ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

Computer Control Unit for Department of Engineering Education

ชื่อนักศึกษา 1. นายเพ็ญประพรรค์ บรรดาศักดิ์ รหัสประจำตัว 48035281

2. นางสาวสุดารัตน์ สมนา รหัสประจำตัว 48035301

3. นายเอกบดินทร์ กลิ่นเกษร รหัสประจำตัว 48035314

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชา วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา อ.อำพล ทองระอา

อาจารย์ที่ปรึกษาร่วม รศ.พีระวุฒิ สุวรรณจันทร์

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อ.ปิยะ ศุภวราสุวัฒน์	
2. อ.สุระชัย พิมพ์สาลี	
3. อ.อำพล ทองระอา	
4. อ.พิชญ์สินี มะโน	
5. อ.สุรพงษ์ สิริพงศ์ดี	

วัน/เดือน/ปีที่สอบ วันศุกร์ที่ 11 เดือนพฤษภาคม พ.ศ. 2550 เวลา 14.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(รศ.สุรสิทธิ์ ราดรี)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 30 เดือน พ.ค. พ.ศ. 50

<BT492592>

ปริญญานิพนธ์

ระบบควบคุมการใช้งานคอมพิวเตอร์สำหรับภาควิชาครุศาสตร์วิศวกรรม

COMPUTER CONTROL UNIT FOR DEPARTMENT OF ENGINEERING EDUCATION

เพ็ญประพรรค์ บรรดาศักดิ์
สุดารัตน์ สมนา
เอกบดินทร์ กลิ่นเกษร

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ปริญญานิพนธ์

เรื่อง ระบบควบคุมการใช้งานคอมพิวเตอร์สำหรับภาควิชาครุศาสตร์วิศวกรรม

Computer Control Unit for Department of Engineering Education

วัตถุประสงค์

1. เพื่อศึกษาการทำงานของระบบโดยคุมการใช้งานและบันทึกคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
2. เพื่อออกแบบชุดควบคุมและบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
3. เพื่อสร้างระบบควบคุมการใช้งานและบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
4. เพื่อทดสอบระบบควบคุมการใช้งานและบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
5. เพื่อนำระบบควบคุมการใช้งานและบันทึกคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ความรู้และเข้าใจการทำงานของระบบควบคุมการใช้งานและบันทึกคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
2. ได้โครงสร้างระบบและวงจรของระบบควบคุมการใช้งานและบันทึกคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
3. ได้เครื่องและระบบควบคุมการใช้และบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
4. ได้การทดสอบการทำงานของระบบควบคุมการใช้งานและบันทึกคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม
5. ติดตั้งการใช้งานและตรวจสอบการได้จริง

ชื่อหัวข้อ	ระบบควบคุมการใช้งานคอมพิวเตอร์สำหรับภาควิชาครุศาสตร์วิศวกรรม
นักศึกษา	นายเพ็ญประพรรค์ บรรดาศักดิ์
	นางสาวสุดารัตน์ สมนา
	นายเอกบดินทร์ กลิ่นเกษร
อาจารย์ที่ปรึกษา	อาจารย์อำพล ทองระอา
อาจารย์ที่ปรึกษาร่วม	รศ. พีระวุฒิ สุวรรณจันทร์
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขา	วิศวกรรมโทรคมนาคม
ปีการศึกษา	2549

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการสร้างระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมเพื่อช่วยในการควบคุมและดูแลการเข้าใช้งานห้องปฏิบัติการคอมพิวเตอร์อย่างเป็นระบบซึ่งตัวระบบจะทำการบันทึกเวลาในการใช้งานเครื่องคอมพิวเตอร์ของแต่ละบุคคลเก็บไว้ ระบบบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมมีส่วนประกอบอยู่สี่ส่วนคือ (1) บาร์โค้ด (2) ส่วนควบคุมหลัก (3) ส่วนควบคุมย่อย (4) ส่วนของโปรแกรมที่ใช้ควบคุมเครื่องคอมพิวเตอร์ การทำงานจะเริ่มจากเครื่องอ่านบาร์โค้ดที่เป็นตัวรับข้อมูลจากบัตรนักศึกษาส่งให้เครื่องควบคุมหลัก ส่วนเครื่องควบคุมหลักจะทำหน้าที่รับข้อมูลที่ได้จากเครื่องอ่านบาร์โค้ดมาตรวจสอบกับฐานข้อมูลที่มีหากข้อมูลที่ได้มาอยู่ในภาควิชาครุศาสตร์วิศวกรรมก็จะสั่งการไปยังเครื่องควบคุมย่อยและทำการเก็บเวลาเริ่มเข้าใช้งานไว้ เครื่องควบคุมย่อยจะทำหน้าที่รับคำสั่งจากเครื่องควบคุมหลักเพื่อสั่งให้ตัวโปรแกรมที่ควบคุมคอมพิวเตอร์เปิดหรือปิดโปรแกรมโดยส่งข้อมูลผ่านพอร์ต RS232 โปรแกรมที่ควบคุมคอมพิวเตอร์นั้นจะถูกติดตั้งที่เครื่องคอมพิวเตอร์แต่ละเครื่องโดยมีหน้าที่ทำการควบคุมเมาส์และคีย์บอร์ดไม่ให้ใช้งานได้จนกว่าจะมีคำสั่งมาจากตัวควบคุมย่อยให้ทำการปิดโปรแกรมจึงจะทำให้เมาส์และคีย์บอร์ดใช้งานได้ตามปกติ

Thesis Title	Computer Control Unit for Department of Engineering Education	
Students	Mr.Penprapan	Bandasak
	Miss.Sudarat	Somana
	Mr.Ekbodin	Glingason
Advisor	Mr.Amphon	Thongra-ar
Co-Advisor	Assoc.Prof.Peerawut	Suwanjan
Education Level	Bachelor of Science in Industrial Education	
Program in	Telecommunication Engineering	
Academic Year	2006	

ABSTRACT

Thesis proposes the time record development system for using computer with the faculty of Industrial Education. This system is used to record the time used computer for each person this system assemble of 4 parts (1) Barcode; (2) The major controller; (3) Sub-controller and (4) Control programme. The working system computers start by reading the barcode from student identification and transfer the data to the major controller. Then the major controller will check up the data with the data base of the data is matching, the major controller will record the time use. The sub-controller duty is to transfer the order from the major controller for opening and closing programme through port RS 232. The last part is the computer controller programme which is attached to each computer set. The programme is to control mouse and keyboard for not operating unit the order is transferred form the sub-controller.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้ด้วยดีนั้น เนื่องมาจากความร่วมมือของสมาชิกทุกคนในกลุ่ม ขอกราบขอบพระคุณอาจารย์อำพล ทองระอา รองศาสตราจารย์ พีระวุฒิ สุวรรณจันทร์ และคณาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่าน ที่คอยชี้แนะจุดบกพร่องและได้กรุณาช่วยเหลือในการให้คำปรึกษาต่างๆ ตลอดจนถึงข้อมูลและอุปกรณ์ในการสร้างโครงงานและการจัดทำปริญญานิพนธ์ฉบับนี้เป็นอย่างสูง ขอกราบขอบพระคุณรองศาสตราจารย์ วิสุทธิ์ สุนทรกนกพงศ์ ที่ได้เอื้อเฟื้อสถานที่ในการทำโครงงาน ขอขอบเพื่อนๆ ครุศาสตร์วิศวกรรม สาขาวิศวกรรมโทรคมนาคมรุ่นที่ 27 ทุกท่านที่คอยให้คำแนะนำและเสียสละเวลาอันมีค่ามาช่วยในการทำตัวชิ้นงานและปริญญานิพนธ์ ขอขอบคุณคณะครุศาสตร์อุตสาหกรรมที่ได้เอื้อเฟื้อสถานที่และสิ่งอำนวยความสะดวกต่างๆ เป็นอย่างยิ่ง

ขอกราบขอบพระคุณบิดา มารดา ผู้มีพระคุณและบูรพคณาจารย์ทุกท่านที่ได้สั่งสอนและประสิทธิ์ประสาทวิชาในทุกๆ ด้านทำให้คณะผู้จัดทำได้มีความรู้ในด้านต่างๆ ตลอดเรื่อยมาจนถึงปัจจุบัน และสุดท้ายนี้ขอขอบคุณเพื่อนๆ สาขาวิศวกรรมโทรคมนาคมทุกคนที่คอยเป็นกำลังใจให้ด้วยดีเสมอมา

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันไม่มีระบบควบคุมการใช้งานภายในห้องปฏิบัติการคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม เราต้องการศึกษาวิธีการเก็บข้อมูลการใช้คอมพิวเตอร์ของนักศึกษาภาควิชาครุศาสตร์วิศวกรรมว่ามีจำนวนการเข้าใช้บริการและระยะเวลาที่เข้าใช้บริการมากน้อยเพียงใดเพื่อเป็นสถิติการใช้งาน โดยระบบที่มีอยู่ทั่วไปในปัจจุบันใช้ระบบของระบบโครงข่าย (LAN) ควบคู่กับโปรแกรมต่างๆ ในการบันทึกและการตรวจสอบข้อมูลการใช้งาน แต่ปัญหาของระบบของระบบโครงข่าย (LAN) นี้ก็คือการเกิดทราฟฟิคในระบบโครงข่าย (LAN) ทำให้เกิดความล่าช้าในการใช้งานเครื่องคอมพิวเตอร์ จึงต้องมีการพัฒนาระบบที่สามารถเก็บข้อมูลการใช้งานคอมพิวเตอร์ได้โดยไม่ต้องใช้ระบบโครงข่าย (LAN)

1.2 จุดมุ่งหมายของโครงงาน

คณะผู้จัดทำได้สร้างระบบควบคุมการใช้งานคอมพิวเตอร์เพื่อต้องการทราบถึงวิธีการเก็บข้อมูลการใช้คอมพิวเตอร์ของนักศึกษาภาควิชาครุศาสตร์วิศวกรรมโดยระบบต้องอ่านแถบบาร์โค้ดจากบัตรนักศึกษาที่เข้าใช้และสามารถเก็บข้อมูลการใช้งานของคอมพิวเตอร์แต่ละเครื่องได้และตัวระบบจะควบคุมคอมพิวเตอร์โดยผ่านสายเคเบิ้ลแบบเฉพาะ (RS485) โดยที่ตัวเครื่องคอมพิวเตอร์จะมีส่วนควบคุมย่อยโดยใช้ไมโครคอนโทรลเลอร์ MCS51 เป็นตัวควบคุมการเปิดปิดคอมพิวเตอร์โดยทำงานคู่กับโปรแกรมเปิดหรือปิดคอมพิวเตอร์เพื่อควบคุมการทำงานเพื่อสั่งการให้คอมพิวเตอร์เปิดหรือปิด

1.3 สมมุติฐานของการจัดทำโครงงาน

ได้เครื่องควบคุมการใช้งานคอมพิวเตอร์ที่สามารถเก็บข้อมูลการใช้คอมพิวเตอร์ของนักศึกษาภาควิชาครุศาสตร์วิศวกรรมได้และเมื่อติดตั้งตัวเครื่องควบคุมการใช้งานคอมพิวเตอร์แล้วสามารถบันทึกข้อมูลการใช้งานคอมพิวเตอร์ของภาควิชาครุศาสตร์วิศวกรรมได้จริง

1.4 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. รับบัตรบาร์โค๊ดนักศึกษาเพื่อใช้ควบคุมชุดควบคุมหลัก

2. การแสดงสถานการณ์ใช้คอมพิวเตอร์จะแสดงผลด้วยหลอด LED และตัวแสดงผลสถานะมีจำนวนไม่เกิน 20 เครื่องและและสามารถขยายเพิ่มได้ไม่เกิน 40 เครื่อง

3. มีการสร้างส่วนควบคุมย่อยสำหรับคอมพิวเตอร์ไม่เกิน 5 เครื่อง

4. ติดตั้งระบบสายจากตัวควบคุมไปยังคอมพิวเตอร์ไม่เกิน 20 ชุด

5. เครื่องควบคุมหลักและเครื่องควบคุมย่อยติดต่อกันโดยผ่านพอร์ต RS485

1.5 ขั้นตอนของการทำโครงงาน

โครงงานนี้จะประกอบไปด้วยฮาร์ดแวร์และซอฟต์แวร์ ซึ่งการทำงานในระยะแรกจะเริ่มต้นจากการทำฮาร์ดแวร์ก่อนหลังจากนั้นเมื่อสร้างฮาร์ดแวร์ในส่วนระบบรับข้อมูลแล้วก็จะไปเริ่มทำในส่วนของซอฟต์แวร์การเขียนโปรแกรมในส่วนของข้อมูลผู้ที่สามารถใช้งาน การบันทึกการใช้งานของแต่ละบุคคล และสั่งการในการเปิดปิดคอมพิวเตอร์ ขั้นต่อไปคือทำในส่วนของฮาร์ดแวร์ที่ใช้ควบคุมการเปิดและปิดเครื่องคอมพิวเตอร์และเมื่อทำโครงงานเสร็จเรียบร้อยแล้วจะให้ผู้ทรงคุณวุฒิทำงานตรวจสอบเพื่อหาประสิทธิภาพของตัวชิ้นงานต่อไป

1.6 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อสะดวกต่อการศึกษาและทำความเข้าใจในแต่ละบทจะประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ ขีดความสามารถของโครงงานและเนื้อหาในบทต่างๆโดยสังเขป

บทที่ 2 ประกอบด้วยทฤษฎีต่างๆเกี่ยวกับ บาร์โค้ด การเชื่อมต่อระหว่างคอมพิวเตอร์ผ่านพอร์ต RS 485 และไมโครคอนโทรลเลอร์

บทที่ 3 กล่าวถึงเนื้อหาที่เกี่ยวกับแผนผังการทำงานของโครงงาน ผังวงจรต่างๆ ที่ใช้ในโครงงานตลอดจนการออกแบบและการสร้างส่วนประกอบต่างๆ ของโครงงานเช่น โครงสร้างเครื่องบันทึกการใช้งานคอมพิวเตอร์ในส่วนของชุดควบคุมหลักและเครื่องควบคุมย่อย โปรแกรมที่ใช้ควบคุมเมาส์และคีย์บอร์ดพร้อมทั้งการทำงานของส่วนประกอบต่างๆ โดยละเอียด

บทที่ 4 ประกอบด้วย การทดลองและผลการทดลองของเครื่องบันทึกการใช้งานคอมพิวเตอร์เมื่อรับค่าจากบัตรนักศึกษาการเลือกเครื่องที่ต้องการเข้าใช้และการเลิกใช้งานเมื่อนำไปใช้งาน

บทที่ 5 เป็นการสรุปผลการจัดทำโครงงาน ปัญหาที่เกิดขึ้นและแนวทางในการแก้ไขรวมทั้งแนวทางการพัฒนา

ภาคผนวก ก ภาพเครื่องต้นแบบ การติดตั้ง การเชื่อมต่อกับอุปกรณ์อื่นๆ ขณะใช้งานจริง

ภาคผนวก ข รายละเอียดวงจรและแผ่นวงจรพิมพ์

ภาคผนวก ค รายการอุปกรณ์ที่ใช้งานในแต่ละวงจร

ภาคผนวก ง รายละเอียดและคุณสมบัติของอุปกรณ์ที่สำคัญในโครงงาน

ภาคผนวก จ ผังงาน (Flowchart) ของโปรแกรมทั้งหมด

ภาคผนวก ฉ รหัสต้นฉบับของโปรแกรม

ภาคผนวก ช คู่มือการใช้งานของเครื่องควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

เนื้อหาของปริญญานิพนธ์ในบทนี้เป็นทฤษฎี และหลักการที่จะนำมาใช้ประกอบการสร้างโครงงานโดยประกอบด้วยบาร์โค้ด พอร์ตในการสื่อสารต่างๆ และไมโครคอนโทรลเลอร์

2.2 รหัสแท่ง (Barcode)

รหัสแท่ง (Barcode) คือ การแทนข้อมูลที่เป็นรหัสเลขฐานสอง (Binary codes) ในรูปแบบของแถบสีดำ และขาวที่มีความกว้างของแถบที่ต่างกันแถบที่มีสี และความกว้างที่แตกต่างกันนี้จะมีค่าเป็นตัวเลขที่แตกต่างกัน และมาตรฐานสากลได้กำหนดค่าไว้เทคโนโลยีบาร์โค้ดถูกนำมาใช้ทดแทนในส่วนการบันทึกข้อมูล (Data entry) จากเดิมที่มนุษย์ใช้คีย์บอร์ดในการบันทึกข้อมูลการบันทึกด้วยคีย์บอร์ดมีอัตราความผิดพลาดอยู่ประมาณ 1 ใน 10 ยกกำลัง 7 หรือบันทึกข้อมูลผิดพลาด 1 ตัวอักษรในทุกๆ 100 ตัวอักษร และเมื่อเปลี่ยนมาใช้ระบบบาร์โค้ดแทนในขั้นตอนการบันทึกข้อมูล อัตราการเกิดความผิดพลาดจะลดลงเหลือเพียง 1 ใน 1,010,000,000 ตัวอักษรเท่านั้น

2.2.1 ความหมาย

คำว่า Barcode ในภาษาไทยนั้นราชบัณฑิตสถาน (2546) กำหนดให้ใช้คำว่ารหัสแท่งแต่คนทั่วไปเรียกบาร์โค้ดทับศัพท์จากคำภาษาอังกฤษโดยตรงคำว่ารหัสแท่งหรือบาร์โค้ดนั้นมีผู้ให้คำจำกัดความไว้แตกต่างกันดังนี้ปัมทมาภรณ์ (2542) กล่าวว่า รหัสแท่ง หรือที่เรียกทับศัพท์ว่าบาร์โค้ด (Barcode) นั้นหมายถึงระบบสัญลักษณ์หรือเครื่องหมายประจำตัวของสินค้าแทนเลขรหัสซึ่งเป็นระบบที่เป็นมาตรฐานสากลประกอบด้วยแถบสีดำสลับขาวหลายๆ เส้นซึ่งมีความหนาบางไม่เท่ากันแถบเส้นเหล่านี้ถูกกำหนดและสร้างขึ้นโดยตัวเลขทั้งชุดเพื่อบ่งบอกประเทศที่ผลิตผู้ผลิตและชนิดสินค้าปีเตอร์ (2545) กล่าวว่าบาร์โค้ดคือการแทนข้อมูลที่เป็นรหัสเลขฐานสอง (Binary codes) ในรูปแบบของแถบสีดำและขาวที่มีความกว้างของแถบที่ต่างกันแถบที่มีสีและความกว้างที่แตกต่างกันนี้จะมีค่าเป็นตัวเลขที่แตกต่างกันและมาตรฐานสากลได้กำหนดค่าไว้ Rowley (1993) อธิบายความหมายของบาร์โค้ดโดยละเอียดว่าบาร์โค้ดนั้นใช้อย่างแพร่หลายทั้งโรงงานและห้องสมุดบาร์โค้ดแต่ละอันแสดงตัวเลขบาร์โค้ดเป็นรูปแบบที่มีแท่งหนาและบางแบ่งโดยพื้นที่ว่างหนาและบางการแบ่งช่องว่างและความหนาหรือบางนั้นมีความสำคัญบาร์โค้ดสามารถพิมพ์ออกมาได้หลายขนาดและสีบาร์โค้ดจะถูกอ่านข้อมูลได้จากเครื่องอ่านบารโค้ดบาร์โค้ดเหมาะสำหรับการป้อนข้อมูลเพื่อแยกแยะแต่ละรายการข้อมูลที่ป้อนเข้าไปจะถูกส่งไปยังฐานข้อมูลเพื่อส่งข้อมูลที่ต้องการกลับมาระบบบาร์โค้ดง่ายต่อการดำเนินงานและข้อผิดพลาดต่ำการเปลี่ยนแปลงรายละเอียดต่างๆ ในฐานข้อมูลทำได้จากจุดศูนย์กลางจากความหมายต่างๆ ข้างต้นสามารถสรุปได้ว่ารหัสแท่งคือสัญลักษณ์ที่มีรูปแบบเป็นแท่ง (Bar) ที่มีความหนา และบางแตกต่างกัน

เพื่อใช้แทนข้อมูล (Code) ตัวเลขฐานสองสามารถแยกแยะหรือระบุให้สิ่งของแต่ละชิ้นมีความแตกต่างกันได้ ความหมายของบาร์โค้ดโดยลักษณะทางกายภาพว่าเป็นแถบรหัสรูปลายทางสีดำ (Bar) และแถบขาว (Space) เรียงขนานกันคล้ายทางม้าลายแต่ขนาดความหนาและแถบระยะห่างมีลักษณะเป็นสัญลักษณ์เฉพาะ (Symbology) ที่กำหนดขึ้นตามเลขที่กำกับอยู่ขนาดของแถบบาร์โค้ดมีความยาวแตกต่างกันขึ้นกับการใช้งาน และรูปแบบบาร์โค้ดที่พบเป็นกันมากในสินค้าผลิตภัณฑ์ต่างๆ อาจอยู่ในรูปแถบกระดาษกาว ภาพพิมพ์กระดาษห่อผลิตภัณฑ์ หรือเป็นส่วนหนึ่งของผลิตภัณฑ์เลยก็ได้บาร์โค้ดที่ปรากฏบนสินค้าต่างๆ นั้นไม่ได้แสดงข้อมูลการขายแต่เป็นตัวเลขอ้างอิง (Reference number) ที่กำหนดขึ้นเพื่อแยกชนิดของสินค้านั้นๆ ส่วนรายละเอียดต่างๆ เช่น บริษัทผู้ผลิต ประเภทของสินค้า ปริมาณ เลขที่ของผลิตภัณฑ์และอื่นๆ จะเก็บไว้ในเซิร์ฟเวอร์ซึ่งจะถูกนำข้อมูลออกมาเมื่อแถบบาร์โค้ดถูกอ่านโดยเครื่องอ่านบาร์โค้ด

2.2.2 ประวัติความเป็นมาของบาร์โค้ด

เริ่มในปีค.ศ.1970 สหรัฐอเมริกามีการจัดตั้งคณะกรรมการเฉพาะกิจด้านพาณิชย์ขึ้นสำหรับค้นคว้ารหัสมาตรฐาน และสัญลักษณ์ที่สามารถช่วยกิจการด้านอุตสาหกรรมค.ศ.1973 คณะกรรมการเฉพาะกิจฯ ได้จัดพิมพ์บาร์โค้ดระบบ UPC (Uniform Product Code) ขึ้นเป็นครั้งแรกสำหรับติดบนสินค้าต่างๆ ในวงการอุตสาหกรรมใช้สำหรับควบคุมยอดการขายและสินค้าคงคลังค.ศ.1975 ประเทศทางยุโรปจัดตั้งคณะทำงานด้านวิชาการขึ้นเพื่อสร้างระบบบาร์โค้ดเรียกว่า EAN (European Article Number) และปีค.ศ.1977 สมาคม EAN ถูกจัดตั้งขึ้นครอบคลุมประเทศในยุโรปและประเทศอื่นๆ ของโลกยกเว้นอเมริกาเหนือ ต่อมาได้เปลี่ยนชื่อเป็น IANA (International Article Numbering Association) แต่อักษรย่อยังคงใช้ EAN ระบบบาร์โค้ดของยุโรปถูกพัฒนาจากระบบ UPC และได้พัฒนาให้มีความสามารถเช่นเดียวกับระบบ UPC ประเทศไทยนำระบบ EAN เข้ามาใช้ตั้งแต่ปีพ.ศ.2530 โดย Thai Product Numbering Association-TPNA ได้รับการแต่งตั้งให้เป็นนายทะเบียน สำหรับรับสมัครสมาชิกระบบบาร์โค้ดและทำหน้าที่รับจดทะเบียนสมาชิกบาร์โค้ดในระบบ EAN ทั้งนี้เพื่อสินค้าที่ผลิตภายในประเทศได้มาตรฐานสากลและเพื่อประโยชน์แก่ผู้ผลิตผู้ส่ง ออก ผู้ซื้อ ผู้ค้าปลีกจนปีพ.ศ.2536 TPNA ได้โอนสิทธิ์การเป็นนายทะเบียนให้กับสภาอุตสาหกรรมแห่งประเทศไทยรหัสประจำของประเทศไทยคือ 855 ระบบ EAN ตามระบบสากลของ EAN International ภายใต้การบริหารงานของสถาบันสัญลักษณ์รหัสแห่งไทย EAN Thailand หมายเลข 855 จะช่วยสร้างภาพลักษณ์ที่ดีให้กับสินค้าไทยในตลาดต่างประเทศ โดยที่ผู้ประกอบธุรกิจจะสามารถตรวจสอบได้ว่า 855 เป็นของประเทศไทย ช่องทางหรือโอกาสทางการตลาดของสินค้าไทยจะกว้างขึ้นสามารถนำสินค้าไทยไปสู่ตลาดใหญ่ๆ ได้โดยง่ายรหัสแห่งประจำประเทศเปรียบเสมือนการประกาศเอกราชในทางระบบเศรษฐกิจเพราะสินค้าจากกว่า 91 ประเทศทั่วโลกใช้ระบบ EAN มีเลขหมายประจำแต่ละประเทศหมายเลขจะพิมพ์อยู่ 2 หรือ 3 ตำแหน่งแรกที่อยู่ใกล้รหัสแห่งการใช้หมายเลข EAN ประจำประเทศทำให้คู่แข่งทางการค้าทั่วโลกทราบว่าประเทศไทยเป็นประเทศที่มีมาตรฐานการผลิตสูงระดับหนึ่ง สามารถแจ้งแหล่งผลิตสินค้าสินค้าส่งออกที่ควรใช้รหัสแห่งคือสินค้าอุปโภคบริโภคเกือบทุกชนิดหรือสินค้าสำเร็จรูปต่างๆ ซึ่งจะช่วยสร้างความเชื่อถือให้กับผู้ซื้อหรือผู้นำเข้า

ในประเทศได้เป็นอย่างดีโดยส่วนใหญ่จะพิมพ์รหัสแท่งลงบนสินค้าเลยหรืออาจไม่จำเป็นต้องติดบนตัวสินค้าก็ได้

2.2.3 หน่วยงานที่ใช้บาร์โค้ด

1. ห้างสรรพสินค้า
2. คลังสินค้า
3. ผู้ผลิตสินค้า/โรงงานอุตสาหกรรม
4. ธุรกิจโทรคมนาคม
5. ห้องสมุด
6. มหาวิทยาลัย
7. โรงพยาบาล
8. งานพัสดุและไปรษณีย์

2.2.4 ส่วนประกอบของบาร์โค้ด

ทวีชัย (2544) อธิบายถึงส่วนประกอบของบาร์โค้ดไว้ว่าสัญลักษณ์ของบาร์โค้ดที่ใช้กันมีการกำหนดขึ้นมาหลายรูปแบบตามมาตรฐานของแต่ละองค์กร และตามจุดประสงค์ของการใช้งานแต่โดยทั่วๆ ไปแล้วบาร์โค้ดจะมีส่วนประกอบต่างๆ ดังต่อไปนี้

1. Quiet Zone เป็นบริเวณที่ว่างเปล่าไม่มีการพิมพ์ข้อความใดๆ โดยจะอยู่ก่อนและหลังบาร์โค้ด

2. Start/Stop Character เป็นบริเวณแถบแท่งหรือช่องว่าง เพื่อเตรียมสั่งให้เซนเซอร์เริ่มต้นหรือหยุดบาร์โค้ด

3. Data เป็นบริเวณแถบแท่งหรือช่องว่างที่แทนข้อมูลต่างๆ ที่เราต้องการ

4. Check Digit เป็นบริเวณแถบแท่งที่ไว้สำหรับเก็บค่าตัวเลขเพื่อตรวจสอบในข้อมูลส่วน Data เพื่อให้มั่นใจว่าถูกต้องแม่นยำ

2.2.5 หลักการทำงานของบาร์โค้ด

รหัสบาร์โคดประกอบด้วย 3 ส่วนดังนี้

1. ส่วนลายเส้น ซึ่งเป็นลายเส้นสีขาว (โปร่งแสง) และสีดำมีขนาดความกว้างของลายเส้นตามมาตรฐานแต่ละชนิดของบาร์โค้ด

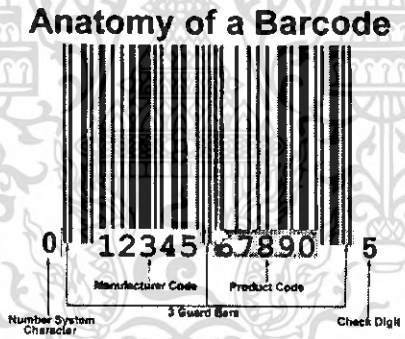
2. ส่วนข้อมูลตัวอักษรเป็นส่วนที่แสดงความหมายของชุดข้อมูลลายเส้นสำหรับให้อ่านเข้าใจ

3. ส่วนแถบว่าง เป็นส่วนที่เครื่องอ่านบาร์โค้ดใช้กำหนดขอบเขตของบาร์โค้ดและกำหนดค่าให้กับสีขาว (ความเข้มข้องการสะท้อนแสงในสีของพื้นผิวแต่ละชนิดที่ใช้แทนสีขาว) โดยทุกเส้นจะมีความยาวเท่ากันเรียงตามลำดับในแนวนอนจากซ้ายไปขวาแถบสีทั้งสีขาวและสีดำที่มีความกว้างจะแทนค่าเป็น 1 และแถบสีที่มีความแคบ (หรือมองด้วยตาเหมือนเป็นเส้นตรงเล็กๆ) ทั้งขาวและดำจะมีค่าเป็น 0 แถบขาวและดำที่มีลักษณะและชื่อที่ใช้คือแถบสีดำที่มีความกว้างมากกว่าเรียกว่า Wide Bar ถ้ามีความกว้างน้อยเรียกว่า Narrow Bar

ช่องว่างหรือแถบสีขาวที่มีความกว้างมากกว่าเรียกว่า Wide Space ถ้ามีความกว้างน้อยเรียกว่า Narrow Space

2.2.6 โครงสร้างของบาร์โค้ด

บาร์โค้ดประกอบด้วยแถบสีดำและสีขาวโดยความกว้างของแถบสีดำสลับขาวเป็นรหัสแทนข้อมูล เรียงจากซ้ายไปขวาตัวอย่างของโครงสร้างบาร์โค้ดแสดงอยู่ในรูปที่ 2.1 การถอดรหัสจำเป็นต้องทาบความกว้างของแถบดำและแถบขาวนำไปเทียบกับตารางมาตรฐานเครื่องอ่านบาร์โค้ดประกอบด้วยหัวอ่านอินฟาเรดแบบปากกา และแบบวงจรถอดรหัสการใช้งานเริ่ม ต้นด้วยการกวาดหัวอ่านผ่านบาร์โค้ดซึ่งหัวอ่านจะมีตัวตรวจจับแสงสะท้อนไปจุดชนวนวงจรอิเล็กทรอนิกส์ทำให้เกิดคลื่นสัญญาณไฟฟ้าแบบพัลส์โดยความกว้างของรูปคลื่นจะเป็นสัดส่วนกับความกว้างของแถบโค้ด ต่อจากนั้นวงจรถอดรหัสจะตรวจสอบความกว้างของรูปคลื่นแล้วนำไปเปรียบเทียบกับแถบขาวดำทั้งหมดที่แทนข้อมูลตัวเลขหรือตัวอักษรโดยปกติเครื่องอ่านจะต่อเข้ากับคอมพิวเตอร์ดังนั้นวงจรภายในเครื่องอ่านจะส่งข้อมูลตัวเลขที่ถอดรหัสได้ไปยังคอมพิวเตอร์เพื่อประมวลผลต่อไป



รูปที่ 2.1 โครงสร้างของบาร์โค้ด

2.2.7 เซนเซอร์อ่านบาร์โค้ด

ทวีชัย (2544) กล่าวว่าเซนเซอร์อ่านบาร์โค้ดส่วนใหญ่แล้วแบ่งออกเป็น 2 ชนิดคือ

1. แบบ Laser จะใช้อ่านบาร์โค้ดที่ติดในสายการผลิต ซุปเปอร์มาร์เก็ต และคลังสินค้าหลักการทำงานคือ ลำแสงเลเซอร์ถูกปล่อยออกจากเลเซอร์ไดโอดมากระทบกับกระจกแบบหลายเหลี่ยมเพื่อที่จะสแกนบาร์โค้ด เมื่อลำแสงเลเซอร์กระทบบาร์โค้ดจะกระจายออก และถูกส่งมาที่โฟโต้ไดโอดลักษณะของลำแสงที่กระจายตามบาร์โค้ดจะถูกแปลงไปเป็นสัญญาณอะนาลอก จากนั้นทำการแปลงสัญญาณเป็นดิจิทัลลักษณะของสัญญาณดิจิทัลจะขึ้นอยู่กับขนาดของแท่ง และที่ว่างในแถบบาร์โค้ด จากนั้นก็จะแปลงรหัสเป็นข้อมูลผ่านพอร์ตคอมพิวเตอร์เพื่อให้คอมพิวเตอร์ไปประมวลผลหรือเก็บข้อมูลไว้ใช้

2. แบบ CCD จะใช้อ่านบาร์โค้ดที่ติดชิ้นงานที่มีขนาดเล็ก เช่น หลอดทดลอง แผงวงจรที่ชิ้นงานกับตัวอ่านใกล้กัน หลักการทำงานคือหลอด LED จะเปล่งแสงมากระทบบาร์โค้ดแล้วสะท้อนมาที่เซนเซอร์ CCD Image เพื่อจับภาพของบาร์โค้ดขึ้นมาเป็นข้อมูลเก็บไว้ใช้งานต่อไป การสแกนของเซนเซอร์ อ่านบาร์โค้ดจะมี 2 แบบคือแบบ Singer Scan จะปล่อยลำแสงขวางในการสแกน 1 แถว ซึ่งเหมาะแก่การเคลื่อนที่ของบาร์โค้ดแบบ Picket Fence Direction และแบบ Raster Scan จะปล่อยลำแสงขวางในการสแกนหลายแถวแม้บาร์โค้ดที่พิมพ์จะคุณภาพไม่ดีก็สามารถอ่านค่าได้ถูกต้อง การสแกนแบบนี้เหมาะสำหรับการเคลื่อนที่ของบาร์โค้ดแบบ Ladder Direction

ตารางที่ 2.1 เปรียบเทียบเซนเซอร์อ่านบาร์โค้ดแบบ Laser และแบบ CCD

เปรียบเทียบ	แบบ Laser	แบบ CCD
ข้อดี	อ่านได้ในระยะไกล มุมในการอ่านกว้าง สามารถอ่านวัตถุเคลื่อนที่ได้	ขนาดเล็ก ราคาไม่แพง อายุการใช้งานยาวนานกว่า
ข้อเสีย	ราคาแพง	ไม่เหมาะกับชิ้นงานเคลื่อนที่

2.2.8 เครื่องอ่านบาร์โค้ด

เครื่องอ่านบาร์โค้ดสามารถแบ่งตามลักษณะการใช้งานได้ดังนี้

1. Moving Bean Scanner เครื่องอ่านอยู่กับที่แต่แสงฉายกวาดไปที่สินค้าเพื่อหาบาร์โค้ดที่กำกับบนสินค้านั้น

2. Fixed Bean Scanner เครื่องอ่านอยู่กับที่ลำแสงไม่เคลื่อนที่สินค้าเคลื่อนที่ผ่านจุดที่แสงฉาย

3. Hand Held Scanner เครื่องอ่านที่ต้องใช้คนควบคุม และถือได้เหมาะสำหรับการอ่านบาร์โค้ดของสินค้าที่มีขนาดใหญ่เคลื่อนที่ยากเช่นม้วนกระดาษใหญ่ที่ผลิตจากโรงงานดังแสดงในรูปที่ 2.2

4. Wand Scanner เครื่องอ่านที่ให้แสงสีแดงอินฟราเรดในการอ่านต้องใช้เครื่องอ่านสัมผัสกับแถบบาร์โค้ด

5. Hand Held Laser Scanner เครื่องอ่านที่มีหลักการทำงานแบบ Moving Bean Scanner ที่ให้แสงเลเซอร์

รูปที่ 2.2 ตัวอย่างเครื่องอ่านบาร์โค้ดประเภท Hand Held Scanner

2.2.9 การเลือกใช้สีสำหรับบาร์โค้ด

เครื่องอ่านบาร์โค้ดทำงานโดยการแยกความกว้างระหว่างพื้นที่สว่างและพื้นที่มืดขณะที่เครื่องอ่านรหัสแท่งโดยมีแสงส่องจากเครื่องไปตกกระทบบริเวณแท่งสีทึบซึ่งเป็นพื้นที่มืดและบริเวณแท่งสีจางซึ่งเป็นพื้นที่สว่างแสงที่สะท้อนออกมาจากพื้นที่มืดจะน้อยกว่าพื้นที่สว่าง เนื่องจากสีมืดมีเปอร์เซ็นต์ในการสะท้อนต่ำ เช่น บาร์เป็นสีดำพื้นเป็นสีขาวแสงที่สะท้อนกลับไปยังเครื่องอ่านจะถูกแปลงเป็นรหัสไปยังเครื่องคอมพิวเตอร์ สีที่ควรใช้สำหรับพื้นที่ว่างด้านหลัง (Background) ของแท่ง เช่น ขาว แดง ส้ม และเหลืองเป็นต้นสีที่ควรเลือกใช้เป็นแท่งบาร์เช่น ดำ น้ำเงิน ม่วง และเขียวเป็นต้น

2.2.10 คู่สีที่ใช้กับบาร์โค้ดไม่ได้

สีเหลืองบนสีขาว สีส้มบนสีขาว สีแดงบนสีขาว สีน้ำตาลอ่อนบนสีขาว สีแดงบนสีเขียว สีน้ำเงินบนสีเขียว สีดำบนสีเขียว สีแดงบนสีน้ำเงิน สีแดงบนสีน้ำตาลอ่อน สีดำบนสีน้ำเงิน สีดำบนสีน้ำตาลเข้ม สีเหลืองบนสีขาว สีดำบนสีทอง สีทองบนสีขาว สีส้มบนสีทอง และสีแดงบนสีทอง

2.2.11 เทคโนโลยีการพิมพ์

ปัจจุบันการพิมพ์บาร์โค้ดสามารถทำได้ระบบด้วยกันคือ

1. Thermal Transfer เป็น 7 ระบบการพิมพ์แบบใช้ความร้อนพิมพ์ลงกระดาษที่เคลือบด้วยสารไวความร้อนเรียกเครื่องพิมพ์ชนิดนี้ว่า Thermal Printer หรือ Bar Code Printer ระบบนี้เหมาะสำหรับการพิมพ์งานจำนวนน้อยและต้องการความรวดเร็วแต่จะเสื่อมคุณภาพได้ง่ายเมื่อถูกแสงอุลตร้าไวโอเล็ต

2. Direct Transfer เป็นระบบการพิมพ์แบบไม่ต้องใช้ผ้าหมึกเส้นบาร์ใหญ่กว่าช่องว่างทำให้อ่านยากและคุณภาพเสื่อมลงเมื่อถูกความร้อน

3. Impact dot Matrix ระบบการพิมพ์ชนิดนี้เส้นบาร์จะเล็กกว่าช่องว่างทำให้อ่านยาก

4. Laser Toner ระบบการพิมพ์ชนิดนี้ทั้งเส้นบาร์และช่องว่างจะใหญ่กว่าปกติ

5. Ink Jet เป็นระบบที่ใช้สำหรับการพิมพ์ลงบนบรรจุภัณฑ์โดยตรงพิมพ์โดยการพ่นหมึกปริมาณมากเพื่อให้เส้นที่ได้คมชัดเหมือนการพิมพ์ Kot Matrix

6. Laser Etching การพิมพ์ระบบนี้เส้นบาร์จะเล็กกว่าช่องว่างทำให้อ่านยาก

7. Wet Ink เป็นการพิมพ์ระบบออฟเซตซึ่งจะมีปัญหาเส้นบาร์ใหญ่หรือบวมเพราะปล่อยหมึกมากไปเวลาพิมพ์ แต่สามารถทำให้คมชัดได้หากช่างมีประสบการณ์การพิมพ์ระบบนี้ต้องใช้ความประณีตมาก โดยการนำ Barcode Film Master ไปถ่ายทำแม่พิมพ์ ซึ่งประกอบด้วยขั้นตอนยุ่งยากทำให้ได้งานที่มีคุณภาพดีที่สุด

2.2.12 ข้อสังเกตในการพิมพ์บาร์โค้ด

การพิมพ์บาร์โค้ดบนฉลาก สติ๊กเกอร์ หรือป้ายแขวนบนตัวสินค้าเป็นงานที่ยากถ้าต้องการงานพิมพ์คุณภาพสูงต้องประกอบด้วยหลายดังนี้

1. ความคมชัดของเส้นแต่ละเส้นถ้าเส้นไม่คมขาดหายแหว่งบาร์โค้ดนั้นจะอ่านไม่ออก

2. สีที่เลือกใช้โดยทั่วไปดีที่สุดคือพิมพ์ตัวบาร์โค้ดสีดำบนพื้นสีขาว ซึ่งจะทำให้อ่านง่ายเนื่องจากเครื่องอ่านอาศัยหลักการสะท้อนแสงของเส้นทึบและพื้นสว่างถ้าใช้คู่สีผิดอาจทำให้อ่านไม่ออกควรหลีกเลี่ยงการใช้สีสะท้อนแสงในการพิมพ์แท่งรหัสสินค้าและพื้นที่ว่างหลังแท่งรหัสเพราะสีสะท้อนแสงจะสะท้อนแสงใส่เครื่องอ่านทำให้อ่านยากหรืออ่านไม่ได้เลยโดยเฉพาะสีน้ำตาลเข้มถือว่าเป็นสีมืดจึงใช้เป็นสีมืดของแท่งบาร์ได้แต่สีน้ำตาลมีส่วนผสมของสีแดงด้วยต้องระมัดระวังไม่ให้สีแดงมากเกินไปอาจทำให้เครื่องสแกนเนอร์ไม่สามารถอ่านได้

3. ขนาดของบาร์โค้ด บาร์โค้ดทุกระบบจะมีขนาดมาตรฐานเรียกว่า 100% สามารถย่อลงมาใช้ในขนาดต่ำสุด 80% คือย่อลง 20% ถ้าต่ำกว่านั้นอาจอ่านไม่ออกไม่แนะนำให้ทำส่วนขยายขนาดถ้าขยายก็ไม่ควรเกิน 200% ความสูงของเส้นบาร์ไม่ควรต่ำกว่า 1.5 เซนติเมตร ถ้าต้องการใช้ฟิล์มบาร์โค้ดขนาดไหนต้องระบุตอนสั่งไม่ควรนำฟิล์มไปย่อหรือขยายอีก

4. พื้นที่ด้านข้าง 2 ข้างของตัวบาร์โค้ด ซึ่งเรียกว่า Quiet Zone ต้องมีเนื้อที่ 10 เท่าของแท่งรหัสที่เล็กที่สุดหรือมากกว่า 3.6 ตารางมิลลิเมตร

5. ระบบการพิมพ์ควรพิมพ์ด้วยระบบออฟเซตซึ่งจะให้คุณภาพงานพิมพ์ออกมาดีที่สุด

6. ผลิตภัณฑ์ที่เป็นวัสดุโปร่งใสแสงจากเครื่องอ่านจะมองผ่านทะลุวัสดุโปร่งใสทำให้เกิดปัญหาในการอ่าน

7. ผลิตภัณฑ์ที่มีหีบห่อเป็นผ้าไม่สามารถพิมพ์บาร์โค้ดได้เนื่องจากเส้นใยที่ได้รับการทอจะเป็นปัญหาในการอ่านวิธีที่ดีที่สุดคือพิมพ์บาร์โค้ดลงบนป้ายสินค้า

2.2.13 มาตรฐานบาร์โค้ด

บาร์โค้ดสามารถแบ่งออกตามชนิดของข้อมูลได้ดังนี้

2.2.13.1 Numeric-only barcodes (ตัวเลขเพียงอย่างเดียว)

1. EAN-13 European Article Numbering (EAN) ประเทศในทวีปยุโรป และเอเชีย

2. EAN-8 European Article Numbering (EAN) สำหรับสินค้าขนาดเล็ก

3. UPC-A Universal Product Code (UPC) ใช้ในประเทศสหรัฐอเมริกาและแคนาดา

4. Code 11 ใช้สำหรับติดเครื่องมือสื่อสาร

5. Interleaved 2 of 5 ใช้อย่างกว้างขวางในงานอุตสาหกรรมขนส่งท่าอากาศยาน

6. Standard 2 of 5

7. Codabar ใช้ในระบบ และธนาคารเลือด

8. Plessey

9. MSI

10. PostNet ใช้ในประเทศสหรัฐอเมริกาสำหรับงานคัดแยกไปรษณีย์

2.2.13.2 Alphanumeric barcodes (ตัวอักษรและตัวเลข)

1. Code 39

2. Code 93

3. Code 128

4.4 LOGMARS

2.2.13.3 Two-Dimensional barcodes (2 มิติ)

1. PDF 417

2. DataMatrix Maxicode กำหนดความยาวใช้สำหรับ United Parcel Service ใช้แยกแยะทีบห่ออัตโนมัติ

3. QR Code ใช้ในการควบคุมวัตถุดิบ และยืนยันการสั่ง Data Code

4. Code 49

5. 16K

2.2.13.4 Industry Standards for Barcodes and Labels (มาตรฐานอุตสาหกรรม)

1. ISBN International Standard Book Number

2. ISSN and the SISAC Barcode International Standard Serial

3. Numbering OPC ของ Optical Industry Association

4. UCC/EAN-128 Code 128

5. UPC Shipping Container Symbol ITF-14

2.2.13.5 บาร์โค้ดมาตรฐานที่ใช้กันอยู่ปัจจุบันมีประมาณ 11 ระบบดังนี้

1. UPC-Uniform Product Code บาร์โค้ดระบบแรกของโลกพัฒนาและทดลองใช้ครั้งแรกในปีค.ศ.1949 โดยชาวอเมริกันชื่อ Mr.Norm Woodland และ Mr.BarnardSilvers และสามารถใช้ได้อย่างสมบูรณ์เมื่อปีค.ศ.1973 โดย Uniform Code Council ตั้งอยู่ที่เมือง Daytor รัฐโอไฮโอสหรัฐอเมริการะบบนี้

นิยมใช้กันมากในประเทศแคนาดาและสหรัฐอเมริการะบบตัวอย่างบาร์โค้ดแบบ UPC แสดงในรูปที่ 2.3 โดยมาตรฐานของ UPC มี 4 ประเภทดังนี้

1.1 แบบย่อมี 8 หลักใช้กับสินค้าที่มีข้อมูลน้อยเรียกว่า UPC-E

1.2 แบบมาตรฐานมี 12 หลักเรียกว่า UPC-A

1.3 แบบเพิ่มตัวเลข 2 หลักถ้า UPC-A เก็บข้อมูลไม่พอเรียกว่า UPC-A+2

1.4 แบบเพิ่มตัวเลข 5 หลักถ้าต้องการเก็บข้อมูลจำนวนมากเรียกว่า UPC-A+5

2. EAN-European Article Number เป็นระบบบาร์โค้ดที่กลุ่มประเทศในแถบยุโรปพัฒนาขึ้นใช้เพราะเริ่มสนใจการใช้บาร์โค้ดระบบ UPC ของประเทศสหรัฐอเมริกาแล้วเสร็จเมื่อปีค.ศ.1976 จนปัจจุบัน EAN เป็นระบบที่ใช้กันอย่างแพร่หลายทั้งยุโรปออสเตรเลียและเอเชียรวมทั้งประเทศไทยด้วยรูปที่ 2.3 แสดงตัวอย่างของบาร์โค้ดแบบ EAN ระบบ EAN มี 4 ประเภทดังนี้

2.1 แบบย่อมี 8 หลักเหมาะสำหรับธุรกิจขนาดเล็กที่ใช้เก็บข้อมูลไม่มากเรียกว่า EAN-8

2.2 แบบมาตรฐานมี 13 หลักเรียกว่า EAN-13

รูปที่ 2.3 ตัวอย่าง UPC

2.3 แบบเพิ่มตัวเลข 2 หลักถ้า EAN-13 เก็บข้อมูลไม่พอเรียกว่า EAN-13+2

2.4 แบบเพิ่มตัวเลข 5 หลักถ้าต้องการเก็บข้อมูลจำนวนมากเรียกว่า EAN-13+5

รูปที่ 2.4 ตัวอย่าง EAN

3. CODE 39 ระบบนี้ได้รับการพัฒนาขึ้นในปีค.ศ.1974 โดย Dr.David Allais และ Ray Steven ชาวอเมริกันวัตถุประสงค์เพื่อนำไปใช้ในธุรกิจอุตสาหกรรมระบบนี้สามารถใช้รวมกับตัวอักษรได้เป็นระบบแรกและเก็บข้อมูลได้ปริมาณมากตัวอย่างบาร์โค้ด CODE 39 ดังแสดงอยู่ในรูปที่ 2.5

รูปที่ 2.5 ตัวอย่าง CODE 39

4. ITF-INTERLEAVE 2 of 5 เรียกย่อๆ ว่า ITF เป็นบาร์โค้ดตัวใหญ่ใช้สำหรับหีบบรรจุสินค้าหรือเรียกว่า Case code ตัวอย่างบาร์โค้ด ITF ดังแสดงอยู่ในรูปที่ 2.6

รูปที่ 2.6 ตัวอย่าง ITF

5. CodaBar ถูกพัฒนาขึ้นมาใช้กับธุรกิจเวชกรรมในปีค.ศ.1972 และนิยมใช้ในระบบงานห้องสมุดตัวอย่างบาร์โค้ด CodaBar ดังแสดงอยู่ในรูปที่ 2.7

