและตัดสินใจได้ในทันทีที่มีสิ่งผิดปกติเกิดขึ้นในอาคาร การใช้ระบบคอมพิวเตอร์ควบคุมอัตโนมัติสำหรับอาคารช่วยให้บรรลุวัตถุประสงค์ของการจัดการด้านพลังงานได้สูงสุด โครงสร้างของระบบควบคุมอัตโนมัติโดยการตั้งโปรแกรม แสดงดังรูปที่ 2.6

ระบบอาคารอัตโนมัติเพื่อการจัดการพลังงานนี้ตั้งไว้เพื่อควบคุมการใช้พลังงาน โดยทำหน้าที่จัดการและดำเนินการให้มีการใช้พลังงานอย่างมีประสิทธิภาพสูงสุด นอกจากนี้ระบบการจัดการพลังงานโดยใช้ไมโครคอมพิวเตอร์ยังสามารถนำไปประยุกต์ใช้งานในการควบคุมระบบอื่นๆ ได้ด้วยเช่นกันได้แก่ระบบปรับอากาศ, ระบบระบายอากาศ, ระบบสุขาภิบาล, ระบบสัญญาณแจ้งเตือนเพลิงไหม้, ระบบดับเพลิงและระบบควบคุมรักษาความปลอดภัยในอาคาร ประโยชน์ที่จะได้รับนอกเหนือไปจากการลดค่าใช้จ่ายในการใช้พลังงาน ซึ่งมีผลทางด้านลดต้นทุนต่างๆแล้ว ระบบยังสามารถเตือนข้อผิดพลาดในการทำงานของอุปกรณ์ต่างๆ ตลอดจนมีการบันทึกข้อมูลทางด้าน

การใช้งาน และประสิทธิภาพของอุปกรณ์ อันเป็นประโยชน์ทางด้านการซ่อมแซมบำรุงรักษา ตลอดจนการพิจารณาจัดหาและติดตั้งอุปกรณ์ใหม่เพิ่มเติมในระบบได้อีกเช่นกัน




รูปที่ 2.6 โครงสร้างระบบ BAS

2.4 ความรู้เบื้องต้นเกี่ยวกับพอร์ตอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงภายนอกหรือคอมพิวเตอร์ด้วยกัน มีด้วยกัน 2 รูปแบบ คือรับส่งข้อมูลแบบขนานและรับส่งข้อมูลแบบอนุกรม

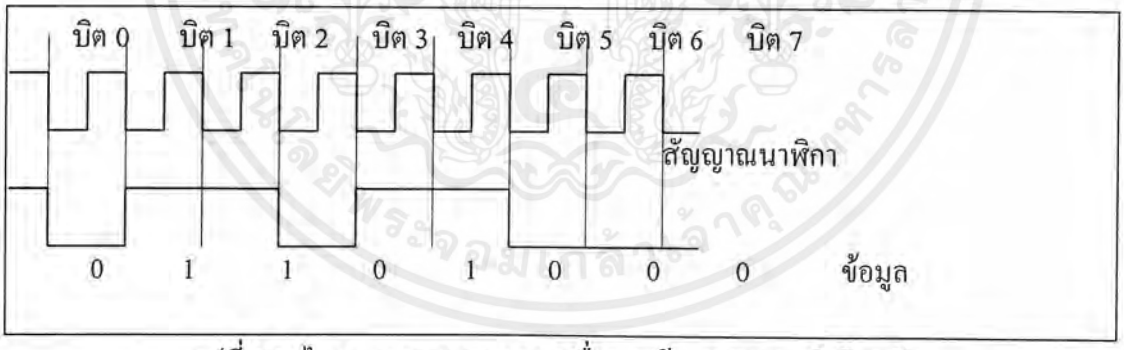
การรับส่งข้อมูลแบบขนานเป็นการรับหรือส่งข้อมูลคราวละ 4 หรือ 8 บิตในเวลาเดียวกัน ทำให้การรับและส่งข้อมูลมีความเร็วสูง กว่าจำนวนของสายที่ใช้ในการถ่ายทอดข้อมูลต้องมีมากเท่ากับจำนวนบิตของข้อมูลที่ทำการถ่ายทอดด้วย นอกจากนั้นยังมีสายที่ใช้สำหรับควบคุม

และตรวจสอบการรับส่งข้อมูลด้วย ซึ่งอาจต้องใช้สายมากเป็น 2 เท่าของจำนวนบิตข้อมูลก็ได้ ส่งผลให้ราคาของสายที่ใช้ในการเชื่อมต่อแบบขนานมักจะมีราคาแพง อีกข้อจำกัดหนึ่งของการถ่ายทอดข้อมูลแบบขนาน คือ ระยะทางในการถ่ายทอดข้อมูลโดยปกติจะอยู่ที่ประมาณ 10 –15 ฟุต

ในขณะที่การรับส่งข้อมูลแบบอนุกรมจะเป็นการรับส่งข้อมูลครั้งละ 1 บิต โดยมีรูปแบบการรับส่งที่เป็นมาตรฐาน ต้องมีการตรวจสอบความพร้อมในการรับและส่งข้อมูลของตัวส่งและตัวรับ การรับส่งข้อมูลแบบอนุกรมมีข้อดีในเรื่องของจำนวนสายสัญญาณที่น้อยมากและไม่แปรผันตามจำนวนบิตของข้อมูล ระยะทางในการรับส่งข้อมูลสูงกว่าแบบขนานมาก โดยปกติถ้าเป็นพอร์ตของอนุกรม RS-232C จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ

2.4.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมแบ่งได้เป็น 2 แบบ คือ การสื่อสารอนุกรมแบบเชิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วยตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือ คีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา ข้อมูล และกราวด์ รูปที่ 2.7 แสดงให้เห็นถึงไดอะแกรมเวลาของการสื่อสารข้อมูลแบบซิงโครนัส




รูปที่ 2.7 ไดอะแกรมเวลาของการสื่อสารข้อมูลแบบอะซิงโครนัส

2.4.1.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัส คือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (baud rate) มีหน่วยเป็นบิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัส ประกอบด้วย 4 ส่วน คือ

1.บิตเริ่มต้น (Start bit) มีขนาด 1 บิต

2.บิตข้อมูลแบบอนุกรม มีขนาด 5 , 6 , 7 หรือ 8 บิต

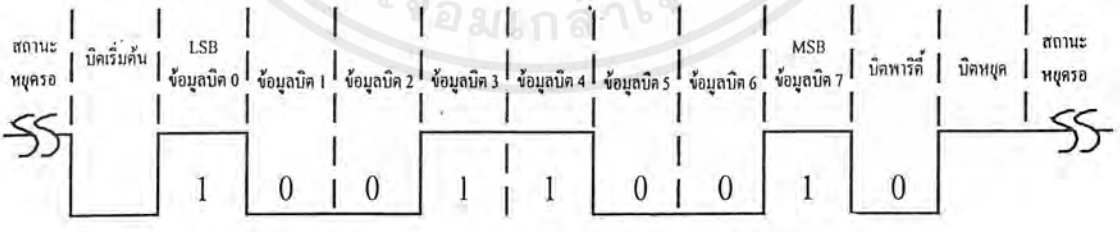
3.บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิต หรือไม่มี

4.บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1, 1:5 หรือ 2 บิต

รูปที่ 2.8 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูลขา DATA จะมีสถานะ ลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการใช้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่าบิตเริ่มต้น (Start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด หรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งอาจมีจำนวน 5, 6, 7, หรือ 8 บิตก็ได้ จากนั้นตามด้วยบิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูลบิตสุดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะ ลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต 1.5 บิต หรือ 2 บิตเพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอด หรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่าได้แก่ 110, 150 , 300 ,600, 1,200 , 2,400, 4,800, 9,600 และ 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโนยีของคอมพิวเตอร์ เนื่องจากบอดเรตคือค่าของจำนวนบิตที่สามรถส่งได้ใน 1 วินาที สมมุติว่า ข้อมูลอนุกรมมีขนาด 8 บิตไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น1 บิต และบิตปิดท้ายอีก 1 บิต ความยาวของข้อมูล 1 ไบต์จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่า 9,600 บิตต่อวินาที ก็สามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (Odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามี




รูปที่ 2.8 รูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส

จำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่างข้อมูลที่จะทำการส่ง ยกตัวอย่างข้อมูลที่ต้องการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99 H 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ URAT(Universul Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม ซึ่งจะกล่าวถึงในรายละเอียดภายหลัง) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวบการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับและส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ไอซี UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ ไอซีเบอร์ 8250 UART เหล่านี้มีระดับแรงดันของลอจิกเป็นแบบทีทีแอล (+5V)แต่เพื่อให้แรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงไปเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก "0" จะมีระดับแรงดัน –3V ถึง -12V และลอจิก "1" มีระดับแรงดัน +3V จนถึง +12V

2.4.2 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้ส่งผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชกุดซึ่งอยู่ห่างไกลกัน โดยสมาคมอุตสาหกรรมอิเล็คทรอนิคส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุตมีระดับสัญญาณตั้งแต่ -3V จนถึง -12V แสดงว่ามีข้อมูล (mark) และ +3V ถึง +12V แสดงเป็นช่องว่าง (Space)

มาตรฐาน RS-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DCE) จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE ทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น

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

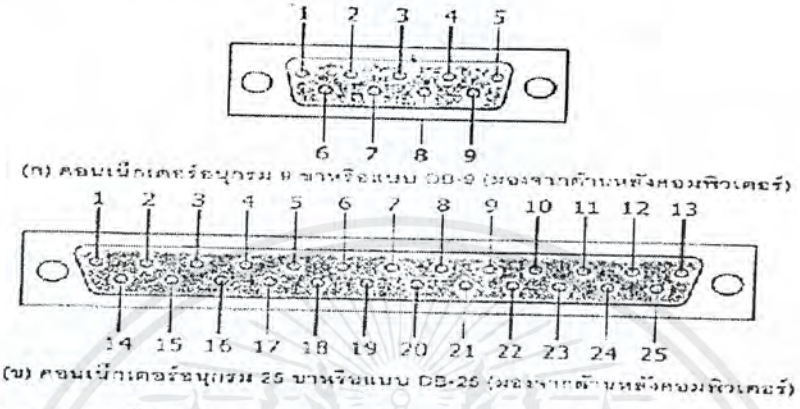
สำหรับการใช้งานในคอมพิวเตอร์ พอร์ตอนุกรม RS-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม เมาส์ และเครื่องพิมพ์ที่สามารถติดต่อกันทางพอร์ตอนุกรมได้

2.4.2.1 คอนเน็กเตอร์สำหรับพอร์ต RS – 232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS – 232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่น ๆ ที่เคยมีการใช้งานมาในอดีตไม่ค่อยมีความสำคัญมากนักจึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2.9

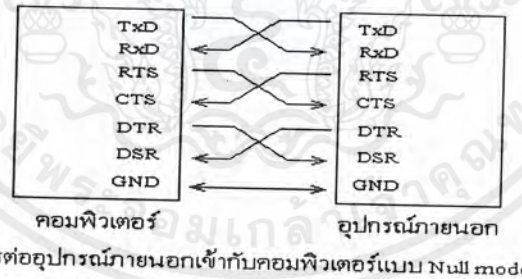
- ขา DATA Carrier Detect : DCD หรืออาจจะเรียกว่า Carrier Detect : CD ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่นโมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ถูกใช้งานมากนัก
- ขา Receive Data หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์โดยจะนำข้อมูลที่อ่านได้ไปเก็บไว้ในรีจีสเตอร์บัฟเฟอร์
- ขา Transmited Data :TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลอนุกรมจากคอมพิวเตอร์โดยการนำข้มูลที่เก็บไว้ในบัฟเฟอร์ส่งออกไป
- ขา Data Terminul Ready : DTR เป็นขาเอาต์พุตที่ใช้สำหรับส่งสัญญาณออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อกับอุปกรณ์ปลายทางโดยขา DTR นี่จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์และถ้าใช้การเชื่อมต่อแบบ 3 สวายต้องเชื่อมต่อขา DTR และ DSR ของพอร์ตอนุกรมเข้าด้วยกันและจะต้องต่อเชื่อมเข้ากับขา DCD ด้วยกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์
- ขา Segnal Ground : GND เป็นขากราวด์ของสัญญาณ
- ขา Data Set Ready : DSR ขานี้จะใช้ควบคู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอก
- ขา Request To Send : RTS เป็นขาเอาต์พุตสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลให้คอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ซึ่งในกรณีนี้มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกัน เพื่อให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- ขา Clear To Send : CTS เป็นขาอินพุตทำหน้าที่รอรับสัญญาณที่างเข้ามาเมื่อมีการส่งสัญญาณก็จะมาที่ขานี้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ขานี้จะใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง

- ขา Ring Indicator :RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยที่ในสายนี้จะไม่ถูกใช้งานจะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้วยังมีความต้องการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

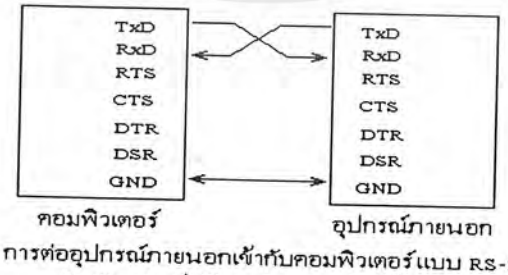



รูปที่ 2.9 แสดงการจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่าง ๆ และหน้าที่การทำงาน

สำหรับการเชื่อมต่อสายระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 4ลูกศรในรูปแสดงถึงทิศทางของข้อมูล การเชื่อมต่อระหว่างรูปที่ 2.10 (ก) การเชื่อมต่อแบบ NULL MODEM หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม ส่วนการเชื่อมต่อในรูปที่ 2.10 (ข) เป็นการเชื่อมต่อโดยใช้สัญญาณน้อยที่สุดเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์




ก ) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ Null modem




ข ) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232
ในลักษณะที่ใช้สายสัญญาณน้อยที่สุด

รูปที่ 2.10 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในรูปแบบต่างๆ

2.5 สถาปัตยกรรมของไมโครคอลโทรลเลอร์แบบชิพเดี่ยวตระกูล 8051

ไมโครคอลโทรเลอร์แบบชิพเดี่ยว (Single Chip Microcontroler) คือไมโครคอมพิวเตอร์ขนาดเล็กที่บรรจุไว้ในแผงวงจรรวม (Integrated Circuit) เพียงชิพเดี่ยวใช้สำหรับงานควบคุมอุปกรณ์ต่างๆ มีความสะดวกในการใช้งาน

2.5.1 โครงสร้างของไมโครคอลโทรลเลอร์

2.5.1.1 หน่วยประมวลผลกลางขนาด 8 บิต

2.5.1.2 หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)

2.5.1.3 ความสามารถในการอ้างตำแหน่งของ ROM ได้ 64 กิโลไบต์

2.5.1.4 ความสามารถในการอ้างตำแหน่งของ RAM ได้ 64 กิโลไบต์

2.5.1.5 หน่วยความจำแบบ RAM ภายในจำนวน 128 ไบต์ ประกอบด้วย

2.5.1.5.1 รีจิสเตอร์แบงก์ 4 แบงก์ประกอบด้วยรีจิสเตอร์ขนาด8จำนวน8รีจิสเตอร์(RO-R7 )

2.5.1.5.2 มีหน่วยความจำจำนวน 16 ไบต์ ที่สามารถอ้างแอดเอรสเพื่อควบคุมการทำงานในระดับบิตได้

2.5.1.5.3 มีหน่วยความจำสำหรับใช้งานทั่วไป 80 ไบต์

2.5.1.6 พอร์ตอินพุต/พอร์ตเอาท์พุต แบบขนานจำนวน 32 บิต แบ่งเป็น 4 พอร์ตดังนี้

2.5.1.6.1 พอร์ต 0 ( ขา 32-39 ) เป็นพอร์ตที่ใช้งานสองหน้า หน้าที่แรกใช้เป็นอินพุต/เอาท์พุตพอร์ต ส่วนอีกหน้าที่หนึ่งนั้นใช้ควบคุมหน่วยความจำภายนอก เมื่อต้องการขยายระบบให้ใหญ่ขึ้นโดยจะให้สัญญาณที่มัลติเฟลกซ์ระหว่างบัสแอดเดรสกับบัสข้อมูลออกมา (AD7-AD0)

2.5.1.6.2 พอร์ต 1 ( ขา 1-8 ) ใช้เป็นอินพุต/เอาท์พุตพอร์ตอย่างเดียว ใช้สัญลักษณ์เรียงกันเป็นP1.0, P1.1 จนถึง P1.7 พอร์ตนี้ใช้เชื่อต่อกับอุปกรณ์ภายนอกหน้าที่เดียวเท่านั้น ยกเว้น 8032/8052 ที่ใช้ P1.0 และ P1.1 เป็นอินพุต/เอาท์พุต หรืออินพุตภายนอกของไทเมอร์ชุดที่สาม

2.5.1.6.3 พอร์ต 2 ( ขา 21-28 ) เป็นพอร์ตที่ใช้งานสองหน้าที่เช่นกันหน้าที่แรกเป็นอินพุต/เอาท์พุต หน้าที่สองใช้ควบคุมหน่วยความจำภายนอก โดยให้สัญญาณแอดเดรสไบต์สูง ( A15-A8 ) ออกมา

2.5.1.6.4 พอร์ต 3 ( ขา10-17 ) เป็นพอร์ตที่ใช้งานสองหน้าที่เช่นกัน หน้าที่แรกเป็นอินพุต/เอาท์พุตหน้าที่ที่สองแยกออกได้หลายฟังก์ชั่นดังนี้

ตารางที่ 2.1 หน้าที่การทำงานต่างๆของขาในพอร์ต 3

| บิต | ชื่อ | ทำหน้าที่ย่อย |
|---|---|---|
| P3.0 | RXD | รับข้อมูลสำหรับพอร์ตอนุกรมย่อย |
| P3.1 | TX | ส่งข้อมูลสำหรับพอร์ตอนุกรม |

| | | |
|---|---|---|
| P3.2 | INTO | อินเทอร์รัพท์ภายนอกหมายเลข 0 |
| P3.3 | INT1 | อินเทอร์รัพท์ภายนอกหมายเลข 1 |
| P3.4 | T0 | ไทเมอร์/เคาน์เตอร์ 0 (อินพุตจากภายนอก) |
| P3.5 | T1 | ไทเมอร์/เคาน์เตอร์ 1 (อินพุตจากภายนอก) |
| P3.6 | WR | สัญญาณเขียนใช้ต่อกับหน่วยความจำภายนอก |
| P3.7 | RD | สัญญาณอ่านใช้ต่อกับหน่วยความจำภายนอก |

2.5.1.7 วงจร Timer/Conter ขนาด 16 บิต จำนวนสองวงจร

2.5.1.8 วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพลกซ์ (Full Dulplex) เรียกว่า SBUF

2.5.1.9 วงจรควบคุมการอินเทอร์รัพท์จากแหล่งกำเนิดสัญญาณ 5 แหล่ง จากภายใน 3 แหล่ง จากภายนอก 2 แหล่ง

2.5.1.10 ส่วนของออสซิลเลเตอร์ และวงจรสร้างสัญญาณนาฬิกาอยู่ภายใน

2.5.2 ไทเมอร์/เคาน์เตอร์

8051 มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 2 ตัวคือ T0 และ T1 ส่วนเคาน์เตอร์จะแบ่งเป็นรีจิสเตอร์ 8 บิต 2 ตัวคือ TH0 และTL0 และรีจิสเตอร์ T1 ซึ่งมีขนาด 16 บิตเช่นกันจะประกอบด้วยรีจิสเตอร์ขนาด 8 บิต 2 ตัวถึง TH1 และ TL1

การควบคุมการทำงานจะสั่งผ่านรีจิสเตอร์ Tcon (Timer control special function register) และ Tmod (Timer mode control special function register) ซึ่งมีรายละเอียดดังนี้

รีจิสเตอร์ Tcon

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

รูปที่ 2.12 ตำแหน่งในแต่ละบิตของรีจิสเตอร์ TCON

TF1 หมายถึง โอเวอร์โฟวจ์แฟล็กของไทเมอร์ 1 มีค่าเป็น 1 เมื่อเกิดโอเวอร์โฟวจ์ขึ้นและจะถูกเคลียร์ให้เป็น 0 เมื่อไมโครคอลโทรลเลอร์กลับจากการทำงานในโปรแกรมบริการอินเตอร์รัปต์

TR1 หมายถึง บิตที่ใช้บังคับให้ไทเมอร์ 1 ทำงานหรือหยุดทำงาน ถ้าเป็น 1 จะเป็นการให้ไทเมอร์ทำงานและถ้าเป็น 0 จะเป็นการให้ไทเมอร์ 1 หยุดนับแต่ค่าภายในจะไม่เป็น 0

TF0 หมายถึง โอเวอร์โฟวจ์แฟล็กของไทเมอร์ 0 มีค่าเป็น 1 เมื่อเกิดโอเวอร์โฟวจ์ขึ้นและจะถูกเคลียร์เป็น 0 เมื่อไมโครคอลโทรลเลอร์กลับจากการทำงานในส่วนของโปรแกรมบริการอินเตอร์รัปต์

TR0 หมายถึง บิตที่ใช้บังคับให้ไทเมอร์ 0 ทำงานหรือหยุดทำงาน ถ้าเป็น 1 จะเป็นการให้ไทเมอร์ทำงานและถ้าเป็น 0 จะเป็นการให้ไทเมอร์ 0 หยุดนับแต่ค่าภายในจะไม่เป็น 0

TE1 หมายถึง การยอมให้อินเตอร์รัปต์จากภายนอก โดยผ่านทางพอร์ต 3 ขา 3.3 บิตนี้จะถูกเซ็ตเป็น 1 เมื่อสัญญาณจากภายนอกเข้ามากระตุ้นที่ขา 3.3 โดยจะทำการอินเตอร์รัปต์เมื่อสัญญาณจากลอจิก 1 มาเป็นลอจิก 0 ตำแหน่ง เริ่มต้นของโปรแกรมบริการอินเตอร์รัปต์อยู่ที่แอดเดรส์ 0013H เมื่อทำคำสั่ง Ret1 (Return From inter rupt) IE1 จะถูกเคลียร์โดยอัตโนมัติ

IT1 หมายถึง การควบคุมสัญญาณที่เข้ามากระตุ้นที่ขา 3.3 ของพอร์ต 3 ต่อ IT1 = 0 การเลือกการเกิดอินเตอร์รัปต์จากระดับลอจิก 0, และบิต IT=1 เป็นการเลือกการเกิดอินเตอร์รัปต์จากขอบขาลง

IE0 หมายถึง การยอมให้อินเตอร์รัปต์จากภายนอก โดยผ่านทางพอร์ต 3 ขา 3.2 การอินเตอร์รัปต์ที่ขานี้สัญญาณต้องเป็นการกระตุ้นแบบของขอบขาลงซึ่งสามารถใช้เช่นเดียวกับ IE1 ถ้าตำแหน่งแอดเดรสเริ่มต้นของโปรแกรมบริการอินเตอร์รัปต์อยู่ที่แอดเดรส 003H

IT0 หมายถึง การควบคุมสัญญาณที่เข้ามาอินเตอร์รัปต์ IE0 เช่นเดียวกับ IT1

รีจิสเตอร์ TMOD

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Gate | C/T | M1 | M0 | Gate | C/T | M1 | M0 |

รูปที่ 2.13 ตำแหน่งในแต่ละบิตของรีจิสเตอร์ TMOD

Gate เป็นการควบคุมให้ไทเมอร์ 1 หรือ 0 ทำงานหรือไม่ ถ้าบิตนี้เป็น 1 จะยอมให้ไทเมอร์ทำงาน แต่ถ้าค่าเป็น 0 จะไม่ยอมให้ไทเมอร์ไม่ทำงาน C/T ถ้าเป็นลอจิก 1 เป็นการเลือกให้ไทเมอร์ 1 หรือ 0 ทำงานในลักษณะไทเมอร์ แต่ถ้าเป็น 1 จะทำงานในลักษณะของเคาน์เตอร์ทำการนับฟิสว์จากภายนอกที่ป้อนเข้ามาโดยผ่านทางขา 3.5 ของ T1 และ 3.4 ของ T0 แต่ถ้าเป็น 0 จะทำงานเป็นไทเมอร์นับฟิสว์ของ 8051

M1 และ M2 จะเป็นการควบคุมโหมดการทำงานของไทเมอร์/เคาน์เตอร์ดังตาราง

ตารางที่ 2.2 แสดงโหมดการทำงานของไทเมอร์

| M1 | M2 | โหมด |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 2 |
| 1 | 1 | 3 |

โหมดการทำงานของไทเมอร์/เคาน์เตอร์

โหมด 0

โหมด 0 ใช้ไทเมอร์ขนาด 13 บิต โดยใช้ไทเมอร์ไบต์สูง 8 บิต ตัวอนุกรมกับไทเมอร์ต่ำอีก 5 บิต (บิต 0-4) รวมทั้งสิ้น 13 บิตอีกสามบิตของไบต์ต่ำจะไม่ถูกนำมาใช้

โหมด 1

ไทเมอร์โหมด 1 ขนาด 16 บิตและทำงานเหมือนกับโหมด 0 ต่างกันเพียงใช้เดิม 16 บิตเท่านั้น สัญญาณนาฬิกาป้อนให้กับรีจิสเตอร์ไทเมอร์ที่เกิดขึ้นจาก TLX และ THX ประกอบกัน เมื่อได้รับสัญญาณนาฬิกาไทเมอร์จะเริ่มนับจาก 0000H ไปจนถึงค่า FFFFH และค่าของโอเวอร์โฟลว์จะเกิดขึ้นเมื่อเกิดการนับเปลี่ยนจาก FFFFH ไปเป็น 0000H โอเวอร์โฟลว์แฟล็กจะเซ็ตค่าที่ใส่ลงไปสามารถกระทำได้โดยซอฟแวร์โดยบิตที่มีความสำคัญสูงสุด (MSB) จะอยู่ในบิตที่ 7 ของ THX และบิตที่มีความสำคัญน้อยสุดจะอยู่ในบิต 0 ของ TLX

โหมด 2

โหมด 2 ใช้ไทเมอร์ไบต์ต่ำ TLX ทำงานเป็นไทเมอร์ขนาด 8 บิต ในขณะที่ไทเมอร์สูง THX ใช้เก็บค่าที่ต้องการให้โหมดใหม่เอาไว้เมื่อเกิดโอเวอร์โฟลว์ขึ้น หลังจากไทเมอร์นับจาก FFH เป็น 00H แล้วจะมีการโหลดค่าใน THX ให้ใช้ใน TLX อีกครั้ง

โหมด 3

โหมด 3 เป็นการแยกใช้ไทเมอร์อิสระจากกัน โดยไทเมอร์ 0 ในโหมด 3 จะถูกแยกออกเป็นไทเมอร์ขนาด 8 บิต 2 ชุดคือ TL0 และ TH ไทเมอร์ทั้งสองชุดจะแยกทำงานอิสระจากกันแฟล็ก TF0 จะเปิดเมื่อ TL0 เกิดการโอเวอร์โฟล์วและแฟล็ก TF1 จะใช้เมื่อ TH0 เกิดโอเวอร์โฟลว์

2.6 โมดูลอินพุท-เอาท์พุท เพื่อการควบคุมระยะไกล (Remote Terminal Input/Output)

REMOTE TERMINAL INPUT (RTI16) AND OUTPUT (RTO16) เป็นโมดูลอินพุท-เอาท์พุท เพื่อการควบคุมระยะไกล ถูกวิจัยและพัฒนาขึ้นมาเพื่อแก้ปัญหาให้กับการควบคุมอุปกรณ์ไฟฟ้า ด้วยเครื่องคอมพิวเตอร์ โดยตัดข้อยุ่งยากทางด้านการอินเตอร์เฟส สามารถที่จะใช้ควบคุมอุปกรณ์ไฟฟ้า โดยสั่งการด้วยเครื่องคอมพิวเตอร์ การควบคุมผ่านทางพอร์ทอนุกรมแบบ RS232C หรือ RS485 ภายใต้ข้อตกลงเดียวกันในการสื่อสารข้อมูล (Protocol) นอกจากนี้ยังสามารถขยายระบบให้มีจำนวนอินพุท-เอาท์พุทมากขึ้น ในรูปแบบการกระจายออกไปให้เป็นเครือข่ายนับได้ว่าเป็นระบบที่มีความยืดหยุ่นสูง

2.6.1 คุณสมบัติทั่วไปของ Remote Terminal Input ( RTI )

เป็นโมดูลอินพุท ที่ออกแบบมาเพื่อรับค่าสภาวะต่าง ๆ จากอุปกรณ์ตรวจจับสัญญาณในงานอุตสาหกรรม หรือ ในเครื่องจักรกล เช่น Proximity Switch , Limit Switch , Photo Switch เป็นต้นที่มีเอาท์พุทของอุปกรณ์ตรวจจับสัญญาณเป็นแบบ Contact หรือ Transistor ชนิด NPN รับค่าสภาวะได้จำนวน 16 จุด วงจรภายในเป็นแบบ Opto - Isolator ส่วนแสดงผลการรับสภาวะด้วย LED ทีละจุด และแสดงผลเป็นเลขฐานสิบหกได้โดยมี 7 Segment LED 4 หลัก ทำให้สะดวกต่อการสังเกตุค่าสภาวะที่มา การสื่อสารข้อมูลกระทำได้โดยตรงทั้งทาง พอร์ทอนุกรมแบบ RS232C (Point to Point) และ RS485 (Multi Drop) ใช้กับแหล่งจ่ายไฟกระแสตรง 24 โวลท์ 1 แอมป์

2.6.2 คุณสมบัติทั่วไปของ Remote Terminal Output ( RTO )

