ปีการศึกษา 2537

การจำลองระบบควบคุมภายในอาคาร

โดย

นาย นพปกรณ์ หุ่นสุวรรณ 35102098
นาย พงษ์ศักดิ์ ผดุงกาญจน์ 35102107
นาย สุจินต์ อันพันลำ 35102122

อาจารย์ที่ปรึกษา

ผศ. วิทยา ทิพย์สุวรรณพร

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

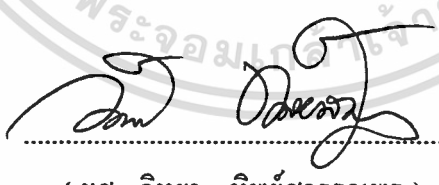
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจำลองระบบควบคุมภายในอาคาร

SIMULATION INSIDE BUILDING CONTROL SYSTEM

ผู้จัดทำ

1. นาย นพปกรณ์ หุ่นสุวรรณ 35102098
2. นาย พงษ์ศักดิ์ ผดุงกาญจน์ 35102107
3. นาย สุจินต์ อันพันลำ 35102122



.....อาจารย์ที่ปรึกษา

(ผศ. วิทยา ทิพย์สุวรรณพร)

การจำลองระบบควบคุมภายในอาคาร

นพปกรณ์ หุ่นสุวรรณ 35102098
พงษ์ศักดิ์ ผดุงกาญจน์ 35102107
สุจินต์ อันพันลำ 35102122

บทคัดย่อ

ในปัจจุบัน นี้มีการก่อสร้างอาคารสูง ๆ ขึ้นมากมายในเขตตัวเมืองซึ่งมีทั้งโรงงานอุตสาหกรรมที่พักอาศัยหรือสำนักงานและภายในอาคารเหล่านี้มีเครื่องใช้ไฟฟ้าเป็นจำนวนมากผลที่ตามมาคือการดูแลรักษาและควบคุมการใช้งานเพื่อประหยัดพลังงานไฟฟ้าและค่าใช้จ่ายรวมทั้งการป้องกันอันตรายอันจะเกิดจากความผิดปรกติของระบบไฟฟ้าอีกด้วย ดังนั้นการควบคุมภายในอาคารจากจุด ๆ เดียวจึงมีความสำคัญต่อการตรวจสอบระบบต่าง ๆ และสามารถแจ้งเหตุฉุกเฉินได้ทันท่วงที และจึงเป็นที่มาของโครงงานการจำลองระบบควบคุมภายในอาคารโดยจะมีเครื่องไมโครคอนโทรลเลอร์ 256 ตัว แต่ละตัวมีขีดความสามารถควบคุมได้ 10 จุด และอีก 2 จุด รับสัญญาณ Input เพียงอย่างเดียวสรุปแล้วสามารถควบคุมได้ 2560 จุด และรับสัญญาณ Input 512 จุด โดยไมโครคอนโทรลเลอร์จะถูกติดตั้งไว้ตามจุดต่างๆ ภายในอาคารแล้วรายงานผลการตรวจสอบผ่านระบบสื่อสารข้อมูลแบบ RS-485 มายังเครื่อง PC ซึ่งเป็นศูนย์กลางเครื่อง PC จะรายงานผลให้ทราบและสามารถส่งคำสั่งผ่าน PC ไปให้ไมโครคอนโทรลเลอร์ ปิด-เปิด อุปกรณ์ไฟฟ้าทุกชิ้นภายในอาคารได้อย่างรวดเร็ว และสามารถใช้เป็นเครื่องแจ้งเหตุอัคคีภัยและโจรภัยหรือความผิดปรกติของระบบใดๆที่อาจจะทำให้เกิดความเสียหายได้อีกด้วย

SIMULATION INSIDE BUILDING CONTROL SYSTEM

Noppakorn Hoonsuwan

Pongsak Padungkarn

Sujin Anphanlam

Assistant Professor Vittaya Tipsuwannaporn Advisor

Abstract

In present have the high buildings in country. There have factorys,homes,offices and inside of that building have electric equipments. The result of that is the mantainance and control the electric equipments for saving the electric power, cost and protection of dangerous that is occured from abnormal of electric system. The project Simulation Inside Building Control System is happened for checking and controlling the electric equipments. That have microcontroller 256 which each of them can control 10 points and receive input signal 2 points, then to abridge is control 2560 points and receive input signal 512 points, which microcontroller is installed at every place that you want to control and check. Then the microcontroller is show of checking through communication system of RS-485 to personal computer which is the center of communication system. Then the personal computer is show the result and send the command through personal computer to command the microcontroller is open-close the electric equipments in the building is very fast and used to inform of the fire and the theft or the abnormal of every system that bring to dangerous.

สารบัญ หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

ความเป็นมาและความสำคัญของปัญหา

นับตั้งแต่มาร์โคนี ได้ทำการทดลองส่งสัญญาณวิทยุข้ามมหาสมุทรแอตแลนติกได้สำเร็จ การสื่อสารด้วยอุปกรณ์อิเลคทรอนิกส์ก็ได้พัฒนาขึ้นมาเป็นลำดับ การสื่อสารได้เริ่มจากการส่งรหัสมอร์ส (เป็นรหัสตัวอักษร) โดยใช้รหัสจุดขีดเพื่อแทนข้อความ รหัสดังกล่าวใช้ได้กันมาถึงปัจจุบัน หลังจากนั้นกิจการสื่อสารก็ได้เจริญรุดหน้า มีการใช้โทรเลขก่อนในยุคแรก จากโทรเลขก็เป็นวิทยุกระจายเสียง โทรพิมพ์ โทรภาพ โทรสำเนา ครั้นเมื่อถึงยุคสมัยของคอมพิวเตอร์ความจำเป็นของการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ก็มีมากขึ้นการพัฒนาการสื่อสารข้อมูลทำให้การใช้งานคอมพิวเตอร์มีประโยชน์มากมายมหาศาล และมีอิทธิพลต่อความเป็นอยู่ของมนุษย์ในปัจจุบัน และ อนาคตกิจการหลายอย่างได้อาศัยความสะดวกสบายของการสื่อสารข้อมูลเช่น กิจการธนาคาร กิจการการบิน การท่องเที่ยว การโรงแรม ฯลฯ จนในที่สุดมนุษย์จะเข้ามาอยู่ในยุคของการใช้ข่าวสารข้อมูลอย่างเต็มที่ จะมีอุปกรณ์เครื่องมือเครื่องใช้ต่าง ๆ ที่เกี่ยวข้องกับการสื่อสารข้อมูลอีกเป็นจำนวนมาก ดังนั้นในปัจจุบันความสำคัญของการสื่อสารจึงเข้ามามีบทบาท และมีความสำคัญต่อชีวิตและสังคมเป็นอย่างมาก

ในอดีตในระบบการสื่อสารส่วนใหญ่จะเป็นแบบอนาล็อก (Analog) ซึ่งระบบการสื่อสารแบบอนาล็อกนี้ ทำให้ข้อมูลหรือข่าวสาร ที่ใช้ไปด้วยนั้นเกิดการผิดพลาดได้ง่าย นอกจากนั้นยังเป็นการสิ้นเปลืองอย่างอื่นอีกมาก เช่น จะต้องมีสถานีทวนสัญญาณ (Repeater) ตามระยะทางเพื่อเพิ่มกำลังการส่งให้ไปถึงจุดหมาย ถ้าระยะทางไกลมากก็ต้องใช้จำนวน Repeater มากขึ้นด้วย เป็นการสิ้นเปลืองแต่มาในปัจจุบันระบบการสื่อสารได้มีการพัฒนาให้มีประสิทธิภาพสูงขึ้น อีกทั้งข่าวสาร และข้อมูลที่ได้ใช้ในการสื่อสารมีจำนวนมาก และสลับซับซ้อนขึ้น ดังนั้นการรับส่งข้อมูลจึงต้องมีความถูกต้องประหยัดและรวดเร็วขึ้นด้วยซึ่งวิธีการหนึ่งที่ช่วยได้ก็คือ การรับส่งข้อมูลด้วยระบบ Digital นี้สามารถที่จะนำไปต่อกับวิธีการสื่อสารแบบเก่า เช่น ระบบโทรทัศน์ ระบบไมโครเวฟได้ ซึ่งทำให้เป็นการประหยัดไม่ต้องมีการสร้างสถานี (Station) หรือ อุปกรณ์สำคัญเพิ่มขึ้นแต่อย่างใด นอกจากนี้ความผิดเพี้ยน (Distortion) ของข้อมูลหรือข่าวสารที่ใช้ส่งด้วยระบบ Digital

2 นี้จะมีความผิดเพี้ยนน้อย เมื่อเทียบกับระบบ Analog เพราะสัญญาณ Digital นั้นมีเพียงสัญญาณเปิดกับปิด หรือ .1. กับ .0. เท่านั้น ซึ่งไม่สลับซับซ้อนเหมือนกับสัญญาณ Analog ทำให้เราสามารถตรวจสอบความผิดเพี้ยนได้ง่ายกว่า จึงสามารถสร้างสัญญาณ Digital ให้มีลักษณะเหมือนเดิมได้ง่ายกว่าการสร้างสัญญาณ Analog คอมพิวเตอร์เป็นอุปกรณ์อีกชนิดหนึ่งที่เข้ามามีบทบาทสำคัญต่อวงการธุรกิจวิทยาศาสตร์วงการแพทย์และวงการอื่น ๆ อีกมาก การติดต่อกับคอมพิวเตอร์นั้นจะใช้ การติดต่อด้วยระบบดิจิตอลเป็นส่วนใหญ่การส่งข้อมูลนั้นอาจจะเป็นข่าวสาร ข้อมูลหรือสัญญาณควบคุมซึ่งส่วนใหญ่แล้วจะได้จากคอมพิวเตอร์ ดังนั้นเมื่อเครื่องคอมพิวเตอร์ เครื่องหนึ่ง ๆ จะรับและส่งข้อมูลซึ่งกันและกันก็จะต้องทำการติดต่อด้วยระบบ Digital นั่นเองโดยการรับและส่งข้อมูลนั้นจะต้องมีมาตราฐาน เพื่อให้สามารถใช้ได้กับเครื่องคอมพิวเตอร์ได้ทุก ๆ เครื่องที่ใช้กันอยู่ในปัจจุบันได้แก่มาตราฐาน RS-232, RS-232C, RS-422, RS-433 และ RS-485 ซึ่งมาตราฐานเหล่านี้สามารถนำไปต่อกับระบบโทรศัพท์ได้ง่าย โดยผ่าน MODEM ทำให้เป็นการประหยัด และสะดวกขึ้น

บทบาทสำคัญของการสื่อสารข้อมูลในยุคปัจจุบันเป็นสิ่งที่มีประโยชน์อย่างมาก ไม่ว่าจะเป็นในเรื่องของการจัดเก็บข้อมูลเป็นไปได้ง่าย สื่อสารได้สะดวก รวดเร็ว ความถูกต้องของข้อมูล เพราะโดยปกติการส่งข้อมูลด้วยสัญญาณอิเลคทรอนิคส์จากจุดหนึ่งไปยังอีกจุดหนึ่งด้วยข้อมูลดิจิตอล วิธีการรับส่งนั้นจะมีการตรวจสอบสภาพของข้อมูล หากข้อมูลผิดพลาดก็จะมีการรับรู้และพยายามหาทางให้ข้อมูลที่ถูกต้อง โดยการข้อร้องให้มีการส่งใหม่ หรือกรณีที่ผิดพลาดไม่มากนัก ฝ่ายผู้รับอาจมีวิธีทางซอฟต์แวร์ของตัวเองที่แก้ไขข้อมูลให้ถูกต้องได้ ในเรื่องของความเร็วของการทำงาน ความเร็วของระบบก็เป็นที่สะดวกสบายต่อผู้ใช้เป็นอย่างยิ่ง

จากที่กล่าวมาจะเห็นว่าการสื่อสารด้วยระบบ Digital นั้น มีความสำคัญมากขึ้นในปัจจุบันนี้ เพราะเป็นการติดต่อระหว่างเครื่องคอมพิวเตอร์ด้วยกัน ดังนั้นจึงเป็นการสมควรที่จะต้องมีการศึกษา และพัฒนาการสื่อสารระบบ Digital ให้มีประสิทธิภาพสูงขึ้น

1.2 <u>วัตถุประสงค์ของโครงงาน</u>

1.2.1 สามารถนำระบบไปใช้ควบคุมเครื่องใช้ไฟฟ้าในระยะไกลหรือในอาคารได้โดยผ่านศูนย์กลางด้วยเครื่องคอมพิวเตอร์ IBM PC

1.2.2 สามารถนำมาใช้กับระบบ Sensor เป็น Switch

1.2.3 สามารถประยุกต์ในระบบรักษาความปลอดภัยได้

1.2.4 สามารถประยุกต์ในระบบเตือนภัยได้

1.3 <u>ขอบเขตของโครงงาน</u>

1.3.1 สร้างวงจร RS-485 Interface card เพื่อติดตั้งบนเครื่อง IBM PC

1.3.2 สร้างเครื่อง Microcontroller ด้วย CPU เบอร์ 8031 ที่สามารถรับส่งข้อมูลกับ IBM PC ได้ด้วยมาตรฐาน RS-485

1.3.3 พัฒนาโปรแกรมภาษา C บนเครื่อง IBM PC เพื่อรับส่งข้อมูลกับเครื่อง Microcontroller

1.3.4 พัฒนาโปรแกรมภาษา C บนเครื่อง IBM PC เพื่อสร้าง HDLC Frame Format และแปลงข้อมูลเป็นข้อความแสดงบนจอภาพ

1.3.5 พัฒนาโปรกแกรมภาษา C บนเครื่อง IBM PC เพื่อติดต่อกับผู้ใช้เป็นแบบ Pull Down Menu

1.3.6 พัฒนาโปรแกรมภาษา Assembly ของ CPU 8031เพื่อใช้ในการส่งข้อมูลกับ IBM PC ในลักษณะ HDLC Frame Format

1.3.7 สร้างวงจร Simulate ระบบที่สร้างขึ้น

1.4 ขั้นตอนและวิธีการดำเนินงาน

1.4.1 ศึกษารายละเอียด การทำงานของระบบเดิม

1.4.2 ศึกษารายละเอียดของ Hardware ในระบบเดิม

1.4.3 หาแนวทางปรับปรุง Hardware ให้ดียิ่งขึ้น

1.4.4 ศึกษารายละเอียดของ Software ของระบบเดิม

1.4.5 ศึกษารายละเอียดเกี่ยวกับการทำงานของ Protocol HDLC และ Format ของการรับส่งรวมถึง Software ของไมโครโปรเซสเซอร์ 8031

1.4.6 ศึกษารายละเอียดเกี่ยวกับอุปกรณ์ที่ใช้ในการสื่อสารแบบอะซิงโครนัส พอร์ทอนุกรมต่าง ๆ

1.4.7 ศึกษาและออกแบบสร้าง Microcontroller 8031 รวมถึงการเขียน Software ควบคุมการรับส่งข้อมูล และออกแบบสร้างวงจร RS-485 Interface Card กับ PC

1.4.8 ทำการทดสอบการทำงานของ Microcontroller และสร้างวงจร RS-485 Interface Card กับ PC

1.4.9 ทำการทดสอบ Software ของระบบ Protocol HDLC

1.4.10 นำ Microcontroller พร้อมกับวงจร RS-485 Interface Card ของ PC มาประกอบการทำงานกับ Software ของ Protocol HDLC เพื่อการรับส่งข้อมูลระหว่าง PC กับ Microcontroller

1.4.11 ตรวจสอบและแก้ไขข้อบกพร่องในส่วนของ Microcontroller วงจร RS-485 Interface Card, Software ของ Protocol HDLC

1.4.12 สรุปผลการดำเนินงาน ข้อเสนอแนะ ปัญหาของการดำเนินงาน พิมพ์รายงานตรวจสอบและแก้ไข เสนอรายงานการดำเนินงานและผลงาน

1.5 <u>ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน</u>

1.5.1 สามารถนำระบบนี้ ไปทำการควบคุมและตรวจสอบการทำงานของอุปกรณ์ไฟฟ้าในโรงแรม อาคาร ได้โดยผ่านทางศูนย์กลางด้วยเครื่อง IBM PC

1.5.2 ใช้เป็นเครื่องต้นแบบ สำหรับการพัฒนาไปสู่ระบบการควบคุมและตรวจสอบระยะไกลแบบอื่น ๆ ได้

1.5.3 สามารถนำไปใช้เป็นเครื่องอำนวยความสะดวกภายในอาคารได้โดยการควบคุมระยะไกล

1.5.4 สามารถเครือข่ายได้เพื่อขยายความสามารถในการติดต่อรับ-ส่ง ข้อมูล

1.5.5 สามารถพัฒนาเทคโนโลยีทางด้านป้องกันความปลอดภัย

บทที่ 2

ไมโครโปรเซสเซอร์ MCS-51 กับการสื่อสารข้อมูลและระบบโปรโตคอล HDLC

2.1 คำนำ

ไมโครโปรเซสเซอร์ตระกูล MCS-51 เป็นไมโครโปรเซสเซอร์แบบซิงเกิลชิพ เพราะภายในชิพประกอบด้วยหน่วยความจำ ส่วนการคำนวณ ส่วนสร้างสัญญาณนาฬิกา พอร์ทแบบขนานวงจรนับเวลาและตั้งเวลา พอร์ทแบบอนุกรม ทำให้มีการใช้งานอย่างกว้างขวางเพราะในการออกแบบระบบจะไม่ต้องใช้อุปกรณ์ภายนอกมาก จึงประหยัดทั้งพลังงานที่ใช้และการลงทุนในระบบไมโครคอมพิวเตอร์ตระกูล MCS-51 มีข้อดีเหนือกว่าไมโครโปรแกรมได้ถึง 64 กิโลไบท์ และมีหน่วยความจำสำหรับการใช้เก็บข้อมูลแยกออกมาอีก 64 กิโลไบท์ ได้หน่วยความจำภายในแบบแรนดอมแอดเซส (RAM) ขนาด 256 ไบท์ จะมีบางส่วนที่โปรแกรมจะสามารถติดต่อทีละบิทและแบบไบท์ พอร์ทขนาดภายในสามารถใช้งานได้ทั้งแบบอินพุท และเอาท์พุท รวมถึงพอร์ท 3 สามารถใช้งานในฟังก์ชั่นอื่น ๆ ได้ วงจรนับเวลา และตั้งเวลาภายใน 2 ชุด ขนาด 16 บิท (Timer o, Timer 1) สามารถโปรแกรมเพื่อการใช้งานแยกกันได้อย่างอิสระสามารถใช้โปรแกรมกำหนดอัตราการส่งข้อมูล (Baudrate) ของพอร์ทอนุกรมซึ่งเป็นฟังก์ชั่นของพอร์ท 3 สามารถเลือกให้ทำงานจากโปรแกรมที่อยู่ภายในชิพ หรือภายนอกชิพ โดยการต่อขาหนึ่งเข้ากับลอจิก 0 หรือลอจิก 1 อินเทอร์รัพท์ภายนอก 2 อินเทอร์รัพท์ และภายใน 3 อินเทอร์รัพท์ ทำให้ใช้งานควบคุมได้อย่างมีประสิทธิภาพ

ซิงเกิลบอร์ด 8031 ถูกออกแบบขึ้นให้ผู้ใช้สามารถศึกษาการทำงานของ 8031 ให้อย่างเต็มประสิทธิภาพการใช้งานของ 8031 ผู้ใช้สามารถใช้ทั้งพอร์ทและพังก์ชั่นภายในได้อีกทั้งสามารถนำเข้าไปใช้ในระบบควบคุมได้สะดวกกว่าซิงเกิลบอร์ดแบบอื่น ๆ

2.2 ไมโครโปรเซสเซอร์ MCS-51

2.2.1 การทำงานภายใน MCS-51

ไมโครโปรเซสเซอร์ในตระกูล MCS-51 ซึ่งเป็นไมโครโปรเซสเซอร์ที่มีขนาด 8 บิท ประกอบด้วยไมโครโปรเซสเซอร์เบอร์ต่าง ๆ ทุก ๆ เบอร์จะมีสถาปัตยกรรมพื้นฐานเหมือนกับรูปที่ 2.1 แต่เดิม 8031 ถูกสร้างด้วยวิธี HMOSII จึงมีชื่อเป็น MCS ไมโครโปรเซสเซอร์ตระกูล 51 เท่านั้น ถึงแม้ว่าจะมีอยู่ด้วยกันหลายเบอร์ แต่เราก็จะเรียกว่าเป็น

.8031. ซึ่งเราจะใช้ชื่อเรียกว่า 8031 นี้โดยตลอด เมื่อหมายถึงไมโครโปรเซสเซอร์ ตระกูล 51 ส่วนเบอร์ 8032 และ 8052 มีหน่วยความจำภายในเพิ่มขึ้นและมีวงจรนับตั้งเวลาขนาด 16 บิทเพิ่มขึ้น โดยวงจรนับและตั้งเวลาสามารถใช้เป็นวงจรนับ วงจรตั้งเวลาและเป็นตัวกำหนดอัตราการส่งข้อมูลทางอนุกรม (Serial Port)

คุณสมบัติสำคัญของตระกูล 51 นี้ คือ

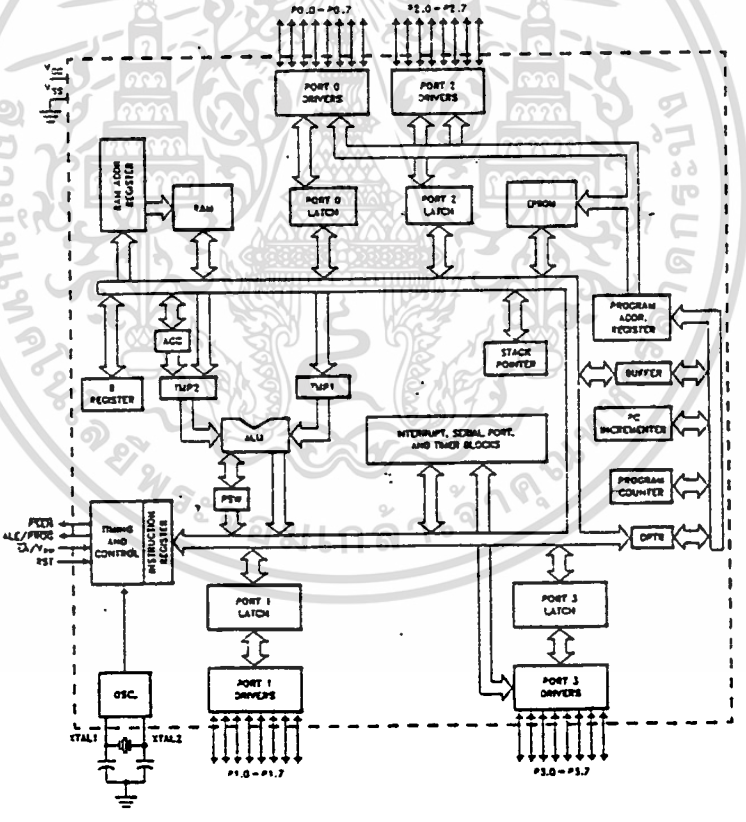
- ไมโครโปรเซสเซอร์แบบ 8 บิท
- มีวงจรกำเนิดสัญญาณนาฬิกาภายใน
- 32 อินพุท/ เอาท์พุท
- หน่วยความจำภายนอกสำหรับเก็บข้อมูล 64 กิโลไบท์
- หน่วยความจำภายนอกสำหรับเก็บโปรแกรม 64 กิโลไบท์
- วงจรนับ ตั้งเวลา 16 บิท 2 ชุด สำหรับ 8032/8052
- 5 อินเทอร์รัพท์ (6 สำหรับ 8032/8052) สามารถจัดลำดับความสำคัญของการอินเทอร์รัพท์ได้ 2 ระดับ
- ส่งข้อมูลอนุกรมแบบ 2 ทิศทางได้
- สามารถประมวลผลตรรกศาสตร์
- มี RAM ภายในขนาด 128 ไบท์สามารถอ้างได้ทั้งแบบบิทและแบบเป็นไบท์

การติดต่อกับหน่วยความจำภายนอกมี 2 แบบ ซึ่งมีการทำงานดังรูป 2.2 คือ ติดต่อกับหน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับข้อมูล การติดต่อกับหน่วยความจำสำหรับโปรแกรมภายนอกจะใช้สัญญาณ PSEN (Program Store Enable) เป็นสัญญาณสโตรป (Strobe) สัญญาณ RD และ WR จะใช้เป็นสัญญาณสโตรป (STROBE) สำหรับการติดต่อกับหน่วยความจำสำหรับข้อมูล การอ่านโปรแกรมจากหน่วยความจำภายนอกจะใช้แอดเดรส 16 บิทเสมอ ส่วนการติดต่อกับหน่วยความจำสำหรับข้อมูลจะใช้แอดเดรสได้ทั้งแบบ 16 บิท (เช่นคำสั่ง MOVC DPTR) หรือให้ แอดเดรส 8 บิท (เช่น คำสั่ง MOVX Ri) เมื่อทำการติดต่อกับหน่วยความจำที่พอร์ท 2 จะไม่เปลี่ยนแปลง ในขณะมีการติดต่อกับหน่วยความจำภายนอกทั้งการติดต่อแบบ 8 บิท และ 16 บิทไบท์ต่ำของแอดเดรส จะเป็นแบบมัลติเพลกซ์กับข้อมูลของพอร์ท 0 สัญญาณ ADDR/DATA จะทำให้ FET ของเอาท์พุทพอร์ท 0 ทำงาน ดังนั้นการใช้พอร์ท 0 ลักษณะนี้จึงไม่ต่อ PULL UP วงจร LATCH ภายนอกจะเก็บค่าของแอดเดรสไบท์ต่ำ โดยใช้สัญญาณ ALE เป็นตัวกำหนด โดย ค่าของแอดเดรสไบท์ต่ำจะมีค่าคงที่ขณะสัญญาณ ALE เปลี่ยนจาก 1 เป็น 0 ดังนั้นในการเขียนข้อมูลไปยังหน่วยความจำนั้น ข้อมูลจะถูกส่ง

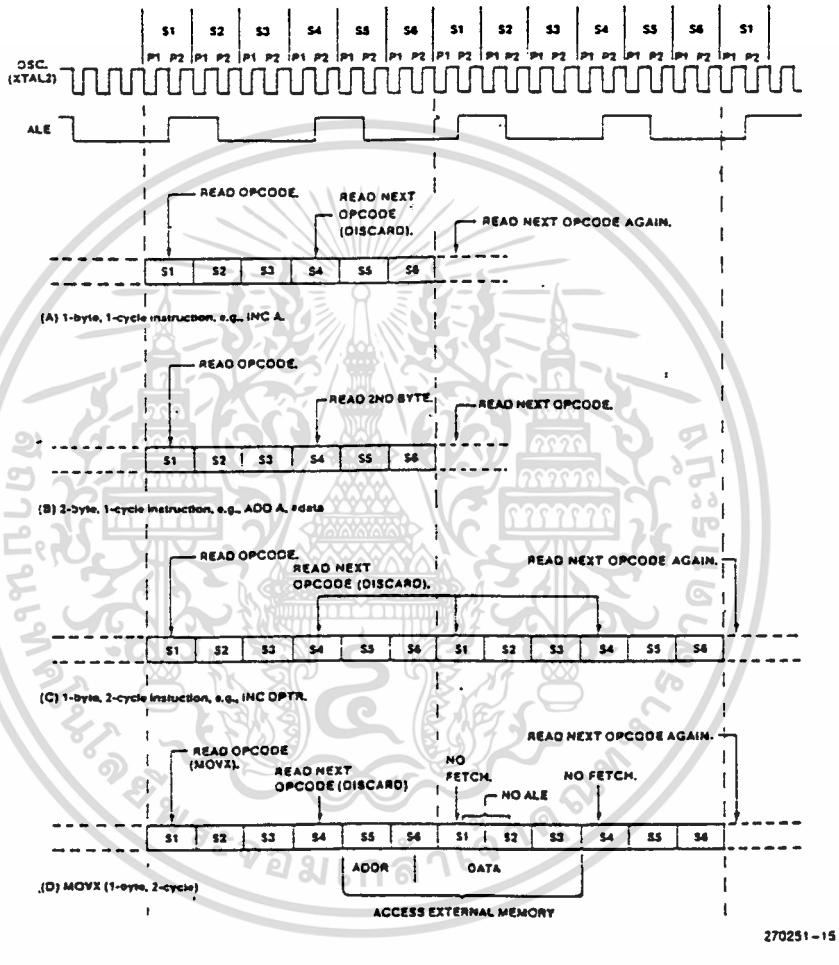
ออกไปทางพอร์ท 0 ก่อนจะมีสัญญาณ WR เป็น 0 และจะคงค่านั้นไว้จนกว่าสัญญาณ WR จะกลับเป็น 1