รูปที่ 2.7 ตัวอย่าง CodaBar

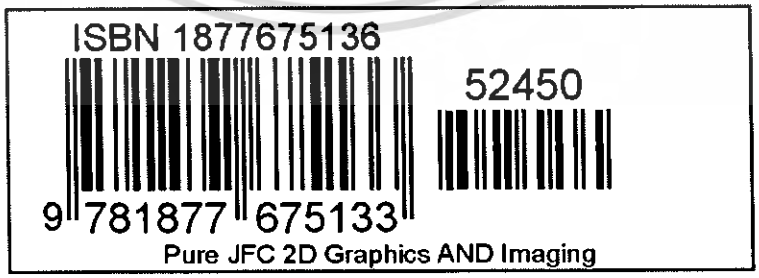
6. CODE 128 ได้ถูกพัฒนาขึ้นและยอมรับว่าได้ใช้เป็นทางการในสหรัฐอเมริกาเมื่อปี ค.ศ.1981 นิยมใช้ในวงการดีไซน์เนอร์และแฟชั่นปัจจุบันกำลังเริ่มนิยมใช้ในสหรัฐอเมริกา

7. CODE 93 ได้เริ่มพัฒนาขึ้นในปีค.ศ.1982 ปัจจุบันเริ่มนิยมใช้ในสหรัฐอเมริกา

8. CODE 49 ได้เริ่มพัฒนาขึ้นในปี ค.ศ.1987 โดย Dr.Davis Allais ผู้คิดค้น CODE 39 ได้ปรับปรุงพัฒนาให้บรรจุข้อมูลได้มากขึ้นด้วยพื้นที่เท่าเดิม

9. CODE 16k เหมาะสำหรับอุตสาหกรรมผลิตสินค้าที่มีขนาดเล็กมากพื้นที่ในการใส่บาร์โค้ด น้อย เช่น อุปกรณ์อะไหล่เครื่องไฟฟ้า

10. ISBN/ISSN International Standard Book Number/International Standard Serial Number ใช้สำหรับหนังสือและนิตยสารตัวอย่างบาร์โค้ด ISBN และ ISSN ดังแสดงอยู่ในรูปที่ 2.8



รูปที่ 2.8 ตัวอย่าง ISBN และ ISSN

11. EAN/UCC 128 (shipping container code) เป็นระบบใหม่ซึ่งเป็นการร่วมมือกันระหว่าง EAN ของยุโรป และ UCC ของสหรัฐอเมริกา โดยนำเอาระบบ EAN มาใช้ร่วมกับ CODE 128 เพื่อบอกรายละเอียดของสินค้ามากขึ้นเช่น วันเดือนปีที่ผลิตครั้งที่ผลิตวันที่สั่งซื้อมีกี่สีกี่ขนาดเป็นต้น

2.3 พอร์ต RS-485

2.3.1 แนวความคิดของระบบควบคุม

ในปัจจุบันการสื่อสารและการควบคุมอุปกรณ์มีความจำเป็นมากขึ้นเนื่องจากจำนวนของอุปกรณ์ที่มากขึ้นความต้องการในการใช้งาน ความสะดวกสบาย ความรวดเร็วในการทำงานที่มีเพิ่มมากขึ้น แนวทางที่ตอบสนองความต้องการดังกล่าวแนวทางทางหนึ่งคือการสร้างระบบควบคุมที่สามารถทำงานได้ดี กล่าวคือใช้งานได้ง่าย มีความยืดหยุ่น มีความผิดพลาดเกิดขึ้นน้อย อีกทั้งยังต้องสามารถขยายระบบที่ต้องการควบคุมออกไปได้ จึงทำให้เกิดแนวคิดที่จะสร้างระบบควบคุมโดยการประยุกต์ใช้งาน RS-485 ขึ้น

การกำหนดขอบเขตของระบบควบคุมนั้นจะถือว่าสิ่งที่ต้องการควบคุมทั้งหมดนั้นเป็นส่วนหนึ่งของระบบควบคุม ในที่นี้จะขอยกตัวอย่างเป็นห้องหลายๆ ห้องในอาคารชั้นหนึ่ง ซึ่งภายในห้องๆ หนึ่งนั้นจะมีอุปกรณ์ที่ต้องการควบคุมอาทิเช่น หลอดไฟ เครื่องปรับอากาศ และเครื่องใช้ไฟฟ้าอื่นๆ โดยแต่ละอุปกรณ์ที่ต้องการควบคุม (Controlled Device : CD) จะต่อเข้ากับตัวลูกข่าย (Module Box : MB) ที่ใช้ควบคุมอุปกรณ์นั้นและตัวลูกข่ายหลายๆ ตัวจะต่อรวมเข้ากับตัวควบคุม (Master Control Box : MCB) ซึ่งตัวควบคุมนี้จะทำหน้าที่เสมือนกับเป็นสถานีจ่ายงานย่อยเนื่องจากระบบควบคุมนี้สามารถเชื่อมต่อตัวควบคุมได้มากกว่าหนึ่งตัวทำให้สามารถแบ่งการควบคุมอุปกรณ์ออกเป็นกลุ่มย่อย (Section Zone) ทำให้ง่ายและสะดวกต่อการควบคุมอุปกรณ์ที่ต้องการควบคุมทีเดียวทั้งหมด

แนวคิดของระบบควบคุมนั้นจะมีการแบ่งห้องต่างๆ ในชั้นของตัวอาคารออกเป็นกลุ่มซึ่งอุปกรณ์ที่ต้องการควบคุมหนึ่งตัวจะทำการเชื่อมต่อเข้ากับตัวลูกข่ายหนึ่งตัว และตัวลูกข่ายจะถูกแบ่งเป็นกลุ่มตามการเชื่อมต่อเข้ากับตัวควบคุมดังจะเห็นได้จากในรูปที่มีการแบ่งห้องในชั้นออกเป็น 4 กลุ่มคือ

1. กลุ่มตัวควบคุมที่ 1 (MCB-1 Area) เชื่อมต่อกับตัวลูกข่ายที่ 1-4 (MB 1-4)

2 .กลุ่มตัวควบคุมที่ 2 (MCB-2 Area) เชื่อมต่อกับตัวลูกข่ายที่ 5-7 (MB 5-7)

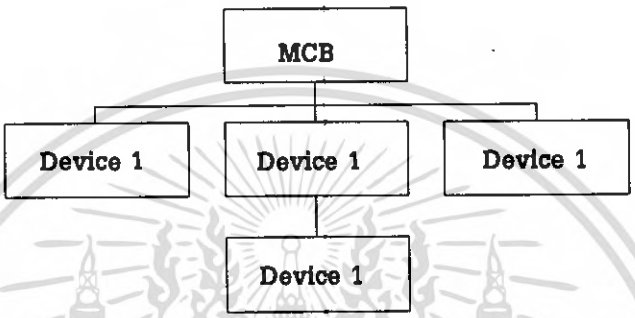
3 .กลุ่มตัวควบคุมที่ 3 (MCB-3 Area) เชื่อมต่อกับตัวลูกข่ายที่ 8-9 (MB 8-9)

4. กลุ่มตัวควบคุมที่ 4 (MCB-4 Area) เชื่อมต่อกับตัวลูกข่ายที่ 10-13 (MB 10-13)

โดยอุปกรณ์ที่ต้องการควบคุมจะเป็นอุปกรณ์ประเภทรับข้อมูล (Input) หรือแสดงผล (Output) ก็ได้เนื่องจากระบบควบคุมนี้เป็นการสื่อสารแบบสองทางในคนละเวลา (Half-duplex) ทำให้ระบบควบคุมนี้สามารถรับและส่งข้อมูลได้บนคู่สายคู่หนึ่ง

2.3.2 โครงสร้างของระบบควบคุม

การทำงานของระบบควบคุมทั้งหมดจะถูกควบคุมโดยคอมพิวเตอร์ส่วนบุคคล (Personal Computer : PC) โดยจะเป็นการควบคุมแบบรวมศูนย์ (Centralize Control) ซึ่งคอมพิวเตอร์ส่วนบุคคลทำหน้าที่เป็นตัวหลักในการติดต่อกับผู้ใช้งาน (User Interface) และควบคุมระบบโดยการจ่ายงานให้กับตัวควบคุมที่เชื่อมต่อกันเป็นเครือข่าย

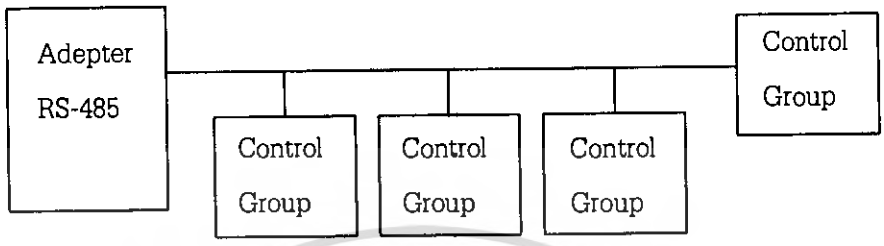


รูปที่ 2.9 โครงสร้างของระบบควบคุม

รูปที่ 2.9 แสดงให้เห็นถึงโครงสร้างของการการเชื่อมต่อระบบควบคุมโดยการใช้การเชื่อมต่อ (Interface) กับคอมพิวเตอร์ส่วนบุคคลโดยผ่านทางพอร์ตสื่อสาร (Communication Port : Com Port) ของคอมพิวเตอร์ส่วนบุคคล ซึ่งในที่นี้จะเลือกใช้พอร์ต D-25 (Com Port2) ในการเชื่อมต่อ โดยการสื่อสารของระบบควบคุมจะเป็นดังนี้คือ สัญญาณที่ออกจากพอร์ตสื่อสารจะถูกดัดแปลงจากสัญญาณตามมาตรฐาน RS-232c ให้เป็นสัญญาณตามมาตรฐาน RS-485 โดยอแดปเตอร์ (Adapter) จากนั้นสัญญาณจะถูกส่งผ่านสายสัญญาณ (RS-485 BUS) ไปยังตัวควบคุมและตัวลูกข่ายตามลำดับ

ส่วนการเชื่อมต่อตัวควบคุมเป็นเครือข่าย (Network) นั้นจะเป็นการเชื่อมต่อเครือข่ายแบบบัส (Bus-type Network) โดยไม่ต้องมีการปิดปลายสัญญาณ (Terminate) เนื่องจากอัตราส่งข้อมูลที่ช้ามีค่าต่ำสัญญาณรบกวนจึงมีผลต่อการส่งสัญญาณในระบบต่ำซึ่งจะไม่ทำให้เกิดความเสียหายในการสื่อสาร การเชื่อมต่อเป็นเครือข่ายของระบบควบคุมจะแสดงให้เห็นดังรูปที่ 2.10 โดยจะจัดตัวควบคุมกับตัวลูกข่ายที่เชื่อต่อเข้าด้วยกันเป็นกลุ่มเรียกว่ากลุ่มตัวควบคุม (MCB Area) และการเชื่อมต่อกลุ่มตัวควบคุมเข้าเป็นเครือข่ายจะใช้สายสัญญาณเพียงคู่เดียวในการเชื่อมต่อ ในกลุ่มของตัวควบคุมนั้นจะประกอบไปด้วย ตัวควบคุม ตัวลูกข่ายและอุปกรณ์ที่ต้องการควบคุม โดยตัวลูกข่ายแต่ละตัวจะเป็นอิสระต่อกันในการทำงาน กล่าวคือสามารถสลับที่หรือต่อตัวลูกข่ายเพิ่มเข้ามาในกลุ่มของตัวควบคุมได้โดยไม่มีผลกระทบต่อระบบควบคุมแต่มีข้อจำกัดคือตัวลูกข่ายจะเชื่อมต่อกับอุปกรณ์ที่ต้องการควบคุมได้แบบหนึ่งต่อหนึ่งเนื่องจากตัวลูก

ข่ายหนึ่งตัวสามารถควบคุมอุปกรณ์ได้เพียงหนึ่งอุปกรณ์เท่านั้นโดยจะแสดงการเชื่อมต่อกลุ่มตัวควบคุมดังรูปที่ 2.11



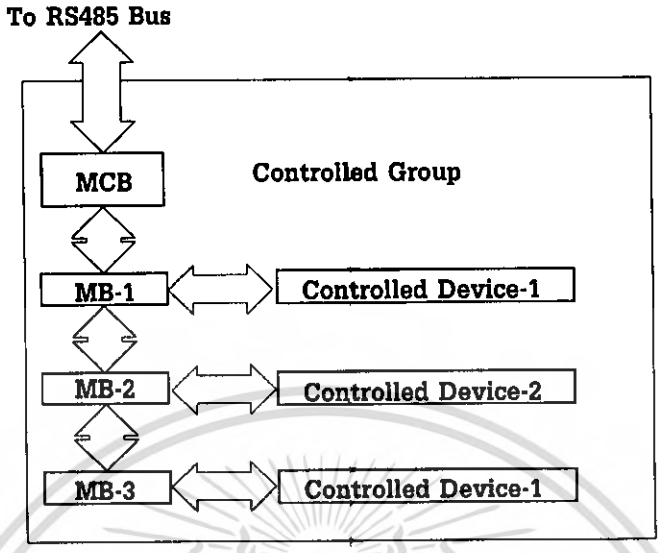
รูปที่ 2.10 การเชื่อมต่อเครือข่ายของระบบควบคุม

ในกลุ่มของตัวควบคุมนั้นจะประกอบไปด้วย ตัวควบคุม ตัวลูกข่ายและอุปกรณ์ที่ต้องการควบคุม โดยตัวลูกข่ายแต่ละตัวจะเป็นอิสระต่อกันในการทำงานกล่าวคือสามารถสลับที่หรือต่อตัวลูกข่ายเพิ่มเข้ามาในกลุ่มของตัวควบคุมได้โดยไม่มีผลกระทบต่อระบบควบคุมแต่มีข้อจำกัดคือตัวลูกข่ายจะเชื่อมต่อกับอุปกรณ์ที่ต้องการควบคุมได้แบบหนึ่งต่อหนึ่งเนื่องจากตัวลูกข่ายหนึ่งตัวสามารถควบคุมอุปกรณ์ได้เพียงหนึ่งอุปกรณ์เท่านั้นโดยจะแสดงการประยุกต์ใช้งานในการควบคุมระบบงานทั่วๆ ไป ในรูปที่ 2.12

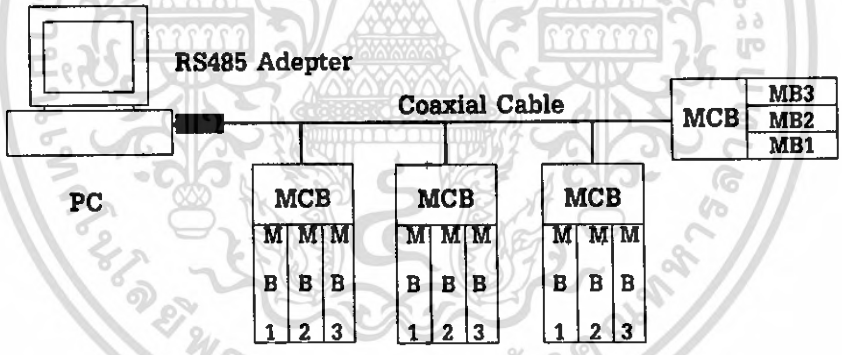
2.3.3 การนำระบบควบคุมไปประยุกต์ใช้งาน

ระบบควบคุมนี้นอกจากจะนำไปประยุกต์ใช้งานในการควบคุมอุปกรณ์ภายในตัวอาคารแล้วยังสามารถนำระบบนี้ไปประยุกต์ใช้ในการควบคุมระบบงานประเภทอื่นได้อีกด้วยโดยอาศัยการปรับเปลี่ยนส่วนประกอบที่ใช้ในการควบคุมอุปกรณ์ให้มีความเหมาะสมกับงานที่จะใช้ซึ่งส่วนประกอบสำคัญที่ต้องมีการปรับเปลี่ยนประกอบด้วย 2 ส่วนใหญ่ๆ คือ ส่วนของตัวลูกข่ายที่ใช้ควบคุมอุปกรณ์จะต้องมีการปรับเปลี่ยนให้ตัวลูกข่ายมีความสามารถในการควบคุมอุปกรณ์ได้ดีมีการทำงานที่เหมาะสมและเข้ากันได้กับอุปกรณ์ที่ต้องการควบคุมโดยการปรับเปลี่ยนตัวลูกข่ายนี้จะขึ้นอยู่กับคุณสมบัติของอุปกรณ์ที่ต้องการควบคุมว่าเป็นอุปกรณ์ชนิดใดมีการทำงานแบบใด และอาศัยอะไรในการควบคุมอุปกรณ์ เนื่องจากปัจจัยเหล่านี้มีผลต่อประสิทธิภาพในการทำงานของระบบควบคุม

ส่วนของโปรแกรมที่ใช้ควบคุมการทำงานของอุปกรณ์จะต้องมีการปรับเปลี่ยนให้โปรแกรมสามารถในการควบคุมตัวลูกข่ายได้อย่างมีประสิทธิภาพโดยมีความเข้ากันได้กับตัวลูกข่ายและมีการทำงานที่สอดคล้องกับตัวลูกข่ายที่ใช้ควบคุมอุปกรณ์



รูปที่ 2.11 การเชื่อมต่อ ตัวควบคุม ตัวลูกข่าย และอุปกรณ์ที่ต้องการควบคุม



รูปที่ 2.12 การประยุกต์ใช้งานในการควบคุมระบบงานทั่วๆ ไป

2.3.4 ลักษณะของการสื่อสาร

ในการสื่อสารหรือการส่งข้อมูลโดยใช้คอมพิวเตอร์ส่วนบุคคลนั้นมีรูปแบบในการสื่อสารที่สำคัญอยู่ 2 แบบคือ

2.3.4.1 การสื่อสารแบบอนุกรม (Serial Communication)

เป็นการสื่อสารโดยการส่งข้อมูลทีละบิต (Bit) ผ่านสายสัญญาณเส้นเดียวจนครบทั้ง 8 บิตหรือไบท์ (byte) โดยจะส่งบิตต่ำ (LSB) ออกไปก่อนซึ่งการสื่อสารและการส่งข้อมูลแบบอนุกรมจะแสดงให้เห็นได้ดังรูปที่ 2.13 และ 2.14

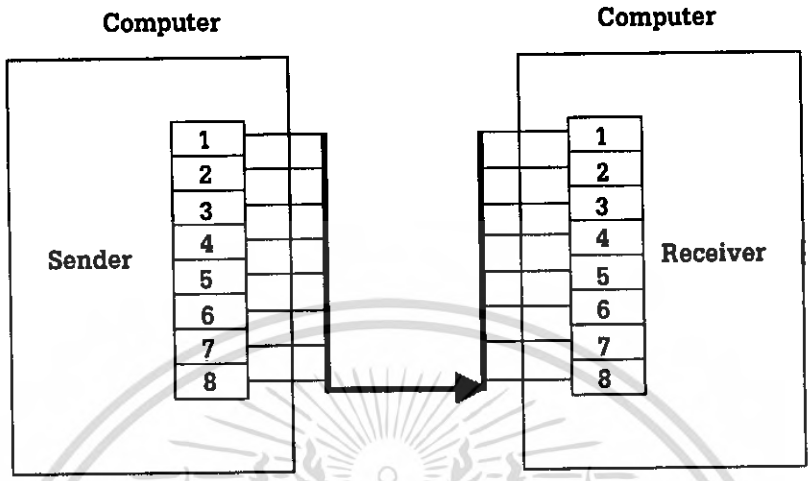
2.3.4.2 การสื่อสารแบบขนาน (Parallel Communication)

เป็นการสื่อสารโดยการส่งข้อมูลไปทีเดียวพร้อมๆ กันทั้ง 8 บิตผ่านสายสัญญาณทั้ง 8 เส้น ซึ่งการสื่อสารและการส่งข้อมูลแบบขนานจะแสดงให้เห็นได้ดังรูป 2.15 และ 2.16 จะเห็นได้ว่าการสื่อสารแบบขนานมีข้อดีคือทำให้สามารถส่งข้อมูลได้ทีละมากๆ และรวดเร็วกว่าการส่งแบบอนุกรม แต่การสื่อสาแบบขนานมีข้อจำกัดคือไม่สามารถส่งข้อมูลในระยะไกลๆ ได้และยังต้องใช้สายสัญญาณหลายเส้นในการส่งข้อมูลทำให้สิ้นเปลืองกว่าการสื่อสารแบบอนุกรมรวมทั้งทำให้ไม่สะดวกในการใช้งาน

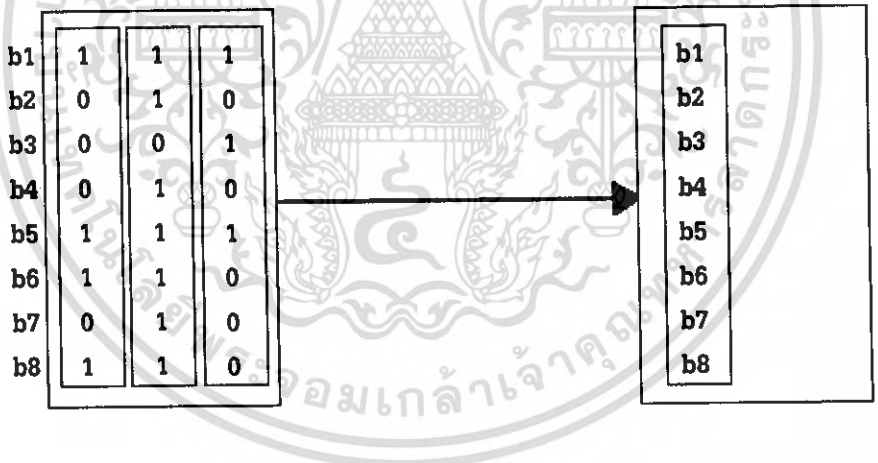
ตัวอย่างของการสื่อสารแบบอนุกรมเช่น การสื่อสารด้วย Modem และเมาส์ตัวอย่างของการสื่อสารแบบขนานเช่น เครื่องพิมพ์ (Printer) และการสื่อสารทางพอร์ตขนาน (EDP Printer port) เป็นต้น

2.3.5 ลักษณะสัญญาณที่ใช้ในการอินเทอร์เฟสตามมาตรฐานต่างๆ

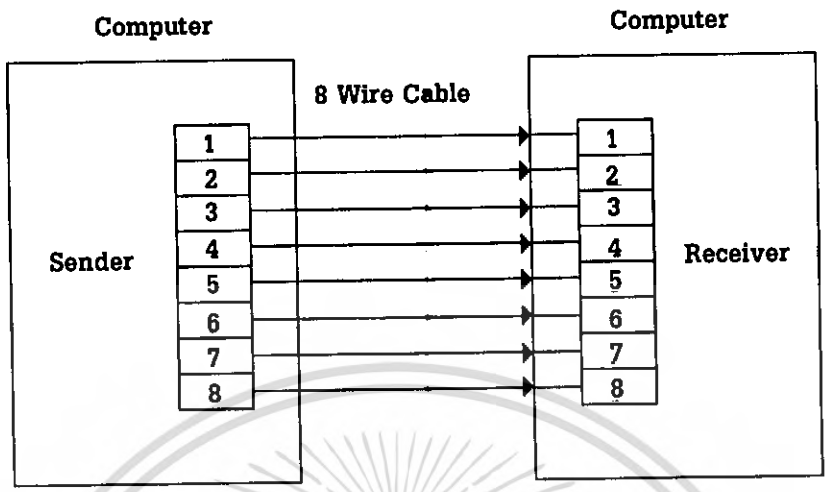
ปกติการอินเทอร์เฟสที่ออกแบบและจัดให้ใช้ควบคุมและตรวจสอบ (Sense) ข้อมูลจะเป็นการอินเทอร์เฟสที่ใช้สัญญาณดิจิตอลเป็นหลัดสัญญาณพวกนี้จะถูกส่งและรับมาจากรีจิสเตอร์อินพุต/เอาต์พุต อินพุตของ Interrupt-request พอร์ตที่ใช้ในการทำ DMA และไทม์เมอร์/เคาน์เตอร์การอินเทอร์เฟสที่กล่าวมานี้ทุกตัวเป็นการใช้สัญญาณดิจิตอลที่มีระดับสัญญาณแบบ TTL ในการอินเทอร์เฟสดังนั้นถ้าเรานำอุปกรณ์อื่นที่ใช้ระดับสัญญาณ TTL ในการอินเทอร์เฟสด้วยเช่นกันการทำอินเทอร์เฟสเข้ากับเครื่อง PC จะทำได้โดยตรง แต่ในหลายกรณีสัญญาณที่ใช้ไม่ได้เป็นระดับสัญญาณ TTL หรือไม่ได้เป็นสัญญาณดิจิตอล และระยะทางที่ทำการอินเทอร์เฟสมีระยะห่างมากซึ่งก่อให้เกิดปัญหาหลายประการเราสามารถแก้ปัญหาเหล่านี้ได้โดยการใช้การอินเทอร์เฟนแบบต่างๆ มาตรฐาน RS-232-C เป็นมาตรฐานที่ได้รับการพัฒนามานานและถูกใช้งานกันอย่างแพร่หลาย เราใช้ RS-232-C เชื่อมต่อ DTE (Data Terminal Equipment) เข้ากับ DCE (Data Communication Equipment) เช่น การต่อเทอร์มินัลเข้ากับโมเด็มมาตรฐาน RS-232-C กล่าวถึงลักษณะทางกลลักษณะของสัญญาณไฟฟ้า และลักษณะการทำงานที่ใช้ในการอินเทอร์เฟสตัวอย่างของอุปกรณ์ที่ใช้ในการอินเทอร์เฟสตามมาตรฐาน RS-232-C ได้แก่ เทอร์มินัล พล็อตเตอร์ ลอจิกอนาไลเซอร์ (Logic analyzer) และเครื่องพิมพ์ถ้าการประยุกต์ใช้งานของเราต้องการอินเทอร์เฟสอุปกรณ์ เข้ากับการอินเทอร์เฟสมาตรฐาน RS-232-C เราจำเป็นต้องแปลงระดับสัญญาณ TTL ให้เป็นระดับสัญญาณแบบอื่น ซึ่งรายละเอียดของระดับสัญญาณที่ใช้ได้กล่าวไว้ในมาตรฐาน RS-232-C ลักษณะที่สัญญาณที่ใช้ในการอินเทอร์เฟสมาตรฐาน RS-232-C ใช้สัญญาณเพียงเส้นเดียวในการส่งสัญญาณโดยสัญญาณที่ส่งไปได้ทิศทางเดียวในกรณีที่อัตราในการส่งข้อมูลมีค่าเท่ากับ 20 kbps (กิโลบิตต่อวินาที) ซึ่งค่านี้เป็ค่าสูงสุดที่ใช้ในการส่งข้อมูล ระยะทางที่ใช้ส่งข้อมูลไม่ควรเกิน 50 ฟุต (ตามข้อกำหนดในมาตรฐาน) สำหรับการแทนแรงดันของระดับสัญญาณ (ในที่นี้เราแทนระดับสัญญาณด้วยลอจิกบวก) มีข้อกำหนดดังนี้ "1" แทนระดับแรดงดันที่มีค่าระหว่าง +5 โวลต์ถึง +15 โวลต์ "0" แทนระดับแรงดันที่มีค่าระหว่าง -5 โวลต์ถึง -15 โวลต์ วงจรแปลงระดับแรงดันตัวอย่างของวงจรที่ใช้แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณที่กำหนดไว้ในมาตรฐาน RS-232-C และแปลงกลับจากระดับแรงดันในมาตรฐาน RS-232-C ไปเป็นระดับสัญญาน TTL



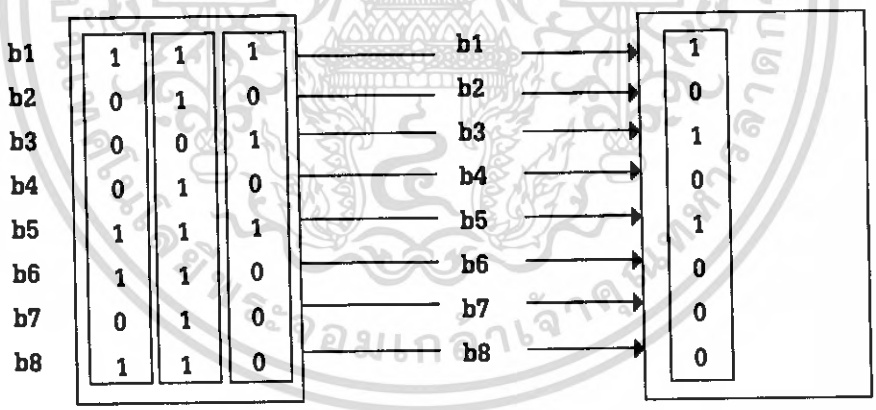
รูปที่ 2.13 ลักษณะการสื่อสารแบบอนุกรม



รูปที่ 2.14 การส่งข้อมูลแบบอนุกรม



รูปที่ 2.15 ลักษณะการสื่อสารแบบขนาน



รูปที่ 2.16 การส่งข้อมูลแบบขนาน

ลักษณะของอินเทอร์เฟสที่ใช้มาตรฐาน RS-423 มาตรฐาน RS-423 ใช้สายสัญญาณเส้นเดียวในการส่งสัญญาณ โดยสัญญาณที่ส่งไปได้ทิศทางเดียว อัตราเร็วในการส่งข้อมูลมีค่าสูถึง 100 kbps ที่ระยะห่าง 40 ฟุต ตัวรับข้อมูลเป็นแบบ Balanced-1 ดังนั้นตัวรับข้อมูล (Receiver) จึงรับข้อมูลแบบขยายความแตกต่างของสัญญาณระหว่างสายกราวนด์กับตัวขับสัญญาณ (Driver) การทำเช่นนี้ช่วยแก้ปัญหาในกรณีที่เกิดความแตกต่างระหว่างแรงดันที่ กราวด์ของตัวรับข้อมูลกับตัวขับสัญญาณ สำหรับการแทนระดับแรงดันนั้นลอจิก "I"

แทนระดับแรงดันที่อยู่ระหว่าง +4 โวลต์ถึง +6 โวลต์ ส่วนลอจิก "0" แทนระดับแรงดันที่มีค่าระหว่าง -4 โวลต์ถึง -6 โวลต์ วงจรแปลงระดับแรงดันของมาตรฐาน RS-423 วงจรที่ใช้แปลงระดับแรงดันจากระดับสัญญาณ TTL ไปเป็นระดับสัญญาณที่ใช้ในมาตรฐาน RS-423 และแปลงระดับสัญญาณจาก RS-423 กลับไปเป็นระดับสัญญาณ TTL

ลักษณะการอินเทอร์เฟสที่ใช้มาตรฐาน RS-422 มาตรฐาน RS-422 ได้พัฒนามาจากมาตรฐาน RS-423 ทำให้อัตราเร็วในการส่งข้อมูลมีค่าสูงขึ้นและระยะทางที่ใช้ส่งข้อมูลระหว่างตัวส่งและตัวรับมีระยะทางไกลขึ้น ลักษณะสัญญาณที่ใช้ในอินเทอร์เฟสมาตรฐาน RS-422 ใช้การส่งข้อมูลในลักษณะของ One-way balanced-line โดยอัตราเร็วในการส่งข้อมูลมีค่าสูงถึง 10 Mbps ที่ระยะทางห่างเท่ากับ 1,000 ฟุต ในกรณีที่ส่งข้อมูลในอัตราเร็วต่ำกว่า 10 Mbps ระยะห่างที่ใช้ในการส่งข้อมูลสามารถขยายได้ถึง 4,000 ฟุต ระดับแรงดันที่ส่งจากตัวขับสัญญาณมีค่าระหว่าง 2 โวลต์ถึง 6 โวลต์ นอกจากนี้ตัวขับสัญญาณสามารถจับสัญญาณที่มีระดับต่ำกว่า 200 mV ได้วงจรแปลงระดับแรงดันที่ใช้ในมาตรฐาน RS-422 วงจรที่ใช้แปลงระดับสัญญาณ TTL ไปเป็นระดับสัญญาณที่ใช้มาตรฐาน RS-422 และแปลงจาก RS-422 กลับไปเป็นระดับสัญญาณ TTL นอกจากนี้ผู้ผลิตบางบริษัทได้ทำวงจรขับสัญญาณเป็นแบบ tri-state ทำให้เราสามารถส่งข้อมูลได้สองทิศทางบนสายคู่เดียว (Single pair) ซึ่งเป็นมาตรฐานใหม่คือ RS-485 คุณสมบัติข้อนี้ทำให้เราสามารถใช้มาตรฐาน RS-485 ในเน็ตเวิร์คที่มีโครงสร้างแบบ Multidrop ซึ่งอุปกรณ์หลายๆ ตัวสามารถรับและส่งข้อมูลแบบ Half duplex บนสายคู่เดียวได้

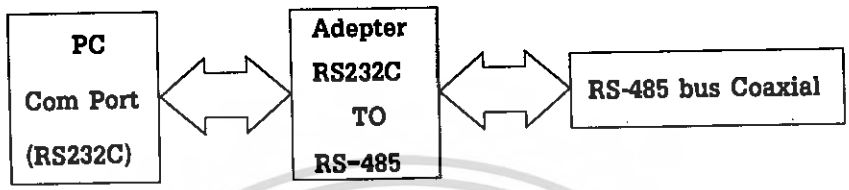
ลักษณะการอินเทอร์เฟสแบบ Current-loop การอินเทอร์เฟสแบบ Current-loop มีข้อดีคือสามารถส่งข้อมูลได้ในระยะทางไกลๆ และค่าใช้จ่ายในการอินเทอร์เฟสแบบนี้มีราคาไม่สูงนักหลักการของอินเทอร์เฟสแบบนี้มีดังนี้ เมื่อลูปถูกปิดวงจร (Current-loop) ระดับแรงดันจะถูกเปลี่ยนเป็นกระแสตามสมการ V=IR เมื่อลูปถูกเปิดวงจรจะไม่มีกระแสไหลในลูปเนื่องจากวงจรของการอินเทอร์เฟสแบบนี้มีค่าอิมพีแดนซ์ต่ำดังนั้นจึงทนต่อสัญญาณรบกวนได้ดีเรามักใช้การอินเทอร์เฟสแบบนี้ในกรณีต้องเดินสายผ่านบริเวณที่มีสัญญาณรบกวนมากๆ นอกจากนี้เราสามารถใช้ในการอินเทอร์เฟสแบบนี้แยกกราวน์ของระบบสองระบบออกจากกัน ซึ่งเป็นการแยก Current-loop ของวงจรผลิตสัญญาณ (Transmitter) ออกจารวงจรรับสัญญาณ (Receiver) วงจรนี้สามารถใช้ข้อมูลได้ในอัตราเร็ว 50 kbps โดยระยะทางที่ใช้ส่งข้อมูลมีค่าได้ไม่เกิน 3,000 ฟุต ตัวจำกัดระยะทางที่ใช้ในการส่งข้อมูลคือความต้านทานของสายที่ประกอบอยู่เป็นลูปซึ่งไม่ควรมีค่าเกิน 30 โอห์ม

2.3.6 ตัวแปลงสัญญาณและตัวควบคุม

2.3.6.1 โครงสร้างของตัวแปลงสัญญาณ

การทำงานของตัวแปลงสัญญาณ (Adapter) นั้นจะต้องทำการปรับสัญญาณระหว่างพอร์ตสื่อสารของคอมพิวเตอร์ส่วนบุคคลซึ่งเป็นมาตรฐาน RS-232-C กับสายสัญญาณที่ใช้ในการส่งข้อมูล (RS-485 BUS) ซึ่งเป็นมาตรฐาน RS-485 และเนื่องจากการสื่อสารตามมาตรฐาน RS-485 นั้นสามารถทากรสื่อสารได้ 2

ทิศทาง ดังนั้นตัวแปลงสัญญาณที่จะต้องสามารถแปลงสัญญาณจากมาตรฐาน RS-485 ให้กลับไปเป็นสัญญาณตามมาตรฐาน RS-232-C ได้ด้วยเพื่อให้สามารถรับและส่งข้อมูลผ่านทางตัวแปลงสัญญาณได้ซึ่งโครงสร้างการทำงานของตัวแปลงสัญญาณสามารถแสดงให้เห็นได้ดังรูป 2.17



รูปที่ 2.17 โครงสร้างของตัวแปลงสัญญาณ

2.3.6.2 การออกแบบตัวแปลงสัญญาณ

จากโครงสร้างของตัวแปลงสัญญาณซึ่งทำให้ทราบหน้าที่และลักษณะการทำงานของตัวแปลงสัญญาณเนื่องจากข้อมูลเหล่านี้มีความจำเป็นในการออกแบบตัวแปลงสัญญาณที่จะใช้กับระบบควบคุมเพื่อให้ระบบควบคุมสามารถสื่อสารกันได้อย่างมีประสิทธิภาพ การเชื่อมต่อกับพอร์ตสื่อสารของคอมพิวเตอร์ส่วนบุคคลจะเลือกใช้พอร์ตสื่อสารแบบอนุกรม 25 เข็ม (DB-25) ซึ่งสามารถทำการรับและส่งข้อมูลได้แบบอนุกรม โดยลักษณะของสัญญาณจะเป็นไปตามมาตรฐาน RS-232-C

วงจรแดปเตอร์ วงจรของตัวแปลงสัญญาณจากมาตรฐาน RS-232-C ไปเป็นสัญญาณตามมาตรฐาน RS-485 ในส่วนวงจรของตัวแปลงสัญญาณจะมีการเชื่อมต่อขาที่ทำหน้าที่ในการส่งและรับสัญญาณขอพอร์ตสื่อสารแบบอนุกรมเข้ากับขาของ IC MAX-232 ทำหน้าที่แปลงไฟระดับ +12 V เป็นระดับ TTL เพื่อให้ใช้ได้กับ IC DS75176B เพื่อแปลงจากมาตรฐาน RS-232-C เป็นมาตรฐาน RS-485

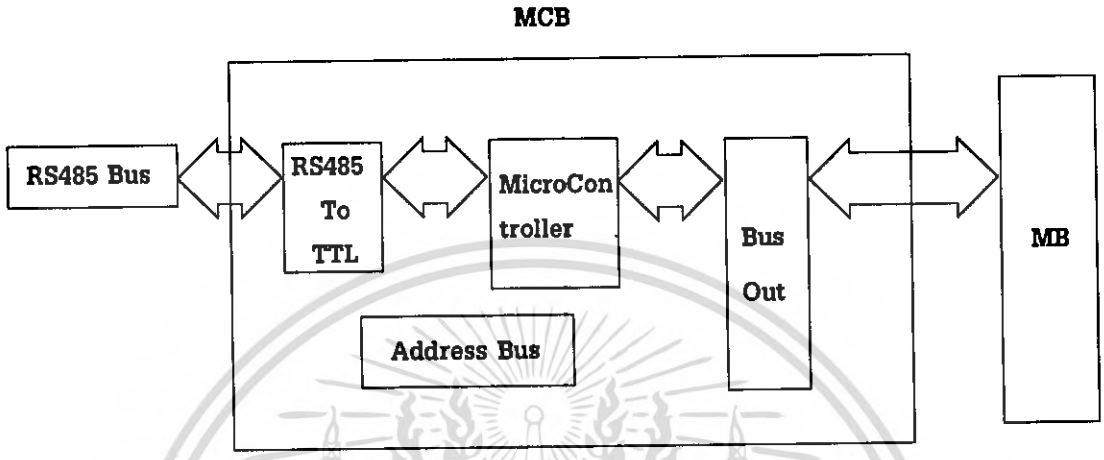
2.3.6.3 โครงสร้างของตัวควบคุม

ตัวควบคุมมีหน้าที่แปลงสัญญาณข้อมูลจาก PC ไปยังส่วนแสดงผลและรับข้อมูลจากส่วนรับข้อมูลส่งไปยัง PC ตามมาตรฐาน RS-485 แสดงโครงสร้างตัวควบคุมในรูปที่ 2.18

2.3.6.4 ส่วนประกอบของตัวควบคุม

1. IC DS75176B ทำหน้าที่แปลงสัญญาณมาตรฐาน RS-485 กับ RS-232C ระดับมาตรฐาน TTL ที่ 89C51 ทำงานได้
2. IC 74LS244 ทำหน้าที่ช่วยเพิ่มกระแสของพอร์ต 89C51 ให้ LED
3. IC 4068 ทำหน้าที่ AND สัญญาณ Input เพื่อเป็นสัญญาณ Interrupt ให้ 89C51
4. LED และ Switch เป็น Input และ Output

5. IC 89C51 ทำหน้าที่ประมวลผลและติดต่อข้อมูล

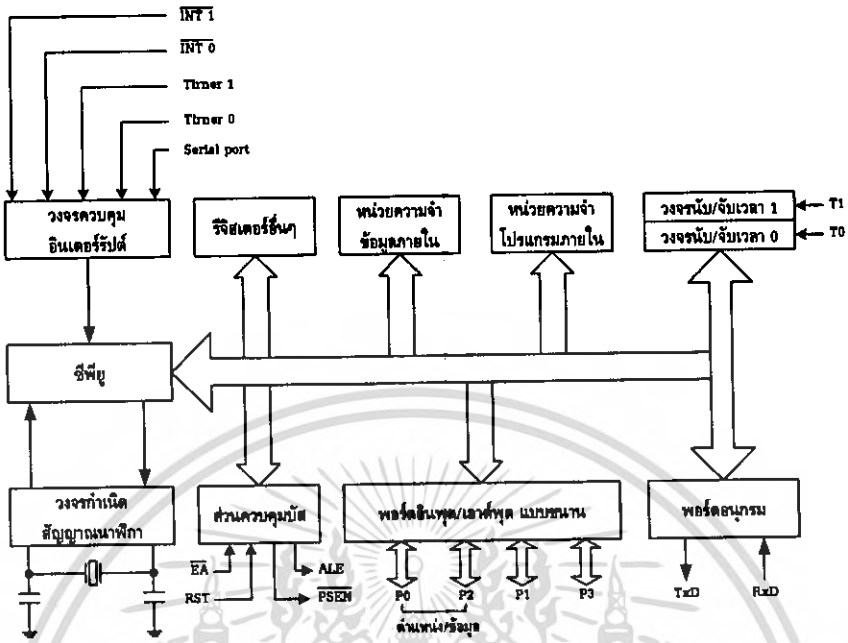


รูปที่ 2.18 โครงสร้างตัวควบคุม

2.4 ไมโครคอนโทรลเลอร์

2.4.1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้ถูกคิดค้น พัฒนา และผลิตโดยบริษัทอินเทล (Intel) เพื่อใช้ในงานควบคุมต่างๆ ไม่ว่าจะเป็นงานควบคุมขนาดเล็กจนถึงงานควบคุมขนาดใหญ่ที่มีความซับซ้อนพอสมควร จากข้อดีของไมโครคอนโทรลเลอร์ที่มีการนำวงจรพื้นฐานต่างๆ มารวมไว้ภายในชิปตัวเดียวกัน ทำให้วงจรควบคุมที่สร้างขึ้นมีขนาดเล็ก มีความสะดวก และคล่องตัวสูงจึงเป็นที่นิยมและแพร่หลายอย่างมาก ทำให้ในปัจจุบันมีไมโครคอนโทรลเลอร์ที่มีมาตรฐานเดียวกันไมโครคอนโทรลเลอร์ขนาด 8 บิต มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูลหน่วยความจำสำหรับโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรมมีสถาปัตยกรรมพื้นฐานที่เหมือนกันสามารถใช้งานแทนกันได้จะต่างกันเพียงขนาดของหน่วยความจำภายใน และหน่วยทำงานภายในเท่านั้น



รูปที่ 2.19 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51

ในรูปที่ 2.19 แสดงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งประกอบด้วย 3 ส่วนหลักๆ ดังนี้

2.4.1.1 หน่วยประมวลผลกลาง

ประกอบไปด้วยส่วนสำคัญ 2 ส่วน คือ ส่วนประมวลผลทางคณิตศาสตร์และลอจิก (Arithmetic Logic Unit : ALU) และส่วนควบคุม (Control Unit : CU) ในส่วนประมวลผลทางคณิตศาสตร์และลอจิกจะทำหน้าที่ประมวลผลข้อมูล เช่น การบวก, ลบ, คูณ หรือการหารข้อมูลแล้วนำผลลัพธ์ไปเก็บไว้ในหน่วยความจำที่ต้องการ ในส่วนควบคุมจะทำหน้าที่สร้างสัญญาณควบคุมในการติดต่อกับส่วนอื่นๆ สัญญาณที่สร้างจากวงจรควบคุมได้แก่ สัญญาณสำหรับการติดต่อกับหน่วยความจำสัญญาณติดต่อกับอุปกรณ์รับข้อมูลเข้าหรือส่งข้อมูลออกรวมทั้งส่วนควบคุมการขัดจังหวะและส่วนควบคุมบัสด้วย ซึ่งซีพียูจะทำการสร้างสัญญาณควบคุมโดยการถอดรหัสคำสั่งที่ได้กำหนดไว้ และสัญญาณที่สร้างขึ้นมาจะอ้างอิงกับสัญญาณนาฬิกาที่สร้างจากวงจรกำเนิดสัญญาณนาฬิกาเพื่อให้ทุกๆ ส่วนทำงานประสานกันอย่างถูกต้อง

2.4.1.2 หน่วยความจำ

มีไว้สำหรับจดจำข้อมูลซึ่งในการนำข้อมูลเข้าและออกจากหน่วยความจำเราจำเป็นต้องรู้ตำแหน่งของหน่วยความจำ ในการนำข้อมูลเข้าไปเก็บในหน่วยความจำเรียกว่า "การเขียนข้อมูล" และการนำข้อมูลออกจากหน่วยความจำเรียกว่า "การอ่านข้อมูล" ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 ข้อมูลในแต่ละตำแหน่งจะมี

ขนาด 8 บิต ดังนั้นในแต่ละตำแหน่งของหน่วยความจำจะสามารถเก็บข้อมูลซึ่งมีค่าระหว่าง 00000000_2 ถึง 11111111_2 หรือ 00_{16} ถึง $0FF_{16}$ ในการติดต่อกับหน่วยความจำจะต้องมีสัญญาณ 3 กลุ่ม คือ

1. ตำแหน่งที่ต้องการติดต่อกับหน่วยความจำซึ่ง MCS-51 สามารถติดต่อกับหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้สูงสุดชนิดละ 65,536 ตำแหน่ง (64 กิโลไบต์) ดังนั้นการอ้างตำแหน่งของหน่วยความจำจะต้องใช้สายสัญญาณกำหนดตำแหน่ง 16 เส้น (2^{16} เท่ากับ 65,536)

2. ข้อมูลที่อ่านหรือเขียนกับหน่วยความจำในตำแหน่งที่เราต้องการ

3. สัญญาณควบคุมที่จะส่งไปยังหน่วยความจำเพื่อบอกกับหน่วยความจำว่าต้องการเขียนหรือข้อมูลซึ่งวงจรถอดรหัสคำสั่งจะทำการสร้างสัญญาณควบคุมจากคำสั่งที่อ่านเข้ามาจากหน่วยความจำโปรแกรม

2.4.1.3 อุปกรณ์อินพุต/เอาต์พุต

เป็นส่วนที่ใช้นำข้อมูลเข้าหรือส่งข้อมูลออกจาก MCS-51 ทำให้สามารถติดต่อกับอุปกรณ์ภายนอกได้ อุปกรณ์อินพุต/เอาต์พุต ได้แก่

1. พอร์ตอินพุต/เอาต์พุตแบบขนานมีทั้งหมด 4 พอร์ตใช้สำหรับรับส่งข้อมูลซึ่งเป็นสัญญาณดิจิตอลเข้าหรือออกจาก MCS-51 โดยแต่ละพอร์ตจะรับส่งข้อมูลได้ 8 บิต มีพอร์ต P0, P1, P2 และ P3 บางพอร์ตจะใช้งานมากกว่า 1 หน้าที่

2. วงจรนับ/จับเวลา ทำงานได้ 2 หน้าที่ คือ เป็นวงจรนับหรือจับเวลา เมื่อเป็นวงจรนับจะทำการนับจำนวนรอบของสัญญาณนาฬิกาภายใน MCS-51 หรือจำนวนรอบของสัญญาณที่ต่ออยู่ภายนอกตัว MCS-51 ก็ได้ สามารถตั้งค่าเริ่มต้นของการนับและอ่านค่าการนับได้โดยซีพียูเมื่อเป็นวงจรจับเวลาจะใช้หลักการเดียวกับวงจรนับเพียงแต่จะกำหนดค่าสูงสุดของการนับไว้ซึ่งค่าสูงสุดของการนับจะคำนวณมาจากค่าเวลาที่ต้องการจับเวลานั่นเอง

3. พอร์ตอนุกรม ซีพียูจะอ่านและเขียนข้อมูลกับพอร์ตอนุกรมโดยเป็นแบบ 8 บิต แต่ข้อมูลถูกส่งออกจาก MCS-51 เรียงไปทีละบิตออกจากขา TxD และในรับข้อมูลก็จะทำการรับเข้ามาทีละบิตทางขา RxD แล้วจัดเรียงใหม่เป็น 8 บิต เพื่อให้ซีพียูอ่านไปใช้งานต่อไป

2.4.2 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

1. เป็นไมโครคอนโทรลเลอร์ที่ใช้หน่วยประมวลผลกลางขนาด 8 บิต
2. หน่วยความจำโปรแกรมภายในมีหลายขนาดขึ้นกับเบอร์ไอซี โดยมีทั้งแบบรอม อีพรอม และแบบแฟลช
3. หน่วยความจำข้อมูลภายในเป็นแบบแรม ในบางเบอร์มีหน่วยความจำอีอีพรอมเพิ่มเติม
4. อ้างตำแหน่งของหน่วยความจำโปรแกรมได้ถึง 64 กิโลไบต์
5. อ้างตำแหน่งของหน่วยความจำข้อมูลได้ถึง 64 กิโลไบต์
6. หน่วยความจำโปรแกรม และหน่วยความจำข้อมูล ทำงานแยกจากกัน