เป็นโมดูลที่ออกแบบมาเพื่อส่งค่าสภาวะจากคอมพิวเตอร์ไปควบคุมอุปกรณ์ไฟฟ้าในงานอุตสาหกรรม หรือ ในเครื่องจักรกล เช่น Motor, Realy, Solinoil Valve, Lamp เป็นต้น ส่งค่าสภาวะได้จำนวน 16 จุด แบบ Opto - Isolator และวงจรขับสัญญาณเป็น Transistor ชนิด NPN Type ดังนั้นถ้าต้องการขับอุปกรณ์ที่ได้กล่าวมาข้างต้นจำเป็นจะต้องขับผ่าน Relay Board (RLY16) แสดงผลการ เปิด/ปิด ด้วย LED ทีละจุด และแสดงผลเป็นเลขฐานสิบหกได้โดยมี 7 Segment LED 4 หลัก ทำให้สะดวกต่อการสังเกตุค่าสภาวะที่กำลังเป็นอยู่ การสื่อสารข้อมูลกระทำได้โดยตรงทั้งทาง พอร์ทอนุกรมแบบ RS232C (Point to Point) และ RS485 (Multi Drop) ใช้กับแหล่งจ่ายไฟกระแสตรง 24 โวลท์ 1 แอมป์

2.6.3 คุณสมบัติทางเทคนิค

| | |
|---|---|
| INPUT Opto – Isolator | 16 Points |
| Input sensor Contact or Transistor (NPN Type) | 24 VDC. active Low<br>7mA./Point |
| Power Supply | 24 VDC / 1A. |
| 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 Bits/S | DIP Switch NO. 1, 2 |
| Communication | FAC-Talk Protocal |
| Unit Number 0 – 31 | DIP Switch NO 4, 5, 6, 7, 8 |

2.6.4 การกำหนดและเลือก DIP Switch ของ RTI/RTO ให้เป็น Unit Number ต่างๆ

ความเร็วในการรับส่งข้อมูลสามารถเลือกได้โดยปรับ DIP Switch No. 1, 2 ดังนี้

| SW 1 | SW 2 | BAUD Rate |
|---|---|---|
| 0 | 0 | 1200 |
| 0 | 1 | 2400 |
| 1 | 0 | 4800 |
| 1 | 1 | 9600 |

DIP SWITCH 3

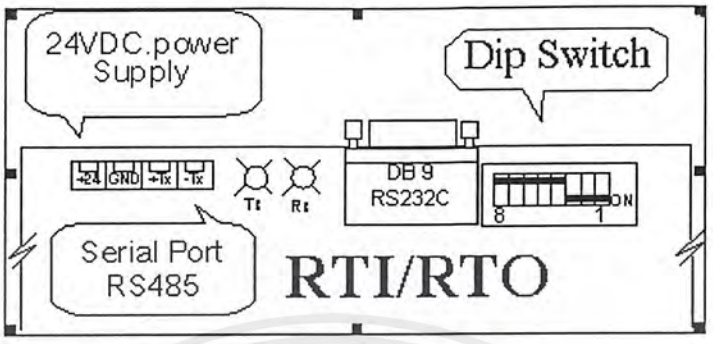
ถ้าเป็น RTI จะต้อง ON เสมอ

ON = 1 OFF = 0

การกำหนดตำแหน่ง Unit Number กำหนดได้โดยการตั้ง 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 |
| 0 | 1 | 0 | 0 | 0 | 8 |
| 0 | 1 | 0 | 0 | 1 | 9 |
| 0 | 1 | 0 | 1 | 0 | 10 |
| 0 | 1 | 0 | 1 | 1 | 11 |
| 0 | 1 | 1 | 0 | 0 | 12 |
| 0 | 1 | 1 | 0 | 1 | 13 |
| 0 | 1 | 1 | 1 | 0 | 14 |
| 0 | 1 | 1 | 1 | 1 | 15 |
| 1 | 0 | 0 | 0 | 0 | 16 |

ON = 1 , OFF = 0



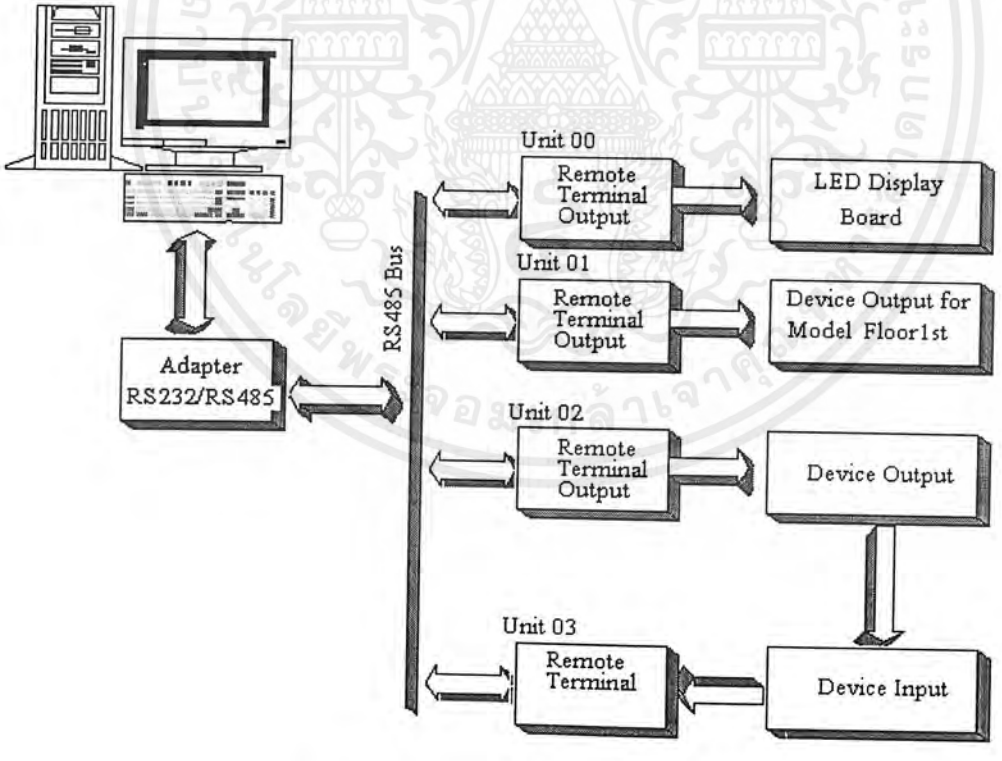

รูปที่ 2.14 ขั้วต่อสายต่าง ๆ ของ RTI16/RIO16

# บทที่ 3
# การคำนวณและการสร้าง

โครงงานนี้แบ่งออกเป็นส่วนหลักต่างๆ ก็คือ

- ส่วนของซอฟต์แวร์ควบคุมการทำงาน
- ฮาร์ดแวร์ของชุดแปลงพอร์ตอนุกรม RS-232 ให้เป็น RS-485 (Convert RS232C/RS485C) โมดูลอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกล (Remote Terminal Input/Output) แผงแสดงผล (LED Display Board) แบบจำลองอาคารภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม (Model Of Instrumentation Department)
- รูปแบบข้อตกลงการสื่อสารข้อมูล
- วิธีการสร้างภาพ 3 มิติ

โดยโครงงานนี้จะมีบล็อกไดอะแกรมการทำงานตามรูปที่ 3.1

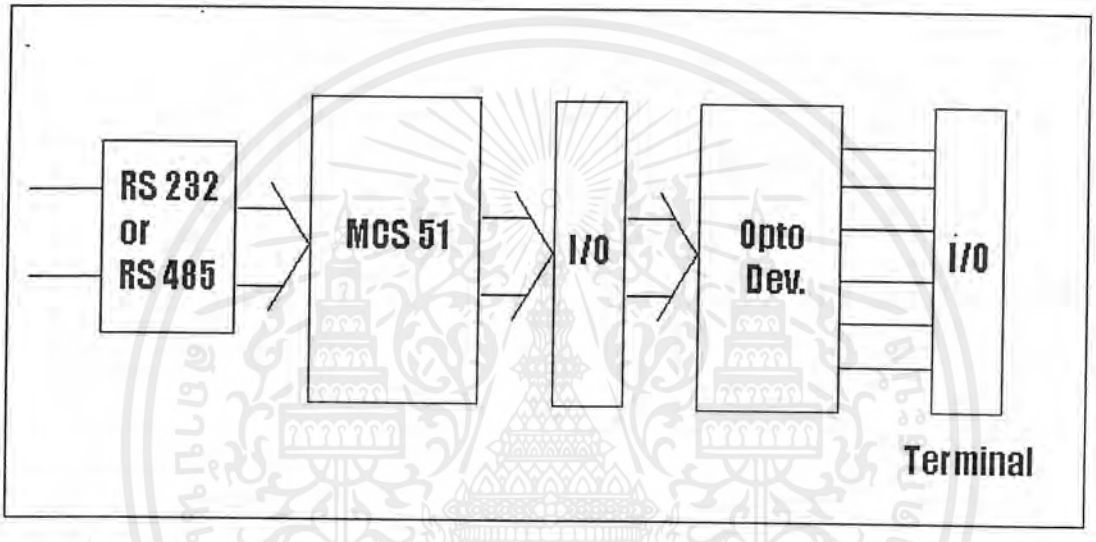



รูปที่ 3.1 แสดงบล็อกไดอะแกรมของระบบทั้งหมด

**3.1 ฮาร์ดแวร์ (Hard Ware)**

3.1.1 โมดูลอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกล (Remote Terminal Input/Output)

ชุดอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกล (Remote Terminal Input/Output) จะรับส่งข้อมูลในรูปแบบของข้อตกลงการสื่อสารข้อมูลแล้วจะวิเคราะห์ข้อมูลเพื่อนำเอาท์พุท/อินพุทไปควบคุมจุดต่างๆ โดยโมดูลอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกลจะมีบล็อกไดอะแกรมของโครงสร้างของโมดูลดังรูปที่ 3.2




รูปที่ 3.2 บล็อกไดอะแกรมโครงสร้างของโมดูลอินพุท/เอาท์พุทเพื่อการควบคุมระยะไกล

- ใช้หน่วยประมวล ผลกลางหรือ CPU : ใช้ไมโครคอนโทรลเลอร์ของ MCS-51
- Watch Dog : เพื่อให้ CPU ทำงานได้ตามวัตถุประสงค์นั้นคือ Watch Dog สั่งให้ CPU เริ่มต้นการทำงานใหม่ทุกๆครั้ง(Reset) เพื่อเข้าสู่การทำงานตามวัตถุประสงค์ เมื่อ CPU ไม่ส่งสัญญาณ ในการทำงานตามวัตถุประสงค์ให้กับมัน
- RS-485 : ใช้ 75176 เป็น Transceiver ในการเชื่อมต่อระหว่าง คอมพิวเตอร์กับ MCS-51 ในการควบคุมการรับการส่งของ 75176 เป็นแบบ Half-Duplex I/O Remote Module จึงตั้งอยู่ในโหมดการรับ เพื่อรอรับคำสั่งจากคอมพิวเตอร์ส่วนการส่งนั้นจะขึ้นอยู่กับคำสั่งที่ได้รับเช่นคำสั่ง RD (Read Data) หรือเป็นการส่ง Error Code ในกรณีของคำสั่งที่ผิดพลาด การรับส่งข้อมูลถูกกำหนดให้เป็นข้อมูลอนุกรมแบบ Asynchnous โครงสร้างของข้อมูลในการรับส่ง 1 Character คือ

  1-Start bit

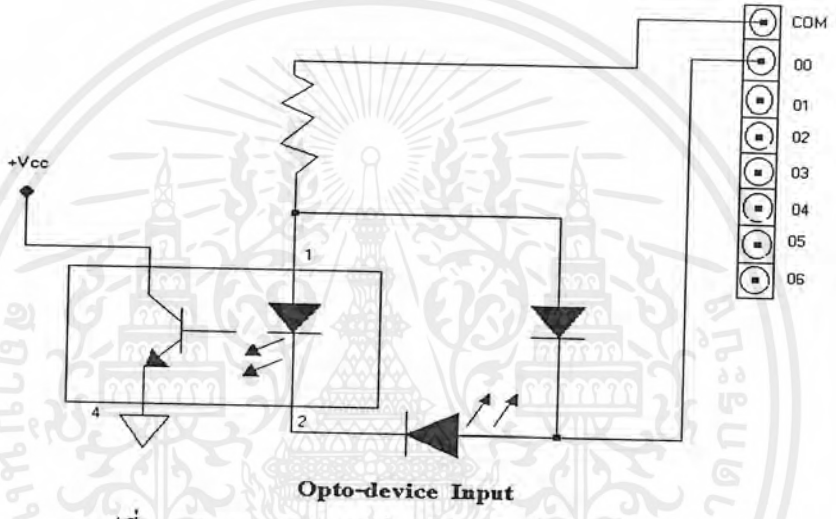
  8-Data bits

  N-Non parity bit

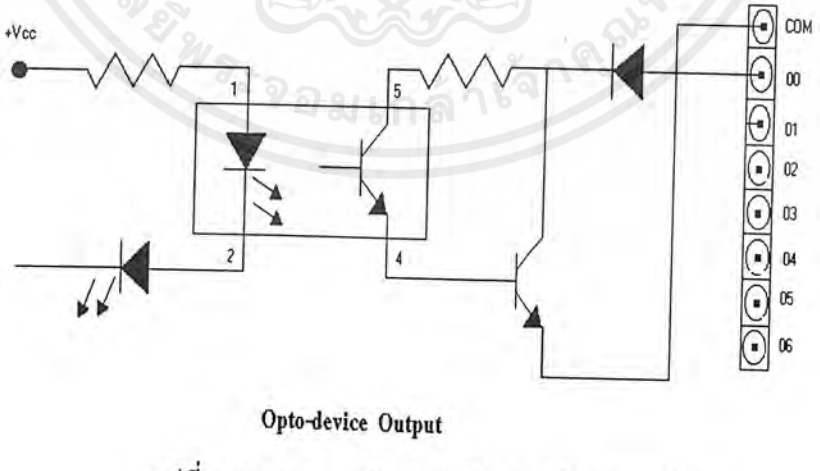
1-Stop bit

Boud Rate ที่ใช้ 9600 Bd,4800Bd,2400Bd และ 1200 Bd โดยการเซ็ตที่ DipSwitch

- I/O : ถูกแยก(Isolate) กันทางไฟฟ้าระหว่าง I/O Remote Module กับอุปกรณ์ที่ถูกควบคุมด้วย Opto-Device มี LED แสดงสถานะ ON/OFF การกำหนดให้เป็น Input หรือ Output ถูกกำหนดโดย Hard ware ที่จะเชื่อมต่อกับอุปกรณ์ที่ถูกควบคุม และ Switch,S3 เพื่อบอกให้ Solf ware ทำงานในโหมด Input หรือ Output ซึ่งในหนึ่งโมดูลมี I/O ทั้งหมด 16 บิท




รูปที่ 3.3 วงจรแยกกันทางไฟฟ้าของโมดูลอินพุท




รูปที่ 3.4 วงจรแยกกันทางไฟฟ้าของโมดูลเอาท์พุท

- Dip Switch : สำหรับตั้ง Configuration ของ Soft ware

  - SW1,SW2 : กำหนด Baud Rate

| SW 1 | SW 2 | BAUD Rate |
|---|---|---|
| 0 | 0 | 1200 |
| 0 | 1 | 2400 |
| 1 | 0 | 4800 |
| 1 | 1 | 9600 |

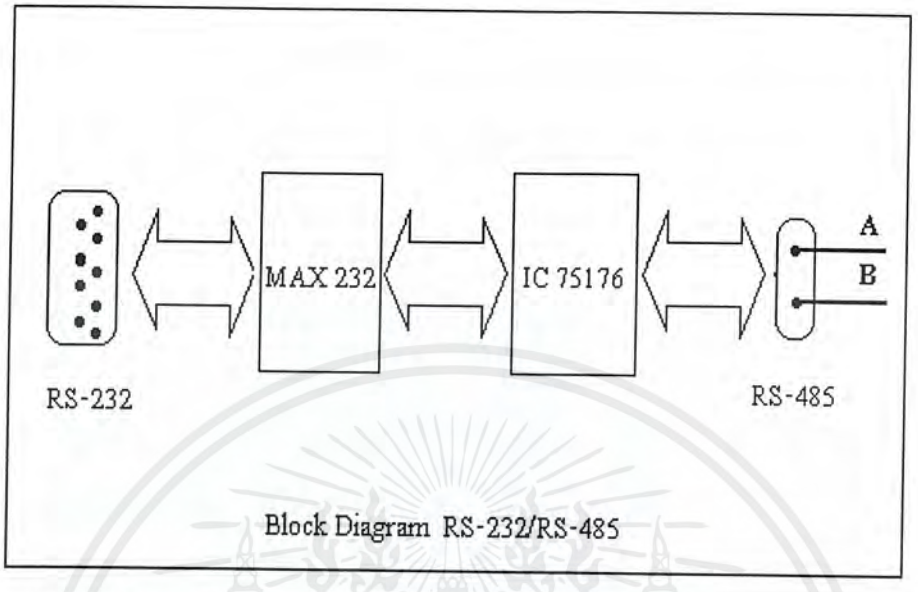
  - SW3 : กำหนดชนิดโมดูล

| SW 2 | Type of Module |
|---|---|
| 0 | Input Module |
| 1 | Output Module |

  - SW4 : Reserve
  - SW5, SW6, SW7, SW8 : กำหนดที่อยู่ของโมดูล (Module Address) 0000B-1111B (0H-0FH)

3.1.2 ชุดแปลงพอร์ตอนุกรม RS-232 ให้เป็น RS-485 (Convert RS232C/RS485C)

ชุดแปลงพอร์ตอนุกรม RS-232 ให้เป็น RS-485 จะทำการแปลงข้อมูลจากพอร์ตอนุกรมตามมาตรฐานแบบRS-232 ให้เป็นมาตรฐานแบบRS-485 เพื่อที่จะรับส่งผ่าน(Bus 485)ออกไปยังชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล โดยจะมีบล็อกไดอะแกรมการทำงานตามรูปที่ 3.5

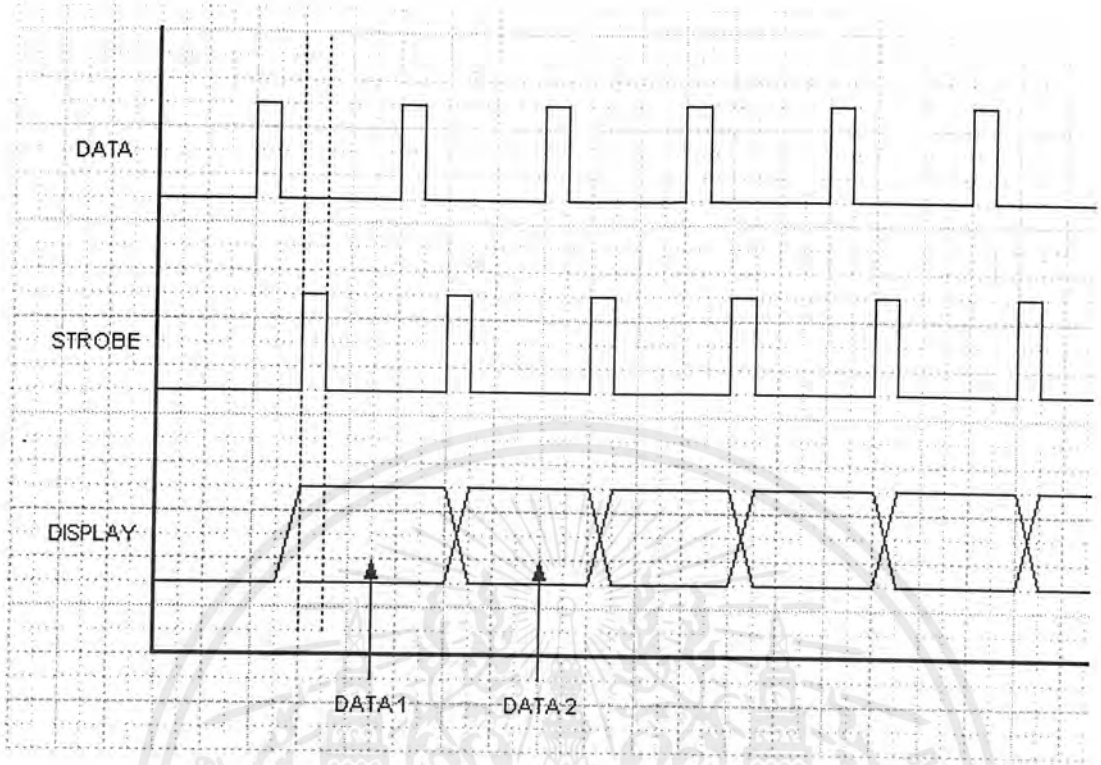



รูปที่ 3.5 บล็อกไดอะแกรมการทำงานของชุดแปลงพอร์ตอนุกรม RS-232 ให้เป็น RS-485

- ไอซี MAX-232 : คอยทำหน้าที่แปลงแรงดันในมาตราฐาน RS-232 ให้อยู่ในรูปทีทีแอล และในทางกลับกันคือแปลงแรงดันอินพุทที่ป้อนเข้า MAX-232 ให้อยู่ในรูปแบบมาตราฐาน RS-232 เพื่อส่งให้คอมพิวเตอร์
- ไอซี 75176 : ทำหน้าที่แปลงสัญญาณให้เป็นมาตราฐาน RS-485 โดย RS-485 เป็นการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์จะใช้สายสัญญาณ RTS มาทำหน้าที่ควบคุมทิศทางของการสื่อสาร

3.1.3 แผงแสดงผล (LED Display Board)

หน่วยแสดงผลแบบตัวอักษรจะแสดงข้อมูลในลักษณะของข้อความที่เป็นตัวอักษร โดยตัวอักษรดังกล่าวจะใช้แสดงด้วยไดโอดชนิดเปล่งแสงแบบสว่างมาก(Super Bright L.E.D.) จัดเรียงขึ้นเป็นเมทริกขนาด 5*7 สามารถแสดงผลเป็นตัวอักษรในภาษาอังกฤษได้ทั้งตัวใหญ่และตัวเล็ก ตัวเลข ตลอดจนเครื่องหมายที่จำเป็นได้อย่างครบถ้วน เพียงแต่ป้อนสัญญาณรหัส ASCII ของตัวอักษรใดๆ และมีสัญญาณกำกับอีกหนึ่งสัญญาณ (Latch) เข้าไปในวงจรหน่วยแสดงผล ก็จะปรากฏตัวอักษรนั้นๆ ซึ่งแผงแสดงผลนี้ได้นำหน่วยแสดงผลแบบตัวอักษรมาจัดเรียงให้มีการแสดงผลแบบ8 ตัวอักษรโดยได้ขนานสัญญาณ ASCII และทำการ Latch ซึ่งแผงแสดงผลจะมีไดอะแกรมเวลาตามรูปที่ 3.6




รูปที่ 3.6 แสดงการทำงานของแผงแสดงผล

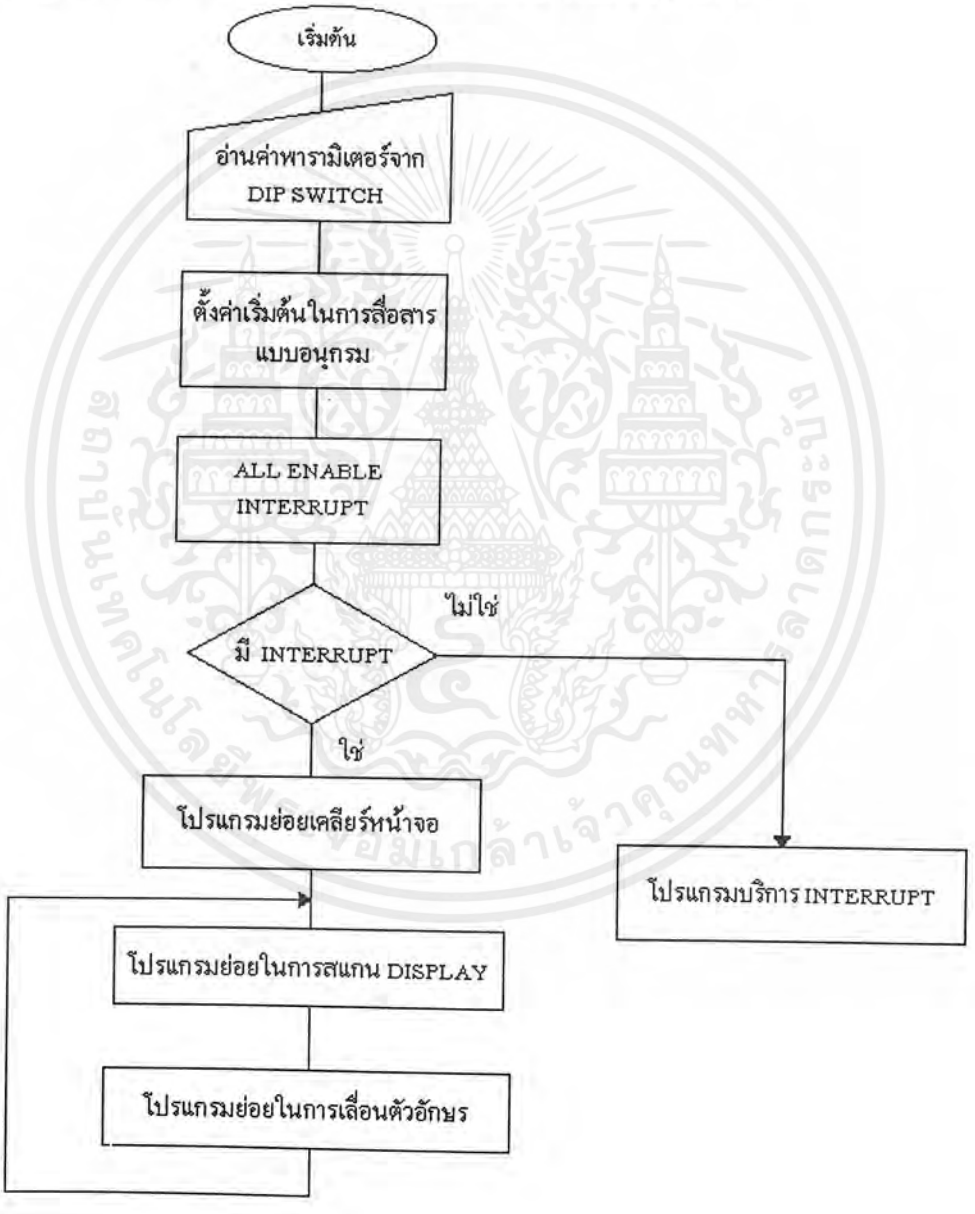
**3.2 Soft ware**

จะแบ่งออกเป็นสองส่วนคือ

3.2.1 การออกแบบโปรแกรมอินเตอร์เฟสระหว่างคอมพิวเตอร์กับชุดเอาท์พุทเพื่อการควบคุมระยะไกลเพื่อแสดงผลออกทาง LED Borad

ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51การทำงานของหน่วยแสดงผลเป็นการประยุกต์หน่วยควบคุมระยะไกลเพื่อให้รับข้อความจากโปรแกรมที่ทำขึ้นในคอมพิวเตอร์โดยผ่านข้อตกลงในการสื่อสารคือเพิ่มส่วนที่เป็นคำสั่งในการรอรับข้อมูลขึ้นมาจากรูปแสดงการทำงานอธิบายได้ว่าในการเริ่มต้นระบบจะมีการตรวจสอบว่าหน่วยแสดงผลนี้มีค่าตำแหน่งที่เท่าไรโดยตรวจจากดิพสวิทช์ที่บนตัวอุปกรณ์และจะเก็บค่าตำแหน่งนั้นไว้เพื่อที่จะนำมาตรวจสอบกับคำสั่งที่ส่งมาว่าคำสั่งนั้นได้ส่งมาเพื่อติดต่อกับตัวหน่วยแสดงผลรึเปล่าเพราะการติดต่อสื่อสารกันในระบบได้ใช้การต่อแบบขนานกับอุปกรณ์ภายในระบบต่อมาจะทำการตั้งค่าการพารามิเตอร์ในการส่งและรับข้อมูลเพื่อตั้งบอตเรทและเงื่อนไขการสื่อสารแบบอนุกรมผ่านการตั้งค่าแล้วจึงเปิดระบบโดยการเซ็ทอีนาเบิ้ลอินเตอร์รัปของไมโครคอนโทรลเลอร์เพื่อให้สามารถมีการขัดจังหวะจากการส่งข้อมูลได้และต่อมาจะมาตรวจสอบว่ามีการขัดจังหวะเข้ามาหรือไม่ถ้ามีการขัดจังหวะเข้ามาก็ให้ไปทำงานในส่วนของ

โปรแกรมบริการอินเตอร์รัปและถ้าไม่มีการขัดจังหวะก็จะไปทำงานที่โปรแกรมเคลียร์หน้าจอของตัวแสดงผลและจะทำการสแกนเพื่อส่งข้อมูลออกมาที่หน้าจอจากนั้นก็จะทำการเลื่อนตัวอักษรที่ส่งมาและจะวนรอบการทำงานไปที่การสแกนอีกครั้งไปเรื่อยๆจนหมดตัวอักษรที่ส่งมาแล้วก็จะมาทำงานในส่วนนี้ใหม่ไปเรื่อยๆในขณะที่ทำงานอยู่ก็จะรอรับการขัดจังหวะซึ่งถ้าไม่มีการขัดจังหวะเข้ามาหน่วยแสดงผลก็จะทำงานอยู่ไปเรื่อยๆ ซึ่งโฟร์ชาร์ทของการทำงานของโปรแกรมหลักได้แสดงไว้ในรูปที่ 3.7 ส่วนซอร์ทโค้ด (source code) แสดงไว้ในภาคผนวก ก.