ในการอ่านข้อมูลจากหน่วยความจำภายนอกนั้นข้อมูลจะต้องปรากฏที่พอร์ท 0 ก่อน สัญญาณ RD จะเป็น 0 เสมอ

ระหว่างการติดต่อกับหน่วยความจำภายนอกนั้น CPU จะส่งข้อมูล OFFH ไปยังพอร์ท 0 Latch ทำให้ข้อมูลที่พอร์ท 0 เปลี่ยนไป



รูปที่ 2.1 แสดงโครงสร้างภายใน MCS-51



รูปที่ 2.2 แสดงการติดต่อกับหน่วยความจำภายนอก MCS-51

- EA เป็น 0

- เมื่อ EA เป็น 1 จะอ่านโปรแกรมจากหน่วยความจำภายนอกเมื่อโปรแกรมเคาร์เตอร์ (PC) มีค่ามากกว่า OFFFH หรือ 1FFFH สำหรับ 8052 ดังนั้นในรุ่นที่ไม่มี ROM ภายในจึงต้องต่อ EA ไว้ที่ 0 เพื่อให้มีการอ่านโปรแกรมจากหน่วยความจำภายนอกเสมอ

1) Serial Interface

Serial พอร์ทเป็นแบบสองทาง (Full Duplex) หมายความว่าสามารถทั้งส่งและรับได้พร้อมกันในการรับจะมีบัฟเฟอร์ซึ่งสามารถให้รับข้อมูลในไบท์ที่สอง โดยไบท์แรกยังไม่ได้ถูกอ่านเข้าไปจากรีจิสเตอร์ตัวรับ แต่ถ้าเมื่อไบท์ที่สองรับเข้ามาครบแล้ว ไบท์ที่ 1 ยังไม่ได้ถูกอ่านเข้าไป แล้วข้อมูลในไบท์ที่ 1 จะหายไปจากรีจิสเตอร์ SBUF จะใช้เป็นบัฟเฟอร์สำหรับทั้งรับและส่งข้อมูล การเขียนข้อมูลไปยัง SBUF จะเป็นการโปลดค่าไปทำการส่ง และโครงสร้างภายในแล้วรีจิสเตอร์ SBUF ของการส่งและรับข้อมูลเป็นคนละตัวกันแต่เรียกเหมือนกัน CPU จะรู้เองว่าเรียกใช้ SBUF สำหรับการส่งหรือการรับ โดยถ้าเป็นการอ่านค่า SBUF จะเป็นการรับข้อมูล ส่วนถ้าเป็นการอ่านค่า SBUF จะเป็นการรับข้อมูล ส่วนถ้าเป็นการอ่านค่า SBUF จะเป็นการส่งข้อมูล SERIAL พอร์ท สามารถทำงานได้ใน 4 โหมด

- โหมด 0 ข้อมูลอนุกรมจะส่งผ่าน RXD และ TXD โดยการเลื่อนด้วยอัตราคงที่เท่ากับ 1/12 ของความถี่ออสซิเลลเตอร์ ข้อมูลที่ส่งและรับเป็นแบบ 8 บิท

- โหมด 1 การส่งข้อมูลผ่าน RXD และ TXD เป็นแบบ 10 บิท ประกอบด้วย 1 Start Bit, Data 8 Bit (บิทต่ำสุด) และ 1 Stop Bit ในตอนรับบิท 10 จะไปอยู่ใน RBB ใน SFR ชื่อ SCON อัตราการส่งสามารถโปรแกรมได้

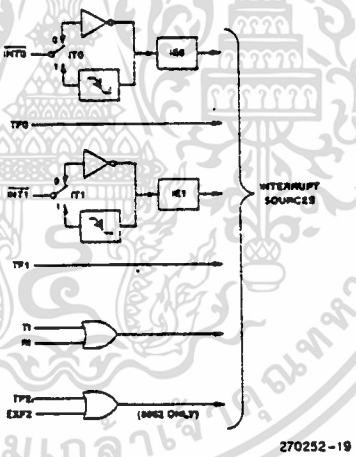
- โหมด 2 การส่งและรับข้อมูลเป็นแบบ 11 บิท คือ 1 Start Bit Date 8 Bit (บิท 0-7) บิทที่สามารถกำหนดได้ และ 1 STOP BIT ในการส่งบิทที่ 9 (TB 8 ใน SCON) สามารถกำหนดค่าเป็น 1 หรือ 0 ก็ได้ เช่น เอาพาริตี้บิท (Parity Bit, P ใน PSW) ย้ายเข้าไปที่ TBB หรือ ในตอนรับบิทที่ 9 จะถูกส่งเข้าไปใน RSS ของ SCON ขณะที่ Stop บิทจะไม่ถูกเก็บไว้อัตราการส่งสามารถโปรแกรมเป็น 1/32 หรือ 1/64 ของความถี่ออสซิเลเตอร์

- โหมด 3 การทำงานของโหมดนี้จะเหมือนกับโหมด 2 ยกเว้นเราสามารถโปรแกรมอัตราการส่งได้

ในทุกโหมดการส่งข้อมูลจะเริ่มจากคำสั่งใดก็ได้ที่มีการส่งข้อมูลไปยัง SBUF ในการรับข้อมูลจะต่างกันเล็กน้อย คือในโหมด 0 การรับข้อมูลจะเริ่มโดย RI = 1 เมื่อมี START บิทเข้ามา

2) Interrupt

สัญญาณอินเทอร์รัพท์จากภายนอก INTO และ INT1 สามารถทำงานได้ทั้งแบบระดับ (Level) และแบบการเปลี่ยนระดับ (Transition) ขึ้นกับ IT1 ใน TCON แฟลกที่ควบคุมสัญญาณอินเทอร์รัพท์นี้คือ IEO และ IE1 ใน TCON เมื่อเกิดการดินเทอร์รัพท์จากภายนอกแฟลกที่สร้างอินเทอร์รัพท์ จะถูกเคลียร์ด้วยฮาร์ดแวร์เมื่อมีตัวชี้ตำแหน่งโปรแกรมตอบสนองอินเทอร์รัพท์ ถ้าเป็นการอินเทอร์รัพท์แบบเปลี่ยนสภาวะเท่าอินเทอร์รัพท์ของ Time 0 และ Timer 1 จะสร้างโดย TE1 ซึ่งถูกเซทโดย การนับถือ 255 ของรีจิสเตอร์นั้น ๆ เมื่อเกิดอินเทอร์รัพท์ของ Timer แฟลกที่เกิดขึ้นถูกเคลียร์โดยฮาร์ดแวร์ภายในเมื่อมีการชี้ไปยังตำแหน่งของโปรแกรมตอบสนองอินเทอร์รัพท์



รูปที่ 2.3 แสดงสัญญาณ INTERRUPT ของ MCS-51

อินเทอร์รัพท์ SERIAL พอร์ท จะสร้างโดยการ OR ของ RI และ TI แฟลกทั้ง 2 ตัวนี้จะไม่ถูกเคลียร์ตัวฮาร์ดแวร์ เมื่อมีการชี้ตำแหน่งโปรแกรมตอบสนองอินเทอร์รัพท์ในโปรแกรมตอบสนองอินเทอร์รัพท์จะต้องรู้ว่าเป็น RI หรือ TI แล้วจึงเคลียร์ด้วยซอฟแวร์

ทุกบิทที่สร้างอินเทอร์รัพท์สามารถเซทหรือเคลียร์ไว้ด้วยซอฟแวร์ เหมือนกับทำด้วยฮาร์ดแวร์ดังนั้นอินเทอร์รัพท์สามารถสร้างหรือยกเลิกได้โดยกำหนดในซอฟแวร์แต่ละอินเทอร์รัพท์สามารถอนาเบิลแยกจากกันโดยอิสระด้วยการกำหนดใน SFR ชื่อ IE (รูป 2.4) ใน IE จะมีบิทหนึ่งคือ EA ซึ่งจะมีดิสเอเบิลทั้งหมดพร้อมกันได้

2.2.2 โครงสร้างหน่วยความจำ แอดเดรสซิงโหมดและตรรกศาสตร์

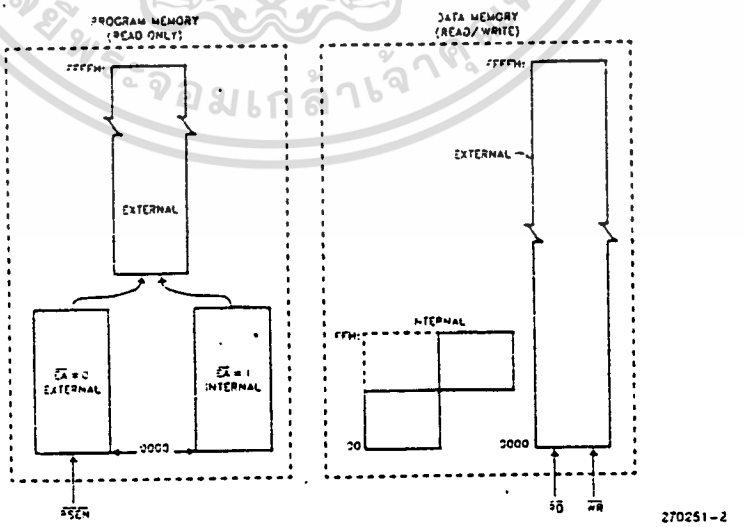
โครงสร้างของตระกูล 51 สามารถใช้งานหน่วยความจำภายใน และภายนอกได้อย่างดี วิธีการแอดเดรสต่าง ๆ จะทำให้คำสั่งมีประสิทธิภาพ

1) การจัดหน่วยความจำ 8031 มีหน่วยความจำพื้นฐานได้ดังนี้

- หน่วยความจำสำหรับโปรแกรม 64 กิโลไบท์
- หน่วยความจำสำหรับข้อมูลภายนอก 64 กิโลไบท์
- หน่วยความจำภายในแบบ RAM 256 ไบท์ (ใน 8021/8052 มีถึง 380 ไบท์)

ก. Program Memory Address Space

หน่วยความจำ 64 กิโลไบท์ สำหนับโปรแกรมจะมีทั้งภายในและภายนอกถ้าขา EA ต่อไว้ที่ High 8031 จะทำงานจากโปรแกรมภายใน นอกเสียจากจะมีการเรียกโปรแกรมจาก OFFFFH (1FFFH ใน 8052) ตำแหน่ง 1000H ถึง OFFFFH (2000H ถึง OFFFFH ใน 8052) จะอยู่ภายนอกเสมอ ถ้าขา EA ต่อไว้ที่ LOW ใน 8051 จะอ่านโปรแกรมจากภายนอกเท่านั้น ทั้งสองแบบจะใช้ PC ขนาด 18 บิท



รูปที่ 2.4 แสดง MEMORY MAP ของ MCS-5

ตารางแสดงที่ 2.1 แสดงตำแหน่งการเก็บโปรแกรมตอบสนองการอินเตอร์รัพท์

Source	Address
External Interrupt 0	0003H
Timer 0 Overflow	000BN
External Interrupt 1	0013H
Timer 1 Overflow	001BN
Serial Port	0023H
Timer 2 Overflow/TZEX	002BH
Negative Transition	

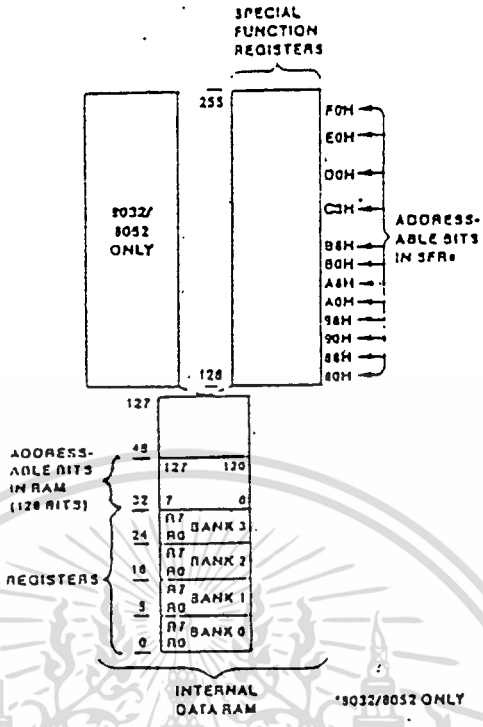
ข) Data Memory Address Space

Data Memory Address Space ประกอบด้วยตัวหน่วยความจำภายในและภายนอก จากการติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง MOVX เท่านั้น

หน่วยความจำ (RAM) ภายในจะแบ่งออกเป็น 3 ส่วน 128 ไบท์ต่ำของ RAM (128 ไบท์สูงของ RAM จะติดต่อได้เฉพาะใน 8032/8052 เท่านั้น) และ 128 ไบท์ของแอดเดรสโหมดต่างกัน ซึ่งจะกล่าวถึงต่อไป

รูปที่ 2.5 แสดง Address mapping ของหน่วยความจำสำหรับข้อมูลภายในมีรีจิสเตอร์อยู่ 4 ชุด (BANK) ชุดละ 8 รีจิสเตอร์ในตำแหน่ง 0 ถึง 31 ใน RAM ชุดล่างการใช้งานแต่ละขณะจะใช้ได้เพียงชุดเดียวเท่านั้น (กำหนดใน PSW) 16 ไบท์ต่อมา (ตำแหน่ง 32 ถึง 47) จะใช้สำหรับ 128 บิทแอดเดรสเอเบิลเป็นช่วงของหน่วยความจำซึ่งสามารถใช้แบบบิท-แอดเดรส ได้ดังรูปที่ 2.5

การอ่านข้อมูลจากตำแหน่งหน่วยความจำภายใน ซึ่งไม่ได้ใช้งานจะได้ข้อมูลไม่แน่นอน



รูปที่ 2.5 แสดง Address Mapping ของหน่วยความจำสำหรับข้อมูลภายใน

2) แอดเดรสซิ่งโหมด 8051 มี 5 แอดเดรสซิ่งโหมด

- รีจิสเตอร์ (Register Addressing)
- ไดเรค (Direct Addressing)
- รีจิสเตอร์ อินไดเรค (Indirect Register Addressing)
- อินมีเดียสท์ (Immediate Addressing)
- เบส-รีจิสเตอร์ร่วมกับ อินเดค-รีจิสเตอร์ อินไดเรค (Base Register+Index-register Indirect Addressing)

ก) รีจิสเตอร์ - แอดเดรสซิ่ง

รีจิสเตอร์แอดเดรสซิงจะทำงานผ่านชุดรีจิสเตอร์ (R0-R7) ที่กำลังใช้งานอยู่ 3 บิทล่างของชุดคำสั่ง จะบอกว่าเป็นการใช้แอดเดรสใด ACC, B, DPTR และ CT, Boolean Accumulator สามารถใช้งานโดยการอ้างแอดเดรสเหมือน เป็นรีจิสเตอร์ไดเรคแอดเดสซิ่ง

ข) ไดเรคแอดเดรสซิง

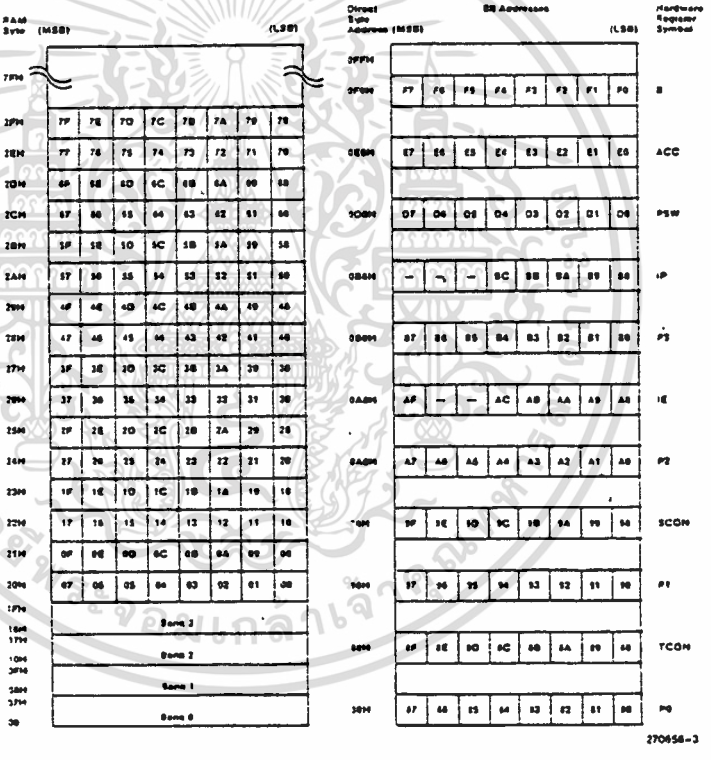
ไดเรคแอดเดรสซิงเป็นวิธีเดียวที่จะใช้ติดต่อกับ SFR128 ไบท์ต่ำของ RAM ภายใน

ค) รีจิสเตอร์ อินไดเรด แอดเดรสซิ่ง

รีจิสเตอร์ อินไดเรดแอดเดรสซิ่ง จะใช้ค่าของ RO หรือ RI (ของชุดที่กำลังใช้งาน) เป็นตัวชี้ไปยังตำแหน่งใน 258 ไบท์ คือ 128 ไบท์ล่างของ RAM ไบท์บนของ RAM ภายใน (ใน8032/8052 เท่านั้น) หรือใน 256 ไบท์ ของหน่วยความจำภายนอกถึง 64 กิโลไบท์จะต้องใช้ Data Pointer แบบ 16 บิทเท่านั้น

การทำงานของคำสั่ง PUSH และ POP จะเป็นแบบ รีจิสเตอร์อินไดเรคแอดเดรสซิง Stack Pointer อาจชี้ตำแหน่งใดก็ได้ในหน่วยความจำภายใน

ตารางที่ 2.2 แสดงหน่วยความจำที่สามารถติดต่อได้ด้วยแต่ละวิธีแอดเดรสซิ่งโหมด



ง) อิมมีเดียท แอดเดรสซิ่ง

อิมมีเดียทแอดเดรสซิ่ง จะมีค่าคงที่ปรากฏอยู่ในรหัสคำสั่ง (Opcode ในโปรแกรม) จ) เบส-รีจิสเตอร์ร่วมกับ อินเดกรีจิสเตอร์อินไดเรคแอดเดรสซิ่งเป็นการแอดเดรส ข้อมูลจากหน่วยความจำของโปรแกรม โดยการใช้แอดเดรสจากผลรวมของเบสรีจิสเตอร์ (DPTR) และอินเดกรีจีสเตอร์ ACC จะใช้งานเป็นแบบเปิดตาราง

3) บูลีนโปรเซสเซอร์ (Boolean Processor)

บูลีนโปรเซสเซอร์เป็นอินทิเกรทบิทโปรเซสเซอร์ใน 8031 จะมีชุดคำสั่งเฉพาะแอดคิวมูเลเตอร์ (Carry Flag) และบิทแอดเดรสเอเบิลใน RAM และอินพุท/เอาท์พุทสามา

* CY จะถูกเซทถ้าการคำนวณเกิดตัวทดที่บิดสูงของผลลัพท์นอกนั้น CY ถูกเคลียร์

* AC จะถูกเซทถ้าผลลัพท์เกิดตัวทดจาก 4 บิท ล่างของผลลัพท์ (ในระหว่างการบวก) หรือเกิดทดที่บิทสูงสุด (ในระหว่างการบบ นอกนั้น AC จะถูกเคลียร์)

* OV จะถูกเซทถ้าการคำนวณเกิดทดตัวเปห็น 1 ในบิทสูงสุดของผลลัพท์ นอกนั้น OV จะถูกเคลียร์ OV จะถูกใช้ในการทำงานแบบ 2' s Complement เพราะ OV จะถูกเซทเมื่อผลลัพธ์ไม่สามารถเก็บในค่า 8 บิท

* P จะถูกเซทเป็น Modulo 2 ผลบวกของ 8 บิท ในแอดคิวมูเลเตอร์เป็น 1 (Odd Parity) นอกจากนั้น P จะถูกเคลียร์ (Even Parity) เมื่อค่านี้ถูกเก็บใน PSW บิท P จะไม่เปลี่ยนแปลงและจะแสดงพาร์ริตี้ของ A เสมอ

ค) ลอจิก (Logic)

CPU 8051 สามารถทำงานเกี่ยวกับลอจิกได้ทั้งแบบบิทและไบท์ โอเปอแรนด์การทำงานโอเปอแรนด์เดียว

* CLR ให้ค่าใน A หรือไดเรคแอดเดรสเอเบิล มีค่าเป็น 0

* SET ให้ค่าในไดเรคเอเบิล มีค่าเป็น 0

* CPL ใช้ทำการคอมพลีเมนต์ค่าของรีจิสเตอร์ A โดยไม่มีผลต่อแฟลก หรือไดเรคแอดเดรสเอเบิลใด ๆ

*RL, RIC, RR, RRC, SWAP เป็นการทำงานสำหรับการรวบรวมข้อมูล โดยสามารถทำงานกับรีจิสเตอร์ ARL เป็นการรวบรวมข้อมูลไปทางซ้าย RR เป็นการรวมข้อมูลไปทางขวา RLC เป็นการรวมข้อมูลไปทางขวาผ่าน CY และ SWAP เป็นการรวมข้อมูลไปทางซ้าย 4 ตำแหน่ง สำหรับ RLC และ RRC ค่าแฟลก CY จะเท่ากับบิทรวมเข้ามายัง CY SWAP วงข้อมูลไปทางซ้าย 4 ตำแหน่ง โดยจะเป็นการสลับข้อมูลบิท 3 ถึ 0 บิท 7 ถึง 4

การทำงานแบบ 2 โอเปอแรนด์

AML จะทำการ AND บิทของ 2 Source ในโอเปอแรนด์ (หรือทั้งบิทและไบท์ โอเปอแรนด์) แล้วเก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

XRL ทำการ Exclusive OR บิทของ Source ใน โอเปอแรนด์ เก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

ง) คอนโทรล ทรานสเฟอร์

รถใช้ตรวจสอบเงื่อนไขเพื่อให้มีการกระโดดหรือไม่กระโดด เมื่อมีการเซทหรือไม่ถูกเซท ข้ามถ้ามีการเซทแล้วเคลียร์ MOVX เข้า Carry แอดเดรสเอเบิล หรือค่าคอมพลีเมนท์ สามารถ AND หรือ OR กับค่าของ Carry Flag ได้ผลลัพท์จะเก็บใน Carry รีจิสเตอร์

2.2.3 ชุดคำสั่ง

ใน MCS-51 จะมีชุดคำสั่งอยู่ 111 คำสั่ง 49 เป็นแบบไบท์เดียว 45 คำสั่งเป็นแบบ 2 ไบท์และ 17 คำสั่งเป็นแบบ 3 ไบท์ รูปแบบรหัสคำสั่งประกอบด้วยนิวโมนิค (MNEMONIC) ตามด้วย "Destination Source" (Operand Field) ซึ่งในโอเปอแรนด์ฟิลด์นี้ จะมีชนิดของข้อมูลและวิธีแอดเดรสซิ่งอยู่ด้วย

1) ฟังก์ชั่นทั้งหมด (Functional Overview)

ชุดคำสั่ง MCS 51 จะแบ่งออกเป็น 4 กลุ่ม ตามการทำงานดังนี้

- เคลื่อนย้ายข้อมูล (Data Transfer)
- คำนวณ (Arithematic)
- ลอจิก (Logic)
- คอนโทรลทรานสเฟอร์ (Control Transfer)

ก) เคลื่อนย้ายข้อมูล (Data Transfer) การเคลื่อนย้ายข้อมูลแบ่งออกเป็น 3 แบบ

- การย้ายข้อมูลไป
- กำหนดแอดคิวมูเลเตอร์
- แอดเดรสออปเจค

การทำงานไม่มีผลต่อแฟลกใน PSW ยกเว้น POP หรือ MOV โดยตรงกับ PSW การย้ายข้อมูลทั่วไป

การย้ายข้อมูลทั่วไป

* MOV จะสามารถย้ายข้อมูลได้ทั้งแบบบิทหรือไบท์จาก Source ในโอเปอแรนด์ไปยัง Destination ใน โอเปอแรนด์

* PUSH จะเพิ่มค่าของรีจิสเตอร์ SP แล้วย้ายข้อมูลจาก Source .ในโอเปอแรนด์ไปยังตำแหน่งที่ชี้โดย SP

* POP จะย้ายข้อมูลจากสแตกตำแหน่งที่ชี้โดยรีจิสเตอร์ SP ไปยัง Destination ในโอเปอแรนด์ แล้วลดค่า SP

การกำหนดแอดคิวมูเลเตอร์

* XCH จะแลกข้อมูลจาก Source Operand กับ Accumulator

* XCHD จะแลกข้อมูลเฉพาะ 4 บิทล่างของไบท์ กับ 4 บิทล่างของ Accumulattor หน่วยความจำภายนอก จะอ้างอิงโดยรีจิสเตอร์ DPTR (16 บิท) หรือรีจิสเตอร์ RL, RO (8 บิท)

*MOVC ย้ายข้อมูลจากในหน่วยความจำของโปรแกรม ไปยังแอดคิวมูเลเตอร์โอเปอแรนด์ใน A จะถูกใช้เป็นอินเด็กซ์ไปยัง 256 ไบท์ โดยใช้ร่วมกับ DPTR หรือ PC ค่าจากหน่วยความจำจะถูกอ่านมายังแอดคิวมูเลเตอร์แอดเดรส-ออปเจค ทรานเฟอร์

MOV DPTR, # data จะกำหนดค่าจากโอดเปอแรนด์ให้กับคูรีจิสเตอร์ DPH และ DPL โดยตรง

ข) คำนวณ (Arithemattic)

8031 สามารถทำการคำนวณได้ 4 แบบ และเป็นแบบ 8 บิทไม่มีเครื่องหมาย มีแฟลกบอกการโอเวอร์โฟลว์ อย่างไรก็ตามในการบวกและลบ จะสามารถทำได้ทั้งแบบมีเครื่องหมายและไม่มีการคำนวณสามารถให้ผลออกมาในรูปของเลขฐานสิบ (BCD) ได้โดยตรง

การบวก

* INC (Increment) บวก 1 เข้ากับ Source ในโอเปอแรนด์ และเก็บผลลัพท์ในโอเปอแรนด์

* ADD บวก A เข้ากับ Source ในโอเปอร์แรนด์ และเก็บผลลัพท์ใน A

* การบวกซึ่งเกิดจากการบวกเลขฐานสิบเข้าด้วยกันให้ได้ผลลัพธ์ เป็นเลขฐานสิบ ค่าที่ได้จะเก็บใน A จะมี CY ถูกเซท ถ้าค่าที่ได้เกิน 99 นอกนั้น CY จะถูกเคลียร์

การลบ

* SUBB (Substract with Borrow) ลบ Source ที่สองในโอเปอแรนด์ออกจาก Source แรก ในโอเปอแรนด์ (แอดคิวมูเลเตอร์) และลบด้วย 1 ถ้า CY ถูกเซทเก็บผลลัพธ์ใน A

* DEC (Decrement) ลบ 1 จาก Source ในโอเปอแรนด์และเก็บผลลัพธ์ในโอเปอแรนด์

การคูณ

* MUL จะทำการคูณค่าจากรีจิสเตอร์ A กับรีจิสเตอร์ B แบบไม่คิดเครื่องหมายเข้าด้วยกัน และผลลัพธ์จะเป็น 2 ไบท์ต่ำเก็บใน A และไบท์สูงเก็บใน B โอเวอร์โฟล์จะถูกเคลียร์ถ้าไบท์บิทเป็น 0 และจะถูกเซ็ทถ้าไม่เป็น 0 CY จะถูกเคลียร์ ACC ไม่มีการเปลี่ยนแปลง

นอกจากที่กล่าวมาข้างบนแล้ว แฟลกใน PSW จะมีการเปลี่ยนแปลงดังนี้

มี 3 วิธีสำหรับการคอนโทรลทรานสเฟอร์ การ CALL แบบไม่มีเงื่อนไข Return และ Jumps จะทำให้การทำงานจากโปรแกรมเคาร์เตอร์เดิม ไปยังแอดเดรตำแหน่งใหม่ทั้งแบบโดยตรงและโดยอ้อม

* ACALL และ LCAL จะเก็บค่าของแอดเดรสของคำสั่งต่อไปไว้ในสแตก แล้วข้ามไปทำงานยังแอดเดรสที่ต้องการ ACALL เป็นคำสั่งแบบ 2 ไบท์ ถูกใช้เมื่อตำแหน่งที่จะไปอยู่ภายในแต่ละช่วง 2 Kbyte โดยเริ่มนับจาก PC ที่ต่อจากคำสั่ง ACALL, LCALL เป็นคำสั่งแบบ 3 ไบท์ ซึ่งอ้างอิงตำแหน่งหน่วยความจำได้ถึง 64 Kbyte ใน ACALL ข้อมูล 11 บิท จากโอเปอแรนด์จะรวมกับ 5 บิทบนของ PC ชี้ไปยังตำแหน่งหน่วยความจำที่ต้องการ ถ้า Acall ชี้ในตำแหน่ง 2 ไบท์สุดท้ายของหน้า 2 Kbyted การ CALL จะทำงานในหน้าต่อไปโดยค่าของ PC จะเพิ่มไปเพื่อชี้ตำแหน่งที่จะทำงาน