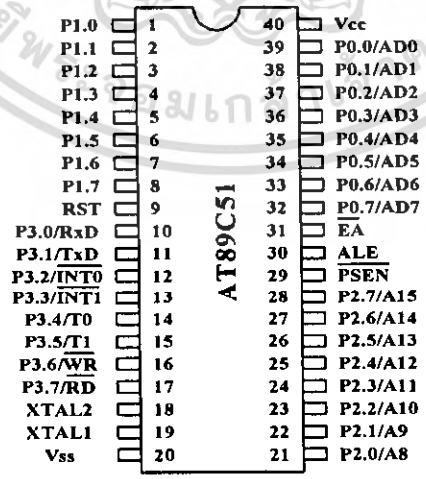
7. มีพอร์ตรับ หรือส่งข้อมูลได้ 2 ทิศทาง จำนวน 4 พอร์ต พอร์ตละ 8 บิตหรือใช้งานเป็นพอร์ตขนาด 1 บิต รวมทั้งหมด 32 บิต ทำงานแยกกันอย่างอิสระ
8. มีวงจรนับ/จับเวลา ขนาด 16 บิต 2 ชุด ทำงานได้ 4 รูปแบบ
9. มีพอร์ตการสื่อสารอนุกรมรับส่งข้อมูลได้ในเวลาเดียวกันสามารถเลือกรูปแบบการส่งได้
10. รับสัญญาณอินเตอร์รัปต์ได้ 6 แหล่ง กระโดดไปทำงานตอบสนองได้ 5 ตำแหน่ง
11. มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายใน
12. ประมวลผลข้อมูลได้ทั้งแบบ 1 บิตและ 8 บิต

ในปัจจุบันไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้มีผู้ผลิตออกมาจำหน่ายมากมาย ในการใช้งานสามารถเลือกใช้ได้ตามความต้องการและความเหมาะสมซึ่งมีส่วนที่แตกต่างกันบางส่วน คือ ส่วนหน่วยความจำข้อมูลภายใน หน่วยความจำโปรแกรมภายใน จำนวนของวงจรนับ/จับเวลา เป็นต้น

2.4.3 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ในรูปที่ 2.20 แสดงลักษณะภายนอกของ MCS-51 แบบ Pin มี 40 ขา หรือเรียกอีกชื่อหนึ่งว่า แบบดีนตะขาบหรือแบบ Dual Inline Package (DIP) โดยแต่ละขามีหน้าที่การทำงานดังนี้

1. Vcc (ขา 40) ต่อไฟเลี้ยง +5 โวลต์
2. Vss (ขา 20) ต่อลงกราวด์
3. Port 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ P0.0-P0.7 ใช้งานเป็นอินพุต/เอาต์พุตพอร์ตทั่วไปใช้เป็นเก็บค่าตำแหน่งหน่วยความจำไบต์ต่ำ (A0-A7) และรับส่งข้อมูล (D0-D7) จากหน่วยความจำภายนอก Port 1 (ขา 1-8) มีทั้งหมด 8 บิต คือ P1.0-P1.7 ใช้งานเป็นอินพุตและ เอาต์พุตของพอร์ตทั่วไป



รูปที่ 2.20 ลักษณะขาภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบ Pin

4. Port 2 (ขา 21-28) มีทั้งหมด 8 บิต คือ P2.0-P2.7 ใช้งานเป็นอินพุต/เอาต์พุตพอร์ตและใช้เป็นที่เก็บค่าตำแหน่งหน่วยความจำไบต์สูง (A8-A15) เพื่อใช้ติดต่อกับหน่วยความจำภายนอก

5. Port 3 (ขา 10-17) มีทั้งหมด 8 บิต คือ P3.0-P3.7 ใช้งานเป็นอินพุต/เอาต์พุตพอร์ตทั่วไป และใช้งานในหน้าที่พิเศษดังนี้

P3.0/RxD (Serial Input Port) ใช้รับข้อมูลแบบอนุกรม

P3.1/TxD (Serial Output Port) ใช้ส่งข้อมูลแบบอนุกรม

P3.2/INT0 (External Interrupt) ใช้เป็นอินพุตเพื่อรับสัญญาณขัดจังหวะจากภายนอก

P3.3/INT1 (External Interrupt) ใช้เป็นอินพุตเพื่อรับสัญญาณขัดจังหวะจากภายนอก

P3.4/T0 (Timer/Counter 0 External Input) ใช้เป็นอินพุตให้วงจรนับ/จับเวลาชุดที่ 0

P3.5/T1 (Timer/Counter 1 External Input) ใช้เป็นอินพุตให้วงจรนับ/จับเวลาชุดที่ 1

P3.6/WR (External Data Memory Write Strobe) ขาควบคุมการเขียนข้อมูลไปยังหน่วยความจำภายนอก

P3.7/RD (External Data Memory Read Strobe) ขาควบคุมการอ่านข้อมูลจากหน่วยความจำภายนอก

RST (ขา 9) Reset ใช้สำหรับรีเซตวงจรทุกอย่างภายในชิปเพื่อเริ่มต้นการทำงานใหม่ในการรีเซตต้องป้อนลอจิก "1" นานอย่างน้อย 2 รอบการทำงานของคำสั่ง

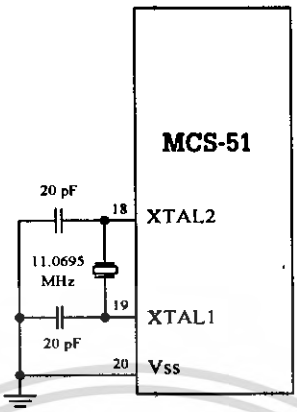
ALE (ขา 30) Address Latch Enable เป็นขาส่งสัญญาณออกไปภายนอก เพื่อควบคุมการคงสภาวะเดิมของค่าตำแหน่งหน่วยความจำไบต์ต่ำจากพอร์ต 0

$\overline{\text{PSEN}}$ (ขา 29) Program Strobe Enable เป็นขาส่งสัญญาณเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอกเมื่อขานี้ Active มีลอจิกเป็น "0" จะอ่านโปรแกรมจากหน่วยความจำโปรแกรมภายนอก และถ้าเป็นการอ่านหน่วยความจำโปรแกรมภายในขานี้จะไม่ Active

$\overline{\text{EA}}$ (ขา 31) External Access เป็นขาที่ใช้สำหรับเลือกว่าให้ทำงานจากหน่วยความจำโปรแกรมภายในหรือหน่วยความจำโปรแกรมภายนอกชิปเมื่อขานี้ Active มีลอจิกเป็น "0" จะเป็นการทำงานตามคำสั่งในหน่วยความจำโปรแกรมภายนอก

XTAL1 (ขา 19) ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรกำเนิดสัญญาณนาฬิกา

XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอกโดยเป็นเอาต์พุตออกจากวงจรกำเนิดสัญญาณนาฬิกา การต่อคริสตอลกับไมครคอนโทรเลอร์แสดงไว้ในรูปที่ 2.21



รูปที่ 2.21 การต่อคริสตอลภายนอกเข้ากับ MCS-51

2.4.4 การจัดหน่วยความจำของ MCS-51

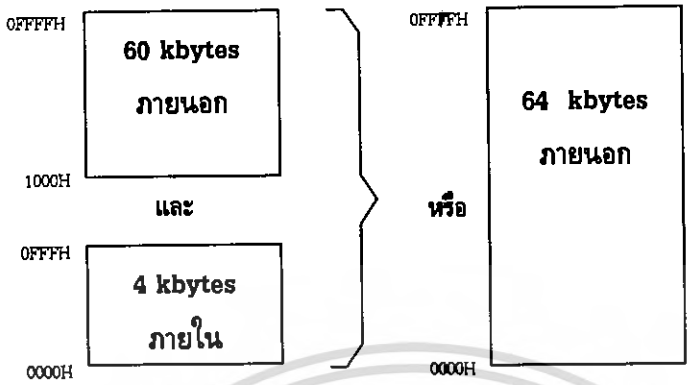
หน่วยความจำของ MCS-51 แบ่งออกเป็น 2 แบบตามลักษณะการใช้งาน ดังนี้

2.4.4.1 หน่วยความจำโปรแกรม (Program Memory)

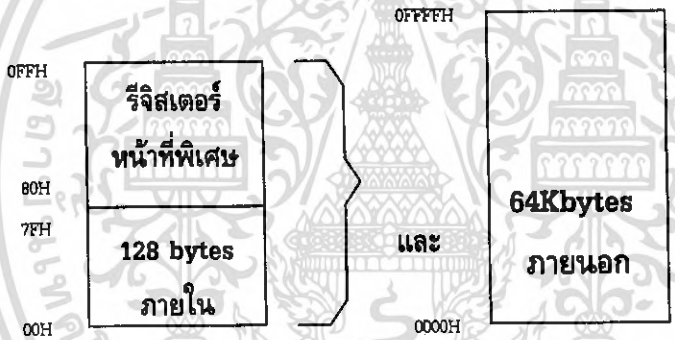
เป็นหน่วยความจำที่ใช้เก็บคำสั่งในรูปของภาษาเครื่องซึ่งต้องการให้ MCS-51 ทำงาน เมื่อ MCS-51 ทำงานก็จะอ่านข้อมูลที่เก็บในหน่วยความจำประเภทนี้ไปทำการถอดรหัสแล้วสร้างสัญญาณควบคุมส่วนอื่นๆ ตามการทำงานของแต่ละคำสั่งนั้น หน่วยความจำแบบนี้เป็นแบบ ROM และผู้ใช้ต้องเขียนข้อมูลในแต่ละตำแหน่งของหน่วยความจำเป็นรหัสภาษาเครื่องของ MCS-51 ตามลำดับการทำงานที่ต้องการส่วนที่เป็นหน่วยความจำโปรแกรมก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง ดังรูปที่ 2.22

2.4.4.2 หน่วยความจำข้อมูล (Data Memory)

หน่วยความจำข้อมูล (RAM) จะทำหน้าที่เก็บรักษาข้อมูลโดยข้อมูลอาจจะเป็นค่าหลังจากไมโครคอนโทรลเลอร์ ทำการการประมวลผล หรือเก็บค่าข้อมูลที่จะให้กับไมโครคอนโทรลเลอร์ประมวลผลในขณะนั้น และจะทำหน้าที่เป็น สแตก (Stack) บางส่วน (ส่วนของสแตกจะอธิบายในลำดับต่อไป) ยกตัวอย่างเช่น ถ้าเป็นเครื่องไมโครเวฟที่ใช้สำหรับอุ่นอาหารก็คือส่วนที่เราป้อนข้อมูลเช่นเวลา หรืออุณหภูมิที่เป็นปัจจุบันหลังจากหน่วยความจำโปรแกรมแสดงรายการหลักที่ LCD นั้นเอง สังเกตว่าหากเราปิดเครื่อง แล้วเปิดเครื่องใหม่อีกครั้งหนึ่ง ค่าข้อมูลที่เป็นเวลา และอุณหภูมิเดิมที่เรากำหนดไว้ในครั้งแรกก็จะหายไป และจะให้เราป้อนค่าข้อมูลใหม่อีกครั้ง ดังนั้นการที่จะรักษาข้อมูลเดิมไว้ได้ จะต้องมีแหล่งจ่ายไฟสำรองไว้สำหรับเพื่อเลี้ยงให้กับตัวไอซีตลอดเวลา หรือที่เรียกว่า Battery Backup สำหรับไอซีไมโครคอนโทรลเลอร์เบอร์ AT89C1051 จะมีหน่วยความจำที่เก็บข้อมูลได้ 64 Bytes ส่วน AT89C2051 และ AT89C4051 จะมีหน่วยความจำที่เก็บข้อมูลได้ 128 Bytes ดังรูปที่ 2.23



รูปที่ 2.22 การจัดพื้นที่หน่วยความจำโปรแกรม



รูปที่ 2.23 การจัดพื้นที่หน่วยความจำข้อมูล

2.4.5 การติดต่อทางพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

พอร์ตสื่อสารอนุกรมของไมโครคอนโทรลเลอร์ จะมีโครงสร้างเป็นแบบฟูลดูเพล็กซึ่งรับและส่งข้อมูลในเวลาเดียวกันได้ โดยจะมีรีจิสเตอร์ SBUF (Serial Data Buffer) เป็นบับเฟอร์สำหรับการรับส่งข้อมูลอนุกรม โดยเริ่มต้นเมื่อมีการเขียนข้อมูลเก็บไว้ในรีจิสเตอร์ SBUF หลังจากนั้นข้อมูลจะถูกจัดการโดยวิธีทางฮาร์ดแวร์ในการเลื่อนบิตเพื่อส่งสัญญาณออกไปภายนอกหลังจากมีการส่งข้อมูลออกไปจนครบแล้ว จึงจะทำการเซตบิตโดยกำหนดค่าของแฟลก TI ในรีจิสเตอร์ SCON ให้เป็นสถานะ "1" เพื่อแจ้งว่ารีจิสเตอร์ SBUF ว่างแล้วและพร้อมที่จะส่งข้อมูลไบต์ต่อไปได้การรับข้อมูลจากพอร์ตอนุกรมจะต้องเริ่มต้น โดยการกำหนดค่าของบิต REN ที่อยู่ในในรีจิสเตอร์ SCON ให้มีค่าเป็นสถานะ "1" หลังจากนั้นเมื่อมีการรับข้อมูลเข้ามาจากภายนอก ก็จะทำการเลื่อนข้อมูลไปโดยอัตโนมัติ และเมื่อบิตสุดท้ายถูกเลื่อนบิตเข้ามาเรียบร้อยแล้ว ข้อมูลจะถูกย้ายมาเก็บไว้ที่รีจิสเตอร์ SBUF และจะทำการเซตที่บิต RI ให้เป็นสถานะ "1" ซึ่งส่งผลให้เกิดการ

อินเตอร์รัพท์โปรแกรมขึ้น โหมดการการติดต่อทางพอร์ตอนุกรมของไมโครคอนโทรลเลอร์การสื่อสารอนุกรมของไมโครคอนโทรลเลอร์จะแบ่งออกได้เป็น 4 โหมดด้วยกัน และในแต่ละโหมดจะสามารถสรุปหน้าที่ได้ดังนี้ โหมด 0 จะเป็นการรับส่งข้อมูลขนาด 8 บิตแบบอนุกรม การส่งข้อมูลจะเลื่อนออกไปทีละบิตโดยจะใช้งานของขา RxD เพียงขาเดียวและจะไม่มีการส่งบิตเริ่มต้น (Start bit) ส่วนขา TxD จะใช้เป็นขาของสัญญาณนาฬิกาในการให้จังหวะการเลื่อน ข้อมูลกับวงจรภายนอก (Shift clock) อัตราการรับส่งข้อมูล (Baud rate) จะเป็น 1/12 เท่าของสัญญาณนาฬิกาการรับและส่งข้อมูลจะเริ่มจากบิตต่ำ (LSB) ก่อนใช้สำหรับเป็นชิพท์รีจิสเตอร์ (Shift Register) จุดประสงค์เพื่อใช้ในการขยายพอร์ตอินพุต หรือพอร์ตเอาต์พุตให้มีจำนวนมาก แต่ในโหมด 0 เรามักจะไม่ค่อยนิยมนำมาใช้งานเพราะไอซีไมโครคอนโทรลเลอร์ในปัจจุบันที่เราใช้อยู่นั้นมีจำนวนพอร์ตที่มากพอและมีไอซีเบอร์อื่นๆ ในตระกูลเดียวกันให้เลือกจำนวนพอร์ตใช้งานมากมายอยู่แล้ว

โหมด 1 จะเป็นการรับและส่งข้อมูลขนาด 10 บิตแบบ UART (Universal Asynchronous Receiver Transmitters) สามารถใช้ในการติดต่อสื่อสารอนุกรมกับมาตรฐานของ RS-232C ของไมโครคอมพิวเตอร์ได้ ซึ่งข้อมูลอนุกรม 10 บิตจะเข้ามาทางขา RxD และส่งข้อมูลออกแบบอนุกรมทางขา TxD โดยจะประกอบด้วย 1 บิตแรกเป็นบิตเริ่มต้น (Start bit ค่า 0) 8 บิตต่อมาจะเป็นบิตของข้อมูล (การรับ/ส่งจะเริ่มจากบิตต่ำก่อน) และบิตหยุดอีก 1 บิต (Stop bit ค่า 1) ส่วนทางด้านรับข้อมูลจะนำค่าบิตหยุด (Stop bit) ที่รับเข้ามาได้นำไปเก็บไว้ในบิต RB8 ที่อยู่ในรีจิสเตอร์ SCON และความเร็วของการส่งข้อมูลในโหมด 1 จะขึ้นอยู่กับบิต SMOD ที่อยู่ในรีจิสเตอร์ PCON และอัตราโอเวอร์โฟลว์ของไทเมอร์ 1 ซึ่งอัตราการรับส่งข้อมูลในโหมดนี้สามารถกำหนดได้ตามต้องการ

โหมด 2 จะเป็นการรับและส่งข้อมูลขนาด 11 บิตแบบ UART (Universal Asynchronous Receiver Transmitters) ข้อมูลแบบอนุกรมจะถูกรับเข้ามาทางขา RxD และส่งข้อมูลออกไปทางขา TxD ซึ่งข้อมูล 11 บิตประกอบด้วย บิตแรกจะเป็นบิตเริ่มต้น (Start bit ค่า 0) 9 บิตต่อมาจะเป็นบิตของข้อมูล และบิตสุดท้ายจะเป็นบิตหยุด 1 บิต (Stop bit ค่า 1) สำหรับข้อมูลในบิตที่ 9 จะกำหนดไว้ใน TB8 ที่อยู่ในรีจิสเตอร์ SCON ซึ่งสามารถกำหนดเป็น 1 หรือ 0 ก็ได้ นิยมนำมาใช้ในการส่งบิตเพื่อตรวจสอบการส่งข้อมูล (Parity bit)

โหมด 3 เป็นการรับส่งข้อมูลแบบ 11 บิตแบบ UART (Universal Asynchronous Receiver Transmitters) เหมือนกับโหมด 2 แต่ในโหมด 3 สามารถกำหนดอัตราความเร็วในการรับและส่งข้อมูลได้ตามต้องการ UART (Universal Asynchronous Receiver Transmitters) เป็นการส่งข้อมูลแบบอนุกรมโดยขึ้นอยู่กับความพร้อมของทางด้านส่งและด้านรับ เป็นการส่งข้อมูลโดยทำการเพิ่มเติมข้อมูลบางอย่างเข้าไป (Start bit, Stop bit, Parity bit) เพื่อให้การรับ และการส่งข้อมูลสามารถจะทำงานให้มีความถูกต้องของข้อมูลมากยิ่งขึ้นการสื่อสารเพื่อเชื่อมต่อไมโครคอนโทรลเลอร์ในการรับและส่งข้อมูลทางอนุกรมมีอยู่ด้วยกัน 2 ระบบคือ 1. Single Processor System คือระบบการสื่อสารโดยใช้ไมโครคอนโทรลเลอร์ 2 ตัวเชื่อมต่อหากัน 2. Multiprocessors System คือระบบการสื่อสารแบบมัลติโปรเซสเซอร์ โดยใช้ไมโครคอนโทรลเลอร์ 1

ตัวเป็นตัวแม่ (Master) และสามารถที่จะเชื่อมต่อกับไมโครคอนโทรลเลอร์ที่เป็นตัวลูก (Slave) ได้อีกเป็นจำนวนหลายๆ ตัว

2.4.6 รีจิสเตอร์ที่ใช้ในการควบคุมและรับส่งข้อมูลของพอร์ตอนุกรม

รีจิสเตอร์ที่ใช้งานในการติต่อสื่อสารทางพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ จะประกอบด้วย รีจิสเตอร์ SCON ที่ทำหน้าที่ควบคุมการทำงาน รีจิสเตอร์ SBUF จะใช้ในการเก็บข้อมูลที่จะรับหรือส่ง และรีจิสเตอร์ PCON ซึ่งจะใช้ในการกำหนดอัตรารับส่งโดยรีจิสเตอร์แต่ละตัวจะมีหน้าที่ และการทำงานในแต่ละบิตดังต่อไปนี้ รีจิสเตอร์ SCON (Serial Port Control Register) เป็นรีจิสเตอร์ขนาด 8 บิต อยู่ในส่วนของรีจิสเตอร์พิเศษ (Special Function Register) ในตำแหน่งแอดเดรสที่ 98H และสามารถเข้าถึงข้อมูลแบบไบต์ และแบบบิตได้โดยจะทำหน้าที่ควบคุมการทำงานของพอร์ตอนุกรมการเลือกโหมดการทำงาน และเก็บข้อมูลในบิตที่ 9 (ซึ่งโดยปกติข้อมูลจะมี 8 บิต อยู่ในรีจิสเตอร์ SBUF) ของการรับข้อมูล (RB8) และส่งข้อมูล (TB8) SM2 เป็นบิตที่ทำหน้าที่ควบคุมการทำงานและเลือกลักษณะการเชื่อมต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์ แบบ Single Processor System หรือ Multi Processors System โดยกำหนดให้ SM2 = 1 เป็นการเลือกแบบ Multi Processors System คือระบบการสื่อสารแบบใช้ซีพียูหลายๆ ตัวร่วมกันทำงาน จะใช้งานในโหมด 2 หรือโหมด 3 SM2 = 0 เป็นการเลือกแบบ Single Processor System โดยสามารถใช้ได้กับทุกโหมด (การใช้งานในโหมด 0 ต้องกำหนดให้ SM2 = 0) ในกรณีที่เลือกให้ SM2 = 1 แบบ Multi Processors System ถ้าข้อมูลที่รับเข้ามาบิตที่ 9 (อยู่ในบิต RB8) มีค่าเป็น "1" ทำให้แฟลกอินเตอร์รัพต์ทางด้านรับจะถูกเซตให้เป็น 1 (RI = 1) แต่ถ้าข้อมูลในบิตที่ 9 รับเข้ามามีค่าเป็น "0" จะทำให้แฟลกอินเตอร์รัพต์ทางด้านรับเป็น 0 (RI=0) การทำงานในโหมด 1 ถ้าให้ SM2 = 1 แฟลกอินเตอร์รัพต์ทางด้านรับ (แฟลก RI) จะไม่ถูกเซตหากข้อมูลที่รับเข้ามาไม่มีบิตหยุด (Stop bit)

REN (Enable Serial Reception) เป็นบิตที่ควบคุมการรับข้อมูลของพอร์ตอนุกรมกำหนดสถานะของบิตได้โดยซอฟต์แวร์ 1 = ให้มีการรับข้อมูล 0 = ไม่ให้มีการรับข้อมูล TB8 (Transmit bit D8) เป็นบิตของข้อมูลบิตที่ 9 ในการส่งข้อมูลใช้งานโหมด 2 และโหมด 3 กำหนดสถานะของบิตได้โดยซอฟต์แวร์ RB8 (Receive bit D8) เป็นบิตของข้อมูลบิตที่ 9 ในการรับข้อมูล โดยใช้งานโหมด 2 และโหมด 3 หากใช้งานในโหมด 1 ถ้ากำหนดให้ SM2 = 0 บิตนี้จะเป็นค่าของ Stop Bit ที่รับเข้ามาสำหรับโหมด 0 จะไม่ใช้งานบิตนี้ TI (Transmit Interrupt Flag) เป็นบิตที่ใช้งานในการอินเตอร์รัพท์ด้านส่งข้อมูล และจะถูกเซตทางฮาร์ดแวร์เมื่อมีการส่งข้อมูลเสร็จสิ้นลงในบิตที่ 8 ของโหมด 0 (Shift register) หรือเมื่อเริ่มต้นส่ง Stop bit ในโหมด 1, 2 หรือ 3 และจะต้องเคลียร์บิตนี้ด้วยซอฟต์แวร์ทุกครั้ง เมื่อโปรแกรมตอบสนองการอินเตอร์รัพท์ของการส่งข้อมูลเรียบร้อยแล้ว RI (Receive Interrupt Flag) เป็นบิตที่ใช้งานในการอินเตอร์รัพท์ทางด้านรับข้อมูล จะถูกเซตทางฮาร์ดแวร์ เมื่อมีการรับข้อมูลเสร็จสิ้นลงในบิตที่ 8 ในโหมด 0 (Shift register) และจะต้องเคลียร์บิตนี้ด้วยซอฟต์แวร์ทุกครั้ง เมื่อโปรแกรมตอบสนองการอินเตอร์รัพท์ของการรับข้อมูลเรียบร้อย

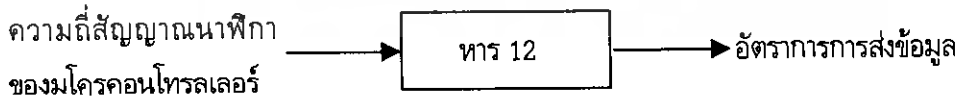
แล้ว หรืออาจกล่าวได้ว่าถ้าบิต RI ถูกเซต เมื่อใดหมายถึงข้อมูลได้เข้ามาเก็บไว้ที่รีจิสเตอร์ SBUF จนครบทั้ง 8 บิตแล้วสามารถที่จะอ่านข้อมูลจากรีจิสเตอร์ SBUF ได้

รีจิสเตอร์ SBUF (serial data buffer register) เป็นรีจิสเตอร์ขนาด 8 บิตหรือ 1 ไบต์มีแอดเดรสอยู่ตำแหน่งที่ 99H และเข้าถึงข้อมูลแบบไบต์ได้อย่างเดียว ซึ่งจะทำหน้าที่รับ และส่งข้อมูลออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ ในการอ่านค่าข้อมูลจากภายนอกที่รับเข้ามาทางพอร์ตอนุกรมจะต้องอ่านค่าจากรีจิสเตอร์ SBUF ซึ่งเป็นบัปเฟอร์ (Buffer) ในการเก็บข้อมูลที่รับเข้ามาได้จากภายนอก และในทำนองเดียวกันขณะที่ต้องการส่งข้อมูลเราก็จะนำเอาค่าข้อมูลที่ส่งออกไปไว้ที่ในรีจิสเตอร์ SBUF ก่อน และหลังจากนั้นจึงจะส่งออกไปโดยจะใช้คำสั่งการโอนย้ายข้อมูลแบบไบต์เช่น MOV SBUF,#20H หรือ MOV SBUF,@R1 ก็ได้การรับข้อมูลในโหมด 0 จะเริ่มต้นรับ เมื่อค่าของบิต RI = 0 และ REN = 1 ส่วนในโหมดอื่นๆ การรับข้อมูลจะเริ่มต้นเมื่อกำหนดบิต REN = 1 และมี Start bit เข้ามาที่ขา RxD รีจิสเตอร์ PCON (Power Control) เป็นรีจิสเตอร์ขนาด 1 ไบต์มีแอดเดรสอยู่ตำแหน่งที่ 87H เข้าถึงข้อมูลได้แบบไบต์อย่างเดียวเท่านั้น

PCON.7 SMOD ในกรณีที่ใช้ไทม์เมอร์ 1 เป็นตัวกำหนดอัตรารับส่ง (Baud rate) และหากกำหนดให้บิตนี้มีค่าเป็น "0" ในการใช้งานกับพอร์ตสื่อสารอนุกรมโหมด 1, 2 และโหมด 3 ค่าอัตรารับส่ง (Baud rate) จะเพิ่มขึ้นเป็นสองเท่ารีจิสเตอร์ PCON ไม่สามารถอ้างตำแหน่งแบบบิตได้แต่จะใช้คำสั่งทางลอจิกของการ OR เช่น ORL PCON,#80H จะเป็นการเซตบิตที่ 7 ของรีจิสเตอร์ PCON และการกำหนดให้บิตมีสถานะเป็น "0" หรือเคลียร์บิตจะใช้การ AND เช่น ANL PCON,#01111111B จะเป็นการเคลียร์บิตที่ 7 ของรีจิสเตอร์ PCON

2.4.7 การกำหนดอัตรารับและส่งข้อมูล (Baud Rate Generator)

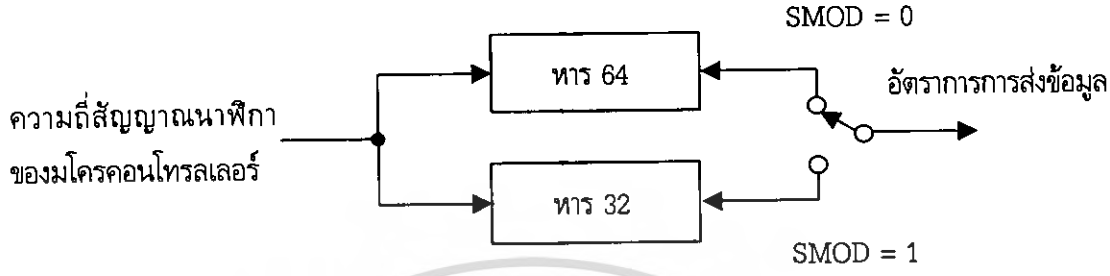
โหมด 0 อัตรารับส่ง (Baud rate) ในโหมด 0 ไม่สามารถกำหนดอัตรารับส่งเองได้ แต่จะมีค่าเท่ากับความถี่สัญญาณนาฬิกาไมโครคอนโทรลเลอร์หารด้วย 12 หรืออาจกล่าวได้ว่าขึ้นกับค่าความถี่ของคริสตอลที่นำมาต่อใช้งานแล้วหารด้วย 12 ดังแสดงในรูปที่ 2.24 นั่นเอง



รูปที่ 2.24 อัตรารับส่งโหมด 0

โหมด 2 อัตรารับส่ง (Baud rate) ในโหมด 2 เลือกได้ 2 อัตราความเร็วในการรับส่งข้อมูลโดยหาได้จากความถี่ของสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์หารด้วย 32 หรือหารด้วย 64 เรียกว่า SMOD 0 และ SMOD 1 ซึ่งจะขึ้นอยู่กับการกำหนดค่าสถานะของบิต SMOD ที่อยู่ในรีจิสเตอร์ PCON เป็นตัวเลือกถ้า

บิต SMOD = 0 อัตารับส่งโหมด 2 = 1/64 เท่าของความถี่สัญญาณนาฬิกาถ้าบิต SMOD = 1 อัตารับส่งโหมด 2 = 1/32 เท่าของความถี่สัญญาณนาฬิกา การเลือกโหมดการถ่ายโอนข้อมูลแสดงในรูปที่ 2.24



รูปที่ 2.25 การเลือกโหมดการถ่ายโอนข้อมูล

หลังจากที่เราทำการรีเซตระบบของไมโครคอนโทรลเลอร์ค่าข้อมูลในบิต SMOD จะเป็นสถานะ "0" เสมอ ดังนั้นเราสามารถที่จะเขียนเป็นสูตรสำหรับการคำนวณหาค่าอัตารับส่ง (Baud rate) ในกรณีที่เราใช้คริสตอลค่า 11.0592 MHz จะได้ค่าอัตารับส่ง (Baud rate) สูงสุด = 345.6 kbps ในกรณีที่เราใช้คริสตอลค่า 12 MHz จะได้ค่าอัตารับส่ง (Baud rate) สูงสุด = 375 kbps โหมด 1 และโหมด 3 จะมีอัตราการรับส่ง (Baud rate) ข้อมูลโดยถูกกำหนดได้ตามต้องการ โดยใช้อัตราการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1 หรือ ไทม์เมอร์ 2 เป็นตัวกำหนด การใช้ไทม์เมอร์ 1 กำหนดอัตารับส่ง (Baud rate) เมื่อใช้ไทม์เมอร์ 1 เป็นตัวสร้างอัตราการรับส่งข้อมูลของพอร์ตอนุกรมในโหมด 1 หรือโหมด 3 สังเกตได้ว่าเมื่อเกิดโอเวอร์โฟลว์ในไทม์เมอร์ตัวใดจะทำให้เกิดสัญญาณอินเตอร์รัพท์เพื่อบอกให้ซีพียูรับทราบดังนั้นในขณะใช้ไทม์เมอร์ 1 เพื่อสร้างอัตารับและส่งข้อมูลจะต้องไม่ให้มีการร้องขออินเตอร์รัพต์ที่เกิดจากไทม์เมอร์ 1 ในกรณีใดๆ ในระหว่างนั้นอีก (โดยการควบคุมที่รีจิสเตอร์ IE) อัตราการรับและส่งข้อมูลจะมาจากอัตราการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1 และค่าของบิต SMOD ที่อยู่ในรีจิสเตอร์ PCON สามารถที่จะเขียนเป็นสูตรสำหรับการคำนวณหาค่าอัตารับส่ง (Baud rate) อัตารับส่งในโหมด 1 และ 3 = x (อัตราการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1) เราสามารถที่จะให้ไทม์เมอร์ 1 มีอัตราการรับส่งต่ำๆ ได้โดยใช้ไทม์เมอร์ 1 ในโหมด 1 ทำงานลักษณะของตัวจับเวลาแบบ 16 บิต (มีค่าของการควบคุมที่รีจิสเตอร์ TMOD = 0001XXXX) และให้มีการอินเตอร์รัพต์จากไทม์เมอร์ 1 โดยให้โปรแกรมตอบสนองการอินเตอร์รัพต์ของไทม์เมอร์ 1 แล้วจะทำการกำหนดค่าเริ่มต้นใหม่ (Reload) ให้กับตัวจับเวลา ซึ่งเป็นการทำงานแบบ 16 บิตทางซอร์ฟแวร์ (Software reload) เนื่องจากการทำงานในโหมด 1 ของไทม์เมอร์ 1 ไม่สามารถโหลดค่าใหม่เองด้วยฮาร์ดแวร์ได้ ตารางที่ 2.2 จะแสดงตารางอัตารับส่งข้อมูลที่กำหนดจากการใช้ไทม์เมอร์ 1 เมื่อใช้บอดเรทค่ามาตรฐานต่างๆ และการกำหนดค่าของรีจิสเตอร์ TMOD ที่บิต C/T และรีจิสเตอร์ PCON ที่บิต SMOD เพื่อใช้งาน

ตารางที่ 2.2 ตัวอย่างอัตรารับส่งข้อมูลที่กำหนดจากการใช้ไทม์เมอร์ 1 เมื่อใช้บอดเรทค่ามาตรฐานต่างๆ

Baudrate	Fosc (MHZ)	บิต SMOD	Timer 1		
			บิต C/T	MODE	RELOAD VALUE
62.5 k	12.0	1	0	2	FFH
19.25 k	11.059	1	0	2	FDH
9600	11.059	0	0	2	FDH
4800	11.059	0	0	2	FAH
2400	11.059	0	0	2	F4H
1200	11.059	0	0	2	E8H
137.5	11.059	0	0	2	1DH
110	6	0	0	2	72H
110	12	0	0	1	FEEBH

จะสังเกตได้ว่าการใช้งานจากความถี่คริสตอลเรามักจะนิยมจะเลือกใช้ค่าความถี่ 11.0592 MHz เนื่องจากการกำหนดค่าอัตราการรับส่งข้อมูลจะสามารถกำหนดค่าบอดเรทในโหมด 1 และโหมด 3 ได้ และเป็นค่าที่มาตรฐานเช่น 1200, 2400, 4800, 9600, 19200 ดังนั้นจึงเป็นเหตุผลที่เราเลือกใช้คริสตอลค่าความถี่ 11.0592 MHz มากกว่าค่า 12 MHz ในการกำหนดอัตรารับส่งข้อมูลยังสามารถที่จะใช้ไทม์เมอร์ 2 กำหนดค่าได้ แต่จะยังไม่ขอกล่าวในที่นี้เพราะไอซีที่เรานำมาทำโครงงานนั้นไม่มีไทเมอร์ 2 ใช้งาน

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

3.1 กล่าวนำ

ไมโครคอนโทรลเลอร์ที่นำมาใช้ในโครงงานนี้อ้างอิงถึงไมโครคอนโทรลเลอร์ MCS-51 ซึ่งมีหน่วยความจำเป็นแบบแฟลช (Flash Memory) ของ ATMEL CORPORATION หมายเลข AT89C52 และเหตุผลที่ใช้ไมโครคอนโทรลเลอร์ชนิดนี้มีหลายประการดังนี้

3.1.1 หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทรลเลอร์

ที่เป็นแบบแฟลชจึงทำให้สามารถลบ และเขียนใหม่ได้เป็นพันครั้งจึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดียว โดยไม่ต้องใช้หน่วยความจำภายนอกส่งผลทำให้สามรถใช้งานพอร์ตอินพุตและเอาต์พุตของ ไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

3.1.2 ต้นทุน

ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูเลเตอร์และเครื่องโปรแกรมอีพรอม

3.1.3 บริษัทผู้ผลิต

บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเลขหมาย โดยมีขีดความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

3.1.4 การใช้หน่วยความจำของไมโครคอนโทรลเลอร์

ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

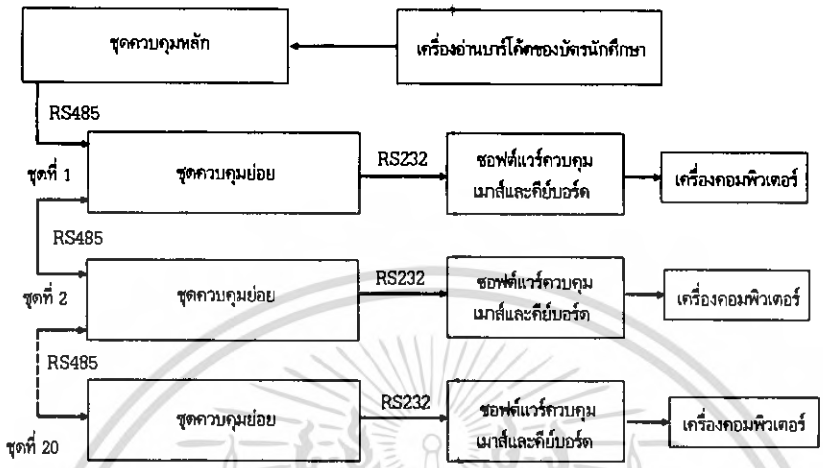
3.1.5 หมายเลขของไมโครคอนโทรลเลอร์

บางหมายเลขของไมโครคอนโทรลเลอร์ซึ่งผลิตโดยบริษัท ATMEL นั้นสามารถโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้ โดยไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาโปรแกรมใหม่ หรือที่เรียกว่า การโปรแกรมในวงจร หรือในระบบ โดยใช้การติดต่อในลักษณะ SPI ทำให้ การพัฒนาหรือการซ่อมบำรุงตลอดจนการปรับปรุง หรือพัฒนาข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก

3.1.6 ชุดคำสั่งและสถาปัตยกรรมพื้นฐาน

ชุดคำสั่งและสถาปัตยกรรมพื้นฐานจะเหมือนกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตอื่นจากเหตุผลข้างต้นคณะผู้จัดทำจึงได้นำไมโครคอนโทรลเลอร์ตระกูล MCS-51 มาใช้ในการออกแบบและสร้างเป็นระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

3.1.7 ผังการทำงานของระบบควบคุมบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม



รูปที่ 3.1 ผังการทำงานของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

ผังการทำงานของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมประกอบไปด้วย 4 ส่วนคือ 1 เครื่องอ่านบาร์โค้ดของบัตรนักศึกษา 2. เครื่องควบคุมหลัก 3.เครื่องควบคุมย่อย 4. ซอฟต์แวร์ควบคุมเมาส์และคีย์บอร์ด การทำงานของเครื่องอ่านบาร์โค้ดจะอ่านรหัสแท่งบนบัตรนักศึกษาแล้วเปลี่ยนเป็นข้อมูลตัวเลขส่งให้เครื่องควบคุมหลัก เครื่องควบคุมหลักเป็นส่วนประมวลผลกลางเป็นส่วนที่ข้อมูลเวลาที่เข้าและออกการใช้งานเครื่องคอมพิวเตอร์ สั่งการเปิดและปิดเครื่องควบคุมย่อยทุกตัว เครื่องควบคุมย่อยเป็นตัวเชื่อมต่อระหว่างเครื่องควบคุมหลักผ่านพอร์ต RS485 และซอร์ฟแวร์บนเครื่องคอมพิวเตอร์ผ่านพอร์ต RS485 เพื่อคอยสั่งการให้ซอร์ฟแวร์ควบคุมเมาส์และคีย์บอร์ดทำการล็อคหรือปลดล็อคเมาส์และคีย์บอร์ด

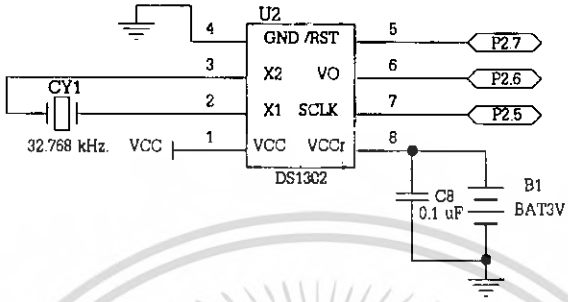
สำหรับการออกแบบ การสร้าง และการทำงานของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมนั้น จะประกอบด้วยการออกแบบส่วนใหญ่ๆ ที่สำคัญคือ การออกแบบวงจรควบคุมหลักและส่วนควบคุมย่อยและการออกแบบทางด้านโปรแกรมที่ใช้ในส่วนควบคุมในแต่ละส่วนซึ่งมีรายละเอียดดังต่อไปนี้

3.2 การออกแบบและการสร้างเครื่องควบคุมหลัก

3.2.1 วงจรฐานเวลาจริง

DS1302 เป็นอุปกรณ์ประเภท I2C ประเภท Real Time Clock (RTC) ซึ่งทำหน้าที่เกี่ยวกับฐานเวลาในลักษณะของ นาฬิกา เวลา และปฏิทิน โดยไอซี DS1302 จะต่อร่วมกับคริสตอลความถี่ 32.768 kHz.

ส่วนไฟเลี้ยงจะอยู่ที่ 5 โวลต์ นอกจากนี้จะมีส่วนสำหรับใส่แบตเตอรี่เพื่อใช้ในการ Back Up ฐานเวลาให้นาฬิกายังคงเดินได้อย่างถูกต้อง เมื่อไม่มีการจ่ายไฟเลี้ยงให้กับวงจรดังแสดงในรูปที่ 3.2



รูปที่ 3.2 วงจรฐานเวลาจริง

3.2.2 วงจรขับ LCD 16x1

สำหรับวงจรขับ LCD นี้ได้เลือกใช้ LCD ขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากว่าราคาถูก หาง่าย และเป็น LCD ที่มีโครงสร้างเป็นมาตรฐานมีขาต่อใช้งานทั้งสิ้น 14 ดังแสดงในรูปที่ 3.3 ขามีการจัดขาดังนี้

Vss (ขา 1) ต่อกราวด์

Vdd (ขา 2) ต่อไฟเลี้ยง +5 โวลต์

Vo (ขา 3) ต่อเป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผล

R/W (ขา 5) เป็นขาที่ใช้เลือกการอ่านหรือการเขียนข้อมูล

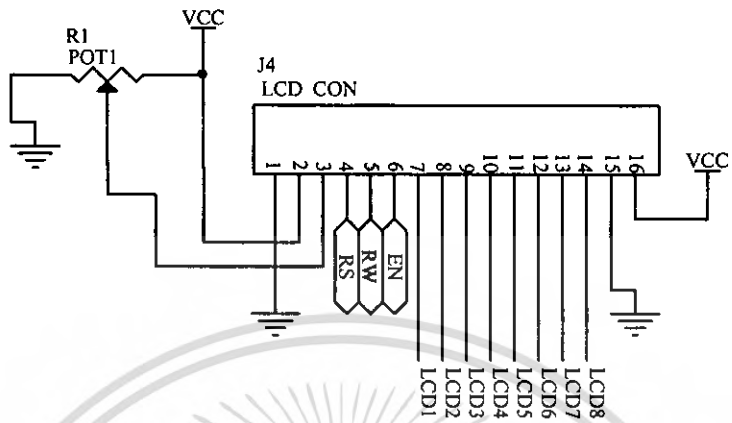
E (ขา 6) เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิ้ลโมดูล LCD ให้ทำงาน

D0-D7 (ขา 7-14) เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก 8 บิตต่อเข้าที่พอร์ต P1.0-P1.7 ของไมโครคอนโทรลเลอร์ U6

อนึ่งขา RS, R/W และ E จะใช้งานร่วมกัน โดยมีความสัมพันธ์แสดงในตารางที่ 3.1

ตารางที่ 3.1 ความสัมพันธ์ในการทำงานของขา RS, R/W และ EN

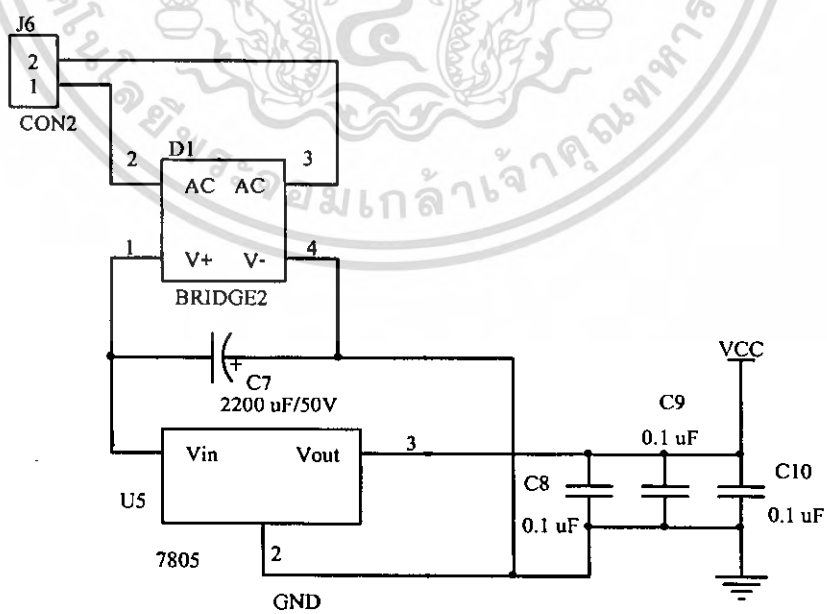
RS	R/W	EN	การทำงาน
0	0		เขียนคำสั่ง
0	1		อ่านสถานะของโมดูล LCD
1	0		เขียนข้อมูล
1	1		อ่านข้อมูล



รูปที่ 3.3 วงจรขับ LCD 16x1

3.2.3 แหล่งจ่ายไฟ 5 โวลต์

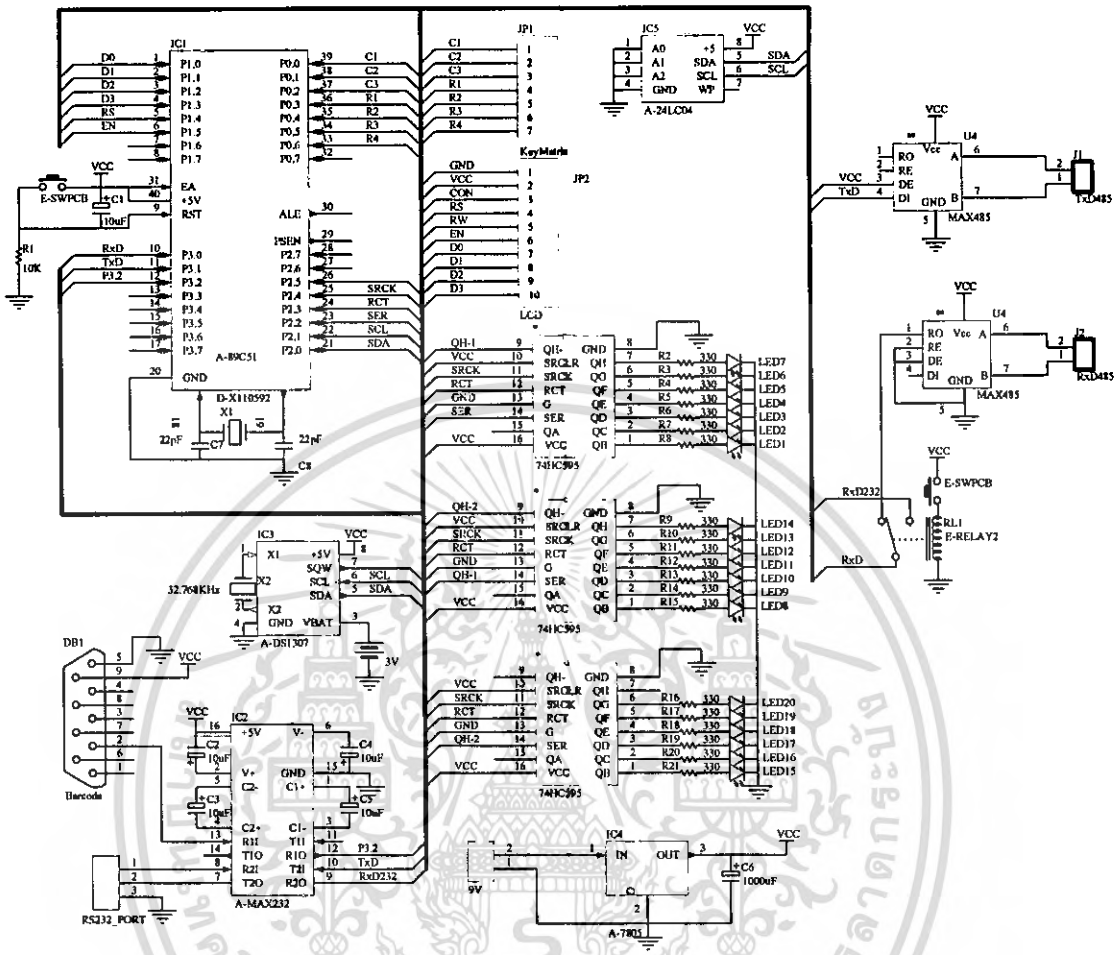
วงจรจ่ายไฟ 5 โวลต์นั้นจะใช้ไอซีเร็กกูเลเตอร์ เบอร์ 7805 ซึ่งให้แรงดันเอาต์พุตเท่ากับ 5 โวลต์ วงจรจ่ายไฟนี้จะใช้เป็นไฟเลี้ยงให้แก่ วงจรฐานเวลา วงจรขับ LCD วงจรขับรีเลย์และวงจรควบคุม โดยไอซีเร็กกูเลเตอร์นี้จะต่อร่วมกับคาปาซิเตอร์เพื่อเพิ่มความเสถียรภาพให้กับแรงดันเอาต์พุตที่ออกมาให้มาความเรียบขึ้นวงจรจ่ายไฟ 5 โวลต์ของเครื่องควบคุมหลักแสดงในรูปที่ 3.4