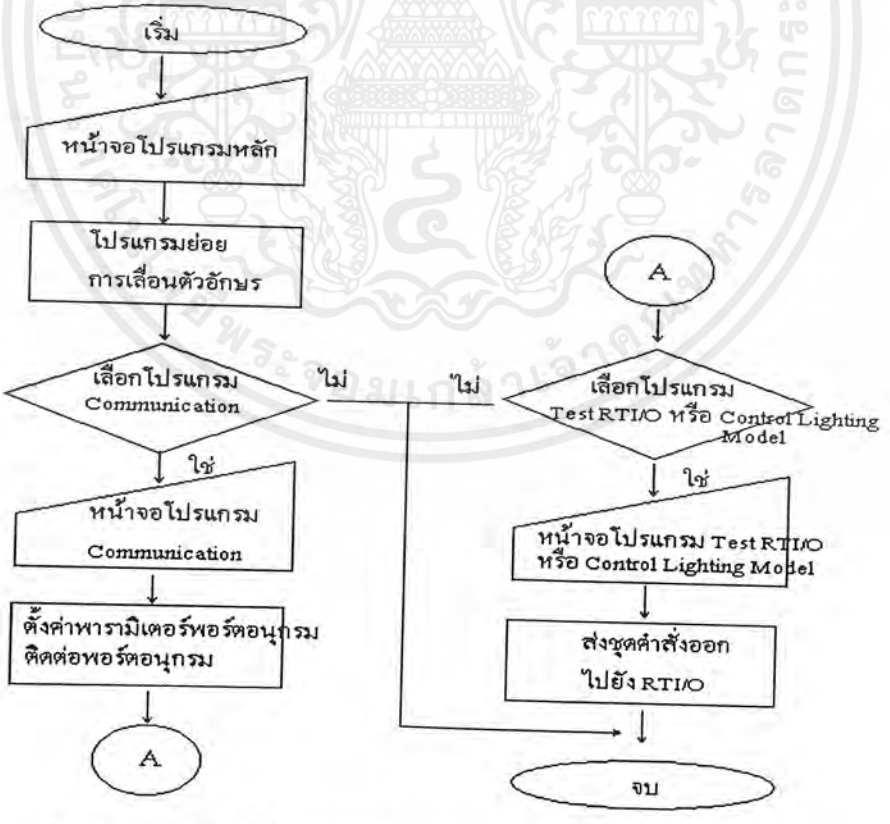



รูปที่ 3.7 ไดอะแกรมการทำงานโปรแกรมหลักของหน่วยแสดงผล

3.2.2 การออกแบบโปรแกรมที่จะติดต่อกับผู้ใช้โปรแกรม ในปริญญานิพนธ์นี้ได้เขียนโดยใช้โปรแกรม Delphi ซอร์ทโค้ด ได้แสดงไว้ในภาคผนวก ก ส่วนลักษณะการออกแบบโปรแกรมเพื่อติดต่อกับผู้ใช้ได้ะแบ่งออกเป็นโปรแกรมส่วนๆเพื่อสะดวกในการใช้งานและการแก้ไขดัดแปลงโปรแกรมให้เหมาะสมกับอุปกรณ์ภายนอก ซึ่งแบ่งออกได้ดังนี้

3.2.2.1 โปรแกรมหลัก (Main Menu Program)

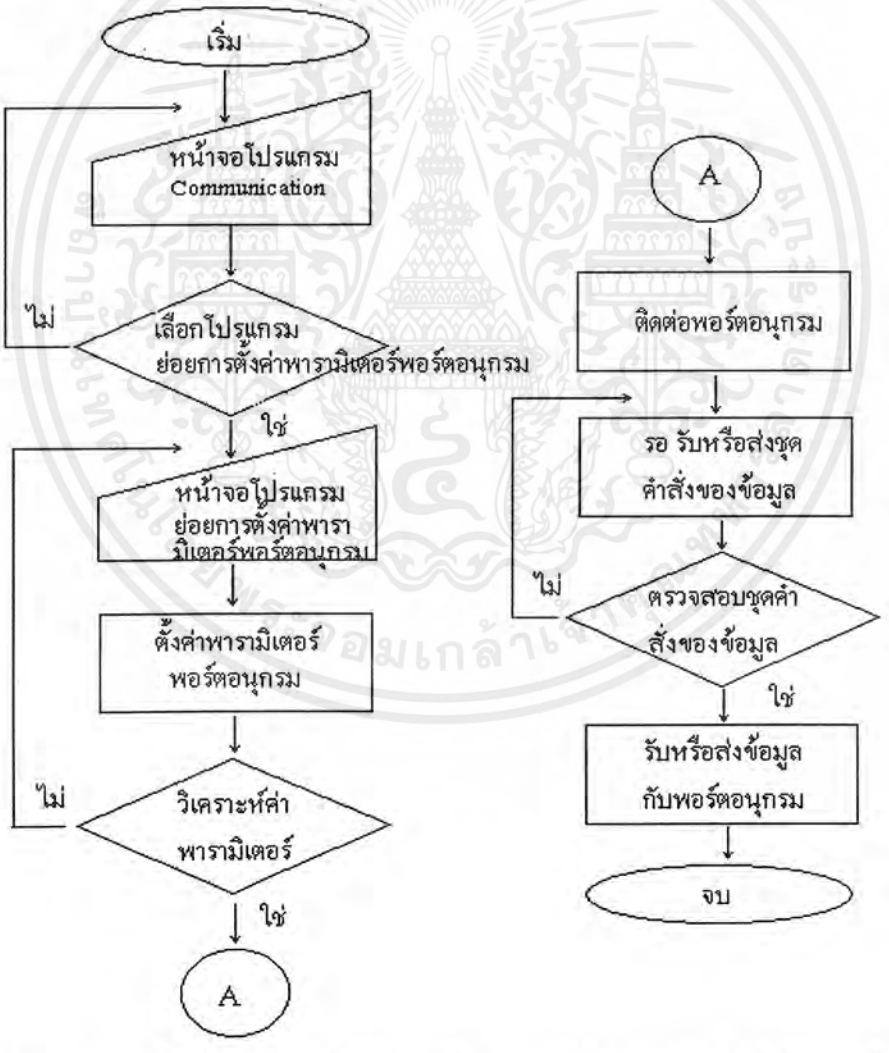
โปรแกรมหลักถูกออกแบบเพื่อที่จะทำหน้าที่ในเรียกใช้โปรแกรมส่วนอื่นๆมาใช้งานโดยจะเริ่มจากการรันโปรแกรมขึ้นมาจะปรากฏหน้าจอโปรแกรมหลักขณะนั้นโปรแกรมย่อยการเลื่อนตัวอักษรและโปรแกรมย่อยแสดงภาพ 3มิติก็จะเริ่มทำงานด้วย แล้วทำการเลือกโปรแกรมการติต่อสื่อสารข้อมูลเพื่อทำการติดต่อกับพอร์ตอนุกรม จากนั้นทำการเลือกโปรแกรมการทดสอบชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล หรือโปรแกรมการควบคุมระบบแสงสว่างของแบบจำลองเพื่อที่จะส่งชุดคำสั่งภายใต้ข้อตกลงการสื่อสารข้อมูลไปยังชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกลวิเคราะห์ชุดคำสั่งต่อไปเพื่อแสดงการทำงาน โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.8 รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.13




รูปที่ 3.8 แสดงลำดับขั้นตอนการทำงานการทำงานของโปรแกรมหลัก

3.2.2.2 โปรแกรมการติดต่อสื่อสารข้อมูล (Communication Program)

โปรแกรมการติดต่อสื่อสารข้อมูลถูกออกแบบเพื่อทำหน้าที่รับส่งและตรวจชุดของคำสั่งข้อมูลระหว่างโปรแกรมใช้งานและพอร์ตอนุกรมทั้งแบบ RS-232 และ RS-485 โดยจะมีส่วนของโปรแกรมย่อยการตั้งค่าพารามิเตอร์สำหรับการตั้งค่าพารามิเตอร์และตรวจสอบความถูกต้องว่าเหมาะสมกับฮาร์ดแวร์หรือไม่ ส่วนการ ตรวจสอบชุดคำสั่งของข้อมูลก็จะมีโปรแกรมย่อยสำหรับตรวจสอบโดยทำการเปรียบเทียบชุดคำสั่งของข้อมูลกับชุดคำสั่งของข้อมูลที่ตั้งไว้ถ้าถูกต้องแล้วก็จะส่งข้อมูลให้โปรแกรมย่อยสำหรับรับส่งข้อมูลทำหน้าที่รับส่งข้อมูลให้กับโปรแกรมการใช้งานหรือพอร์ตอนุกรมต่อไป โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.9 รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.14 และรูปที่ 3.15

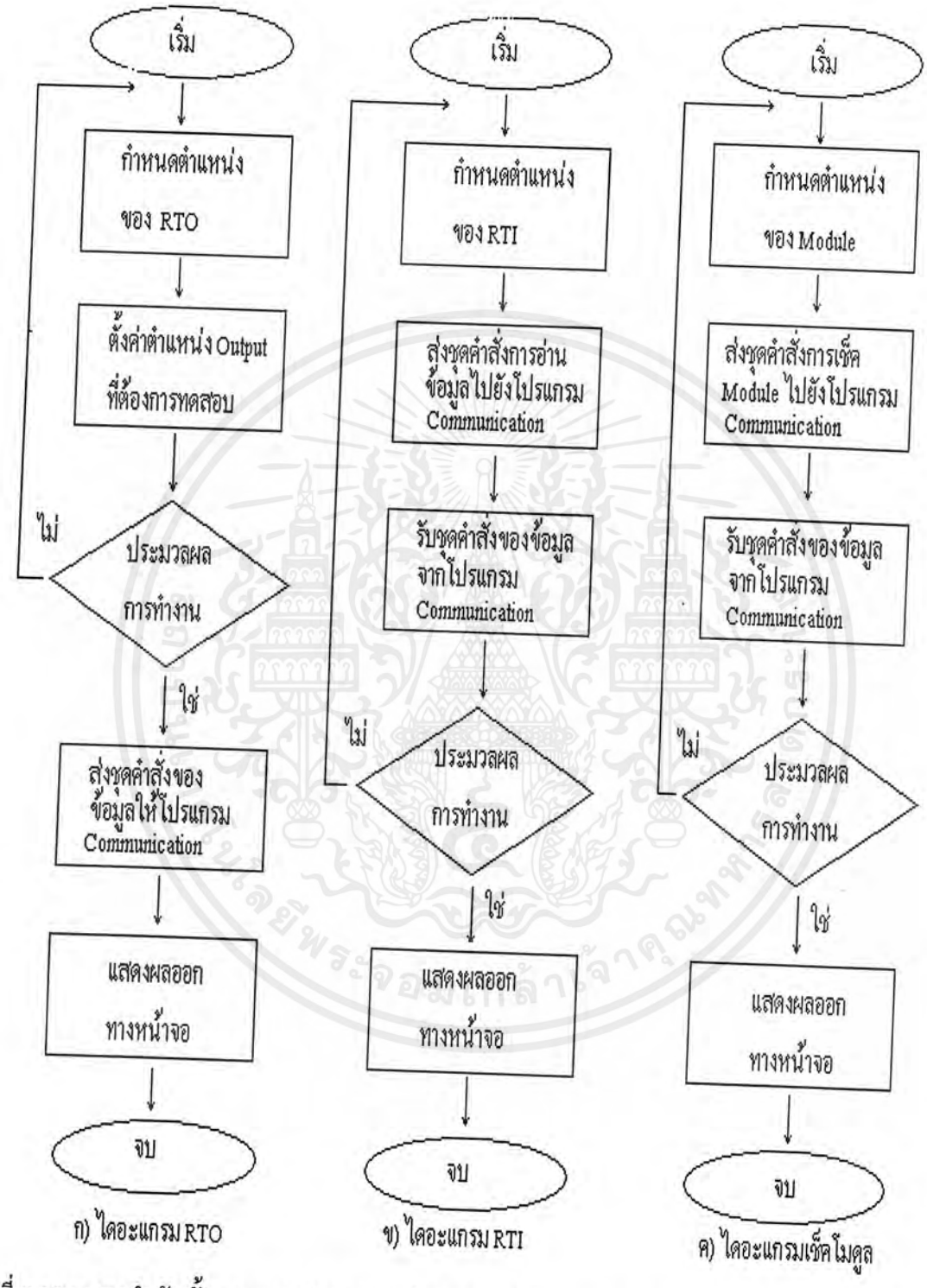



รูปที่ 3.9 แสดงลำดับขั้นตอนการทำงานการทำงานของโปรแกรมการติดต่อสื่อสารข้อมูล

3.2.2.3 โปรแกรมทดสอบชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล (Remote Terminal Test Program)

โปรแกรมทดสอบชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล ถูกออกแบบเพื่อใช้ทดสอบการทำงานทั้งของชุดอินพุทและชุดเอาท์พุท เพื่อการควบคุมระยะไกล พร้อมทั้งสามารถเช็คโมดูลว่าเป็นโมดูลอินพุทหรือโมดูลเอาท์พุท ซึ่งโปรแกรมนี้แบ่งออกเป็น 3 ส่วนคือ

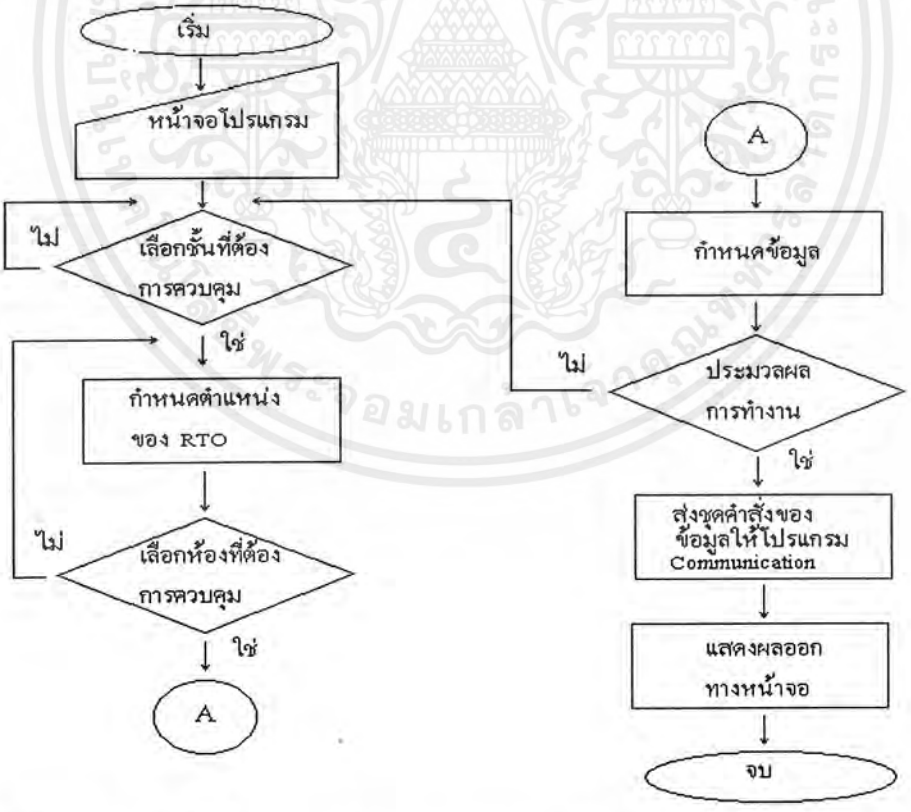
- ส่วนทดสอบชุดเอาท์พุท เพื่อการควบคุมระยะไกล (RTO) โดยจะมีขั้นตอนการทำงานดังนี้ เริ่มจากรับค่าข้อมูลที่เป็นรหัส ASCII เพื่อเป็นตำแหน่งของชุดเอาท์พุทเพื่อการควบคุมระยะไกล(RTO) จากนั้นก็จะรับค่าของข้อมูลที่ต้องการให้แสดงออกทางเอาท์พุทของ RTO ทั้งแบบเป็นรหัสเลขฐานสอง(Binary) หรือรหัสเลขฐานสิบหก(Hex) โดยจะมีโปรแกรมย่อยสำหรับแปลงค่าข้อมูลให้เป็นเลขทั้งสองฐาน จากนั้นจะนำตำแหน่งของ RTO และค่าข้อมูลไปใส่ในชุดคำสั่งของมูลของการส่งค่าไปยัง RTO เพื่อที่จะส่งชุดคำสั่งไปยังโปรแกรม Communication โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.10 ก) รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.16
- ส่วนทดสอบชุดอินพุท เพื่อการควบคุมระยะไกล (RTI) โดยจะมีขั้นตอนการทำงานคล้ายๆกับส่วนทดสอบชุดเอาท์พุท เพื่อการควบคุมระยะไกล โดยจะแตกต่างกันที่จะต้องส่งชุดคำส่งของการค่าข้อมูลไปยัง RTI ก่อนแล้วรอรับชุดคำสั่งของข้อมูลกลับจาก RTI แล้วนำข้อมูลมาวิเคราะห์เพื่อที่จะแสดงให้อยู่ในรูปของเลขฐานสอง หรือเลขฐานสิบหก โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.10 ข) รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.16
- ส่วนเช็คโมดุลของชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล (RTI/RTO) จะมีขั้นตอนการทำงานคล้ายๆกับส่วนทดสอบชุดเอาท์พุท เพื่อการควบคุมระยะไกล โดยจะแตกต่างกันที่จะต้องส่งชุดคำส่งของเช็คโมดูลไปยังชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล ก่อนแล้วรอรับชุดคำสั่งของข้อมูลกลับจากชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล แล้วนำข้อมูลมาวิเคราะห์ตรวจสอบว่าข้อมูลที่ได้รับกลับมานั้นตรงกลับข้อมูลของโมดูลอินพุทหรือโมดูลเอาท์พุท แล้วนำค่าที่ได้มาแสดงทางหน้าจอ โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.10 ค) รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.16




รูปที่ 3.10 แสดงลำดับขั้นตอนการทำงานการทำงานของโปรแกรมทดสอบชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล

3.2.2.4 โปรแกรมการควบคุมระบบแสงสว่างของชุดจำลอง (Control Lighting Model Program)

โปรแกรมการควบคุมระบบแสงสว่างของชุดจำลองถูกออกแบบให้ใช้ในการเปิด-ปิดไฟตามห้องของชั้นต่างๆของชุดจำลอง โดยลักษณะของการออกแบบโปรแกรมจะเริ่มจากการกำหนดตำแหน่งของชุดเอาท์พุท เพื่อการควบคุมระยะไกล(RTO) 1ชุดต่อ 1ชั้น และตำแหน่งเอาท์พุทของชุดเอาท์พุท เพื่อการควบคุมระยะไกล ทั้ง 16 จุดให้ตามห้องต่างต่างๆของชั้นนั้นๆ จากนั้นใช้ชุดคำสั่งของการส่งค่าไปยังชุดเอาท์พุท เพื่อการควบคุมระยะไกล ส่วนขั้นตอนการทำงานเริ่มการเลือกโปรแกรมการควบคุมระบบแสงสว่างของชุดจำลองจากหน้าจอโปรแกรมหลัก จากนั้นเลือกชั้นที่ต้องการควบคุมดังนั้นจะได้ตำแหน่งของ RTO แล้วก็ทำการเลือกห้องที่ต้องการควบคุมจะได้ค่าข้อมูลของชุดคำสั่ง แล้วทำการประมวลผลก็จะได้ชุดคำสั่งของการส่งค่าไปยังโปรแกรม Communication เพื่อทำการส่งชุดคำสั่งของข้อมูลผ่านพอร์ตอนุกรมไปยังชุดเอาท์พุท เพื่อการควบคุมระยะไกลทำงาน แล้วทำการแสดงผลการทำงานทางหน้าจอโปรแกรม โดยซอร์ทโค้ดได้แสดงไว้ในภาคผนวก ก. และไดอะแกรมแสดงลำดับขั้นตอนการทำงานแสดงดังรูปที่ 3.11 รูปแสดงหน้าจอโปรแกรมดังรูปที่ 3.17




รูปที่ 3.11 แสดงลำดับขั้นตอนการทำงานการทำงานของโปรแกรมการควบคุมระบบแสงสว่าง

### 3.3 ข้อตกลงในการสื่อสาร (Protocol)

ข้อกำหนดของข้อมูลแบบอนุกรม [ 8 N 1 ]

| 1 บิทเริ่ม | 8 บิทข้อมูล | N พาริตี้ | 1 บิทหยุด |
|---|---|---|---|

ข้อตกในการสื่อสารถูกออกแบบให้อยู่ในรูปของรหัส ASCII จัดเป็นบล็อค โดยแบ่งเป็น

- บล็อคคำสั่ง (Command Block)
- บล็อคตอบสนอง (Response Block)

ในแต่ละบล็อคจะประกอบด้วยสาระสำคัญดังนี้

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

3.3.1 รูปแบบบล็อคคำสั่ง

| | |
|---|---|
| @ | เป็นอักขระเริ่มต้นของบล็อค ค่าฐานสิหกเท่ากับ 40(Hex) |
| Unit Number | ตำแหน่งของโมดูลที่คอมพิวเตอร์อ้างอิง (0 - 31)<br>เลือกจากการกำหนดที่ DIP Switch No. 4,5,6,7,8 |
| Header | คำสั่ง / รหัส ให้ปฏิบัติ เป็น ASCII 2 อักขระ อันได้แก่<br>**RD** : หมายถึงคำสั่งในการอ่านข้อมูลจากโมดูล RTI16<br>**WR** : หมายถึงคำสั่งในการเขียนข้อมูลให้กับโมดูล RTO16<br>**MD** : หมายถึงคำสั่งในการอ่านชนิดของโมดูลว่าเป็น RTI16 หรือ RTO16<br>**SM** : หมายถึงคำสั่งในการแสดงข้อความใช้ติดต่อกับหน่วยแสดงผล |
| Data | ข้อมูลประกอบคำสั่ง / รหัส เป็น ASCII 2 - 4 อักขระ<br>**RD** : ไม่ต้องมีข้อมูลประกอบ<br>**WR** : ข้อมูลที่เป็นรหัส ASCII แทนค่าฐานสิบหก เช่น ข้อมูล 12AB(Hex) ชุดข้อมูลASCII จะเป็น 31324142<br>**MD** : ไม่ต้องมีข้อมูลประกอบ<br>**LM** : ข้อมูล 2 ตัวแรกเป็นรูปแบบการแสดงผลหลังจากนั้นเป็นข้อมูลที่จะแสดงที่หน่วยแสดงผลซึ่งรับค่าเป็นรหัส ASCII ขนาดไม่เกิน 25 อักษร |
| * | เป็นอักขระปิดท้ายข้อมูล ค่าฐานสิหกเท่ากับ 2A(Hex) |
| BCS | รหัสกำกับบล็อค เป็น ASCII 2 อักขระ |

| | |
|---|---|
| หรือ<br>XX | ได้จากการ XOR ข้อมูล ASCII ทุกๆ อักขระในแต่ละบล็อค<br>ถ้าเป็น XX โมดูลจะรู้ว่า ไม่มีการตรวจสอบ BCS |
| [CR] | รหัส ASCII (Return Code) 1 อักขระ |