* RET จะกระโดดข้ามการทำงานไปยังตำแหน่งแอดเดรสที่ถูกเก็บอยู่ในสแตค โดยการ CALL ที่ผ่านมาและลดค่าของ SP ลง 2 เพื่อให้ SP จะได้ POP แอดเดรส ต่อไปได้ถูกต้อง

* AJMP LJMP และ SJMP เป็นคำสั่งข้ามการทำงานไปยังตำแหน่งในโอเปอแรนด์การทำงานของ AJMP และ LJMP จะเหมือนกับการทำงานในช่วง 128 ไบท์ จากแอดเดรสของคำสั่งต่อไปนี้ (-128 ถึง + 127) โดยมีแอดเดรสนั้นเป็นกึ่งกลาง

* JMP @A+DPTR คำสั่งการทำงานแบบอ้างอิงกับ DPTR โดยที่โอเปอแรนด์ใน A จะใช้สำหรับเป็นออฟเซท (0-225) ยังแอดเอรสในรีจิสเตอร์ DPTR ดังนั้นสามารถ ข้ามไปยังตำแหน่งใด ๆ ในหน่วยความจำ

การข้ามแบบมีเงื่อนไข

จะเป็นการข้ามตามสภาวะที่กำหนดปลายทางอยู่ใน +128 ไบท์จากกึ่งกลาง คือ แอดเดรสเริ่มต้นของคำสั่งต่อไปนี้ (-128 ถึง + 127)

JZ เป็นคำสั่งการข้ามถ้าแอดคิวมูเลเตอร์เป็น 0

JNZ เป็นคำสั่งให้ข้ามถ้าแอดคิวมูเลเตอร์ไม่เป็น 0

JC คำสั่งให้ข้ามถ้าแฟลกตัวทด (Carry Flag) ถูกเซท

JNC คำสั่งให้ข้ามแฟลกตัวทด (Carry Flag) ถูกเคลียร์

JB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ถูกเซท

JNB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ไม่ถูกเซท

CJNE จะเปรียบค่าใน 2 โอเปอแรนด์และการกระโดด ถ้าสมมุติว่าโอเปอแรนด์ตัวแรกมีค่าน้อยกว่าโอเปอแรนด์ที่ 2CY จะถูกเซท นอกนั้นแล้วจะถูกเคลียร์ การเปรียบเทียบอาจเป็นระหว่าง A รีจิสเตอร์ที่กำลังใช้งาน หรือ กับรีจิสเตอร์-อินไดเรคแอด

เดรสไบท์ ของ RAM ภายใน DJNZ ลดค่าของ Sourc ในโอเปอแรนด์และเก็บค่าผลลัพธ์ในโอเปอแรนด์การข้ามจะเกิดขึ้นถ้าผลลัพธ์ไม่เป็นศูนย์ (0) Source ในโอเปอแรนด์คำสั่ง DJNZ อาจจะเป็นไบท์ใดในหนึ่งความจำข้อมูล ภายในก็ได้ หรือรีจิสเตอร์แอดเดรสแอดเดรสซึ่งก็จะใช้เป็นแอดเดรสใน Source ในโอเปอแรนด์ได้ Interrupt Rturn RETI จะเป็นคำสั่งคล้ายกับ RET จะเพิ่มการทำงานในการอีนาเบิลอินเทอร์รัพท์ที่มีลำดับความสำคัญเท่ากันเพื่อให้มีการอินเทอร์รัพท์ต่อไปได้

2.3 MCS-51 กับการติดต่อสื่อสารทางพอร์ทอนุกรม

MCS-51 เป็นชิงเกิลชิพที่ค่อนข้างจะสมบูรณ์ ซึ่งภายในตัว MCS-51 ประกอบด้วยไทเมอร์/ตัวนับ ,ออสซิลเลเตอร์, พอร์ท I/O , พอร์ทอนุกรม จากรูปที่ 2.6 เป็นบล็อกการทำงานของ MCS-51 โดยแสดงส่วนที่เกี่ยวข้อง และการจัดการเกี่ยวกับการสื่อสารข้อมูลทางพอร์ทอนุกรม โดยใช้พอร์ทอนุกรม โดยใช้พอร์ท 3 บิท P.3.0. และ P.3.1 เป็นส่วนติดต่อข้อมูลโดยที่ P.3.0. จะทำหน้าที่รับข้อมูลเข้ามาเรียกว่า ขา RXD ส่วนในการส่งข้อมูล MCS-51 จะส่งออกที่ขา P.3.1 เรียกว่าขา TXD

2.3.1 รีจิสเตอร์ที่เกี่ยวข้อง

การใช้งาน MCS-51 ในด้านสื่อสารข้อมูลทางพอร์ทอนุกรมจะมีรีจิสเตอร์ที่เกี่ยวข้องหลายตัวคือ

ก) PCON = Power Mode Control Register)

SMOD	-	-	-	GF1	GFO	PD	IDL

รูปที่ 2.6 แสดงบิทต่าง ๆ ของรีจิสเตอร์ PCON

รีจิสเตอร์ตัวนี้จะใช้งานเพียงบิตเดียวคือ SMOD ถ้า SMOD เป็น “0” ค่า K เป็น 1 SMOD เป็น “1” ค่า K จะเป็น 2

ข) IE: (Interrupt Enable Register)

EA	-	ET2	ES	ET1	EX1	ET0	EX0

รูปที่ 2.7 แสดงบิทต่าง ๆ ของรีจิสเตอร์ IE

รีจิสเตอร์ตัวนี้เป็นตัวที่ใช้กำหนดความสำคัญของการอินเตอร์รัพท์ ในกรณีที่ต้องการให้พอร์ทอนุกรมมีการอินเตอร์รัพท์ขณะทำการรับส่งข้อมูลให้ใช้คำสั่ง SET EA และ SET ES แล้วนำเอาโปรแกรมการรับส่งไปไว้ที่ตำแหน่ง RI & TI (0023H)

ค) IP : (Interrupt Priority Register)

-	-	SM2	REN	TB8	RB8	TI	RI

รูปที่ 2.8 แสดงบิทต่าง ๆ ของรีจิสเตอร์ IP

ในกรณีที่ต้องการให้พอร์ทอนุกรมมีการอินเตอร์รัพท์ ขณะที่รับส่งข้อมูลรีจิสเตอร์ตัวนี้เป็นตัวที่ใช้กำหนดความสำคัญของการอินเตอร์รัพท์ ถ้าบิทใดเป็น "1" ลักษณะการอินเตอร์รัพท์ในบิทนั้นจะมีความสำคัญสูงสุด คำสั่งที่ใช้เป็นเช่น SETB IP

ง) TCON : (Timer Control Register)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

รูปที่ 2.9 แสดงบิทต่าง ๆ ของรีจิสเตอร์ TCON

รีจิสเตอร์นี้มีบิทที่ต้องใช้คือ TRO และ TR1 เป็นตัวควบคุมไทเมอร์และเคาน์เตอร์โดยถ้าเป็น "0" จะหมายถึง "off" และถ้าเป็น "1" หมายถึง "on" ฉะนั้นหลังจากเซทค่าต่าง ๆ หมดแล้วเราก็จะเปิดใช้คำสั่ง SETB TRO หรือ SBTB TR1 ถ้าต้องการปิดก็ใช้คำสั่งCLR TR1 หรือ CLR TR1

จ) TMOD : (Timer control register)

GATE	C/T	M1	M0	GATE	C/T	M1	M0

รูปที่ 2.10 แสดงบิทต่าง ๆของรีจิสเตอร์ TMOD

ในการใช้งานพอร์ทอนุกรมจะใช้งานรีจิสเตอร์นี้บางบิทคือ

-GATE ถ้า GATE เป็น "1" Trx (TRO หรือ TR1 ก็ได้) เป็น "1" ไทเมอร์/เคาน์เตอร์จะทำงานเมื่อ INTx เป็น "1" เท่านั้น และถ้า GATE เป็น "1" เท่านั้น นั่นคือถ้าให้ GATE ขอไทเมอร์ 1 เป็น "1" และ TR1 ใน TCON เป็น "1" จะทำงานเมื่อ INT1 เป็น "1" เท่านั้น ไทเมอร์ 0 ก็เช่นเดียวกัน

-C/T ถ้า C/T เป็น "1" จะเป็นโหมดเคาน์เตอร์โดยจะนับสัญญาณนาฬิกาจากขา Tx ถ้า C/T เป็น "0" จะเป็นโหมดไทเมอร์โดยจะเอาสัญญาณนาฬิกาจากภายใน CPU มาตั้งเวลา

M1, MO เป็นตัวเลือกโหมดการทำงานของไทเมอร์/เคาร์เตอร์โดยมีรายละเอียดดังตารางที่ 2.3 จะใช้ไทเมอร์ 1 ในการสร้างบ็อดเรทโดยใช้ไทเมอร์โหมด 2 โดยต้องกำหนดค่าการทำงานดังนี้

GATE - "0"

ตารางที่ 2.4 การเลือกโหมดการทำงานบ็อดเรทของ SCON โดยใช้ MO และ M1

MO	M1	โหมด	คำอธิบาย	บ็อดเรท
0	0	0	ชิฟต์รีจิสเตอร์	ความถี่สัญญาณนาฬิกา/2
0	1	1	USART 8 บิท	เลือกค่าได้
1	0	2	USART 9 บิท	ความถี่สัญญาณนาฬิกา/2 หรือ 64
1	1	3	USART 9 บิท	เลือกค่าได้

* USART : Universal Asynchronous Reciver Transciver

2.3.2 การคำนวณบ็อดเรท

โหมด 1

บ็อดเรท -2 SMOD/32 * 12* [(256-(TH1)]

โหมด 0

บ็อดเรท - ความถี่สัญญาณนาฬิกา/12

TH1 - 256 [(2SMOD * ความถี่สัญญาณนาฬิกา/384* บ็อดเรต)]

โหมด 2

บ็อดเรท - [2SMOD * ความถี่สัญญาณนาฬิกา]/64

ถ้าใช้ไทเมอร์ 1 เป็นตัวกำเนิดบ็อดเรทในการคำนวณจะเหมือนกับกรณีของ 8031, 8051,8751 แต่ถ้าใช้ไทเมอร์ 2 การคำนวณบ็อดเรทจะทำได้ดังนี้

บ็อดเรท - ความถี่สัญญาณนาฬิกา/ 32* [65536 - (RCAP2H, RCAP2L)]

RCAP2H,RCAP2L - 65536 - ความถี่ของสัญญาณนาฬิกา/32 * บ็อดเรท

2.4 การสื่อสารข้อมูล

ความก้าวหน้าทางคอมพิวเตอร์ในยุคแรก ๆ นั้นจะติดตั้งและทำงานแต่เพียงในศูนย์คอมพิวเตอร์เท่านั้น ระบบการทำงานเบ็ดเสร็จภายในศูนย์ เมื่อต้องการให้บริการต่อผู้อื่นที่อยู่ห่างไกล และเพิ่มความสามารถต่อการใช้งาน ระบบคอมพิวเตอร์ก็เริ่มเป็นเครือข่ายทำงานตอบสนองต่อผู้ใช้ได้เป็นจำนวนมาก ครั้นถึงยุคของไมโครคอมพิวเตอร์ในขั้นแรกมักมีการทำงานในรูปที่เรียกว่า Stand alone หรือทำงานส่วนตัว แต่ต่อมาก็สามารถเชื่อมโยงเข้าหาเครื่องคอมพิวเตอร์ที่ใหญ่ขึ้นและเชื่อมโยงกันเองเป็นเครือข่ายที่สลับซับซ้อนจนในที่สุด คอมพิวเตอร์ต่าง ๆ กันในโลกนี้สามารถที่จะส่งข่าวสารถึงกันได้

C/T - "0"(โหมดไทเมอร์)

MO - "0"

M1 - "0" (8 บิท ออโตรีโหลด)

ตารางที่ 2.3 การเลือกโหมดการทำงานของไทเมอร์/เคาร์น์เตอร์โดยใช้ MO, M1

MO	M1	โหมด	คำอธิบาย
0	0	0	13 บิทไทเมอร์
0	1	1	13 บิทไทเมอร์
1	0	2	8 บิทออโตรีโหลด
1	1	3	ไทเมอร์/เคาน์เตอร์ 8 บิท 2 ชุด

ฉ) SCON : (Serial Port Control Register)

SM0	SM1	SM2	REN	TB8	RB8	T1	R1

รูปที่ 2.11 แสดงบิทต่าง ๆ ของรีจิสเตอร์ SCON

การเลือกโหมดการทำงานของ SCON จะต้องใช้การควบคุมจาก MO, M1 ดังตารางที่ 2.4 บิทต่าง ๆ ที่เกี่ยวข้องกับการสื่อสารทางพอร์ทอนุกรมในรีจิสเตอร์ SCON ได้แก่

- SM2 ถ้า SM2 เป็น"1" ในโหมด 2 หรือ 3 RI จะไม่เป็น "1" ถ้าข้อมูลบิทที่ 9 เป็น "0" ส่วนในโหมด 1 ถ้า SM2 เป็น "1" R1 จะไม่เป็น "1" ถ้ายังไม่ได้รับสตอปบิท

- REN ถ้าบิทนี้เป็น "0" จะส่งได้ทางเดียว ถ้าเป็น "1" จะทำงานได้ทั้งรับส่ง

- TB8/TR8 ถ้าใช้งานในโหมด 2 หรือ 3 RB8 จะเป็นข้อมูลที่รับบิทที่ 8 และ TB8 จะเป็นข้อมูลที่ต้องการส่งออกบิทที่ 8

- TI จะเป็น "1" เมื่อข้อมูลใน SBUF ได้ส่งไปแล้ว

- RI จะเป็น "1" เมื่อได้รับข้อมูลครบทุกบิทหรือพบสตอปบิทแล้ว

2.4.1 หลักการสื่อสาร

สมมุติว่าคอมพิวเตอร์เครื่องหนึ่งต้องการติดต่อสื่อสารกับคอมพิวเตอร์อีกเครื่องหนึ่งสิ่งที่ต้องการกระทำต่อกันคือ การเชื่อมโยงให้เกิดช่องสัญญาณระหว่างทั้งสองถ้าเป็นการส่งในลักษณะทางเดียวในสายสัญญาณช่องเดียวเราก็จะเรียกวิธีการส่งแบบนี้ว่าซิมแพล็กซ์ (Simplex) แต่ถ้าสายสัญญาณช่องเดียวแต่พลัดกันส่งโดยใช้เวลาไม่ตรงกันก็เรียกวิธีการส่งแบบนี้ว่า ฮาร์ฟดูเพล็กซ์ (Half Duplex) แต่ถ้าเป็นการส่งสัญญาณไปและกลับพร้อมกันได้เราเรียกการส่งข้อมูลแบบนี้ว่า ฟูลดูเพล็กซ์ (Full Duplex)

อนึ่งการสื่อสารที่จะกว่าถึงนี้ จะเป็นเฉพาะถึงการสื่อสารข้อมูลแบบอนุกรมเท่านั้นซึ่งเป็นที่นิยมใช้งานกันอย่างกว้างขวางทั้งระหว่างไมโครคอมพิวเตอร์กับไมโครคอมพิวเตอร์ด้วยกัน หรือระหว่างไมโครคอมพิวเตอร์กับมินิคอมพิวเตอร์ในการสื่อสารข้อมูลแบบอนุกรมเรามีหน่วยวัดความเร็วเป็นจำนวนบิทต่อวินาที (bps) ซึ่งเครื่องที่จะใช้รับส่งจะต้องมีความเร็วในการส่ง และรับข้อมูลที่เท่ากันจึงจะติดต่อสื่อสารกันได้ อัตราความเร็วบิทต่อวินาทีนี้ เป็นผลรับที่เกิดขึ้นจากระบบการสื่อสาร แต่เรายังมีหน่วยที่วัดการเปลี่ยนแปลงของสัญญาณในหนึ่งวินาทีซึ่งเรียกว่า “บ็อดเรท” (Baud Rate) การเปลี่ยนแปลงของสัญญาณ 1 ครั้งอาจหมายถึงการส่งข้อมูลหลาย ๆ บิทก็ได้ ดังนั้นอัตราบ็อดเรทจึงเป็นตัวเลขที่ไม่เท่ากับอัตราบิทได้ และถ้าเขียนเป็นรูปสมการจะได้

อัตราบิท - อัตราบ็อด X (บิทใน 1 บ็อด) .....(2.1)

2.4.2 การรับส่งข้อมูลแบบซิงโครนัส

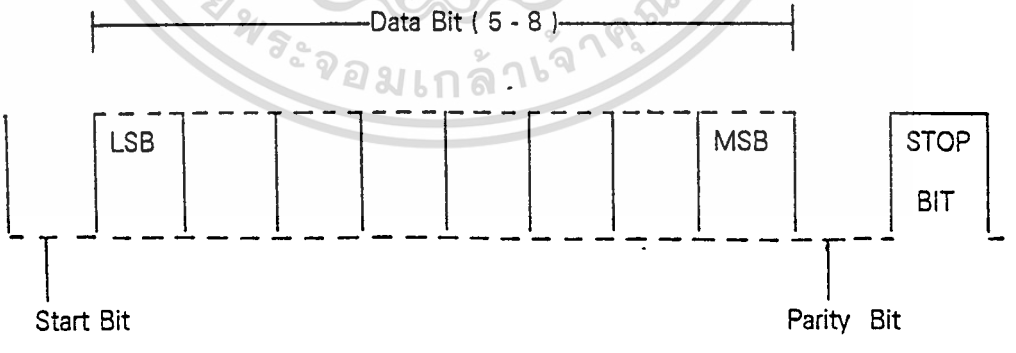
ไม่ว่าการส่งข้อมูลจะเป็นแบบขนานหรืออนุกรม การส่งข้อมูลแบบซิงโครนัสก็คือ ระบบการส่งข้อมูลแต่ละเวิร์ดถูกส่งออกไปตามเวลาที่แน่นอน ซึ่งหมายถึงระยะเวลาระหว่างข้อมูลแต่ละเวิร์ดที่ถูกส่งออกไปมีค่าแน่นอน มีความต่อเนื่องข้อมูลไม่มีบิทสตาร์ทหรือบิทสต็อป หรือแม้กระทั่งบิทพารีตี้ โพรโทคอลที่ใช้ในการรับและส่งแบบซิงโครนัสจึงแตกต่างไปจากโพรโทคอลแบบอะซิงโครนัสมีโพรโทคอลหลายแบบที่ใช้ในการรับส่งแบบซิงโครนัส เช่น Bisync Protoca, SDLC, HDLC เป็นต้น

ระบบซิงโครนัสนี้จะมีประสิทธิภาพในการส่งสัญญาณข้อมูล และเหมาะสำหรับการส่งข้อมูลในอัตราความเร็วและปริมาณข้อมูลสูง

2.4.3 การรับส่งข้อมูลแบบอะซิงโครนัส

การรับส่งแบบนี้ พัฒนามาจากการส่งโทรพิมพ์สมัยก่อน ลักษณะของสัญญาณได้แสดงไว้ดังรูปที่ 2.6 เพื่อเพิ่มกลไกในการรับ และส่งอย่างถูกต้อง สัญญาณอะซิงโครนัสจะประกอบด้วย บิทสตาร์ท (Start Bit) และ บิทสต็อป (Stop Bit) ขณะที่สถานะของการ

ส่งแบบว่าง (IBLE) คือยังไม่มีสัญญาณส่งออก จะมีสัญญาณ หรือมีแรงอัดตลอดเวลาเพื่อให้แน่ใจว่าฝ่ายรับยังคงติดต่ออยู่กับฝ่ายส่ง เมื่อเริ่มส่งสัญญาณข้อมูลของอะซิงโครนัสจะเป็นศูนย์หนึ่งช่วงสัญญาณนาฬิกา บิทนี้เราเรียกว่า START BIT และข้อมูลที่ตามหลัง START BIT ก็จะเป็นข้อมูลสำหรับหนึ่งตัวอักษรซึ่งจะมีขนาดตั้งแต่ 5 บิท จนถึง 8 บิท โดยที่บิทมีค่าน้อยที่สุด (LSB) ถูกส่งออกมาก่อนไปจนถึงบิทที่มีค่ามากที่สุด (MSB) การเข้ารหัสอักขระนี้จะใช้รหัส ASCLL เป็นตัวส่งผ่าน ซึ่งแรกเริ่มทีเดียวในงานของโทรพิมพ์จะใช้รหัสโบดอต (Boudot Code) ซึ่งใช้จำนวน 5 บิท ในการแทนอักขระ 1 ตัว และบิทที่ตามหลัง ข้อมูลจะเป็นพาริตี้บิท ซึ่งอาจจะใช้หรือไม่ก็ได้ พาริตี้บิทจะทำหน้าที่เป็นตัวตรวจสอบความถูกต้องของสัญญาณ ที่ได้รับพาริตี้บิทอาจจะเป็นแบบคู่ (Even) หรือแบบที่ (Odd) หมายความว่า ถ้าเป็นพาริตี้คู่ ผู้ส่งจะต้องทำหน้าที่ตรวจสอบข้อมูลแล้วใส่พาริตี้บิทเอง ฝ่ายรับเมื่อรับแล้วก็จะต้องตรวจสอบดูว่าเป็นดังสถานการณ์ที่ตั้งเอาไว้หรือไม่ หากผิดพลาดก็หมายความว่าสัญญาณที่รับนั้นผิดพลาดไปจากสถานีส่งที่ออกมา ทั้งนี้จะต้องเป็นจำนวนคี่เท่านั้นคือผิดไป 1 บิท 3 บิท หรือ 5 บิท พร้อมกันจึงจะตรวจสอบได้ว่าผิด ซึ่งมองเป็นง่าย ๆ ว่าถ้าผิดเป็นจำนวนคู่ผลรวมของจำนวนที่เป็นหนึ่งก็ยังเป็นคู่ดีไม่ได้ หมายความว่าพาริตี้ที่ (Odd Parity) จะตรวจความผิดพลาดเป็นจำนวนคี่ ความจริงแล้วตรวจสอบความผิดพลาดได้เหมือนพาริตี้คู่ (Even Parity) แต่แทนที่จะตรวจสอบดูว่าสัญญาณที่รับเข้ามามีจำนวนคู่ก็ตรวจสอบดูว่ามีจำนวนคี่หรือเปล่าอย่างไรก็ตามโอกาสที่จะผิดพลาด 2 บิท พร้อมกันมีน้อยมาก



รูปที่ 2.12 แสดง Format การสื่อสารแบบอะซิงโครนัส

หลังจากที่ส่งพาริตี้บิทออกไปแล้ว ก็ต้องมีสต็อปบิทตามหลัง ซึ่งความกว้างของสต็อปบิทอาจจะเป็น 1, 1.5, หรือ 2 ฟิลส์ของสัญญาณนาฬิกา แล้วแต่ผู้รับและผู้ส่งจะตกลงกัน การเริ่มใช้พอร์ทอนุกรมจึงจำเป็นต้องกำหนดค่าต่าง ๆ สำหรับการส่งแบบอนุกรมอันได้แก่

1) ความเร็วในการส่ง

2) ความยาวของรหัส 1 ตัวอักษร

3) บิทตรวจสอบ

4) จำนวนสต็อปบิท

ในการส่งโทรพิมพ์ หรือ โทรเลข เมื่อก่อนนี้ใช้ความเร็วแค่ 70 บอต และ 110 บอต สำหรับคอมพิวเตอร์ ความเร็วในการส่งมีให้เลือกตั้งแต่ 110, 100, 300, 1200, 2400, 4800, 9600 บอต และสูงไปกว่านั้น เนื่องจากมีไอซีหลายเบอร์ที่ทำหน้าที่ส่งแบบอะซิงโครนัสการส่งแบบอนุกรมจึงสะดวกสบายสำหรับการออกแบบ (การส่งสัญญาณด้วยความเร็ว 300 บอต สำหรับการเข้ารหัส 7 บิท มิได้หมายความว่าส่งได้ 300 หารด้วย 7 อักขระต่อวินาที)

เทคนิคในการควบคุมความเร็วในการส่งมีอยู่หลายรูปแบบซึ่งอาจจะแบ่งออกเป็น 2 ลักษณะ คือ การส่งข้อมูลบอกการออนออฟการทำงาน (On-Off Data Flow Toggle) และหาข้อมูลชั่วคราวหรือสร้างบัฟเฟอร์ (Temporary Data Storag Machanism)

ก) การมีบัฟเฟอร์ในการสื่อสารข้อมูล (Buffer) ในที่นี้หมายถึงหน่วยความจำในคอมพิวเตอร์ ซึ่งแยกออกมาจากหน่วยความจำหลักสำหรับเก็บพักข้อมูล ในการติดต่อชั่วคราวบัฟเฟอร์สำหรับการสื่อสารนี้ส่วนมากใช้สำหรับฝ่ายรับเท่านั้น เนื่องจากฝ่ายรับจำเป็นต้องตามฝ่ายส่งให้ทันถ้าหากฝ่ายรับใช้ภาษาแอสแซมบลี้ควบคุมซึ่งมีความเร็วเพียงพอก็ไม่จำเป็นต้องใช้บัฟเฟอร์สำหรับการสื่อสาร เนื่องจากภาษาแอสแซมบลี้เป็นภาษาที่มีความเร็วสูง

- ข้อมูลที่จัดส่งให้คอมพิวเตอร์ที่เป็นฝ่ายรับส่วนมากจะอ่านมาจากแฟ้มที่บันทึกไว้ในดิสก์ อาจจะพิจารณาระหว่างการส่งข้อมูลออกข้อมูลที่อ่านมาจากดิสก์จะมีลักษณะเป็นกลุ่มได้รับการนำมาสู่บัฟเฟอร์ การอ่านแต่ละกลุ่มดำเนินไปจนกระทั่งบัฟเฟอร์เต็ม การอ่านจะหยุดลงจนกระทั่งบัฟเฟอร์ถูกส่งออกไปหมดในลักษณะของเข้าก่อนออกก่อน ข้อมูลก็จะอ่านเข้ามาใส่บัฟเฟอร์โดยปกติบัฟเฟอร์ส่งจะมีขนาด 255 ตัวอักษรหรือประมาณ 30 บรรทัดของ 80 ตัวอักษรบัฟเฟอร์ของฝ่ายรับมีผลกระทบต่อการรับส่งของข้อมูลมากกว่าบัฟเฟอร์รับหน้าที่เช่นเดียวกับบัฟเฟอร์ส่ง แต่ทิศทางการไหลของข้อมูลอยู่ในทางตรงกันข้ามฝ่ายรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์รับก่อนจนกว่าโปรแกรมควบคุมการสื่อสารจะนำเอาข้อมูลออกจากบัฟเฟอร์รับเพื่อนำออกไปที่จอภาพ หรือนำไปไว้ในแฟ้มก็แล้วแต่ ในระบบควบคุมการทำงานของไมโครคอมพิวเตอร์อย่างเช่น IBM PC มีบัฟเฟอร์รับส่งข้อมูลนี้อยู่ภายในโปรแกรมภาษาสูงจึงทำหน้าที่เพียงแต่หน้าที่ถึงเอาข้อมูลจากบัฟเฟอร์นี้มาใช้เราจะเห็นถึงความจำเป็นในการใช้บัฟเฟอร์เมื่อใช้ความเร็วในการส่ง

สูงเกินกว่า 600 บอด ภาษาในระดับสูง เช่น ภาษาเบสิกไม่สามารถที่จะรับข้อมูลจากพอร์ทอนุกรมได้ทันระบบควบคุมการทำงานจึงถูกออกแบบมาเพื่อการสื่อสารรับข้อมูลโดยการใช้การขัดจังหวะ (Interrupt) เข้าช่วย เมื่อมีข้อมูลเข้ามาพอร์ทอนุกรมระบบควบคุมจะเข้ามาขัดจังหวะการทำงานเพื่อดึงข้อมูลไปใส่ในบัฟเฟอร์รับทันที เพื่อไม่ให้ข้อมูลที่รับหายไปก่อนที่จะมีข้อมูลตัวใหม่ส่งยังพอร์ทอนุกรม

หน้าที่โปรแกรมควบคุมการรับส่ง ก็คือ การอ่านข้อมูลจากบัฟเฟอร์รับไปใช้เมื่อถูกอ่านจากบัฟเฟอร์รับไปแล้ว ตัวที่อ่านออกไปก็จะหายไปจากบัฟเฟอร์ โปรแกรมที่เขียนด้วยภาษาแอสแซมบลี้สามารถทำงานได้เร็ว และอาจจะไม่จำเป็นต้องใช้บัฟเฟอร์สำหรับการรับส่งก็ได้แต่ถ้าหากเขียนด้วยภาษาเบสิกก็จำเป็นต้องมีบัฟเฟอร์อย่างน้อย 1024 ไบท์ สำหรับการสื่อสารมีความเร็วไม่เกิน 600 บอด