รูปที่ 3.4 วงจรจ่ายไฟ 5 โวลต์ของเครื่องควบคุมหลัก

3.2.4 วงจรควบคุมหลัก

วงจรควบคุมมีไอซีไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89C52 เป็นหัวใจสำคัญในการสร้างโครงงานนี้โดยไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงานทั้งหมด เมื่อเปิดเครื่องที่จอแอลซีดีจะปรากฏข้อความ "INSERT CARD" และที่หลอดแอลอีดีทางด้านขวามือของตัวเครื่องจำนวน 20 หลอดจะติดหมดทุกหลอด เมื่อมีการรูดบัตรนักศึกษาที่เครื่องอ่านบัตรบาร์โค้ดข้อมูลที่ได้จากถูกส่งยังไอซีแม็ก 232 เพื่อเปลี่ยนระดับสัญญาณจากสัญญาณซีเรียล +12 โวลต์ถึง -12 โวลต์เป็นสัญญาณทีทีแอล 0 ถึง 5 โวลต์เข้าไปที่ไมโครคอนโทรเลอร์ตัวที่หนึ่งจากนั้นไมโครคอนโทรลเลอร์ก็จะทำการเปรียบเทียบข้อมูลที่ได้เปรียบเทียบกับฐานข้อมูลในฐานข้อมูลอีอีพรอมถ้าไม่มีอยู่ในฐานข้อมูลจะรอรับค่าการรูดบัตรนักศึกษาใหม่ หากตรวจสอบดูแล้วพบว่ามีอยู่ในฐานข้อมูลก็จะสั่งการไปที่จอแอลซีดีให้ข้อความจาก "INSERT CARD" ให้เป็น "CHOOSE COMPUTER" จากนั้นจะเก็บข้อมูลรหัสประจำตัวนั้นไว้ที่ตัวไมโครคอนโทรลเลอร์เองแล้วไปรับค่าจากคีย์สวิตช์ได้ ก็จะตรวจสอบการกดที่คีย์สวิตช์แล้วนำค่าที่ได้ไปแสดงผลไปที่จอแอลซีดีเมื่อกดครบแล้วก็จะนำค่าที่ได้มาประมวลผลสัญญาณการเลือกเครื่องแล้วก็จะนำข้อมูลมาตรวจสอบว่าข้อมูลที่ได้มาเกิน 20 ถ้าเกินจะแสดงข้อความว่า "err" ขึ้นที่หน้าจอแอลซีดีและจะไปรับค่าคีย์ใหม่หากว่าไม่เกินก็จะเปลี่ยนข้อความจาก "CHOOSE COMPUTER" ให้เป็น "INSERT CARD" เหมือนค่าเริ่มต้น ข้อมูลการเลือกเครื่องที่ได้จากการเลือกเครื่องจะนำไปทำอีก 3 ขั้นตอนคือ (1) จะส่งข้อมูลเลขเครื่องไปประมวลผลแล้วนำข้อมูลที่ได้ไปที่ไอซีแม็ก 485 เพื่อทำการเข้ารหัสแล้วส่งข้อมูลไปยังเครื่องควบคุมย่อยต่างๆ (2) นำข้อมูลเลขเครื่องที่ได้และข้อมูลเวลาวันเดือนปีจากฐานเวลาจริงมาเก็บไว้ที่ตัวไมโครคอนโทรลเลอร์แล้วทำการบันทึกข้อมูลทั้งหมดลงไอซีอีอีพรอม (3) นำข้อมูลจากการกดสวิตช์ที่มาทำการการประมวลผลด้วยไมโครคอนโทรลเลอร์แล้วทำการส่งไปยังไอซี 74HC595 เพื่อทำการดับหลอดแอลอีดีในตำแหน่งเลขเครื่องนั้นๆ เพื่อบอกสถานะว่าเครื่องที่ถูกเลือกนั้นไม่ว่างแล้ว และเมื่อมีการเลิกใช้งานเครื่องคอมพิวเตอร์แล้วเครื่องควบคุมย่อยก็จะส่งข้อมูลกลับมายังเครื่องควบคุมหลัก เครื่องควบคุมหลักก็จะนำข้อมูลที่ได้มาประมวลผลแล้วทำการบันทึกเวลาออกและเลขเครื่องที่เลิกใช้งานเก็บไว้ที่หน่วยความจำอีอี พรอมและสั่งการไปยังไอซี 74HC595 เพื่อกลับสถานะของหลอดในตำแหน่งเครื่องที่ล็อคเอาต์จากหลอดที่เคยดับจะกลับกลับมาติดอีกครั้งเหมือนตอนเริ่มต้นทำงานในครั้งแรก ในรูปที่ 3.5 แสดงวงจรของเครื่องควบคุมหลัก



รูปที่ 3.5 วงจรเครื่องควบคุมหลัก

3.2.5 การออกแบบทางด้านโปรแกรมของเครื่องควบคุมหลัก

การทำงานบาร์โค้ดจะเป็นตัวทำหน้าที่ถอดรหัสจากบัตรนักศึกษาและแสดงออกมาเป็นเลขแปดหลักตามรหัสของบัตรนักศึกษา เมื่อรูดบัตรเครื่องควบคุมหลักจะนำรหัสที่ได้ไปทำการตรวจสอบกับฐานข้อมูลที่มีซึ่งถูกบันทึกอยู่ในหน่วยความจำสำรองอีอีพรอม และเมื่อตรวจสอบแล้วพบว่าไม่มีอยู่ในฐานข้อมูลเครื่องควบคุมหลักจะกลับไปรอรับค่าจากบาร์โค้ดใหม่และจะแสดงประโยค "INSERT CARD" และถ้าหากมีอยู่ในฐานข้อมูลเครื่องควบคุมจะแสดงประโยค "CHOOSE COMPUTER" ที่จอแอลซีดีเพื่อบอกว่าเสร็จสิ้นการตรวจสอบจากฐานข้อมูลแล้วและให้ทำขั้นตอนต่อไปคือการเลือกเครื่องคอมพิวเตอร์โดยจะมีหลอดแอลอีดีแสดงผลว่าเครื่องใดอยู่ในสถานะว่างหรือไม่ว่างหากเครื่องนั้นว่างจะแสดงผลให้หลอดแอลอีดีติดและหากมีผู้ใช้งานแล้วหลอดแอลอีดีก็จะดับหากได้เครื่องที่ต้องการแล้วให้กดที่คีย์ซึ่งคีย์กดนั้นจะมี 12 คีย์คือคีย์ศูนย์ถึงเก้าคีย์ดอกจันท์และคีย์ชาร์ปโดยคีย์ดอกจันท์จะเป็นการลบเลขเครื่องที่เลือกไว้เพื่อเลือกเครื่องใหม่ส่วยคีย์ชาร์ปเมื่อกดจะเป็นการยืนยันเลขเครื่องที่ต้องการใช้งานเสมือนการกด Enter การกดเลือกจะต้องกดสองครั้ง

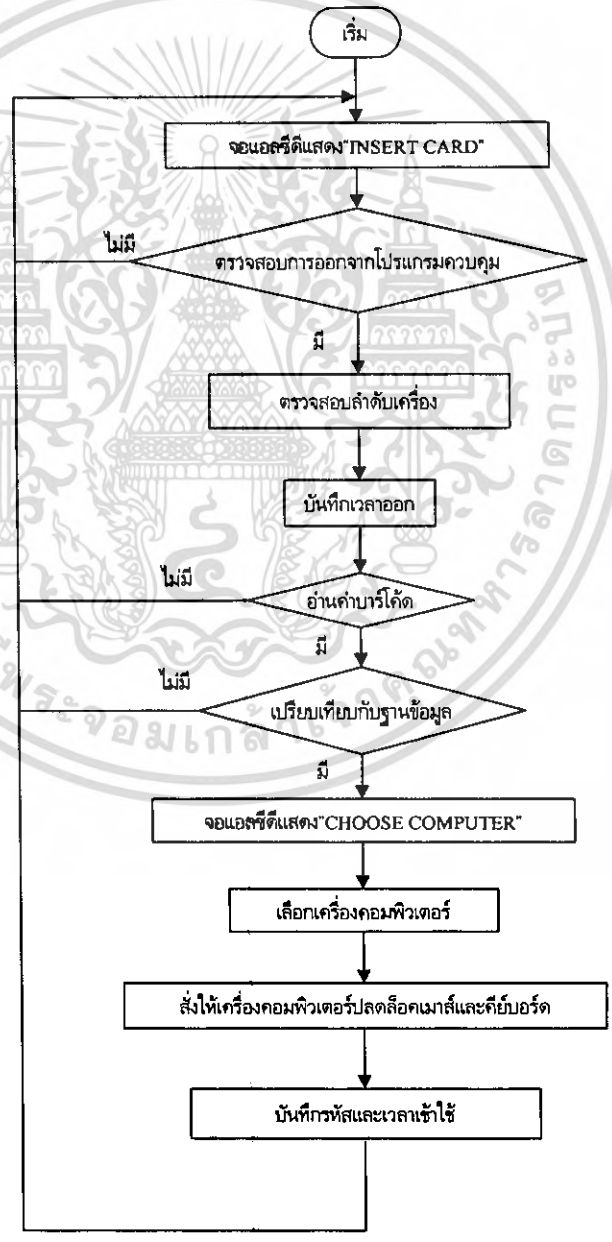
โดยเครื่องที่หนึ่งถึงเครื่องที่หนึ่งถึงเก้าให้กดเลขศูนย์ก่อนโดยเลขที่กดจะแสดงผลที่จอแอลซีดี เครื่องควบคุมหลักจะส่งข้อมูลเป็นตัวเลขและตัวอักษรในลักษณะสตริงโค้ดตามที่กำหนดไว้ออกไปยังเอาต์พุตเช่น เลือกเครื่องที่หนึ่งเราก็ต้องกดเลข 0 และ 1 ที่เครื่องควบคุมหลักจากนั้นเครื่องควบคุมหลักก็จะส่ง :01I$ ตามที่ได้กำหนดไว้ในโปรแกรมออกไปที่พอร์ตเอาต์พุตซึ่งเป็นพอร์ต RS485 และจะกลับมารอรับค่าการตอบกลับจากเครื่องควบคุมย่อยว่าเครื่องในตำแหน่งนั้นๆ ได้รับข้อมูลที่ส่งไปให้แล้วและเมื่อเครื่องควบคุมหลักได้รับข้อมูลการยืนยันการรับข้อมูลที่ส่งมาจากเครื่องควบคุมย่อยแล้วเครื่องควบคุมหลักก็จะทำการบันทึกข้อมูลวันเวลาการใช้เข้างานไว้และกลับไปรอรับค่าที่เครื่องบาร์โค้ดอีกครั้ง เมื่อเลิกใช้งานเครื่องควบคุมย่อยจะส่งข้อมูลการเลิกใช้งานมาที่เครื่องควบคุมหลักจากนั้นเครื่องควบคุมหลักก็จะตรวจสอบว่าเป็นข้อมูลการเลิกใช้งานจากเครื่องใดเมื่อตรวจสอบได้แล้วเครื่องควบคุมหลักจะทำการสั่งให้บันทึกวันเวลาของการออกของผู้เข้าใช้และจะให้เครื่องนั้นกลับไปสู่สภาวะไม่มีผู้ใช้งานอีกครั้งและหลอดแอลอีดีที่แสดงสถานะของเครื่องนั้นก็จะกลับมาติดอีกครั้งข้อมูลที่เครื่องควบคุมหลักส่งออกและข้อมูลยืนยันการรับที่ส่งกลับมาจากเครื่องคุมย่อยแสดงในตารางที่ 3.2 และผังการทำงานของโปรแกรมควบคุมหลักแสดงในรูปที่ 3.6

ตารางที่ 3.2 ข้อมูลที่เครื่องควบคุมหลักส่งออกและข้อมูลยืนยันการรับที่ส่งกลับมาจากเครื่องคุมย่อย

เครื่องที่	ข้อมูลที่เครื่องควบคุมหลักส่งไปที่เครื่องควบคุมย่อย	ข้อมูลที่เครื่องควบคุมย่อยส่งกลับไปที่เครื่องควบคุมหลัก
01	:01I$	:01A$
02	:02I$	:02A$
03	:03I$	:03A$
04	:04I$	:04A$
05	:05I$	:05A$
06	:06I$	:06A$
07	:07I$	:07A$
08	:08I$	:08A$
09	:09I$	:09A$
10	:10I$	:10A$
11	:11I$	:11A$
12	:12I$	:12A$
13	:13I$	:13A$
14	:14I$	:14A$
15	:15I$	:15A$
16	:16I$	:16A$
17	:17I$	:17A$

ตารางที่ 3.2 (ต่อ) ข้อมูลที่เครื่องควบคุมหลักส่งออกและข้อมูลยืนยันการรับที่ส่งกลับมาจากเครื่องคุมย่อย

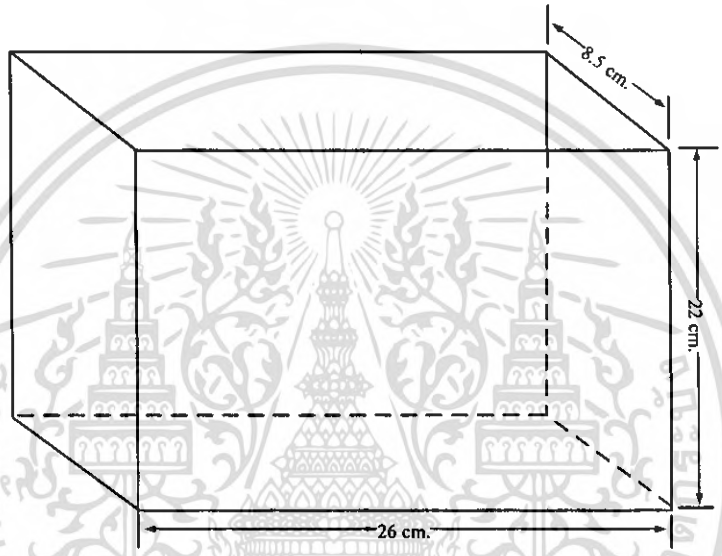
เครื่องที่	ข้อมูลที่เครื่องควบคุมหลักส่งไปที่เครื่องควบคุมย่อย	ข้อมูลที่เครื่องควบคุมย่อยส่งกลับไปที่เครื่องควบคุมหลัก
18	:18I$	:18A$
19	:19I$	:19A$
20	:20I$	:20A$



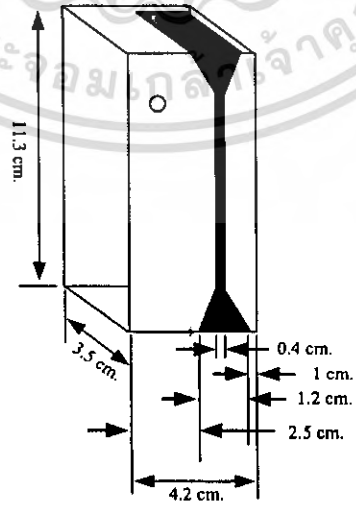
รูปที่ 3.6 ผังการทำงานของโปรแกรมควบคุมหลัก

3.2.6 การออกแบบทางโครงสร้างของเครื่องควบคุมหลัก

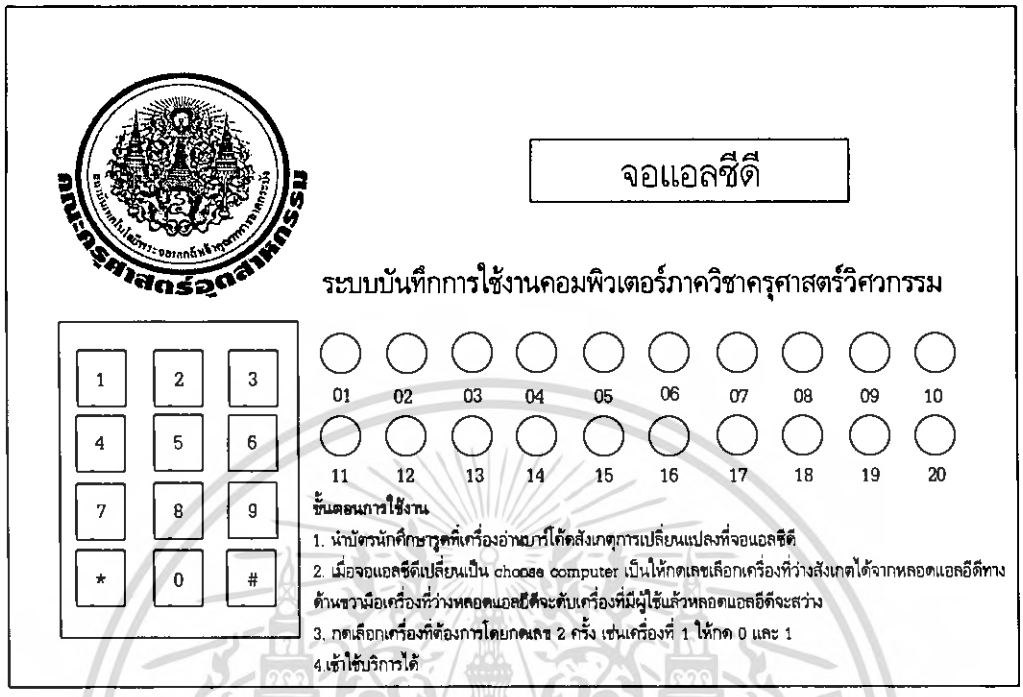
การออกแบบทางโครงสร้างของเครื่องควบคุมหลักจะประกอบไปด้วยกล่องที่บรรจุวงจรและอุปกรณ์ต่างๆ ของเครื่องควบคุมหลักเช่น หม้อแปลง จอแอลซีดี แป้นกด หลอดแอดอีดีแสดงสถานะ และจุดเชื่อมต่อของพอร์ต RS232 และ RS485 โดยขนาดของกล่องจะแสดงในรูปที่ 3.6 และขนาดของเครื่องอ่านบาร์โค้ดจะแสดงในรูปที่ 3.7 ในรูปที่ 3.8 จะเป็นการแสดงรูปหน้าปัดของเครื่องควบคุมหลักและระยะการวางอุปกรณ์ทั้งหมดของเครื่องควบคุมหลักจะแสดงในรูปที่ 3.9



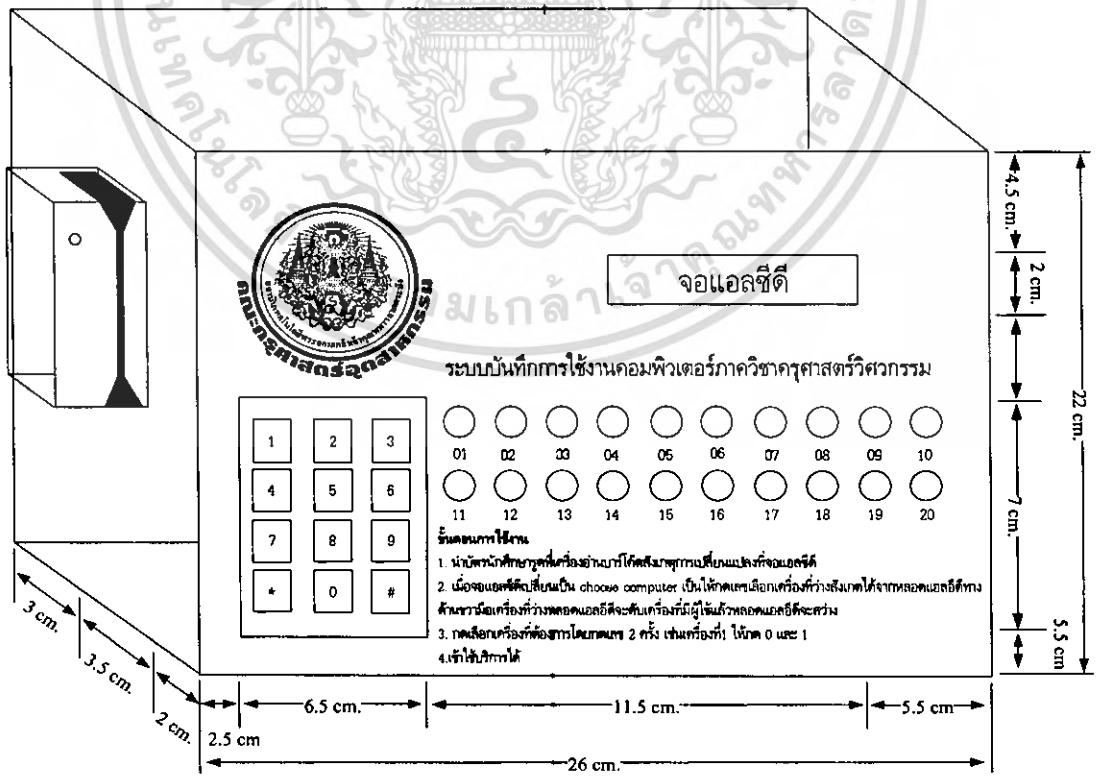
รูปที่ 3.7 กล่องเครื่องควบคุมหลักของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม



รูปที่ 3.8 เครื่องอ่านบาร์โค้ด



รูปที่ 3.9 หน้าปัทม์เครื่องควบคุมหลักของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

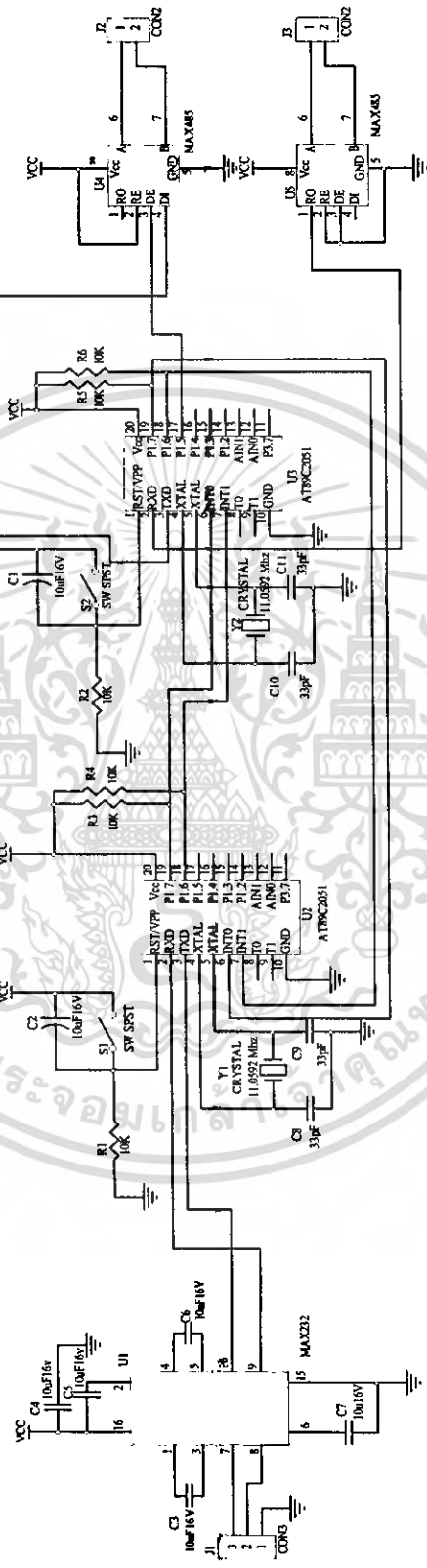


รูปที่ 3.10 เครื่องควบคุมหลักของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

3.3 การออกแบบและการสร้างเครื่องควบคุมย่อย

3.3.1 การควบคุมการทำงานของเครื่องเครื่องควบคุมย่อย

เครื่องควบคุมย่อยจะเป็นการนำไมโครคอนโทรลเลอร์เบอร์ AT89S52 เป็นตัวควบคุมการส่งข้อมูลไปให้ตัวโปรแกรมที่ควบคุมที่ไม่อนุญาตให้ใช้งานเมาส์และคีย์บอร์ด โดยเครื่องควบคุมย่อยเป็นตัวเชื่อมระหว่างเครื่องควบคุมหลักและโปรแกรม โดยเครื่องควบคุมย่อยจะใช้ไมโครคอนโทรลเลอร์จำนวน 2 ตัวดังแสดงในรูปที่ 3.11 สาเหตุที่ต้องใช้สองตัวก็เพราะว่าต้องการใช้พอร์ตอนุกรม 2 ชุดในการเชื่อมต่อระหว่างเครื่องควบคุมหลักกับตัวโปรแกรมที่อยู่บนเครื่องคอมพิวเตอร์จึงกำหนดให้ที่อินพุตของเครื่องควบคุมย่อยจะถูกต่อไว้กลับพอร์ต RS485 เพื่อเชื่อมต่อกับเครื่องควบคุมหลักและที่เอาต์พุตของเครื่องควบคุมย่อยจะต่อกับ RS232 เพื่อนำไปเชื่อมต่อกับเครื่องคอมพิวเตอร์และจะกำหนดให้เครื่องทำงานเมื่ออินพุตมีข้อมูลส่งเข้ามาโดยแต่ละเครื่องจะกำหนดการรับข้อมูลไว้ต่างกันซึ่งการรับข้อมูลนั้นดูได้จากตารางที่ 3.2 การทำงานของวงจรนั้นพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ตัวที่ 1 จะถูกต่อกับไอซีแม็ก 485 จำนวน 2 ตัวเรียกว่าการต่อแบบฟลูเอฟเฟคคือการใช้ไอซีแม็ก 485 ต่อ 2 ตัวทำให้สามารถรับและส่งข้อมูลได้พร้อมกัน การรับข้อมูลของเครื่องควบคุมย่อยนั้นจะรับจาก Tx ของเครื่องควบคุมหลักนำมาเข้าเข้าที่ Rx ของเครื่องควบคุมย่อย เมื่อมีข้อมูลถูกส่งเข้ามาที่อินพุตของเครื่องควบคุมย่อยไมโครคอนโทรลเลอร์ตัวที่ 1 ของเครื่องควบคุมย่อยก็จะนำข้อมูลไปตรวจสอบว่าใช่ข้อมูลที่ต้องการหรอไม่ถ้าไม่ใช่ก็จะไม่ทำงานต่อถ้าใช่ก็จะส่งสัญญาณไปบอกไมโครคอนโทรลเลอร์ตัวที่ 2 ให้ส่งข้อมูลออกไปที่พอร์ตอนุกรม Tx ผ่านไอซีแม็ก 232 เข้ารหัสจากทีทีแอลเป็นซีเรียลแล้วส่งไปที่โปรแกรมคอมพิวเตอร์ผ่านพอร์ต RS232 เช่นเครื่องที่หนึ่งถูกตั้งไว้ว่าเมื่อค่า 01 ส่งมาให้เริ่มทำงานและเมื่อมีข้อมูล 01 เข้ามาเครื่องที่หนึ่งเท่านั้นที่รับค่าเข้ามาตรวจสอบแล้วว่าตรงตามที่กำหนดจากนั้นเครื่องควบคุมย่อยจะส่งข้อมูล :I$ อยู่ในรูปของสตริงโค้ดส่งไปที่ตัวโปรแกรมที่ควบคุมเมาส์และคีย์บอร์ดผ่านพอร์ต RS232 ตัวโปรแกรมก็จะปลดล็อคเมาส์และคีย์บอร์ดให้ใช้งานได้ตามปกติ เมื่อมีการยกเลิกการใช้งานที่คอมพิวเตอร์ตัวโปรแกรมจะส่งข้อมูล :O$ เข้าที่พอร์ตอนุกรม RS232 ของเครื่องควบคุมย่อยผ่านไอซีแม็ก 232 เพื่อถอดรหัสสัญญาณจากซีเรียลเป็นทีทีแอลเข้าไปที่ Rx ของไมโครคอมพิวเตอร์ตัวที่ 2 เมื่อไมโครคอมพิวเตอร์ตัวที่ 2 ได้รับข้อมูลการเลิกใช้งานก็จะแจ้งกลับไปยังไมโครคอมพิวเตอร์ตัวที่ 1 เพื่อให้ส่งข้อมูลกลับไปที่เครื่องควบคุมหลักให้ทำการบันทึกเวลาออกรูปวงจรของเครื่องควบคุมย่อยแสดงในรูปที่ 3.11



รูปที่ 3.11 วงจรเครื่องควบคุมย่อย

3.3.2 การออกแบบทางด้านโปรแกรมของเครื่องควบคุมย่อย

ในการเขียนโปรแกรมของเครื่องควบคุมย่อยจากรูปที่ 3.12 นั้นเริ่มต้นที่ให้โปรแกรมนำค่าที่รับเข้ามาจากเครื่องควบคุมหลักผ่านทางพอร์ต RS485 มาเก็บไว้ที่รีจิสเตอร์เอจากนั้นจะตรวจสอบข้อมูลที่เข้ามาว่าตรงตามที่ตัวโปรแกรมกำหนดไว้หรือไม่ โดยแต่ละเครื่องของเครื่องควบคุมย่อยนั้นจะถูกเขียนไว้ให้รับค่าเฉพาะเป็นเครื่องๆไปเช่นเครื่องที่ 1 ก็จะถูกเขียนให้ตรวจสอบว่าข้อมูลเข้ามาใช่ :01I$ หรือไม่ถ้ายังไม่ใช่ข้อมูล :01I$ ตามที่เครื่องควบคุมย่อยเครื่องที่ 1 ต้องการเครื่องควบคุมย่อยเครื่องที่ 1 ก็จะไม่ส่งข้อมูล :I$ ออกไปที่เอาต์พุตเพื่อไปปลดล็อกเมาส์และคีย์บอร์ดการส่งข้อมูลไปปลดล็อคเมาส์และคีย์บอร์ดดูได้ในตารางที่ 3.3 การตวรจสอบข้อมูลอินพุตของเครื่องควบคุมย่อยแต่ละเครื่องตั้งแต่เครื่องที่ 1 ถึงเครื่องที่ 20 นั้นข้อมูลที่เป็นอินพุตที่ต้องการอยู่ในตารางที่ 3.4 เมื่อมีการส่งข้อมูลไปปลดล็อคเมาส์และคีย์บอร์ดแล้วขั้นต่อไปของโปรแกรมเครื่องควบคุมย่อยคือการตรวจสอบที่เอาต์พุตที่เชื่อมต่อระหว่างเครื่องควบคุมย่อยกับเครื่องคอมพิวเตอร์ว่ามีการส่งข้อมูลผ่านพอร์ต RS232 มาที่เครื่องควบคุมย่อยหรือไม่ เมื่อมีการเลิกใช้งานคอมพิวเตอร์ตัวโปรแกรมที่คอมพิวเตอร์จะส่งข้อมูล :O$ ออกทางพอร์ต RS232 ของเครื่องคอมพิวเตอร์เข้ามาที่เครื่องควบคุมย่อยเมื่อเครื่องควบคุมย่อยได้รับตรวจสอบที่เอาต์พุตแล้วว่ามีข้อมูล :O$ เข้ามาก็จะทำการส่งข้อมูลกลับไปที่เครื่องควบคุมหลักโดยข้อมูลที่ส่งกลับไปที่เครื่องควบคุมหลักนั้นส่งไปเพื่อบอกให้เครื่องควบคุมหลักให้บันทึกเวลาออกส่วนข้อมูลที่เครื่องคุมย่อยจะส่งออกไปเพื่อบอกให้เครื่องควบคุมหลักทำการบันทึกเวลาเลิกใช้งานนั้นอยู่ในตารางที่ 3.3 และตารางที่ 3.4

ตารางที่ 3.3 ข้อมูลที่เครื่องควบคุมย่อยรับจากเครื่องควบคุมหลักและข้อมูลที่ส่งไปที่เครื่องคอมพิวเตอร์

เครื่องที่	ข้อมูลที่เครื่องควบย่อยรับมาจากเครื่องควบคุมหลัก	ข้อมูลที่เครื่องควบคุมย่อยส่งไปที่โปรแกรมบนเครื่องคอมพิวเตอร์
01	:01I$	:I$
02	:02I$	:I$
03	:03I$	:I$
04	:04I$	:I$
05	:05I$	:I$
06	:06I$	:I$
07	:07I$	:I$
08	:08I$	:I$
09	:09I$	:I$
10	:10I$	:I$
11	:11I$	:I$

ตารางที่ 3.3 (ต่อ) ข้อมูลที่เครื่องควบคุมย่อยรับจากเครื่องควบคุมหลักและข้อมูลที่ส่งไปที่เครื่องคอมพิวเตอร์

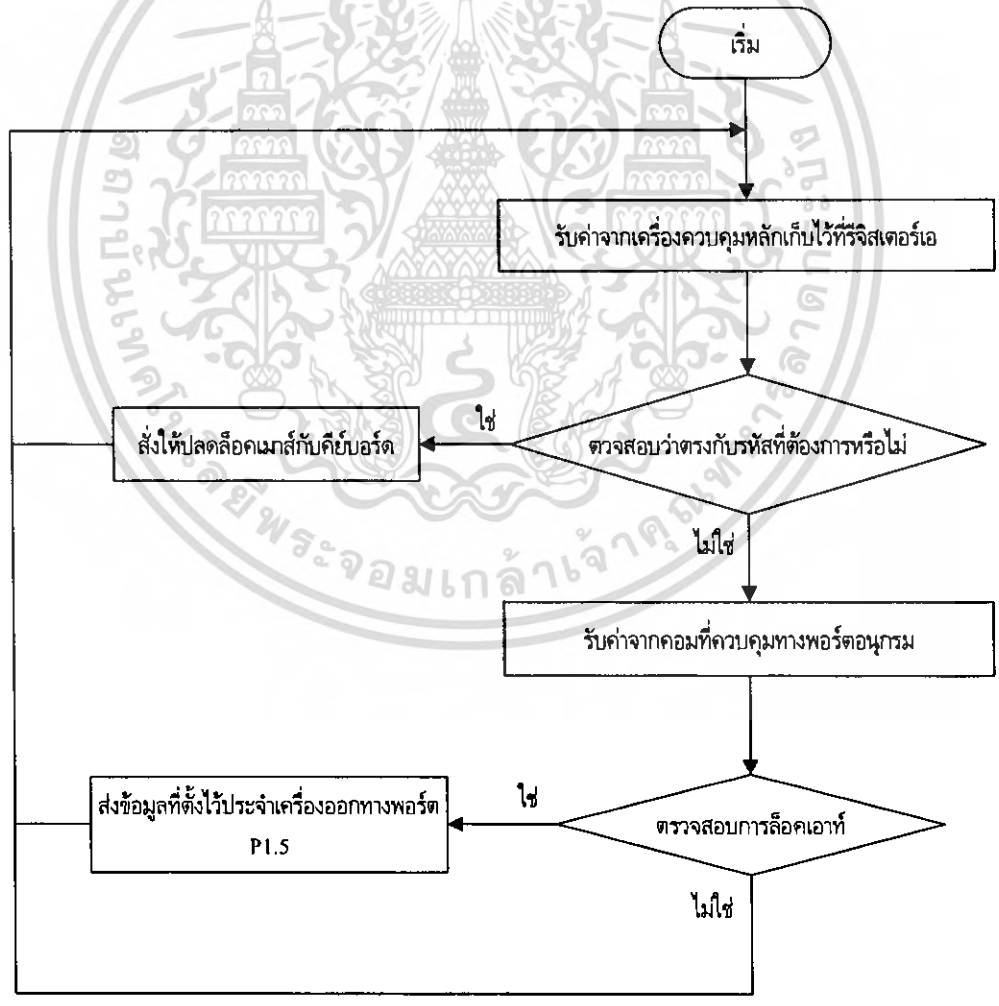
เครื่องที่	ข้อมูลที่เครื่องควบย่อยรับมาจากเครื่องควบคุมหลัก	ข้อมูลที่เครื่องควบคุมย่อยส่งไปที่โปรแกรมบนเครื่องคอมพิวเตอร์
12	:121$	:1$
13	:131$	:1$
14	:141$	:1$
15	:151$	:1$
16	:161$	:1$
17	:171$	:1$
18	:181$	:1$
19	:191$	:1$
20	:201$	:1$

ตารางที่ 3.4 ข้อมูลที่เครื่องควบคุมย่อยรับจากโปรแกรมบนเครื่องคอมพิวเตอร์และส่งข้อมูลที่ส่งกลับไปเครื่องควบคุมหลัก

เครื่องที่	ข้อมูลที่เครื่องควบย่อยรับมาจากเครื่องควบคุมคอมพิวเตอร์	ข้อมูลที่เครื่องควบคุมย่อยส่งไปที่เครื่องควบคุมหลัก
01	:O$	:01O$
02	:O$	:02O$
03	:O$	:03O$
04	:O$	:04O$
05	:O$	:05O$
06	:O$	:06O$
07	:O$	:07O$
08	:O$	:08O$
09	:O$	:09O$
10	:O$	:10O$
11	:O$	:11O$
12	:O$	:12O$
13	:O$	:13O$
14	:O$	:14O$
15	:O$	:15O$

ตารางที่ 3.4 (ต่อ) ข้อมูลที่เครื่องควบคุมย่อยรับจากโปรแกรมบนเครื่องคอมพิวเตอร์และส่งข้อมูลที่ส่งกลับไปเครื่องควบคุมหลัก

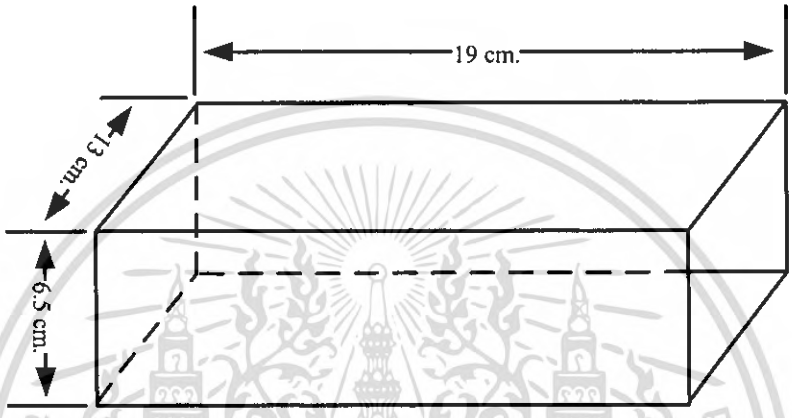
เครื่องที่	ข้อมูลที่เครื่องควบย่อยรับมาจากเครื่องควบคุมคอมพิวเตอร์	ข้อมูลที่เครื่องควบคุมย่อยส่งไปที่เครื่องควบคุมหลัก
16	:O$	:16O$
17	:O$	:17O$
18	:O$	:18O$
19	:O$	:19O$
20	:O$	:20O$



รูปที่ 3.12 ผังการทำงานของโปรแกรมควบคุมเครื่องควบคุมย่อย

3.3.3 การออกแบบทางโครงสร้างของเครื่องควบคุมย่อย

การออกแบบทางโครงสร้างของเครื่องควบคุมย่อยนั้นต้องคำนึงถึงการวางจร การวางหม้อแปลงดังนั้น เครื่องควบคุมย่อย และตำแหน่งของการวางพอร์ตในการเชื่อมต่อระหว่างเครื่องควบคุมหลักและเครื่องคอมพิวเตอร์โดยรูปที่ 3.13 เป็นการแสดงขนาดที่เหมาะสมของเครื่องควบคุมย่อย



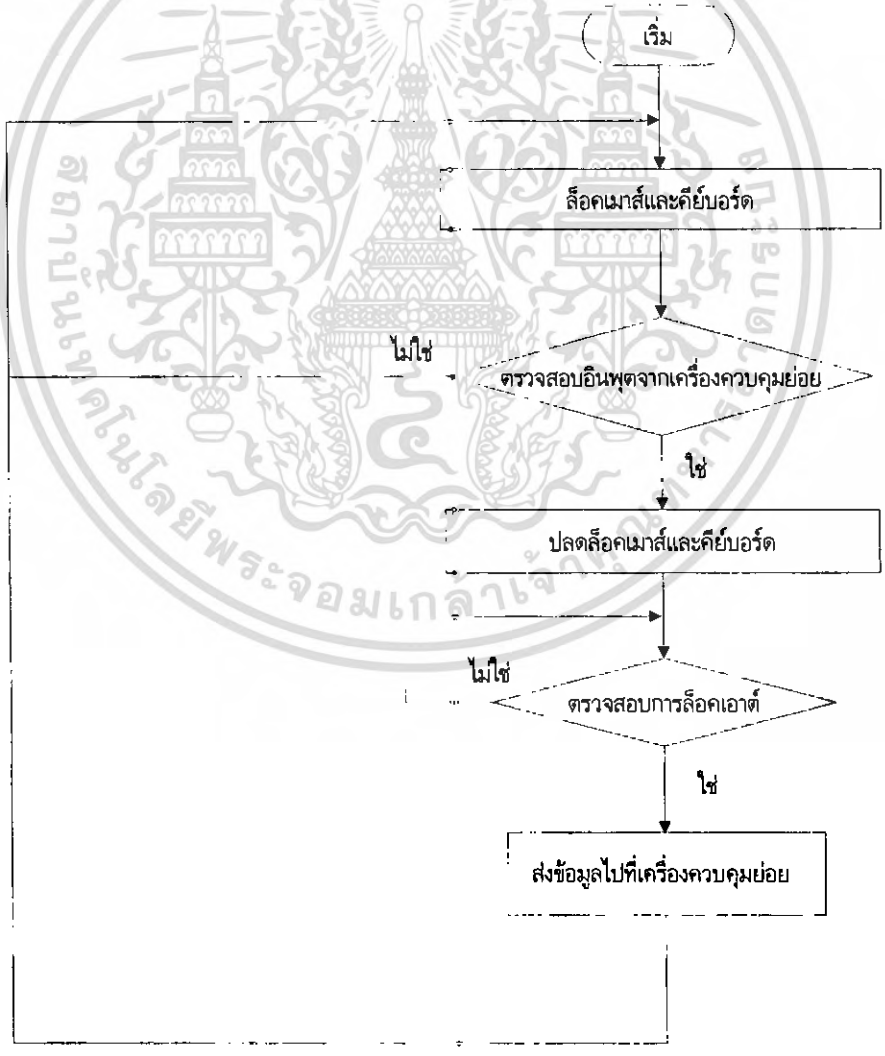
รูปที่ 3.13 กล่องเครื่องควบคุมย่อยของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

3.4 การพัฒนาซอฟต์แวร์ควบคุมคอมพิวเตอร์

ในการพัฒนาซอฟต์แวร์ควบคุมคอมพิวเตอร์ในตัวโครงงานนี้จะใช้วิชวลเบสิคในการพัฒนาโดยขั้นตอนการพัฒนาโปรแกรมจะอยู่ในรูปที่ 3.15 โดยอ้างอิงจาก API ที่ภาคผนวก ฉ ในส่วนของโปรแกรมควบคุมเมาส์และคีย์บอร์ด เมื่อเปิดโปรแกรมโปรแกรมจะทำการล็อคเมาส์และคีย์บอร์ดไว้ทำให้ไม่สามารถใช้งานได้โดยการประกาศฟังชั่นล็อคอินพุต "Private Declare Function BlockInput Lib "user32" (ByVal fBlock As Long) As Long" และตัวโปรแกรมจะทำการตวรจสอบที่พอร์ต RS232 ซึ่งเป็นพอร์ตอินพุตโดยใช้คำสั่ง "Private Sub Buffer_txt_Change( )" โดยจะตวรจสอบว่ามีข้อมูลส่งมาหรือยังโดยข้อมูลที่กำหนดไว้ให้โปรแกรมทำการปลดล็อคเมาส์และคีย์บอร์ดคือ :I$ จะเป็นข้อมูลที่บอกให้โปรแกรมปลดล็อคเมาส์และคีย์บอร์ดโดยการตรวจสอบนั้นจะกำหนดให้ ":" เป็นบิตสตาร์ทแทนด้วยตัวแปร s และให้บิตสต๊อปคือ "$" แทนตัวแปร "f" เมื่อตรวจสอบได้ตัวโปรแกรมจะนำค่าหลังบิตสตาร์ทและบิตสต๊อปมาทำการตรวจสอบว่าใช่ข้อมูลที่ต้องการหรือไม่ถ้าใช่ตัวโปรแกรมก็ปลดล็อคเมาส์และคีย์บอร์ดด้วยคำสั่ง "BlockInput False" และเปลี่ยนรูปที่หน้าจอคอมพิวเตอร์จากรูปที่ 3.16 เป็นรูปที่ 3.17 จากนั้นก็สามารถใช้งานคอมพิวเตอร์ได้อย่างปกติและเมื่อผู้ใช้งานได้ใช้งานคอมพิวเตอร์เสร็จและต้องการเลิกใช้งานก็ให้กดปุ่ม "LOCK OUT เมื่อกด "LOCK OUT แล้วตัวโปรแกรมที่ควบคุมเมาส์และคีย์บอร์ดก็จะส่งข้อมูล :O$ ออกที่พอร์ต RS232 ด้วยคำสั่ง "MSComm1.Output = ":O$" " เพื่อให้เครื่องควบคุมย่อยทราบว่ามีการล็อคเอาต์

และตัวโปรแกรมก็จะทำการล็อคเมาส์และคีย์บอร์ดด้วยคำสั่ง "BlockInput True"เพียงเท่านี้เครื่องคอมพิวเตอร์เครื่องนั้นก็ไม่สามารถใช้งานเมาส์และคีย์บอร์ดได้แล้วและภาพหน้าจอเครื่องคอมพิวเตอร์ก็จะเปลี่ยนจากรูปที่ 3.17 เป็นรูปที่ 3.16

เมื่อเปิดเครื่องคอมพิวเตอร์ครั้งแรกเจ้าของระบบต้องรันโปรแกรมขึ้นมาเองทำให้ผู้ใช้จะยังไม่สามารถใช้เมาส์และคีย์บอร์ดได้ จนกว่าจะทำการตรวจสอบจากเครื่องหลักเสียก่อน โดยหลังจากเครื่องหลักทำการตรวจสอบสิทธิ์การใช้แล้ว ถ้าผ่านเครื่องหลักจะส่งข้อมูลคำสั่งมายังเครื่องที่รันโปรแกรมนี้รออยู่ผ่านทางพอร์ตอนุกรมจากนั้นโปรแกรมจะทำการตรวจสอบว่าข้อมูลที่ส่งมานั้นเป็นข้อมูลอะไร และในทางกลับกันหลังจากทำการล็อกเอาต์แล้ว โดยโปรแกรมจะทำการส่งข้อมูลกลับไปยังเครื่องหลักเพื่อแจ้งให้เครื่องหลักทราบว่าทำการล็อกเอาต์แล้วโดยมีรูปแบบดังนี้

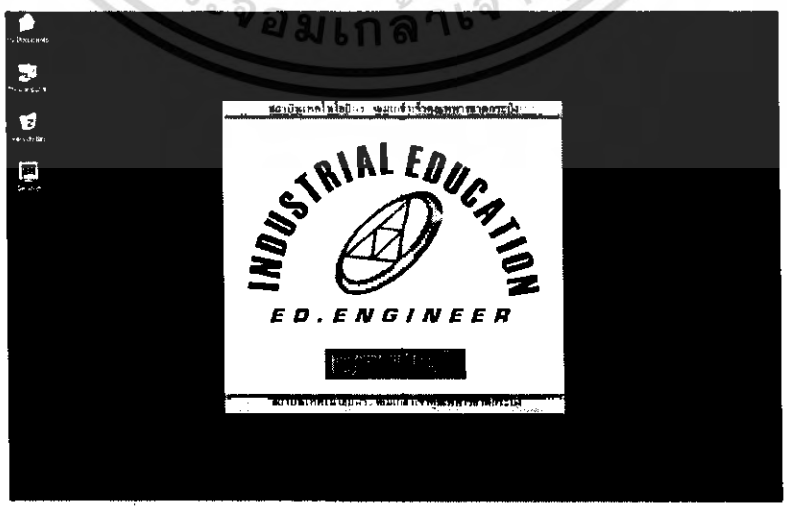


รูปที่ 3.14 ผังการทำงานของโปรแกรมควบคุมเครื่องคอมพิวเตอร์

ตารางที่ 3.5 การรับและส่งข้อมูลของตัวโปรแกรม

รูปแบบข้อมูล	คำสั่งที่กำหนดไว้
:O$	จำกัดการใช้ส่วนนำเข้าข้อมูลเครื่องคอมพิวเตอร์โดยทำการล็อคเมาท์และคีย์บอร์ด ผ่านทางรีจิสเตอร์คีย์ในไลบรารี user32 ของฟังก์ชั่น block input ของวินโดวส์จากตัวโปรแกรม
:I$	ปลดล็อคเมาท์และคีย์บอร์ด