3.3.2 รูปแบบบล็อคตอบสนอง

| | |
|---|---|
| @ | เป็นอักขระเริ่มต้นของบล็อค ค่าฐานสิหกเท่ากับ 40(Hex) |
| Unit<br>Number | ตำแหน่งของโมดูลที่คอมพิวเตอร์อ้างอิง (0 - 31)<br>เลือกจากการกำหนดที่ DIP Switch No. 4,5,6,7,8 |
| Header | คำสั่ง / รหัส ให้ปฏิบัติ เป็น ASCII 2 อักขระ อันได้แก่<br>**RD** : หมายถึงคำสั่งที่โมดูลตอบกลับเพื่อส่งข้อมูลให้PC84SF หรือคอมพิวเตอร์ จาก RTI16<br>**MD** : หมายถึงคำสั่งที่โมดูลตอบกลับเพื่อบอกชนิดของโมดูลว่าเป็น RTI16 หรือ RTO16<br>**ER** : เมื่อการปฏิบัติคำสั่งใด ๆ เกิดข้อผิดพลาดโมดูลจะตอบกลับ |
| Data | ข้อมูลประกอบคำสั่ง / รหัส เป็น ASCII 2- 4 อักขระ<br>**RD** : ข้อมูลที่ตอบกลับมาจะเป็นค่าสภาวะปัจจุบันขณะอ่าน เป็นข้อมูลที่เป็นรหัส ASCII แทนค่าฐานสิบหก เช่น ข้อมูล 12AB(Hex) ชุดข้อมูล ASCII จะเป็น 31324142 เป็นต้น<br>**MD** : 00 เป็น RTO16 [ASCII 2 อักขระ]<br>01 เป็น RTI16<br>**ER** : 00 :Over Range อักขระในบล็อคไม่อยู่ในพิกัด<br>01 :Format Error รูปแบบผิด<br>02 :Block Check Sum Error รหัสกำกับบล็อคไม่ถูกต้อง<br>03 :Unknow Command ไม่รู้จักคำสั่งนี้<br>04 :Execute Error ปฏิบัติการไม่ได้ หรือมีข้อผิดพลาดอื่น ๆ |
| * | เป็นอักขระปิดท้ายข้อมูล ค่าฐานสิหกเท่ากับ 2A(Hex) |
| BCS | รหัสกำกับบล็อค เป็น ASCII 2 อักขระ<br>ได้จากการ XOR ข้อมูล ASCII ทุกๆ อักขระในแต่ละบล็อค |
| [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,Chksum  : byte;
        BlockCommand  : string [128];
begin
        Clrscr;
        BlockCommand := '@02WR90AF*';
        Chksum := ord(BlockCommand[1]);
        for j := 2 to LENGTH(BlockCommand) do
           begin
              Chksum := Chksum XOR ord(BlockCommand[j]);
              writeln (BlockCommand[j],'  ',Chksum);
           end;
end.
```

ตัวอย่างการใช้งาน

1. การเขียนข้อมูลไปยังโมดูลเอาท์พุทเพื่อการควบคุมระยะไกล ซึ่ง ถูกตั้งค่าตำแหน่งไว้ที่ 02 และต้องการให้ค่าข้อมูลเป็น 90AF (Hex) จะต้องทำการจัดรูปแบบบล็อคคำสั่งเพื่อส่งให้โมดูลเอาท์พุทดังนี้

| | | | | | | |
|---|---|---|---|---|---|---|
| @ | 02 | WR | 90AF | * | 63 | [CR] |

"@ 0 2 W R 9 0 A F * 6 3" , [CR] การคำนวณ BCS จะได้ค่า 63 หรือ "@ 0 2 W R 9 0 A F * XX" , [CR] ก็ได้ และเมื่อปฏิบัติเสร็จสิ้นไม่มีข้อผิดพลาดใดๆ จะไม่มีข้อมูลใดตอบกลับออกมา

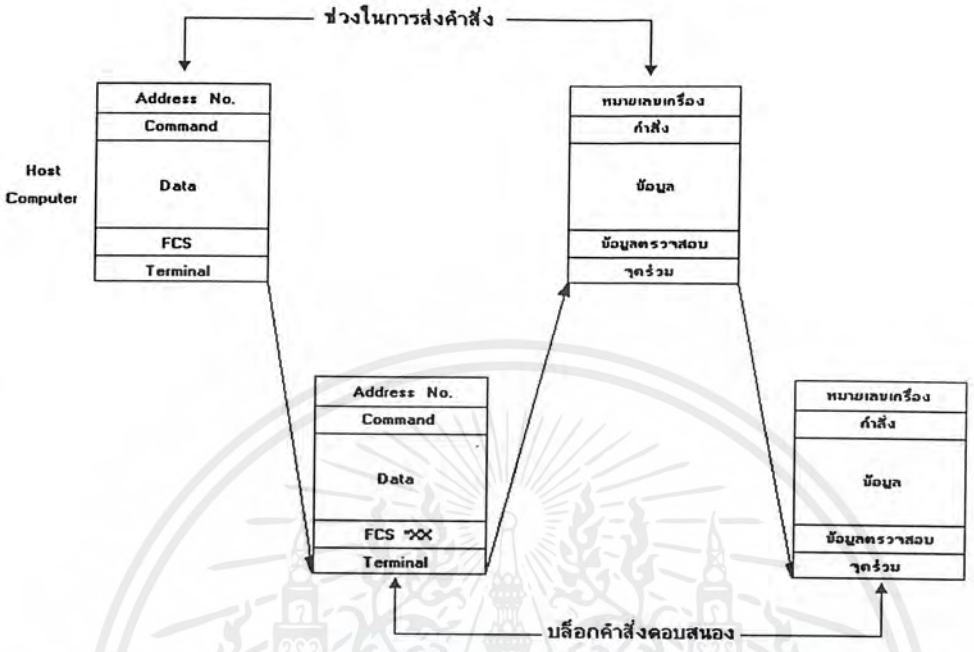
2. การอ่านข้อมูลจากโมดูลอินพุทเพื่อการควบคุมระยะไกล โดยถูกตั้งค่าตำแหน่งไว้ที่ 10 จะต้องทำการจัดรูปแบบบล็อคคำสั่ง เพื่อไปถามค่าของข้อมูลปัจจุบันดังนี้

| | | | | | |
|---|---|---|---|---|---|
| @ | 0A | RD | * | 0D | [CR] |

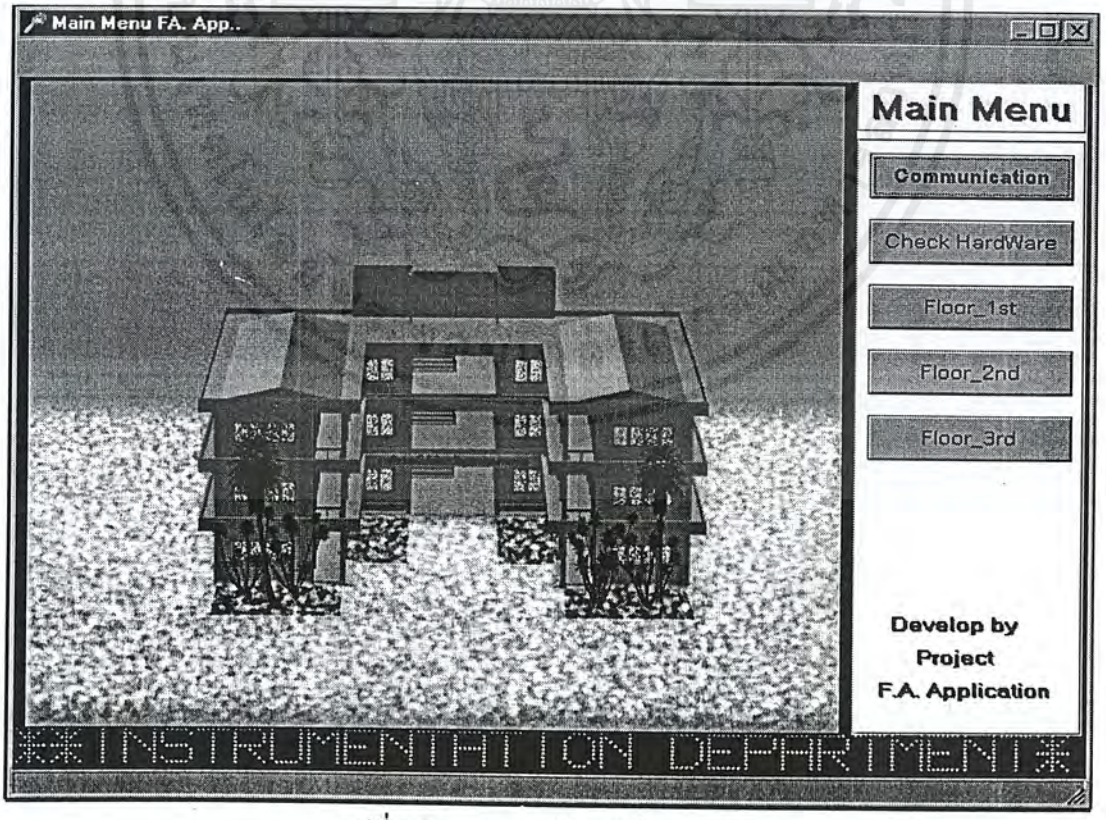
"@0ARD*0D", [CR] คำนวณ BCS ได้ 0D หรือ "@0ARD*XX", [CR] สมมุติว่าข้อมูลปัจจุบันของ RTI16 นั้นเป็น 55AA (Hex) Module จะตอบออกมาดังนี้

| | | | | | | |
|---|---|---|---|---|---|---|
| @ | 0A | RD | 55AA | * | 0D | [CR] |

"@0ARD55AA*0D" , [CR]

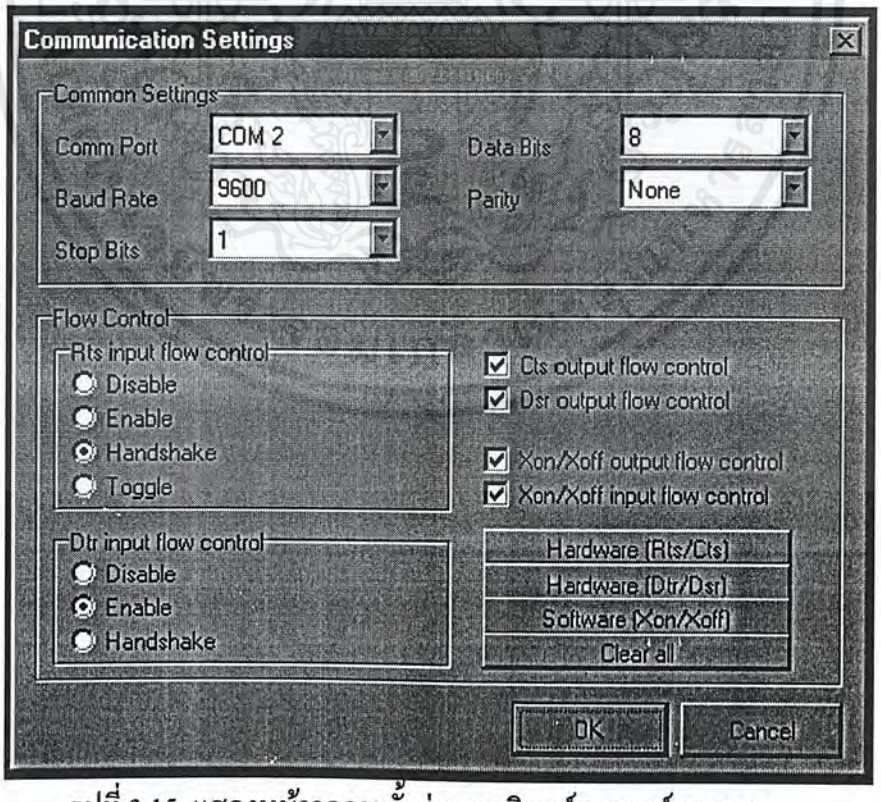



รูปที่ 3.12 ลักษณะการติดต่อสื่อสารข้อมูล

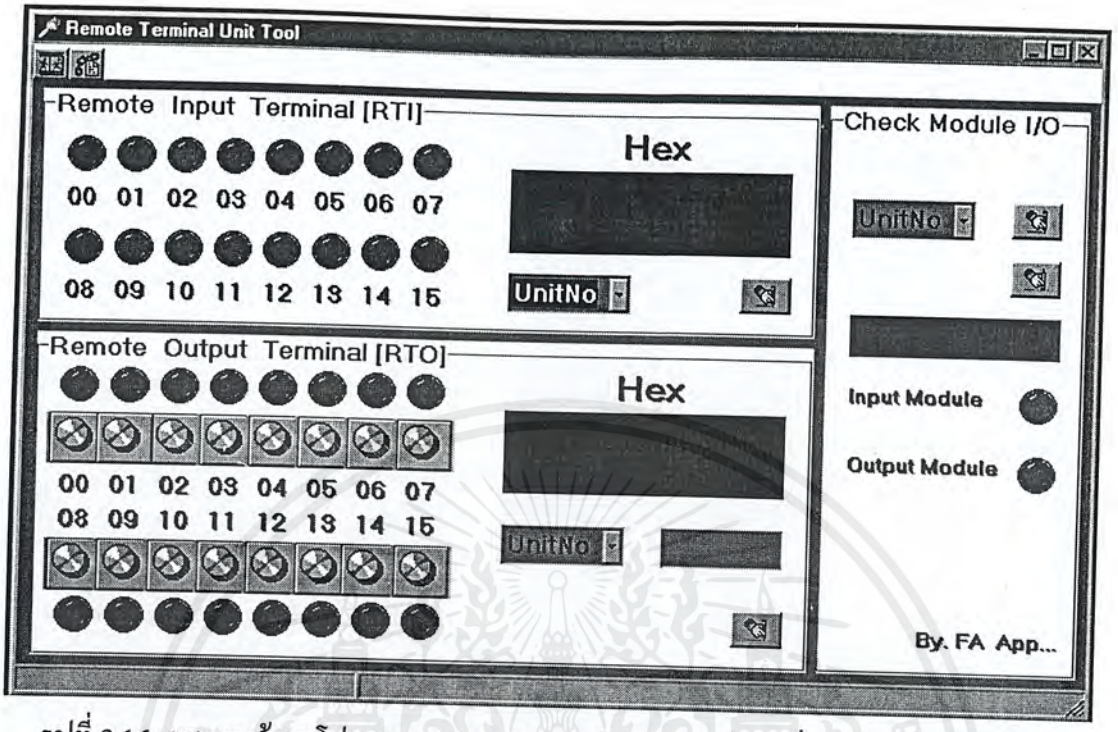



รูปที่ 3.13 แสดงหน้าจอโปแกรมหลัก

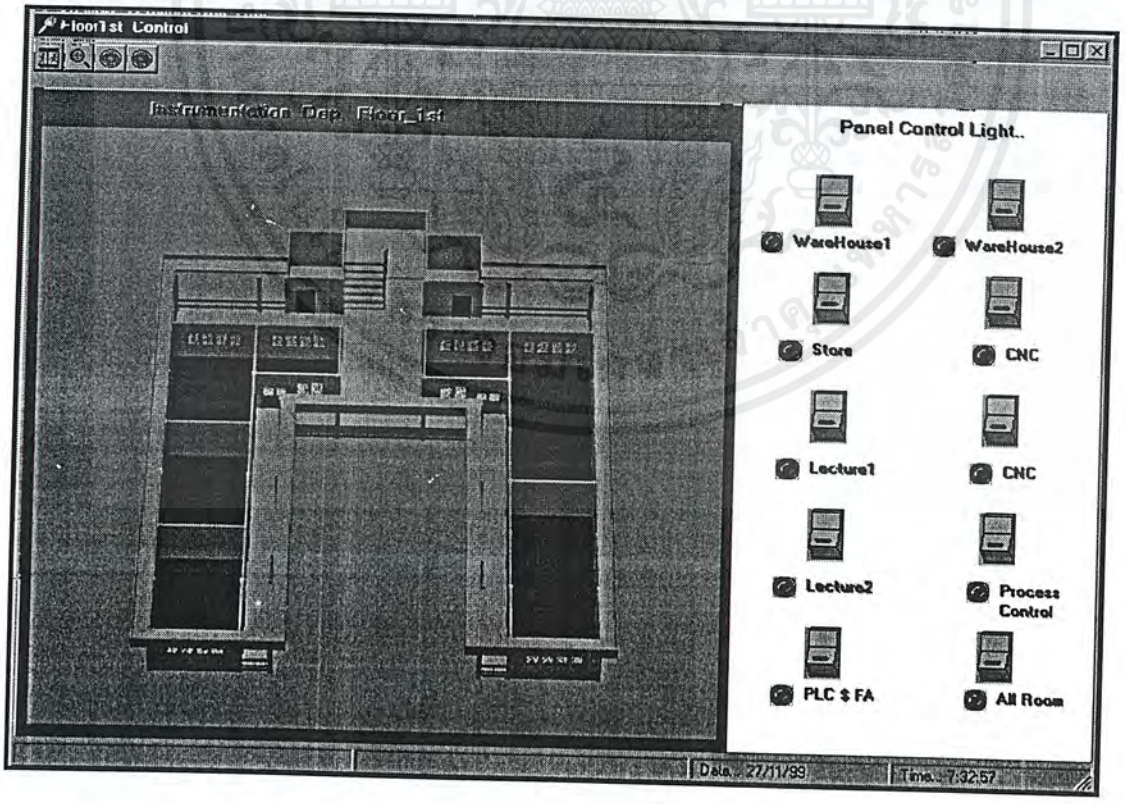

รูปที่ 3.14 แสดงหน้าจอโปรแกรมการติดต่อสื่อสารข้อมูล




รูปที่ 3.15 แสดงหน้าจอการตั้งค่าพารามิเตอร์ของพอร์ตอนุกรม




รูปที่ 3.16 แสดงหน้าจอโปรแกรมการทดสอบชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล




รูปที่ 3.17 แสดงหน้าจอโปรแกรมการควบคุมระบบแสงสว่างของชั้นที่ 1

### 3.4 วิธีการสร้าง Model 3 มิติ

ในการสร้างภาพ 3 มิติมีสิ่งสำคัญในการสร้างภาพ 3 มิติอยู่ 3 ข้อคือ

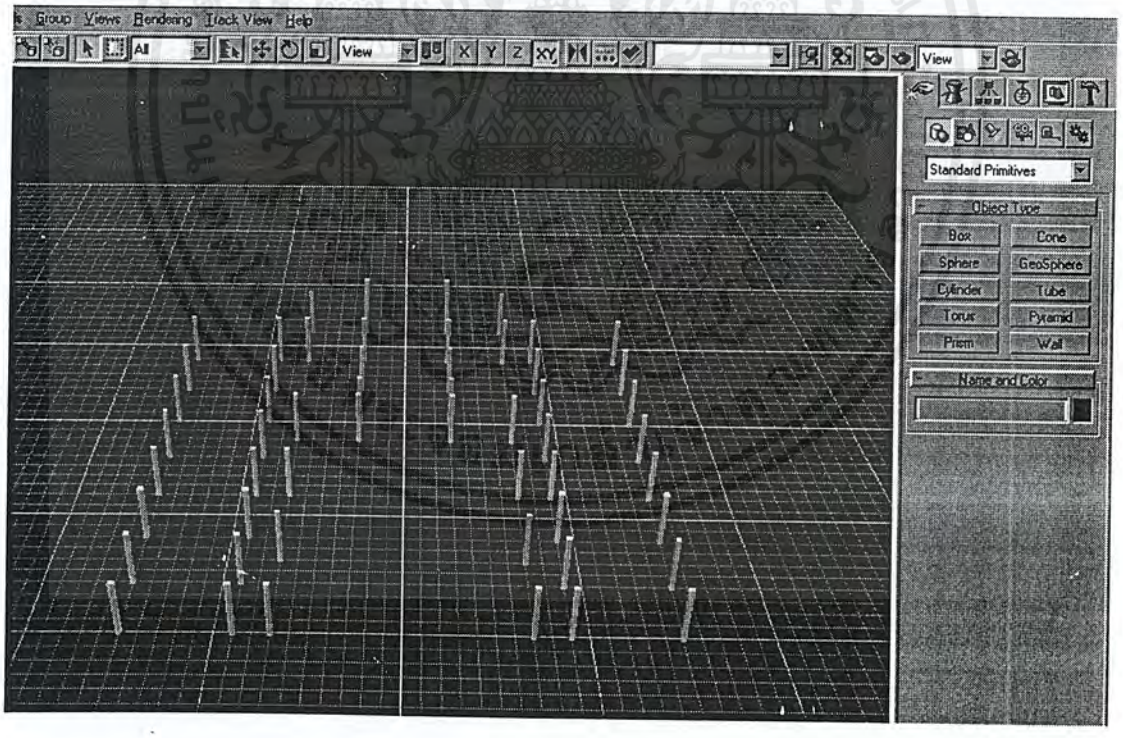
1.ตำแหน่งพิกัด X,Y,Z จะต้องมีความแน่นอนเพื่อทำให้การสร้างภาพ 3 มิติจะสามารถรวมภาพต่างๆที่สร้างขึ้นได้อย่างแม่นยำและถูกต้อง

2.สภาพแวดล้อม(Environment) เป็นสิ่งสำคัญที่จะทำให้ภาพ 3 มิติที่สร้างขึ้นมีความสวยงามและเหมือนจริง

3.พื้นผิววัสดุ(Material Editor) จะเป็นองค์ประกอบที่ทำให้ภาพที่สร้างขึ้นมีพื้นผิวที่เหมือนจริงมากยิ่งขึ้นเช่น การทำผิวของกระจก

ในการสร้าง Model 3 มิติมีขั้นตอนในการสร้างดังต่อไปนี้คือ

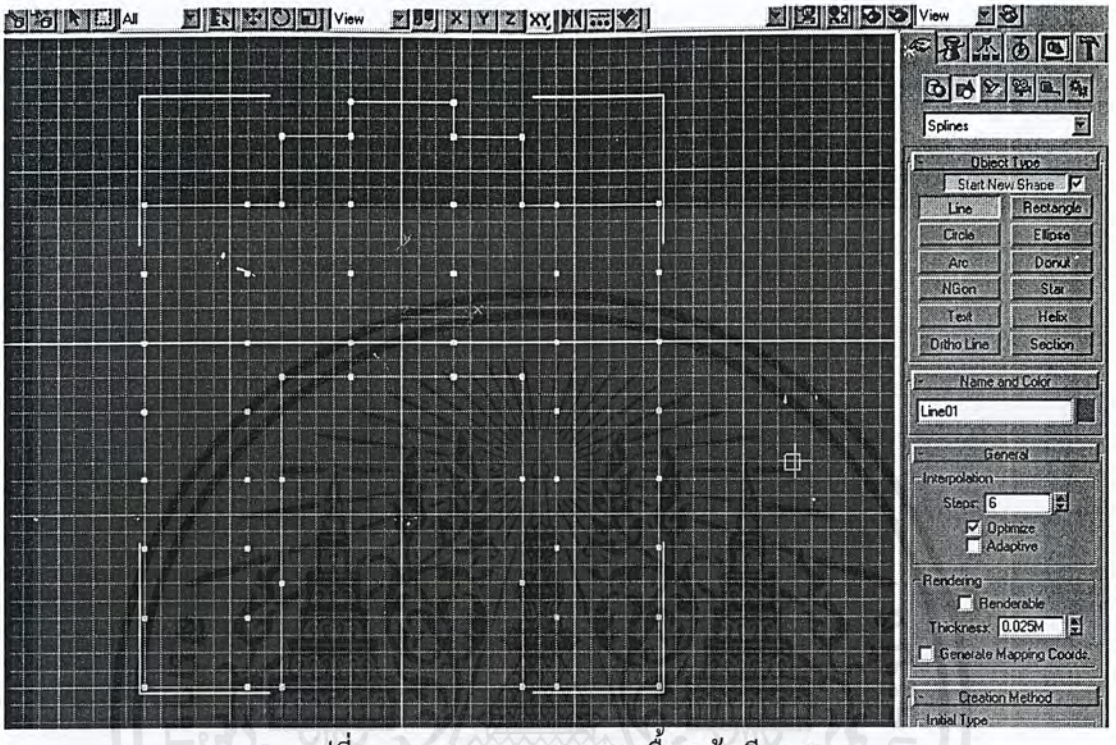
1.สร้างเสาของอาคาร คำสั่งที่ใช้ในการสร้างคือ BOX โดยจำนวนเสาทั้งหมดเท่ากับ 56 ต้น ในการสร้างจะสร้างเพียง 1 ต้น จากนั้นใช้การ Copy อีกจำนวน 55 ต้น ดังรูปที่ 3.18 แสดงการสร้างเสาของอาคาร



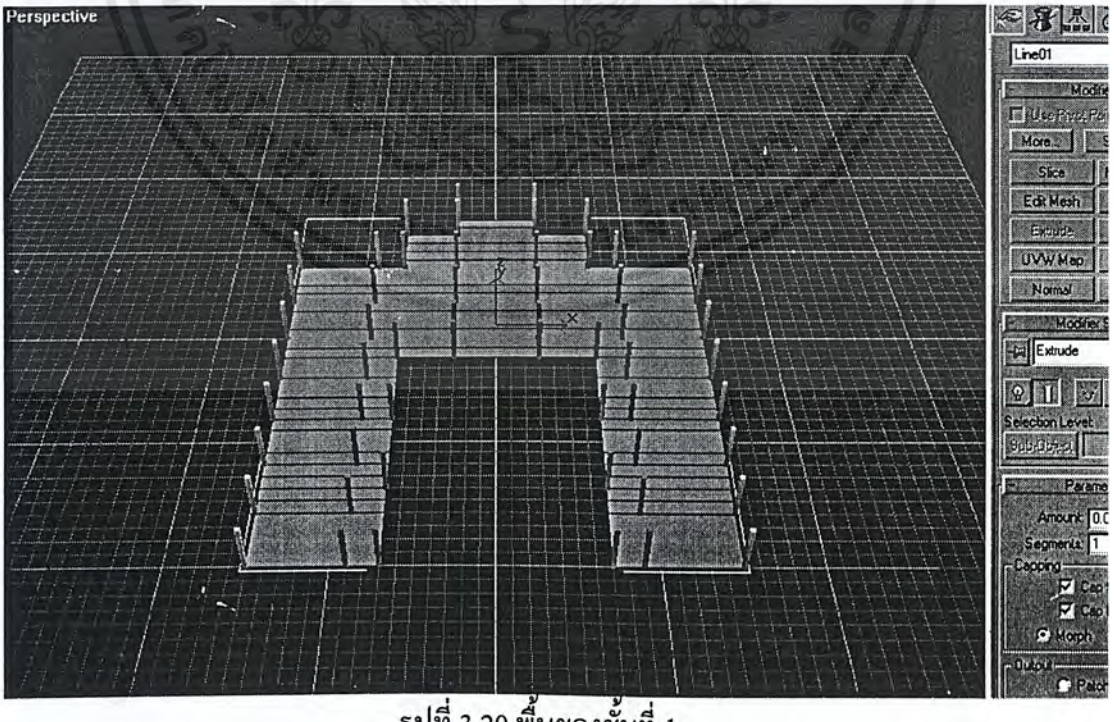

รูปที่ 3.18 การสร้างเสาของอาคาร

2.สร้างพื้นของชั้นที่ 1 ในการสร้างพื้นจะต้องวางขอบเขตในการสร้างโดยใช้คำสั่ง Line เพื่อกำหนดขอบเขต แสดงดังรูปที่ 3.19 หลังจากนั้นจึงใช้คำสั่ง Extrude เพื่อสร้างพื้น

สามารถกำหนดขนาดความหนาของพื้นและเลือกสีของพื้นได้ ในรูปจะใช้ความหนาขนาด 0.3 m และเลือกเป็นสีขาวดังแสดงดังรูปที่ 3.20

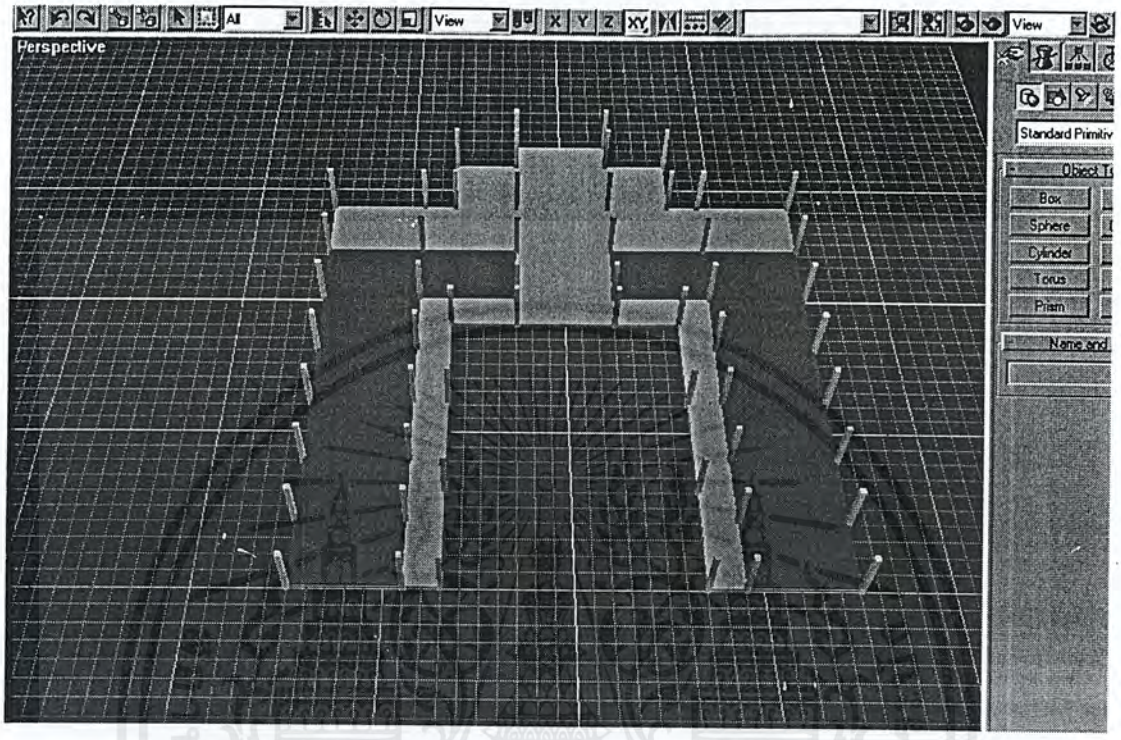



รูปที่ 3.19 การวางขอบเขตของพื้น (เส้นสีแดง)




รูปที่ 3.20 พื้นของชั้นที่ 1

หลังจากนั้นเปลี่ยนสีที่อยู่ในพื้นของห้องทั้งหมด ซึ่งจะใช้สีฟ้าดังรูปที่ 3.21

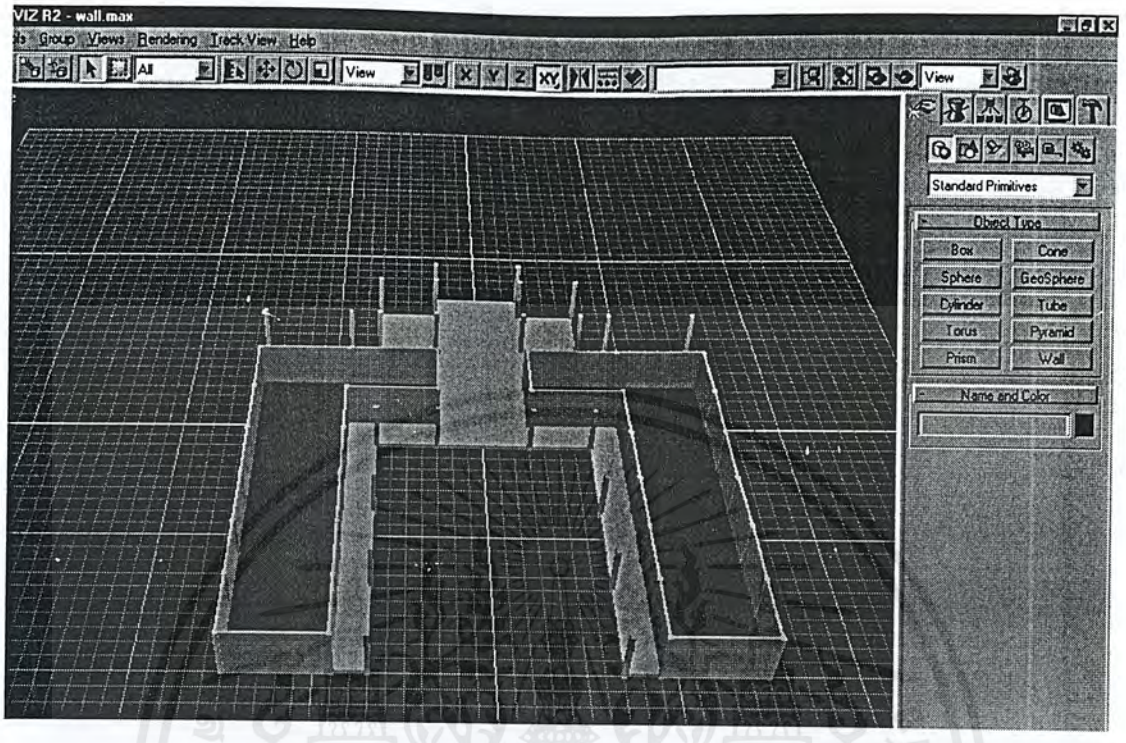



รูปที่ 3.21 สร้างพื้นในห้องทั้งหมด

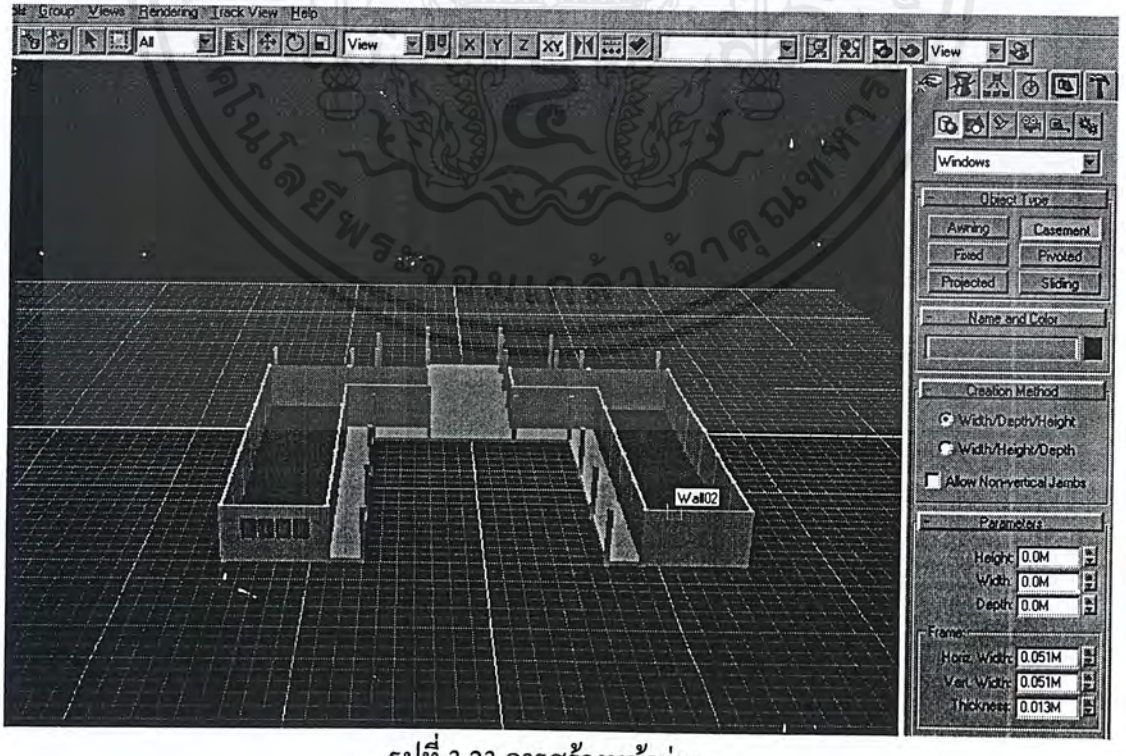
ในการสร้างพื้นในห้องจะทำเหมือนกับพื้นของชั้นที่ 1 ที่มีสีขาว โดยจะสร้างพื้นขึ้นมาอีกเป็นพื้นอันที่สอง แต่เป็นสีฟ้า พื้นอันแรกถูกทับด้วยพื้นอันที่สองอย่างกลมกลืนกัน

3.สร้างกำแพงของชั้นที่ 1 โดยใช้คำสั่ง Wall ในการสร้างจะต้องกำหนดขอบเขตของกำแพงด้วยคำสั่ง Line เหมือนกับการสร้างพื้น หลังจากนั้นจะใช้คำสั่ง Wall จะสามารถกำหนดขนาดความหนาของกำแพงและสีของกำแพงได้ เมื่อกดปุ่มคำสั่ง Wall แล้วจากนั้นเลือกที่ปุ่ม pick spline แล้วคลิกไปที่เส้นขอบเขตที่สร้างไว้ ก็จะได้กำแพงตามต้องการดังรูปที่ 3.21

4.สร้างหน้าต่างโดยเลือกที่คำสั่ง Window ใช้หน้าต่างแบบ Casement เลือกไปที่ปุ่ม Casement จากนั้นวางตำแหน่งของหน้าต่างโดยการกดปุ่มซ้ายของเมาส์ค้างเป็นการกำหนดจุดเริ่มต้นของหน้าต่างลากเมาส์ไปทางขวาปล่อยปุ่มซ้ายของเมาส์เป็นการกำหนดจุดสิ้นสุดของหน้าต่าง หลังจากนั้นลากขึ้นกดปุ่มซ้ายเพื่อกำหนดความหนา สุดท้ายลากขึ้นอีกทีกดปุ่มซ้ายเพื่อกำหนดความสูงก็จะได้หน้าต่าง สามารถปรับความหนาและความสูงได้อีกทีโดยใช้การเปลี่ยนค่าและเปลี่ยนสีของหน้าต่างได้ สามารถปรับการเปิด-ปิดของหน้าต่างได้ว่าจะให้เปิดกี่เปอร์เซนต์และที่สำคัญคือวัสดุของหน้าต่าง สามารถทำให้เป็นกระจกโดยใช้ Material Editor สร้างเพียง 1 บานหลังจากนั้นใช้ copy เพิ่มทั้งหมดการสร้างกระจกแสดงได้ดังรูปที่ 3.22

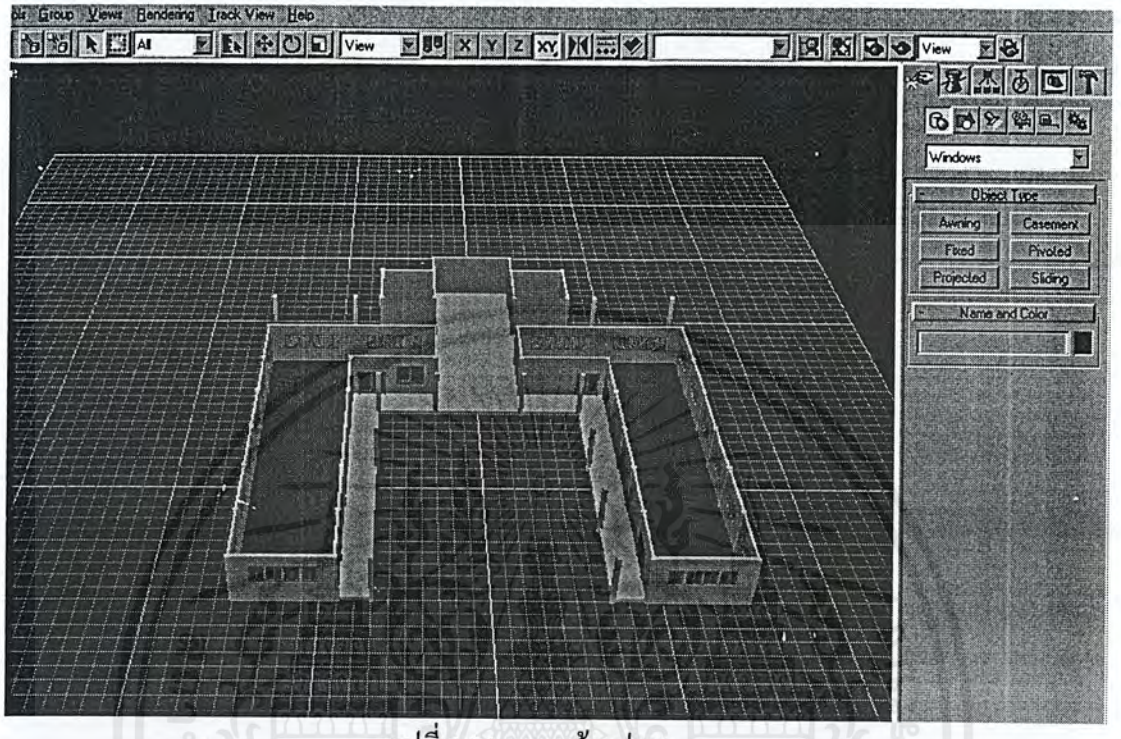



รูปที่ 3.22 การสร้างกำแพง

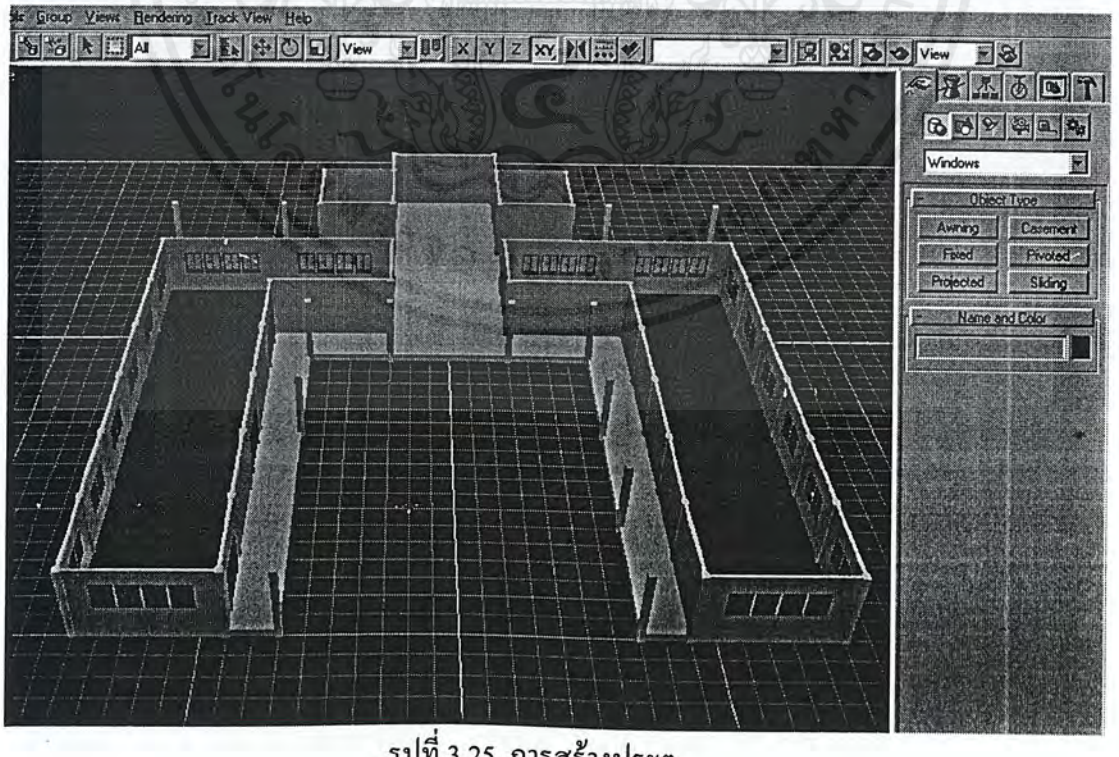



รูปที่ 3.23 การสร้างหน้าต่าง

5.สร้างประตูโดยเลือกที่ Door วิธีการสร้างประตูจะเหมือนกับการสร้างหน้าต่างแสดงดัง รูปที่ 3.23 และ 3.24

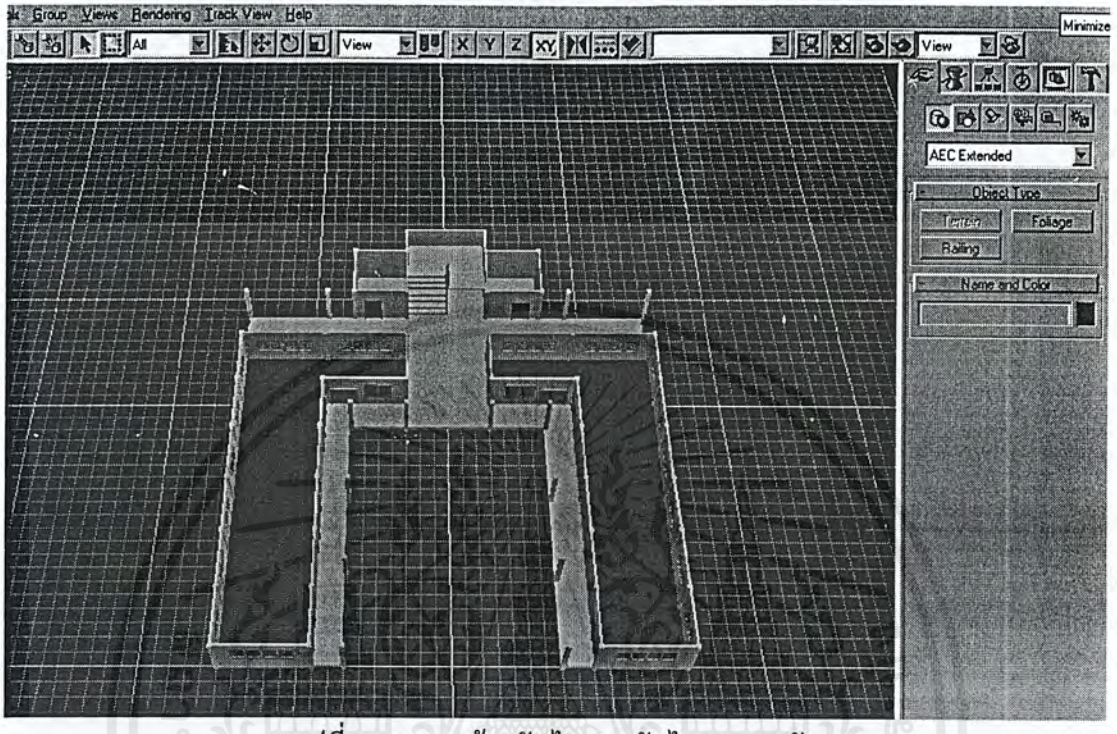



รูปที่ 3.24 การสร้างประตู

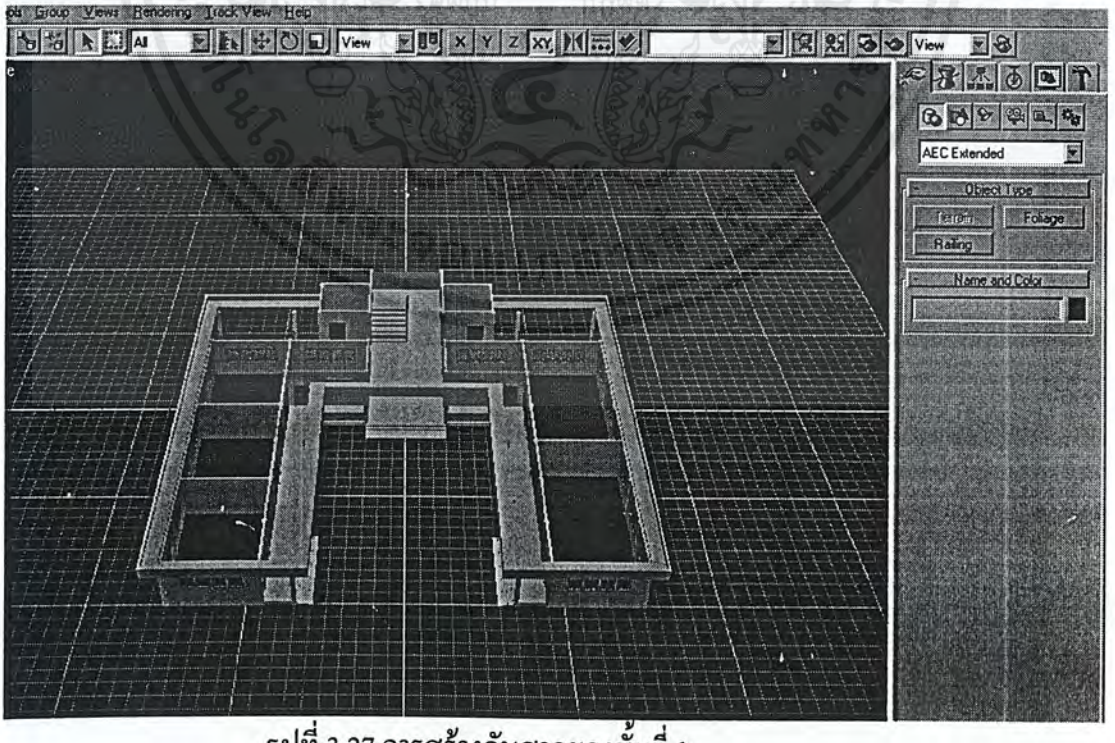



รูปที่ 3.25 การสร้างประตู

6.สร้างบันไดโดยใช้คำสั่ง Stair ใช้ Stair แบบ U-type Stair สร้างราวบันไดโดยใช้คำสั่ง AEC Extended เลือกที่ Railing เพื่อสร้างราวจับและราวบันได ดังรูปที่ 3.25



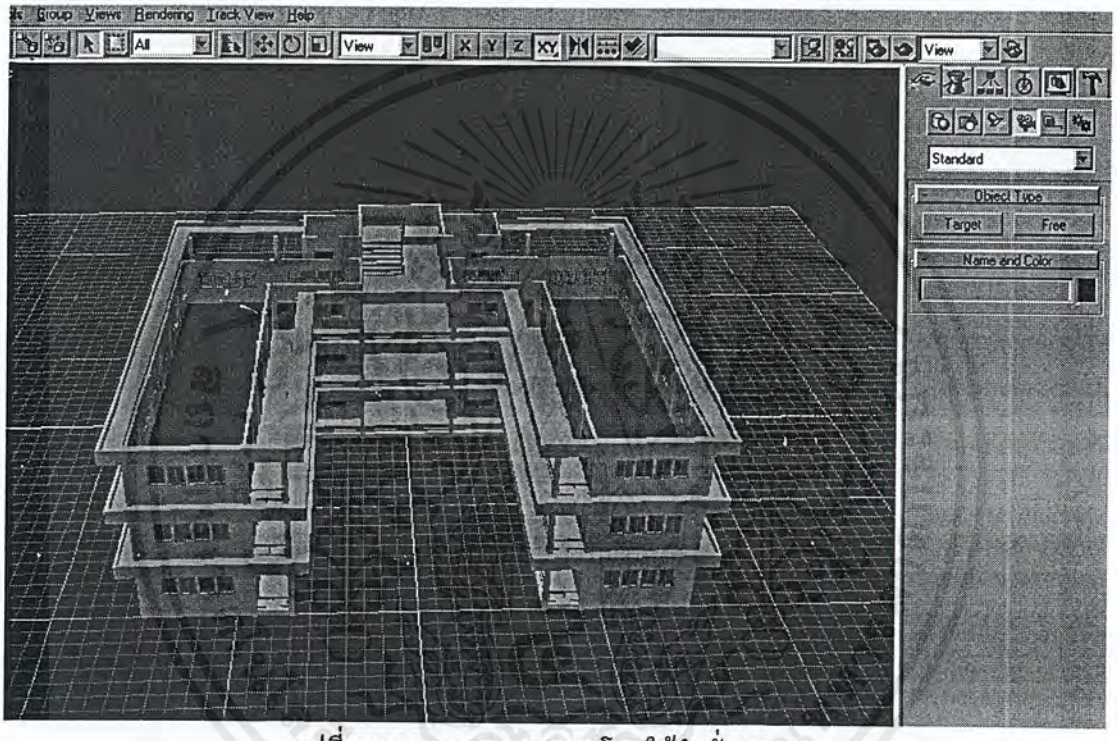

รูปที่ 3.26 การสร้างบันได,ราวบันไดและราวจับ




รูปที่ 3.27 การสร้างกันสาดของชั้นที่ 1

7.สร้างกัดสาดของชั้นที่ 1 โดยการสร้างจะสร้างเหมือนพื้นคือกำหนดขอบเขตของกันสาดแล้วทำให้เป็นพื้นโดยใช้คำสั่ง Extrude จากนั้นเลื่อนตำแหน่งขึ้นตามแกน Z =1.5m ก็จะได้กัดสาดของชั้นที่ 1 แสดงดังรูปที่ 3.26

8.เมื่อสร้างชั้นที่ 1 เสร็จหลังจากนั้นจะใช้คำสั่ง Array เปลี่ยนค่าโดยเลือกให้ Copy ขึ้นตามแกน Z = 1.5m เลือกจำนวนที่จะ Copy ใช้ค่าเท่ากับ 3 หลังจากนั้นกด OK จะทำให้ Model ชั้นที่ 1 ถูก Copy ขึ้นอีก 2 ชั้นรวมของเดิมด้วยเป็น 3 ชั้นดังรูปที่ 3.27




รูปที่ 3.28 แสดงการ Copy โดยใช้คำสั่ง Array

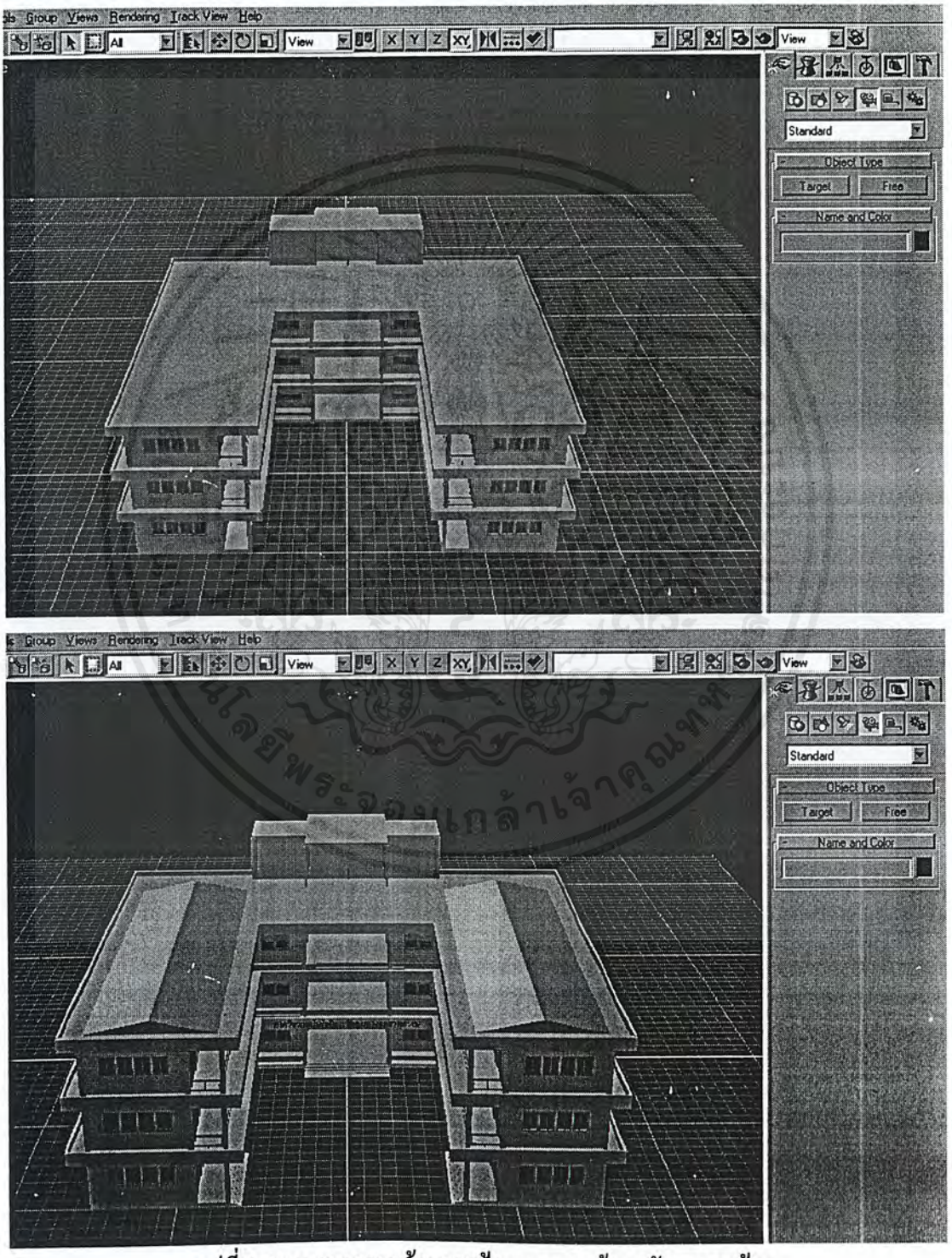
9.สร้างดาดฟ้า การสร้างจะเหมือนกับการสร้างพื้นชั้นที่ 1 หลังจากสร้างเสร็จแล้วจะสั่งให้พื้นดาดฟ้าเลื่อนขึ้นโดยใช้คำสั่ง Move ขึ้นตามแนวแกน Z = 4.5m ดังรูป 3.28

10.สร้างหลังคาดาดฟ้าโดยใช้คำสั่ง Prism แล้วใช้คำสั่ง Move ขึ้นตามแกน Z = 4.5m ดังแสดงในรูปที่ 3.28

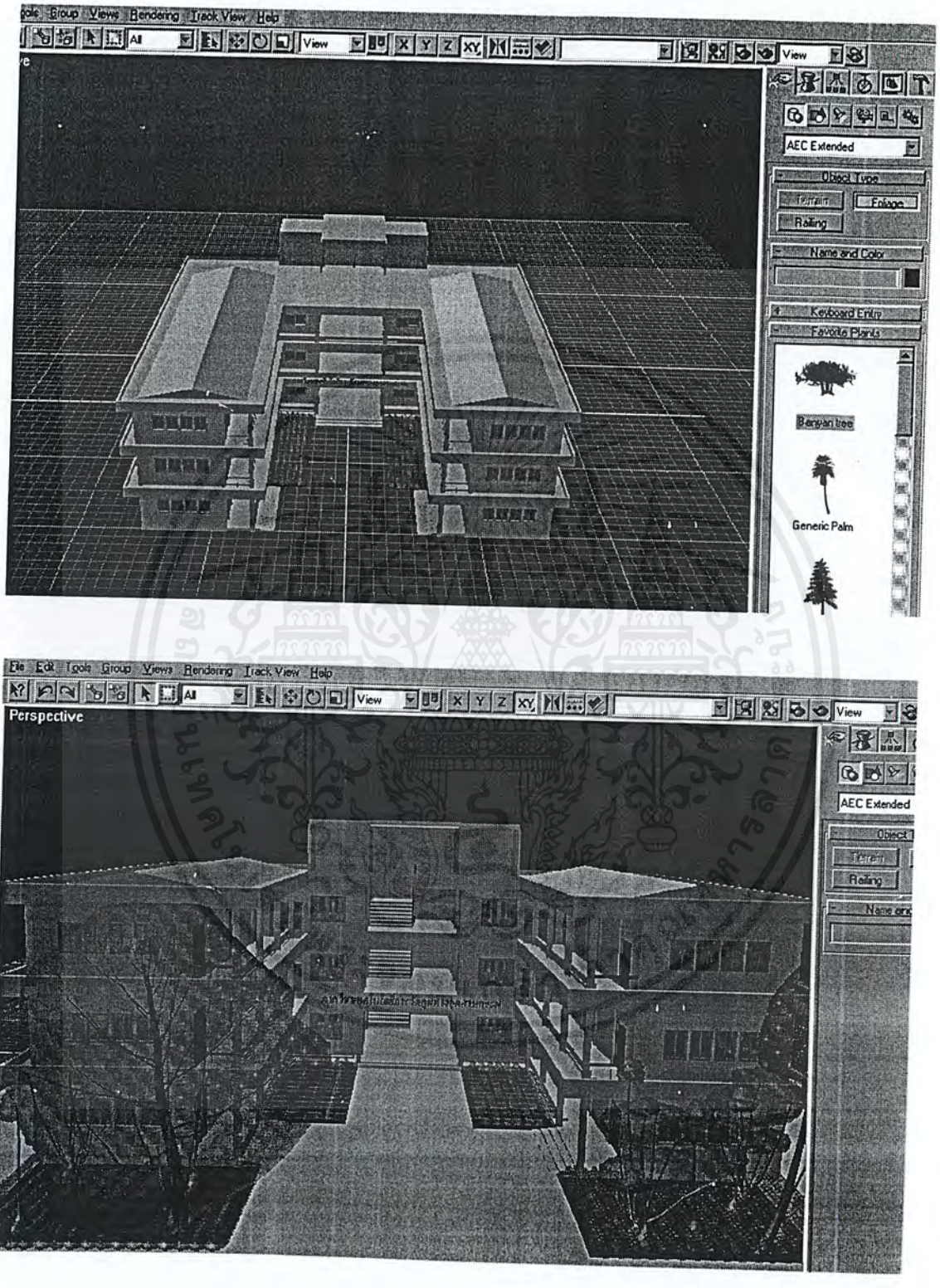
11.สร้างต้นไม้และสนามหญ้าโดยใช้คำสั่ง AEC Extended เลือกที่ Foliage จะมีชนิดของต้นไม้ให้เลือกใช้ โดยการเลือกต้นไม้กดปุ่มซ้ายของเมาส์ค้างไว้แล้วลากมาใส่ยังตำแหน่งที่ต้องการ จะเห็นว่าต้นไม้จะมีลักษณะคล้ายๆวุ้น เพื่อความรวดเร็วในการสร้างภาพ 3 มิติแสดงดังรูปที่ 3.29

12.สร้างภาพหมุนโดยใช้คำสั่ง Camera โดยอันดับแรกจะต้องสร้าง Path ก่อน หลังจากนั้นจึงสร้างกล้อง Camera แล้วสั่งให้กล้องหมุนโดยคลิกไปที่ Path ดังรูปที่ 3.30

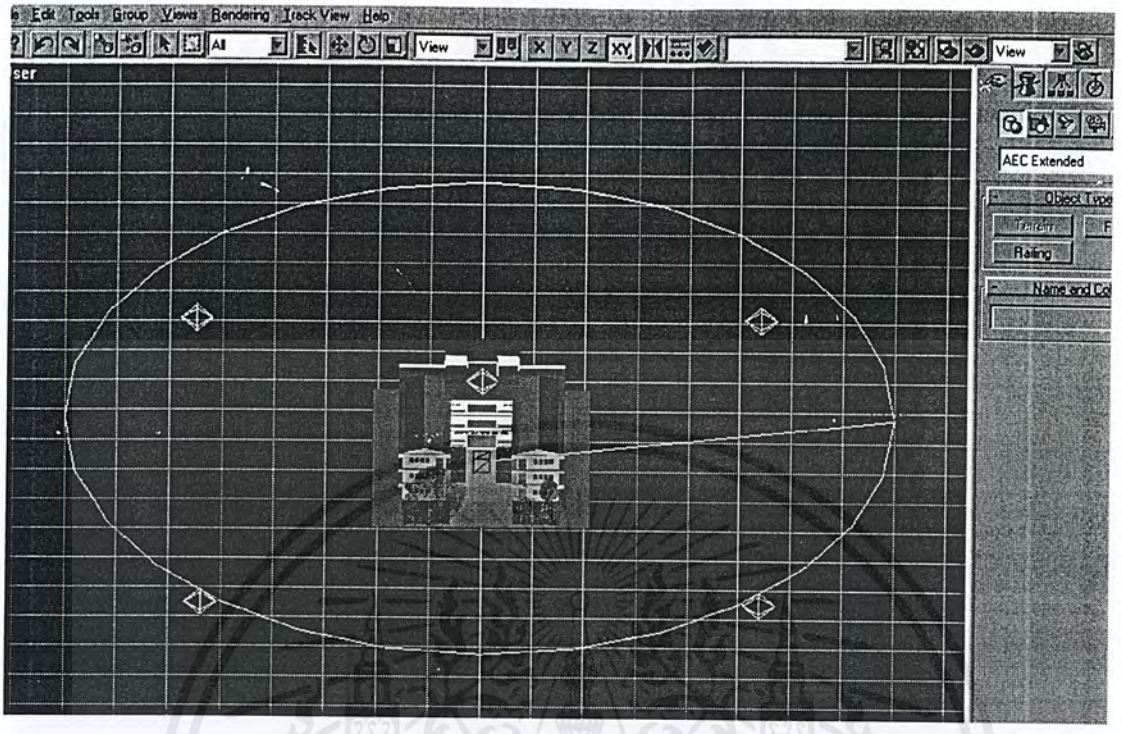