ข) การควบคุมโดยใช้ XON/XOFF ถึงแม้ว่าเราจะมีบัฟเฟอร์สำหรับที่จะใช้ในการสื่อสารข้อมูลแล้วก็ตาม บางครั้งการถ่ายข้อมูลด้วยความเร็วสูงและด้วยขนาดของแฟ้มที่มีขนาดใหญ่กว่าบัฟเฟอร์สื่อสาร โอกาสที่ข้อมูลจะหายไปมีอยู่มาก เช่น ถ้าใช้ความเร็วในการถ่ายข้อมูล 9600 บิทต่อวินาที สต๊อปบิทเป็น 1 ข้อมูลขนาด 7 บิท และใช้พาริตี้เป็นคู่ในการส่ง 1 ตัวอักษรจะต้องใช้จำนวน 11 บิท เพราะฉะนั้นฝ่ายรับจะต้องอ่านข้อมูลออกมาจากพอร์ทอนุกรมทุก ๆ 110/9600 หรือประมาณ 0.001 วินาที ถ้าเปรียบเทียบเป็นผลพัลส์จะได้ประมาณ 5000 พัลส์นาฬิกา (ความเร็วของสัญญาณนาฬิกาของ IBM KPC - 4.77 Mhz หรือประมาณ 0.2 usec ต่อ หนึ่งพัลส์ 10000/0.2 - 5000) ในเมื่อบัฟเฟอร์ไม่เพียงพอเราจะเป็นจะต้องควบคุมการรับส่ง โดยการบอกให้ฝ่ายส่งหยุดการรับส่งชั่วคราว (XOFF) จนกว่าฝ่ายรับจะจัดการเอาข้อมูลออกจากบัฟเฟอร์สื่อสารหมดเสียก่อนจึงบอกให้ฝ่ายส่งจัดการส่งต่อไป (XON) ในรหัสแอดสกี้ XON มีค่าเท่ากับ 17 และ XOFF มีค่าเท่ากับ 19 ซึ่งเป็นหน้าที่ของผู้เขียนโปรแกรมที่จะต้องจัดส่ง XOFF ออกไปให้ฝ่ายส่งได้รับรู้ก่อนที่บัฟเฟอร์สื่อสารจะเต็มเสียก่อน

สรุปได้ว่าการส่งข้อมูลแบบอะซิงโครนัสนั้นข้อมูลที่ส่งออกไปนั้นไม่จำเป็นต้องต่อเนื่องและช่วงเวลาระหว่างข้อมูลไม่กำหนดแน่นอน แต่สัญญาณที่ส่งออกไปแล้วนั้นมีตัวบอกจุดเริ่มต้นและจุดสุดท้ายของข้อมูลเพิ่มเข้าไปด้วย ซึ่งเรียกว่า Start Bit และ Stop Bit เพื่อบออกฝ่ายรับให้ทราบถึงขอบเขตของการแยกข้อมูลได้ ตามข้อตกลงตามมาตรฐานของสมาคมอุตสาหกรรมอิเลคทรอนิกส์ (EIA) กำหนดมาตรฐาน RS-232C เกี่ยวกับสัญญาณที่ใช้ในการติดต่อสื่อสารให้เป็นที่เข้าใจดังนี้

1) ข้อมูลที่เป็น 1 ให้แทนด้วย -12 โวลท์

2) ข้อมูลที่เป็น 0 ให้แทนด้วย +12 โวลท์

3) Start Bit สัญญาณเปลี่ยนจาก -12 โวลท์ เป็น +12โวลท์

4) Stop Bit โดยจะมีสัญญาณแบบ High เป็นระยะเวลา 1 ช่วงสัญญาณนาฬิกา หรื 1.5 ของสัญญาณนาฬิกา

การกำหนดขอบเขตของอักขระขึ้นอยู่กับข้อตกลงเกี่ยวกับการรับส่งข้อมูลที่เรียกกันว่า โพรโทคอล (Protocol)

โพรโทคอล คือ ข้อตกลงหรือกฎเกณฑ์ที่จะต้องปฏิบัติ ระหว่างฝ่ายรับและฝ่ายส่งในเรื่องต่อไปนี้

ก) ขอบเขตของข้อมูล

ข) การตรวจสอบข้อผิดพลาด และการตอบสนองต่อการผิดพลาด

ค) ลำดับของข้อมูลที่ส่ง

ง) การส่งข้อมูลที่เหมือนกันกับค่าอักขระควบคุม

จ) การควบคุมการรับส่งในสายส่ง

ฉ) กรณีพิเศษเช่น กรณีไม่มีข้อมูลที่จะส่ง ควรจะทำอย่างไร

ช) เวลาที่ต้องการตอบสนองจะต้องทดลองส่งใหม่กี่ครั้งในกรณีที่ไม่มีการตอบสนองกันมา

ซ) การเลิกการติดต่อหรืออาจกล่าวอีกนัยหนึ่งได้ว่า โพรโทคอล หมายถึง ข้อกำหนดคุณลักษณะทั้งกายภาพ และทางไฟฟ้าของอุปกรณ์ที่ถูกออกแบบมาเพื่อใช้สำหรับเชื่อมต่อในงานที่เกี่ยวข้องกับการสื่อสารข้อมูล สาเหตุที่จำเป็นต้องมีการกำหนดคุณลักษณะดังเช่นที่ว่านี้ ก็เพื่อที่จะหลีกเลี่ยงถึงปัญหาบางอย่างที่อาจเกิดขึ้นได้เช่น ความเข้ากันได้ระหว่างอุปกรณ์ประเภทเดียวกันของผู้ผลิตแต่ละราย มีการกำหนดโพรโทคอลขึ้นมาใช้ในการสื่อสารแบบอนุกรมหลายแบบ เช่น Bisync HDLC, SDLE เป็นต้น ลักษณะที่สำคัญของโพรโทคอลเหล่านี้ คือจะต้องมีอักขระพิเศษบอกว่าหมดข้อมูลแล้ว

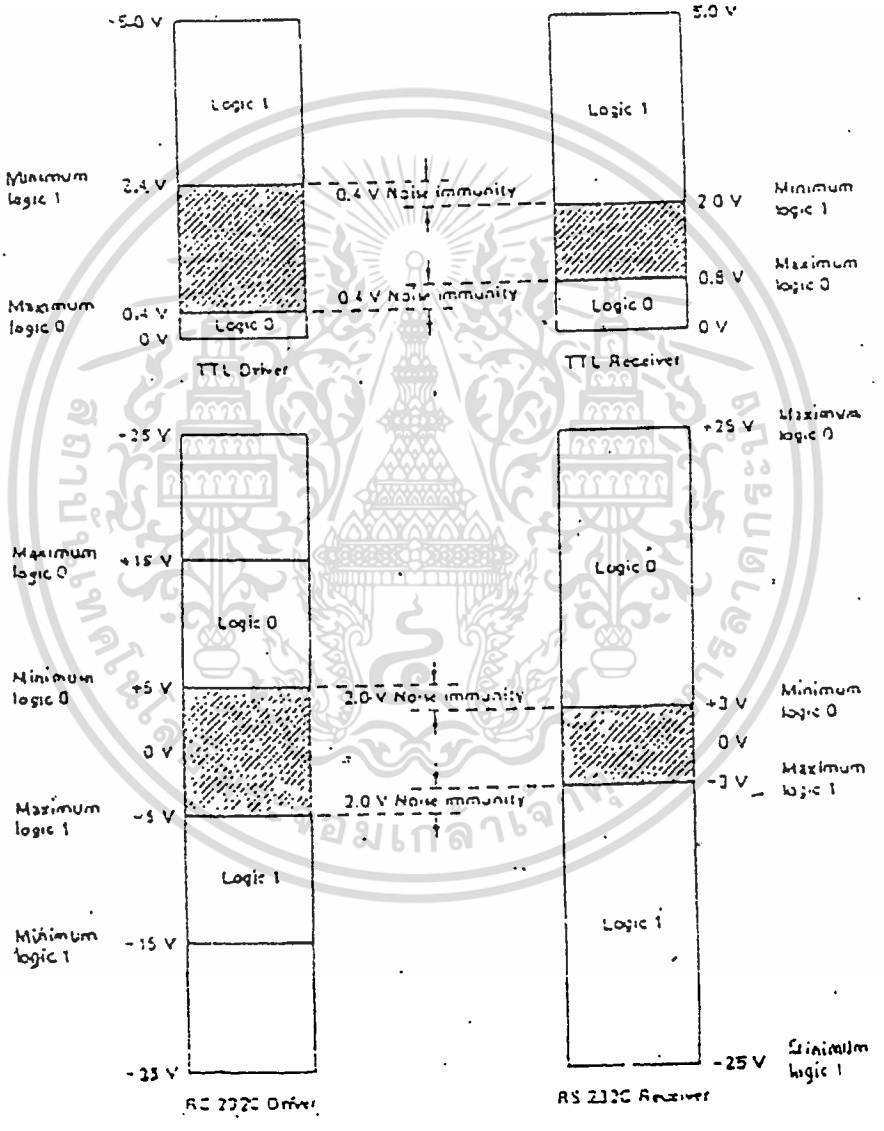
2.4.4 มาตราฐานการสื่อสารข้อมูลตระกูล RS

มาตราฐานตระกูล RS ถูกออกแบบมาจาก EIA (Electronic industry Association) โดย RS ย่อมาจาก Recomend Standard ส่วนตัวเลขที่ตามหลัง RS นั้น เป็นรหัสประจำของมาตราฐานหนึ่ง ๆ ส่วนอักษร A และ C ที่ตามหลังนี้บ่งให้รู้ว่ามาตราฐานนี้ได้รับการพัฒนามาแล้วกี่ครั้ง เช่น RS-232C, RS-449 เป็นต้น

มาตราฐาน RS-232C ใช้ครั้งแรกเมื่อปี พ.ศ. 2505 ซึ่งใช้ในการเชื่อมต่อของอุปกรณ์ระหว่าง อุปกรณ์ตัวส่งและอุปกรณ์ปลายทาง โดยตามศัพท์ที่ใช้งานคอมพิวเตอร์จะหมายถึงเทอร์มินอล (Terminal) และโมเด็มตามลำดับ โดยทางเทอร์มินอลมักจะถูก

เรียกว่าเป็นอุปกรณ์ DTE (Dataterminal Equipment) และโมเด็มก็จะเรียกว่าอุปกรณ์ DCE (Data Communication Equipment)

ในทางงานทฤษฎีข้อกำหนดหลักที่บ่งถึงข้อจำกัดการใช้งานของ RS-232C คือ อัตราความเร็วในการส่งข้อมูลสูงสุด สำหรับสายเคเบิ้ลความยาว 50 ฟุต จะได้ไม่เกิน 19,000 บอต แต่



รูป 2.13 แสดงการเปรียบเทียบการกำหนดลอจิกระหว่างลอจิกแบบ TTL กับระดับลอจิกที่เป็นมาตรฐานของ RS-485

จิกที่เป็นมาตราฐานของ RS-485

ในทางปฏิบัติสายเคเบิ้ลสามารถที่จะให้ยาวกว่านี้ได้แต่ต้องส่งข้อมูลในอัตราความเร็วที่ต่ำกว่านี้แรงดันมาตราฐานของ RS-232C จะใช้ระดับลอจิก 1 มีค่าแรงดันเท่ากับ -12 โวลท์ และลอจิก 0 ด้วยแรงดัน +12 โวลท์

จากรูปที่ 2.13 จะสังเกตุเห็นได้ว่าผลจากการเปรียบเทียบระดับลอจิกของแรงดันมาตราฐานทั้งสองมาตราฐานซึ่ง TTL จะใช้ระดับแรงดันบอกแทนลอจิก และมี Noise Immunity 0.4 Volts แต่มาตราฐาน RS-232C จะแตกต่างไปอย่างมาก โดยใช้ระดับแรงดันสูงขับออกทางเอ้าพุท 5/-12 โวลท์ ซึ่งทำเป็นการต่อใช้งานทั้งนี้ก็เพื่อให้แน่ใจว่าการทำงานกับสายของเคเบิ้ลซึ่งยาวจะมีความถูกต้องและเป็นไปได้สำหรับ Noise Immunity จะกำหนดให้มีค่าเป็น 2 โวลท์ ทั้งนี้ก็เพื่อหลีกเลี่ยงสัญญาณรบกวนจากภายนอก ที่อาจสร้างปัญหาให้กับระบบได้

การเชื่อมต่อระหว่าง RS-232C กับ TTL ระดับกำหนดให้ทางด้านส่ง (Driver) และทางด้านรับ (Receiver) เป็นไปตามรูปที่ 2.13 เราจะใช้ไอซีเบอร์ MC 1488 เป็น Driver) เปลี่ยนระดับลอจิกจาก RS-232C และ MC 1489 เป็น ลอจิกจาก RS-232C เป็นลอจิกระดับ TTL ผ่านทางสายสัญญาณไปยังอุปกรณ์ไอซีซีพ USART

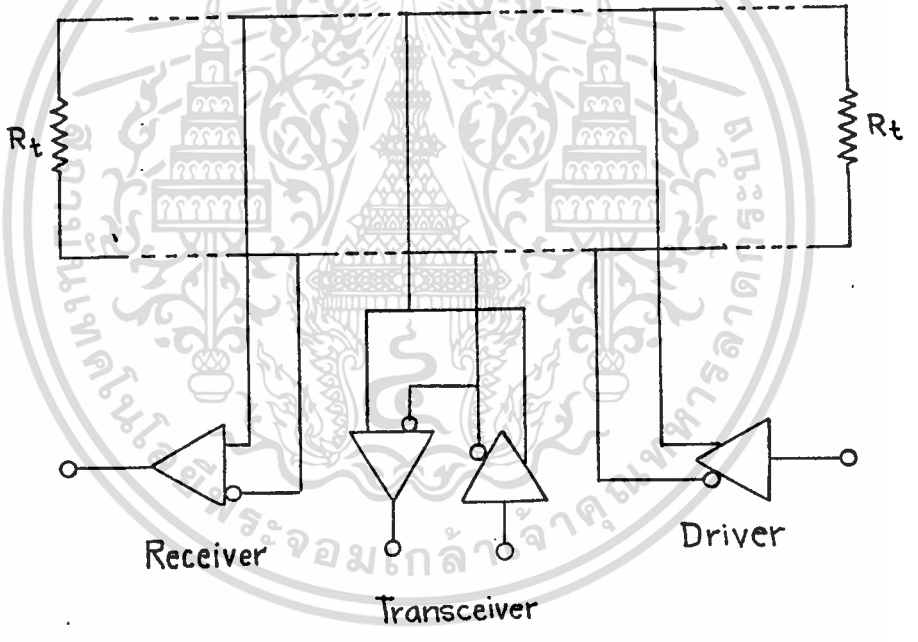
คุณลักษณะทางไฟฟ้าประการหนึ่งที่ถือได้ว่าเป็นมาตราฐานของ RS-232C ก็คือค่าเวลาการเปลี่ยนสถานะ (Transition Time) ก็หมายถึงช่วงเวลาที่ใช้สำหรับการเปลี่ยนจากระดับลอจิกหนึ่งไปสู่อีกลอจิกหนึ่ง จะต้องไมเกิน 4% ของเวลา 1 บิท ดังนั้นถ้าอัตราความเร็วในการส่งเป็น 19,000 บอต จะต้องมีค่าความยาวของเคเบิ้ลนี้ เป็นตัวประกอบ (FACTOR) อย่างหนึ่งที่จะมาจำกัดค่าเวลาของการเปลี่ยนแปลงสถานะทั้งนี้เป็นเพราะสายเคเบิ้ลยิ่งยาวเท่าไรค่าความจุ(Capacitive Load) ของสายเคเบิ้ลก็จะยิ่งมากขึ้นตามอันเป็นผลทำให้เวลาของการเปลี่ยนสถานะถูกทำให้ช้าลง ซึ่งในเรื่องนี้ก็ได้มีการทดสอบกันมาแล้วว่าที่ความเร็วบอดเรด 19,000 บอต นั้นค่าความยาวของสายเคเบิ้ลจะมีได้ไม่เกิน 50 ฟุต

2.4.5 มาตราฐาน RS-422A

RS-422A เป็นมาตราฐานที่ใช้ในการติดต่อสื่อสารแบบอนุกรมชนิด Balance Voltage ซึ่งจะเหมาะสำหรับการส่งข้อมูลที่ความเร็วสูง ๆ ถึง 10 Mds ในสายส่งที่ยาวและมีสัญญาณรบกวนสูง ดังรูปที่ 2.14 แสดงวงจรของ Balance Circuit Mode

จากรูปที่ 2.14 Generator จะมีอิมพีแดนซ์ด้านออก (Output Impedance) ประมาณ100 โอทม์ หรือน้อยกว่าค่าความแตกต่างของแรงดันไฟฟ้า (Differential Voltage) ที่ตกคร่อมระหว่างสายส่งจะมีค่าประมาณ 2-6 โวลท์

สำหรับมาตราฐานการเชื่อมต่อทางไฟฟ้าแบบใหม่ที่มีมาตรฐานว่า RS-423A และ RS422A ก็ดูได้จากรูปที่ 2.11 เช่นกันโดย RS-422 จะใช้การส่งแบบดิฟเฟอเรนเชียล (Differential) ระหว่างตัวส่งและตัวรับทั้งนี้ก็เพื่อที่ตัดปัญหาเกี่ยวกับสายกราวด์ ที่เกิดขึ้นกับ RS-232C ทางด้านรับจะเป็นฝ่ายตรวจจับความแตกต่างของสัญญาณอินพุททั้งสองว่าเป็นบวกหรือลบ ส่วน RS423A ก็จะมีลักษณะคล้ายคลึงกันแต่จะใช้สายส่งสัญญาณเพียงเส้นเดียวโดยให้ทางด้านรับได้รับสัญญาณแบบดิฟเฟอเรนเชียลแทน(การทำเช่นนี้ไม่ถือว่าเกิดจุดกราวด์ร่วมเกิดขึ้น)



รูปที่ 2.14 แสดงการต่อ RS-485 Bus

ตารางที่ 2.5 คุณสมบัติต่าง ๆ ของ RS-485

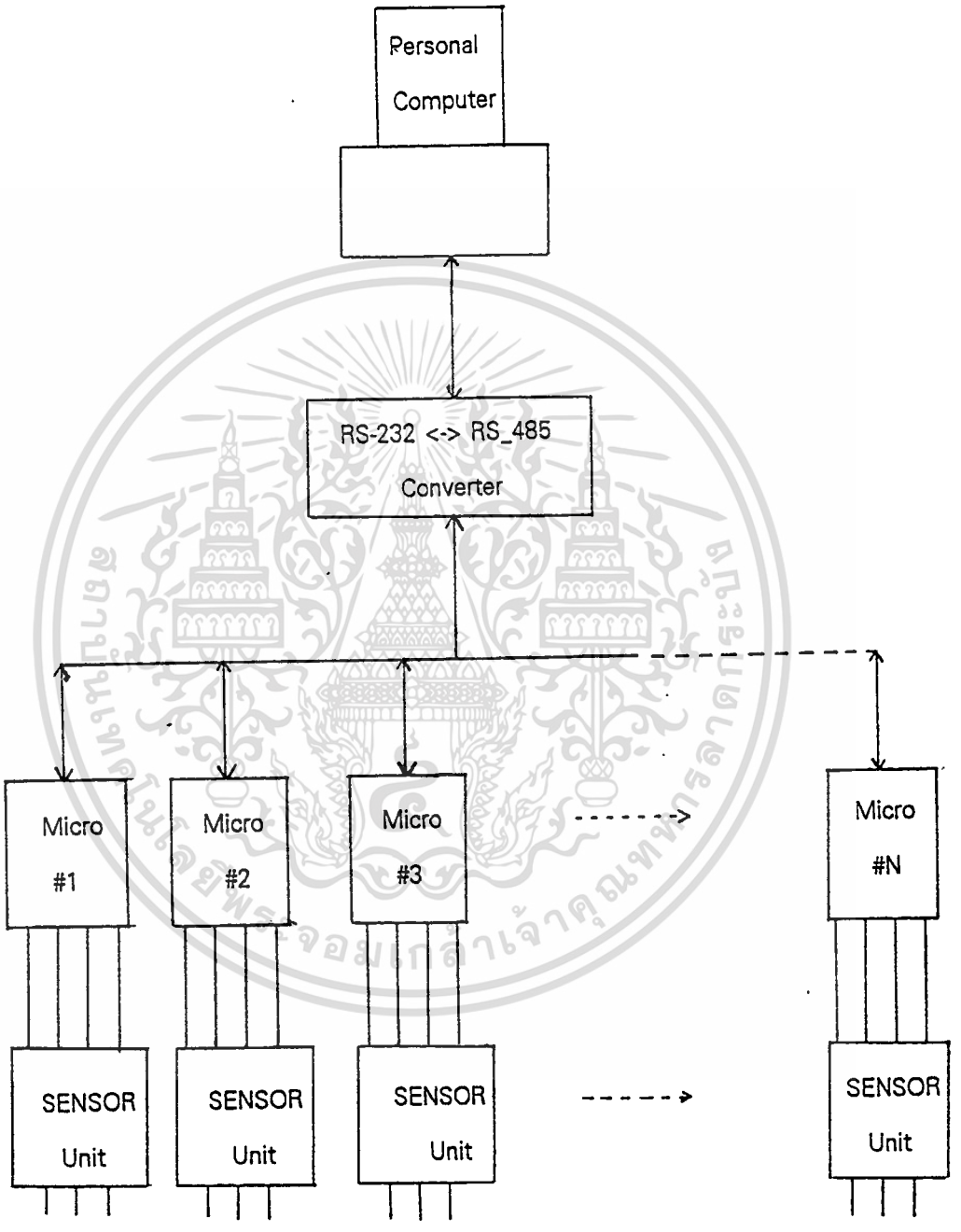
Parameter	RS-422A	RS-485
Maximum cable length	4000	4000 ft
Maximum data rate	12 M	10 M
Dirver output	+ -2 min	+ - 1.5 min
Driver load	100 min	60 min
Driver output short circuit current limit	150 mA to ground	150 mA to Ground 250 mA to -8 V or +12 V
Driver output resistance high impedance state		
- Power On	N/A	120K
- Power Off	60 K	120K
Maximum number of driver and recivers on line	1 driver 10 receivers	32 divers 32 receivers
Receivers sensitivity	+ - 200 mV	+ - 200mV
Receivers input resistance	4 K	12 K
Maximum common-mode voltage	+ 6V. -0.25 V	+12 V. -7 V.

บทที่ 3
การออกแบบและการจัดสร้างโครงงาน

3.1 คำนำ

ในปัจจุบันได้มีการนำเอา Computer มาประยุกต์ใช้งานกันอย่างแพร่หลายเช่นในด้านการออกแบบการคำนวณ การเก็บข้อมูล การประมวลผลข้อมูล เป็นต้น ปัจจุบันหลาย ๆ หน่วยงานและหลาย ๆ บริษัทนำเอาระบบ Network เข้ามาใช้เพราะสามารถเพิ่มประสิทธิภาพในการทำงานความถูกต้องแม่นยำของข้อมูล และความสะดวกรวดเร็วในการตัดสินใจแต่การนำเอา Computer มาต่อเป็น Network นั้นเราจำเป็นต้องรู้ระบบของ Communication Interface เพราะจุดประสงค์การนำไปใช้งานอาจจะต่างกัน โดยเฉพาะในด้านวิศวกรรม การออกแบบ และการควบคุม ดังนี้นเราจะต้องเลือกระบบ Communication Interface ให้เหมาะสมกับการนำไปใช้งาน เช่น ระบบ RS-232, RS-423, RS-422, RS-485, IEEE-488 หรือระบบ GPIB ซึ่งแต่ละระบบจะมีข้อดีข้อเสียและขีดจำกัดต่างกันเช่นความยาวของสาย Cable, Maximun Baaud, Signal Level Driver, Output Current, Input Resistance, Input Voltage Range จำนวน driver และ receiver ที่สามารถต่อใช้งานได้ภายใน 1 Line เป็นต้น ถ้าเลือกระบบ Interface แบบ Multipoint Communication คือ ภายใน 1 Line สามารถต่อ Driver และ Reciver ได้หลายตัว ดังนั้นจำเป็นต้องมี Software ในรูปแบบของ Protocal เพื่อเป็นตัวจัดระบบการรับส่งข้อมูลและ Protocal ที่จะใช้ก็ควรจะคำนึงถึงการใช้งานที่สามารถนำไปใช้ร่วมกับผู้อื่นได้ด้วยในโครงการนี้เลือกใช้การสื่อสารข้อมูลในระบบ HDLC

รายละเอียดที่จะจัดสร้างโครงสร้างนั้นก็พอจะแบบออกได้เป็น Block Diagram ได้ดังรูป



รูปที่ 3.1 แสดง Block Diagram ระบบควบคุมแสดงผลระยะไกลโดยใช้ระบบ HDLC

3.2 การออกแบบสร้างเครื่อง Microcontroller

การทำงานของเครื่อง Microcontroller จะใช้ CPU 8031 เป็นตัวควบคุมระบบ และมี Eprom #2716 (2K Bytes.) เป็น Monitor Program เพื่อควบคุมการทำงานของ CPU 8031 และมี S-RAM #6116 (2K Bytes.) ทำหน้าที่เป็น Buffer เพื่อเก็บข้อมูลในขณะที่ CPU ทำการประมวลใช้ IC #DS75176 ทำหน้าที่เป็นตัวรับส่งข้อมูลในระบบ RS-485

ในการเขียนโปรแกรมในโครงงานนี้ (เราสามารถจะใช้รูปแบบของการติดต่อสื่อสารของ HDLC ซึ่งในส่วนนี้เราจะกว่าถึงเฉพาะในส่วนที่สำคัญ) และเหมาะสมที่จะนำไปเขียนโปรแกรมเท่านั้น ซึ่ง Protocal HDLC จะแบ่งออกได้เป็น 2 Lrvel ได้ดังต่อไปนี้

Level 1 คือ Phisical layer เป็นระดับของการเชื่อมต่อระหว่างอุปกรณ์ทางไฟฟ้าทั้ง 2 สถานี เช่น การใช้สาย Transmitsion Line เป็นต้น ซึ่งในโครงงานนี้เราได้ใช้การเชื่อมต่อเป็นแบบ RS-485

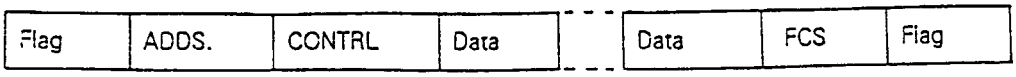
Level 2 คือ Data Link เป็นระดับการติดต่อแลกเปลี่ยนข้อมูลกันระหว่าง 2 สถานีโดยจะใช้ Protocal ชนิด HDLC (High Level Data Link CONTROL) เป็นรูปแบบมาตรฐานสากลซึ่ง Protocal ชนิด HDLC จะมีรูปแบบดังนี้คือ การส่งข้อมูลแบบ HDLC จะเรียกว่าเป็น Flag (F) ขนาด 8 Bit, Address(A) ขนาด 8 Bit, Information (I) ขนาด 8 Bit (สามารถขยายขนาดได้ครั้งละ 8 Bit) Frame Chake Sequence (FCS) ขนาด 16 Bit และ Flag (F) ขนาด 8 Bit

HDLC Frame แบ่งออกได้เป็น 3 ประเภทคือ

- Nonsequence Frame (Managment Frame)
- Supervisory Frame
- Information Frame

Nonsequence Frame จะเป็น Frame ของคำสั่งและการตอบสนองคำสั่ง (Command และ Responde) ซึ่งใช้สำหรับติดต่อและควบคุมลูกข่ายและบอกถึงการผิดพลาดในการรับส่งข้อมูล Nonsequence Frame นี้สามารถส่งจากสถานีแม่ไปยังลูกข่ายหรือจากสถานีลูกข่ายไปยังสถานีแม่ได้ โดยที่ผลการตอบสนองจะเกิดขึ้นที่สถานีต้นทาง

Supervisory Frame เป็น Frame ที่ใช้หน้าที่สำหรับส่งข้อมูลไปยังปลายทาง HDLC Frame



รูปที่ 3.2 แสดงรูปแบบ HDLC Frame มาตรฐาน

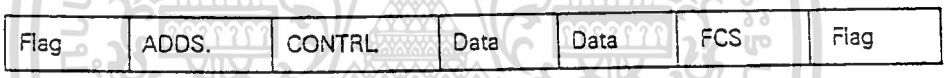
- Flag จะเป็นตัวบอกจุดเริ่มต้นและจุดสิ้นสุดของ HDLC Frame ซึ่ง Flag จะประกอบไปด้วยข้อมูลขนาด 8 Bit มีรหัสคือ 7E10

- Address จะประกอบไปด้วยข้อมูลขนาด 16 Bit ซึ่งทำหน้าที่เป็นตัวระบุตำแหน่งที่ต้องการจะติดต่อด้วย