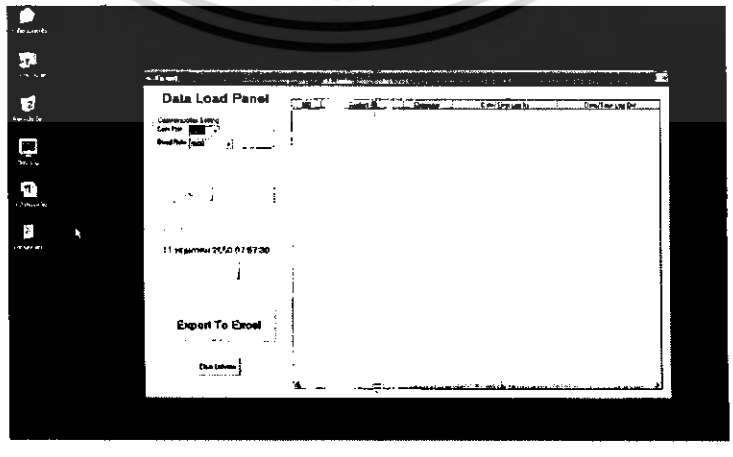
รูปที่ 3.15 หน้าจอระบบขณะเปิดเครื่องครั้งแรกหรือยังไม่ทำการล็อกอิน



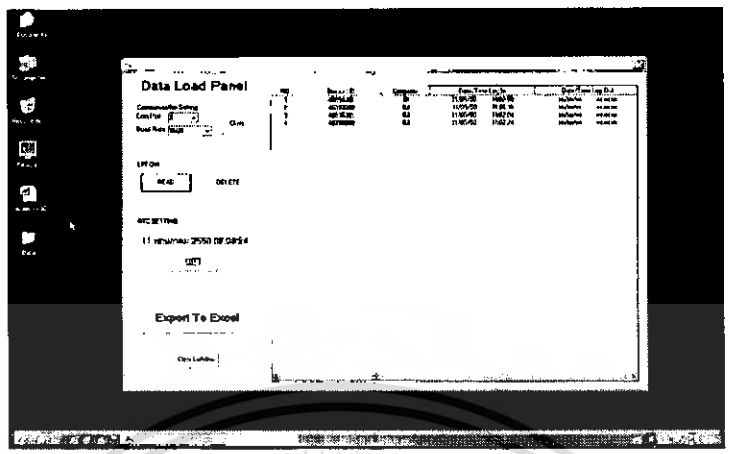
รูปที่ 3.16 หน้าจอระบบหลังจากทำการล็อกอินแล้ว

3.5 การพัฒนาซอฟต์แวร์โหลดค่าบันทึกการใช้งานจากอีอีพรอม

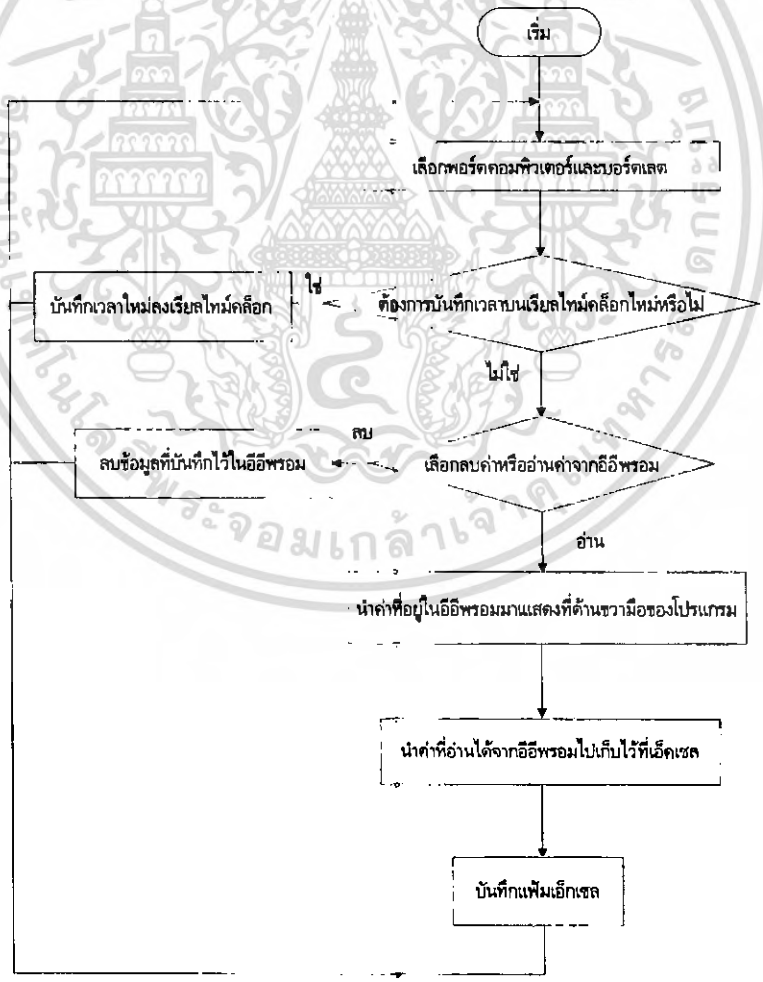
โปรแกรมที่ใช้โหลดค่าการใช้งานในอีอีพรอมมาเก็บที่เครื่องคอมพิวเตอร์นั้นใช้โปรแกรมวิชวลเบสิคในการพัฒนาโดยขั้นตอนการพัฒนาโปรแกรมแสดงในรูปที่ 3.19 โดยใช้ชื่อโปรแกรมว่า Data Load Panel ดังแสดงในรูปที่ 3.17 โดยคุณสมบัติของตัวโปรแกรมมีดังนี้ (1) ส่วนของ Communication Setting มี 2 หัวข้อย่อยคือ 1.1 Com Port เป็นการเลือกพอร์ตที่คอมพิวเตอร์ที่ใช้โหลดค่าจากเครื่องควบคุมหลัก 1.2 Baud Rate เป็นตัวกำหนดอัตราการส่งข้อมูลระหว่างเครื่องควบคุมหลักกับเครื่องคอมพิวเตอร์โดยตัวโครงงานนี้ใช้ Baud Rate ที่ 9600 บิต/วินาที (bps) (2) ส่วนของ EEPROM จะมีตัวเลือกอยู่ 2 ตัวคือ Read และ Delete หากเลือก Read จะเป็นการโหลดค่าจากอีอีพรอมมาเก็บที่ช่องด้านขวามือของโปรแกรม ดังแสดงในรูปที่ 3.18 ซึ่งจะแยกข้อมูลที่ถูกบันทึกเก็บไว้มาเก็บเป็นช่องแยกให้เห็นอย่างชัดเจนเช่น รหัสนักศึกษา เครื่องคอมพิวเตอร์ที่เลือกใช้งาน วัน เดือน ปี เวลา ที่เข้าใช้งาน และเวลาที่เลิกใช้งานแสดงเป็นส่วนๆ เพื่อง่ายแก่การตรวจสอบ หากเลือกปุ่ม Delete จะเป็นการลบข้อมูลจากอีอีพรอมทั้งหมดตัวโปรแกรมไม่สามารถเลือกลบข้อมูลบางส่วนที่ไม่ต้องการได้ (3) RTC SETTING เป็นการตั้งเวลาให้กับ RTC (Real Time Clock) เพราะบางครั้งเมื่อมีการใช้งานไปนานๆ เวลาที่แสดงไว้ที่เครื่องควบคุมหลักก็จะมีการผิดเพี้ยนไปเพียงแค่เลือกปุ่ม SET ตัวโปรแกรมก็จะทำการโปรแกรมเวลาใหม่ให้ RTC (Real Time Clock) ทันทีการใช้ตัวโปรแกรม Data Load Panel นี้มาแก้ไขเวลาที่ RTC (Real Time Clock) ทำให้ง่าย และใช้เวลาที่น้อยกว่าการใช้ไมโครคอนโทรลเลอร์ในการติดต่อกับ RTC (Real Time Clock) เพื่อแก้ไขข้อมูลต่างๆ ใน RTC (Real Time Clock) (4) Export To Excel เป็นการนำข้อมูลที่บันทึกการใช้งานคอมพิวเตอร์ที่แสดงทางด้านขวามือของโปรแกรมที่ได้จากการโหลดจากอีอีพรอมไปเก็บไว้ที่โปรแกรม Excel ที่มีอยู่ในเครื่องคอมพิวเตอร์ จากนั้นเจ้าของระบบก็สามารถเลือกที่จะเก็บข้อมูลการใช้งานที่โหลดได้อยู่ในโปรแกรม Excel นั้นเก็บไว้ส่วนใดก็ได้ตามที่เจ้าของระบบต้องการ



รูปที่ 3.17 หน้าจอโปรแกรม Data Load Panel



รูปที่ 3.18 หน้าจอโปรแกรม Data Load Panel เมื่อโหลดค่าจากอีอีพรอมมาเก็บที่โปรแกรมแล้ว



รูปที่ 3.19 ผังการทำงานของโปรแกรมโหลดค่าที่บันทึกจากเครื่องควบคุมหลัก

บทที่ 4

การทดลอง และผลการทดลอง

4.1 การทดลองตรวจสอบสัญญาณทางเอาต์พุตของตัวควบคุมหลัก

4.1.1 ลำดับขั้นการทดลอง

1) ต่อแหล่งจ่ายไฟเข้ากับเครื่องควบคุมหลัก
2) ต่อพอร์ตเอาต์พุตของเครื่องควบคุมหลักเข้ากับเครื่องคอมพิวเตอร์เพื่อตรวจสอบค่า
3) ต่อแหล่งจ่ายไฟเข้ากับวงจร
4) ทำการรูดบัตรนักศึกษา
5) สังเกตผลการเปลี่ยนแปลงที่จอแอลซีดีที่เครื่องควบคุมหลัก บันทึกผลการทดลอง
6) กดเลือกเครื่องที่ต้องการ
7) สังเกตผลที่หน้าจอคอมพิวเตอร์ บันทึกผลการทดลอง

4.1.2 ผลการทดลอง

จากการทดลองการรูดบัตรนักศึกษาและการกดรหัสโดยฐานข้อมูลที่บันทึกเก็บไว้รหัส 48035281 48035301 และ 48035314 และลองกดเลือกเครื่องตามลำดับเมื่อลองรูดบัตรนักศึกษาจะได้ผลตามตารางที่ 4.1

ตารางที่ 4.1 ผลการทดลองการรูดบัตรและตรวจสอบข้อมูลที่ส่งออกทางเอาต์พุตของเครื่องควบคุมหลัก

ครั้งที่	รหัส นักศึกษา	จอแอลซีดี ก่อนรูดบัตร	จอแอลซีดีหลังรูดบัตร	กดเลือก เครื่องที่	เอาต์พุตของ เครื่องควบคุม หลัก
1	ไม่ได้รูด	"INSERT CARD"	"INSERT CARD"	01-20	:01I$-:20R$
2	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	01	:02I$
3	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	02	:03I$
4	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	03	:04I$
5	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	04	:04I$
6	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	05	:05I$
7	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	06	:06I$
8	48035314	"INSERT CARD"	"CHOOSE COMPUTER"	07	:07I$
9	48035281	"INSERT CARD"	"CHOOSE COMPUTER"	08	:08I$

ตารางที่ 4.1 (ต่อ) ผลการทดลองการรูดบัตรและตรวจสอบข้อมูลที่ส่งออกทางเอาต์พุตของเครื่องควบคุมหลัก

ครั้งที่	รหัสนักศึกษา	จอแอลซีดีก่อนรูดบัตร	จอแอลซีดีหลังรูดบัตร	กดเลือกเครื่องที่	เอาต์พุตของเครื่องควบคุมหลัก
10	48035281	"INSERT CARD"	"CHOOSE COMPUTER"	09	:09I$
11	48035281	"INSERT CARD"	"CHOOSE COMPUTER"	10	:10I$
12	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	11	:11I$
13	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	12	:12I$
14	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	13	:13I$
15	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	14	:14I$
16	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	15	:15I$
17	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	16	:16I$
18	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	17	:17I$
19	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	18	:18I$
20	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	19	:19I$
21	48035301	"INSERT CARD"	"CHOOSE COMPUTER"	20	:20I$
22	48035400	"INSERT CARD"	"INSERT CARD"	01	:01I$-:20R$
23	48035256	"INSERT CARD"	"INSERT CARD"	02	:01I$-:20R$
24	48035201	"INSERT CARD"	"INSERT CARD"	03	:01I$-:20R$
25	48035444	"INSERT CARD"	"INSERT CARD"	04	:01I$-:20R$
26	48035350	"INSERT CARD"	"INSERT CARD"	05	:01I$-:20R$
27	48035389	"INSERT CARD"	"INSERT CARD"	06	:01I$-:20R$
28	48035244	"INSERT CARD"	"INSERT CARD"	07	:01I$-:20R$
29	48035248	"INSERT CARD"	"INSERT CARD"	08	:01I$-:20R$
30	48035251	"INSERT CARD"	"INSERT CARD"	09	:01I$-:20R$
31	48035253	"INSERT CARD"	"INSERT CARD"	10	:01I$-:20R$
32	48035255	"INSERT CARD"	"INSERT CARD"	11	:01I$-:20R$
33	48035312	"INSERT CARD"	"INSERT CARD"	12	:01I$-:20R$
34	48035313	"INSERT CARD"	"INSERT CARD"	12	:01I$-:20R$
35	48035315	"INSERT CARD"	"INSERT CARD"	14	:01I$-:20R$
36	48035316	"INSERT CARD"	"INSERT CARD"	15	:01I$-:20R$
37	48035317	"INSERT CARD"	"INSERT CARD"	16	:01I$-:20R$
38	48035318	"INSERT CARD"	"INSERT CARD"	17	:01I$-:20R$

ตารางที่ 4.1 (ต่อ) ผลการทดลองการรูดบัตรและตรวจสอบข้อมูลที่ส่งออกทางเอาต์พุตของเครื่องควบคุมหลัก

ครั้งที่	รหัสนักศึกษา	จอแอลซีดีก่อนรูดบัตร	จอแอลซีดีหลังรูดบัตร	กดเลือกเครื่องที่	เอาต์พุตของเครื่องควบคุมหลัก
39	48035319	"INSERT CARD"	"INSERT CARD"	18	:01I$-:20R$
40	48035320	"INSERT CARD"	"INSERT CARD"	19	:01I$-:20R$
41	48035325	"INSERT CARD"	"INSERT CARD"	20	:01I$-:20R$

4.2 การทดลองการตรวจสอบข้อมูลที่พอร์ตเอาต์พุตของเครื่องควบคุมย่อย

4.2.1 ลำดับขั้นการทดลอง

1) ต่อเครื่องควบคุมหลักเข้ากับเครื่องควบคุมย่อย
2) ต่อเอาต์พุตของเครื่องควบคุมย่อยเข้ากับเครื่องคอมพิวเตอร์
3) ต่อแหล่งจ่ายไฟเข้ากับวงจร
4) รูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่อง
5) สังเกตผล และบันทึกผลการทดลอง

4.2.2 ผลการทดลอง

จากการทดลองต่อเครื่องควบคุมหลักเข้ากับเครื่องควบคุมย่อยแล้วรูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่องได้ผลการทดลองแสดงดังตารางที่ 4.2

ตารางที่ 4.2 ผลการทดลองการตรวจสอบข้อมูลที่พอร์ตเอาต์พุตของเครื่องควบคุมย่อย

เลือกเครื่องที่	เอาต์พุตของเครื่องควบคุมย่อยที่ 1	เอาต์พุตของเครื่องควบคุมย่อยที่ 2	เอาต์พุตของเครื่องควบคุมย่อยที่3	เอาต์พุตของเครื่องควบคุมย่อยที่ 4	เอาต์พุตของเครื่องควบคุมย่อยที่ 5
01	:I$	1	1	1	1
02	1	:I$	1	1	1
03	1	1	:I$	1	1
04	1	1	1	:I$	1
05	1	1	1	1	:I$
06	1	1	1	1	1
07	1	1	1	1	1

ตารางที่ 4.2(ต่อ) ผลการทดลองการตรวจสอบข้อมูลที่พอร์ตเอาต์พุตของเครื่องควบคุมย่อย

เลือกเครื่องที่	เอาต์พุตของเครื่องควบคุมย่อยที่ 1	เอาต์พุตของเครื่องควบคุมย่อยที่ 2	เอาต์พุตของเครื่องควบคุมย่อยที่3	เอาต์พุตของเครื่องควบคุมย่อยที่ 4	เอาต์พุตของเครื่องควบคุมย่อยที่ 5
08	1	1	1	1	1
09	1	1	1	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	1	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	1	1	1	1	1
20	1	1	1	1	1

4.3 การทดลองโปรแกรมเมื่อรับคำสั่งจากตัวควบคุมย่อย

4.3.1 ลำดับขั้นการทดลอง

1) ต่อเครื่องควบคุมหลักเครื่องควบคุมย่อย

2) ต่อเอาต์พุตของเครื่องควบคุมย่อยเข้ากับเครื่องคอมพิวเตอร์ที่ลงโปรแกรมควบคุมเมาส์และคีย์บอร์ด

3) ต่อแหล่งจ่ายไฟเข้ากับวงจร

4) รูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่อง

5) สังเกตผล และบันทึกผลการทดลอง

4.3.2 ผลการทดลอง

จากการจากการทดลองต่อเครื่องควบคุมหลัก เครื่องควบคุมย่อยและต่อเข้ากับเครื่องคอมพิวเตอร์ที่ลงโปรแกรมควบคุมเมาส์และคีย์บอร์ดแล้วรูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่องได้ผลการทดลองแสดงดังตารางที่ 4.3

ตารางที่ 4.3 ผลการทดลองโปรแกรมเมื่อรับคำสั่งจากตัวควบคุมย่อย

เลือกเครื่องที่	โปรแกรมของเครื่องที่ 1	โปรแกรมของเครื่องที่ 2	โปรแกรมของเครื่องที่ 3	โปรแกรมของเครื่องที่ 4	โปรแกรมของเครื่องที่ 5
00	LOCK	LOCK	LOCK	LOCK	LOCK
01	UNLOCK	LOCK	LOCK	LOCK	LOCK
02	LOCK	UNLOCK	LOCK	LOCK	LOCK
03	LOCK	LOCK	UNLOCK	LOCK	LOCK
04	LOCK	LOCK	LOCK	UNLOCK	LOCK
05	LOCK	LOCK	LOCK	LOCK	UNLOCK
06	LOCK	LOCK	LOCK	LOCK	LOCK
07	LOCK	LOCK	LOCK	LOCK	LOCK
08	LOCK	LOCK	LOCK	LOCK	LOCK
09	LOCK	LOCK	LOCK	LOCK	LOCK
10	LOCK	LOCK	LOCK	LOCK	LOCK
11	LOCK	LOCK	LOCK	LOCK	LOCK
12	LOCK	LOCK	LOCK	LOCK	LOCK
13	LOCK	LOCK	LOCK	LOCK	LOCK
14	LOCK	LOCK	LOCK	LOCK	LOCK
15	LOCK	LOCK	LOCK	LOCK	LOCK
16	LOCK	LOCK	LOCK	LOCK	LOCK
17	LOCK	LOCK	LOCK	LOCK	LOCK
18	LOCK	LOCK	LOCK	LOCK	LOCK
19	LOCK	LOCK	LOCK	LOCK	LOCK
20	LOCK	LOCK	LOCK	LOCK	LOCK

4.4 การทดลองการส่งข้อมูลเมื่อเลิกใช้โปรแกรม

4.4.1 ลำดับขั้นการทดลอง

1) ต่อเครื่องควบคุมหลักเครื่องควบคุมย่อย

2) ต่อเอาต์พุตของเครื่องควบคุมย่อยเข้ากับเครื่องคอมพิวเตอร์ที่ลงโปรแกรมควบคุมเมาส์และคีย์บอร์ด

3) ต่อแหล่งจ่ายไฟเข้ากับวงจร

4) รูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่อง

5) ทำการกดปุ่มล็อคเอาต์เพื่อเลิกใช้งาน

6) สังเกตผล และบันทึกผลการทดลอง

4.4.2 ผลการทดลอง

จากการจากการทดลองต่อเครื่องควบคุมหลัก เครื่องควบคุมย่อยและต่อเข้ากับเครื่องคอมพิวเตอร์ที่ลงโปรแกรมควบคุมเมาส์และคีย์บอร์ดแล้วรูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่องได้ผลการทดลองแสดงดังตารางที่ 4.4

ตารางที่ 4.4 ผลการทดลองโปรแกรมเมื่อเลิกใช้โปรแกรม

เครื่องที่	ข้อมูลที่โปรแกรมส่งไปที่เครื่องควบคุมย่อยเมื่อเลิกใช้งาน	ข้อมูลที่เครื่องควบคุมย่อยส่งไปที่เครื่องควบคุมหลักเมื่อเลิกใช้งาน
01	:O$	:01O$
02	:O$	:02O$
03	:O$	:03O$
04	:O$	:04O$
05	:O$	:05O$
06	:O$	:06O$
07	:O$	:07O$
08	:O$	:08O$
09	:O$	:09O$
10	:O$	:10O$
11	:O$	:11O$
12	:O$	:12O$
13	:O$	:13O$
14	:O$	:14O$
15	:O$	:15O$
16	:O$	:16O$
17	:O$	:17O$
18	:O$	:18O$
19	:O$	:19O$
20	:O$	:20O$

4.5 การทดลองหลอดแอลอีดีแสดงสถานะการเข้าใช้งานคอมพิวเตอร์

4.5.1 ลำดับขั้นการทดลอง

1) ต่อเครื่องควบคุมหลักเข้ากับเครื่องควบคุมย่อย

2) ต่อเอาต์พุตของเครื่องควบคุมย่อยเข้ากับเครื่องคอมพิวเตอร์

3) ต่อแหล่งจ่ายไฟเข้ากับวงจร

4) รูดบัตรที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่อง

5) สังเกตผล และบันทึกผลการทดลอง

6) เข้าใช้งานคอมพิวเตอร์แล้วกดล็อคเอาต์ที่คอมพิวเตอร์

7) สังเกตผล และบันทึกผลการทดลอง

4.5.2 ผลการทดลอง

จากการจากการทดลองต่อเครื่องควบคุมหลักเข้ากับเครื่องควบคุมย่อยแล้วรูดบัตรนักศึกษาที่สามารถเข้าใช้งานได้แล้วกดเลขเครื่องได้ผลการทดลองแสดงดังตารางที่ 4.5

ตารางที่ 4.5 ผลการทดลองหลอดแอลอีดีแสดงสถานการณ์เข้าใช้งานคอมพิวเตอร์

เลือกเครื่องเครื่องที่	หลอดที่ดับ	เลิกใช้งานเครื่องที่	สถานะของหลอดที่เคยดับ
01	01	01	หลอดที่ 01 กลับมาติดเหมือนเดิม
02	02	02	หลอดที่ 02 กลับมาติดเหมือนเดิม
03	03	03	หลอดที่ 03 กลับมาติดเหมือนเดิม
04	04	04	หลอดที่ 04 กลับมาติดเหมือนเดิม
05	05	05	หลอดที่ 05 กลับมาติดเหมือนเดิม
06	06	06	หลอดที่ 06 กลับมาติดเหมือนเดิม
07	07	07	หลอดที่ 07 กลับมาติดเหมือนเดิม
08	08	08	หลอดที่ 08 กลับมาติดเหมือนเดิม
09	09	09	หลอดที่ 09 กลับมาติดเหมือนเดิม
10	10	10	หลอดที่ 10 กลับมาติดเหมือนเดิม
11	11	11	หลอดที่ 11 กลับมาติดเหมือนเดิม
12	12	12	หลอดที่ 12 กลับมาติดเหมือนเดิม
13	13	13	หลอดที่ 13 กลับมาติดเหมือนเดิม
14	14	14	หลอดที่ 14 กลับมาติดเหมือนเดิม
15	15	15	หลอดที่ 15 กลับมาติดเหมือนเดิม
16	16	16	หลอดที่ 16 กลับมาติดเหมือนเดิม

ตารางที่ 4.5 (ต่อ) ผลการทดลองหลอดแอลอีดีแสดงสถานการณ์เข้าใช้งานคอมพิวเตอร์

เลือกเครื่องเครื่องที่	หลอดที่ดับ	เลิกใช้งานเครื่องที่	สถานะของหลอดที่เคยดับ
17	17	17	หลอดที่ 17 กลับมาติดเหมือนเดิม
18	18	18	หลอดที่ 18 กลับมาติดเหมือนเดิม
19	19	19	หลอดที่ 19 กลับมาติดเหมือนเดิม
20	20	20	หลอดที่ 20 กลับมาติดเหมือนเดิม
21-30	ไม่มีการเปลี่ยนแปลง	ไม่มี	ไม่มี

จากการทดลองเราสามารถสรุปการผลทำงานได้ว่า การเข้าใช้งานเครื่องคอมพิวเตอร์จะต้องทำตามขั้นตอนการใช้งานดังนี้ 1. รูดบัตรนักศึกษาที่ช่องรับบัตรบาร์โค้ด 2. สังเกตการเปลี่ยนแปลงที่จอแอลซีดีถ้าสามารถใช้งานได้จอจะเปลี่ยนจากคำว่า "INSERT CARD" เป็นคำว่า "CHOOSE COMPUTER" ให้ทำขั้นตอนต่อไป 3. สังเกตหลอดแอลอีดีทางด้านขวามือของเครื่องว่าเครื่องใดว่างอยู่โดยสังเกตจากหลอดแอลอีดีหมายเลขใดติดแสดงว่าเครื่องหมายเลขนั้นยังไม่มีคนเข้าใช้หลอดหลอดที่หมายเลขใดดับแสดงว่ามีผู้เข้าใช้บริการอยู่ 4. กดเลือกเครื่องที่ต้องการใช้บริการ 5. เข้าไปใช้บริการเครื่องที่เลือก 6. เมื่อต้องการเลิกใช้งานให้กดปุ่มล็อกเอาต์ เมื่อกดปุ่มล็อคเอาต์แล้วสภาวะของหลอดแอลอีดีของเครื่องที่ล็อคเอาต์จากที่ดับก็จะกลับมาติดเหมือนเดิมในการใช้งานจริงอาจมีการผิดพลาดเกิดขึ้นเช่น เลือกเครื่องแล้วไม่สามารถเข้าใช้งานได้ไม่ได้อาจจะเกิดจากตัวโปรแกรมของเครื่องควบคุมหรือการเชื่อมต่อระหว่างอุปกรณ์ทั้งสามส่วนไม่แน่นทำให้ส่งข้อมูลไปไม่ถึงปลายทาง

บทที่ 5
บทสรุป

5.1 สรุป

ระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมสามารถที่จะบันทึกเวลาเข้าและออกของนักศึกษาได้แต่อาจจะเกิดความผิดพลาดเล็กน้อย ซึ่งระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมสามารถบันทึกเวลาวันเดือนปีที่เข้าใช้งานได้จริงและจะเก็บค่าที่บันทึกไว้ที่อีอีพรอมซึ่งมีคุณสมบัติก็คือสามารถเก็บข้อมูลได้แม้จะไม่มีไปเลี้ยงป้อนให้วงจร เครื่องควบคุมหลักนั้นมีหน่วยความจำอีอีพรอมขนาด 4 กิโลไบต์หรือ 32000 บิต การบันทึกการลงทะเบียนการใช้งานใช้หน่วยความจำ 8 ไบต์ต่อ 1 คนและการบันทึกเวลาเข้าและออกจากระบบใช้ทั้งหมด 35 ไบต์ต่อการใช้งานเข้าและออกจากระบบจำนวน 1 ครั้งสามารถบันทึกค่าการใช้งานได้คนละ 37 ครั้งจากจำนวนนักศึกษาที่อยู่ในฐานข้อมูลทั้งหมด 3 คน และขั้นตอนการใช้งานจะมีจอแอลซีดีและวิธีการใช้งานบอกขั้นตอนการทำงานทุกขั้นตอนเมื่อเข้าใช้งาน

จากผลการทดลองในการใช้งานระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมจะทำให้ประหยัดเวลาในการเก็บข้อมูลการเข้าใช้งานคอมพิวเตอร์ซึ่งมีความเหมาะสมกับผู้ที่ต้องการเก็บข้อมูลการเข้าใช้งานห้องคอมพิวเตอร์โดยไม่ต้องใช้ระบบ LAN เพราะต้องการหลีกเลี่ยงปัญหาการเกิดทราฟฟิคขึ้นในระบบ LAN ซึ่งระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมได้ถูกสร้างขึ้นมาด้วยเหตุผลนี้และสามารถนำไปใช้งานได้จริง

5.2 ปัญหาและวิธีการแก้ไข

จากการดำเนินการสร้างและทดสอบโครงงานระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมปรากฏว่ามีปัญหาเกิดขึ้น 2 ประการ ซึ่งสามารถสรุปได้ดังนี้

1. ไม่สามารถเลือกเครื่องเข้าใช้งานได้

   วิธีการแก้ไข ตั้งค่าการตรวจสอบการกดคีย์ให้นานกว่าเดิม

2. ไม่สามารถเชื่อมต่อระหว่างเครื่องควบคุมหลักและโปรแกรมบนเครื่องคอมพิวเตอร์ได้

   วิธีการแก้ไข ปรับปรุงโปรแกรมในส่วนของเครื่องควบคุมย่อยให้มีบิตสตาร์ทเป็น ":" และบิตสต๊อปเป็น "$" และให้บอร์ดเลตที่ใช้ในการติดต่อเป็น 9600 Bit/Sec

5.3 แนวทางการพัฒนา

1. เพิ่มหน่วยความจำอีอีพรอมเข้าให้กับเครื่องควบคุมหลักเพื่อให้เก็บจำนวนของข้อมูลการเข้าใช้งานได้มากขึ้น

2. ในส่วนของโปรแกรมที่ใช้ควบคุมเมาส์และคีย์บอร์ดที่เครื่องคอมพิวเตอร์ให้มีตัวบันทึกเวลาการเข้าใช้งานเพิ่มขึ้นมาเพื่อง่ายในการตรวจสอบกับข้อมูลที่บันทึกเก็บไว้ในอีอีพรอมที่อยู่ที่เครื่องควบคุมหลักเปรียบเทียบกับเวลาที่ถูกบันทึกเก็บไว้ที่เครื่องคอมพิวเตอร์

3. เพิ่มข้อความที่จอแอลซีดีหรือเปลี่ยนจอแอลซีดีเป็นชนิดที่เป็นข้อความภาษาไทย เช่น ทำให้ขั้นตอนที่แสดงเป็นภาษาไทยเพื่อให้ง่ายต่อการใช้งานเช่น เมื่อรูดบัตรแล้วไม่พบรหัสนั้นๆอยู่ในฐานข้อมูลก็ให้มีข้อความขึ้นมาบอกว่า "ไม่พบข้อมูลของท่านในฐานข้อมูล" แทน "Card ID Invalid" ที่มีอยู่เดิม

บรรณานุกรม

ธีรวัฒน์ ประกอบผล. การประยุกต์ใช้งานไมโครคอนโทรลเลอร์. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น). 2543

ธีรวัฒน์ ประกอบผล. ภาษาแอสเซมบลีสำหรับ MCS-51. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น). 2548

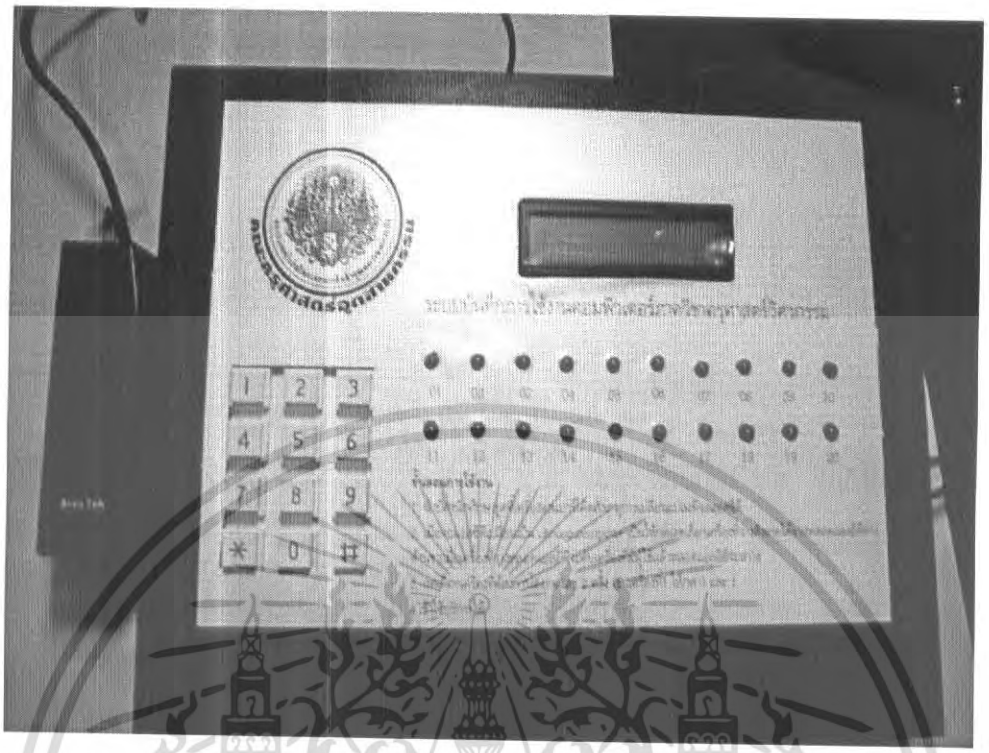
นริศรา เพชรพนาภรณ์. BARCODE. BARCODE หรือ รหัสแท่ง, สืบค้นเมื่อ 11 มกราคม 2549, จาก http://www.student.chula.ac.th/~46801474/index.html

วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรจิตรวิไล. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์แบบแฟลช ฉบับ AT89C5x ของ Atmel. กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด.

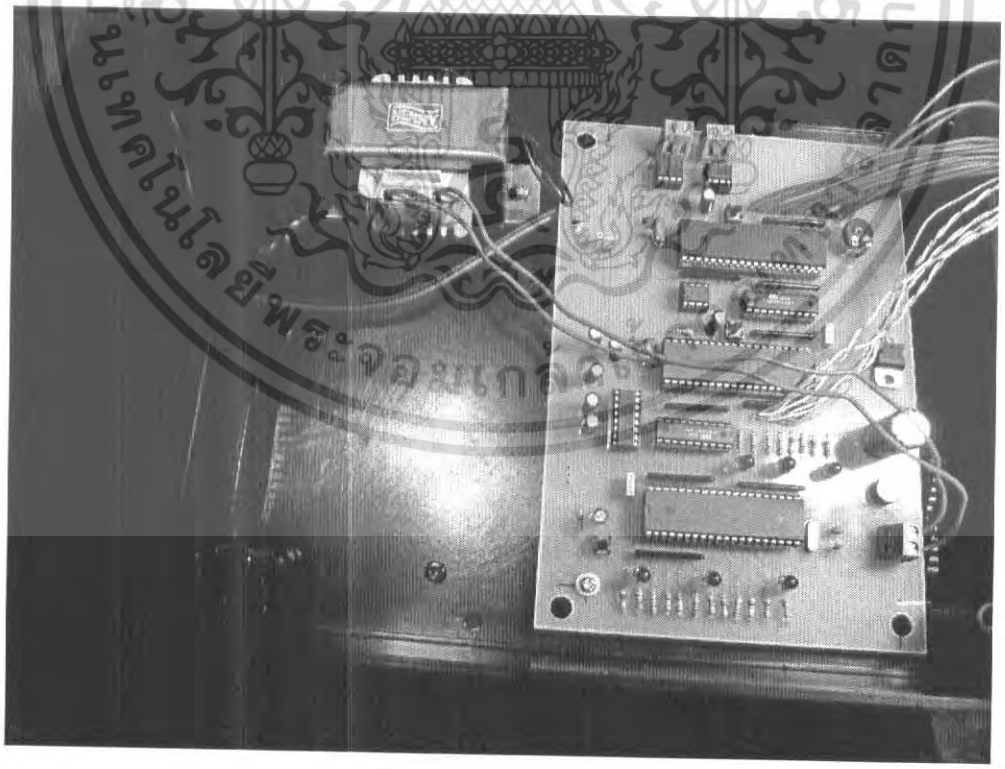
วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรจิตรวิไล. เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์แบบแฟลช ฉบับ AT89C5x/AT89Sxxx (ปรับปรุงครั้งที่ 4). กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

Global Innovative Technology. Barcode Scanner. ระบบและอุปกรณ์บาร์โค้ด (Barcode System), สืบค้นเมื่อ 10 ธันวาคม 2549, จาก Gitthailand.com/Content/barcodeintro.htm

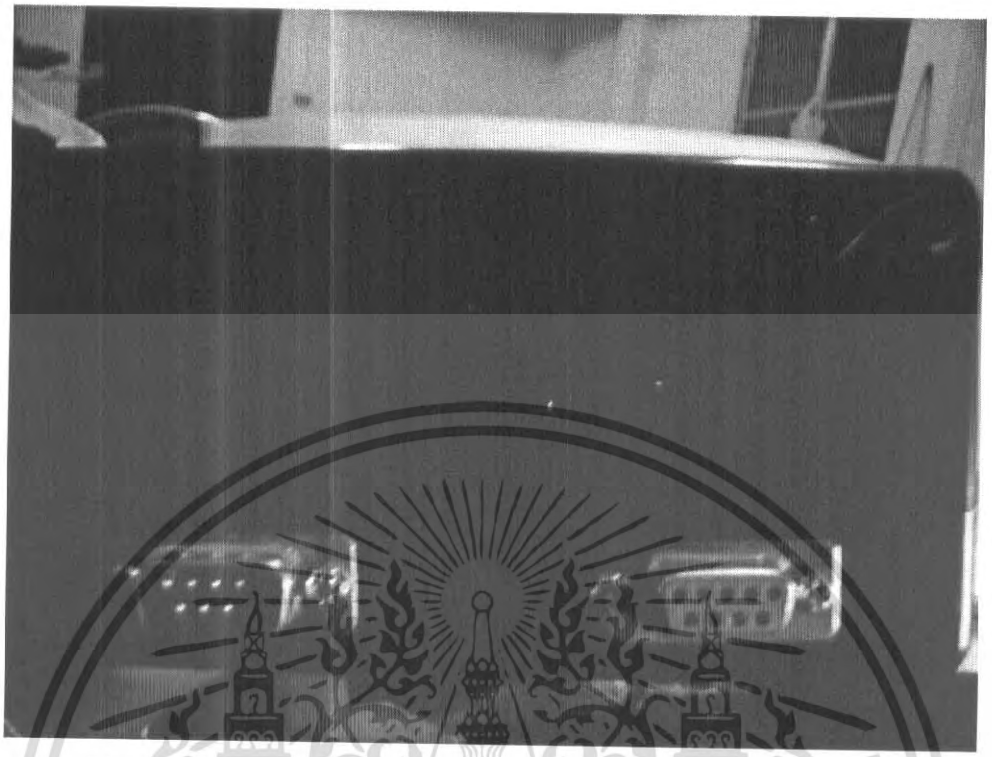
ภาคผนวก ก
เครื่องต้นแบบ



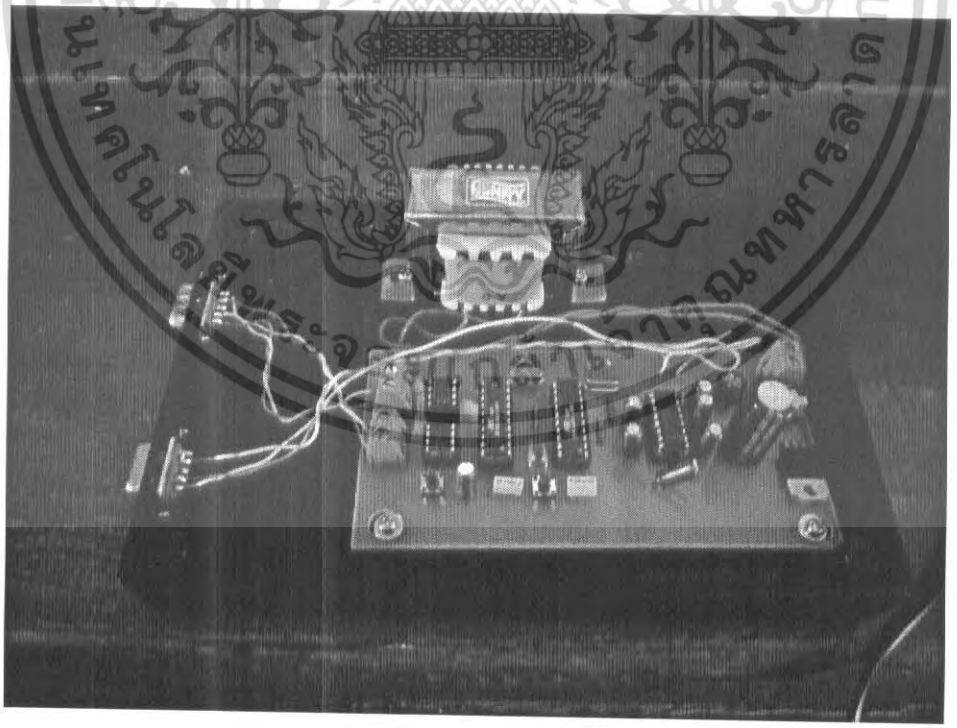
รูปที่ ก.1 เครื่องควบคุมหลัก



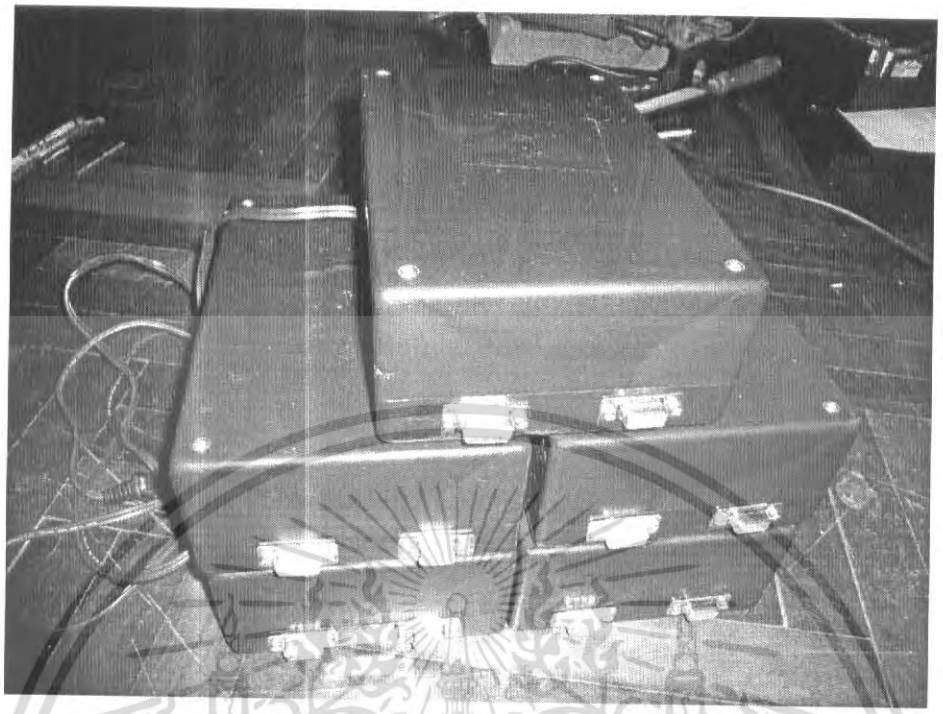
รูปที่ ก.2 ภายในเครื่องต้นแบบเครื่องควบคุมหลัก



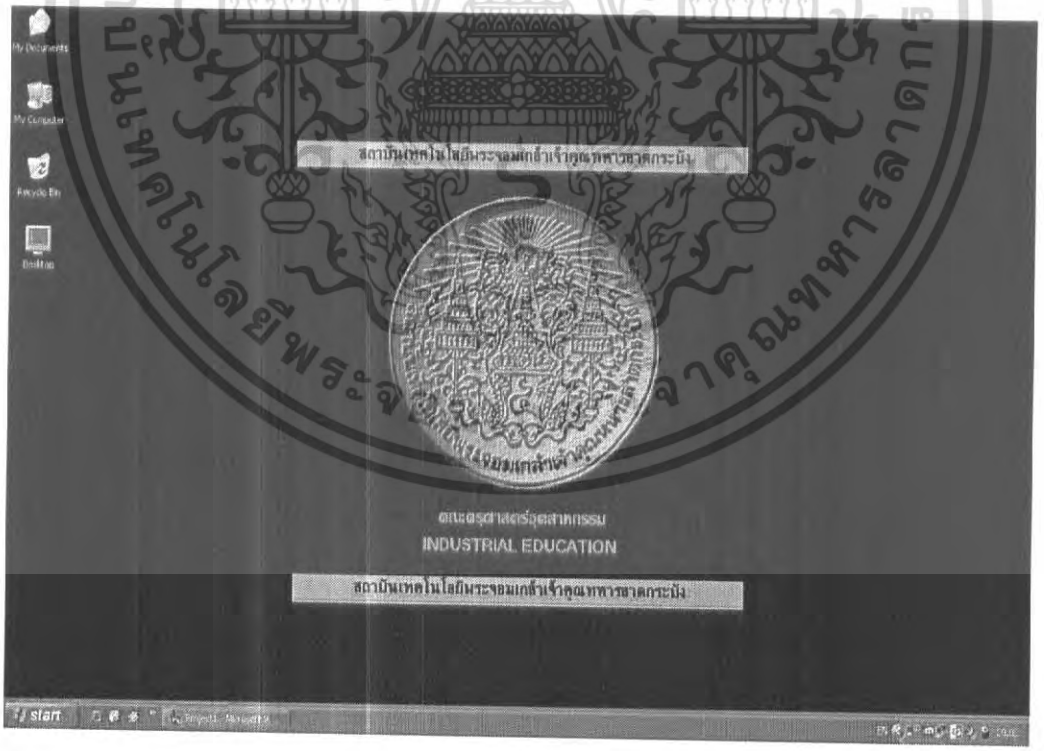
รูปที่ ก.3 พอร์ตเชื่อมต่อเครื่องควบคุมหลักและเครื่องคอมพิวเตอร์ของเครื่องควบคุมย่อย



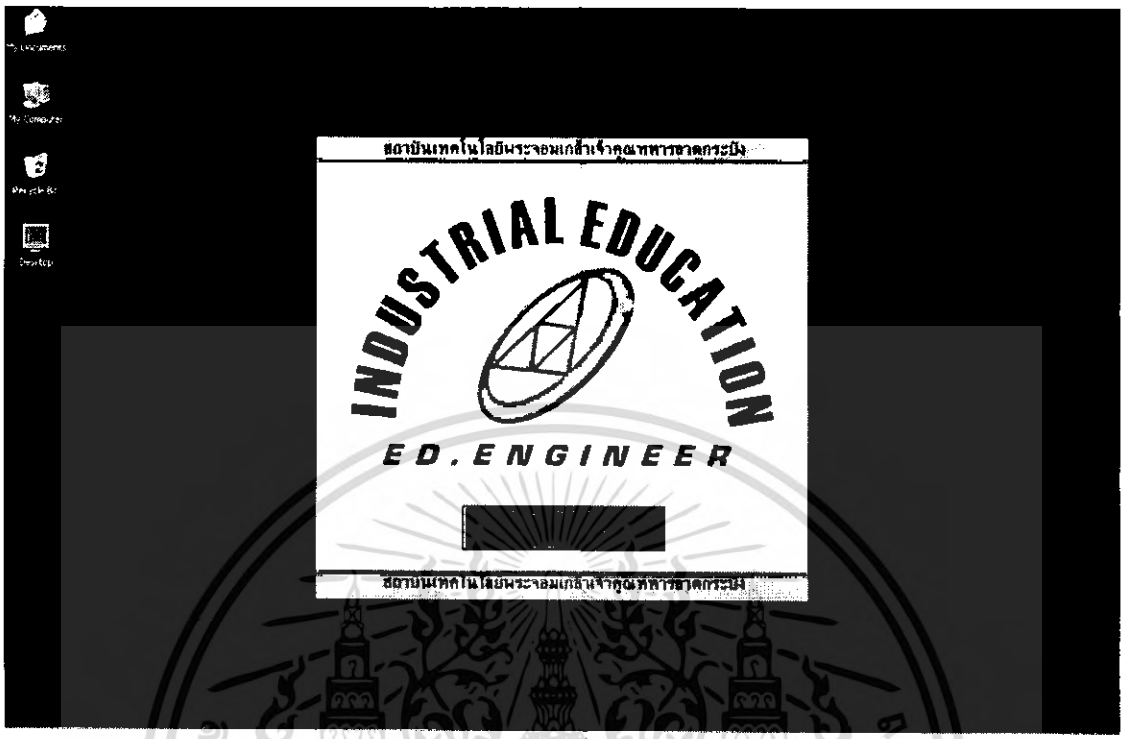
รูปที่ ก.4 ภายในเครื่องต้นแบบเครื่องควบคุมย่อย