13.เมื่อสร้างภาพเสร็จขั้นตอนสุดท้ายคือ การประมวลผลภาพ(Render) คลิกที่ Rendering เลือก Render จะมีหน้าต่างปรากฏขึ้น สามารถปรับจำนวนเฟรมที่จะต้องการ Render เลือกขนาดของภาพที่ต้องการ เลือกรูปแบบในการ Render และสิ่งสำคัญที่สุดคือการเลือกนามสกุลไฟล์ที่ต้องการเมื่อ Render เสร็จ



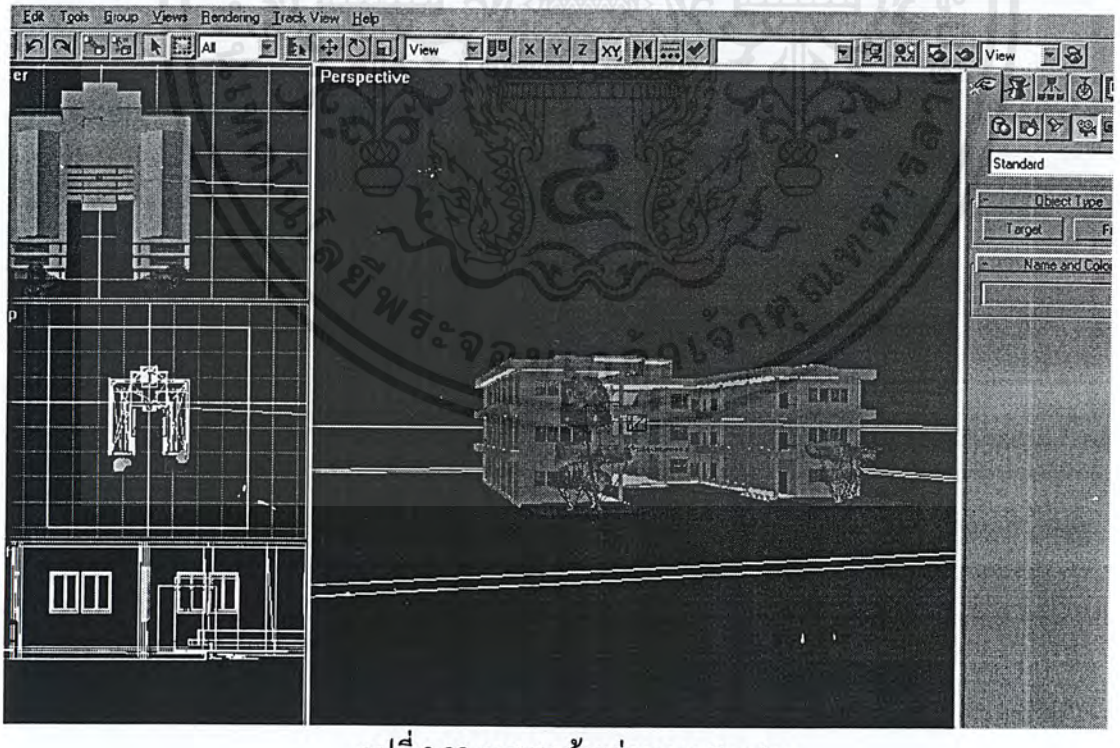

รูปที่ 3.29 แสดงการสร้างดาดฟ้าและการสร้างหลังคาดาดฟ้า




รูปที่ 3.30 การสร้างสนามหญ้าและการสร้างต้นไม้




รูปที่ 3.31 การสร้างภาพหมุนโดยใช้ Camera




รูปที่ 3.32 ภาพสุดท้ายก่อนการ Render

# บทที่ 4

## การทดลองและผลการทดลอง

ในการทดลองชุด Remote Terminal Input (RTI) และ Remote Terminal Output (RTO) จะต้องปรับแต่ง Dip Switch เพื่อกำหนด Baud Rate, กำหนดการทำงานว่าจะให้ทำงานเป็น RTI หรือ RTO และสุดท้ายเพื่อกำหนด Unit Number ให้กับ Remote Terminal Unit รูปแบบของ Dip Switch จะเป็นดังนี้

| SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 | SW8 |
|---|---|---|---|---|---|---|---|
| OFF | OFF | OFF | OFF | OFF | OFF | OFF | OFF |

- SW1,SW2 : กำหนดความเร็วอัตราบอร์ด (Baud Rate) ในการทดลองจะกำหนดความเร็วอัตราบอร์ด (Baud Rate) เท่ากับ 9600 bps คือ SW1 และ SW2 'ON' ทั้งคู่
- SW3,SW4 : กำหนดการทำงานของ Remote Terminal Unit
  - SW3 'ON' และ SW4 'OFF' เป็นการกำหนดให้เป็น RTI
  - SW3 'OFF' และ SW4 'OFF' เป็นการกำหนดให้เป็น RTO
- SW5,SW6,SW7,SW8 : กำหนด Unit Number ให้กับ Remote Terminal Unit
  - SW5 'OFF', SW6 '0FF', SW7 'OFF', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '00'
  - SW5 'OFF', SW6 '0FF', SW7 'OFF', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '01'
  - SW5 'OFF', SW6 '0FF', SW7 'ON', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '02'
  - SW5 'OFF', SW6 '0FF', SW7 'ON', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '03'
  - SW5 'OFF', SW6 '0N', SW7 'OFF', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '04'
  - SW5 'OFF', SW6 '0N', SW7 'OFF', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '05'

- SW5 'OFF', SW6 '0N', SW7 'ON', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '06'
- SW5 'OFF', SW6 '0N', SW7 'ON', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '07'
- SW5 'ON', SW6 '0FF', SW7 'OFF', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '08'
- SW5 'ON', SW6 '0FF', SW7 'OFF', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '09'
- SW5 'ON', SW6 '0FF', SW7 'ON', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '10'
- SW5 'ON', SW6 '0FF', SW7 'ON', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '11'
- SW5 'ON', SW6 '0N', SW7 'OFF', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '12'
- SW5 'ON', SW6 '0N', SW7 'OFF', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '13'
- SW5 'ON', SW6 '0N', SW7 'ON', SW8 'OFF' กำหนดให้เป็น Unit Number ที่ '14'
- SW5 'ON', SW6 '0N', SW7 'ON', SW8 'ON' กำหนดให้เป็น Unit Number ที่ '15'

รวมทั้งหมดจะสามารถต่อ Remote Terminal Unit ได้ทั้งหมด 16 ชุด

ในการทดลองจะมีการปรับแต่ง Dip Switch อยู่ 4 รูปแแบบ ดังนี้

| SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 | SW8 |
|---|---|---|---|---|---|---|---|
| ON | ON | OFF | OFF | OFF | OFF | OFF | OFF |

:ซึ่งจะได้ตำแหน่งของชุดอินพุทเพื่อการควบคุมระยะไกล (RTO) เป็น Unit Number 00 และความเร็วอัตราบอร์ด (Baud Rate) เป็น 9600 bps

| SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 | SW8 |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ON | ON | OFF | OFF | OFF | OFF | OFF | ON |

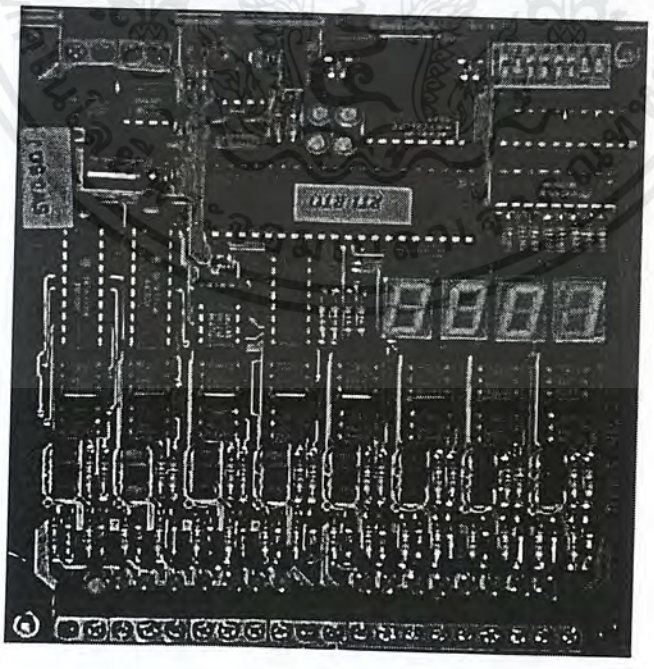
:ซึ่งจะได้ตำแหน่งของชุดอินพุทเพื่อการควบคุมระยะไกล (RTO) เป็น Unit Number 01 และความเร็วอัตราบอร์ด (Baud Rate) เป็น 9600 bps

| SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 | SW8 |
|---|---|---|---|---|---|---|---|
| ON | ON | OFF | OFF | OFF | OFF | ON | OFF |

:ซึ่งจะได้ตำแหน่งของชุดอินพุทเพื่อการควบคุมระยะไกล (RTO) เป็น Unit Number 02 และความเร็วอัตราบอร์ด (Baud Rate) เป็น 9600 bps

| SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 | SW8 |
|---|---|---|---|---|---|---|---|
| ON | ON | ON | OFF | OFF | OFF | ON | ON |

:ซึ่งจะได้ตำแหน่งของชุดอินพุทเพื่อการควบคุมระยะไกล (RTI) เป็น Unit Number 03 และความเร็วอัตราบอร์ด (Baud Rate) เป็น 9600 bps



รูปที่ 4.1 แสดงรูป RTO Unit 03 กำหนดอัตราบอร์ดเท่ากับ 9600

**ลักษณะการเชื่อมต่อ**

RTI/RTO จะมีไฟเลี้ยง +24 โวลท์ และต่อสาย RS-485 ที่ต่อออกมาจากตัว Convert เพียง 2 เส้น ซึ่งสามารถต่อขนานกันทั้ง 3 ตัว

ใน Project นี้จะนำ O/P ของ RTO Unit 02 มาแทน Sensor ที่จุดต่างๆใน Application จริงๆโดยต่อ O/P ของ RTO Unit 00 ต่อเข้าที่ I/P ของ RTI Unit 03 ดังนั้น Display ที่แสดงทาง 7-Segment ของทั้ง 2 ตัวจะมีค่า Address เดียวกัน

สำหรับ RTO Unit 01 จะนำไปควบคุมระบบแสงสว่างใน Building Model โดยจะควบคุมการปิด-เปิดหลอดไฟด้วย Computer ซึ่งในการทดลองจะต้องป้อน Command เพื่อใช้ในการควบคุมดังนี้

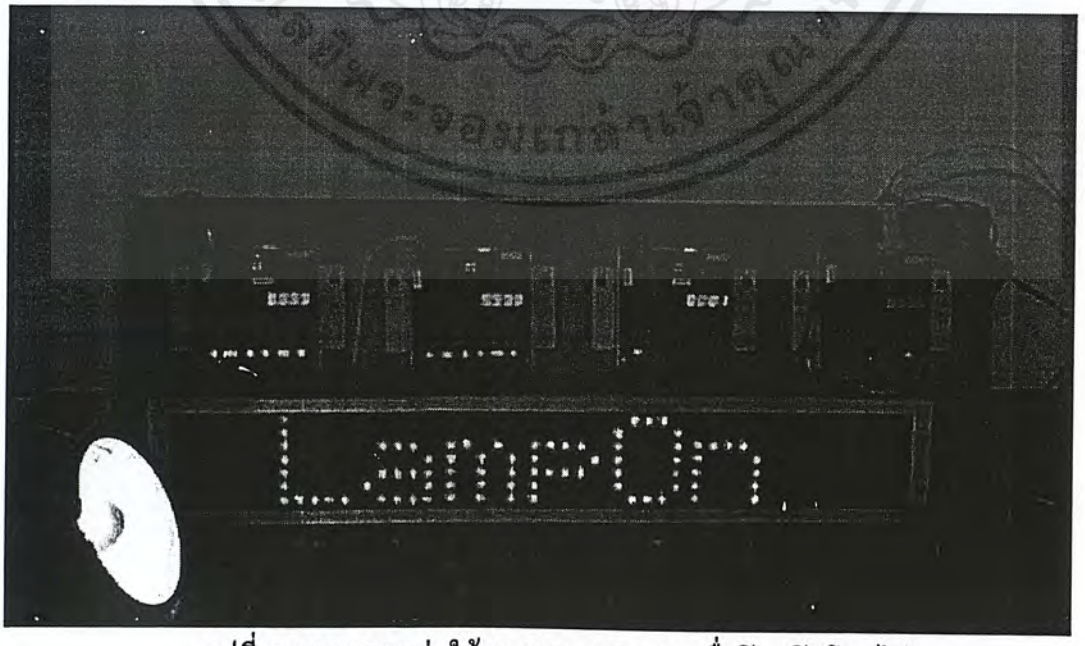
@01WR DATA*XX

ตัวอย่างเช่น

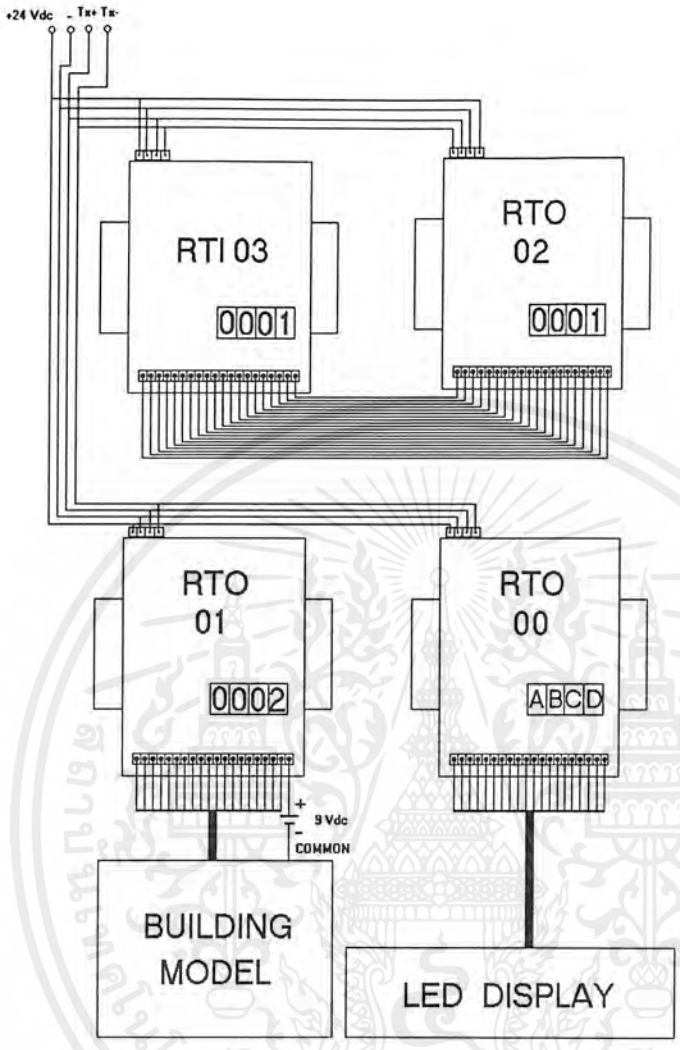
@01WR0001*XX

:จะทำให้ 7-Segment จะทำให้ RTO Unit 01 แสดงค่า 0001 มีผลทำให้หลอดไฟของ Building Model ที่ต่อกับ Bit ที่ 1 ของ RTO Unit 01 ติดสว่าง

สำหรับ RTO Unit 00 จะนำไปควบคุมการแสดงผลทาง LED Display Board โดยอักษรที่แสดงออกทาง LED Display Board จะแสดงด้วยไดโอดชนิดเปล่งแสงแบบสว่างมาก จัดเรียงขึ้นเป็นแมทริกขนาด 5 x 7 สามารถแสดงได้ทั้งตัวใหญ่และตัวเล็ก ตลอดจนเครื่องหมายที่จำเป็นได้อย่างครบถ้วน



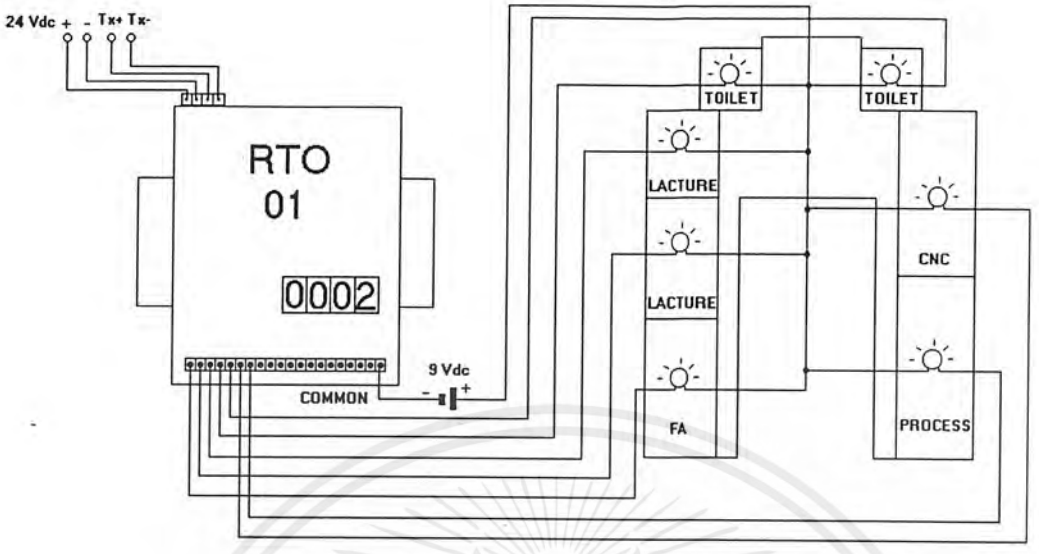
**รูปที่ 4.2 แสดงการต่อใช้งานของ RTO/RTI เพื่อปิด-เปิดโคมไฟ**




รูปที่ 4.3 แสดงลักษณะต่อใช้งานของ RTI/RTO กับ Building Model

ในการนำไปควบคุมจริงๆ จะใช้ Sensor ที่จุดต่างในอาคารต่อกับ RTI เพื่อแจ้งสภาวะ ON-OFF ของหลอดไฟแก่คอมพิวเตอร์จากนั้นคอมพิวเตอร์จะควบคุม RTO อีกทีหนึ่ง

RTO Unit 01 จะนำไปควบคุมการ ON-OFF หลอดไฟชั้นที่ 1 ซึ่งจะมีทั้งหมด 7 ห้อง แต่ละห้องจะต่อกับ RTO ซึ่งแต่ละห้องจะต่อกับ RTO Unit 01 เหมือนกัน แต่บิตจะต่างกันดังรูปการต่อใช้งาน RTO Unit 01 ควบคุมการ ON-OFF หลอดไฟชั้นที่ 1




รูปที่ 4.4 แสดงการต่อใช้งาน RTO Unit 01 ควบคุมการ ON-OFF หลอดไฟชั้นที่ 1

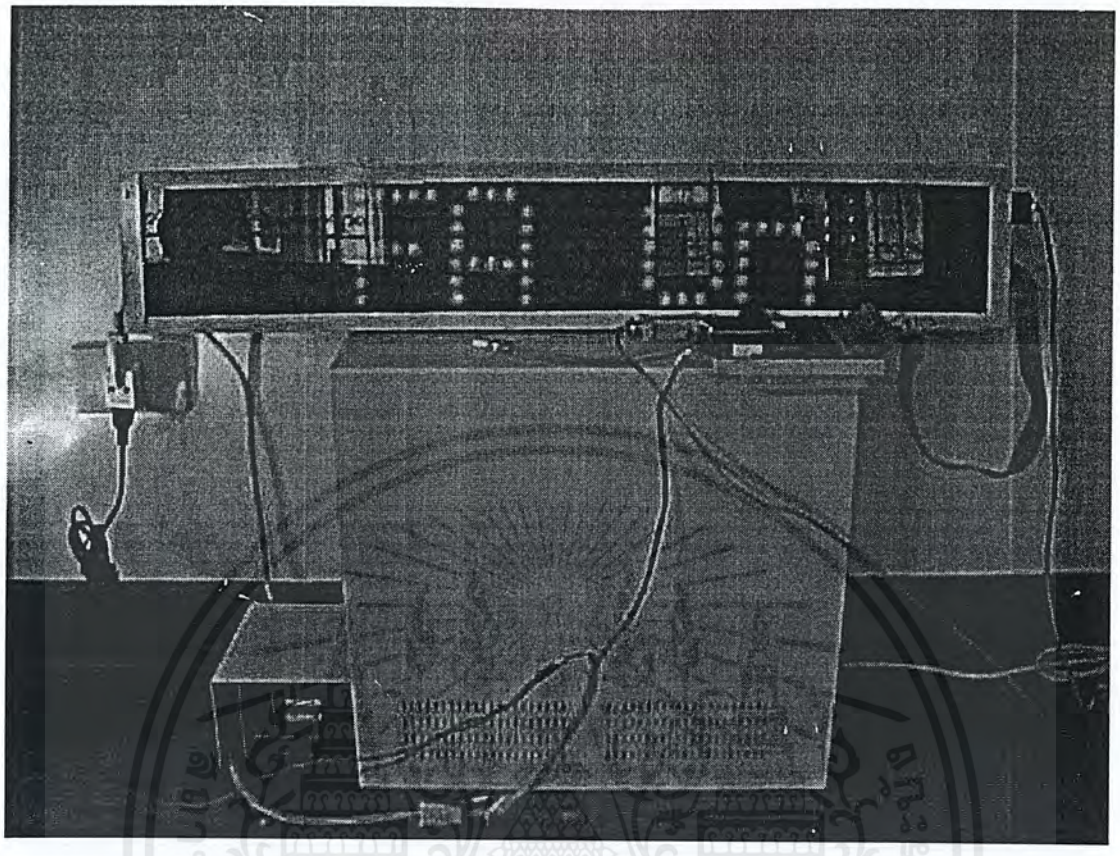
สำหรับ Building Model จะต้องต่อ Common (+) และ RTO จะต้องต่อ Common (-) เพราะฉะนั้นจะต้องต่อ Common โดยนำแหล่งจ่ายไฟ DC 9 V ต่อดังรูป

รูปแบบของ Command ที่ใช้ในการควบคุมจะต้องป้อน DATA ดังนี้

@01WR **DATA** *XX

| DATA | ROOM |
|---|---|
| 0001 | FA |
| 0002 | LACTURE |
| 0004 | LACTURE |
| 0008 | TOILET |
| 0010 | TOILET |
| 0020 | CNC |
| 0040 | PROCESS |

ตารางที่ 4.1 แสดง DATA ที่ใช้ในการควบคุมห้องต่างๆในชั้นที่ 1



รูปที่ 4.5 แสดงการนำ RTO Unit 03 มาควบคุมห้อง FA

# บทที่ 5
# สรุปและวิจารณ์

จากโครงสร้างของระบบควบคุมโรงงานหรืออาคารอัตโนมัติที่จำเป็นต้องใช้ชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminal Input/Output) ดังนั้นปริญญานิพนธ์ฉบับนี้จึงทำการพัฒนาความสามารถของชุดเอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminal Output) โดยการเขียนโปรแกรมจัดระบบงาน(Operating System) เพื่อให้สามารถแสดงผลออกทางหน่วยแสดงผลแบบตัวอักษรจำนวน 8 หน่วยโดยการเพิ่มชุดคำสั่งของการติดต่อสื่อสารข้อมูลให้กับชุดเอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminal Output) ตามรูปแบบข้อตกลงการสื่อสารข้อมูลตามที่กล่าวในบทที่ 3 ในส่วนของโปรแกรมประยุกต์ได้ทำการเขียนโปรแกรมประยุกต์โดยวัตถุประสงค์เพื่อใช้งานเฉพาะอย่างโดยแบ่งออกได้เป็น

โปรแกรมประยุกต์เพื่อใช้ในการติดต่อสื่อสารข้อมูลแบบอนุกรมระหว่างชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminal Input/Output) กับคอมพิวเตอร์(Computer) ซึ่งสามารถติดต่อสื่อสารข้อมูลอนุกรมทั้งในแบบมาตรฐาน RS-232 และแบบมาตรฐาน RS-485 โดยสามารถติดต่อกับอุปกรณ์ภายนอกชนิดอื่นๆได้โดยตั้งค่าพารามิเตอร์ในการติดต่อสื่อสารข้อมูลแบบอนุกรมให้เหมาะสมกับอุปกรณ์นั้นๆ

โปรแกรมประยุกต์เพื่อใช้ในการตรวจสอบการทำงานของชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminel Input/Output) ซึ่งสามารถตรวจสอบการทำงานของชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกล(Remote Terminel Input/Output) ได้ถึงจำนวน 16 ตำแหน่ง และตรวจสอบเอาท์พุท/อินพุท ต่อชุดอินพุท/เอาท์พุท เพื่อการควบคุมระยะไกลต่อ 1 ตำแหน่งได้ 16 จุด

โปรแกรมประยุกต์เพื่อออกคำสั่งเปิดและปิดไฟฟ้าแสงสว่างของแบบจำลองภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม โดยทำการสร้างภาพ 3 มิติของแบบจำลองให้แสดงในโปรแกรม

ซึ่งในแต่ละโปรแกรมที่กล่าวถึงยังสามารถนำไปพัฒนากับใช้ในระบบควบคุมโรงงานหรืออาคารอัตโนมัติอื่นๆได้อีก

# ภาคผนวก