- Control จะเป็นตัวกำหนดชนิดของ Frame ที่กำลังส่งอยู่ ประกอบด้วยข้อมูล 8 Bit

- Data จะเป็นข้อมูลที่รับส่งกัน ข้อมูลมีความยาวไม่แน่นอน

- FCS (Frame Check Sequence) จะใช้สำหรับตรวจเช็คข้อมูลไม่ให้เกิดความผิดพลาดโดยใช้การตรวจแบบ Summing Check



รูปที่ 3.3 แสดงรูปแบบ HDLC Frame ซึ่งใช้ในโครงงานนี้

- Address จะประกอบไปด้วยข้อมูลขนาด 8 Bit ซึ่งทำหน้าที่เป็นตัวระบุตำแหน่งที่ต้องการจะติดต่อด้วย (ในที่นี้สามารถติดต่อได้ 256 สถานี)

- Control จะเป็นตัวกำหนดชนิดของ คำสั่งที่ส่งระหว่าง PC และ Microcontroller ประกอบด้วยข้อมูล 4 Bit รายละเอียดแสดงดังตาราง

- Data จะเป็นข้อมูลที่รับส่งกันระหว่าง Microcontroller และ PC ในโครงงานนี้มีข้อมูล Data 18 Bit

-FCS (Frame Check Sequence) จะใช้วิธีนำข้อมูลทั้งหมดบวกกันแล้วหารเอาเศษด้วย FF16 สำหรับตรวจเช็คข้อมูลไม่ให้เกิดความผิดพลาด

3.3 <u>การออกแบบและจัดสร้างโปรแกรมของ Microcontroller</u>

เมื่อทราบรูปแบบของการรับส่งข้อมูลแล้วจึงทำการเขียนโปรแกรมโดยเริ่มจากการ Initial 8031 และ 8255 ก่อน

3.3.1 การเขียนโปรแกรม Initial

1) การ Initial 8031

การ Initial 8031 นั้นเราต้องการใช้ 8031 ทำงานในการรับส่งข้อมูลแบบอนุกรมโหมด 1 (Serial Data Mode 1 : Standard UART)

ก) SCON : (Serial Port Control Register)

SM0	SM1	SM2	REN	TB8	RB1	TI	RI

รูปที่ 3.4 แสดงบิทต่าง ๆ ของรีจิสเตอร์ SCON

การเลือกโหมดการทำงานจะต้องเลือกควบคุมค่า MO, M1 ให้ถูกต้องตามตารางที่ 3.3 REN ถ้าบิทนี้เป็น "0" จะส่งได้ทางเดียวถ้าเป็น "1" จะทำงานได้ทั้งรับส่ง

ตารางที่ 3.1 การเบลือกโหมดการทำงานบ็อดเรท SCON โดยใช้ MO และ M1

MO	M1	โหมด	คำอธิบาย	บ็อดเรต
0	0	0	ชิพตรีจิสเตอร์	ความถี่สัญญาณนาฬิกา/2
0	1	1	Usart 8 Bit	เลือกค่าได้
1	0	2	Usart 9 Bit*	ความถี่สัญญาณนาฬิกา/2 หรือ 64
1	1	3	Usart 9 Bit*	ค่าเลือกได้

ดังนั้นจึงให้ Scon-50H คือทำงานในโหมด Usart 8 Bit บ็อดเรทสามารถที่จะเลือกเอาได้

ข) PCON : (Power Mode Control Register)

SMOD	-	-	-	GF1	GF0	PD	IDL

รูปที่ 3.5 แสดงบิทต่าง ๆ ของรีจิสเตอร์ PCON

รีจิสเตอร์ตัวนี้ใช้งานเพียงบิทเดียวคือ SMOD เป็น "0" ค่า K เป็น 1 SMOD เป็น "1" ค่า K จะเป็น 2 ดังนั้นจึงให้ค่า PCON = OOH (K=0)

ค) IE : (Interrupt Enable Register)

เนื่องจากไม่มีการใช้ Interrupt เราจึงให้ IE = 00

ง) IP : (Interrupt Priority Register)

เนื่องจากไม่มีการใช้ Interrupt เราจึงให้ IP = 00

จ) TMOD : (Timer Mode Control Register)

GATE	C/T	M1	M0	GATE	C/T	M1	M0

รูปที่ 3.6 แสดงบิทต่าง ๆ ของรีจิสเตอร์ TMOD

ในการใช้งานพอร์ทอนุกรมจะใช้งานรีจิสเตอร์นี้บางบิท คือ M1, MO เป็นตัวเลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์โดยมีรายละเอียดดังตารางที่ 2.3 จะใช้ไทเมอร์ 1 ในการสร้างบ็อดเรทโดยใช้ไทเมอร์

โหมด 2 โดยต้องกำหนดค่าการทำงานดังนี้

GATE = "0"

C/T = "0" (โหมดไทเมอร์)

MO = "0"

M1 = "0" (18 บิท ออโตรีโหลด)

ตารางที่ 3.2 การเลือกโหมดการทำงานของไทเมอร์/เคาเตอร์โดยใช้ MO,M1

M0	M1	โหมด	คำอธิบาย
0	0	0	13 บิทไทเมอร์
0	1	1	13 บิทไทเมอร์
1	0	2	8 บิทออโตรีโหลด
1	1	3	ไทเมอร์/เคาน์เตอร์ 8 บิท / ชุด

จึงให้ TCON - 2OH

เพราะต้องการให้โหมดเคารน์เตอร์ที่เป็น 8 บิท ออโตรีโหลด

การคำนวณหาค่าของรีจิสเตอร์ TH1

สำหรับโครงงานนี้ซึ่งใช้บ็อดเรทเท่ากับ 9600 bps. (Bit Per Second) สำหรับค่าที่เซ็ทไว้แล้วมี

K = ๐ (SMOD)

Oscillator Frequency - 11.09 Mhz

ต้องการบ็อดเรท = 9600 bps.

จาก

fbaud = [(2k * osc freq.)/(32d * 12d * (256d -TH1)]

TH1 = 256D - [(2K * osc freq.)/(32d *12D * fbaud)]

ดังนั้น

TH1 = 256d - [12* 11.092 Mhz)/(32d * 12d * 9600d)]

TH1 = OFDh

ฉ) TCON : (Timer Control Register)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

รูปที่ 3.7 แสดงบิทต่าง ๆ ของรีจิสเตอร์ TCON

รีจิสเตอร์นี้มีบิทที่ต้องใช้คือ TRO และ TR1 เป็นตัวควบคุมไทเมอร์และเคาน์เตอร์โดยถ้าให้เป็น "0" หมายถึง "0" ฉะนั้นหลังจากเซ็ทค่าต่าง ๆ หมดแล้ว เราก็จะเปิดเคาน์เตอร์โดยใช้คำสั่ง SETB TRO หรือ SETB TR1 คือทำให้ TR1 เป็น "1"

2) การ Initial 8255

ในโครงงานนี้ต้องการให้ 8255 ทำงานในโหมด 0 คือ จะทำงานในลักษณะเป็นรีจิสเตอร์อินพุท/เอ้าท์พุต ดังนั้นจึงต้องส่งคำสั่งควบคุม (Control Word) ให้กับรีจิสเตอร์ควบคุมก่อน

D7	D6	D5	D4	D3	D2	D1	D0

รูปที่ 3.8 Control Wold Register

D7 : Mode Select Flag = "1" ACTIVE

D6 D5 : Mode Selection

= 00 : Mode 0

= 01 : Mode 1

= 1X : Mode 2

D4 : Port A

= 0 : Output

= 1 : Input

D3 : Port C (Upper)

= 0 : Output

= 1 : Input

D2 : Mode Selection Of Port B

= 0 : Mode 0

= 1 : Mode 1

D1 : Port B

= 0 : Output

= 1 : Input

DO : Port C (Lower)

= 0 : Output

= 1 : Input

ดังนั้นเราจึงให้ค่า Control Word Register - 8Ah คือ Port A, C เป็น Port เอ้าท์พุทส่วน Port B เป็น Port อินพุท

3.3.2 การเขียนโปรแกรมหลัก

ในส่วนของ Microcontroller นี้จะทำหน้าที่ถอดรหัสเพื่อแปลงออกมาเป็นโหมดคำสั่งการทำงานได้ 6 โหมดด้วยกันคือ

โหมด 1 เป็นการนำข้อมูลที่ PC ส่งให้ไปเขียนลงบนพอร์ทของ 8255 (Port A, C)

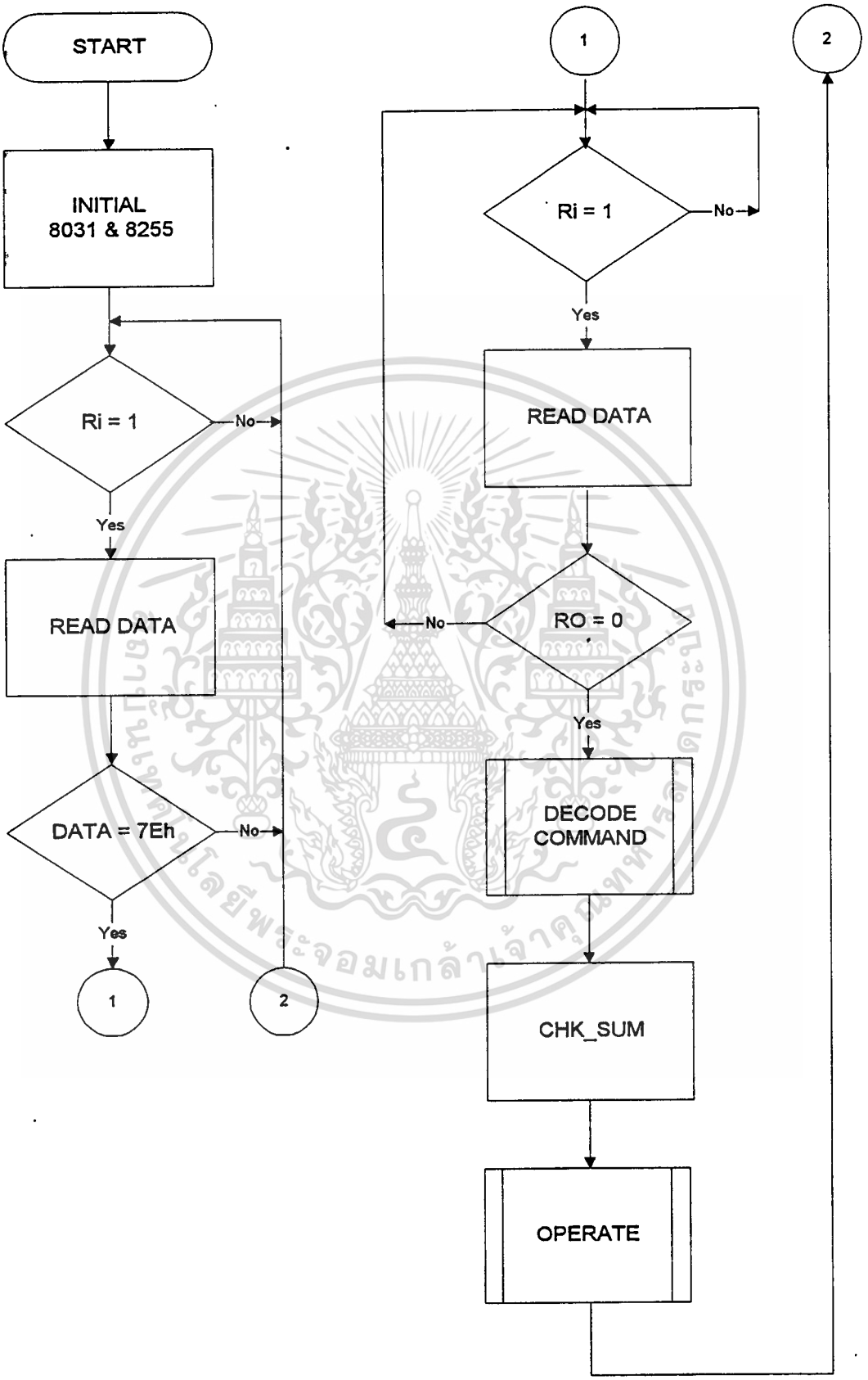
โหมด 2 เป็นการอ่านข้อมูลมาจากพอร์ท 8255 ทั้ง 3 พอร์ท แล้วส่งไปให้ PC

โหมด 3 เป็นการนำข้อมูลที่ PC ส่งให้ไปเขียนลงบนพอร์ทของ 8255 (Port A, C) แต่จะกระทำงานได้หลายเครื่องพร้อม ๆ กัน

โหมด 4 เป็นการอ่านข้อมูลจาก 8255 ที่ต่ออยู่กับสัญญาณฉุกเฉิน (Port C)

โหมด 5 เป็นการรีเซ็ตวงจรอิเล็กทรอนิกส์

โหมด 6 เป็นการนำข้อมูลที่ PC ส่งให้ไปเขียนลงบนพอร์ทของ 8255 (Port A, C) แต่จะกระทำงานทุกเครื่องพร้อมๆ กัน



รูปที่ 3.9 Flow Chart **การทำงานของโปรแกรมบน** Microcontroller

รายละเอียดของโปรแกรม ในแต่ละสับรูทีน

- Main Program

```
MAIN :
        -MOV IE # OOH          : อินนิเซี่ยน 8031
        MOV IP, #0             :
        MOV PCON, #OOH         :
        MOV SCON, #052H        : สื่อสารแบบอนุกรม โปมด 1, 8 บิต
        MOV TMOD, #2OH         :
        MOV TH1, #OFDH         : บ็อดเรท - 9600 bps.
        MOV TCON,#4OH          : เปิดเคาน์เตอร์
        MOV DPTR, #8003H       : CONTROL PORT TO 8255
        MOV A,#8AH             : PORT A, C -OUTPUT
        MOVX @DPTR             : PORT B - INPUT
MAIN2:
        LCALL REC              : เรียกสับรูทีน REC รับข้อมูล
        MOV DPTR, #2002H       : เช็คว่าเป็นข้อมูลจาก PC ส่งมาหรือไม่
        MOV A, @DPTR           :
        ANL A,#OCH             : โดยถ้า PC ส่งมา ข้อมูลต้องเท่ากับ "11"
        CJNE A,#OCH,MAIN2      : ถ้าไม่ใช่ข้อมูลจาก PC ให้กลับไปรอรับข้อมูล
                                 ใหม่
        LCALL CHK_INS          : เรียกสับรูทีน CHK_INS ตรวจสอบว่าเป็นคำ
                                 สั่งใดแล้วนำไปทำงานตามคำสั่งนั้น ๆ
        SJMP MAIN2             : กลับไปรอรับคำสั่งใหม่
        END
```

เป็นส่วนที่เป็นโปรแกรมหลัก (MAIN PROGRAM) จะเริ่มโดยทำการ Initial CPU 8031 และ8255 โดยจะให้ 8255 ทำงานในโหมด 0 (BASIC Input Output REGISTER) โดยให้ Port A และ Port C ทำงานเป็น Output Port ส่วน Port B นั้นจะเป็น Input Port

หลังจากทำการ Initial ทั้ง CPU และ 8255 แล้ว จะรอรับข้อมูลโดยเรียก สับรูทีน REC (รับข้อมูล HDLC Frame)

พร้อมข้อมูลได้ครบ Frame แล้วจะทำการ check ว่าใช้ข้อมูลจาก PC ส่งมาให้หรือเปล่าโดยจะเช็คจากข้อมูลส่วนของ CONTROL (c2,c3) ต้องเป็น "11" ซึ่งเป็นการส่งข้อมูลจาก PC ไปให้ Microcontroller หากไม่ใช่ "11" จะเป็นการส่งข้อมูลจาก PC

จากนั้นก็จะทำการเช็คคำสั่งว่าจะให้ทำอะไร โดยการเรียกสับรูทีน CHK_IN ซึ่งในสับรูทีนนี้ก็จะรวมการทำงานตามที่ PC สั่งมาด้วย

- สับรูทีน REC : (Recive Data HDLC Frame)

```
REC:
        MOV dptr, #8002h          : ควบคุม IC DS785176 ให้เป็นการรับข้อมูล
        MOV a,#OOh                : โดยการให้ PC3 = "0"
        MOV @dptr,a
        MOV RO, #07H
        MOV DPTR, #2000H          : แอดเดรสเริ่มต้นที่เก็บข้อมูล
RX1:
        LCALL RETRIC
        JNB   RI,RX1
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#7EH, RX1         : ตัวแรกถ้าไม่ใช่ 7'Eh จะไม่รับ
        SJMP  RX
RX2_7
        JNB   RI, RX2_7
        CLR   RI
        MOV   A, SBUF

RX:
        MOVX @DPTR,A
        INC   DPTR
        DJNZ  RO, RX2_7
        RET
```

สับรูทีนนี้จะเป็นการรับข้อมูลที่ส่งมาจาก PC เข้ามาเก็บไว้ในหน่วยความจำที่แอดเดรสตั้งแต่ 2000h-20006h จำนวน 7 byte ด้วยกัน ซึ่งก็เท่ากับ 1 Frame ของระบบ HDLC

- สับรูทีน CHK_INS : (Check Instruction)

```
CHK_INS:
        MOV   DPTR,#2000H
        MOVX A,@DPTR                : นำไบต์คำสั่งออกมาเช็ค
        ANL   A,#OFOH               : A - คำสั่ง
        CJNE  A,#10H, CROSS1        : Write Port
        LCALL CHK_ADDR              : เรียกสับรูทีนเช็คแอดเดรส
        CJENE R1,#OFFH, EX_CI       : ถ้าเป็นแอดเดรสที่ต้องทำงาน R1-
                                    : OFFh ถ้าไม่ใช่ให้ออกจากสับรูทีน
        LCALL INS1                  : เรียกสับรูทีนเพื่อ
        SJMP  EX_Ci
CROSS1:
        CJNE  A,#2OH, CROSS2        : Read Port
        LCALL CHK_ADDR              : เรียกสับรูทีนเช็คแอดเดรส
        CJNE  R1,#OFFH,Ex_CI        : ถ้าเป็นแอดเดรสที่ต้องทำงาน R1-
                                      OFFh
                                    : ถ้าไม่ใช่ให้ออกจากสับรูทีนนี้
        LCALL INS 2                 : เรียกสับรูทีนเพื่อคำสั่งที่ 2
        SJMP EX_CL
CROSS2:
        CJNE  A, #30H, CROSS3       : ADDR-Data(b) Working
        LCALL CHK-AD2               : เรียกสับรูทีนเช็คแอดเดรส
        CJNE  R1,#OFFH, EX_CI       : ถ้าเป็นแอดเดรสที่ต้องทำงาน R1-
                                      OFFh
                                    : ถ้าไม่ใช่ให้ออกจากสับรูทีนนี้
        LCALL INS3                  : เรียกสับรูทีนเพื่อคำสั่งที่ 3
        SJMP  EX_CI
CROSS3:
```

```
        CJNE   A,#40H, CROSS4          : Emergency
        ICALL  INS4                    : เรียกสับรูทีนเพื่อคำสั่งที่ 4
        SJMP EX_CI
CROSS4:
        CJNE   A,50H< CROSS            : Reset Electic Lelay
        LCALL CHK_ADDR                 : เรียกสับรูทีนเช็คแอดเดรส
        CJNE   R1,#OFFH,EX_CI          : ถ้าเป็นแอดเดรสที่ต้องทำงาน R1-
                                         OFFh
                                       : ถ้าไม่ใช่ให้ออกจากสับรูทีนนี้
        LCALL INS5                     : เรียกสับรูทีนเพื่อคำสั่งที่ 5
        SJMP   EX_CI
CROSS5:
        CJNE   A,#060H, EX_CI          : Working All Station
        LCALL INS6                     : เรียกสับรูทีนเพื่อคำสั่งที่ 6
EX_CI:
        RET
```

สับรูทีนนี้จะทำการเช็คคำสั่งที่ PC ส่งมาว่าจะให้ Microcontroller ทำอะไรอย่างไร โดยจะตรวจสอบลจากข้อมูลใน Control Bit7-Bit4 (4 Bit บนของข้อมูลในแอดเดรส 20025h) ซึ่งมีรายละเอียดดังตารางที่ 3.3 นี้

ตารางที่ 3.3 แสดงคำสั่งที่ใช้ส่งกับ Microcontroller กับ PC

Control bit7-bit4	Operate
1	Write Port A, C
2	Read Port A,B,C
3	Write Port / Group (Address-Data B)
4	Emergency
5	Reset Electrical Relay
6	Working All Station

หากเป็นคำสั่งใดก็จะทำงานดังคำสั่งนั้น แต่ว่าจะต้องทำการเช็คแอดเดรส ก่อนว่าตรงกับแอดเดรสที่ PC สั่งให้ทำงานหรือไม่ โดยการเรียกใช้สับรูทีน CHK_ADDR ในกรณีที่ทำงานเครื่องเดียว และสับรูทีน CHK_ADD2 ในกรณีที่ทำงานเป็นกลุ่มหากเป็นแอดเดรสที่ต้องทำงานก็จะเรียกไปที่สับรูทีน ที่จะทำงานตามคำสั่งต่างๆ

- สับรูทีน CHK_ADDR : (Check Only 1 Address)
- สับรูทีน CHK_ADD2 : (Check Group Address)

```
CHK_ADDR:
        MOV DPTR, #2001H
        MOVX A, @DPTR
        CJNE A,P1, NOT_ADDR          : เปรียบเทียบ address ถูกต้องหรือไม่
        MOV R1, #OFFH                : ถ้าถูก เซ็ท R1 - OFFH
        SJMP BE_ADDR
NOT_ADDR:
        MOV R1, #00H                 : ถ้าไม่ถูก เซ็ท R1 - 00H
BE_ADDR:
        RET
;-------------------------------------------------------------------------------------------
CHK_AD2:
        MOV DRTR, #2001H
        MOVX A, @DPTR
        MOV R7, A                    : R7 - แอดเดรสต่ำ
        MOV A, P1                    : A - P1
        MOV R6, A                    : R6 - P1 (แอดเดรสจริง)
        CLR   C
        SUBB A, R7                   : ถ้า P1<ADDR ไม่ใช่แอดเดรส
        JC      NOT_ADD
        MOV   DPTR, #2004H
        MOVX A, @DPTR                : A - ข้อมูล B
        CLR   C
        SUBB A, R6
        JC      NOT_ADD              : ถ้า ข้อมูล B<P1 จะไม่ใช่แอดเดรส
```

```
        MOV   R1, #OFFH
        SJMP  IS_ADD
NOT_ADD:
        MOV   R1, #00H
IS_ADD:
        RET
```

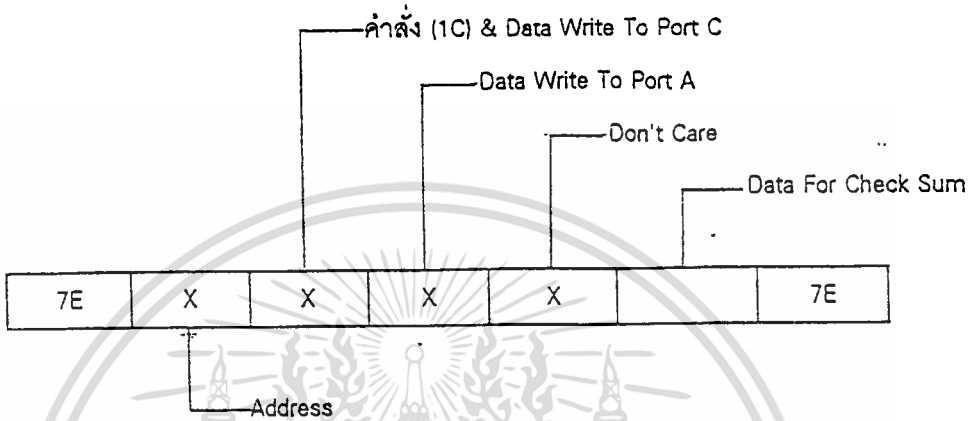
สับรูทีนทั้ง 2 ตัวนี้จะเป็นการเช็คแอดเดรสว่าต้องทำงานด้วยหรือไม่ โดยที่สับรูทีน ADDR จะเป็นการเช็คแอดเดรสเดียว แต่สับรูทีน ADD2 จะเป็นการเช็คแอดเดรสเป็นกลุ่ม โดยถ้าเป็นแอดเดรสที่จะต้องทำงานจะเซ็ตรีจิสเตอร์ R1-"1" แต่ถ้าไม่ใช่จะให้ R1 - "0" สับรูทีนที่เรียกใช้จึงต้องดูค่าของรีจิสเตอร์ตัวนี้

- สับรูทีน INS1 : (Write Port)

```
INS1:
        LCALL CHK_SUM              : เรียกสับรูทีน CHK_SUM
        CJNE R1,#OFFH, ERR1        : ถ้า CHK_SUM ผิดให้ส่งข้อมูล
INS6_1:
        MOV DPTR,#2003H            : นำข้อมูลจากแอดเดรส 2003h
        MOV A, @DPTR
        MOV DPTR, #8000H
        MOVX @DPTR, A              : เขียนข้อมูลลงบน Port A
        MOV DPTR, #2002H           : นำข้อมูลจากแอดเดรส 2003h
        MOVX A, #DPTR
        MOV DPTR, #8002H
        MOVX @DPTR, A              : เขียนข้อมูลลงบน Port A
        SJMP EX_INS1
ERR1:
        LCALL ERR_TRN              : เรียกสับรูทีน ERR_TRN เพื่อส่ง
                                   : ERR ไปบอก PC
EX_INS1:
        RET
```

สับรูทีนนี้จะเป็นการนำข้อมูลจากที่ PC ส่งให้มาไปเขียนลงบน 8255 ใน PortA และ Port C แต่แรกสุดสับรูทีนนี้จะต้องทำการเช็คข้อมูลที่ส่งมาว่าถูกต้องหรือเปล่าก่อน

โดยเรียกสับรูทีน CHK_SUM หากไม่ถูกต้อง (R1<>offh) โปรแกรมก็จะสั่งให้ส่งข้อมูลกลับไปให้ PC เพื่อบอกให้ PC ทราบว่าเกิดความผิดพลาดขึ้น โดยรูปแบบของข้อมูลที่รับมาเป็นดังนี้



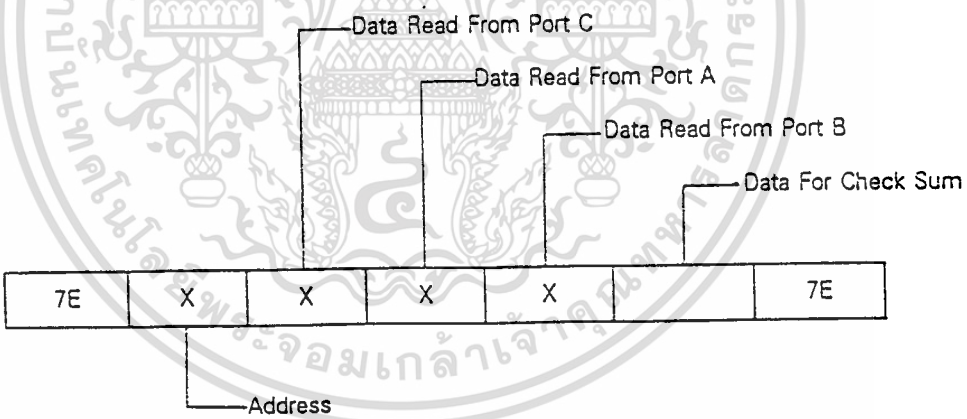
รูปที่ 3 แสดง Frame Format ของข้อมูล

- สับรูทีน INS2 : (Read Port)

```
INS2:
        LCALL CHK_SUM           : เรียกสับรูทีน CHK_SUM
        CJNE R1,#OFFH, ERR2     : ถ้า CHK_SUM ผิดให้ส่งข้อมูล
                                : กลับไปบอก PC
        MOV DPTR, #8OO2H        : อ่านข้อมูลจาก Port c
        MOVX A,@DPTR
        ANL A,#OF3H             : ทำให้ c2,c3 -"00"
        MOV DPTR,#2OO2H         : นำข้อมูลไปเกิบไว้ในแอดเดรส
                                : 2OO2h
        MOVX @DPTR,A
        MOV DPTR, #8OO1H        : อ่านข้อมูลจาก Port B
        MOVX A,@DPTR
        MOV DPTR,#2004H         : นำข้อมูลไปเก็บไว้ในแอดเดรส
                                : 2OO2h
        MOVX @DPTR,A
        MOV DPTR,#8OOOH         : อ่านข้อมูลจาก Port A
```

```
        MOVX a,@DPTR
        MOV DPTR,#2003H          : นำข้อมูลไปเก็บไว้ในแอดเดรส
                                 : 2002h
        MOVX @DPTR,A
        LCALL TRN                : ส่งข้อมูลไปให้ PC
        SJMP EX2
ERR2:
        LCALL ERR_TRN            : ส่ง ERR ไปบอก PC
X2:
        RET
```