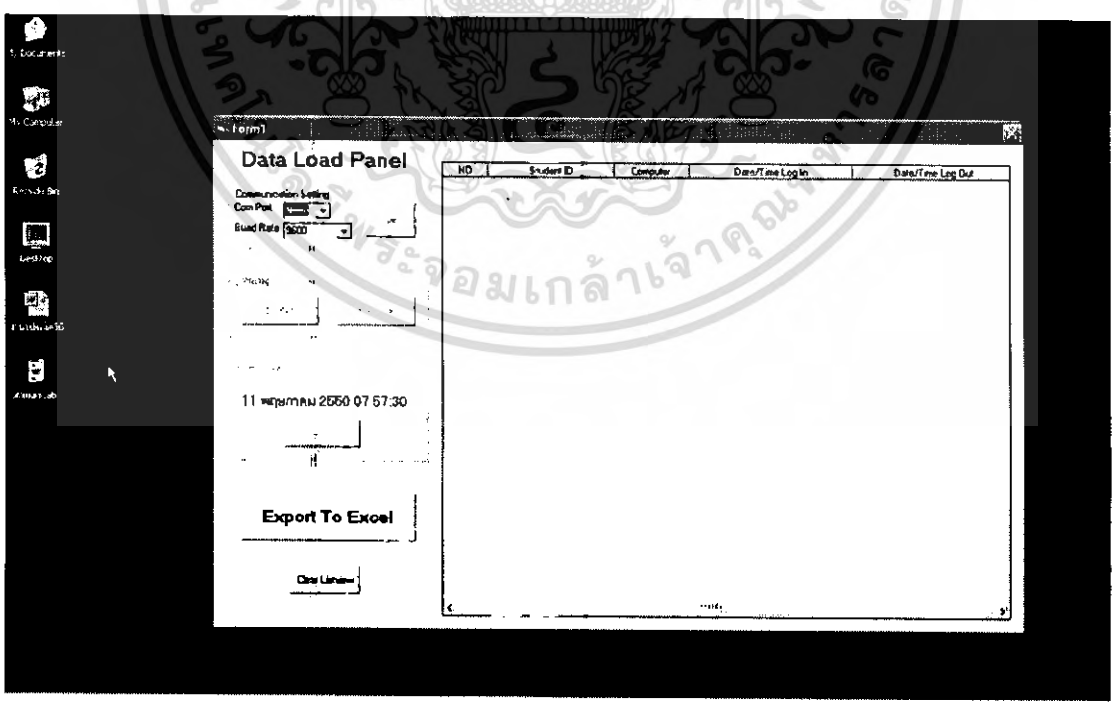
รูปที่ ก.5 เครื่องต้นแบบเครื่องควบคุมย่อยทั้งหมด



รูปที่ ก.6 หน้าจอระบบขณะเปิดเครื่องครั้งแรกหรือยังไม่ทำการล็อกอิน



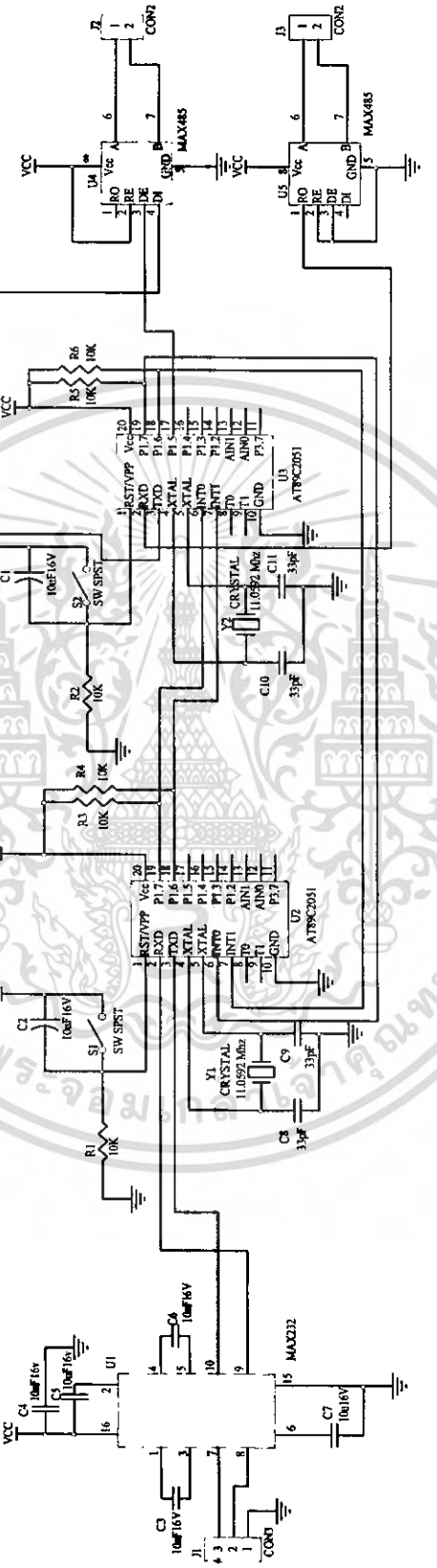
รูปที่ ก.7 หน้าจอระบบหลังจากทำการล็อกอินแล้ว



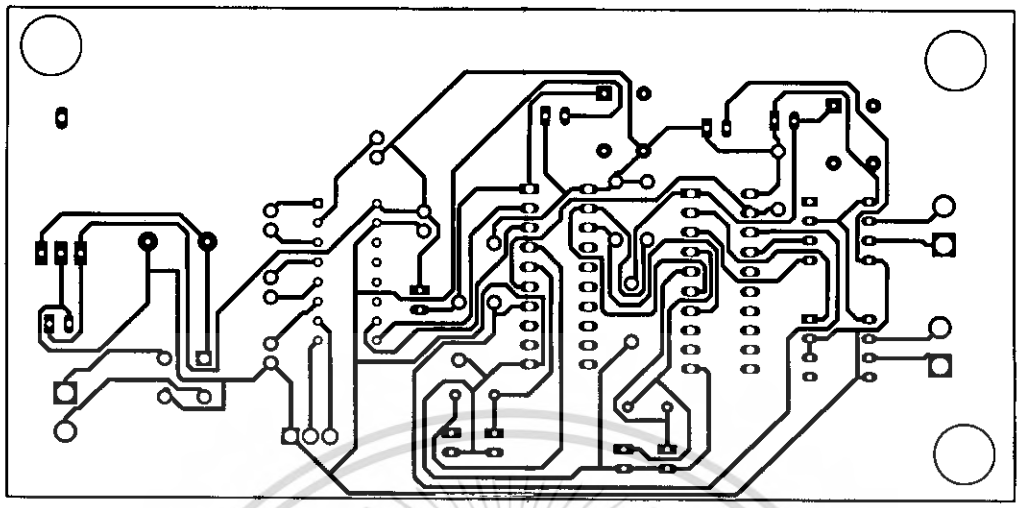
รูปที่ ก.8 หน้าจอโปรแกรม Data Load Panel

ภาคผนวก ข

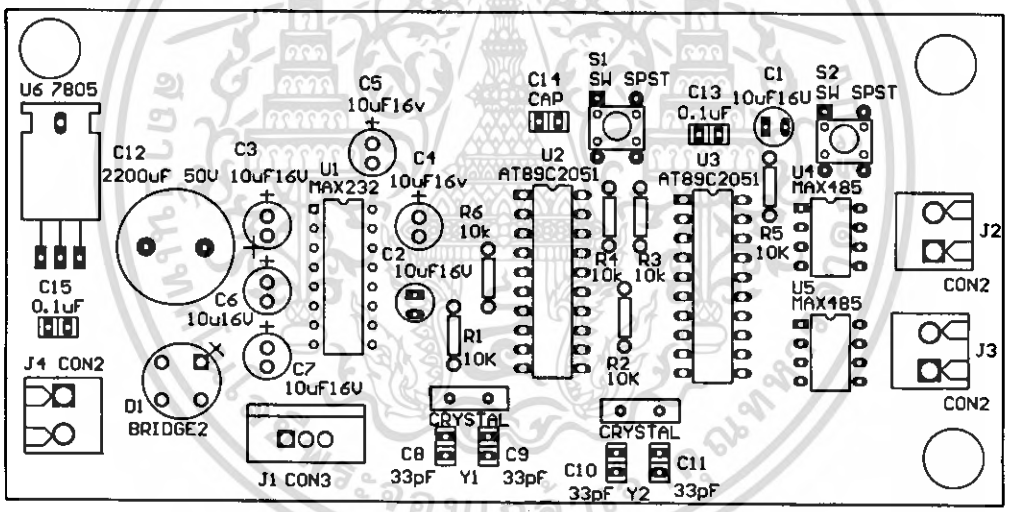
วงจรและแผ่นวงจรพิมพ์



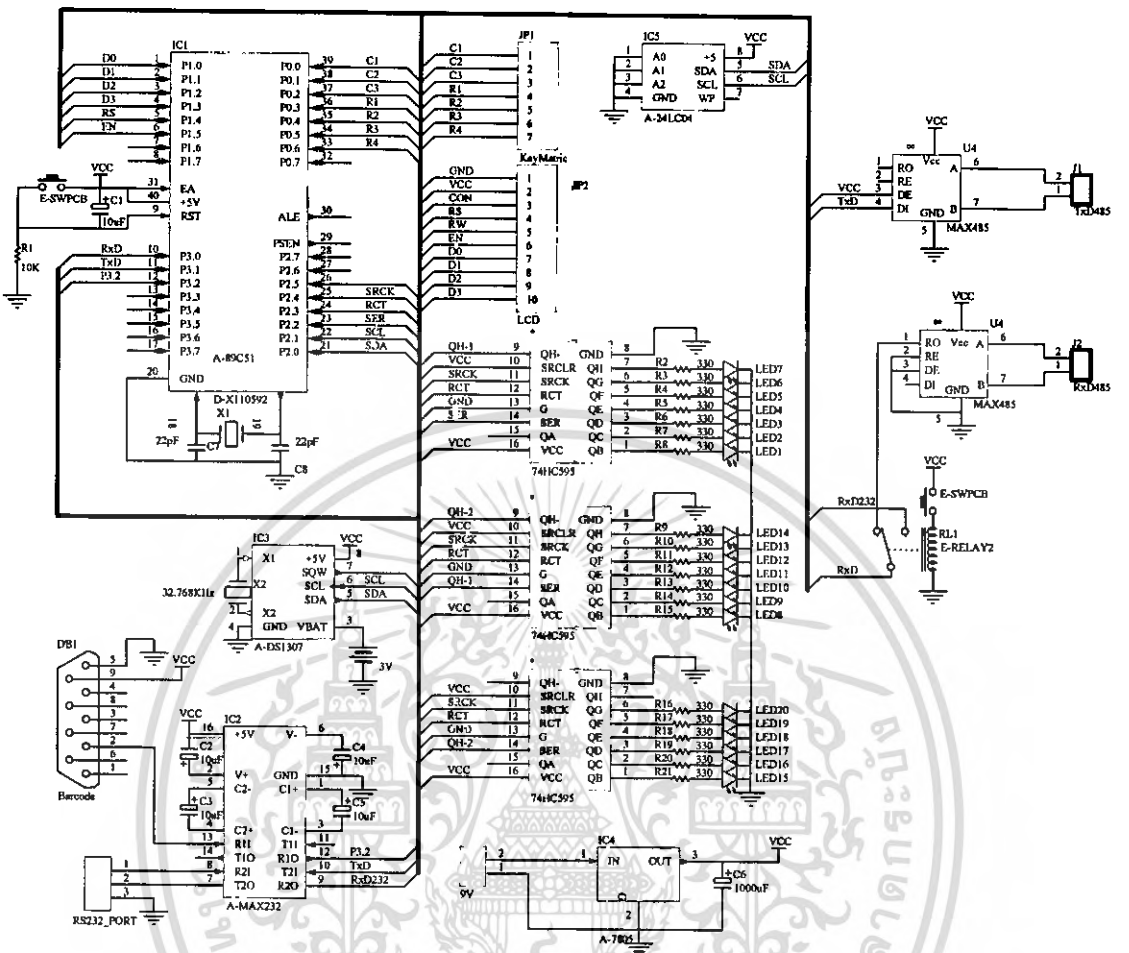
รูปที่ ข.1 เครื่องควบคุมย่อย



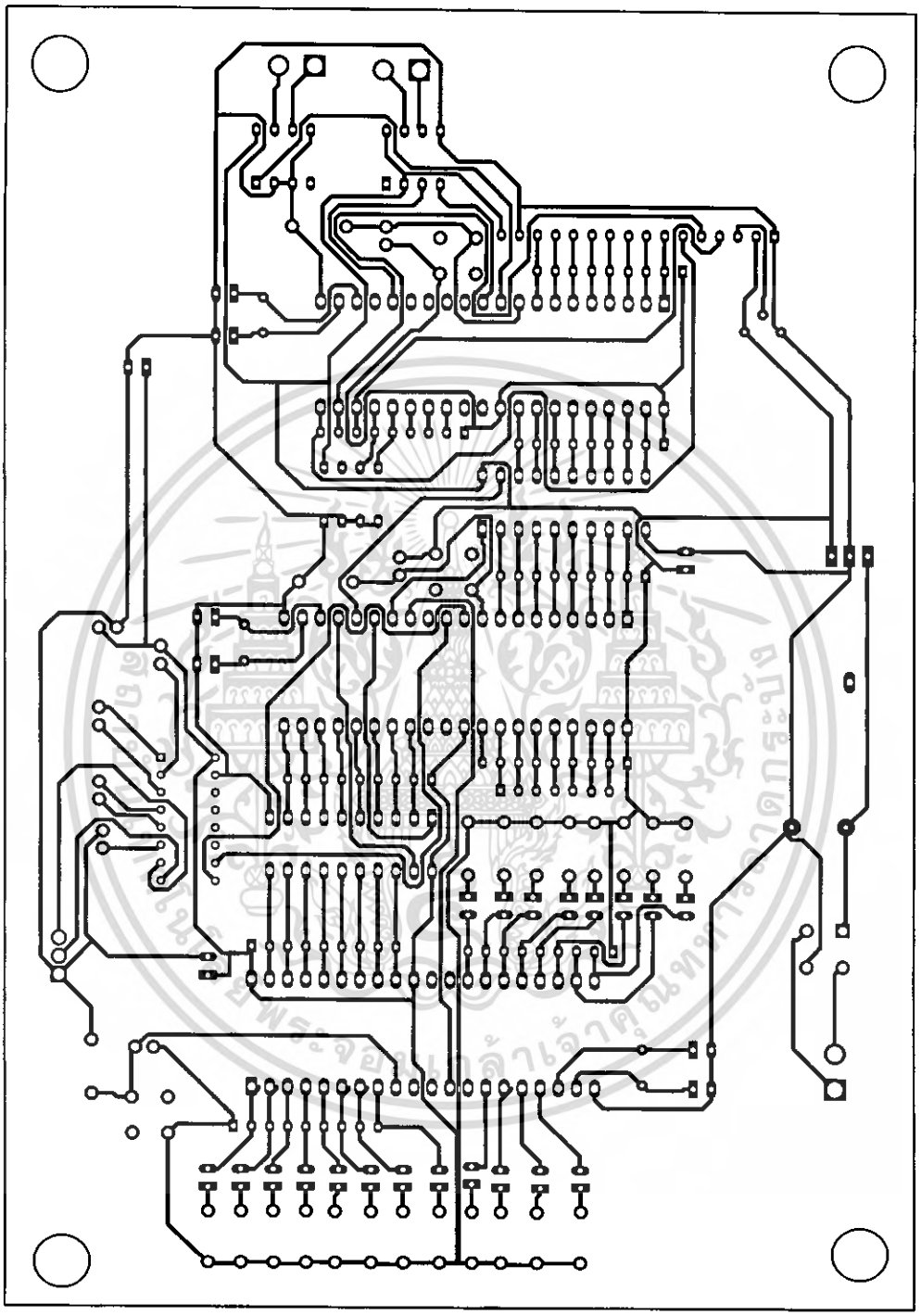
รูปที่ ข.2 แผ่นวงจรพิมพ์เครื่องควบคุมย่อย



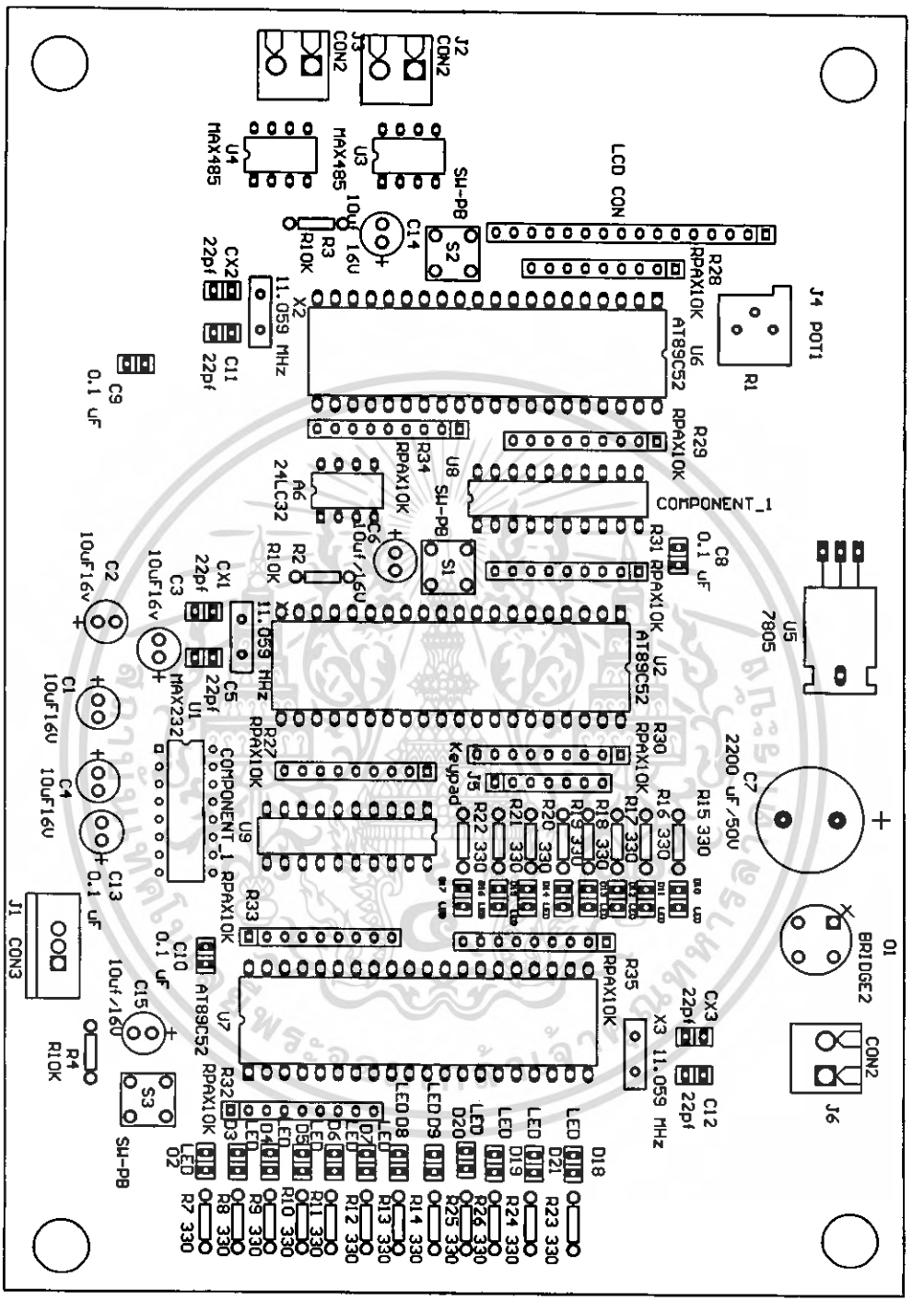
รูปที่ ข.3 ตำแหน่งการวางอุปกรณ์ลงแผ่นวงจรพิมพ์ของเครื่องควบคุมย่อย



รูปที่ ข.4 วงจรเครื่องควบคุมหลัก



รูปที่ ข.5 แผ่นวงจรพิมพ์ของวงจรควบคุมหลัก



รูปที่ ข.6 ตำแหน่งการวางอุปกรณ์ลงแผ่นวงจรพิมพ์ของวงจรควบคุมหลัก

ภาคผนวก ค
รายการอุปกรณ์

ตารางที่ ค.1 รายการอุปกรณ์ของวงจรควบคุมหลัก

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
A6	24LC32	1 ตัว
U8,U9	74hc245	2 ตัว
U1	MAX232	1 ตัว
U3,U4	MAX485	3 ตัว
U2,U7,U6	AT89C52	3 ตัว
U5	7805	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
D1	BRIDGE2	2 ตัว
D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15, D16,D17,D18,D19,D20,D21	LED	20 ตัว
ตัวความต้านทาน		
R1	VR 10 kΩ	1 ตัว
R2,R3,R4	10 kΩ	1 ตัว
R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19, R20,R21,R22,R23,R24,R25,R26	330 kΩ	20 ตัว
R27,R28,R29,R30,R31,R32,R33,R34,R35	RPAX-9PIN 10 kΩ	9 ตัว
ตัวเก็บประจุ		
C1,C2.C3,C4,C6,C15,C16	10 μF	7 ตัว
CX1,CX2,CX3,C5,C11,C12	22 pF	3 ตัว
C8,C9,C10,C13	0.1 μF	6 ตัว
C7	2200 μF	2 ตัว
อุปกรณ์อื่นๆ		
X1,X2,X3	11.0592 MHz.	3 ตัว
J2,J3,J6	CON2	3 ตัว
KEYPAD	MATRIX-SWITCH	1 ตัว
LCD	LCD DISPLAY	1 ตัว
S1,S2,S3	TAC SW SPST	3 ตัว
6 V/1 A	TRANFORMER	1 ตัว
HDL15PTK1YS/RH Connector RS232	Connecter 9 pin	2 ตัว

ตารางที่ ค.2 รายการอุปกรณ์ของวงจรควบคุมย่อย

ชื่ออุปกรณ์	รายละเอียด	จำนวน
วงจรรวม		
U2,U3	AT89C52	2 ตัว
U1	MAX232	1 ตัว
U4,U5	MAX485	2 ตัว
U6	7805	1 ตัว
อุปกรณ์สารกึ่งตัวนำ		
D1	BRIDGE2	1 ตัว
ตัวความต้านทาน		
R1,R2,R3,R4,R5,R6	10 k Ω	6 ตัว
ตัวเก็บประจุ		
C1,C2,C3,C4,C5,C6,C7	10 μF	7 ตัว
C8,C9,C10,C11	33 pF	4 ตัว
C13,C15,C15	0.1 μF	3 ตัว
C12	2200 μF	1 ตัว
อุปกรณ์อื่นๆ		
Y1,Y2	11.0592 MHz.	2 ตัว
J2,J3,J4	CON2	3 ตัว
S1,S2	TAC SW SPST	2 ตัว
6 V/1 A	TRANFORMER	1 ตัว
HDL15PTK1YS/RH - Connector RS232	Connecter 9 pin	2 ตัว

ภาคผนวก ง
รายการละเอียดและคุณสมบัติของอุปกรณ์

DALLAS SEMICONDUCTOR

DS1302
Trickle Charge Timekeeping Chip

www.dalsemi.com

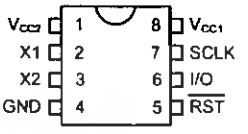
FEATURES

- Real time clock counts seconds, minutes hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100
- 31 x 8 RAM for scratchpad data storage
- Serial I/O for minimum pin count
- 2.0–5.5V full operation
- Uses less than 300 nA at 2.0V
- Single–byte or multiple–byte (burst mode) data transfer for read or write of clock or RAM data
- 8–pin DIP or optional 8–pin SOICs for surface mount
- Simple 3–wire interface
- TTL–compatible (V_{CC} = 5V)
- Optional industrial temperature range –40°C to +85°C
- DS1202 compatible
- Recognized by Underwriters Laboratory

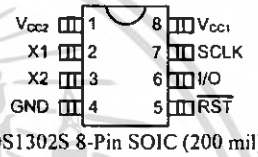
ORDERING INFORMATION

PART #	DESCRIPTION
DS1302	8–Pin DIP
DS1302N	8-Pin DIP (Industrial)
DS1302S	8–Pin SOIC (200 mil)
DS1302SN	8–Pin SOIC (Industrial)
DS1302Z	8–Pin SOIC (150 mil)
DS1302ZN	8–Pin SOIC (Industrial)
DS1302S-16	16-Pin SOIC (300 mil)
DS1302SN-16	16-Pin SOIC (Industrial)

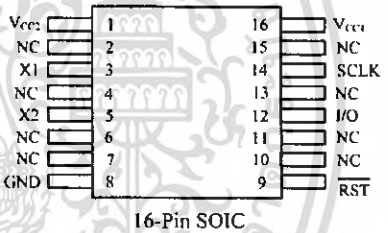
PIN ASSIGNMENT



DS1302
8-Pin DIP (300 mil)



DS1302S 8-Pin SOIC (200 mil)
DS1302Z 8-Pin SOIC (150 mil)



16-Pin SOIC

PIN DESCRIPTION

X1, X2	– 32.768 kHz Crystal Pins
GND	– Ground
$\overline{RST}$	– Reset
I/O	– Data Input/Output
SCLK	– Serial Clock
V_{CC1}, V_{CC2}	– Power Supply Pins

DESCRIPTION

The DS1302 Trickle Charge Timekeeping Chip contains a real time clock/calendar and 31 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real time clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with less than 31 days, including corrections for leap year. The clock operates in either the 24–hour or 12–hour format with an AM/PM indicator.

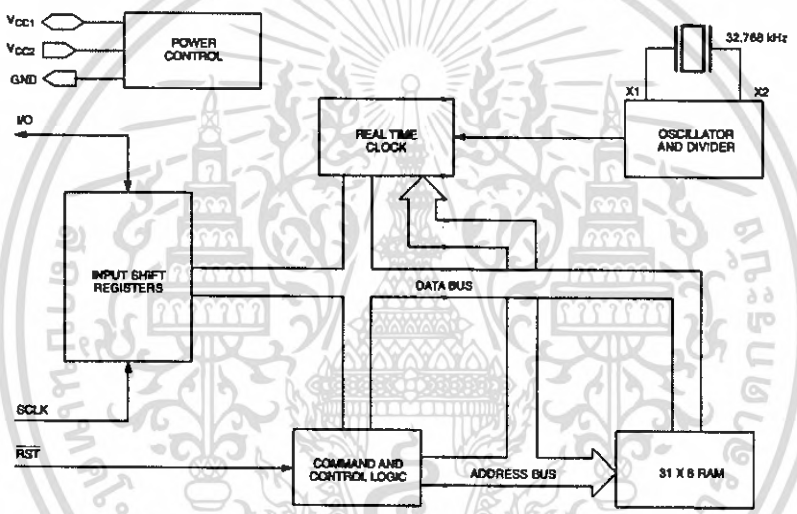
Interfacing the DS1302 with a microprocessor is simplified by using synchronous serial communieation. Only three wires are required to communicate with the clock/RAM: (1) $\overline{RST}$ (Reset), (2) I/O (Data line), and (3) SCLK (Serial clock). Data can be transferred to and from the clock/RAM 1 byte at a time or in a burst of up to 31 bytes. The DS1302 is designed to operate on very low power and retain data and clock information on less than 1 microwatt.

The DS1302 is the successor to the DS1202. In addition to the basic timekeeping functions of the DS1202, the DS1302 has the additional features of dual power pins for primary and back–up power supplies, programmable trickle charger for V_{CC1}, and seven additional bytes of scratchpad memory.

OPERATION

The main elements of the Serial Timekeeper are shown in Figure 1: shift register, control logie, oscillator, real time clock, and RAM.

DS1302 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC1} – V_{CC1} provides low power operation in single supply and battery operated systems as well as low power battery backup. In systems using the trickle charger, the rechargeable energy source is connected to this pin.

V_{CC2} – Vcc2 is the primary power supply pin in a dual supply configuration. V_{CC1} is connected to a backup source to maintain the time and date in the absence of primary power.

The DS1302 will operate from the larger of V_{CC1} or V_{CC2}. When V_{CC2} is greater than V_{CC1} + 0.2V, V_{CC2} will power the DS1302. When V_{CC2} is less than V_{CC1}, V_{CC1} will power the DS1302.

SCLK (Serial Clock Input) – SCLK is used to synchronize data movement on the serial interface.

I/O (Data Input/Output) – The I/O pin is the bi-directional data pin for the 3-wire interface.

$\overline{RST}$ (Reset) – The reset signal must be asserted high during a read or a write.

X1, X2 – Connections for a standard 32.768 kHz quartz crystal. The internal oscillator is designed for operation with a crystal having a specified load capacitance of 6 pF. For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks." The DS1302 can also be driven by an external 32.768 kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

COMMAND BYTE

The command byte is shown in Figure 2. Each data transfer is initiated by a command byte. The MSB (Bit 7) must be a logic 1. If it is 0, writes to the DS1302 will be disabled. Bit 6 specifies clock/calendar data if logic 0 or RAM data if logic 1. Bits 1 through 5 specify the designated registers to be input or output, and the LSB (bit 0) specifies a write operation (input) if logic 0 or read operation (output) if logic 1. The command byte is always input starting with the LSB (bit 0).

ADDRESS/COMMAND BYTE Figure 2

RESET AND CLOCK CONTROL

All data transfers are initiated by driving the $\overline{RST}$ input high. The $\overline{RST}$ input serves two functions. First, $\overline{RST}$ turns on the control logic which allows access to the shift register for the address/command sequence. Second, the $\overline{RST}$ signal provides a method of terminating either single byte or multiple byte data transfer.

A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, data must be valid during the rising edge of the clock and data bits are output on the falling edge of clock. If the $\overline{RST}$ input is low all data transfer terminates and the I/O pin goes to a high impedance state. Data transfer is illustrated in Figure 3. At power–up, $\overline{RST}$ must be a logic 0 until $V_{CC} > 2.0$ volts. Also SCLK must be at a logic 0 when $\overline{RST}$ is driven to a logic 1 state.

DATA INPUT

Following the eight SCLK cycles that input a write command byte, a data byte is input on the rising edge of the next eight SCLK cycles. Additional SCLK cycles are ignored should they inadvertently occur. Data is input starting with bit 0.

DATA OUTPUT

Following the eight SCLK cycles that input a read command byte, a data byte is output on the falling edge of the next eight SCLK cycles. Note that the first data bit to be transmitted occurs on the first falling edge after the last bit of the command byte is written. Additional SCLK cycles retransmit the data bytes should they inadvertently occur so long as $\overline{RST}$ remains high. This operation permits continuous burst mode read capability. Also, the I/O pin is tri–stated upon each rising edge of SCLK. Data is output starting with bit 0.

BURST MODE

Burst mode may be specified for either the clock/calendar or the RAM registers by addressing location 31 decimal (address/command bits 1 through 5 = logic 1). As before, bit 6 specifies clock or RAM and bit 0

specifies read or write. There is no data storage capacity at locations 9 through 31 in the Clock/Calendar Registers or location 31 in the RAM registers. Reads or writes in burst mode start with bit 0 of address 0.

When writing to the clock registers in the burst mode, the first eight registers must be written in order for the data to be transferred. However, when writing to RAM in burst mode it is not necessary to write all 31 bytes for the data to transfer. Each byte that is written to will be transferred to RAM regardless of whether all 31 bytes are written or not.

CLOCK/CALENDAR

The clock/calendar is contained in seven write/read registers as shown in Figure 4. Data contained in the clock/ calendar registers is in binary coded decimal format (BCD).

CLOCK HALT FLAG

Bit 7 of the seconds register is defined as the clock halt flag. When this bit is set to logic 1, the clock oscillator is stopped and the DS1302 is placed into a low-power standby mode with a current drain of less than 100 nanoamps. When this bit is written to logic 0, the clock will start. The initial power on state is not defined.

AM-PM/12-24 MODE

Bit 7 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10-hour bit (20 – 23 hours).

WRITE PROTECT BIT

Bit 7 of the control register is the write-protect bit. The first seven bits (bits 0 – 6) are forced to 0 and will always read a 0 when read. Before any write operation to the clock or RAM, bit 7 must be 0. When high, the write protect bit prevents a write operation to any other register. The initial power on state is not defined. Therefore the WP bit should be cleared before attempting to write to the device.

TRICKLE CHARGE REGISTER

This register controls the trickle charge characteristics of the DS1302. The simplified schematic of Figure 5 shows the basic components of the trickle charger. The trickle charge select (TCS) bits (bits 4 -7) control the selection of the trickle charger. In order to prevent accidental enabling, only a pattern of 1010 will enable the trickle charger. All other patterns will disable the trickle charger. The DS1302 powers up with the trickle charger disabled. The diode select (DS) bits (bits 2 – 3) select whether one diode or two diodes are connected between V_{CC2} and V_{CC1}. If DS is 01, one diode is selected or if DS is 10, two diodes are selected. If DS is 00 or 11, the trickle charger is disabled independently of TCS. The RS bits (bits 0 -1) select the resistor that is connected between V_{CC2} and V_{CC1}. The resistor selected by the resistor select (RS) bits is as follows:

RS Bits	Resistor	Typical Value
00	None	None
01	R1	2 kΩ
10	R2	4 kΩ
11	R3	8 kΩ

If RS is 00, the trickle charger is disabled independently of TCS.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	0°C to 70°C or - 40°C to +85°C for industrial
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds (DIP) See IPC/JEDEC Standard J-STD-020A for Surface Mount Devices

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

(0°C to 70°C or -40°C to +85°C)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Supply Voltage V_{CC1}, V_{CC2}	V_{CC1}, V_{CC2}		2.0		5.5	V	1, 11
Logic 1 Input	V_{IH}		2.0		V_{CC}+0.3	V	1
Logic 0 Input	V_{IL}	V_{CC}=2.0V	-0.3		+0.3	V	1
		V_{CC}=5V	–0.3		+0.8		

*-40°C to +85°C for industrial device.

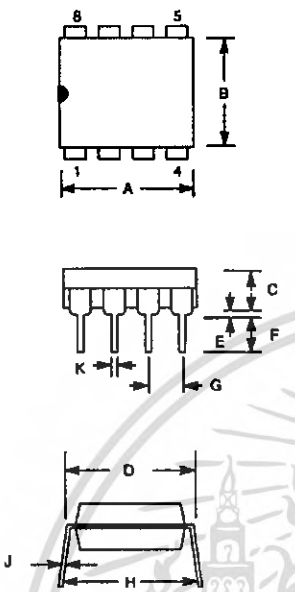
DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C or -40°C to +85°C; V_{CC} = 2.0 to 5.5V*)

PARAMETER	SYMBOL		MIN	TYP	MAX	UNITS	NOTES
Input Leakage	I_{LI}				+500	µA	6
I/O Leakage	I_{LO}				+500	µA	6
Logic 1 Output	V_{OH}	V_{CC}=2.0V	1.6			V	2
		V_{CC}=5V	2.4				
Logic 0 Output	V_{OL}	V_{CC}=2.0V			0.4	V	3
		V_{CC}=5V			0.4		
Active Supply Current	I_{CC1A}	V_{CC1}=2.0V			0.4	mA	5, 12
		V_{CC1}=5V			1.2		
Timekeeping Current	I_{CC1T}	V_{CC1}=2.0V			0.3	µA	4, 12
		V_{CC1}=5V			1		
Standby Current	I_{CC1S}	V_{CC1}=2.0V			100	nA	10, 12, 14
		V_{CC1}=5V			100		
		IND			200		
Active Supply Current	I_{CC2A}	V_{CC2}=2.0V			0.425	mA	5, 13
		V_{CC2}=5V			1.28		
Timekeeping Current	I_{CC2T}	V_{CC2}=2.0V			25.3	µA	4, 13
		V_{CC2}=5V			81		
Standby Current	I_{CC2S}	V_{CC2}=2.0V			25	µA	10, 13
		V_{CC2}=5V			80		
Trickle Charge Resistors	R1 R2 R3			2 4 8		kΩ kΩ kΩ	
Trickle Charge Diode Voltage Drop	V_{TD}			0.7		V	

*Unless otherwise noted.

DS1302 SERIAL TIMEKEEPER 8–PIN DIP (300-MIL)



PKG	8-PIN	
DIM	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

Features

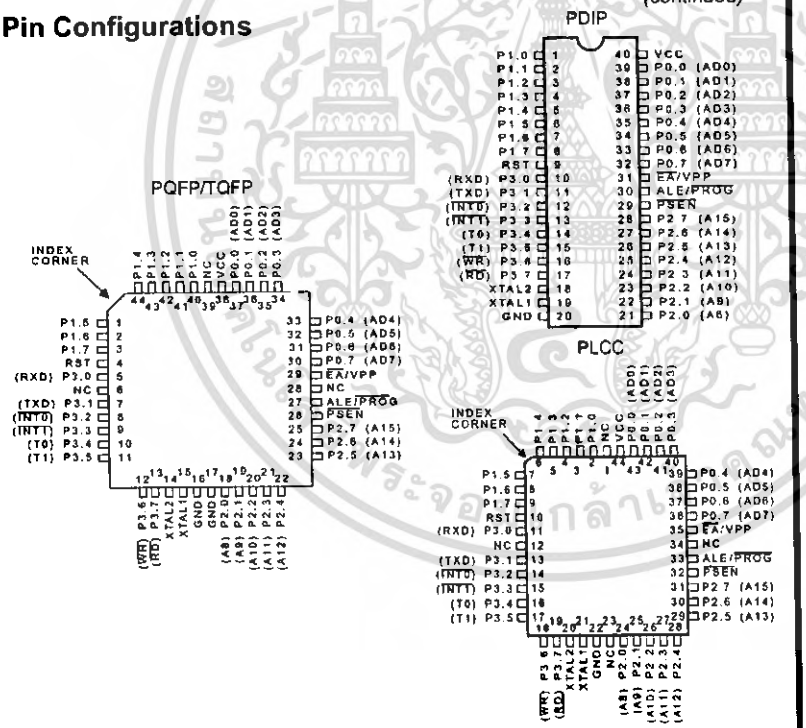
- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt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



8-Bit Microcontroller with 4K Bytes Flash

AT89C51

0265F-A–12/97

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

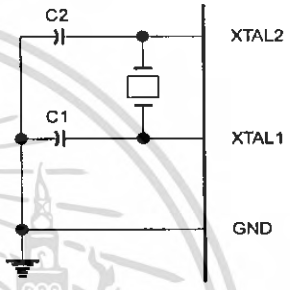
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected 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.

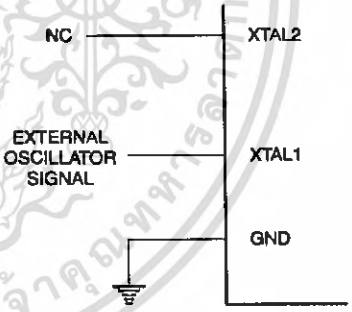
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port 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

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 eny 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.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

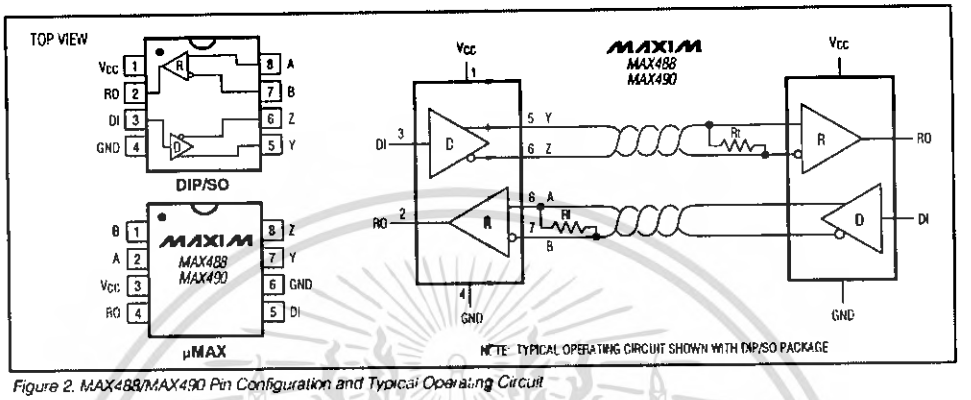


Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

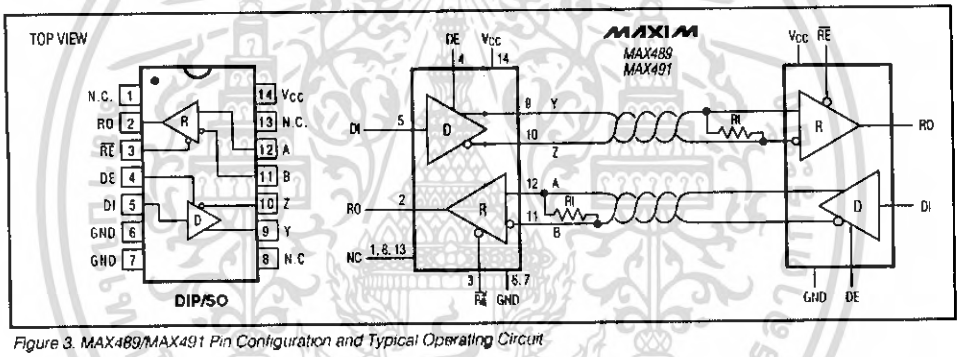


Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

Applications Information

The MAX481/MAX483/MAX485/MAX487–MAX491 and MAX1487 are low-power transceivers for RS-485 and RS-422 communications. The MAX481, MAX485, MAX490, MAX491, and MAX1487 can transmit and receive at data rates up to 2.5Mbps, while the MAX483, MAX487, MAX488, and MAX489 are specified for data rates up to 250kbps. The MAX488–MAX491 are full-duplex transceivers while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are half-duplex. In addition, Driver Enable (DE) and Receiver Enable ($\overline{RE}$) pins are included on the MAX481, MAX483, MAX485, MAX487, MAX489, MAX491, and MAX1487. When disabled, the driver and receiver outputs are high impedance.

MAX487/MAX1487: 128 Transceivers on the Bus

The 48kΩ, ¼-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12kΩ input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488–MAX491 have standard 12kΩ Receiver Input impedance.

Test Circuits

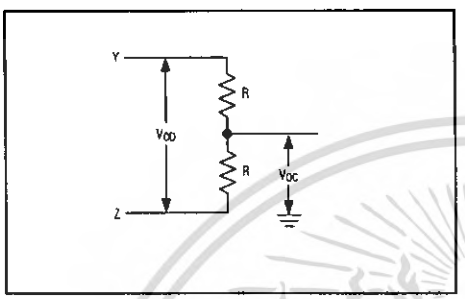


Figure 4. Driver DC Test Load

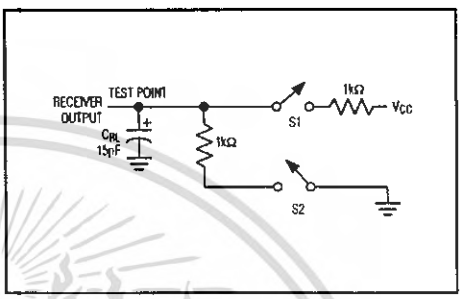


Figure 5. Receiver Timing Test Load

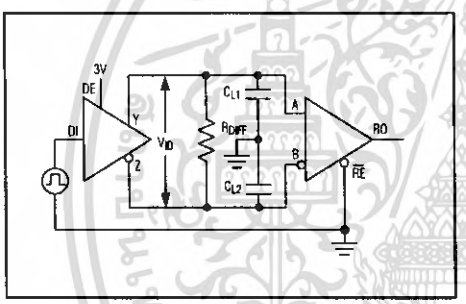


Figure 6. Driver/Receiver Timing Test Circuit

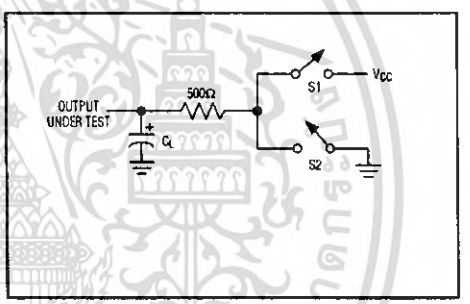


Figure 7. Driver Timing Test Load

MAX483/MAX487/MAX488/MAX489: Reduced EMI and Reflections

The MAX483 and MAX487–MAX489 are slew-rate limited, minimizing EMI and reducing reflections caused by improperly terminated cables. Figure 12 shows the driver output waveform and its Fourier analysis of a 150kHz signal transmitted by a MAX481, MAX485, MAX490, MAX491, or MAX1487. High-frequency harmonics with large amplitudes are evident. Figure 13 shows the same information displayed for a MAX483, MAX487, MAX488, or MAX489 transmitting under the same conditions. Figure 13's high-frequency harmonics have much lower amplitudes, and the potential for EMI is significantly reduced.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Line Length vs. Data Rate

The RS-485/RS-422 standard covers line lengths up to 4000 feet. For line lengths greater than 4000 feet, see Figure 23.

Figures 19 and 20 show the system differential voltage for the parts driving 4000 feet of 26AWG twisted-pair wire at 110kHz into 120Ω loads.

Typical Applications

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 transceivers are designed for bidirectional data communications on multipoint bus transmission lines. Figures 21 and 22 show typical network applications circuits. These parts can also be used as line repeaters, with cable lengths longer than 4000 feet, as shown in Figure 23.

To minimize reflections, the line should be terminated at both ends in its characteristic impedance, and stub lengths off the main line should be kept as short as possible. The slew-rate-limited MAX483 and MAX487–MAX489 are more tolerant of imperfect termination.

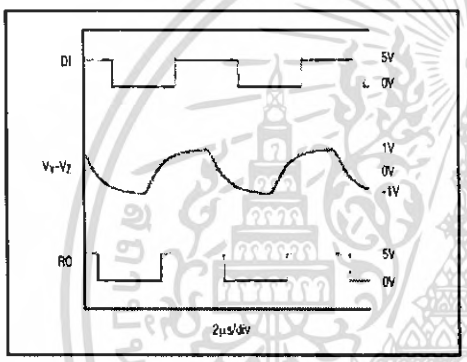


Figure 19. MAX481/MAX485/MAX490/MAX491/MAX1487 System Differential Voltage at 110kHz Driving 4000ft of Cable

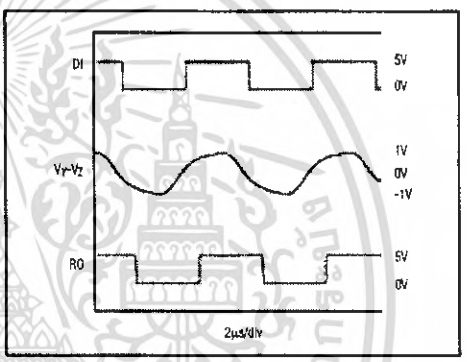


Figure 20. MAX483, MAX487–MAX489 System Differential Voltage at 110kHz Driving 4000ft of Cable

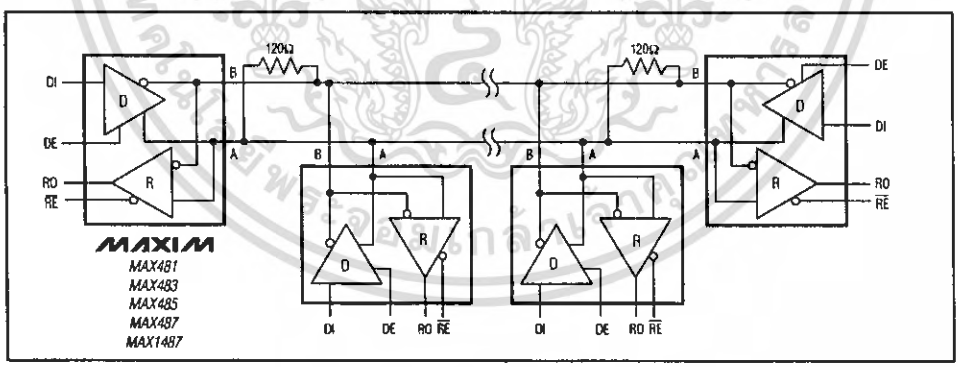


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX481CPA	0°C to +70°C	8 Plastic DIP
MAX481CSA	0°C to +70°C	8 SO
MAX481CUA	0°C to +70°C	8 µMAX
MAX481C/D	0°C to +70°C	Dice*
MAX481EPA	-40°C to +85°C	8 Plastic DIP
MAX481ESA	-40°C to +85°C	8 SO
MAX481MJA	-55°C to +125°C	8 CERDIP
MAX483CPA	0°C to +70°C	8 Plastic DIP
MAX483CSA	0°C to +70°C	8 SO
MAX483CUA	0°C to +70°C	8 µMAX
MAX483C/D	0°C to +70°C	Dice*
MAX483EPA	-40°C to +85°C	8 Plastic DIP
MAX483ESA	-40°C to +85°C	8 SO
MAX483MJA	-55°C to +125°C	8 CERDIP
MAX485CPA	0°C to +70°C	8 Plastic DIP
MAX485CSA	0°C to +70°C	8 SO
MAX485CUA	0°C to +70°C	8 µMAX
MAX485C/D	0°C to +70°C	Dice*
MAX485EPA	-40°C to +85°C	8 Plastic DIP
MAX485ESA	-40°C to +85°C	8 SO
MAX485MJA	-55°C to +125°C	8 CERDIP
MAX487CPA	0°C to +70°C	8 Plastic DIP
MAX487CSA	0°C to +70°C	8 SO
MAX487CUA	0°C to +70°C	8 µMAX
MAX487C/D	0°C to +70°C	Dice*
MAX487EPA	-40°C to +85°C	8 Plastic DIP
MAX487ESA	-40°C to +85°C	8 SO
MAX487MJA	-55°C to +125°C	8 CERDIP
MAX488CPA	0°C to +70°C	8 Plastic DIP
MAX488CSA	0°C to +70°C	8 SO
MAX488CUA	0°C to +70°C	8 µMAX
MAX488C/D	0°C to +70°C	Dice*
MAX488EPA	-40°C to +85°C	8 Plastic DIP
MAX488ESA	-40°C to +85°C	8 SO
MAX488MJA	-55°C to +125°C	8 CERDIP
MAX489CPD	0°C to +70°C	14 Plastic DIP
MAX489CSD	0°C to +70°C	14 SO
MAX489C/D	0°C to +70°C	Dice*
MAX489EPD	-40°C to +85°C	14 Plastic DIP
MAX489ESD	-40°C to +85°C	14 SO
MAX489MJD	-55°C to +125°C	14 CERDIP

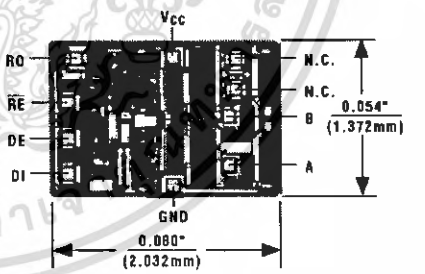
Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX490CPA	0°C to +70°C	8 Plastic DIP
MAX490CSA	0°C to +70°C	8 SO
MAX490CUA	0°C to +70°C	8 µMAX
MAX490C/D	0°C to +70°C	Dice*
MAX490EPA	-40°C to +85°C	8 Plastic DIP
MAX490ESA	-40°C to +85°C	8 SO
MAX490MJA	-55°C to +125°C	8 CERDIP
MAX491CPD	0°C to +70°C	14 Plastic DIP
MAX491CSD	0°C to +70°C	14 SO
MAX491C/D	0°C to +70°C	Dice*
MAX491EPD	-40°C to +85°C	14 Plastic DIP
MAX491ESD	-40°C to +85°C	14 SO
MAX491MJD	-55°C to +125°C	14 CERDIP
MAX1487CPA	0°C to +70°C	8 Plastic DIP
MAX1487CSA	0°C to +70°C	8 SO
MAX1487CUA	0°C to +70°C	8 µMAX
MAX1487C/D	0°C to +70°C	Dice*
MAX1487EPA	-40°C to +85°C	8 Plastic DIP
MAX1487ESA	-40°C to +85°C	8 SO
MAX1487MJA	-55°C to +125°C	8 CERDIP

* *Contact factory for dice specifications.*

Chip Topographies

MAX481/MAX483/MAX485/MAX487/MAX1487

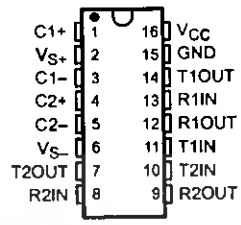


MAX232, MAX232I
DUAL EIA-232 DRIVER/RECEIVER

SLLS047G – FEBRUARY 1989 – REVISED AUGUST 1998

- Operates With Single 5-V Power Supply
- LinBiCMOS™ Process Technology
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Designed to be Interchangeable With Maxim MAX232
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers
- ESD Protection Exceeds 2000 V Per MIL-STD-883, Method 3015
- Package Options Include Plastic Small-Outline (D, DW) Packages and Standard Plastic (N) DIPs

D, DW, OR N PACKAGE
(TOP VIEW)

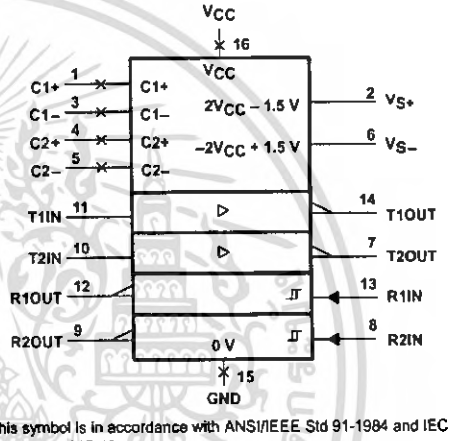


description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

The MAX232 is characterized for operation from 0°C to 70°C. The MAX232I is characterized for operation from –40°C to 85°C.