```
unit MainMenu;
interface
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  Std2,
    StdCtrls, Mystd, Zseg, ToolWin, ComCtrls, ExtCtrls,Buttons,zAnimate,Menus;
type
        TFormMainMenu = class(TForm)
        StatusBar1: TStatusBar;
        zcDotLabel1: TzcDotLabel;
        GroupBox1: TGroupBox;
        GroupBox2: TGroupBox;
        z3DLabel1: Tz3DLabel;
        zcTimerText: TzcTimer;
        zBitColBtn1: TzBitColBtn;
        zBitColBtn2: TzBitColBtn;
        zBitColBtn3: TzBitColBtn;
        zBitColBtn4: TzBitColBtn;
        zBitColBtn5: TzBitColBtn;
        z3DLabel2: Tz3DLabel;
        z3DLabel3: Tz3DLabel;
        z3DLabel4: Tz3DLabel;
        Image1: TImage;
        ToolBar1: TToolBar;
        procedure FormCreate(Sender: TObject);
        procedure zcTimerTextTimer(Sender: TObject);
        procedure zBitColBtn1Click(Sender: TObject);
        procedure zBitColBtn2Click(Sender: TObject);
        procedure zBitColBtn3Click(Sender: TObject);
        procedure zBitColBtn5Click(Sender: TObject);
        procedure zBitColBtn4Click(Sender: TObject);
private
        { Private declarations }
```

```
  public
        { Public declarations }
  end;
  var
        FormMainMenu: TFormMainMenu;
        p: Integer;
  implementation
  uses RtioZc2, Commu, Floor1, Floor3, Floor2;
  {$R *.DFM}
  procedure TFormMainMenu.FormCreate(Sender: TObject);
  begin
        p:= 1;
  end;
  procedure TFormMainMenu.zcTimerTextTimer(Sender: TObject);
  begin
  Case p of
    1:begin zcDotLabel1.Caption := '  INSTRUMENTATION DEPARTMENT '
  ;p:=p+1;end;
    2:begin zcDotLabel1.Caption := ' INSTRUMENTATION DEPARTMENT *'
  ;p:=p+1;end;
    3:begin zcDotLabel1.Caption := 'INSTRUMENTATION DEPARTMENT **'
  ;p:=p+1;end;
    4:begin zcDotLabel1.Caption := 'NSTRUMENTATION DEPARTMENT **I'
  ;p:=p+1;end;
    5:begin zcDotLabel1.Caption := 'STRUMENTATION DEPARTMENT **IN'
  ;p:=p+1;end;
    6:begin zcDotLabel1.Caption := 'TRUMENTATION DEPARTMENT **INS'
  ;p:=p+1;end;
    7:begin zcDotLabel1.Caption := 'RUMENTATION DEPARTMENT **INST'
  ;p:=p+1;end;
    8:begin zcDotLabel1.Caption := 'UMENTATION DEPARTMENT **INSTR'
  ;p:=p+1;end;
```

```
  9:begin zcDotLabel1.Caption :=  'MENTATION DEPARTMENT **INSTRU'
;p:=p+1;end;
  10:begin zcDotLabel1.Caption := 'ENTATION DEPARTMENT **INSTRUM'
;p:=p+1;end;
  11:begin zcDotLabel1.Caption := 'NTATION DEPARTMENT **INSTRUME'
;p:=p+1;end;
  12:begin zcDotLabel1.Caption := 'TATION DEPARTMENT **INSTRUMEN'
;p:=p+1;end;
  13:begin zcDotLabel1.Caption := 'ATION DEPARTMENT **INSTRUMENT'
;p:=p+1;end;
  14:begin zcDotLabel1.Caption := 'TION DEPARTMENT **INSTRUMENTA'
;p:=p+1;end;
  15:begin zcDotLabel1.Caption := 'ION DEPARTMENT **INSTRUMENTAT'
;p:=p+1;end;
  16:begin zcDotLabel1.Caption := 'ON DEPARTMENT **INSTRUMENTATI'
;p:=p+1;end;
  17:begin zcDotLabel1.Caption := 'N DEPARTMENT **INSTRUMENTATIO'
;p:=p+1;end;
  18:begin zcDotLabel1.Caption := ' DEPARTMENT **INSTRUMENTATION'
;p:=p+1;end;
  19:begin zcDotLabel1.Caption := 'DEPARTMENT **INSTRUMENTATION '
;p:=p+1;end;
  20:begin zcDotLabel1.Caption := 'EPARTMENT **INSTRUMENTATION D'
;p:=p+1;end;
  21:begin zcDotLabel1.Caption := 'PARTMENT **INSTRUMENTATION DE'
;p:=p+1;end;
  22:begin zcDotLabel1.Caption := 'ARTMENT **INSTRUMENTATION DEP'
;p:=p+1;end;
  23:begin zcDotLabel1.Caption := 'RTMENT **INSTRUMENTATION DEPA'
;p:=p+1;end;
  24:begin zcDotLabel1.Caption := 'TMENT **INSTRUMENTATION DEPAR'
;p:=p+1;end;
```

```
    25:begin zcDotLabel1.Caption := 'MENT **INSTRUMENTATION DEPART'
  ;p:=p+1;end;
    26:begin zcDotLabel1.Caption := 'ENT **INSTRUMENTATION DEPARTM'
  ;p:=p+1;end;
    27:begin zcDotLabel1.Caption := 'NT **INSTRUMENTATION DEPARTME'
  ;p:=p+1;end;
    28:begin zcDotLabel1.Caption := 'T **INSTRUMENTATION DEPARTMEN'
  ;p:=p+1;end;
    29:begin zcDotLabel1.Caption := ' **INSTRUMENTATION DEPARTMENT'
  ;p:=p+1;end;
    30:begin zcDotLabel1.Caption := '**INSTRUMENTATION DEPARTMENT*'
  ;p:=p+1;end;
    31:begin zcDotLabel1.Caption := '                    ' ;p:=p+1;end;
    32:begin zcDotLabel1.Caption := '**INSTRUMENTATION DEPARTMENT*'
  ;p:=p+1;end;
    33:begin zcDotLabel1.Caption := '                    ' ;p:=p+1;end;
    34:begin zcDotLabel1.Caption := '**INSTRUMENTATION DEPARTMENT*'
  ;p:=p+1;end;
    35:begin zcDotLabel1.Caption := '                    ' ;p:=1;end;
    end;
  end;
  procedure TFormMainMenu.zBitColBtn1Click(Sender: TObject);
  begin
          FormCommu.show;
          FormRtioZc.hide;
          FormFloor1st.hide;
          Floor2nd.hide;
          FormFloor3rd.Hide;
  end;

  procedure TFormMainMenu.zBitColBtn2Click(Sender: TObject);
  begin
```

```
        FormRtioZc.show;
        FormFloor1st.hide;
        Floor2nd.hide;
        FormFloor3rd.Hide;
        FormCommu.hide;
end;
procedure TFormMainMenu.zBitColBtn3Click(Sender: TObject);
begin
        FormFloor1st.show;
        Floor2nd.Hide;
        FormFloor3rd.Hide;
        FormRtioZc.hide;
        FormCommu.hide;
end;
procedure TFormMainMenu.zBitColBtn5Click(Sender: TObject);
begin
        FormFloor3rd.show;
        Floor2nd.Hide;
        FormFloor1st.Hide;
        FormRtioZc.hide;
        FormCommu.hide;
end;
procedure TFormMainMenu.zBitColBtn4Click(Sender: TObject);
begin
        Floor2nd.show;
        FormFloor1st.Hide;
        FormFloor3rd.Hide;
        FormRtioZc.hide;
        FormCommu.hide;
end;
end.
```