สับรูทีนนี้ก็จะมีลักษณะคล้าย ๆ กัน INS1 แต่จะเป็นการอ่าน Port แทนการเขียน Port โดยจะอ่านข้อมูลที่ Port A, B และ C มีการเช็คความผิดพลาดของข้อมูลเช่นกัน โดยรูปแบบของข้อมูลที่จะส่งกลับไปให้ PC เป็นดังนี้

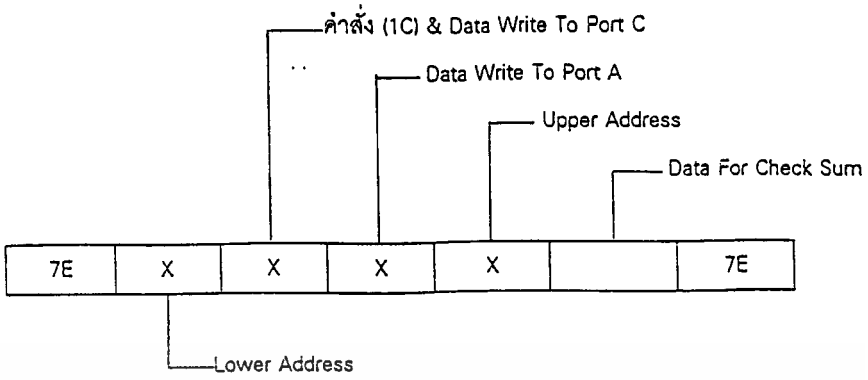


รูปที่ 3.10 แสดง Fram Format ของข้อมูล

- สับรูทีน INS3 : (Write Port/Group)

```
INS3:
        LCALL INS6_1             : เรียกใช้สับรูทีนเดียวกัน INS1
        RET
```

สับรูทีนนี้ทำงานเหมือนสับรูทีน INS1 ต่างกันที่ทำงานเป็นกลุ่ม (หลาย STATION) เท่านั้น โดยรูปแบบของข้อมูลที่รับมาจาก PC เป็นดังนี้

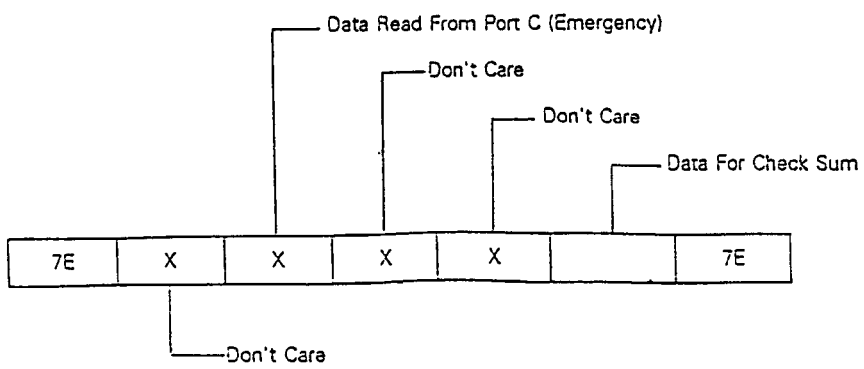


รูปที่ 3.11 แสดง Fram Format ของข้อมูล

- สับรูทีน INS 4 : (Emergency)

```
INS 4:
        MOV   DPTR,#8OO2H          : อ่านข้อมูลจาก Port C
        MOVX A,@DPTR
        ANL   A,#OFOH              : เฉพาะ Port Ch
        JZ    EX4
        MOV DPTR,#2002H            : นำข้อมูลไปเก็บที่แอดเดรส 2OO2h
        MOVX @DPTR,A
        LCALL TRN                  : ส่งข้อมูลไปยัง PC
        RET
```

สับรูทีนนี้จะไปอ่านข้อมูลจาก Port C บนของ 8255 (ซึ่งต่อไว้กับส่วนที่เป็นการฉุกเฉิน) หากมีส่วนใดผิดปรกติ 8031 ก็จะส่งข้อมูลกลับไปให้ PC เพื่อแจ้งเตือนให้ทราบ โดยรูปแบบของข้อมูลที่จะส่งให้ PC เป็นดังนี้

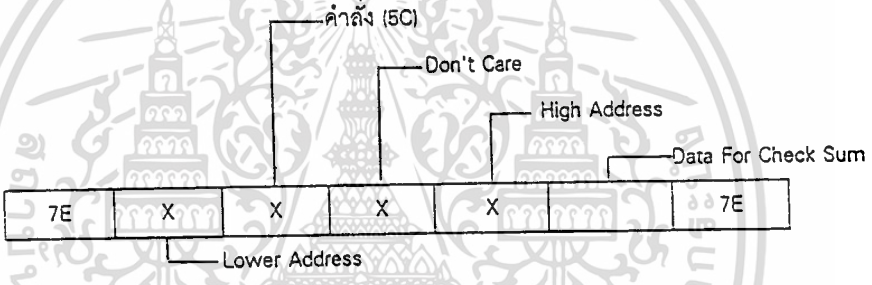


รูปที่ 3.12 แสดง Fram Format ของข้อมูล

- สับรูทีน INS5 : (Reset Electronic Devices)

```
INS5 :
        MOV DPTR,#4OOH          : ส่งข้อมูลออกเพื่อไปรีเซ็ทอุปกรณ์ที่
                                : ต่อกับพอร์ทนี้
        MOVX @DPTR, A
        SJMP EX5
        RET
```

สับรูทีนนี้เป็นเพียงแค่การรีเซ็ตวงจร Electronics เท่านั้นโดยสามารถที่จะรีเซ็ตครั้งละหลาย ๆ Station ได้ โดยรูปแบบของข้อมูลที่รับมาจาก PC เป็นดังนี้

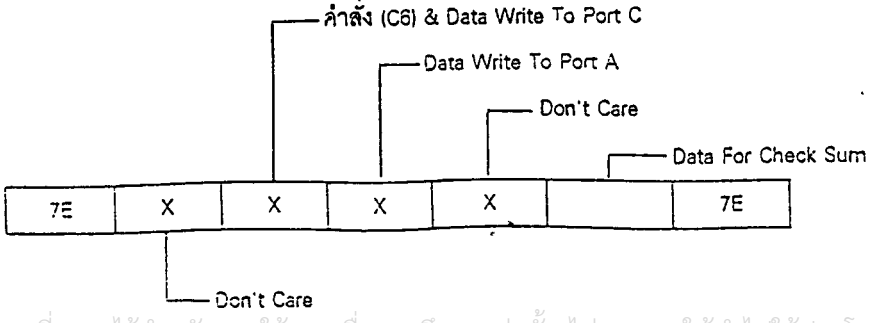


รูปที่ 3.13 แสดง Fram Format ของข้อมูล

- สับรูทีน INS6 : (Working All Station)

```
INS6:
        LCALL INS6_1            : ทำงานเช่นเดียวกับ INS1
        RET
```

สับรูทีนของคำสั่งสุดท้ายเป็นการสั่งให้ Microcontroller ทุกเครื่องทำการเขียน Port A และ Port C โดยรูปแบบของข้อมูลที่รับมาจาก PC เป็นดังนี้



รูปที่ 3.14 แสดง Fram Format ของข้อมูล

- สับรูทีน CHK_SUM : (CHECK SUM)

```
CHK_SUM
        MOV RO,#05H
        MOV R3,#OOH
        MOV DPTR,#2OOOH
        CLR C
CHK:
        MOVX A,@DPTR
        ADDC A,R3
        MOV R3,A
        INC DPTR
        DJNZ RO,CHK             : บวกข้อมูลไบต์ 1 - 5
        MOV DPTR,#2OO6H         : นำข้อมูลไบต์ที่ 7 มาบวกด้วย
        MOVX A,@DPTR
        ADDC A,R3               : เก็บข้อมูลที่บวกกันแล้วมาเก็บไว้ใน
                                : R3
        MOV B,#OFFH
        MOV R3,#OOH
        JNC NO_C
        MOV R3,#01H
O_C :
        DIV   AB                : หารข้อมูลด้วย OFFh
        MOV   A,B
        ADD   A,R3
        MOV   R3,A              : เศษเก็บไว้ใน R3
        MOV   DPTR,#2OO5h       : นำข้อมูลในไบต์ที่ 6 มาเปรียบเทียบ
        MOVX A,@DPTR
        SUBB  A,R3
        JNZ   ERR               : ถ้า R3 <> ข้อมูลไบท์ที่ 6 จะ Error
        MOV   R1,#OFFH
        SJMP  EX_CS
```

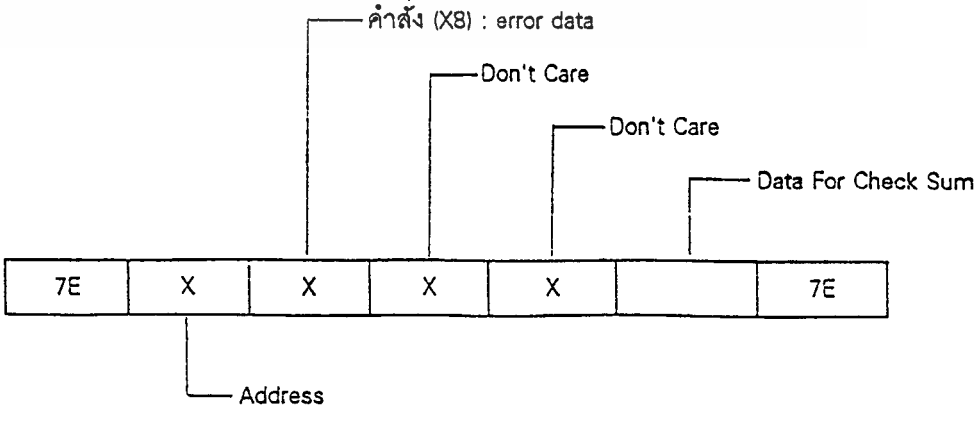
```
ERR :
        MOV   R1,#OOH                    : Check Sum Error
EX_CS :
        RET
```

สับรูทีนนี้เป็นการตรวจสอบข้อมูลที่ส่งมาว่าถูกต้องหรือไม่วิธีตรวจทำได้ โดยการนำข้อมูลที่ PC ส่งมาทุกตัว (ยกเว้นตัว FSC) นำมาบวกกันได้เท่าไร แล้วนำไปหารด้วย Offh เอาเฉพาะเศษมาเปรียบเทียบกับ FSC ว่าเท่ากันหรือไม่ ถ้าไม่เท่ากัน (R1<> offh) Microcontroller ต้องทำการส่งข้อมูลไปบอกให้ PC ทราบว่าข้อมูลที่ส่งมานั้นผิด โดยการเรียกสับรูทีน ERR_TRN

- สับรูทีน ERR_TRN : (TRANSMITION ERROR DATA)

```
ERR_TRN :
        MOV   DPTR,#2OOH
        MOV   DPTR,#QOOH              : C3,C2 - "10" - Error Code
        MOVX @DPTR,A
        LCALL TRN
        RET
```

- สับรูทีนนี้จะถูกเรียกใช้เมื่อมีการเช็คข้อมูลแล้วเกิดการผิดพลาดขึ้น สับรูทีนนี้จะส่งข้อมูลไปบอกให้ PC ทราบว่าเกิดข้อผิดพลาดขึ้น โดยรูปแบบของข้อมูลที่จะส่งไปให้ PC เป็นดังนี้



รูปที่ 3.15 แสดง Fram Format ของข้อมูล

- สับรูทีน TRN (TRANSMITION DATA)

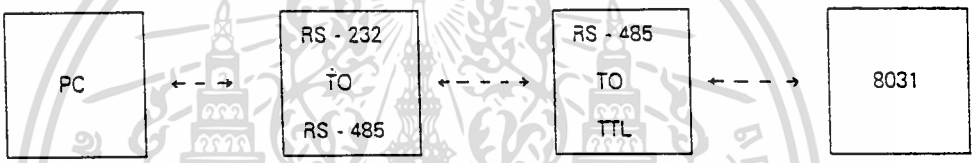
```
TRN :
        MOV   A,#O8H              : OOH - RECIVE, O8H - TRANSMIT (PC3)
        MOV   DPTR,#8OO2H
        MOVX @DPTR,A
        LCALL DEL_TX
        MOV   A,#7EH
        LCALL TRN2
        MOV   A,P1
        LCALL TRN2
        MOV   DPTR,#2OO2H
        MOVX A,@DPTR
        LCALL TRN2
        INC   DPTR
        MOVX A,@DPTR
        LCALL TRN2
        INC   DPTR
        MOVX A,@DPTR
        LCALL TRN2
        LCALL CHK_SUM             : นำข้อมูลที่ทำการ CHK_SUM ส่งออกไป
                                  : ด้วย
        MOV   A,R3
        LCALL TRN2
        MOV   A,#7HE
        LCALL TRN2
        RET
TR2 :
        JNB   TI,TRN2
        CLR   TI
        MOV   SBUF,A
        LCALL DEL_DX
```

RET

สับรูทีนนี้จะทำหน้าที่เพียงแค่นำข้อมูลในแอดเดรส 2000h - 2006h (ที่ได้จากสับรูทีนต่าง ๆ ก่อนแล้ว) ส่งออกไปให้ PC เท่านั้นเอง

3.4 การออกแบบและจัดสร้าง RS-485 Intertface CARD และโปรแกรมควบคุม

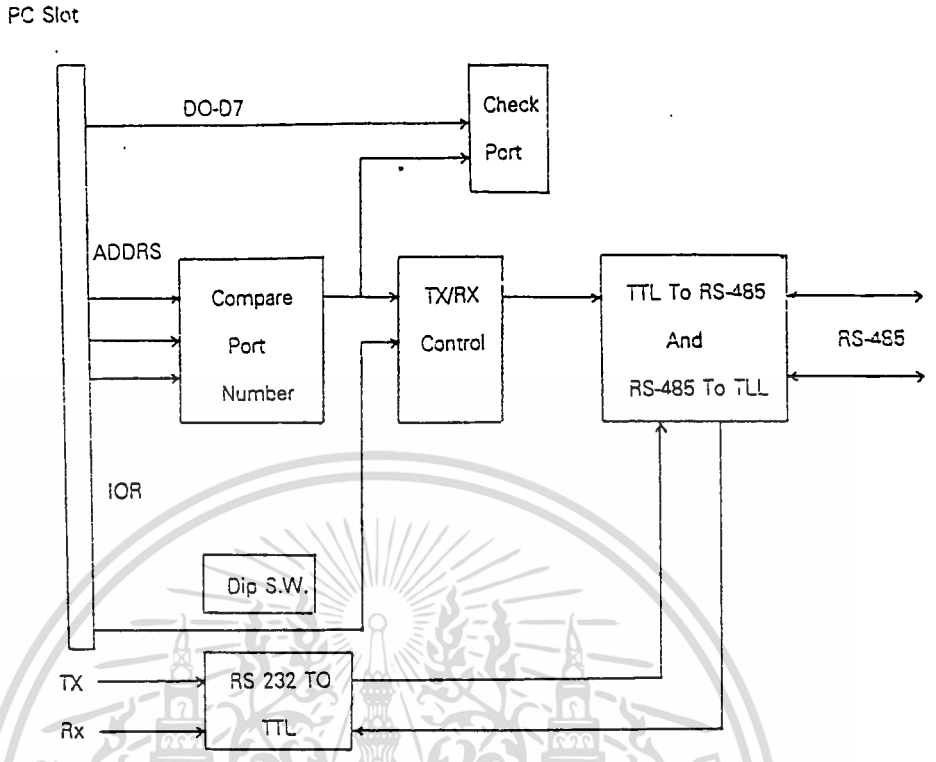
เนื่องจาก Computer มี Com Port ระบบ RS-232 อยู่ในการสื่อสารของระบบ RS-232 เป็นแบบ Point To Point และความยาวของ Cable มีขีดจำกัดที่ 50 ฟุต ซึ่งนับว่าเป็นระยะทางที่สั้นไม่เหมาะแก่การนำไปใช้งานในระบบควบคุมของโรงงานอุตสาหกรรม จึงต้องมีการเปลี่ยนเป็นระบบ RS-485 ซึ่งข้อดีของมันคือ สามารถใช้ความยาวของ Cable ได้ถึง 4,000 ฟุต และเป็นแบบ Multidrop ซึ่งง่ายและสะดวกในการใช้งาน



รูปที่ 3.16 แสดง Block Diagram ของ RS-232<-> RS-485 และ RS-485 <-> TTL

ในการแปลงระบบ RS-232 ไปเป็น RS-485 ไม่มีบริษัททางด้านการผลิต Semiconductor ที่ผลิต IC ที่ใช้โดยตรงแต่มี IC ที่สามารถแปลงจาก RS-232 ไปเป็น TTL Level ซึ่งคือ IC MC#1488 และ MC#1489 และมี IC ที่สามารถแปลงจาก TTL Lavel ไปเป็น RS-485 คือ IC #75176 ซึ่ง IC ทั้งสองเบอร์นี้มีวงจร Driver และ Reciver อยู่ภายในตัวเดียวกัน ซึ่ง Specification ของ IC ได้รวมอยู่ในภาคผนวก

จากรูป 3.16 เป็น Block Diagram การทำงานระบบ Interface บน Computer ซึ่งจะมีรายละเอียดของวงจรดังแสดงในรูปที่ 3.7 และรายละเอียดของการทำงานเป็นดังนี้คือ



รูปที่ 3.17 แสดง Block Diagram การทำงานของ RS-458 Interface Card บน PC

โดยกล่าวถึงส่วนของการ Decode Address ของ PC ก่อนซึ่งในส่วนที่นั้นจะต้องมีไว้ก็เพื่อจะเป็นการกำหนด Port ที่จะใช้ในการติดต่อระหว่าง PC กับชุด Microcontroller โดยจะใช้ A0 - A9, AEN (Address Enable) และใช้ขาสัญญาณ I/O Write) จาก SLOT ของ PC โดยนำสัญญาณขา A0,A1,A2,A3,A7,A8,A9 และ AEN มาเข้า IC #74LS688 เพื่อเปรียบเทียบกับ Dip Switch เพื่อจะเป็นการเลือก Port ใช้งานได้มากขึ้น ซึ่งในที่นี้ต้องการจะกำหนดให้ใช้ Port ตั้งแต่ 3OOH-3OFH โดยสามารถเลือก Port Number ต่าง ๆ ได้จากการเลือก Dip-Switch ดังแสดงให้เห็นในตารางข้างล่างนี้

ตารางที่ 3.4 แสดงค่า Number Port จากการเลือกค่า Dip-Switch

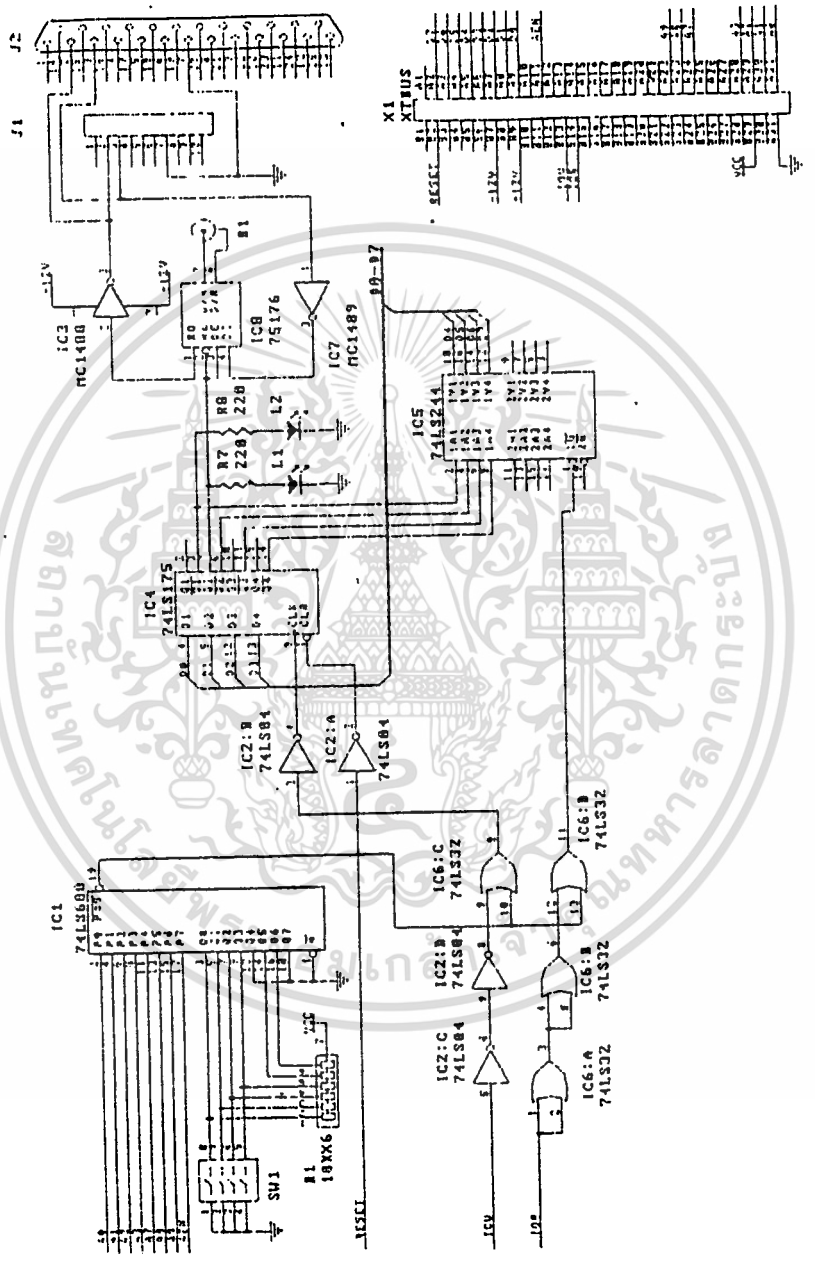
Bit1	Bit2	Bit3	Bit4	Port No.
0	0	0	0	300H
0	0	0	1	301H
0	0	1	0	302H
0	0	1	1	303H
0	1	0	0	304H
0	1	0	1	305H
0	1	1	0	306H
0	1	1	1	307H
1	0	0	0	308H
1	0	0	1	309H
1	0	1	0	30AH
1	0	1	1	30BH
1	1	0	0	30CH
1	1	0	1	30DH
1	1	1	0	30EH
1	1	1	1	30FH

ส่วนสัญญาณที่ได้จาก IC #74LS688 จะป้อนเป็นสัญญาณ Clock ให้กับ IC #74 LS175 และ Output ที่ออกจากขา 7 จะเป็นตัวเลือกให้เกิดการ Tramsmitting หรือ Receiving คือ ถ้าขา 7 ของ IC #74LS175 มีสภาวะเป็น "1" จะเป็นการ Enable ขา DE ของ IC #DS75176

สภาวะนี้จะเป็นการส่งข้อมูลจาก Computer ไปยังชุด Microcontroller โดยข้อมูลจาก RS-232 จะผ่าน IC#1489 เพื่อแปลงเป็น TTL Level เสียก่อน จากนั้นจึงส่งผ่าน Driver ของ IC#DS75176 เพื่อแปลงเป็น RS-485 ส่วนในการรับข้อมูลจาก Microcontroller มายัง PC จะเกิดขึ้นโดยในสภาวะที่ขา 7 ของ IC#74LS175 เป็น "0" ซึ่งก็จะเป็นการไป Enable ขา RE ของ IC # DS75176 ซึ่งเมื่อสัญญาณผ่าน Reciver มาแล้วก็จะถูกเปลี่ยนกลับไปเป็น TTL Level อีกครั้งต่อจากนั้นสัญญาณจะผ่าน IC #1488 เพื่อทำสัญญาณเปลี่ยนกลับไป

เป็น RS-232 Level ในสภาวะที่มีการรับหรือส่งข้อมูล LED จะเป็นตัวบอก Status ขณะนั้นด้วย

ในส่วนของการตรวจสอบเพื่อหา Port Number ที่จะใช้ในการควบคุมการติดต่อระหว่าง Computer กับ Microcontroller นั้นเราจะใช้ IC#74LS175 และ IC#74LS244 โดยการตรวจสอบนั้นเราจะเริ่มจากการป้อนค่าให้กับ D0-D3 แล้วสั่งให้ IC#74LS175 เก็บค่านั้นไว้จากนั้นจึงป้อนกับมาที่ขา D4-D7 โดยผ่านทาง IC #74LS244 จากนั้น Computer ก็จะทำการตรวจสอบค่าที่ป้อนไปที่ขา D0-D3 และค่าที่รับได้ที่ขา D4-D7 ว่ามีตรงกันข้ามหรือไม่ถ้าใช่ Computer ก็จะเก็บค่าของ Port Number ไปทีละหนึ่งจนกระทั่งถึง Port Number ตัวสุดท้ายหากยังไม่ใช่โปรแกรมก็จะทำการหยุดตรวจสอบและเลิกการทำงานโดยค่าของ Port ในการเริ่มตรวจสอบคือ 300H และค่าสุดท้ายคือ 30FH ซึ่ง Port Number ที่ได้นี้จะเอาไว้ทำการควบคุมการทำงาน IC #DS75176 อีกที่หนึ่งโดยผ่าน IC #74LS175



รูปที่ 3.18 แสดงวงจรของ RS-485 Interface Card บน PC

3.5 การออกแบบและจัดสร้างโปรแกรมที่ใช้บนเครื่อง PC เพื่อติดต่อกับ Microcontroller

การเขียนโปรแกรมในส่วนนี้ เราจำเป็นต้องแบ่งการเขียนโปรแกรมออกเป็นส่วน ๆ เพื่อให้ง่ายต่อการเขียนและแก้ไขข้อบกพร่องต่าง ๆ ที่จะเกิดขึ้นได้สะดวก โดยถ้าหากเขียนเป็นโปรแกรมเดียวจะทำให้เกิดความสับสนได้ ซึ่งในที่นี้จะแบ่งการเขียนโปรแกรมออกเป็นส่วน ๆ ได้ดังนี้

3.5.1 การเขียนโปรแกรมในส่วนของการควบคุมการติดต่อต่าง ๆ

การเขียนโปรแกรมในส่วนนี้ได้แก่ การเลือก Serial Port ที่จะใช้ในการติดต่อ ซึ่งในเครื่องคอมพิวเตอร์โดยทั่วไปจะมี Serial Port อยู่ 2 ช่อง คือ Com1 และ Com2 และการกำหนดความเร็วในการสื่อสาร (Baud Nate) จะถูกกำหนดอยู่ในแฟ้มข้อมูลชื่อ Config.bld ซึ่งอยู่ใน Urren Directory ขณะเริ่มการทำงานของระบบ หากไม่พบแฟ้มข้อมูลนี้ช่องการสื่อสารจะถูกกำหนดให้เป็น Com1 ด้วย Seed 9600B/S และสามารถแก้ไขเปลี่ยนแปลงได้ โดยผู้ใช้เลือกเมนู System Edit และโปรแกรมในส่วนของการ Check เบอร์ของ Port ที่จะใช้ในการควบคุมการติดต่อคือควบคุมการทำงานของ IC #DS75176 ซึ่งการทำงานของ Hardware ในส่วนนี้ได้กล่าวถึงแล้วในหัวข้อข้างต้น ในส่วนของโปรแกรมนั้นเราจะเริ่มจากการค่าตรวจสอบที่ 300H แล้วทำการตรวจสอบไปจนถึง 30H ซึ่ง Port Number และ Serial Port ที่ได้นั้นเราจะทำการเก็บค่าเอาไว้ เพื่อนำไปใช้ในโปรแกรมในส่วนต่อไปถ้าหากแผงวงจร RS-485 Inteface ไม่ได้ถูกติดตั้งในโปรแกรม ก็จะตรวจหาไม่พบและจะแจ้งให้ทราบ และหลุดออกไปจากโปรแกรมไปสู่ Dos Port ทันที

3.5.2 การเขียนโปรแกรมแสดงผลทางหน้าจอ

การเขียนโปรแกรมในส่วนนี้ได้แก่ การเขียนโปรแกรมแสดงผลทางด้านหน้าจอหรือ Menu ซึ่งจะใช้ในการควบคุมและตรวจสอบต่าง ๆ ซึ่ง MENU ที่เขียนนี้จะอยู่ในลักษณะของ Text Mode โดยมีภาพลักษณะของตึก ซึ่งสามารถที่จะกำหนดได้ว่าต้องการสูงขนาดไหน โดยสมมติว่าในแต่ละชั้นจะมีการติดตั้ง Microcontroller เพียงตัวเดียวและเรียงตามหมายเลขของชั้นไปเรื่อย ๆ ซึ่งกำหนดไว้สูงสุดถึง 100 หรือ 100 ชั้น