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES		
	SMALL OUTLINE (D)	SMALL OUTLINE (DW)	PLASTIC DIP (N)
0°C to 70°C	MAX232D‡	MAX232DW‡	MAX232N
–40°C to 85°C	MAX232ID‡	MAX232IDW‡	MAX232IN

‡ This device is available taped and reeled by adding an R to the part number (i.e., MAX232DR).

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) ..... −0.3 V to 6 V
Positive output supply voltage range, V_{S+} ..... V_{CC} − 0.3 V to 15 V
Negative output supply voltage range, V_{S-} ..... −0.3 V to −15 V
Input voltage range, V_I: Driver ..... −0.3 V to V_{CC} + 0.3 V
Receiver ..... ±30 V
Output voltage range, V_O: T1OUT, T2OUT ..... V_{S-} −0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT ..... −0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT ..... Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 113°C/W
DW package ..... 105°C/W
N package ..... 78°C/W
Storage temperature range, T_{stg} ..... −65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

recommended operating conditions

		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.5	5	5.5	V
High-level input voltage, V_{IH} (T1IN,T2IN)		2			V
Low-level input voltage, V_{IL} (T1IN, T2IN)				0.8	V
Receiver input voltage, R1IN, R2IN				±30	V
Operating free-air temperature, T_A	MAX232	0		70	°C
	MAX232I	−40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
		R1OUT, R2OUT	I_{OH} = –1 mA		3.5			
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
		R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	
V_{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
V_{IT-}	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
V_{hys}	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
r_i	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ
r_o	Output resistance	T1OUT, T2OUT	V_{S+} = V_{S-} = 0,	V_O = ± 2 V	300			Ω
I_{OS}§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
I_{IS}	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA
I_{CC}	Supply current		V_{CC} = 5.5 V, T_A = 25°C	All outputs open,		8	10	mA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	See Figure 1		500		ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	See Figure 1		500		ns
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(tr)	Driver transition region slew rate	See Figure 3		3		V/µs

APPLICATION INFORMATION

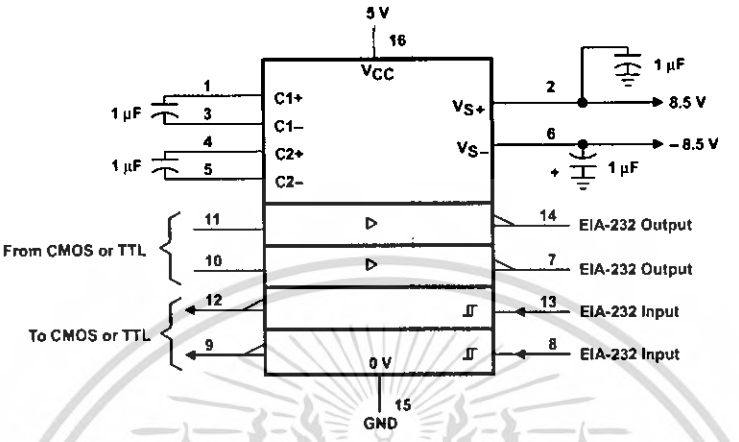
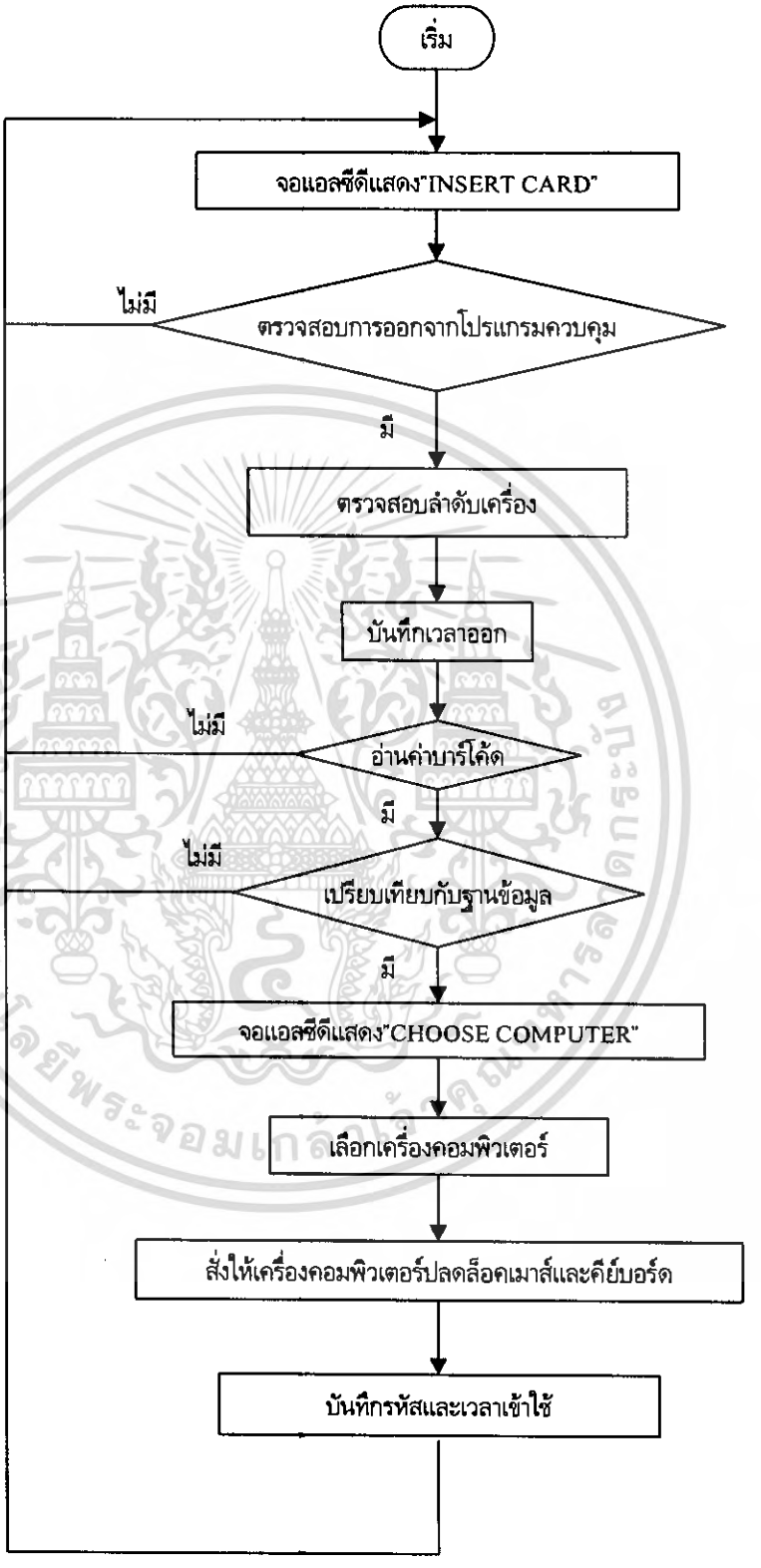


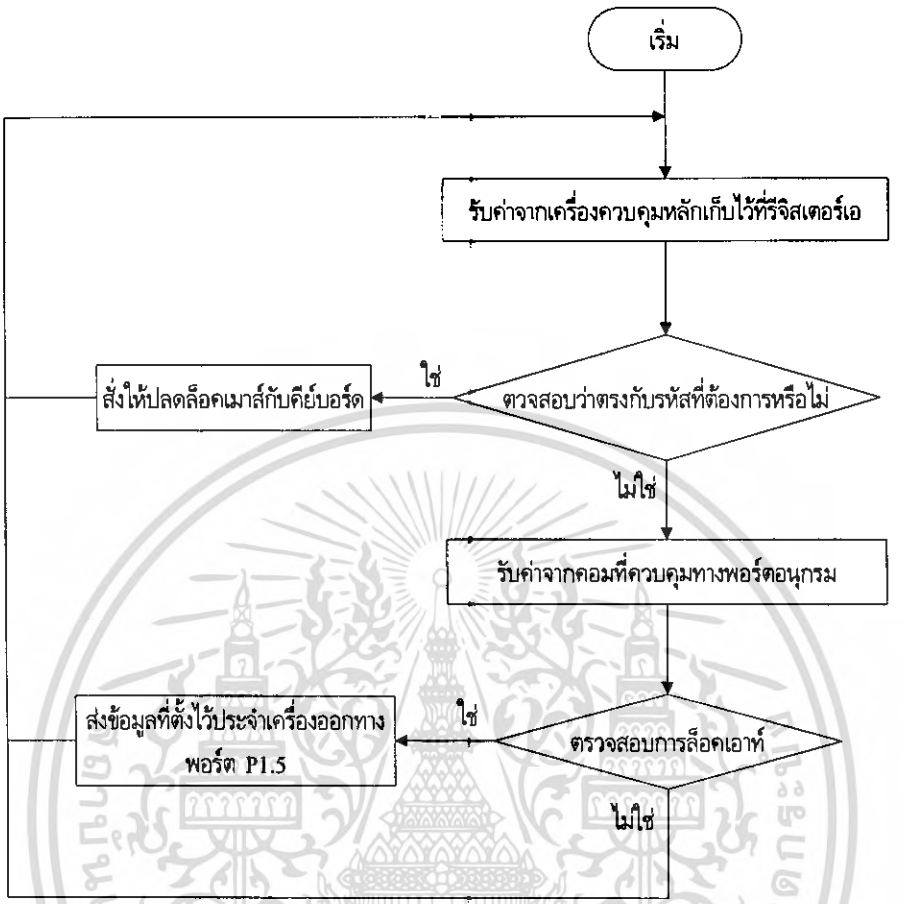
Figure 4. Typical Operating Circuit

ภาคผนวก จ

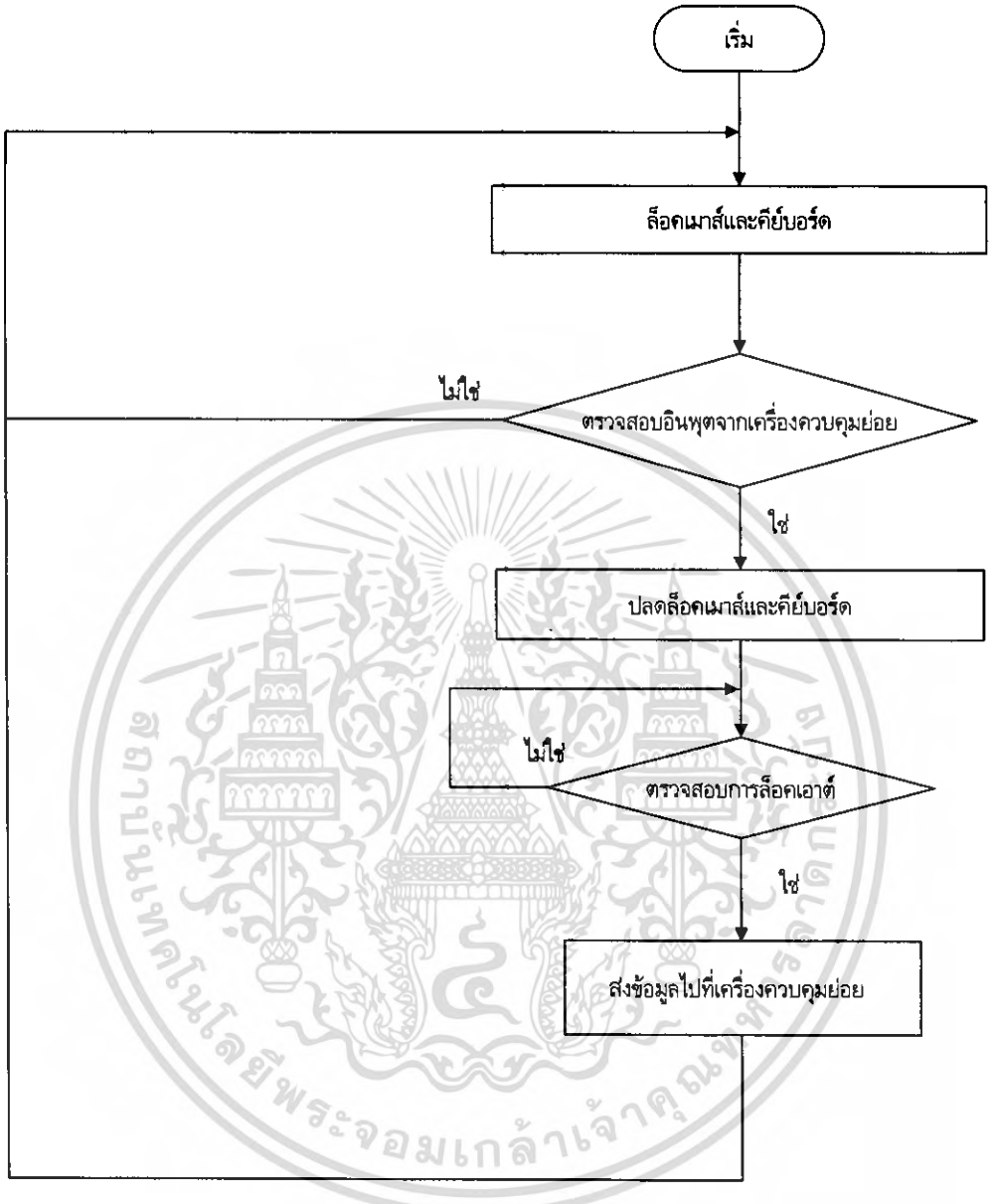
ผังงาน



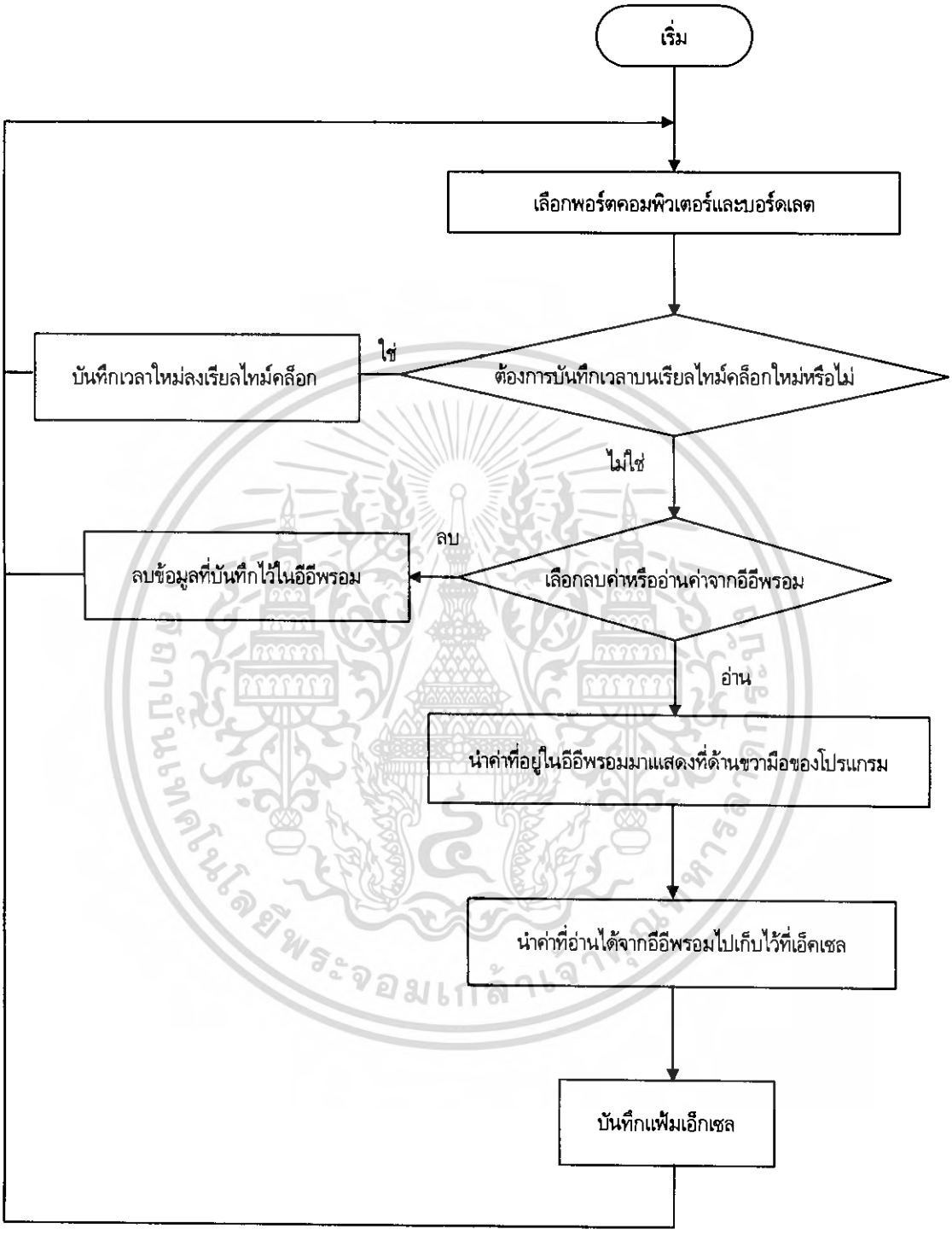
รูปที่ จ.1 ผังการทำงานของโปรแกรมควบคุมหลัก



รูปที่ จ.2 ผังการทำงานของโปรแกรมควบคุมเครื่องควบคุมย่อย



รูปที่ จ.3 ผังการทำงานของโปรแกรมควบคุมเครื่องคอมพิวเตอร์



รูปที่ จ.4 ผังการทำงานของโปรแกรมโหลดค่าที่บันทึกจากเครื่องควบคุมหลัก

ภาคผนวก ฉ
รหัสต้นฉบับของโปรแกรม

โปรแกรมเครื่องควบคุมหลัก

```
/*****************************************************************/
/***************************** EEPROM MAP ************************/
/*              Addr                        Use For
//             0x0001-0x0004          Password to Register Mode
//             0x0005-0x03FF                ID CARD
               0x0400-0xFFFF                Log Data
*/
#include <REG51F.H>
#include <string.h>
#include <stdio.h>
#include "RTC.H"
#include "lib_eeprom.h"
// LCD bit define
Sbit rs_LCD=P1^4;// select command (0) or data(1)
Sbit en_LCD=P1^5;
Sbit D3=P1^3;
Sbit D2=P1^2;
Sbit D1=P1^1;
Sbit D0=P1^0;

// Ket Pad
Sbit C1=P0^0;
Sbit C2=P0^1;
Sbit C3=P0^2;

Sbit R1=P0^3;
Sbit R2=P0^4;
Sbit R3=P0^5;
Sbit R4=P0^6;
```

```
// Barcode receive pin
Sbit BarcodeRx=P3^2;

#define TimeWait 400
// about Cursor LCD
#define ON              0x02
#define OFF             0x00
#define BLINK   0x01
// boundrate 4800
#define D4800 {dusec(51);}
#define D9600 {dusec(26);}

#define DisTx   TxEn=1;
#define EnTx    TxEn=0;
Unsigned char BarcodeBuffer[10],BarcodeLen=0,RX3Len=0;
Unsigned char RecieveBuffer[6];
Unsigned char PwdBuff[4]={'1','2','3','4'};

Unsigned int KeyIn,Num,Num2,LoadIndex;

Unsigned char StrBuff[30];
Unsigned char SelBuff[3];
Unsigned char ComBuff[20];
Unsigned char LogOnBuff[20];
Xdata unsigned char IDBuff[128];

Int Addr[5];
Bit TimeBlink,BarcodeReady,TxBusy=0,RxReady,RX3Ready;
Bit Start,Stop,StartGLED,BarcodeWait,WaitAck;
Bit WaitData,DataReady;
```

```
Bit WaitR1,WaitR2,WaitR3,WaitR4;
Int TimeCount;
Unsigned char Index,PageCount,DataIndex,iBuff;
Char Mode,PwdCount;
Bit InputPwd,PwdCorrect,StartLoadLogData;

Bit Add,Del,Select,ChangePwd;

Extern bit L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16,L17,L18,L19,L20;
Extern void ClrSHR(void);
Extern void UpOut(void);
Void dusec(unsigned char time)
{
  while(time--);
}
Void dmsec(unsigned int time)
{
        while(time--)
        {
                dusec(250);
                dusec(250);
        }
}
//////////////////////////////////////////////////////////////
////////////////////// LCD ///////////////////////////////////
//////////////////////////////////////////////////////////////
Void LCD_Pulse(void)
{
en_LCD=0;
dusec(20);
en_LCD=1;
```

```
}
Void LCD_Command_write(unsigned char ch)
{
en_LCD=1;
rs_LCD=0;
if((ch&128)==128)D3=1; else D3=0;
if((ch&64)==64)D2=1;    else D2=0;
if((ch&32)==32)D1=1;    else D1=0;
if((ch&16)==16)D0=1;    else D0=0;
LCD_Pulse();
if((ch&8)==8)D3=1;      else D3=0;
if((ch&4)==4)D2=1;      else D2=0;
if((ch&2)==2)D1=1;      else D1=0;
if((ch&1)==1)D0=1;      else D0=0;
LCD_Pulse();
}
Void LCD_Data_write(unsigned char ch)
{
en_LCD=1;
rs_LCD=1;
if((ch&128)==128)D3=1; else D3=0;
if((ch&64)==64)D2=1;    else D2=0;
if((ch&32)==32)D1=1;    else D1=0;
if((ch&16)==16)D0=1;    else D0=0;
LCD_Pulse();
if((ch&8)==8)D3=1;      else D3=0;
if((ch&4)==4)D2=1;      else D2=0;
if((ch&2)==2)D1=1;      else D1=0;
if((ch&1)==1)D0=1;      else D0=0;
LCD_Pulse();
```

```
}
Void ClearLCD(void)
{
LCD_Command_write(0x01);
dmsec(50);
}
Void CursorLCD (unsigned char attribute)
{
        If (attribute == BLINK)
                Attribute |= ON;

LCD_Command_write(0x0C | attribute);

}


Void putc(char ch, unsigned char index)
{
        If (index > 0x1F)
                return;
        if (index > 0x0F)
                index += 0x30;


        LCD_Command_write(0x80 | index);// set Cursor
        LCD_Data_write(ch);
}
Void PrintLCD(char *str, unsigned char index)
{
Unsigned char i;
        for (I = 0; I < strlen(str); i++)
{
```

```
                putc(str[i], I + index);
}
}
Void InitializeLCD(void)
{
dmsec(200);
LCD_Command_write(0x33);
        dmsec(90);
LCD_Command_write(0x32);
        dmsec(50);
LCD_Command_write(0x28);
        dmsec(50);
LCD_Command_write(0x08);
        dmsec(50);
LCD_Command_write(0x0C);
        dmsec(50);
        CursorLCD(BLINK);//
        ClearLCD();
        dmsec(50);
LCD_Command_write(0x06);

        CursorLCD(OFF);
}
Void initializeRS232(void)// initial at 9600 bps 8 bit none parity and 1 stop bit
{
        TxBusy = 0;
        RCAP2H = 0xFF;
        RCAP2L = 0xB8;
SCON = 0x40;
        T2CON = 0x30;
        TR2 = 1;
```

```
        RxReady=0;
        REN=1;
}
Void SendByte(unsigned char sendout)
{
        SBUF = sendout;
        TxBusy = 1;
        while(TxBusy);
}
//************************** Barcode receive Function ******************************
//-------__S__|__0__|__1__|__2__|__3__|__4__|__5__|__6__|__7__|__E__|
Void BarcodeRecieve(void) interrupt 0
{
unsigned char Buff,i,j;
unsigned int TimeOut;
i=0;
EX0=0;//disable INT0
while(1)
{
TimeOut=20000;
        while(BarcodeRx)// WAIT START BIT
                {
                        if(TimeOut==0)
                                break;
                        else
                                 TimeOut--;
}
        dusec(12);
        Buff=0;
        if(TimeOut==0&&i>0)// IF no DATA Timeout will decrease to Zero
        {
```

```
BarcodeBuffer[i-1]='\0';
        break;
        }
        Else if(TimeOut)// timeout not equal 0
        {
for(j=0;j<8;j++)
                        {dusec(29);Buff=Buff>>1; CY=0;if(BarcodeRx) Buff|=0x80;}
                dusec(29);
                while(BarcodeRx==0);// stop bit
        BarcodeBuffer[i]=Buff;
        i++;
        }
}
BarcodeLen=i-1;
BarcodeReady=1;
EX0=1;//Enable INT0
}

////// Serial RS485/RS232
// Receive Serial Data And Store In "RecieveBuffer" Parameter
// XXXXXXXX$==> XXXXXXXXX
Void ConnectPC(void) interrupt 4 // serial interrupt
{
        If (TI)
        {
                TxBusy = 0;
                TI = 0;
        }
        if(RI)
        {
                if(!Start&&SBUF==':')
```

```
                {
                        GLED=0;
                        Index=0;
                Start=1;
                        Stop=0;
                }
                Else if(Start&&SBUF=='$')
                {
                        GLED=1;
                        Stop=1;
                        RecieveBuffer[Index]=0;
                }
                Else if(Start&&!Stop)
        {
                        RecieveBuffer[Index]=SBUF;
                        Index++;
                        if(Index>100)
                                Start=0;
                }
                RI=0;
        }
}

//****************************************
// ********** Get Press Key Pad *********
Unsigned char GetKey(void)
{
        C1=0;C2=1;C3=1;
        if(R1==0) {while(R1==0); return '1';}
        else if(R2==0) {while(R2==0); return '4';}
        else if(R3==0) {while(R3==0); return '7';}
```

```
        else if(R4==0) {while(R4==0); return '*';}

        C1=1;C2=0;C3=1;
        if(R1==0) {while(R1==0); return '2';}
        else if(R2==0) {while(R2==0); return '5';}
        else if(R3==0) {while(R3==0); return '8';}
        else if(R4==0) {while(R4==0); return '0';}

        C1=1;C2=1;C3=0;
        if(R1==0) {while(R1==0); return '3';}
        else if(R2==0) {while(R2==0); return '6';}
        else if(R3==0) {while(R3==0);  return '9';}
        else if(R4==0) {while(R4==0); return '#';}
return 0;
}
Void LED(unsigned char CH,char ON_OFF)
{
        switch(CH)
        {
                Case 1:
                        L1=(bit)ON_OFF;
                break;
                case 2:
                        L2=(bit)ON_OFF;
                break;
                case 3:
                        L3=(bit)ON_OFF;
                break;
                case 4:
                        L4=(bit)ON_OFF;
                break;
```

```
case 5:
        L5=(bit)ON_OFF;
break;
case 6:
        L6=(bit)ON_OFF;
break;
case 7:
        L7=(bit)ON_OFF;
break;
case 8:
        L8=(bit)ON_OFF;
break;
case 9:
        L9=(bit)ON_OFF;
break;
case 10:
        L10=(bit)ON_OFF;
break;
case 11:
        L11=(bit)ON_OFF;
break;
case 12:
        L12=(bit)ON_OFF;
break;
case 13:
        L13=(bit)ON_OFF;
break;
case 14:
        L14=(bit)ON_OFF;
break;
case 15:
```

```
                    L15=(bit)ON_OFF;
                break;
                case 16:
                    L16=(bit)ON_OFF;
                break;
                case 17:
                    L17=(bit)ON_OFF;
                break;
                case 18:
                    L18=(bit)ON_OFF;
                break;
                case 19:
                    L19=(bit)ON_OFF;
                break;
                case 20:
                    L20=(bit)ON_OFF;
                break;
        }
        UpOut();// Send LED Output
}
/*****************/
// EEPROM ZONE


// PassWord Store
Void GetPwd(void)
{
        PwdBuff[0]=ReadEEPROM(0x01);
        PwdBuff[1]=ReadEEPROM(0x02);
        PwdBuff[2]=ReadEEPROM(0x03);
        PwdBuff[3]=ReadEEPROM(0x04);
```

```
}
Void SetPwd(void)
{
        WriteEEPROM(0x01,PwdBuff[0]);
        WriteEEPROM(0x02,PwdBuff[1]);
        WriteEEPROM(0x03,PwdBuff[2]);
        WriteEEPROM(0x04,PwdBuff[3]);
}
// Register Card Store
Unsigned char CheckID(char *IdCard)
{
Unsigned char ii,Buff;
Unsigned int Addr;
Addr=5;
for(ii=0;ii<127;ii++)
{
Buff=ReadEEPROM(Addr);
//putc(Buff,0);
if((Buff!='X') &&(Buff==IdCard[0]))
{
Buff=ReadEEPROM(Addr+1);
putc(Buff,1);
if(Buff==IdCard[1])
{
Buff=ReadEEPROM(Addr+2);
//putc(Buff,2);
if(Buff==IdCard[2])
{
Buff=ReadEEPROM(Addr+3);
//putc(Buff,3);
if(Buff==IdCard[3])
```

```
{
Buff=ReadEEPROM(Addr+4);
//putc(Buff,4);
if(Buff==IdCard[4])
{
Buff=ReadEEPROM(Addr+5);
//putc(Buff,5);
if(Buff==IdCard[5])
{
Buff=ReadEEPROM(Addr+6);
//putc(Buff,6);
if(Buff==IdCard[6])
{
Buff=ReadEEPROM(Addr+7);
//putc(Buff,7);
if(Buff==IdCard[7])
return ii+1;
}
}
}
}
}
}
Addr+=8;
}
Return 0;
}
Void AddNew(char *IdCard)
{
        unsigned char ii,Buff,jj;
```

```
        unsigned int Addr;
        Addr=5;
        for(ii=0;ii<127;ii++)
        {
                Buff=ReadEEPROM(Addr);
                if(Buff=='X')
                {
                        for(jj=0;jj<8;jj++)
                                WriteEEPROM(Addr+jj,IdCard[jj]);
                                break;
                }
                Addr+=8;
        }

}
Void Del ID(char *IdCard)
{
Unsigned char ii,Buff,jj;
Unsigned int Addr;
Addr=5;
for(ii=0;ii<127;ii++)
{
Buff=ReadEEPROM(Addr);
if((Buff!='X') &&(Buff==IdCard[0]))
{
Buff=ReadEEPROM(Addr+1);
if(Buff==IdCard[1])
{
Buff=ReadEEPROM(Addr+2);
if(Buff==IdCard[2])
{
```

```
Buff=ReadEEPROM(Addr+3); if(Buff==IdCard[3])
{
Buff=ReadEEPROM(Addr+4);
if(Buff==IdCard[4])
{
Buff=ReadEEPROM(Addr+5);
if(Buff==IdCard[5])
{
Buff=ReadEEPROM(Addr+6); if(Buff==IdCard[6])
{
Buff=ReadEEPROM(Addr+7);
if(Buff==IdCard[7])
{
WriteEEPROM(Addr,'X');
break;
}
}
}
}
}
}
}
}
Addr+=8;
}

}
Void EraseLogData(void)
{
        Unsigned int Addr,ii;
        Addr=0x0400;
```

```
        for(ii=0;ii<1840;ii++)
        {
                WriteEEPROM(Addr,'X');
                WriteEEPROM(Addr+8,'X');
                Addr+=35;
                //      putc(Addr&0xFF,31);
        }
}
Void EraseID(unsigned char Addr)
{
        WriteEEPROM((Addr*8)+5,'X');
}
void FormatRegisterID(void)
{
        Unsigned char ii;
        for(ii=0;ii<127;ii++)
        {
                EraseID(ii);
                putc(ii,31);
        }
}
Void SendLogData(unsigned int iIndex)
{
        Unsigned int Addr;
        Unsigned int ii;
        SendByte(':');
        Addr=0x400+(iIndex*35);
                if(ReadEEPROM(Addr)=='X')
                {
                        SendByte('E');
```

```
                    SendByte('N');
                    SendByte('D');
                    StartLoadLogData=0;// Stop Load Log Data
                }
                else
                {
                    for(ii=0;ii<35;ii++)
                    {
                    SendByte(ReadEEPROM(Addr+ii));
                    }
                }
        SendByte('$');
        dmsec(10);
}
//////////////////////////////
Void SaveLogOn(void)
{
        Unsigned int Addr,ii;
        Unsigned char rBuff;
        Addr=0x400;

        for(ii=0;ii<1840;ii++)
        {
                rBuff=ReadEEPROM(Addr);
//              putc(rBuff,15);
//              dmsec(600);
                if(rBuff=='X')break;
                Addr+=35;
        }
        sprintf(StrBuff,"%d",(int)Addr);
        PrintLCD(StrBuff,0);
```

```
        dmsec(500);

        //Addr=0x400;
        for(ii=0;ii<8;ii++)
        {
                WriteEEPROM(Addr+ii,BarcodeBuffer[ii]);
        }
        WriteEEPROM(Addr+8,SelBuff[0]);
        WriteEEPROM(Addr+9,SelBuff[1]);
        sprintf(StrBuff,"%2d%2d%2d%2d%2d%2dxxxxxxxxxxxx",(int)ReadDate(),(int)ReadMont
h(),(int)ReadYear(),(int)ReadHour(),(int)ReadMinute(),(int)ReadSecond());
        if((StrBuff[0]==' '))StrBuff[0]='0';
        if((StrBuff[2]==' '))StrBuff[2]='0';
        if((StrBuff[4]==' '))StrBuff[4]='0';
        if((StrBuff[6]==' '))StrBuff[6]='0';
        if((StrBuff[8]==' '))StrBuff[8]='0';
        if((StrBuff[10]==' '))StrBuff[10]='0';
        for(ii=0;ii<24;ii++)
        {
                WriteEEPROM(Addr+ii+10,StrBuff[ii]);
        }

}
Void SaveLogOut(void)
{
        Unsigned int Addr,ii;
        Addr=0x0400+(1839*35);

        for(ii=1840;ii>=0;ii--)
        {
                if(ReadEEPROM(Addr+8)==RecieveBuffer[0] //  Check Addr High
```

```
                    if(ReadEEPROM(Addr+9)==RecieveBuffer[1])  //  Check Addr Low
                        break;
                Addr-=35;
        }
        sprintf(StrBuff,"%2d%2d%2d%2d%2d%2d",(int)ReadDate(),(int)ReadMonth(),(int)ReadY
ear(),(int)ReadHour(),(int)ReadMinute(),(int)ReadSecond());
        if((StrBuff[0]==' '))StrBuff[0]='0';
        if((StrBuff[2]==' '))StrBuff[2]='0';
        if((StrBuff[4]==' '))StrBuff[4]='0';
        if((StrBuff[6]==' '))StrBuff[6]='0';
        if((StrBuff[8]==' '))StrBuff[8]='0';
        if((StrBuff[10]==' '))StrBuff[10]='0';
        for(ii=0;ii<13;ii++)
        {
                WriteEEPROM(Addr+ii+22,StrBuff[ii]);
        }
}
/*****************/
void SelectComputer(void)
{
                if((KeyIn=='*')&& (Index>0))
                {
                        Index=0;
                        putc(' ',24);
                        putc(' ',25);
                }
                Else if(KeyIn=='*')
                {
                        Select=0;
                        ClearLCD();
                }
```

```
Else if(KeyIn=='#')
{
        if(Index==2)
        {
                Num=SelBuff[0]-0x30;
                Num=Num*10;
                Num=Num+(SelBuff[1]-0x30);
                Num--;
                if((ComBuff[Num]==0)&&(Num<20))
                {
                        Select=0;
                        ClearLCD();
                        LogOnBuff[Num]=1;
                }
                else
                {
                        putc('E',29);
                        putc('r',30);
                        putc('r',31);
                        dmsec(400);
                        putc(' ',29);
                        putc(' ',30);
                        putc(' ',31);
                        Index=0;
                        putc(' ',24);
                        putc(' ',25);
                }
        }
        else
        {
                putc('E',29);
```

```
                    putc('r',30);
                    putc('r',31);
                    dmsec(400);
                    putc(' ',29);
                    putc(' ',30);
                    putc(' ',31);

                }
            }
            Else if((KeyIn>='0') && (KeyIn<='9') && (Index<2))
            {
                SelBuff[Index]=KeyIn;
                putc(KeyIn,24+Index);
                Index++;
            }
}
void RegisterMode(void)
{
            if(Add==1)// while Ask Add New Card
            {
                if(KeyIn=='1')
                {
                    AddNew(BarcodeBuffer);
                    ClearLCD();
                    Add=0;
                }
                if(KeyIn=='3')
                {
                    ClearLCD();
                    Add=0;
                }
```

```
}
Else if(Del==1)// while Ask Add New Card
{
        if(KeyIn=='1')
        {
                DelID(BarcodeBuffer);
                ClearLCD();
                Del=0;
        }
        if(KeyIn=='3')
        {
                ClearLCD();
                Del=0;
        }
}
Else if(ChangePwd==1) //*******Change Password******/
{
        if((KeyIn>='0') && (KeyIn<='9'))
        {
                putc(KeyIn,PwdCount+22);
                PwdBuff[PwdCount]=KeyIn;
                PwdCount++;

                if(PwdCount==4)
                {
                        SetPwd();
                        ChangePwd=0;
                        PrintLCD("Change Complete ",0);
                        ClearLCD();
                }
        }
```

```
}
Else if(BarcodeReady)
{
        BarcodeReady=0;
        // Check Add HERE
        if(Check ID (BarcodeBuffer)==0)
        {
                BarcodeBuffer[8]=0;
                PrintLCD(BarcodeBuffer,0);
                PrintLCD("Add ?(1=Yes,3=No",16);
                Add=1;
        }
        else
        {
                BarcodeBuffer[8]=0;
                PrintLCD(BarcodeBuffer,0);
                PrintLCD("Del ?(1=Yes,3=No",16);
                Del=1;
        }
}
Else if(KeyIn=='0')
{
        PrintLCD("Initialize ",0);
        PrintLCD("EEPROM ",16);
        // Code Here For Init EEPROM
        // SetPwd();
        FormatRegisterID();
        EraseLogData();
        PrintLCD("EEPROM Complete  ",16);
        dmsec(1700);
        ClearLCD();
```

```
                    }
                    Else if(KeyIn=='#')
                    {
                            ClearLCD();
                            PrintLCD("Change Password ",0);
                            ChangePwd=1;
                            PwdCount=0;
                    }

                    Else if(KeyIn=='*')
                    {
                            Mode=0;
                            ClearLCD();
                    }
                    else
                    {
                            putc('R',31);
                    }
}
void CheckPwd(void)
{
                    if((KeyIn>='0')  && (KeyIn<='9'))
                    {
                            putc('*',PwdCount+22);
                            if(PwdCorrect==1)
                            {
                                    if(KeyIn!=PwdBuff[PwdCount])
                                    {
                                            PwdCorrect=0;
                                    }
                            }
```

```
                        PwdCount++;
                        if(PwdCount==4)
                        {
                                if(PwdCorrect==1)
                                {
                                        PrintLCD(" Register Mode  ",0);
                                        Mode=1;
                                }
                                else
                                {
                                        Mode=0;
                                        PrintLCD(" Incorrect Pwd ",0);
                                }
                                dmsec(900);
                                InputPwd=0;
                                ClearLCD();
                        }
                }
}

Void main(void)
{
        Int i;
        Char Retry;
        Unsigned char Addr;
  InitializeLCD(); // LCD Monitor
  initializeRS232(); // Serial Port Initialize at 9600 bps 8,1,n
  EnableRTC(); // Start RTC


  IE=0x91;// Enable Serial Interrupt
```

```
// End Inittialize
Mode=0; // Mode=0 is RUN, Mode =1 is Register


InputPwd=0;
PwdCorrect=0;
i=0;
for(i=0;i<20;i++)
{
        ComBuff[i]=0;
        LogOnBuff[i]=0;
}
for(i=0;i<128;i++)
{
        IDBuff[i]=0;
}
       i=0;
       GetPwd();// Read Password From EEPROM
       Addr=0;
       TimeCount=1200;


       // RUNNING LOOP
       while(1)
       {
                // LED UpDate
                i++;
                if(i==20)i=0;
                if(ComBuff[i]==1)LED(i+1,0);else LED(i+1,1);
```

```
        KeyIn=GetKey(); // Read Key Pad

if(StartLoadLogData==1)  // #1 UpLoad Log data to Computer
        {
                SendLogData(LoadIndex);
                LoadIndex++;
        }
        Else if(Mode==1)   // #2 Register
        {
                RegisterMode();
        }
        Else if((InputPwd==0)&&(KeyIn=='*')) // #3 Enter To Input Password
        {
                InputPwd=1;
                PwdCount=0;
                PwdCorrect=1;
                ClearLCD();
                PrintLCD("*** Password ***",0);
        }
        Else if(InputPwd==1)   // #4 Key In Password
        {
                CheckPwd();
        }
        else if(BarcodeReady)     // #5 RecieveBarcode
        {
                BarcodeReady=0;
                // Check Add HERE
                Num2=CheckID(BarcodeBuffer);  // Check Card ID from EEPROM
                if(Num2>0)  // If Find
                {
```

```
                    ClearLCD();
                    BarcodeBuffer[8]=0;
                    PrintLCD("CARD ID:",0);
                    PrintLCD(BarcodeBuffer,9);
                    dmsec(800);
                    if(IDBuff[Num2-1]==0)// No Use
                    {
                            PrintLCD("CHOOSE COMPUTER ",0);
                            Select=1;// Set Flag
                            Index=0;
                    }
                    else
                    {
                            PrintLCD("Already Used.    ",16);
                            dmsec(1500);
                            ClearLCD();
                    }
            }
            else  // can not find
            {
                    PrintLCD("Card ID Invalid.",0);
                    dmsec(1500);
                    ClearLCD();
            }

    }
    Else if(Select==1)// #6 Select Computer
    {
            SelectComputer();
    }
    Else if(Start && Stop)// Receive Serial Data Start=':',Stop='$'
```

```
{
        Start=0;
        Stop=0;
        if(RecieveBuffer[0]=='T')// Set Date Time (From Computer)
        {
        // Write Date
                iBuff=RecieveBuffer[1]-0x30;
                iBuff=iBuff*10;
                iBuff=iBuff+RecieveBuffer[2]-0x30;
                WriteDate(iBuff);
        // WriteMonth
                iBuff=RecieveBuffer[3]-0x30;
                iBuff=iBuff*10;
                iBuff=iBuff+RecieveBuffer[4]-0x30;
                WriteMonth(iBuff);
        // WriteYear
                iBuff=RecieveBuffer[5]-0x30;
                iBuff=iBuff*10;
                iBuff=iBuff+RecieveBuffer[6]-0x30;
                WriteYear(iBuff);
        // WriteHour
                iBuff=RecieveBuffer[7]-0x30;
                iBuff=iBuff*10;
                iBuff=iBuff+RecieveBuffer[8]-0x30;
                WriteHour(iBuff);
        // WriteMinute
                iBuff=RecieveBuffer[9]-0x30;
                iBuff=iBuff*10;
                iBuff=iBuff+RecieveBuffer[10]-0x30;
                WriteMinute(iBuff);
        // WriteSecond
```

```
                    iBuff=RecieveBuffer[11]-0x30;
                    iBuff=iBuff*10;
                    iBuff=iBuff+RecieveBuffer[12]-0x30;
                    WriteSecond(iBuff);
                }
                Else if(RecieveBuffer[0]=='R')  // From Computer    ,For Read Log
Data in EEPROM
                {
                    StartLoadLogData=1;
                    LoadIndex=0;
                }
                Else if(RecieveBuffer[2]=='A') // From RS485
                {
                    if(LogOnBuff[Addr-1]==1)
                    {
                        LogOnBuff[Addr-1]=0;
                        ComBuff[Num]=1;
                        IDBuff[Num2-1]=Num;
                        SaveLogOn();
                    }
                }
                Else if(RecieveBuffer[2]=='O')SaveLogOut();  // From RS485


            }
            Elseif   (TimeCount--==0)//////////////////////
            {
                TimeCount=250;

                Addr++;
                if(Addr==21)Addr=1;// 1-20
```

```
// Send Header
        SendByte(':');
// Send Address
        if(Addr==20)
        {
                SendByte('2');
                SendByte('0');
        }
        Else if(Addr>=10)
        {
                SendByte('1');
                SendByte('0'+Addr-10);
        }
        else
        {
                SendByte('0');
                SendByte('0'+Addr);
        }
if(LogOnBuff[Addr-1]==1)//0-19
{
        SendByte('I');
}
else
{
        SendByte('R');
}
        SendByte('$');
PrintLCD(" INSERT CARD ",0);
sprintf(StrBuff,"%2d:%2d:%2d
",(int)ReadHour(),(int)ReadMinute(),(int)ReadSecond());
```

```
//
        sprintf(StrBuff,"%2d%2d%2d%2d%2d%2dI",(int)ReadDate(),(int)ReadMonth(),(int)Read
Year(),(int)ReadHour(),(int)ReadMinute(),(int)ReadSecond());
                        if((StrBuff[0]==' '))StrBuff[0]='0';
                        if((StrBuff[3]==' '))StrBuff[3]='0';
                        if((StrBuff[6]==' '))StrBuff[6]='0';
                        PrintLCD(StrBuff,20);
                }
        }
}
///// Lib RTC
#include "i2c.h"

#define RTC_ADDRESS 0xD0
#define RTC_WRITE   0x00
#define RTC_READ    0x01

#define RTC_SECONDS 0x00
#define RTC_MINUTES 0x01
#define RTC_HOURS   0x02
#define RTC_DAY     0x03
#define RTC_DATE    0x04
#define RTC_MONTH   0x05
#define RTC_YEAR    0x06
#define RTC_CONTROL 0x07

#define AM              0x00
#define PM              0x01
#define NOT             0xFF
#define dbyte unsigned int
#define byte unsigned char
```

```
#define bool bit
#define true 1
#define false 0

Enum DAY      {SUN = 0x01, MON, TUE, WED, THU, FRI, SAT};
Enum MONTH {JAN = 0x01, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
DEC};

#ifndef _ARE_RTC_LIBRARY_
#define _ARE_RTC_LIBRARY_

Extern void                    EnableRTC(void);
Extern void                    write_rtc(unsigned char address, unsigned char buffer);
Extern unsigned char read_rtc(unsigned char address);
Extern unsigned char ReadSecond(void);
Extern unsigned char ReadMinute(void);
Extern unsigned char ReadHour(void);
Extern unsigned char ReadAMPM(void);
Extern unsigned char ReadDay(void);
Extern unsigned char ReadDate(void);
Extern unsigned char ReadMonth(void);
Extern unsigned char ReadYear(void);
Extern unsigned char ReadHour12(void);
Extern unsigned char ReadHour24(void);
Extern void                    WriteSecond(unsigned char buffer);
Extern void                    WriteMinute(unsigned char buffer);
Extern void                    WriteHour12(unsigned char buffer, unsigned char ampm);
Extern void                    WriteHour24(unsigned char buffer);
Extern void                    WriteDay(unsigned char buffer);
Extern void                    WriteDate(unsigned char buffer);
Extern void                    WriteMonth(unsigned char buffer);
```

```
Extern void                 WriteYear(unsigned char buffer);
Extern void                 WriteTime12(unsigned char hour, unsigned char ampm,
unsigned char minute, unsigned char second);
Extern void                 WriteTime24(unsigned char hour, unsigned char minute,
unsigned char second);
Extern void                 WriteCalendar(unsigned char date, unsigned char month,
unsigned char year);

#endif
///// EEPROM //////

#ifndef _ARE_EEPROM_LIBRARY_
#define _ARE_EEPROM_LIBRARY_

extern void                 WriteEEPROM(unsigned int address, unsigned char value);
extern unsigned char ReadEEPROM(unsigned int address);
extern void                 WriteString(unsigned int address, unsigned char *buffer,
unsigned int ndata);
extern unsigned char *ReadString(unsigned int address, unsigned char *buffer, unsigned int
ndata);

#endif
////// Lib EEPROM /////
#include "i2c.h"

#ifndef _ARE_EEPROM_LIBRARY_
#define _ARE_EEPROM_LIBRARY_

extern void         EEPROM_delay(unsigned int msec);
extern void         WriteEEPROM(unsigned int address, unsigned char value);
extern void         set_address(unsigned int address);
```

```
extern unsigned char byte_read(void);
extern unsigned char ReadEEPROM(unsigned int address);
extern void          WriteString(unsigned int address, unsigned char *buffer, unsigned int ndata);
extern unsigned char *ReadString(unsigned int address, unsigned char *buffer, unsigned int ndata);
extern unsigned char part_write(unsigned int address, unsigned char *buffer, unsigned int ndata);
extern unsigned char part_read(unsigned int address, unsigned char *buffer, unsigned int ndata);

#endif

#define EEPROM_ADDRESS    0xA0
#define EEPROM_WRITE      0x00
#define EEPROM_READ       0x01
//// I2C ////

#ifndef _ARE_I2C_LIBRARY_
#define _ARE_I2C_LIBRARY_

extern void          I2C_Stop(void);
extern void          I2C_Start(void);
extern void          I2C_SendByte(unsigned char byteout);
extern unsigned char I2C_ReadAck(void);
extern void          I2C_SendAck(unsigned char ack);
extern unsigned char I2C_ReadByte(void);

#endif
////// Lib IC2 //////
#include <REG51F.H>
```

```
#ifndef _ARE_I2C_LIBRARY_
#define _ARE_I2C_LIBRARY_

extern void                 I2C_delay(void);
extern void                 I2C_Stop(void);
extern void                 I2C_Start(void);
extern void                 I2C_SendByte(unsigned char byteout);
extern unsigned char I2C_ReadAck(void);
extern void                 I2C_SendAck(unsigned char ack);
extern unsigned char I2C_ReadByte(void);

#endif

sbit SCL = P2^1; //STD
sbit SDA = P2^0;
///////// RTC /////

#define AM                  0x00
#define PM                  0x01
#define NOT                 0xFF

enum DAY{SUN = 0x01, MON, TUE, WED, THU, FRI, SAT};
enum MONTH {JAN = 0x01, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
DEC};

#ifndef _ARE_RTC_LIBRARY_
#define _ARE_RTC_LIBRARY_

extern void                         EnableRTC(void);
extern unsigned char ReadSecond(void);
```

```
extern unsigned char ReadMinute(void);
extern unsigned char ReadHour(void);
extern unsigned char ReadAMPM(void);
extern unsigned char ReadDay(void);
extern unsigned char ReadDate(void);
extern unsigned char ReadMonth(void);
extern unsigned char ReadYear(void);
extern unsigned char ReadHour12(void);
extern unsigned char ReadHour24(void);
extern void WriteSecond(unsigned char buffer);
extern void WriteMinute(unsigned char buffer);
extern void WriteHour12(unsigned char buffer, unsigned char ampm);
extern void WriteHour(unsigned char buffer);
extern void WriteDay(unsigned char buffer);
extern void WriteDate(unsigned char buffer);
extern void WriteMonth(unsigned char buffer);
extern void WriteYear(unsigned char buffer);
extern void WriteTime12(unsigned char hour, unsigned char ampm,
unsigned char minute, unsigned char second);
extern void WriteTime24(unsigned char hour, unsigned char minute,
unsigned char second);
extern void WriteCalendar(unsigned char date, unsigned char month,
unsigned char year);

#endif
```