```
unit RtioZc2;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, ComCtrls,
  StdCtrls, zonzoff, Zseg, Mystd, zLed, Buttons, ToolWin;
type
      TFormRtioZc = class(TForm)
      ToolBar1: TToolBar;
      SpeedButton1: TSpeedButton;
      SpeedButton3: TSpeedButton;
      GroupBox1: TGroupBox;
      RTI0: TzLed;
      z3DLabel1: Tz3DLabel;
      RTI1: TzLed;
      RTI2: TzLed;
      RTI3: TzLed;
      RTI4: TzLed;
      RTI5: TzLed;
      RTI6: TzLed;
      RTI7: TzLed;
      z3DLabel2: Tz3DLabel;
      z3DLabel3: Tz3DLabel;
      z3DLabel4: Tz3DLabel;
      z3DLabel5: Tz3DLabel;
      z3DLabel6: Tz3DLabel;
      z3DLabel7: Tz3DLabel;
      z3DLabel8: Tz3DLabel;
      RTI8: TzLed;
      RTI9: TzLed;
      RTI10: TzLed;
```

RTI11: TzLed;

RTI12: TzLed;

RTI13: TzLed;

RTI14: TzLed;

RTI15: TzLed;

z3DLabel9: Tz3DLabel;

z3DLabel10: Tz3DLabel;

z3DLabel11: Tz3DLabel;

z3DLabel12: Tz3DLabel;

z3DLabel13: Tz3DLabel;

z3DLabel14: Tz3DLabel;

z3DLabel15: Tz3DLabel;

z3DLabel16: Tz3DLabel;

zcDotLabelIn: TzcDotLabel;

z3DLabel17: Tz3DLabel;

UnitNumberIn: TComboBox;

PortRead: TSpeedButton;

GroupBox2: TGroupBox;

RTO0: TzLed;

z3DLabel18: Tz3DLabel;

z3DLabel19: Tz3DLabel;

z3DLabel20: Tz3DLabel;

SW0: TzcSingleOnOff;

SW2: TzcSingleOnOff;

SW3: TzcSingleOnOff;

SW4: TzcSingleOnOff;

SW5: TzcSingleOnOff;

SW6: TzcSingleOnOff;

SW7: TzcSingleOnOff;

RTO1: TzLed;

RTO2: TzLed;

RTO3: TzLed;

RTO4: TzLed;

RTO5: TzLed;

RTO6: TzLed;

RTO7: TzLed;

RTO8: TzLed;

SW8: TzcSingleOnOff;

SW9: TzcSingleOnOff;

SW10: TzcSingleOnOff;

SW11: TzcSingleOnOff;

SW12: TzcSingleOnOff;

SW13: TzcSingleOnOff;

SW14: TzcSingleOnOff;

SW15: TzcSingleOnOff;

RTO9: TzLed;

RTO10: TzLed;

RTO11: TzLed;

RTO12: TzLed;

RTO13: TzLed;

RTO14: TzLed;

RTO15: TzLed;

z3DLabel21: Tz3DLabel;

z3DLabel22: Tz3DLabel;

z3DLabel23: Tz3DLabel;

z3DLabel24: Tz3DLabel;

z3DLabel25: Tz3DLabel;

z3DLabel26: Tz3DLabel;

z3DLabel27: Tz3DLabel;

z3DLabel28: Tz3DLabel;

```
z3DLabel29: Tz3DLabel;
z3DLabel30: Tz3DLabel;
z3DLabel31: Tz3DLabel;
z3DLabel32: Tz3DLabel;
z3DLabel33: Tz3DLabel;
zcDotLabelOut: TzcDotLabel;
z3DLabel34: Tz3DLabel;
UnitNumberOut: TComboBox;
Edit2: TEdit;
PortWrite: TSpeedButton;
StatusBar1: TStatusBar;
SW1: TzcSingleOnOff;
GroupBox3: TGroupBox;
UnitNumberMod: TComboBox;
PortMod: TSpeedButton;
zcDotLabelMod: TzcDotLabel;
z3DLabel35: Tz3DLabel;
z3DLabel36: Tz3DLabel;
zLedModIn: TzLed;
zLedModOut: TzLed;
z3DLabel37: Tz3DLabel;
ClearMod: TSpeedButton;
procedure InputWordtoBitLed (HexWord : String);
procedure OutputWordtoBitLed (HexWord : String);
procedure ShowBitLED(b,digit : byte);
procedure UpdatePortWrite;
procedure SW0Click(Sender: TObject);
procedure SW1Click(Sender: TObject);
procedure SW2Click(Sender: TObject);
procedure SW3Click(Sender: TObject);
```

```
        procedure SW4Click(Sender: TObject);
        procedure SW5Click(Sender: TObject);
        procedure SW6Click(Sender: TObject);
        procedure SW7Click(Sender: TObject);
        procedure SW8Click(Sender: TObject);
        procedure SW9Click(Sender: TObject);
        procedure SW10Click(Sender: TObject);
        procedure SW11Click(Sender: TObject);
        procedure SW12Click(Sender: TObject);
        procedure SW13Click(Sender: TObject);
        procedure SW14Click(Sender: TObject);
        procedure SW15Click(Sender: TObject);
        procedure PortWriteClick(Sender: TObject);
        procedure PortReadClick(Sender: TObject);
        procedure UnitNumberInChange(Sender: TObject);
        procedure UnitNumberOutChange(Sender: TObject);
        procedure SpeedButton3Click(Sender: TObject);
        procedure UnitNumberModChange(Sender: TObject);
        procedure PortModClick(Sender: TObject);
        procedure ClearModClick(Sender: TObject);
        procedure SpeedButton1Click(Sender: TObject);
private
        { Private declarations }
public
        { Public declarations }
end;
var
        FormRtioZc: TFormRtioZc;
        OutPutWord:String;
        UnitRti:String;
```

```
        UnitRto:String;
        UnitMod:String;
implementation
uses Commu, MainMenu;
{$R *.DFM}
function HextoStr (Hex : byte) : String;
var     MSB,LSB   : byte;
        MChr,LChr : Char;
begin
        MSB := ((Hex AND $F0) SHR 4) AND $0F;
        LSB := Hex AND $0F;
        if MSB < 10 then MChr := Chr(MSB+$30) else MChr := Chr(MSB+55);
        if LSB < 10 then LChr := Chr(LSB+$30) else LChr := Chr(LSB+55);
        HextoStr := Mchr+LChr;
end;
function HextoBin (Digit : Char) : String ;
begin
        case Digit of
        '0':HextoBin := '0000';
        '1':HextoBin := '0001';
        '2':HextoBin := '0010';
        '3':HextoBin := '0011';
        '4':HextoBin := '0100';
        '5':HextoBin := '0101';
        '6':HextoBin := '0110';
        '7':HextoBin := '0111';
        '8':HextoBin := '1000';
        '9':HextoBin := '1001';
        'A':HextoBin := '1010';
        'B':HextoBin := '1011';
```

```
        'C':HextoBin := '1100';
        'D':HextoBin := '1101';
        'E':HextoBin := '1110';
        'F':HextoBin := '1111';
        end;
end;
function BintoHex (HexBit : String) : Char ;
begin
        if HexBit ='0000' then BintoHex := '0';
        if HexBit ='0001' then BintoHex := '1';
        if HexBit ='0010' then BintoHex := '2';
        if HexBit ='0011' then BintoHex := '3';
        if HexBit ='0100' then BintoHex := '4';
        if HexBit ='0101' then BintoHex := '5';
        if HexBit ='0110' then BintoHex := '6';
        if HexBit ='0111' then BintoHex := '7';
        if HexBit ='1000' then BintoHex := '8';
        if HexBit ='1001' then BintoHex := '9';
        if HexBit ='1010' then BintoHex := 'A';
        if HexBit ='1011' then BintoHex := 'B';
        if HexBit ='1100' then BintoHex := 'C';
        if HexBit ='1101' then BintoHex := 'D';
        if HexBit ='1110' then BintoHex := 'E';
        if HexBit ='1111' then BintoHex := 'F';
end;
procedure  TFormRtioZc.InputWordtoBitLed(HexWord :  String);
var     Digit : String;
begin
       Digit     := HextoBin(HexWord[4]);
        if Digit[4] = '1' then RTI0.Enabled := true else RTI0.Enabled := False;
```

```
        if Digit[3] = '1' then RTI1.Enabled := true else RTI1.Enabled := False;
        if Digit[2] = '1' then RTI2.Enabled := true else RTI2.Enabled := False;
        if Digit[1] = '1' then RTI3.Enabled := true else RTI3.Enabled := False;
        Digit     := HextoBin(HexWord[3]);
        if Digit[4] = '1' then RTI4.Enabled := true else RTI4.Enabled := False;
        if Digit[3] = '1' then RTI5.Enabled := true else RTI5.Enabled := False;
        if Digit[2] = '1' then RTI6.Enabled := true else RTI6.Enabled := False;
        if Digit[1] = '1' then RTI7.Enabled := true else RTI7.Enabled := False;
        Digit     := HextoBin(HexWord[2]);
        if Digit[4] = '1' then RTI8.Enabled := true else RTI8.Enabled := False;
        if Digit[3] = '1' then RTI9.Enabled := true else RTI9.Enabled := False;
        if Digit[2] = '1' then RTI10.Enabled := true else RTI10.Enabled := False;
        if Digit[1] = '1' then RTI11.Enabled := true else RTI11.Enabled := False;
        Digit     := HextoBin(HexWord[1]);
        if Digit[4] = '1' then RTI12.Enabled := true else RTI12.Enabled := False;
        if Digit[3] = '1' then RTI13.Enabled := true else RTI13.Enabled := False;
        if Digit[2] = '1' then RTI14.Enabled := true else RTI14.Enabled := False;
        if Digit[1] = '1' then RTI15.Enabled := true else RTI15.Enabled := False;
end;
procedure  TFormRtioZc.OutputWordtoBitLed(HexWord :  String);
var     Digit : String;
begin
        Digit     := HextoBin(HexWord[4]);
        if Digit[4] = '1' then RTO0.Enabled := true else RTO0.Enabled := False;
        if Digit[3] = '1' then RTO1.Enabled := true else RTO1.Enabled := False;
        if Digit[2] = '1' then RTO2.Enabled := true else RTO2.Enabled := False;
        if Digit[1] = '1' then RTO3.Enabled := true else RTO3.Enabled := False;
        Digit     := HextoBin(HexWord[3]);
        if Digit[4] = '1' then RTO4.Enabled := true else RTO4.Enabled := False;
        if Digit[3] = '1' then RTO5.Enabled := true else RTO5.Enabled := False;
```

```
        if Digit[2] = '1' then RTO6.Enabled := true else RTO6.Enabled := False;
        if Digit[1] = '1' then RTO7.Enabled := true else RTO7.Enabled := False;
        Digit    := HextoBin(HexWord[2]);
        if Digit[4] = '1' then RTO8.Enabled := true else RTO8.Enabled := False;
        if Digit[3] = '1' then RTO9.Enabled := true else RTO9.Enabled := False;
        if Digit[2] = '1' then RTO10.Enabled := true else RTO10.Enabled := False;
        if Digit[1] = '1' then RTO11.Enabled := true else RTO11.Enabled := False;
        Digit    := HextoBin(HexWord[1]);
        if Digit[4] = '1' then RTO12.Enabled := true else RTO12.Enabled := False;
        if Digit[3] = '1' then RTO13.Enabled := true else RTO13.Enabled := False;
        if Digit[2] = '1' then RTO14.Enabled := true else RTO14.Enabled := False;
        if Digit[1] = '1' then RTO15.Enabled := true else RTO15.Enabled := False;
end;
//*********************************************
procedure TFormRtioZc.ShowBitLED(b,digit:byte);
var     Bit : Char;
        HexBin : String;
begin
        HexBin := HextoBin(OutputWord[digit]);
        Bit := HexBin[b];
        if Bit = '0' then Bit := '1' else Bit := '0';
        HexBin[b] := Bit;
        OutPutword[digit] := BintoHex(HexBin);
        Edit2.Text := OutputWord;
        zcDotLabelOut.Caption := Edit2.Text;
        OutputWordtoBitLed(OutputWord);
end;
procedure TFormRtioZc.SW0Click(Sender: TObject);
begin
        ShowBitLED(4,4);
```

```
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW1Click(Sender: TObject);
    begin
        ShowBitLED(3,4);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW2Click(Sender: TObject);
    begin
        ShowBitLED(2,4);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW3Click(Sender: TObject);
    begin
        ShowBitLED(1,4);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW4Click(Sender: TObject);
    begin
        ShowBitLED(4,3);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW5Click(Sender: TObject);
    begin
        ShowBitLED(3,3);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW6Click(Sender: TObject);
    begin
        ShowBitLED(2,3);
```

```
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW7Click(Sender: TObject);
    begin
        ShowBitLED(1,3);
        UpdatePortWrite;
    end;
    //*****************************************
    procedure TFormRtioZc.SW8Click(Sender: TObject);
    begin
        ShowBitLED(4,2);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW9Click(Sender: TObject);
    begin
        ShowBitLED(3,2);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW10Click(Sender: TObject);
    begin
        ShowBitLED(2,2);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW11Click(Sender: TObject);
    begin
        ShowBitLED(1,2);
        UpdatePortWrite;
    end;
    procedure TFormRtioZc.SW12Click(Sender: TObject);
    begin
```

```
        ShowBitLED(4,1);
        UpdatePortWrite;
end;
procedure TFormRtioZc.SW13Click(Sender: TObject);
begin
  ShowBitLED(3,1);
  UpdatePortWrite;
end;
procedure TFormRtioZc.SW14Click(Sender: TObject);
begin
        ShowBitLED(2,1);
        UpdatePortWrite;
end;
procedure TFormRtioZc.SW15Click(Sender: TObject);
begin
        ShowBitLED(1,1);
        UpdatePortWrite;
end;
procedure TFormRtioZc.UpdatePortWrite;
begin
        UnitRto:= UnitNumberOut.Text;
        FormCommu.EditTransmit.Text := '@'+UnitRto+'WR'+OutputWord+'*XX';
end;
procedure TFormRtioZc.PortWriteClick(Sender: TObject);
begin
        if Edit2.Text = '' then Edit2.Text := '0000';
        OutputWord := Edit2.Text;
        zcDotLabelOut.Caption := Edit2.Text;
        OutputWordtoBitLed (OutputWord);
        UpdatePortWrite;
```

```
        FormCommu.PortTransmit;
        StatusBar1.Panels[1].Text:='Command.....'+FormCommu.EditTransmit.Text;
end;
procedure TFormRtioZc.PortReadClick(Sender: TObject);
begin
        UnitRTi:=UnitNumberIn.Text;
        FormCommu.EditTransmit.Text := '@'+UnitRti+'RD*XX';
        FormCommu.PortTransmit;
        StatusBar1.Panels[1].Text:= 'Command.....'+'@'+UnitRti+'RD*XX';
end;
procedure TFormRtioZc.PortModClick(Sender: TObject);
begin
        UnitMod:=UnitNumberMod.Text;
        FormCommu.EditTransmit.Text := '@'+UnitMod+'MD*XX';
        FormCommu.PortTransmit;
        StatusBar1.Panels[1].Text:= 'Command.....'+'@'+UnitMod+'MD*XX';
        zcDotLabelMod.Caption := 'Unit'+UnitMod;
end;
procedure TFormRtioZc.ClearModClick(Sender: TObject);
begin
        UnitNumberMod.Text:='UnitNo';
        zcDotLabelMod.Caption :=' ';
        StatusBar1.Panels[1].Text:= 'Clear Module ';
        zLedModIn.Enabled:=False;
        zLedModOut.Enabled:=False;
end;
procedure TFormRtioZc.UnitNumberInChange(Sender: TObject);
begin
        case UnitNumberIn.ItemIndex of
        0:UnitNumberIn.Text:='00';
```

```
            1:UnitNumberIn.Text:='01';
            2:UnitNumberIn.Text:='02';
            3:UnitNumberIn.Text:='03';
            4:UnitNumberIn.Text:='04';
            5:UnitNumberIn.Text:='05';
            6:UnitNumberIn.Text:='06';
            7:UnitNumberIn.Text:='07';
            8:UnitNumberIn.Text:='08';
            9:UnitNumberIn.Text:='09';
            10:UnitNumberIn.Text:='0A';
            11:UnitNumberIn.Text:='0B';
            12:UnitNumberIn.Text:='0C';
            13:UnitNumberIn.Text:='0D';
            14:UnitNumberIn.Text:='0E';
            15:UnitNumberIn.Text:='0F';
    end;
end;
procedure TFormRtioZc.UnitNumberOutChange(Sender: TObject);
begin
        case UnitNumberOut.ItemIndex of
        0:UnitNumberOut.Text:='00';
        1:UnitNumberOut.Text:='01';
        2:UnitNumberOut.Text:='02';
        3:UnitNumberOut.Text:='03';
        4:UnitNumberOut.Text:='04';
        5:UnitNumberOut.Text:='05';
        6:UnitNumberOut.Text:='06';
        7:UnitNumberOut.Text:='07';
        8:UnitNumberOut.Text:='08';
        9:UnitNumberOut.Text:='09';
```

```
            10:UnitNumberOut.Text:='0A';
            11:UnitNumberOut.Text:='0B';
            12:UnitNumberOut.Text:='0C';
            13:UnitNumberOut.Text:='0D';
            14:UnitNumberOut.Text:='0E';
            15:UnitNumberOut.Text:='0F';
    end;
end;
procedure TFormRtioZc.UnitNumberModChange(Sender: TObject);
begin
            case UnitNumberMod.ItemIndex of
            0:UnitNumberMod.Text:='00';
            1:UnitNumberMod.Text:='01';
            2:UnitNumberMod.Text:='02';
            3:UnitNumberMod.Text:='03';
            4:UnitNumberMod.Text:='04';
            5:UnitNumberMod.Text:='05';
            6:UnitNumberMod.Text:='06';
            7:UnitNumberMod.Text:='07';
            8:UnitNumberMod.Text:='08';
            9:UnitNumberMod.Text:='09';
            10:UnitNumberMod.Text:='0A';
            11:UnitNumberMod.Text:='0B';
            12:UnitNumberMod.Text:='0C';
            13:UnitNumberMod.Text:='0D';
            14:UnitNumberMod.Text:='0E';
            15:UnitNumberMod.Text:='0F';
    end;
end;
```