3.5.3 การเขียนโปรแกรมควบคุมการติดต่อสื่อสาร

สำหรับโปรแกรมในส่วนนี้ คือ ในส่วนที่ทำหน้าที่ติดต่อสื่อสารระหว่าง Computer กับ Microcontroller ซึ่งจะทำการติดต่อโดยใช้ Serial Port ที่เลือกไว้ แล้วผ่านการแปลงจาก RS-232 ไปเป็น RS-485 ไปเป็น RS-232 ซึ่งโปรแกรมในส่วนนี้จะมีความยากลำบากพอสมควร เพราะจะต้องเขียนโปรแกรมให้สามารถรับส่งกับ Microcontroller ให้ได้ โดยใช้หลักการติดต่อสื่อสารในรูปแบบของ HDLC ซึ่งได้อธิบายมาแล้วข้างต้น

เมื่อการเขียนโปรแกรมในส่วนต่าง ๆ ที่ได้กำหนดไว้เสร็จเรียบร้อยแล้วก็นำโปรแกรมทั้งหมดมารวมกัน เป็นโปรแกรมที่ใช้งานอย่างสมบูรณ์ โดยสามารถที่จะเขียนเป็น FLOW CHART การทำงานของโปรแกรมทั้งหมดได้ดังรูปที่ 3.22 ส่วนโปรแกรมที่ใช้งานนั้นได้พิมพ์และรวบรวมไว้ที่ภาคผนวกแล้ว และจะมีรายละเอียดสำคัญที่จะแสดงเป็นส่วนต่าง ๆ ดังต่อไปนี้

- เมื่อระบบเริ่มทำงาน โปรแกรมจะพยายามตรวจสอบการติดตั้ง RS-485 Interface Card โดยการสแกนหาจาก Port เบอร์ 300H ไปจนถึง 30FH โดยการเขียนข้อมูลลง Port เหล่านั้นและอ่านกลับมาทีละ Port หากพบว่า Port เบอร์ใดที่อ่านกลับมาแล้วข้อมูล Bit ที่ D4-D7 มีค่า Invert กับค่าของ DO-D3 ของข้อมูลที่เขียนเข้าไปแล้ว ก็จะเปลี่ยนค่าข้อมูลใหม่ แล้วเขียนและอ่านใหม่ซ้ำอีกครั้ง หากเป็นไปตามเงื่อนไขดังกล่าวก็จะแสดงว่ามีการติดตั้ง RS-485 Interface Card และจะเก็บหมายเลขของ Port นั้นไว้เพื่อใช้ควบคุมการรับหรือส่งข้อมูลต่อไป หากตรวจสอบจนถึง Port เบอร์ 30FH แล้วไม่พบ Port เบอร์ใดเป็นไปตามเงื่อนไขดังกล่าว โปรแกรมก็จะแสดงข้อความ "RS-485 Interface Not Install Program Abort"

- เมื่อพบว่ามีการติดตั้ง RS-485 Interface Card แล้วก็จะอ่านแฟ้มข้อมูลชื่อ Config. bld ซึ่งเป็นข้อมูลในรูป Textfile ซึ่งสามารถสร้างด้วย Editer ทั่วไปในแฟ้มข้อมูลนี้ จะมีข้อมูลบอกหมายเลขในช่องสื่อสาร Com1 หรือ Com2 ด้วย Speed เท่าใดและหมายเลขห้อง Address ของ Micro Controler แต่ละชุดพร้อมทั้งชื่ออุปกรณ์ที่ Micro controller คอยตรวจสอบอยู่และมี Micro controller เป็นลูกข่ายอยู่จำนวนเท่าใด ซึ่งแฟ้มข้อมูลสามารถแก้ไขและเปลี่ยนได้โดยผู้ใช้เลือกเมนู System Edit - Load และสามารถ Save ที่แก้ไขแล้วลง Diskette ได้

- เมื่อได้ข้อมูลจากแฟ้ม Config.bld แล้วก็จะทำการ Set ค่าเริ่มต้นของระบบเช่น ใช้ Com2, Speed 9600 มีจำนวน Micro controller อยู่ 105 ชุด แต่ละชุดมีอุปกรณ์อะไรที่ที่ต่อพ่วงเพื่อตรวจสอบบ้าง

- ตรวจสอบดูว่ามีใครกด Keyboard หรือไม่ ถ้าไม่ก็ทำการส่งข้อมูลไปให้ Micro controller อ่านสภาวของอุปกรณ์ที่มาต่อพ่วงส่งกลับมาให้ PC และแสดงสถานะของอุปกรณ์ต่าง ๆ บนจอภาพ โดยจะเริ่มที่หมายเลขเครื่องที่มีแอดเดรสประจำเครื่องต่ำสุดก่อน และเมื่อได้รับข้อมูลตอบกลับมาแสดงผล แล้วก็แสดงคำสั่งให้หมายเครื่องถัดไปส่งข้อมูลกลับมาแสดงผลเป็นเช่นนี้ต่อไป

- เมื่อส่งไปครบ 4 เครื่องแล้วจะส่งคำสั่งไปให้ลูกข่ายทุกตัวที่มี Alarm ทั้ง 4 แบบ คือ ไฟ, โจรกรรม, การขอความช่วยเหลือ และ MANI Breaker off ถ้า Micro controller

ตัวใดตรวจพบ Alarm อย่างใดอย่างหนึ่งใน 4 แบบ ก็จะส่งข้อมูลกลับมาแสดงบนจอภาพ ว่าเกิดที่แอดเดรสใด Alarm ชนิดใด การส่งคำสั่งถาม Alarm ลักษณะนี้จะเกิดทุกๆ การติดต่อครบ 4 ครั้ง หรือครบจำนวนที่หารด้วย 4 ลงตัว เช่น 0 4 8 16 32 เป็นต้น เมื่อครบจำนวนแล้วจะกลับไปเริ่มตัวแรกใหม่

- เมื่อมีกด Keyboard ก็จะเข้าสู่ MENU ซึ่งใช้ในการควบคุมการติดต่อการ Control ให้ Microcontroller ทำงานตามที่ผู้ที่ใช้ต้องการ ซึ่งจะกล่าวไว้ในหัวข้อการใช้งาน Program รายละเอียด

3.5.4 การเขียนโปรแกรมในส่วนการสื่อสารข้อมูล

ในเครื่องหมาย IBMPC จะมีการติดตั้ง Communication Port RS-232C ไว้เป็นมาตรฐาน 1 Port และเว้นที่ว่างสำรองไว้อีก 1 Port โดยมีชื่อว่า Com1 และ Com2 ตามลำดับการที่จะติดต่อกับ Com1 และ Com2 ทำได้โดยอ้างถึง Port 3F8-3FE และ 2F8-2FE ตามลำดับโดยที่ Port แต่ละ Port มีความหมายแสดงไว้ตามตารางที่

3.5.5 การโปรแกรมใช้งานโดยมีการกำหนดค่าต่าง ๆ ลงในรีจิสเตอร์ของ IC เบอร์ 8250 โดยการโปรแกรมผ่าน Port ต่าง ๆ ดังตารางที่ 3.5

ตารางที่ 3.5 หมายเลขอินพุทเอาต์พุตพอร์ทของ Com1 Com2

อินพุตเอาต์พุตพอร์ต		เลือกรีจิสเตอร์	สถานะ DLAB
พอร์ต Com_1	พอร์ต Com_2		
3F8	2F8	บัฟเฟอร์ IX	DLAB 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB 0 (อ่าน)
3F8	2F8	แลตช์ตัวหาร (LSB)	DLAB 1
3F9	2F9	แลตช์ตัวหาร (MSB)	DLAT 1
3F9	2F9	อีนาเบิลอินเตอร์รัพต์	
3FA	2FA	กำหนดอินเตอร์รัพต์	
3FB	2FB	ควบคุมการสื่อสาร	
3FC	2FC	ควบคุมโมเดม	
3FD	2FD	แสดงสถานะสายสื่อสาร	
3FE	2FE	แสดงสถานะโมเดม	

1) การเขียนโปรแกรมควบคุม Computer ด้วยภาษา C

ก) การโปรแกรมบ็อดเรต ใช้คำสั่งภาษา C ตามลำดับดังนี้

outportb (Ox3fb,Ox3f8);

outportb(Ox3f8,LSB);

outportb(Ox3f9,MSB);

ค่าของ MSB และ LSB ดูได้จากตาราง 9.5

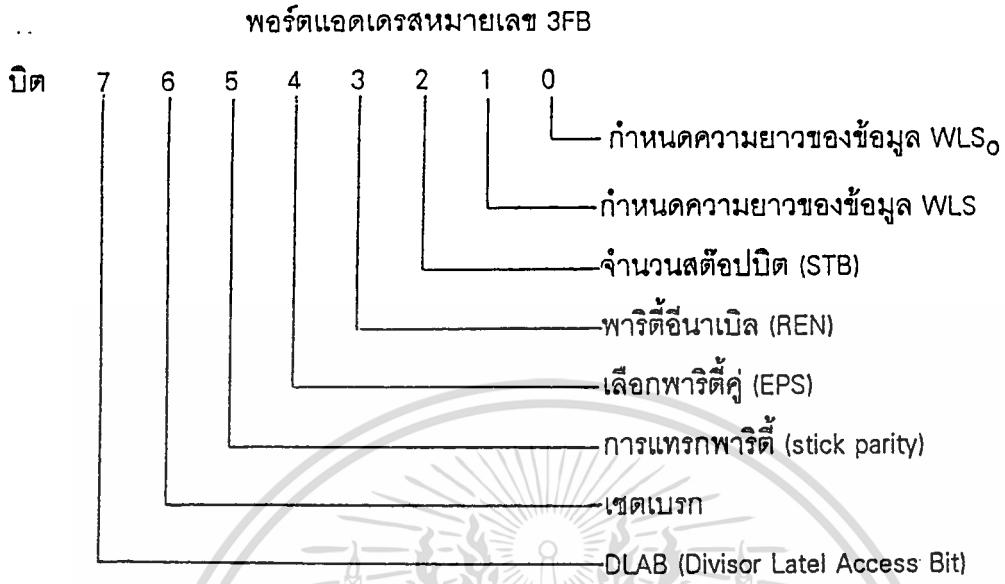
ตารางที่ 3.6 แสดงค่าตัวหารสำหรับการกำหนดอัตราบ๊อด

อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

ข) การโปรแกรมรูปแบบของการสื่อสารใช้คำสั่งในภาษา C ดังนี้

outportb (Ox2fb,INFO)

โดยมีค่าของ INFO มีหลายความหมายดังนี้

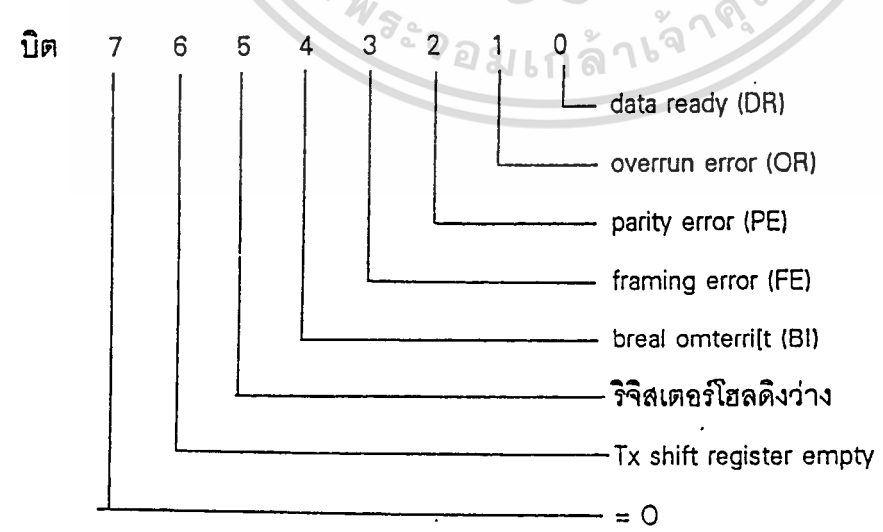


รูปที่ 3.19 ค่าของบิทในรีจิสเตอร์ควบคุมสายการสื่อสาร

ค) การส่งข้อมูลกระทำโดยตรวจสอบสถานะของ Comport ก่อนเมื่อพบว่า Buffer ว่างจึงเขียนข้อมูลลงบน Port 3F8h ซึ่งเขียนภาษา C เป็นดังนี้

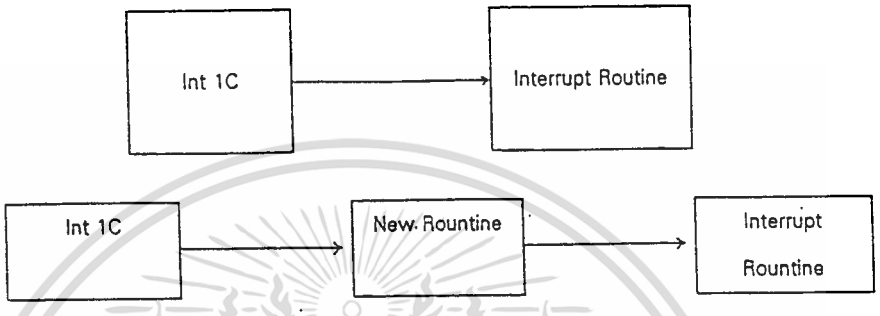
```
While (!k&Ox20) k = inportb(Ox2F)
outportb(Ox2F8, Data)
```

สถานะของ Comport อ่านจาก Port หมายเลข 2FDH ซึ่งบิทต่างมีความหมายดังนี้



รูปที่ 3.20 ค่าของบิทในรีจิสเตอร์แสดงสถานะสื่อสาร

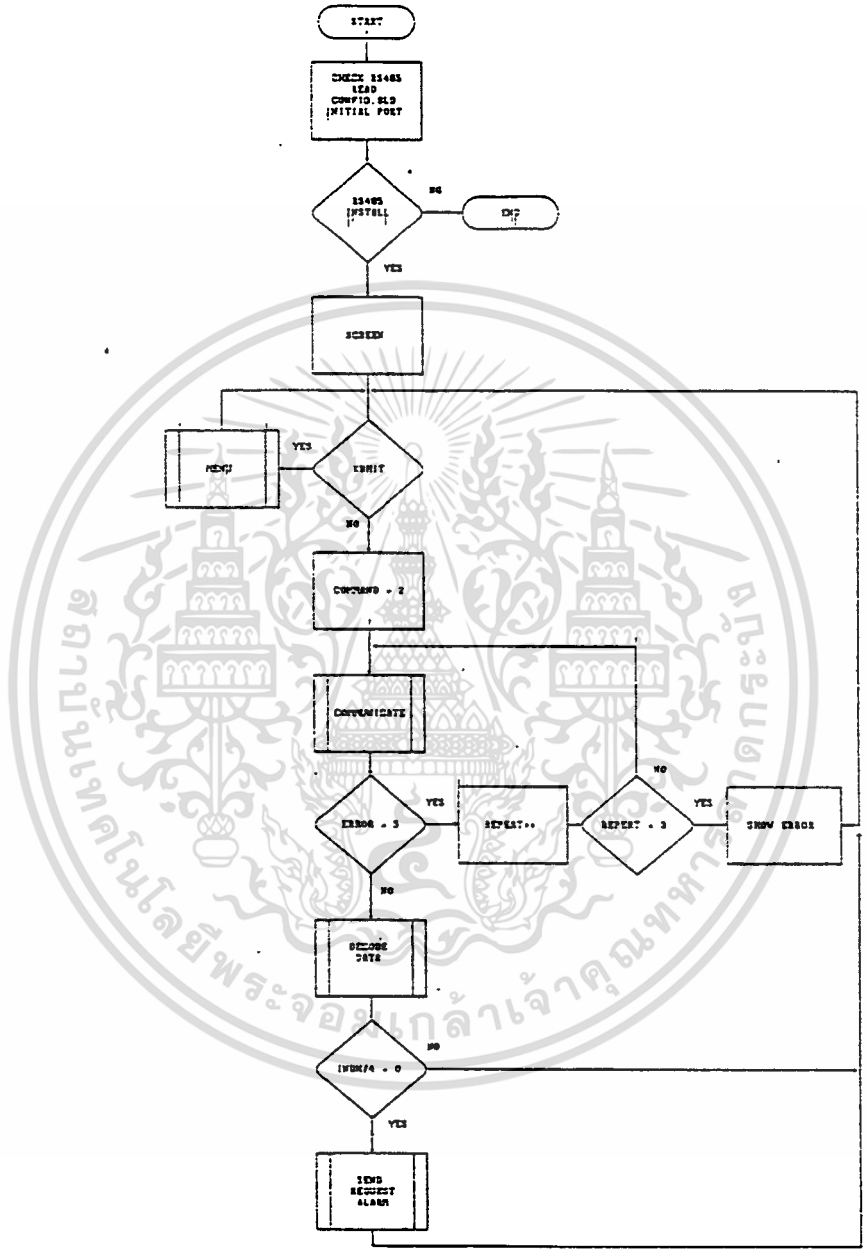
ในเครื่อง IBM PC มีระบบเวลามาตรฐานอยู่ กล่าวคือ INT 1CH จะเกิดขึ้นวินาทีละ 18.2 ครั้ง การนำเอาลักษณะนี้มาใช้งานจับเวลาโดยเขียนโปรแกรมขึ้นมารองรับการ Interrupt ดังกล่าวเสร็จแล้วส่งคำสั่งให้ทำงานตามฟังค์ชั่นเดิม ซึ่งมีแนวความคิดเป็นดังรูปข้างล่าง



รูที่ 3.21 แสดงการเขียน interrupt routine

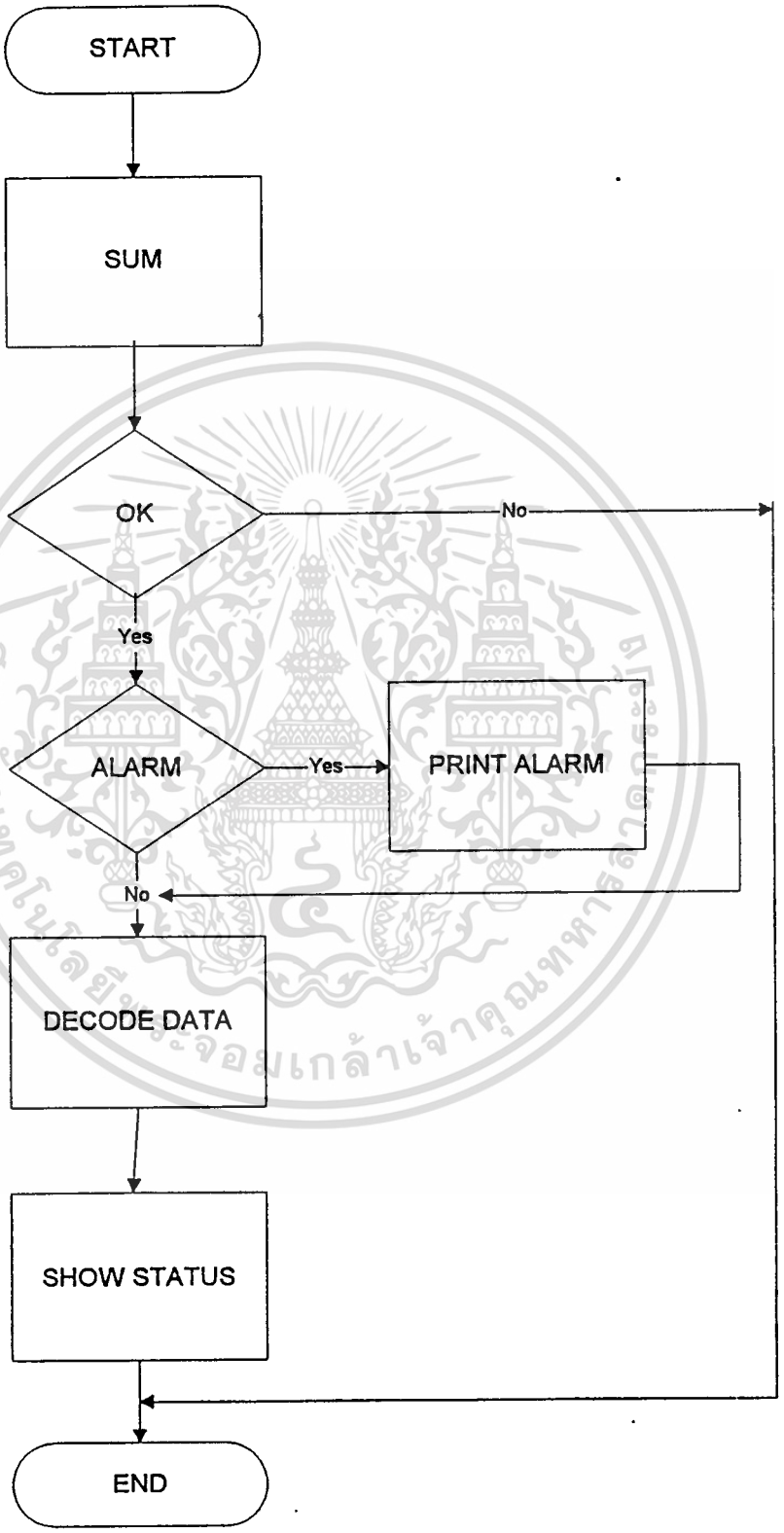
การเขียนโปรแกรมใช้คำสั่งภาษา C ดังนี้

```
#include (dos.h)
#define INTR Ox1C
void interrupt (*oldhandler)(void);
void interrupt handler(void)
{       Count ++;
        oldhandler( );
}
main ( )
{       oldhandler = getvect(TNTR);
setvect(INTR,handler);
}
```



รูปที่ 3.22 FLOW CHART การทำงานของโปรแกรมบนเครื่อง PC

3.6 การทำงานของฟังก์ชั่นในโปรแกรมภาษา C



รูปที่ 3.23 แผนผังการทำงานในส่วนของการนำข้อมูลมาดีเทคเพื่อส่งออกหน้าจอ

ในส่วนของการนำข้อมูลมาดีเทคมีฟังก์ชั่นต่าง ๆ ดังนี้

3.6.1 ฟังก์ชั่น Conv DatStr

เป็นฟังก์ชั่นทำหน้าที่เปลี่ยนข้อมูลที่ได้มาจากการอ่าน Port กลับมาจากซึ่งเป็นข้อมูลระดับบิท ("0" หรือ "1")ให้เป็น String (ข้อความ) เพื่อประโยชน์ในการแสดงทางหน้าจอ ซึ่งเป็นลักษณะการทำงานต้องมีการดีเทค ข้อมูลในยระดับบิท โดยใช้การ Shift ที่ละบิท พร้อมกับทำการตรวจสอบข้อมูลบางบิทที่สำคัญ เช่น บิทที่แสดงว่าเกิดไฟไหม้ ขโมย เป็นต้น เพื่อไปแจ้งกับหน้าจอให้ผู้ควบคุมทราบในทันที

3.6.2 ฟังก์ชั่น Checksum

เป็นฟังก์ชั่นที่ทำหน้าที่ตรวจเช็ค Summing ของข้อมูลที่ส่งมาใน byte ที่ 6 ว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องก็แสดงว่าข้อมูลที่ส่งมามี error แต่ถ้าถูกต้องก็ทำหน้าที่อื่นต่อไป วิธีการตรวจสอบคือ นำข้อมูลทุก ๆ byte ที่เป็น sum มาบวกกัน จากนั้นนำผลลัพธ์มาหารด้วย FF (เลขฐาน 16)แล้วนำเศษที่ได้จากการหารมาตรวจสอบว่าตรงกับที่ส่งมาหรือไม่

3.6.3 ฟังก์ชั่น seach Addr

เป็นฟังก์ชั่นที่ทำหน้าที่ นำ address ที่ส่งมากับข้อมูลมาตรวจสอบกับ address ในตารางให้ตรงกัน เพื่อนำเอาข้อมูลต่าง ๆ ในตารางของ address นั้นมาเก็บใน buffer เพื่อให้จอภาพใช้ต่อไป เช่น address ที่ส่งมาเป็น 1A ก็จะไปหาในตารางว่าที่ address 1A มีข้อมูลอะไรบ้าง เป็นห้องหมายเลขอะไร มีอุปกรณ์อะไรติดตั้งอยู่บ้าง เป็นต้น

3.6.4 ฟังก์ชั่น utility ต่าง ๆ

ฟังก์ชั่น utility นี้จะเป็นฟังก์ชั่นที่ใช้ในงานทั่ว ๆ ไป เช่น การ sort ตาราง, การสร้าง summing,savefile,openfile,delete,read Table, และ edit ซึ่งฟังก์ชั่นเหล่านี้อาจเป็นฟังก์ชั่นย่อยของฟังก์ชั่นอื่น ๆ อีกก็ได้ และสามารถเรียกใช้งานได้สะดวก ฟังก์ชั่นต่าง ๆ มีดังต่อไปนี้

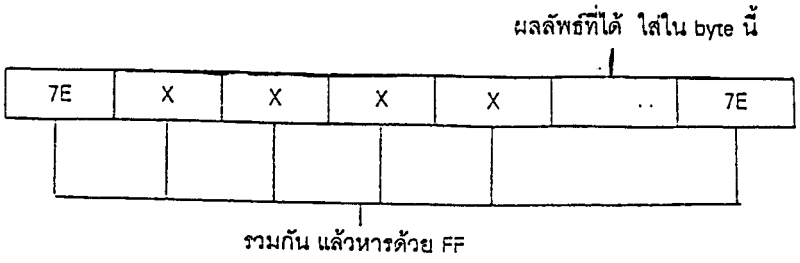
3.6.5 ฟังก์ชั่น sort Table

เป็นฟังก์ชั่นที่ทำหน้าที่เรียงลำดับของหมายเลขห้องในตารางให้เรียกจากน้อยไปยังมาก เพื่อสะดวกในการตรวจสอบ (ข้อมูลอื่น ๆ เช่น รายการอุปกรณ์ต่าง ๆ และ address ก็มีการสับเปลี่ยนตามหมายเลขห้องด้วย)

3.6.6 ฟังก์ชั่น create Sum

เป็นฟังก์ชั่นที่ทำหน้าที่ สร้าง Sum เพื่อใส่ในเฟรม ฟอร์แมทของข้อมูลที่จะส่ง โดยมีหลักการสร้างคือ นำเอาข้อมูลทุก ๆ byte มารวมกัน แล้วหารด้วย FF (เลขฐาน 16) นำเอาเศษที่ได้มาใส่ลงใน byte ของ summing ผลลัพธ์ที่ได้ ใส่ใน byte นี้

ผลลัพธ์ที่ได้ ใส่ใน byte นี้



3.6.7 ฟังก์ชั่น save File

เป็นฟังก์ชั่นที่ทำหน้าที่ นำเอาข้อมูลต่าง ๆ ที่อยู่ในตาราง(ปัจจุบัน) เก็บลงใน file ที่ต้องการเก็บเพื่อประโยชน์ในการ update -ข้อมูลได้ตลอดเวลาเมื่อมีการแก้ไขปรับปรุงเพิ่มเติมลักษณะของตารางมีลักษณะดังตารางที่ 3.7

ตารางที่ 3.7 แสดงข้อมูลที่สามารถทำการแก้ไขได้ -

comport..... broudrate.....		
room	address	component.....
.	.	.
.	.	.
.	.	.

3.6.8 ฟังก์ชั่น open File

เป็นฟังก์ชั่นที่ทำหน้าที่ในการ Initial การทำงานของเครื่องเพื่อเก็บข้อมูลที่ user คีย์ลง file ไว้เพื่อใช้เป็นข้อมูลในการอ้างอิง ในฟังก์ชั่นอื่นๆ เมื่อใช้เสร็จก็จะมีการบิด file ด้วย.