โปรแกรมแลทของเครื่องควบคุมหลัก

```
#include <REG51F.H>
sbit SER=P2^2;
sbit RCK=P2^3;
```

```
sbit SRCK=P2^4;
bit L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13,L14,L15,L16,L17,L18,L19,L20;
void SrClk(void)
{


        SRCK=1;
        SRCK=0;



        }
void RClk(void)
{
        RCK=1;

        RCK=0;

        }

void ClrSHR(void)
{
        char ii;
        for(ii=0;ii<26;ii++)
        {
                SER=0;
                SrClk();
        }
        RClk();
}


void UpOut(void)
{
```

```
SER=1;
SrClk();
if(L20)SER=1; else SER=0;   // 1.7
SrClk();
if(L19)SER=1; else SER=0;   // 1.6
SrClk();
if(L18)SER=1; else SER=0;          // 1.5
SrClk();
if(L17)SER=1; else SER=0;   // 1.4
SrClk();
if(L16)SER=1; else SER=0; //1.3
SrClk();
if(L15)SER=1; else SER=0; //1.3
SrClk();

SER=1;
SrClk();

if(L14)SER=1; else SER=0; //1.1
SrClk();


if(L13)SER=1; else SER=0;   // 1.7
SrClk();
if(L12)SER=1; else SER=0;   // 1.6
SrClk();
if(L11)SER=1; else SER=0;          // 1.5
SrClk();
if(L10)SER=1; else SER=0;   // 1.4
SrClk();
if(L9)SER=1; else SER=0; //1.3
```

```
        SrClk();
        if(L8)SER=1; else SER=0; //1.3
        SrClk();

        SER=1;
        SrClk();

        if(L7)SER=1; else SER=0; //1.1
        SrClk();
        if(L6)SER=1; else SER=0;   // 1.7
        SrClk();
        if(L5)SER=1; else SER=0;   // 1.6
        SrClk();
        if(L4)SER=1; else SER=0;          // 1.5
        SrClk();
        if(L3)SER=1; else SER=0;   // 1.4
        SrClk();
        if(L2)SER=1; else SER=0; //1.3
        SrClk();
        if(L1)SER=1; else SER=0; //1.3
        SrClk();
        SER=1;
        SrClk();

        RClk();
        RClk();
}
```

โปรแกรมเรียลไทม์ของเครื่องควบคุมหลัก

```
#include <REG51F.H>

#define RTC_ADDRESS 0xD0
#define RTC_WRITE   0x00
#define RTC_READ    0x01

#define RTC_SECONDS 0x00
#define RTC_MINUTES 0x01
#define RTC_HOURS   0x02
#define RTC_DAY     0x03
#define RTC_DATE    0x04
#define RTC_MONTH   0x05
#define RTC_YEAR    0x06
#define RTC_CONTROL 0x07

#define AM          0x00
#define PM          0x01
#define NOT         0xFF

enum DAY   {SUN = 0x01, MON, TUE, WED, THU, FRI, SAT};
enum MONTH {JAN = 0x01, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
DEC};

extern void          I2C_delay(void);
extern void          I2C_Stop(void);
extern void          I2C_Start(void);
extern void          I2C_SendByte(unsigned char byteout);
extern unsigned char I2C_ReadAck(void);
extern void          I2C_SendAck(unsigned char ack);
extern unsigned char I2C_ReadByte(void);
```

```
unsigned char read_rtc(unsigned char address)
{
        unsigned char buffer;

        I2C_Start();
        I2C_SendByte(RTC_ADDRESS | RTC_WRITE);
        I2C_ReadAck();
        I2C_SendByte(address);
        I2C_ReadAck();
        I2C_Stop();

        I2C_Start();
        I2C_SendByte(RTC_ADDRESS | RTC_READ);
        I2C_ReadAck();
        buffer = I2C_ReadByte();
        I2C_SendAck(0);
        I2C_Stop();

        return buffer;
}
void write_rtc(unsigned char address, unsigned char buffer)
{
        I2C_Start();
        I2C_SendByte(RTC_ADDRESS | RTC_WRITE);
        I2C_ReadAck();
        I2C_SendByte(address);
        I2C_ReadAck();
        I2C_SendByte(buffer);
        I2C_ReadAck();
        I2C_Stop();
```

```
}
void EnableRTC(void)
{
        unsigned char old_second;

        old_second = read_rtc(0) & 0x7F;
        I2C_Start();
        I2C_SendByte(RTC_ADDRESS | RTC_WRITE);
        I2C_ReadAck();
        I2C_SendByte(0x00);
        I2C_ReadAck();
        I2C_SendByte(old_second);
        I2C_ReadAck();
        I2C_Stop();
}

void WriteDate(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_DATE, buffer & 0x3F);
}
void WriteHour(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_HOURS, (buffer & 0x3F) & 0xBF);
}
void WriteMinute(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_MINUTES, buffer & 0x7F);
}
```

```
void WriteMonth(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_MONTH, buffer & 0x1F);
}
void WriteSecond(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_SECONDS, buffer & 0x7F);
}void WriteYear(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_YEAR, buffer);
}
unsigned char ReadSecond(void)
{
        unsigned char buffer = read_rtc(RTC_SECONDS);

        return (((buffer & 0x7F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadDate(void)
{
        unsigned char buffer = read_rtc(RTC_DATE);

        return (((buffer & 0x3F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadHour(void)
{
        unsigned char buffer, temp;

        buffer = read_rtc(RTC_HOURS);
```

```
        if (buffer & 0x40)
                temp = 0x1F;
        else
                temp = 0x3F;

        return (((buffer & temp) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadMinute(void)
{
        unsigned char buffer = read_rtc(RTC_MINUTES);

        return (((buffer & 0x7F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadMonth(void)
{
        unsigned char buffer = read_rtc(RTC_MONTH);

        return (((buffer & 0x1F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadYear(void)
{
        unsigned char buffer = read_rtc(RTC_YEAR);

        return ((buffer >> 4) * 10) + (buffer & 0x0F);
}
```

โปรแกรมส่วนที่ติดต่อกับอีอีพรอมของเครื่องควบคุมหลัก

```
//Byte Read
#include "lib_eeprom.h"
```

```
unsigned char byte_read(void)
{
        unsigned char result;

        I2C_Start();
        I2C_SendByte(EEPROM_ADDRESS | EEPROM_READ);
        I2C_ReadAck();
        result = I2C_ReadByte();
        I2C_SendAck(0);
        I2C_Stop();

        return result;
}
///Byte write
#include "lib_eeprom.h"

void WriteEEPROM(unsigned int address, unsigned char value)
{
        I2C_Start();
        I2C_SendByte(EEPROM_ADDRESS | EEPROM_WRITE);
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address >> 8));
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address & 0xFF));
        I2C_ReadAck();
        I2C_SendByte(value);
        I2C_ReadAck();
        I2C_Stop();

        EEPROM_delay(25);
}
```

```
///// EEPROM LELAY
#include "lib_eeprom.h"

void EEPROM_delay(unsigned int msec)
{
        unsigned char i;

        for (; msec--;)
                for (i=0; i<170; i++)
                        ;
}
///// Page read
#include "lib_eeprom.h"

unsigned char *ReadString(unsigned int address, unsigned char *buffer, unsigned int ndata)
{
        unsigned int  nread;
        unsigned char *temp;

        temp = buffer;
        for (; ndata;)
        {
                nread = part_read(address, temp, ndata);
                address += nread;
                temp += nread;
                ndata -= nread;
        }

        return buffer;
}
//// Page Write
```

```
#include "lib_eeprom.h"

void WriteString(unsigned int address, unsigned char *buffer, unsigned int ndata)
{
        unsigned char nwrite;

        for (; ndata;)
        {
                nwrite = part_write(address, buffer, ndata);
                address += nwrite;
                buffer += nwrite;
                ndata -= nwrite;
        }
}
//////Part Read
#include "lib_eeprom.h"

unsigned char part_read(unsigned int address, unsigned char *buffer, unsigned int ndata)
{
        unsigned char i;

        set_address(address);

        for (i=0; i<ndata; i++)
        {
                if (!((address + i) % 64))
                        if (i)
                                break;
                buffer[i] = byte_read();
        }
        buffer[ndata] = '\0';
```

```
        return i;
}
////// Part Write
#include "lib_eeprom.h"

unsigned char part_write(unsigned int address, unsigned char *buffer, unsigned int ndata)
{
        unsigned char i;

        I2C_Start();
        I2C_SendByte(EEPROM_ADDRESS | EEPROM_WRITE);
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address >> 8));
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address & 0xFF));
        I2C_ReadAck();

        for (i=0; i<ndata; i++)
        {
                if (!((address + i) % 64))
                        if (i)
                                break;
                I2C_SendByte(buffer[i]);
                I2C_ReadAck();
        }
        I2C_Stop();
        EEPROM_delay(10);

        return i;
}
```

```
//// Part Write
#include "lib_eeprom.h"

unsigned char ReadEEPROM(unsigned int address)
{
        I2C_Start();
        I2C_SendByte(EEPROM_ADDRESS | EEPROM_WRITE);
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address >> 8));
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address & 0xFF));
        I2C_ReadAck();
        I2C_Stop();

        return byte_read();
}
////// Set Address
#include "lib_eeprom.h"

void set_address(unsigned int address)
{
        I2C_Start();
        I2C_SendByte(EEPROM_ADDRESS | EEPROM_WRITE);
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address >> 8));
        I2C_ReadAck();
        I2C_SendByte((unsigned char)(address & 0xFF));
        I2C_ReadAck();
        I2C_Stop();
}
```

โปรแกรม I2C

```
#include "lib_i2c.h"

void I2C_delay(void)
{
        unsigned char msec = 1;

        while (msec--);
}
#include "lib_i2c.h"

unsigned char I2C_ReadAck(void)
{
        unsigned char result;

        SCL = 1;
        I2C_delay();
        result = SDA;
        SCL = 0;

        return !result;
}
#include "lib_i2c.h"

unsigned char I2C_ReadByte(void)
{
        unsigned char i, result;

        result = 0x00;
        for (i=0; i<8; i++)
        {
                SCL = 1;
```

```
                I2C_delay();
                result <<= 1;
                result |= (unsigned char)SDA;
                SCL = 0;
                I2C_delay();
        }

        return result;
}
#include "lib_i2c.h"

void I2C_SendAck(unsigned char ack)
{
        SDA = (bit)(!ack);
        SCL = 1;
        I2C_delay();
        I2C_delay();
        SCL = 0;
        SDA = 1;
}
#include "lib_i2c.h"

void I2C_SendByte(unsigned char byteout)
{
        unsigned char i;

        for (i=0; i<8; i++)
        {
                SCL = 0;
                SDA = (byteout >> (7 - i)) & 0x01;
                SCL = 1;
```

```
                I2C_delay();
        }
        SCL = 0;
        SDA = 1;
}
#include "lib_i2c.h"

void I2C_Start(void)
{
        SDA = 1;
        SCL = 1;
        SDA = 0;
        SCL = 0;
}
#include "lib_i2c.h"

void I2C_Stop(void)
{
        SDA = 0;
        SCL = 1;
        SDA = 1;
}

#include <REG51F.H>

#define RTC_ADDRESS 0xD0
#define RTC_WRITE    0x00
#define RTC_READ     0x01

#define RTC_SECONDS 0x00
#define RTC_MINUTES 0x01
```

```
#define RTC_HOURS    0x02
#define RTC_DAY      0x03
#define RTC_DATE     0x04
#define RTC_MONTH    0x05
#define RTC_YEAR     0x06
#define RTC_CONTROL0x07

#define AM           0x00
#define PM           0x01
#define NOT          0xFF

enum DAY     {SUN = 0x01, MON, TUE, WED, THU, FRI, SAT};
enum MONTH {JAN = 0x01, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,
DEC};

extern void              I2Cdelay(void);
extern void              I2CStop(void);
extern void              I2CStart(void);
extern void              I2CSendByte(unsigned char byteout);
extern unsigned char I2CReadAck(void);
extern void              I2CSendAck(unsigned char ack);
extern unsigned char I2CReadByte(void);

unsigned char read_rtc(unsigned char address)
{
        unsigned char buffer;

        I2CStart();
        I2CSendByte(RTC_ADDRESS | RTC_WRITE);
        I2CReadAck();
        I2CSendByte(address);
```

```
        I2CReadAck();
        I2CStop();

        I2CStart();
        I2CSendByte(RTC_ADDRESS | RTC_READ);
        I2CReadAck();
        buffer = I2CReadByte();
        I2CSendAck(0);
        I2CStop();

        return buffer;
}
void write_rtc(unsigned char address, unsigned char buffer)
{
        I2CStart();
        I2CSendByte(RTC_ADDRESS | RTC_WRITE);
        I2CReadAck();
        I2CSendByte(address);
        I2CReadAck();
        I2CSendByte(buffer);
        I2CReadAck();
        I2CStop();
}
void EnableRTC(void)
{
        unsigned char old_second;

        old_second = read_rtc(0) & 0x7F;
        I2CStart();
        I2CSendByte(RTC_ADDRESS | RTC_WRITE);
        I2CReadAck();
```

```
        I2CSendByte(0x00);
        I2CReadAck();
        I2CSendByte(old_second);
        I2CReadAck();
        I2CStop();
}

void WriteDate(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_DATE, buffer & 0x3F);
}
void WriteHour(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_HOURS, (buffer & 0x3F) & 0xBF);
}
void WriteMinute(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_MINUTES, buffer & 0x7F);
}
void WriteMonth(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_MONTH, buffer & 0x1F);
}
void WriteSecond(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_SECONDS, buffer & 0x7F);
```

```
}void WriteYear(unsigned char buffer)
{
        buffer = ((buffer / 10) << 4) | (buffer % 10);
        write_rtc(RTC_YEAR, buffer);
}
unsigned char ReadSecond(void)
{
        unsigned char buffer = read_rtc(RTC_SECONDS);

        return (((buffer & 0x7F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadDate(void)
{
        unsigned char buffer = read_rtc(RTC_DATE);

        return (((buffer & 0x3F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadHour(void)
{
        unsigned char buffer, temp;

        buffer = read_rtc(RTC_HOURS);
        if (buffer & 0x40)
                temp = 0x1F;
        else
                temp = 0x3F;

        return (((buffer & temp) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadMinute(void)
{
```

```
        unsigned char buffer = read_rtc(RTC_MINUTES);

        return (((buffer & 0x7F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadMonth(void)
{
        unsigned char buffer = read_rtc(RTC_MONTH);

        return (((buffer & 0x1F) >> 4) * 10) + (buffer & 0x0F);
}
unsigned char ReadYear(void)
{
        unsigned char buffer = read_rtc(RTC_YEAR);

        return ((buffer >> 4) * 10) + (buffer & 0x0F);
}
```

โปรแกรมเครื่องควบคุมย่อย

```
/////// RS232
#include <AT892051.H>
sbit LogOffCmd=P1^7;
sbit LogOffAck=P3^2;
sbit LogOnCmd=P1^6;
sbit LogOnAck=P3^3;

bit AddrCheck;
bit HaveLogOff;
bit Start,Stop;
char Index;
unsigned char RecieveBuffer[3];
void delay(unsigned int Time)
```

```
{
        while(Time)
        {
                Time--;
                Time++;
                Time--;
        }
}
void InitSerial(void)// at 9600
{
        // Set Timer 1 for Boudrate 2400
        TMOD=0x20;
        TL1=0xFD
        ;///
        TH1=0xFD;//
        SCON=0x50;
        TR1=1;
        TI=0;
        RI=0;
}
void RS232_ISR(void) interrupt 4
{
        if(RI)
        {
                RI=0;
                if(!Start&&SBUF==':')
                {
                        Index=0;
                    Start=1;
                        Stop=0;
                }
```

```
                else if(Start&&SBUF=='$')
                {
                        Stop=1;
                        RecieveBuffer[Index]=0;
                }
                else if(Start&&!Stop)
            {
                        RecieveBuffer[Index]=SBUF;
                        Index++;
                        if(Index>100)
                                Start=0;
                }
                RI=0;
        }

}
void SendByte(unsigned char DataByte)
{
        SBUF=DataByte;
        while(TI==0);
        TI=0;
}
void main(void)
{
        InitSerial();
        while(1)
        {
                if((Start==1)&&(Stop==1))
                {
                        Start=0;
                        Stop=0;
```

```
                    if(RecieveBuffer[0]=='O')
                    {
                        LogOffCmd=0;// Send LogOffCmd
                    }
                }
                if(LogOffCmd==0)
                {
                    if(LogOffAck==0)// if Ack from RS485 Controller
                    {
                        LogOffCmd=1;// Finish Send LogOffCmd
                    }
                }
                if(LogOnCmd==0)
                {
                    LogOnAck=0;
                    SendByte(':');
                    SendByte('I');
                    SendByte('$');
                }
                else
                {
                    LogOnAck=1;
                }

        }
}
///// RS485
#include <AT892051.H>

// 3.2=>1.7,3.3=>1.6
```

```
#define NodeAddrH '1'
#define NodeAddrL '1'
sbit LogOffCmd=P3^2;
sbit LogOffAck=P1^7;
sbit LogOnCmd=P3^3;
sbit LogOnAck=P1^6;

sbit EN_485_TX=P1^5;
bit HaveLogOff;
bit SendLogOff;
bit SendLogOn,SendAck;
bit Start,Stop;
char Index;
unsigned char RecieveBuffer[5];
void delay(unsigned int Time)
{
        while(Time)
        {
                Time--;
                Time++;
                Time--;
        }
}
void InitSerial(void)// at 9600
{
        // Set Timer 1 for Boudrate 2400
        TMOD=0x20;
        TL1=0xFD;///
        TH1=0xFD;//
        SCON=0x50;
        TR1=1;
```

```
        TI=0;
        RI=0;

        IE=0x90;

}
void RS232_ISR(void) interrupt 4
{
        if(RI)
        {
                RI=0;
                if(!Start&&SBUF==':')
                {
                        Index=0;
                    Start=1;
                        Stop=0;
                }
                else if(Start&&SBUF=='$')
                {
                        Stop=1;
                        RecieveBuffer[Index]=0;
                }
                else if(Start&&!Stop)
            {
                        RecieveBuffer[Index]=SBUF;
                        Index++;
                        if(Index>100)
                                Start=0;
                }
                RI=0;
        }
```

```
}
void SendByte(unsigned char DataByte)
{
        SBUF=DataByte;
        while(TI==0);
        TI=0;
}
void main(void)
{
        InitSerial();
        EN_485_TX=0;
        while(1)
        {
                // Check Log Off Command from RS232 Controller
                if(LogOffCmd==0)// If LogOffCmd Occur
                {
                        LogOffAck=0; // Reply Ack to RS232 Controller
                        HaveLogOff=1;
                }
                else
                {
                        LogOffAck=1;
                }
                // while send LogOnCmd
                if(LogOnCmd==0)
                {
                        if(LogOnAck==0)
                        {
                                LogOnCmd=1;// Finish Send LogOnCmd
                        }
```

```
}

// Check Data From RS485
if((Start==1) && (Stop==1))
{
        Start=0;
        Stop=0;
        if((RecieveBuffer[0]==NodeAddrH)                                    &&
(RecieveBuffer[1]==NodeAddrL))
        {
            EN_485_TX=1;
            switch(RecieveBuffer[2])
            {
                case 'R':
                    if(HaveLogOff==1)
                    {
                        HaveLogOff=0;
                        SendByte(':');
                        SendByte(NodeAddrH);
                        SendByte(NodeAddrL);
                        SendByte('O');
                        SendByte('$');
                    }
                    else
                    {
                        SendByte(':');
                        SendByte(NodeAddrH);
                        SendByte(NodeAddrL);
```

```
                                        SendByte('A');
                                        SendByte('$');

                                    }
                                break;
                                case 'I':
                                        SendByte(':');
                                        SendByte(NodeAddrH);
                                        SendByte(NodeAddrL);
                                        SendByte('A');
                                        SendByte('$');

                                        LogOnCmd=0; // Start Send LogOnCmd
                                break;
                            }
                        EN_485_TX=0;
                    }
                }
        }
}
```

โปรแกรมควบคุมเมาส์และคีย์บอร์ด

```
Private Declare Function BlockInput Lib "user32" (ByVal fBlock As Long) As Long


Private Sub Buffer_txt_Change()
    Dim s, f, x
    If Buffer_txt.Text = "" Then Exit Sub
    s = InStr(1, Buffer_txt.Text, ":")
    f = InStr(2, Buffer_txt.Text, "$")
    x = Mid(Buffer_txt.Text, s + 1, 1)
    ':I$
    If x = "I" Then
```

```
        Image1.Visible = False
        Label1.Visible = False
        Label2.Visible = False
        Form1.BackColor = &H80000009
        Image2.Visible = True
        Command1.Visible = True
        Command1.Enabled = True
        'BlockInput False
    End If
    Buffer_txt.Text = ""
End Sub

Private Sub Command1_Click()
    If Command1.Visible = True Then
        MSComm1.Output = ":O$"
        Image1.Visible = True
        Label1.Visible = True
        Label2.Visible = True
        Form1.BackColor = &H0&
        Image2.Visible = False
        Command1.Visible = False
        Command1.Enabled = False
        'BlockInput True
    End If
End Sub

Private Sub Form_Load()
    MSComm1.PortOpen = True
    'BlockInput True
End Sub
```

```
Private Sub MSComm1_OnComm()
    Buffer_txt.Text = Buffer_txt.Text & MSComm1.Input
End Sub
```

โปรแกรมโหลดค่าจากอีอีพรอม

```
Dim count2 As Integer
Private Sub Buffer_txt_Change()
    'Data=":SSSSSSSSCCDDMMYYHHmmSSDDMMYYHHmmSSX$"
    Dim s, f, id, com, d_in, m_in, y_in, th_in, tm_in, ts_in, d_out, m_out, y_out, th_out,
tm_out, ts_out As String
    Dim date_in, date_out, time_in, time_out As String
    If Timer2.Enabled = True Then
        Buffer_txt.Text = ""
    End If
    If Buffer_txt.Text = "" Then Exit Sub
    s = InStr(1, Buffer_txt.Text, ":")
    f = InStr(2, Buffer_txt.Text, "$")
    If s > 0 And f < s And f > 0 Then
        Buffer_txt.Text = ""
        Exit Sub
    End If
    If s > 0 And f > 0 And f < 6 Then
        Buffer_txt.Text = ""
        Exit Sub
    End If
    If s > 0 And f = 0 Then Exit Sub

    id = Mid(Buffer_txt.Text, s + 1, 8)
    com = Mid(Buffer_txt.Text, s + 9, 2)
    d_in = Mid(Buffer_txt.Text, s + 11, 2)
    m_in = Mid(Buffer_txt.Text, s + 13, 2)
```

```
    y_in = Mid(Buffer_txt.Text, s + 15, 2)
    th_in = Mid(Buffer_txt.Text, s + 17, 2)
    tm_in = Mid(Buffer_txt.Text, s + 19, 2)
    ts_in = Mid(Buffer_txt.Text, s + 21, 2)
    d_out = Mid(Buffer_txt.Text, s + 23, 2)
    m_out = Mid(Buffer_txt.Text, s + 25, 2)
    y_out = Mid(Buffer_txt.Text, s + 27, 2)
    th_out = Mid(Buffer_txt.Text, s + 29, 2)
    tm_out = Mid(Buffer_txt.Text, s + 31, 2)
    ts_out = Mid(Buffer_txt.Text, s + 33, 2)
    date_in = d_in & "/" & m_in & "/" & y_in
    date_out = d_out & "/" & m_out & "/" & y_out
    time_in = th_in & ":" & tm_in & ":" & ts_in
    time_out = th_out & ":" & tm_out & ":" & ts_out
    count2 = count2 + 1

        Set itemreturn = ListViewRun.ListItems.Add()
        itemreturn.SubItems(1) = count2
        itemreturn.SubItems(2) = id
        itemreturn.SubItems(3) = com
        itemreturn.SubItems(4) = date_in & "        " & time_in
        itemreturn.SubItems(5) = date_out & "        " & time_out


    Buffer_txt.Text = ""
End Sub


Private Sub Command1_Click()
    If Command1.Caption = "Open" Then
        MSComm1.CommPort = cmb_Com.Text
        MSComm1.Settings = cmb_Buadrate.Text & ",n,8,1"
        MSComm1.PortOpen = True
```

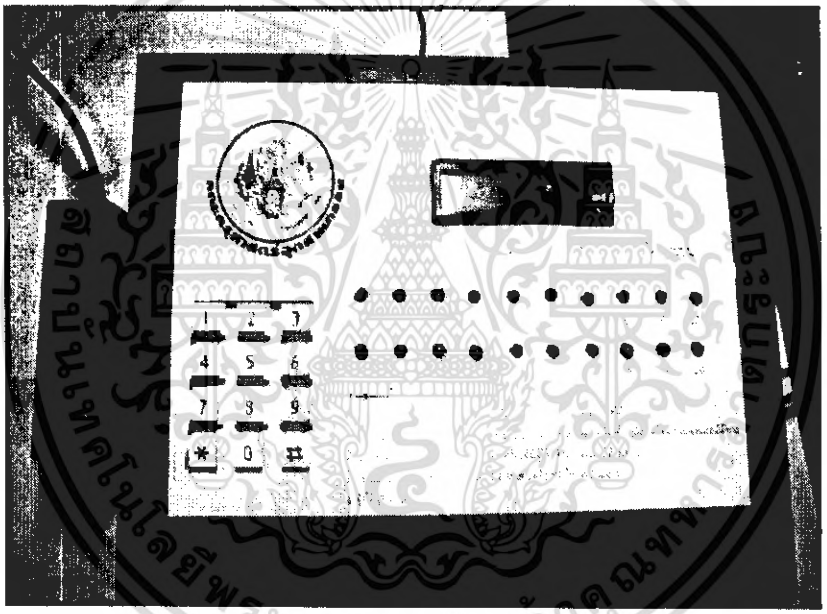
```
        Command1.Caption = "Close"
        Frame1.Enabled = True
        Frame2.Enabled = True
        Command2.Enabled = True
        Command3.Enabled = True
        Command6.Enabled = True
        Timer2.Enabled = True
    Else
        MSComm1.PortOpen = False
        Command1.Enabled = False
        Command1.Caption = "Open"
        cmb_Com.Text = "None"
        Frame1.Enabled = False
        Frame2.Enabled = False
        Command2.Enabled = False
        Command3.Enabled = False
        Command6.Enabled = False
    End If
End Sub

Private Sub cmb_Com_click()
    Command1.Enabled = True
End Sub

Private Sub Command2_Click()
    MSComm1.Output = ":R$"
    ListViewRun.ListItems.Clear
    count2 = 0
End Sub

Private Sub Command3_Click()
```

```
    MSComm1.Output = ":D$"
End Sub

Private Sub Command4_Click()
    ListViewRun.ListItems.Clear
End Sub

Private Sub Command6_Click()
    Dim a As String
    a = Format(Now, "DD" & "MM" & "YY" & "hh" & "mm" & "ss")
    MSComm1.Output = ":T" & a & "$"
End Sub

Private Sub Command7_Click()
    Dim count3 As Integer
    Set Exlobj = CreateObject("excel.application")
    Exlobj.Workbooks.Add
    Exlobj.Visible = True
    With Exlobj.ActiveSheet
    .Cells(1, 1).Value = "NO"
    .Cells(1, 2).Value = "Student ID"
    .Cells(1, 3).Value = "Computer"
    .Cells(1, 4).Value = "Date/Time Log In"
    .Cells(1, 5).Value = "Date/Time Log Out"
     For count3 = 1 To ListViewRun.ListItems.Count
        .Cells(count3 + 1, 1).Value = ListViewRun.ListItems(count3).ListSubItems(1).Text
        .Cells(count3 + 1, 2).Value = ListViewRun.ListItems(count3).ListSubItems(2).Text
        .Cells(count3 + 1, 3).Value = ListViewRun.ListItems(count3).ListSubItems(3).Text
        .Cells(count3 + 1, 4).Value = ListViewRun.ListItems(count3).ListSubItems(4).Text
        .Cells(count3 + 1, 5).Value = ListViewRun.ListItems(count3).ListSubItems(5).Text
     Next
```

```
    End With
End Sub

Private Sub MSComm1_OnComm()
    Buffer_txt.Text = Buffer_txt.Text & MSComm1.Input
End Sub

Private Sub Timer1_Timer()
    lbTime.Caption = Format(Now, "DD MMMM YYYY HH:MM:SS")
End Sub

Private Sub Timer2_Timer()
    Timer2.Enabled = False
End Sub
```

ภาคผนวก ช
คู่มือการใช้งาน

คู่มือการใช้งาน
ระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม



ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

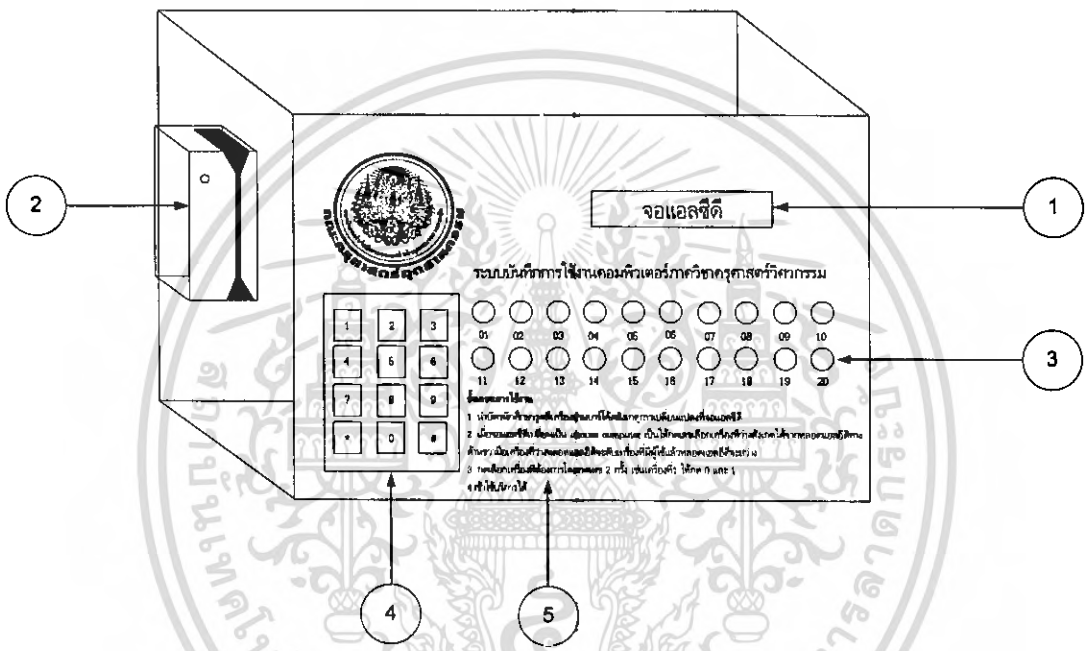
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

1. คำแนะนำเบื้องต้น

ก่อนใช้งานระบบบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม ควรทำการศึกษาการใช้งานเบื้องต้นจากคู่มือก่อน เพื่อให้เข้าใจการใช้งานที่ถูกต้องและเป็นการป้องกันการเสียหายที่อาจจะเกิดขึ้นได้

2. ส่วนประกอบและปุ่มควบคุม

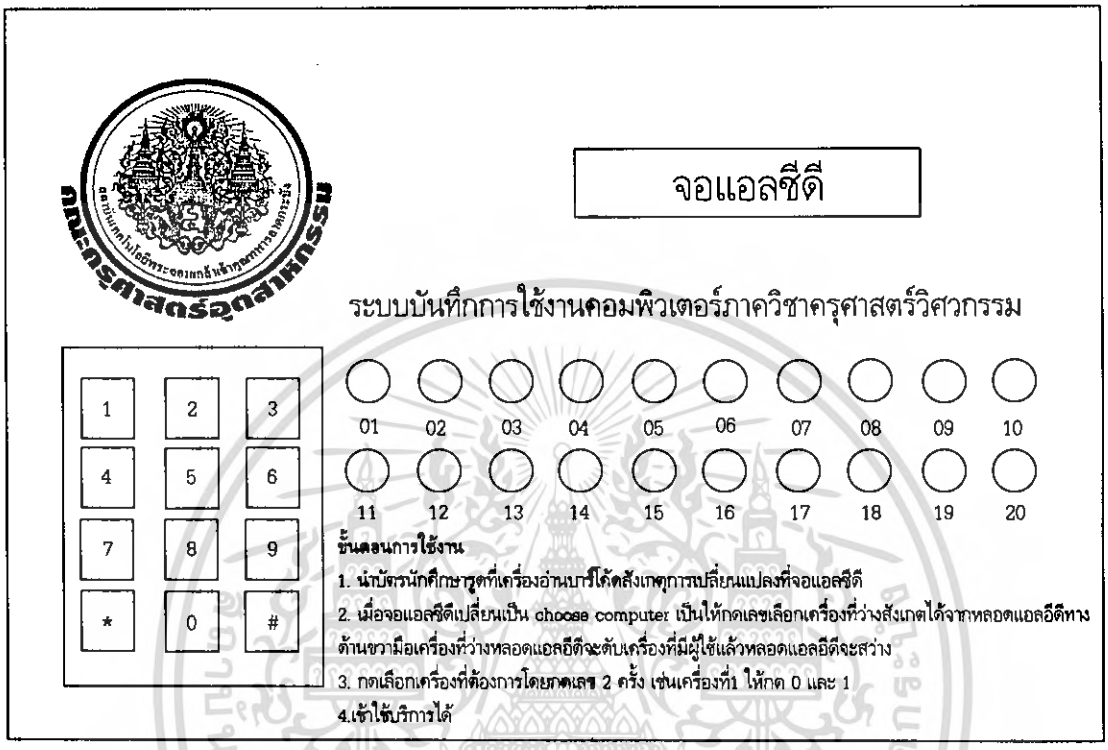


รูปที่ ช.1 หน้าปัดเครื่องควบคุมหลักของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

จากรูปที่ ช.1 มีรายละเอียดต่างๆดังนี้

1. จอแอลซีดีแสดงขั้นตอนการทำงาน
2. เครื่องอ่านแถบบาร์โค้ดจากบัตรนักศึกษา
3. หลอดแอลอีดีจำนวน 20 หลอดแสดงสถานะของเครื่องคอมพิวเตอร์
4. สวิตช์เลือกเครื่องและโหมดการทำงานของเครื่องควบคุมหลัก
5. ขั้นตอนการใช้งาน

3. การติดตั้งและการใช้งาน



รูปที่ ช.2 หน้าปัดเครื่องควบคุมหลักของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

3.1 ต่อเครื่องอ่านบาร์โค้ดเข้าที่ Input เครื่องควบคุมหลักบริเวณด้านล่างของเครื่องควบคุมหลัก

3.2 นำสาย RS485 เชื่อมต่อระหว่างเครื่องควบคุมหลักไปยังเครื่องควบคุมย่อยโดยต่อสายจาก Output ของเครื่องควบคุมหลักเข้ากับ Input ของเครื่องควบคุมย่อยทุกเครื่อง

3.3 นำสาย RS232 มาต่อเข้าที่ Output ของเครื่องควบคุมย่อยจากนั้นนำปลายสายไปต่อเข้ากับเครื่องคอมพิวเตอร์เข้าทางพอร์ต RS232

3.4 เปิดเครื่องควบคุมหลัก เครื่องควบคุมย่อย และเครื่องคอมพิวเตอร์

3.5 การลงทะเบียนเพื่อให้เข้าใช้งานได้สามารถทำได้เจ้าของระบบเนื่องจากต้องป้อนรหัสก่อนเข้าระบบเพื่อเพิ่มหรือลดจำนวนสมาชิกในระบบ

3.6 กด * จากนั้นป้อนรหัส 1234 เพื่อเข้าสู่การเพิ่มหรือลบสมาชิกของระบบ

3.7 นำบัตรบาร์โค้ดที่ต้องการจะเพิ่มเข้าไปในระบบรูดที่เครื่องอ่านบาร์โค้ด รหัสนักศึกษาที่อ่านได้จะแสดงที่หน้าจอแอลซีดี

3.8 จะมีข้อความปรากฏขึ้นด้านล่างของจอแอลซีดีมีอยู่ 2 กรณี คือถ้าขึ้นคำว่า Add หมายถึงต้องการที่จะเพิ่มรหัสนี้เข้าไปในระบบหรือไม่ และอีกข้อความคือ Del จะขึ้นในกรณีที่รหัสที่ป้อนเข้าไปให้ม่ถูก

ตรวจสอบแล้วว่าได้ถูกบันทึกไว้ก่อนหน้านี้แล้วแล้วทั้ง 2 กรณี การกดยืนยันการเพิ่มหรือลบรหัสออกจากเครื่องควบคุมหลักให้กดหมายเลข 1 การยกเลิกการการเพิ่มหรือลบรหัสออกจากเครื่องควบคุมหลักให้กดหมายเลข 3

3.9 การออกจากโหมดลงทะเบียนให้กดเครื่องหมาย #

3.10 กลับมาอยู่ในโหมดการเข้าใช้งานเครื่องคอมพิวเตอร์

3.11 สังเกตที่หน้าจอของเครื่องควบคุมหลักจะแสดงคำว่า "INSERT CARD"

3.12 ทำการรูดบัตรนักศึกษาเข้ากับเครื่องรับบัตรบาร์โค้ด

3.13 สังเกตการเปลี่ยนแปลงที่จอดิสเพล

3.14 ถ้ามีข้อความ Card ID Invalid แสดงว่ารหัสของท่านไม่มีในฐานข้อมูล

3.15 ถ้ามีการเปลี่ยนแปลงจาก "INSERT CARD" เป็น "CHOOSE COMPUTER" แสดงว่าท่านสามารถเลือกเครื่องใช้งานได้

3.16 เลือกเครื่องคอมพิวเตอร์ที่ต้องการโดยเครื่องที่ใช้งานได้จะมีหลอดแอลอีดีแสดงให้เห็นถ้าเครื่องว่างหลอดแอลอีดีจะติด

3.17 ทำการเลือกคอมพิวเตอร์ด้วยการกดสวิตช์สองครั้งโดยจากเครื่องที่ 1 ถึง 9 ให้กดเลขศูนย์นำหน้าเลขเครื่อง

3.18 หากกดเลขเครื่องผิดสามารถแก้ไขได้โดยการกด * จะเป็นการลบหมายเลขเครื่องที่ทำการกดไปจากนั้นให้กดหมายเลขเครื่องใหม่อีกครั้ง

3.19 เมื่อเลือกเครื่องที่ต้องการได้แล้วให้กดเครื่องหมาย # เพื่อยืนยันการเข้าใช้งาน

3.20 เข้าใช้งานตามหมายเลขเครื่องที่เลือกไว้

3.21 โปรแกรมบนเครื่องคอมพิวเตอร์หากยังไม่มีการเลือกเข้าใช้งานตัวโปรแกรมจะล็อคไม่ให้สามารถใช้งานเมาส์และคีย์บอร์ดได้หากมีการถูกเลือกเข้าใช้งานโปรแกรมที่เครื่องคอมพิวเตอร์จะทำการปลดล็อคเมาส์และคีย์บอร์ดทำให้สามารถใช้งานคอมพิวเตอร์ได้ตามปกติ

3.22 หากต้องการเลิกใช้งานสามารถทำได้โดยการกดล็อคเอาท์ที่ปุ่ม Lock Out บนหน้าจอคอมพิวเตอร์

3.23 เมื่อทำการกดปุ่ม Lock Out แล้วตัวโปรแกรมจะทำการล็อคเมาส์และคีย์บอร์ดไม่ให้สามารถใช้งานได้เหมือนตอนเริ่มต้นหากต้องการออกใช้งานอีกครั้งให้กลับไปทำในขั้นตอนที่ 3.11

4. การแก้ปัญหาเบื้องต้น

เมื่อผู้ใช้งานประสบปัญหาในการใช้งานระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมสามารถตรวจสอบแนวทางแก้ไขปัญหาเบื้องต้นได้ดังนี้

ตารางที่ ช.1 การแก้ปัญหาเบื้องต้น

อาการ	สาเหตุและวิธีการแก้ไข
รูดบัตรที่เครื่องแล้วเครื่องบาร์โค้ดไม่ทำการอ่านบัตร	ตรวจสอบพอร์ตที่เชื่อมต่อระหว่างบาร์โค้ดกับเครื่องควบคุมหลักบริเวณ Input ของเครื่องควบคุมหลักว่ามีการเชื่อมต่อถูกต้องหรือไม่
เมื่อเลือกเครื่องแล้วไม่สามารถเข้าใช้งานได้	ตรวจสอบการเชื่อมต่อระหว่างอุปกรณ์ทั้งสามส่วนว่าถูกต้องหรือไม่ หรือลืมเสียบปลั๊กเครื่องควบคุมย่อยทำให้ข้อมูลไม่สามารถส่งไปยังเครื่องคอมพิวเตอร์ได้

5. การดูแลรักษาและข้อควรระวัง

5.1 การดูแลรักษา

1. ตรวจสอบแบตเตอร์รี่แบ็คอัพของวงจรเรียลไทม์ในเครื่องควบคุมหลักเสมอเมื่อไม่ได้ใช้งานเครื่องควบคุมหลักเป็นเวลานานๆ

2. ควรโหลดค่าข้อมูลที่บันทึกเก็บไว้ที่อีอีพรอมจากเครื่องควบคุมหลักมาเก็บไว้ที่คอมพิวเตอร์

3. ตรวจสอบขั้วต่อสายไฟและพอร์ตการเชื่อมต่อระหว่างอุปกรณ์ทั้งสามส่วนให้อยู่ในสภาพพร้อมใช้งานเสมอ

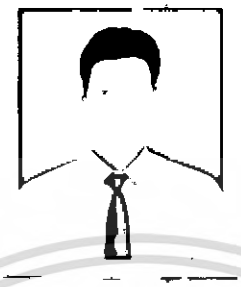
4. หมั่นทำการตรวจและซ่อมบำรุงอุปกรณ์ต่างๆ เป็นระยะเพื่อป้องกันและลดอัตราการเสื่อมสภาพของระบบทั้งหมด

5.2 ข้อควรระวัง

1. ควรศึกษาคู่มือการใช้งานของระบบควบคุมการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรมก่อนการใช้งาน

2. การติดตั้งควรระมัดระวังอย่าให้เครื่องควบคุมในส่วนต่างๆ มีการกระแทกเพื่อป้องกันความเสียหายของระบบบันทึกการใช้งานคอมพิวเตอร์ภาควิชาครุศาสตร์วิศวกรรม

ประวัติผู้แต่ง



ชื่อ-สกุล นายเอกบดินทร์ กลิ่นเกษร

วัน เดือน ปีเกิด 7 เมษายน พ.ศ. 2528

ภูมิลำเนา 550 หมู่ 4 ถนน ร้อยเอ็ด-โพนทอง หมู่บ้านโชคสำรวย
ตำบลเหนือเมือง อำเภอเมือง จังหวัดร้อยเอ็ด 45000

ประวัติการศึกษา

ประถมศึกษา โรงเรียนอนุบาลร้อยเอ็ด จังหวัดร้อยเอ็ด

มัธยมศึกษาตอนต้น โรงเรียนร้อยเอ็ดวิทยาลัย จังหวัดร้อยเอ็ด

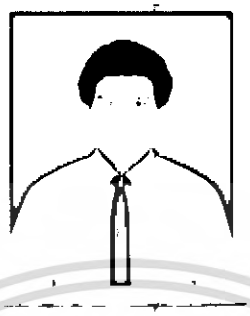
ประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคร้อยเอ็ด จังหวัดร้อยเอ็ด

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคร้อยเอ็ด จังหวัดร้อยเอ็ด

ปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม
ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม สจล.

คติพจน์ อย่ากลัวถ้ายังไม่ได้ลอง

ประวัติผู้แต่ง



ชื่อ-สกุล นายเพ็ญประพรรค์ บรรดาศักดิ์

วัน เดือน ปีเกิด 7 พฤศจิกายน พ.ศ.2527

ภูมิลำเนา 9 หมู่ 6 ตำบลช่อผกา อำเภอชำนิ
จังหวัดบุรีรัมย์ 31110

ประวัติการศึกษา

ประถมศึกษา โรงเรียนบ้านหัวสะพาน

มัธยมศึกษาตอนต้น โรงเรียนสิงหวิทยาคม

ประกาศนียบัตรวิชาชีพ วิทยาลัยการอาชีพนางรอง

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคท่าหลวงซิเมนต์ไทยอนุสรณ์

ปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม
ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม สจล.

คติพจน์ ดวงดาวจะไม่เปล่งแสงเมื่อใกล้ดวงอาทิตย์

ประวัติผู้แต่ง

ชื่อ-สกุล	นางสาวสุดารัตน์ สมนา
วัน เดือน ปีเกิด	21 สิงหาคม พ.ศ. 2527
ภูมิลำเนา	1569/1 ถนน วันลูกเสือ ตำบลเมืองใต้ อำเภอเมือง จังหวัดศรีสะเกษ 33000
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนมารีวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนไกรภักดีวิทยาคม
ประกาศนียบัตรวิชาชีพ	วิทยาลัยการอาชีพศรีสะเกษ
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคศรีสะเกษ
ปริญญาตรี	สาขาวิชาวิศวกรรมโทรคมนาคม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สจล.
คติพจน์	ตามรอยพระราชดำริ