```
procedure TFormRtioZc.SpeedButton3Click(Sender: TObject);
begin
        FormCommu.Show;
end;
procedure TFormRtioZc.SpeedButton1Click(Sender: TObject);
begin
        FormMainMenu.Show;
        FormRtioZc.Hide;
end;
end.
```

## Features

- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

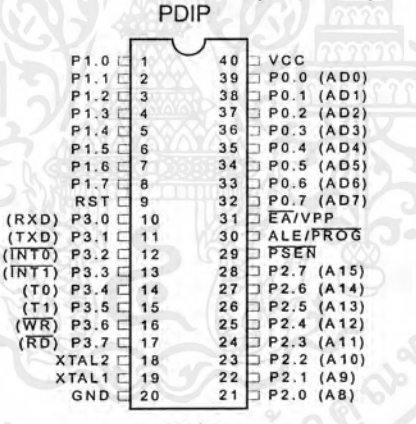
## Description

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

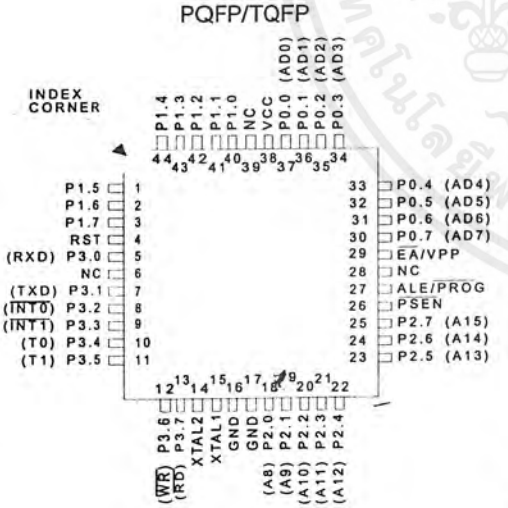
*(continued)*

## Pin Configurations

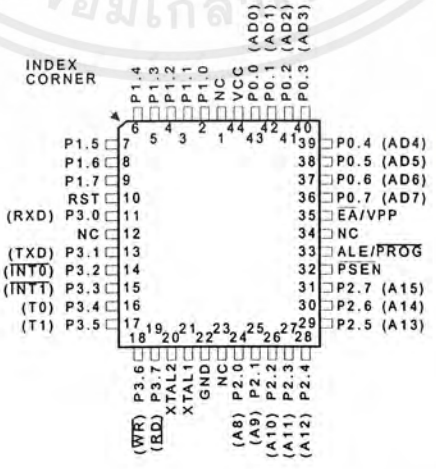
PDIP




PQFP/TQFP




PLCC

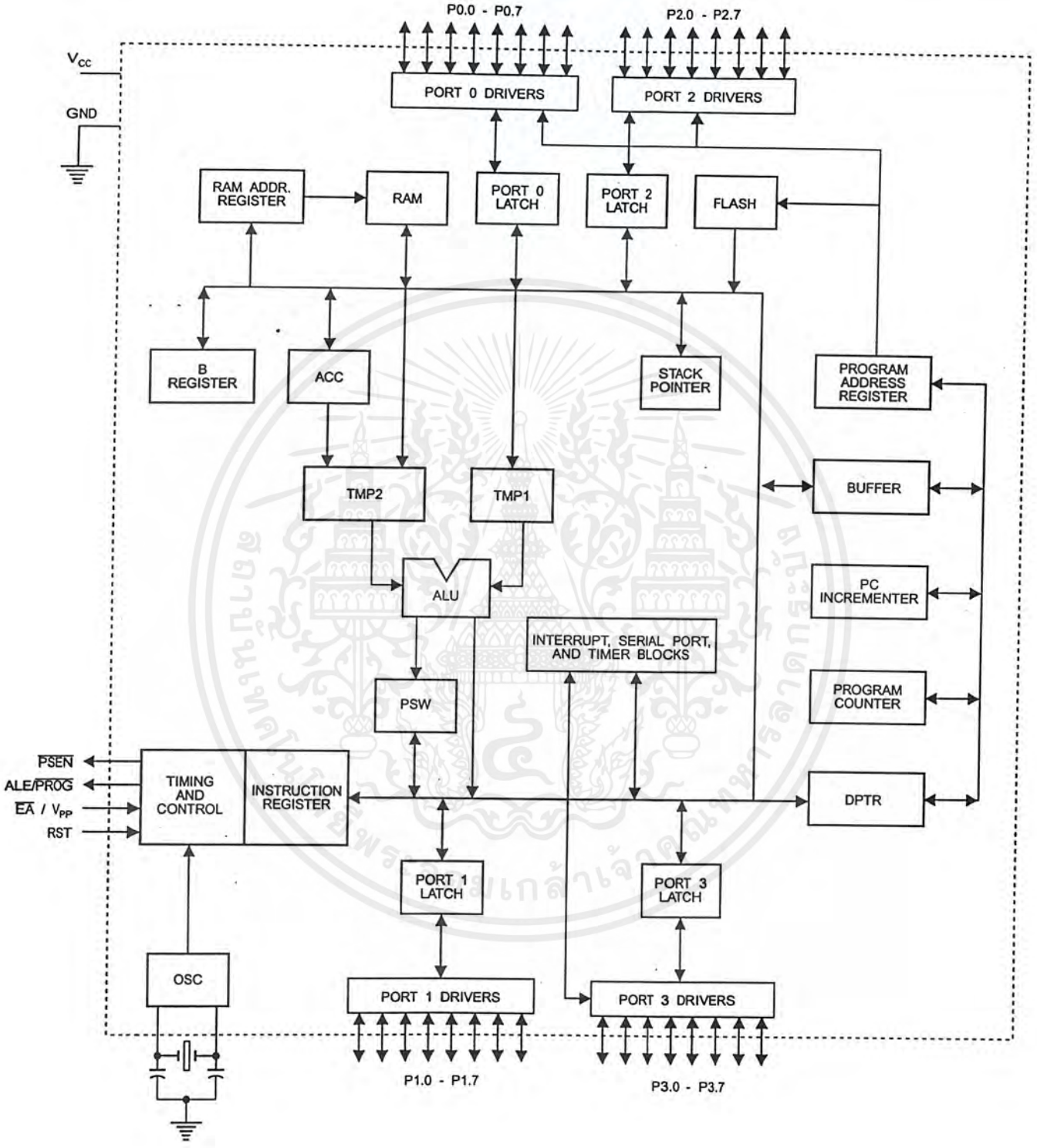



# 8-Bit Microcontroller with 4K Bytes Flash

# AT89C51

0265F-A–12/97

## Block Diagram



The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

## Pin Description

**$V_{CC}$**
Supply voltage.

**GND**
Ground.

**Port 0**
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 may also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pullups are required during program verification.

**Port 1**
Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current ($I_{IL}$) because of the internal pullups.

Port 1 also receives the low-order address bytes during Flash programming and verification.

**Port 2**
Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current ($I_{IL}$) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

**Port 3**
Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current ($I_{IL}$) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C51 as listed below:

| Port Pin | Alternate Functions |
|---|---|
| P3.0 | RXD (serial input port) |
| P3.1 | TXD (serial output port) |
| P3.2 | $\overline{INT0}$ (external interrupt 0) |
| P3.3 | $\overline{INT1}$ (external interrupt 1) |
| P3.4 | T0 (timer 0 external input) |
| P3.5 | T1 (timer 1 external input) |
| P3.6 | $\overline{WR}$ (external data memory write strobe) |
| P3.7 | $\overline{RD}$ (external data memory read strobe) |

Port 3 also receives some control signals for Flash programming and verification.

**RST**
Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

**ALE/$\overline{PROG}$**
Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

**$\overline{PSEN}$**
Program Store Enable is the read strobe to external program memory.

When the AT89C51 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

**$\overline{EA}$/$V_{PP}$**
External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to $V_{CC}$ for internal program executions.

This pin also receives the 12-volt programming enable voltage ($V_{PP}$) during Flash programming, for parts that require 12-volt $V_{PP}$.

**XTAL1**
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

**XTAL2**
Output from the inverting oscillator amplifier.

## Oscillator Characteristics

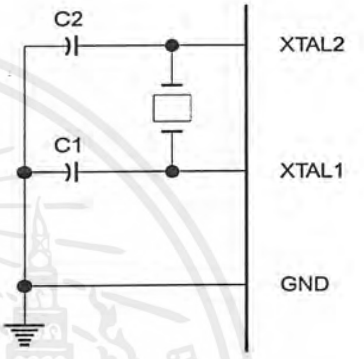
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

## Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

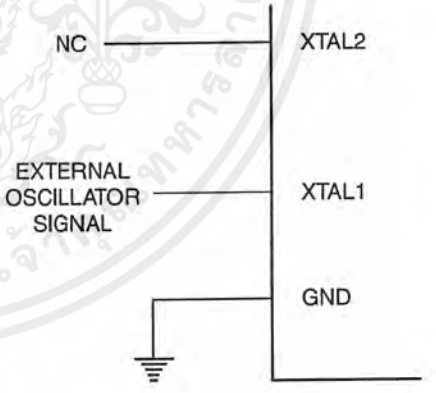
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

**Figure 1.** Oscillator Connections




Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

**Figure 2.** External Clock Drive Configuration




## Status of External Pins During Idle and Power Down Modes

| Mode | Program Memory | ALE | $\overline{PSEN}$ | PORT0 | PORT1 | PORT2 | PORT3 |
|---|---|---|---|---|---|---|---|
| Idle | Internal | 1 | 1 | Data | Data | Data | Data |
| Idle | External | 1 | 1 | Float | Data | Address | Data |
| Power Down | Internal | 0 | 0 | Data | Data | Data | Data |
| Power Down | External | 0 | 0 | Float | Data | Data | Data |

## Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is terminated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before $V_{CC}$ is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

## Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

## Lock Bit Protection Modes

| Program Lock Bits | | | | Protection Type |
|---|---|---|---|---|
| | LB1 | LB2 | LB3 | |
| 1 | U | U | U | No program lock features. |
| 2 | P | U | U | MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled. |
| 3 | P | P | U | Same as mode 2, also verify is disabled. |
| 4 | P | P | P | Same as mode 3, also external execution is disabled. |

## Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage ($V_{CC}$) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

| | $V_{PP}$ = 12V | $V_{PP}$ = 5V |
|---|---|---|
| Top-Side Mark | AT89C51<br>xxxx<br>yyww | AT89C51<br>xxxx-5<br>yyww |
| Signature | (030H)=1EH<br>(031H)=51H<br>(032H)=FFH | (030H)=1EH<br>(031H)=51H<br>(032H)=05H |

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

**Programming Algorithm:** Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

**$\overline{Data}$ Polling:** The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

**Ready/$\overline{Busy}$:** The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

**Program Verify:** If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

**Chip Erase:** The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

**Reading the Signature Bytes:** The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 51H indicates 89C51
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

## Programming Interface

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

## Flash Programming Modes

| Mode | | RST | $\overline{PSEN}$ | ALE/$\overline{PROG}$ | $\overline{EA}$/$V_{PP}$ | P2.6 | P2.7 | P3.6 | P3.7 |
|---|---|---|---|---|---|---|---|---|---|
| Write Code Data | | H | L |  | H/12V | L | H | H | H |
| Read Code Data | | H | L | H | H | L | L | H | H |
| Write Lock | Bit - 1 | H | L |  | H/12V | H | H | H | H |
| | Bit - 2 | H | L |  | H/12V | H | H | L | L |
| | Bit - 3 | H | L |  | H/12V | H | L | H | L |
| Chip Erase | | H | L |  (1) | H/12V | H | L | L | L |
| Read Signature Byte | | H | L | H | H | L | L | L | L |

Note: 1. Chip Erase requires a 10-ms PROG pulse.

**Figure 3.** Programming the Flash

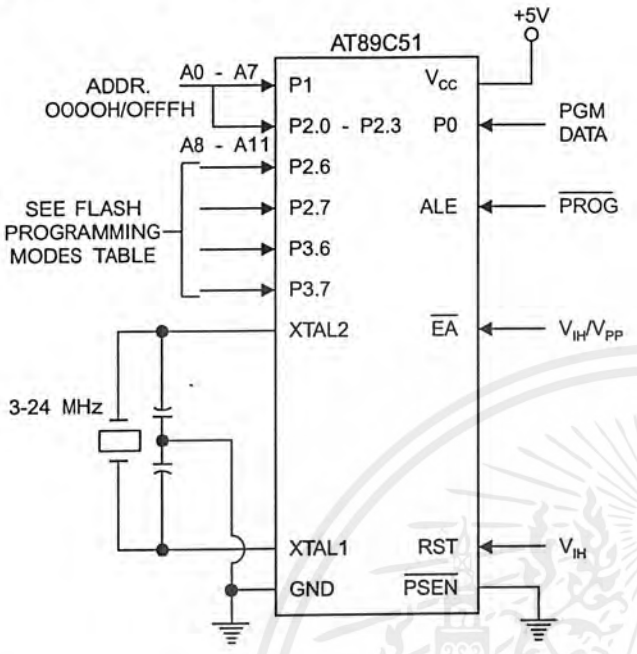
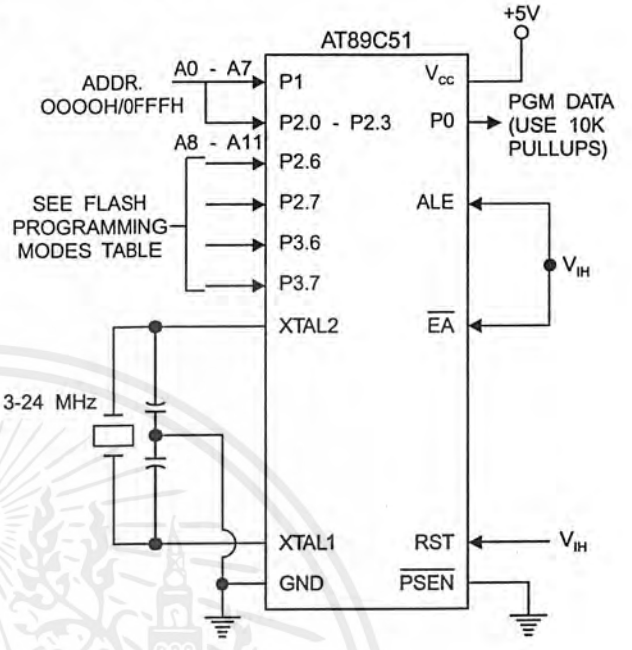



**Figure 4.** Verifying the Flash



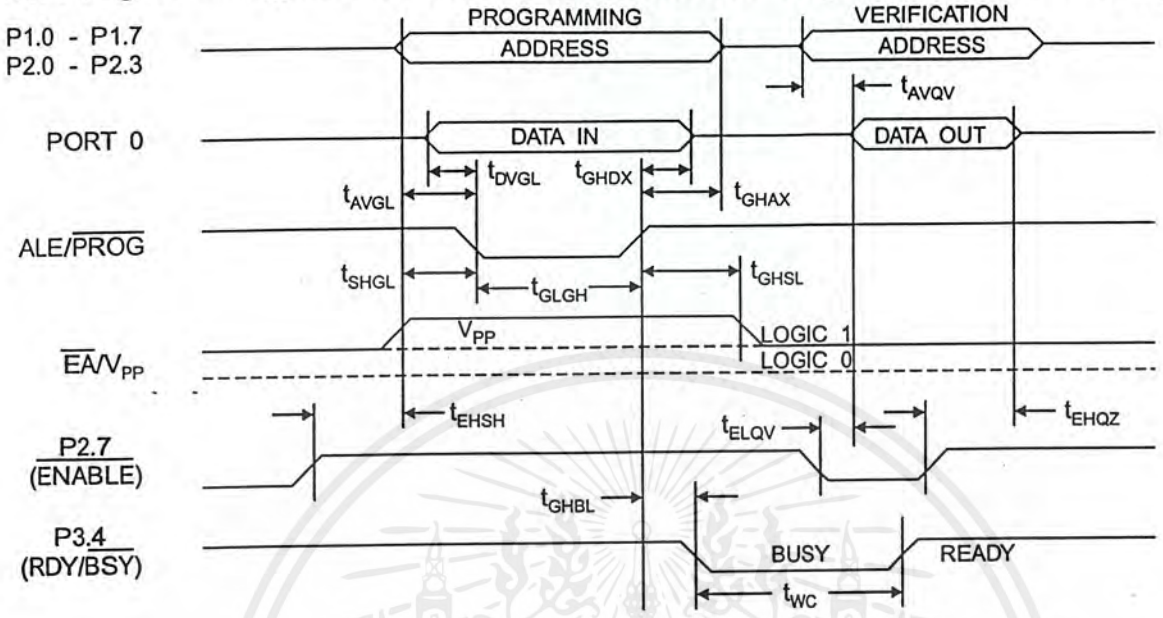

## Flash Programming and Verification Characteristics

$T_A$ = 0°C to 70°C, $V_{CC}$ = 5.0 ± 10%

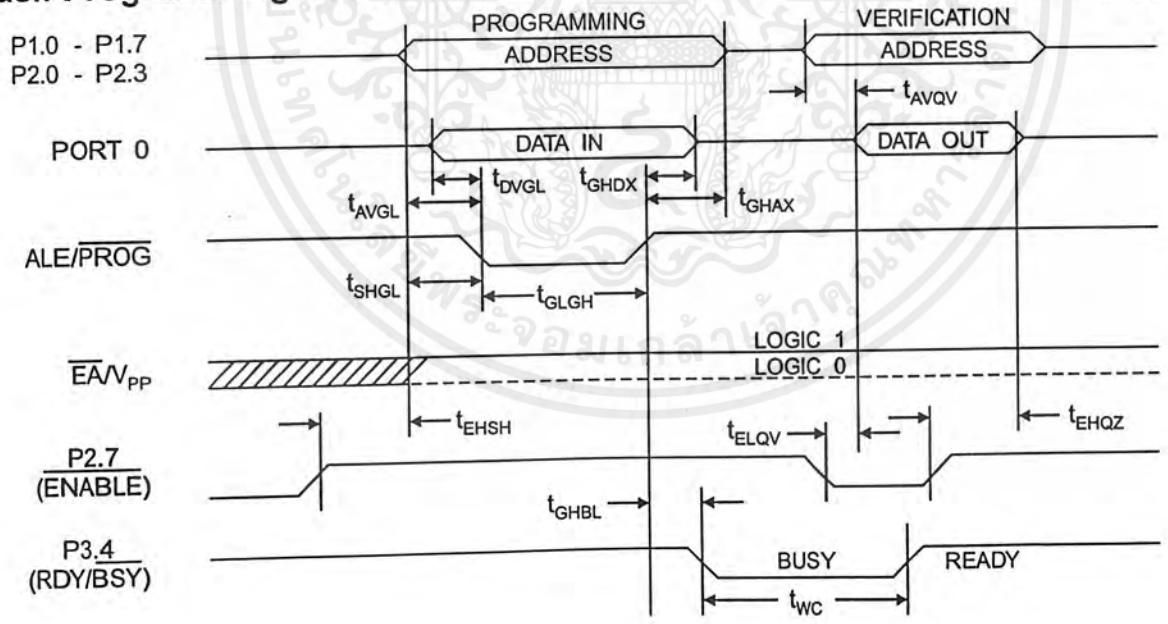
| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $V_{PP}$(1) | Programming Enable Voltage | 11.5 | 12.5 | V |
| $I_{PP}$(1) | Programming Enable Current | | 1.0 | mA |
| $1/t_{CLCL}$ | Oscillator Frequency | 3 | 24 | MHz |
| $t_{AVGL}$ | Address Setup to $\overline{PROG}$ Low | $48t_{CLCL}$ | | |
| $t_{GHAX}$ | Address Hold After $\overline{PROG}$ | $48t_{CLCL}$ | | |
| $t_{DVGL}$ | Data Setup to $\overline{PROG}$ Low | $48t_{CLCL}$ | | |
| $t_{GHDX}$ | Data Hold After $\overline{PROG}$ | $48t_{CLCL}$ | | |
| $t_{EHSH}$ | P2.7 ($\overline{ENABLE}$) High to $V_{PP}$ | $48t_{CLCL}$ | | |
| $t_{SHGL}$ | $V_{PP}$ Setup to $\overline{PROG}$ Low | 10 | | µs |
| $t_{GHSL}$(1) | $V_{PP}$ Hold After $\overline{PROG}$ | 10 | | µs |
| $t_{GLGH}$ | $\overline{PROG}$ Width | 1 | 110 | µs |
| $t_{AVQV}$ | Address to Data Valid | | $48t_{CLCL}$ | |
| $t_{ELQV}$ | $\overline{ENABLE}$ Low to Data Valid | | $48t_{CLCL}$ | |
| $t_{EHQZ}$ | Data Float After $\overline{ENABLE}$ | 0 | $48t_{CLCL}$ | |
| $t_{GHBL}$ | $\overline{PROG}$ High to $\overline{BUSY}$ Low | | 1.0 | µs |
| $t_{WC}$ | Byte Write Cycle Time | | 2.0 | ms |

Note: 1. Only used in 12-volt programming mode.

## Flash Programming and Verification Waveforms - High Voltage Mode ($V_{PP}$ = 12V)




## Flash Programming and Verification Waveforms - Low Voltage Mode ($V_{PP}$ = 5V)



## Absolute Maximum Ratings*

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage on Any Pin with Respect to Ground | -1.0V to +7.0V |
| Maximum Operating Voltage | 6.6V |
| DC Output Current | 15.0 mA |

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## DC Characteristics

$T_A$ = -40°C to 85°C, $V_{CC}$ = 5.0V ± 20% (unless otherwise noted)

| Symbol | Parameter | Condition | Min | Max | Units |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | (Except $\overline{EA}$) | -0.5 | 0.2 $V_{CC}$ - 0.1 | V |
| $V_{IL1}$ | Input Low Voltage ($\overline{EA}$) | | -0.5 | 0.2 $V_{CC}$ - 0.3 | V |
| $V_{IH}$ | Input High Voltage | (Except XTAL1, RST) | 0.2 $V_{CC}$ + 0.9 | $V_{CC}$ + 0.5 | V |
| $V_{IH1}$ | Input High Voltage | (XTAL1, RST) | 0.7 $V_{CC}$ | $V_{CC}$ + 0.5 | V |
| $V_{OL}$ | Output Low Voltage(1) (Ports 1,2,3) | $I_{OL}$ = 1.6 mA | | 0.45 | V |
| $V_{OL1}$ | Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$) | $I_{OL}$ = 3.2 mA | | 0.45 | V |
| $V_{OH}$ | Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$) | $I_{OH}$ = -60 µA, $V_{CC}$ = 5V ± 10% | 2.4 | | V |
| | | $I_{OH}$ = -25 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -10 µA | 0.9 $V_{CC}$ | | V |
| $V_{OH1}$ | Output High Voltage (Port 0 in External Bus Mode) | $I_{OH}$ = -800 µA, $V_{CC}$ = 5V ± 10% | 2.4 | | V |
| | | $I_{OH}$ = -300 µA | 0.75 $V_{CC}$ | | V |
| | | $I_{OH}$ = -80 µA | 0.9 $V_{CC}$ | | V |
| $I_{IL}$ | Logical 0 Input Current (Ports 1,2,3) | $V_{IN}$ = 0.45V | | -50 | µA |
| $I_{TL}$ | Logical 1 to 0 Transition Current (Ports 1,2,3) | $V_{IN}$ = 2V, VCC = 5V ± 10% | | -650 | µA |
| $I_{LI}$ | Input Leakage Current (Port 0, $\overline{EA}$) | 0.45 < $V_{IN}$ < $V_{CC}$ | | ±10 | µA |
| RRST | Reset Pulldown Resistor | | 50 | 300 | KΩ |
| $C_{IO}$ | Pin Capacitance | Test Freq. = 1 MHz, $T_A$ = 25°C | | 10 | pF |
| $I_{CC}$ | Power Supply Current | Active Mode, 12 MHz | | 20 | mA |
| | | Idle Mode, 12 MHz | | 5 | mA |
| | Power Down Mode(2) | $V_{CC}$ = 6V | | 100 | µA |
| | | $V_{CC}$ = 3V | | 40 | µA |

Notes: 1. Under steady state (non-transient) conditions, $I_{OL}$ must be externally limited as follows:
Maximum $I_{OL}$ per port pin: 10 mA
Maximum $I_{OL}$ per 8-bit port: Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total $I_{OL}$ for all output pins: 71 mA
If $I_{OL}$ exceeds the test condition, $V_{OL}$ may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum $V_{CC}$ for Power Down is 2V.

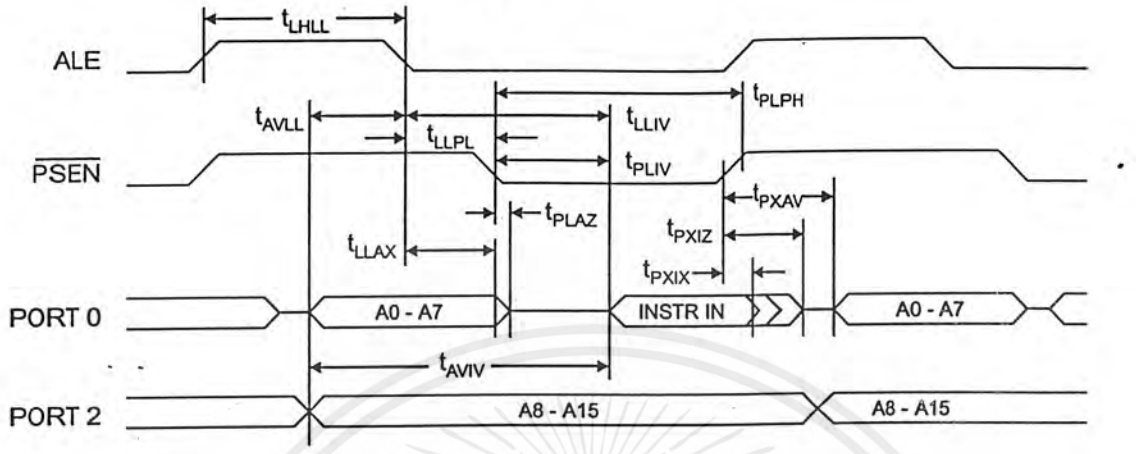
## AC Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all other outputs = 80 pF)

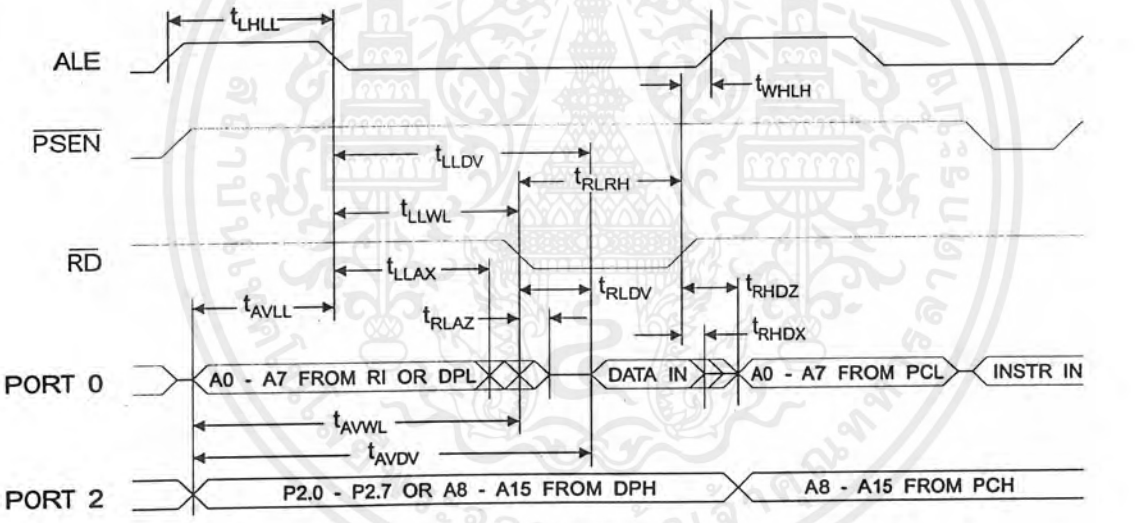
## External Program and Data Memory Characteristics

| Symbol | Parameter | 12 MHz Oscillator | | 16 to 24 MHz Oscillator | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | | | 0 | 24 | MHz |
| $t_{LHLL}$ | ALE Pulse Width | 127 | | $2t_{CLCL}$-40 | | ns |
| $t_{AVLL}$ | Address Valid to ALE Low | 43 | | $t_{CLCL}$-13 | | ns |
| $t_{LLAX}$ | Address Hold After ALE Low | 48 | | $t_{CLCL}$-20 | | ns |
| $t_{LLIV}$ | ALE Low to Valid Instruction In | | 233 | | $4t_{CLCL}$-65 | ns |
| $t_{LLPL}$ | ALE Low to $\overline{PSEN}$ Low | 43 | | $t_{CLCL}$-13 | | ns |
| $t_{PLPH}$ | $\overline{PSEN}$ Pulse Width | 205 | | $3t_{CLCL}$-20 | | ns |
| $t_{PLIV}$ | $\overline{PSEN}$ Low to Valid Instruction In | | 145 | | $3t_{CLCL}$-45 | ns |
| $t_{PXIX}$ | Input Instruction Hold After $\overline{PSEN}$ | 0 | | 0 | | ns |
| $t_{PXIZ}$ | Input Instruction Float After $\overline{PSEN}$ | | 59 | | $t_{CLCL}$-10 | ns |
| $t_{PXAV}$ | $\overline{PSEN}$ to Address Valid | 75 | | $t_{CLCL}$-8 | | ns |
| $t_{AVIV}$ | Address to Valid Instruction In | | 312 | | $5t_{CLCL}$-55 | ns |
| $t_{PLAZ}$ | $\overline{PSEN}$ Low to Address Float | | 10 | | 10 | ns |
| $t_{RLRH}$ | $\overline{RD}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| $t_{WLWH}$ | $\overline{WR}$ Pulse Width | 400 | | $6t_{CLCL}$-100 | | ns |
| $t_{RLDV}$ | $\overline{RD}$ Low to Valid Data In | | 252 | | $5t_{CLCL}$-90 | ns |
| $t_{RHDX}$ | Data Hold After $\overline{RD}$ | 0 | | 0 | | ns |
| $t_{RHDZ}$ | Data Float After $\overline{RD}$ | | 97 | | $2t_{CLCL}$-28 | ns |
| $t_{LLDV}$ | ALE Low to Valid Data In | | 517 | | $8t_{CLCL}$-150 | ns |
| $t_{AVDV}$ | Address to Valid Data In | | 585 | | $9t_{CLCL}$-165 | ns |
| $t_{LLWL}$ | ALE Low to $\overline{RD}$ or $\overline{WR}$ Low | 200 | 300 | $3t_{CLCL}$-50 | $3t_{CLCL}$+50 | ns |
| $t_{AVWL}$ | Address to $\overline{RD}$ or $\overline{WR}$ Low | 203 | | $4t_{CLCL}$-75 | | ns |
| $t_{QVWX}$ | Data Valid to $\overline{WR}$ Transition | 23 | | $t_{CLCL}$-20 | | ns |
| $t_{QVWH}$ | Data Valid to $\overline{WR}$ High | 433 | | $7t_{CLCL}$-120 | | ns |
| $t_{WHQX}$ | Data Hold After $\overline{WR}$ | 33 | | $t_{CLCL}$-20 | | ns |
| $t_{RLAZ}$ | $\overline{RD}$ Low to Address Float | | 0 | | 0 | ns |
| $t_{WHLH}$ | $\overline{RD}$ or $\overline{WR}$ High to ALE High | 43 | 123 | $t_{CLCL}$-20 | $t_{CLCL}$+25 | ns |

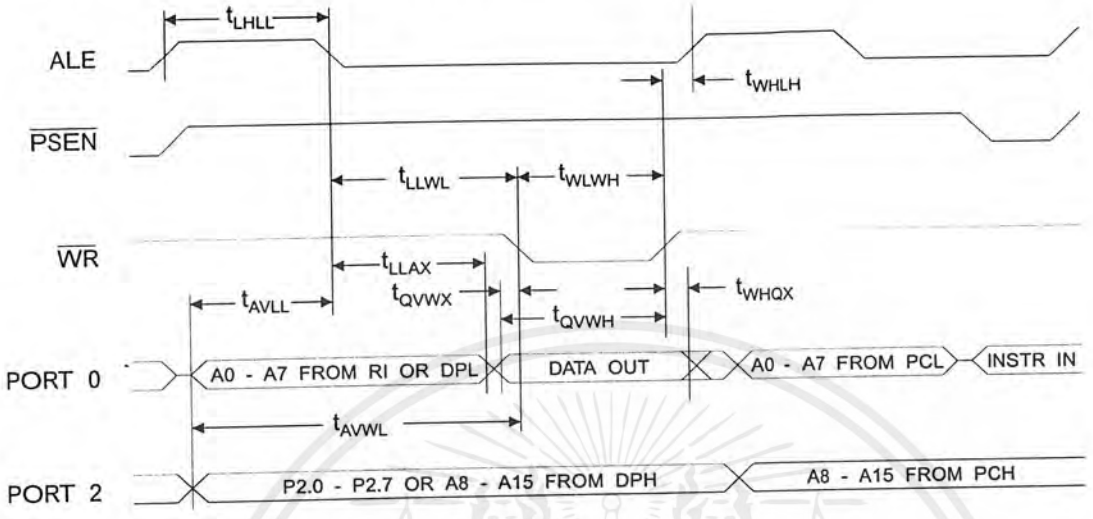
## External Program Memory Read Cycle



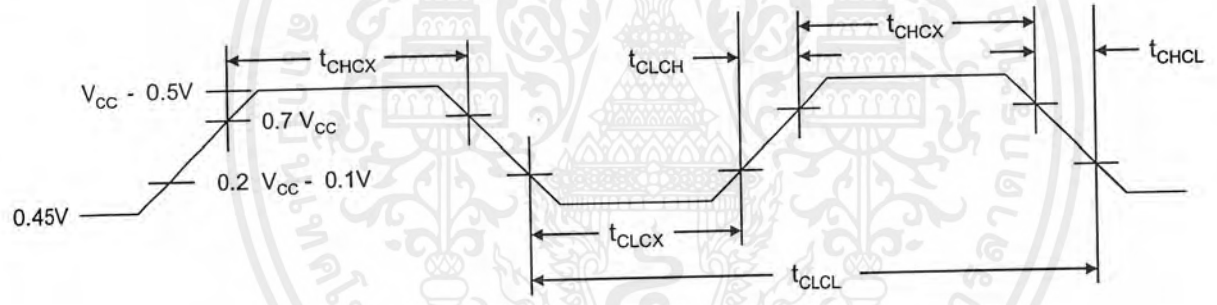

## External Data Memory Read Cycle



## External Data Memory Write Cycle




## External Clock Drive Waveforms




## External Clock Drive

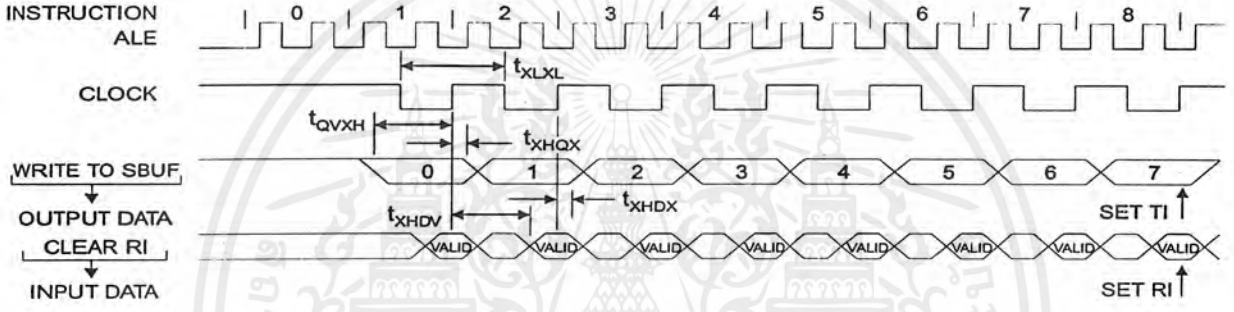
| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $1/t_{CLCL}$ | Oscillator Frequency | 0 | 24 | MHz |
| $t_{CLCL}$ | Clock Period | 41.6 | | ns |
| $t_{CHCX}$ | High Time | 15 | | ns |
| $t_{CLCX}$ | Low Time | 15 | | ns |
| $t_{CLCH}$ | Rise Time | | 20 | ns |
| $t_{CHCL}$ | Fall Time | | 20 | ns |

## Serial Port Timing: Shift Register Mode Test Conditions

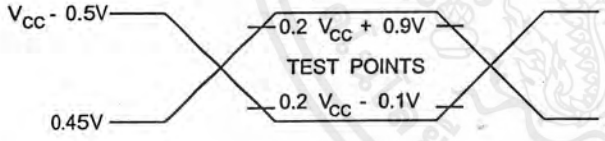
($V_{CC}$ = 5.0 V ± 20%; Load Capacitance = 80 pF)

| Symbol | Parameter | 12 MHz Osc | | Variable Oscillator | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{XLXL}$ | Serial Port Clock Cycle Time | 1.0 | | $12t_{CLCL}$ | | µs |
| $t_{QVXH}$ | Output Data Setup to Clock Rising Edge | 700 | | $10t_{CLCL}$-133 | | ns |
| $t_{XHQX}$ | Output Data Hold After Clock Rising Edge | 50 | | $2t_{CLCL}$-117 | | ns |
| $t_{XHDX}$ | Input Data Hold After Clock Rising Edge | 0 | | 0 | | ns |
| $t_{XHDV}$ | Clock Rising Edge to Input Data Valid | | 700 | | $10t_{CLCL}$-133 | ns |

## Shift Register Mode Timing Waveforms

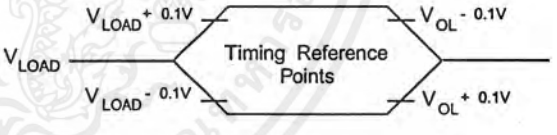



## AC Testing Input/Output Waveforms(1)




Note: 1. AC Inputs during testing are driven at $V_{CC}$ - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at $V_{IH}$ min. for a logic 1 and $V_{IL}$ max. for a logic 0.

## Float Waveforms(1)




Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded $V_{OH}/V_{OL}$ level occurs.

## Ordering Information

| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 12 | 5V ± 20% | AT89C51-12AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-12JC | 44J | |
| | | AT89C51-12PC | 40P6 | |
| | | AT89C51-12QC | 44Q | |
| | | AT89C51-12AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-12JI | 44J | |
| | | AT89C51-12PI | 40P6 | |
| | | AT89C51-12QI | 44Q | |
| | | AT89C51-12AA | 44A | Automotive (-40°C to 105°C) |
| | | AT89C51-12JA | 44J | |
| | | AT89C51-12PA | 40P6 | |
| | | AT89C51-12QA | 44Q | |
| 16 | 5V ± 20% | AT89C51-16AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-16JC | 44J | |
| | | AT89C51-16PC | 40P6 | |
| | | AT89C51-16QC | 44Q | |
| | | AT89C51-16AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-16JI | 44J | |
| | | AT89C51-16PI | 40P6 | |
| | | AT89C51-16QI | 44Q | |
| | | AT89C51-16AA | 44A | Automotive (-40°C to 105°C) |
| | | AT89C51-16JA | 44J | |
| | | AT89C51-16PA | 40P6 | |
| | | AT89C51-16QA | 44Q | |
| 20 | 5V ± 20% | AT89C51-20AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-20JC | 44J | |
| | | AT89C51-20PC | 40P6 | |
| | | AT89C51-20QC | 44Q | |
| | | AT89C51-20AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-20JI | 44J | |
| | | AT89C51-20PI | 40P6 | |
| | | AT89C51-20QI | 44Q | |

## Ordering Information

| Speed (MHz) | Power Supply | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 24 | 5V ± 20% | AT89C51-24AC | 44A | Commercial (0°C to 70°C) |
| | | AT89C51-24JC | 44J | |
| | | AT89C51-24PC | 44P6 | |
| | | AT89C51-24QC | 44Q | |
| | | AT89C51-24AI | 44A | Industrial (-40°C to 85°C) |
| | | AT89C51-24JI | 44J | |
| | | AT89C51-24PI | 44P6 | |
| | | AT89C51-24QI | 44Q | |

| Package Type | |
|---|---|
| 44A | 44 Lead, Thin Plastic Gull Wing Quad Flatpack (TQFP) |
| 44J | 44 Lead, Plastic J-Leaded Chip Carrier (PLCC) |
| 40P6 | 40 Lead, 0.600" Wide, Plastic Dual Inline Package (PDIP) |
| 44Q | 44 Lead, Plastic Gull Wing Quad Flatpack (PQFP) |

# บรรณานุกรม

กฤษฎา ใจเย็น , ชัยวัฒน์ ลิ้มพรจิตวิไล , "เรียนรู้การเชื่อมต่อ PC กับอุปกรณ์ภายนอกผ่านพอร์ทอนุกรม serial port interfaceing starter book", บริษัท ชินโนเวตีฟ เอ็กเฟอร์เมนต์ จำกัด, 117 หน้า, 2542

กนกพร ภาวศุทธิกุล, จักรพงษ์ สุขประเสริฐ, สัจจะ จรัสรุ่งรวีวร, "Delphi 4.0", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 328 หน้า, 2542

จิรายุ วิริยะพิบูล, "Borland Delphi 3.0 Amazing Components", บริษัท ซอฟท์แวร์ มิชชั่น จำกัด, 270 หน้า, 2537

ชัยวัฒน์ คำรัตน์, "นิตยสาร Computer Time", บริษัท คอมพิวเตอร์ เอจ เทคโนโลยี จำกัด, 328 หน้า, 2542

ภาสกร ศิวะโสภา, ภูวนัย สงวนวรรณ, "3D Studio Max 2", บริษัท เฟิสท์ แปซิฟิก มีเดีย (ไทยแลนด์), 420 หน้า, 2541

อำนาจ ผดุงศิลป์, "วารสาร Industrial Technology Review ฉบับที่ 31 เดือนเมษายน 2540", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 135 หน้า, 2540