3.6.9 ฟังก์ชั่น delete

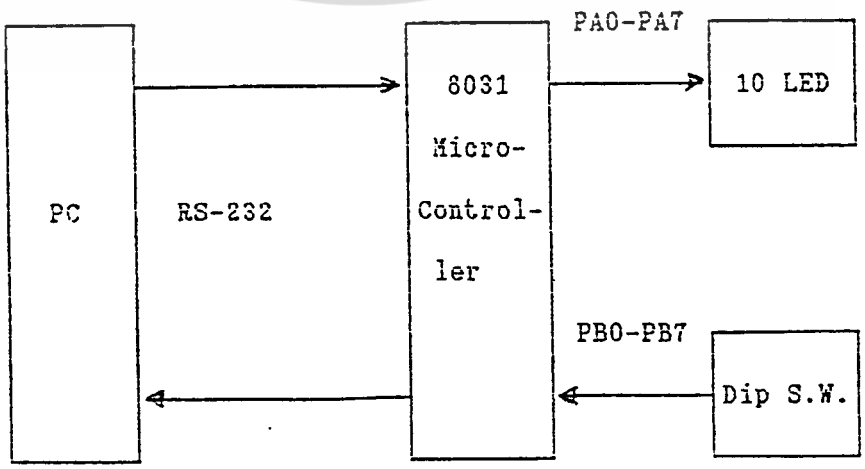
เป็นฟังก์ชั่นที่ทำหน้าที่ลบข้อมูลในตารางทิ้ง เช่น แต่เดิมห้องหมายเลข 1000 address ที่ติดตั้งคือ 1A มีอุปกรณ์ที่ต่อไว้คือ TV และ AIR และห้องหมายเลข 2000 address คือ 2A มีอุปกรณ์ คือ VDO TV LAMP ต่อมาต้องการลบหมายเลขห้อง 1000 ทิ้ง คือปัจจุบันได้ยกเลิกห้องนี้ไปแล้ว ก็สามารถแก้ไขในตารางใหม่โดยใช้ฟังก์ชั่นนี้ ฟังก์ชั่นนี้จะทำการหาห้องที่ระบุ (ในตารางจะมีการ sort ไว้ก่อนแล้ว) จากนั้นจะลบทิ้งพร้อมกับเลื่อน pointer ที่ชี้ไว้ และลบจำนวนห้องที่ไม่ต้องการเก็บแล้วออกจากจำนวนห้องที่ติดตั้งระบบไว้ทั้งหมด ฟังก์ชั่นนี้สามารถระบุห้องเป็นกลุ่มได้ เช่น ตั้งแต่ห้องหมายเลข 1000-5000 ต้องการลบทิ้งก็สามารถใช้ฟังก์ชั่นนี้ได้ด้วย

3.6.10 ฟังก์ชั่น read Table

เป็นฟังก์ชั่นที่ทำหน้าที่อ่านข้อมูลจากตารางมาเก็บไว้ที่ Buffer ดังหนึ่งโดยเมื่อ user ต้องการที่จะอ่านข้อมูลจากตารางที่ประกอบด้วย ห้อง Address อุปกรณ์ต่าง ๆ เพื่อนำไปแสดงหน้าจอ โดย user ต้องเป็นผู้กำหนดหมายเลขห้องมาก่อน จากนั้นจึงต้องมีการค้นหาหมายเลขห้องในตาราง ถ้าค้นหาห้องนี้ไม่เจอจะจะเก็บ NULL ไว้ใน Buffer แทน ฟังก์ชั่นที่เกี่ยวกับการ Out Port

3.7 <u>การออกแบบสร้างวงจรทดสอบ (simulate) เพื่อใช้เป็นชุดทดลอง</u>

การทดสอบเครื่อง Microcontroller สามารถทำได้โดยอิสระ ซึ่งการทดสอบตัว Microcontroller นี้ จะทำการทดสอบ Port I/O ของ 8255 โดยการเขียนโปรแกรมผ่าน Eprom Emulator ทำการ Set ให้ Port CO,C1,C2,C3 เป็น O/P และ C4,C5,C6,C7 เป็น I/P และทำการ Set Dip S.W. ตัว Micro ให้เป็นตำแหน่งหนึ่งก่อน ส่วนการติดต่อกับเครื่อง Computer นั้นในช่วงแรกนี้จะใช้ RS-232 เป็นตัวติดต่อกับ PC ก่อนซึ่งการทดสอบการติดต่อกับ PC นั้นทำได้โดยการใช้ S.W. แทน Sensor Unit โดยการกำหนดสภาวะ Logic "0" เป็นสภาวะปกติ และถ้า S.W. เป็น Logic "1" ก็จะเป็นสภาวะที่ Error ส่วนการแสดงผลทางเอาท์พุทใช้ LED เป็นตัวบอกสภาวะของการ ON หรือ OFF ของอุปกรณ์ไฟฟ้า ซึ่งในขณะทำงานก็จะแสดงบนจอ (monitor) ของ Cumputer ว่ามีอะไรเกิดขึ้นในช่วงเวลานั้น ที่ Microcontroller ตัว ใดมีสภาวะเป็นอย่างไร แล้วทดลองส่งข้อมูลจาก PC สั่งให้ microcontroller ปิด/เปิด LED โดยจะป้อนค่าที่จะทำให้ LED ติดหรือดับผ่าน Keyboard การติด/ดับของ LED แสดงว่าเครื่องทำงานในที่นี้จะใช้ LED 10 ดวง (แทนการควบคุมอุปกรณ์ 10 ชุด) ซึ่งกำลังสามารถทำให้ LED ทั้ง 10 ดวงดับและติดได้ตามต้องการ แสดงว่าตัว Microcontroller และโปรแกรมสามารถทำงานได้ตามต้องการแล้วจึงทำการทดสอบส่วนประกอบอื่น ๆ ต่อไป



รูปที่ 3.24 แสดงการทดสอบการทำงานของ Microcontroller และโปรแกรม

บทที่ 4
ผลการทดสอบหรือทดสอบโครงงาน

4.1 คำนำ

ในการจัดสร้างโครงงานนี้ประกอบด้วยส่วนย่อย ๆ จำนวนมาก ดังนั้นในการจัดสร้างจึงต้องจัดสร้างในส่วนที่ เมื่อสร้างเสร็จแล้วสามารถนำไปทำการทดสอบการทำงานได้เลย ไม่ต้องรอส่วนประกอบอื่น ๆ ซึ่งก็ได้แก่ตัว Microcontroller ซึ่งเมื่อทำการจัดสร้างเสร็จแล้วก็สามารถนำมาทดสอบตามโปรแกรมได้เลย และยังสามารถนำมาใช้พัฒนาโปรแกรมการติดต่อกับเครื่อง PC ได้อีก ในขณะที่ทำการจัดสร้างตัว Microcontroller นั้นก็สามารถสร้าง RS-485 Interface Card ซึ่งก็คือ การจัดสร้างระบบการติดต่อ RS-485 ได้เลย เพราะในส่วนของการทำงานของทั้ง 2 ส่วนนั้นจะเกี่ยวข้องกันในส่วนของการติดต่อสื่อสารภายในระบบ RS-485 เท่านั้น ส่วนในการเขียนโปรแกรมควบคุม PC และ RS-485 นั้นสามารถเขียนได้โดยใช้เครื่องคอมพิวเตอร์ทั่วไปที่มีอยู่เขียนโดยในขั้นต้นจะใช้ระบบการติดต่อแบบ RS-232 เป็นตัวติดต่อสื่อสารระหว่าง Computer กับ Microcontroller

จากลำดับการจัดสร้างนี้ทำให้ง่ายต่อการตรวจสอบและทดสอบส่วนประกอบต่าง ๆ เมื่อเกิดการขัดข้องขึ้นมา ก็ง่ายต่อการตรวจสอบแก้ไขปัญหาที่เกิดขึ้นได้

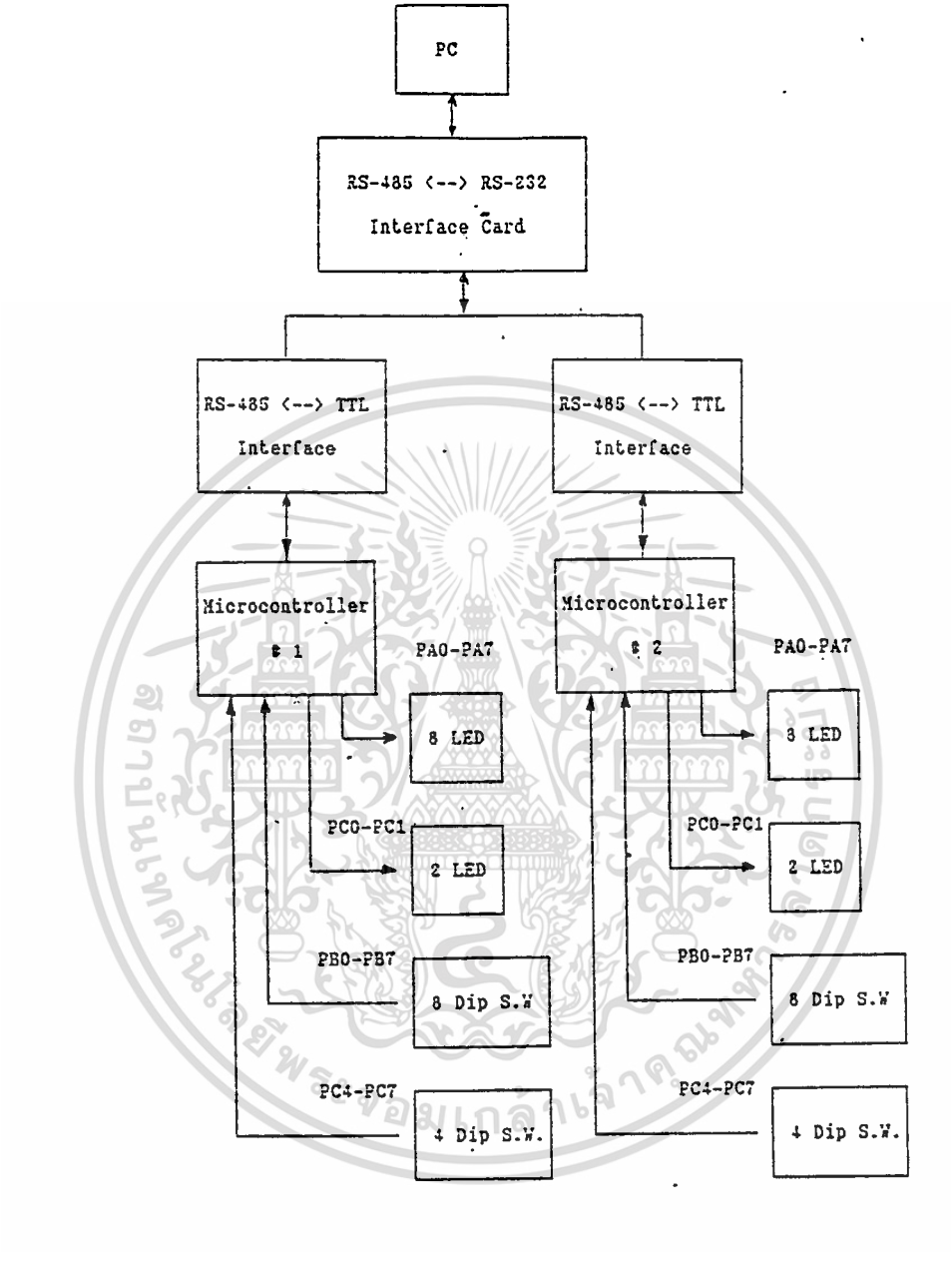
4.2 การทดสอบโครงงาน

4.2.1 อุปกรณ์ที่ใช้

1) Personal Computer 1 เครื่อง
2) Microcontroller 2 เครื่อง
3) Interface card 1 เครื่อง
4) LED 20 ดวง
5) Dip switch 20 ตัว
6) สายสัญญาณระหว่าง PC กับ Microcontroller

4.2.2 วิธีการทดลอง

1) ประกอบชุด Microcontroller และ ชุดของ PC ดังรูปที่ 4.1 ปรับ Dip S.W. ของ Microcontroller #1 ไว้ที่ตำแหน่ง 0Ah กำหนดให้ห้องเป็น 3000 และปรับ Dip S.W. ของ Microcontroller #2 ไว้ที่ตำแหน่ง Odh กำหนดเป็นห้องที่ 1200



รูปที่ 4.1 แสดงชุดสำหรับการทดลองโครงงาน

2) ทดลองการสื่อสารระหว่าง Microcontroller กับ PC โดยการต่อและไม่ต่อ Microcontroller กับ PC สังเกตุจอภาพ

3) ทดลองการอ่านสถานะของเครื่องใช้ไฟฟ้าโดยการปรับ Dip S.W. ชุด 8 ตัว (พอร์ท B) ใช้ Dip S.W. ของ Microcontroller #1 (ห้อง 3000)

ก) ปรับให้มีค่าเท่ากับ OAAh (pb7 - pb0 - '10101010','1'-ON,'0'-OFF)

ข) ปรับให้มีค่า OFOh (PB7 - PB0 -'11110000') สังเกตุสถานะของเครื่องใช้ไฟฟ้าจากจอภาพที่แสดงขึ้นมา

4) ทดลองการควบคุมอุปกรณ์ไฟฟ้า โดยการใช้เมนู Control แล้วเลือกเมนูย่อย Control Room ทดลองควบคุมให้ LED ติดและดับ โดยให้ควบคุมห้อง 3000

ก) ให้ ON เครื่องใช้ไฟฟ้าทุกเครื่อง

ข) ให้ OFF ใช้เครื่อง 5 เครื่องแรก (PC 1, PC1, PA1-PA2) สังเกตุ LED ทั้ง 10 ดวง

5) ทดลองการตรวจจับเหตุการณ์ฉุกเฉินโดยการปรับ Dip S.W.ชุด 4 (Port Ch ห้อง 3000) ทีละตัวสังเกตุจอภาพที่แสดงขึ้นมา

4.3 ผลการทดสอบโครงงาน

4.3.1 ผลการทดสอบการสื่อสารระหว่าง Microcontroller กับ PC

ก) กรณีที่ไม่ต่อ Microcontroller กับชุด PC จอภาพจะแสดง Error "Communication Fail! (ROOM)"

ข) กรณีที่ไม่ต่อ Microcontroller กับชุด PC จอภาพจะแสดงตามปกติ (ไม่แสดง Error ใด ๆ) จะแสดงตามปกติ

4.3.2 ผลการทดสอบการอ่านสถานะของเครื่องใช้ไฟฟ้าจากการปรับ Dip S.W. Port B

ก) ปรับให้มีค่าเท่ากับ OAAh

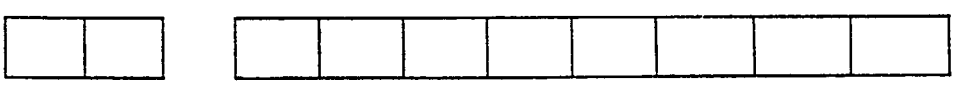
ข) ปรับให้มีค่าเท่ากับ Ofh

4.3.3 ผลการทดสอบการควบคุมเครื่องใช้ไฟฟ้า

ก) Control ให้ ON ทุกเครื่องจะได้ผลการทดลองดังนี้

LED จะสว่างดังนี้

PCO PC1 PA7.....PAO



"0"- OFF, "1" - ON

ข) Control ให้ OFF 5 เครื่องแรก จะได้ผลการทดลองดังนี้

LED จะสว่างดังนี้

PCO PC1 PA7.....PAO

"0" - OFF, "1" - ON

4.3.4 ผลการทดสอบเหตุการณ์ฉุกเฉิน

ก) โดยการปรับ Dip S.W. PC7 (FIRE) ให้ ON

ข) โดยการปรับ Dip S.W. PC6 (ROBBER) ให้ ON

ค) โดยการปรับ Dip S.W. PC5 (HELP ME) ให้ ON

ง) โดยการปรับ Dip S.W. PC4 (BREAKER) ให้ ON

4.4 การใช้งานโปรแกรมตรวจสอบและควบคุมระบบ

เริ่มต้นใช้งานโดยเรียกชื่อ File Master จากนั้นโปรแกรมจะทำงานโดยการ Sequence ไปตามห้องต่าง ๆ ที่ติดตั้งไว้ในระบบ ซึ่งการแสดงต่าง ๆ ที่ติดตั้งไว้ในระบบ ซึ่งการแสดงต่าง ๆ เกี่ยวกับหน้าจอได้แสดงไว้คร่าว ๆ แล้วในเรื่องของการทดสอบโครงงาน

การเลือกเมนูต่าง ๆ นั้นจะเลือกได้โดยการกด "F10" โปรแกรมจะเข้าสู่เมนูต่าง ๆ หรืออาจจะได้จากการกด Hot Key ของเมนูนั้น ๆ แล้วสามารถใช้ Key ลูกศรเป็นตัวเลือกค่าที่ต้องการอีกครั้งหนึ่ง หรือบางเมนูก็อาจจะต้องป้อนค่าต่าง ๆ โดยการพิมพ์ค่าเข้าไป

เมนูหลักของโปรแกรมที่แสดงหน้าจอจะแบ่งเป็นเมนูหลักต่าง ๆ ได้ดังนี้

4.4.1 เมนู Syster แบ่งออกเป็นเมนูย่อย 7 เมนู

1) Seriel port เป็นเมนูที่ใช้ในการเปลี่ยน Comport ของ PC ในกรณีที่มีการเปลี่ยนแปลง Comport ใหม่ เมื่อเลือกเมนูนี้แล้วจะมีเมนูย่อยให้เลือก Comport ว่าจะเลือก Comport 1 หรื 2 มีการเลือกโดยคีย์ลูกศร

2) Baudrate เป็นเมนูใช้สำหรับเลือก Baudrate ในการรับส่งข้อมูลระหว่าง Microcontroller กับ PC หากเลือกเมนูนี้จะให้ User เลือกค่า Baudrate ใหม่ซึ่งจะมี 4 ค่าให้เลือกด้วยกัน คือ 9600, 4800, 2400, 1200

3) Delete Room เป็นเมนูที่ใช้เมื่อต้องการยกเลิกห้องที่กำหนดซึ่งเมนูการเลิกห้องนี้จะเป็นการยกเลิกห้องเดียวหรือยกเลิกเป็นกลุ่ม

4) Edit Room เป็นเมนูที่มีลักษณะคล้ายกับเมนู Delete Room คือเมื่อทำการเพิ่มห้องขึ้นมา 1 ห้อง (เมนูนี้เพิ่มได้ห้องเดียวเท่านั้น) เมื่อเลือกเมนูนี้โปรแกรมจะ

กำหนดให้ user ใส่เลขหมายห้องที่เพิ่มเข้ามา และชื่อของอุปกรณ์ที่ใช้งานอยู่ภายในห้องนั้น

5) Load เป็นเมนูที่ทำหน้าที่โหลดไฟล์ที่สร้างไว้ก่อนขึ้นมาใช้งาน ซึ่งไฟล์ที่โหลดมานี้จะเก็บข้อมูลเกี่ยวกับระบบของโปรแกรม เช่น Comport, Baud rate, หมายเลขห้อง Address รายการอุปกรณ์ต่าง ๆ ที่อยู่ภายในห้องนั้น

6) Update เป็นเมนูที่ใช้เก็บข้อมูลที่ทำการแก้ไขแล้วลงไฟล์ (Save)

7) Clr error เป็นเมนูที่ใช้ลบ Error ที่แสดงบนหน้าจอออก

4.4.2 เมนู Control แบ่งออกเป็นเมนูย่อยได้ 3 เมนู

1) Re Start เป็นเมนูที่ใช้เมื่อต้องการให้เริ่มต้นการ Sequence ที่ห้องที่กำหนดใหม่เมื่อเลือกเมนูนี้โปรแกรมจะให้ User ใส่หมายเลขห้องที่ต้องการ

2) Control Room เป็นเมนูที่ใช้สำหรับการควบคุมสถานะของอุปกรณ์เครื่องใช้ไฟฟ้าให้ ON หรือ OFF โดยสามารถเลือกจำนวนห้องที่จะควบคุมได้ 3 แบบคือ

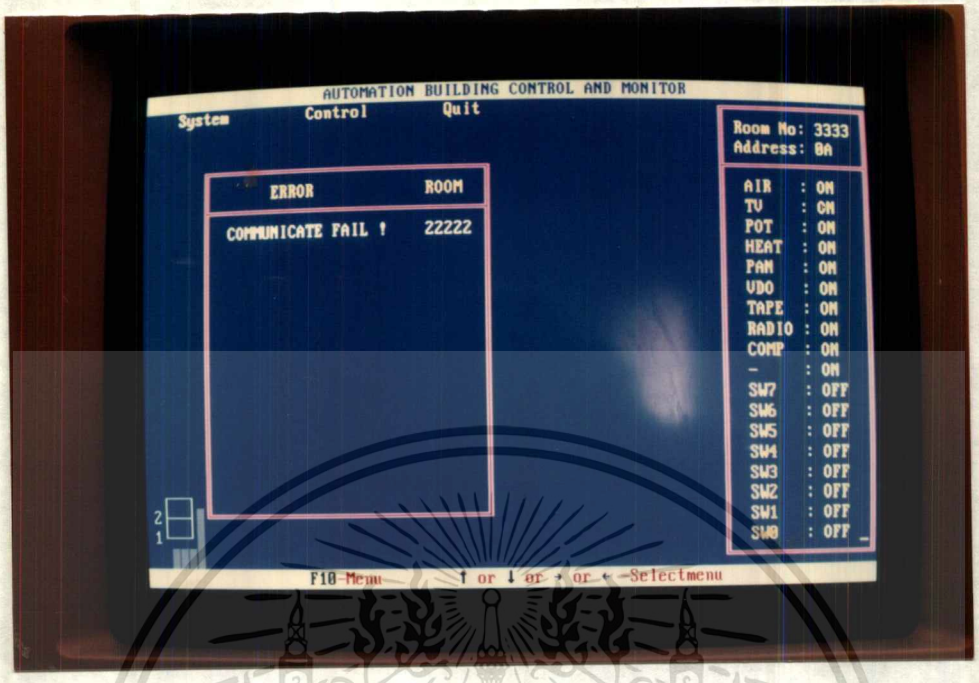
ก) เลือกควบคุมเพียงห้องเดียว โดยการป้อนหมายเลขห้องที่ต้องการควบคุมลงไปแล้วทำการควบคุมโดยการกด Enter ที่ชนิดของเครื่องที่ทำการควบคุม การกดจะเป็นลักษณะของท็อกเกิล กด 1 ครั้งจะเป็นสถานะเดิม กดอีกทีจะเป็นลักษณะตรงข้าม ON <->OFF

ข) เลือกควบคุมเป็นกลุ่ม โดยการป้อนหมายเลขห้องเป็นกลุ่มเช่น 1200 - 2000 แล้วทำการควบคุมเครื่องใช้ไฟฟ้าต่อไป

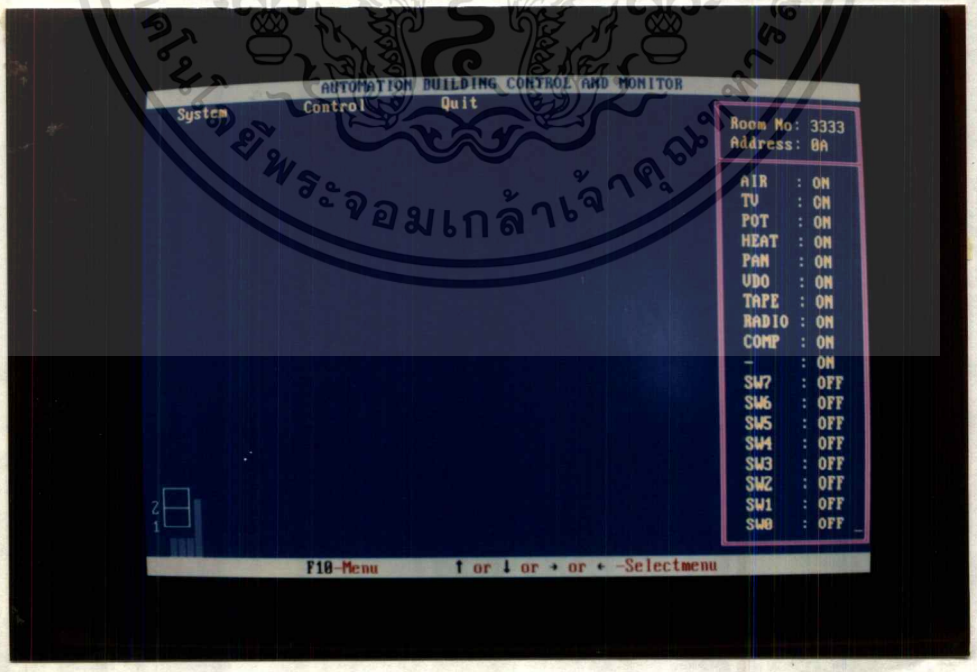
ค) จะควบคุมทั้งหมดทุกเครื่องที่ต่ออยู่ในระบบ ให้พิมพ์คำว่า "ALL" แล้วควบคุมสถานะเครื่องใช้ไฟฟ้า ซึ่งจะเหมือนกันหมดทั้งระบบ

3) Clear deivce เป็นเมนูที่ใช้สำหรับการรีเซทอุปกรณ์อิเล็กทรอนิกส์ (ประเภท รีเลย์ภายในห้องที่กำหนดไว้เพียงห้องใดห้องหนึ่งเท่านั้น

4.4.3 เมนู Quit เป็นเมนูที่จะใช้เมื่อจะยกเลิกการ Run โปรแกรม เมื่อกดเมนูนี้หน้าจอจะแสดงการยืนยันว่าจะออกจากโปรแกรมหรือไม่ โดยการให้กด "Y" หรือ "N" อีกครั้งหนึ่ง

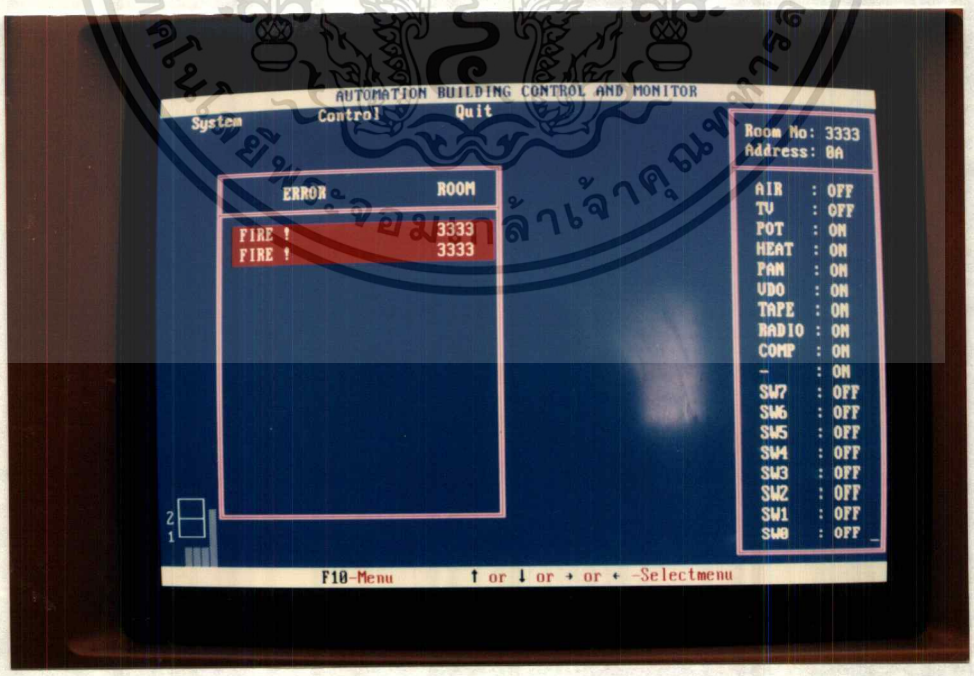


รูปที่ 4.2 แสดงการ ERROR ในกรณี ที่ไม่ต่อ Microcontroller กับ PC

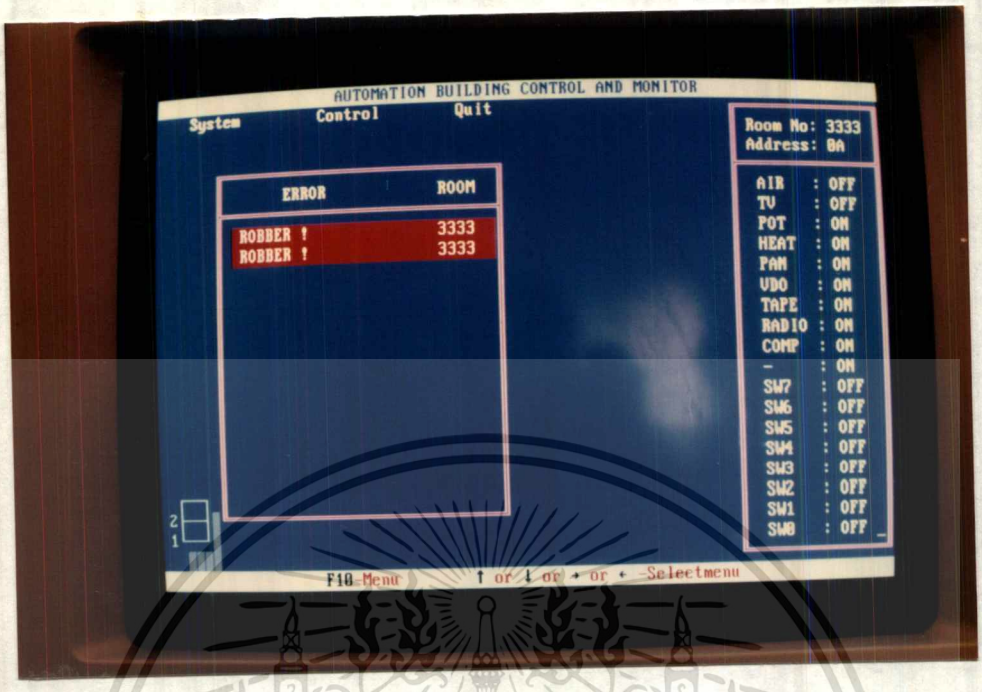


รูปที่ 4.3 แสดงการอ่านสถานะของเครื่องใช้ไฟฟ้าซึ่งปรับ DIP S.W. เป็น $0A_H$

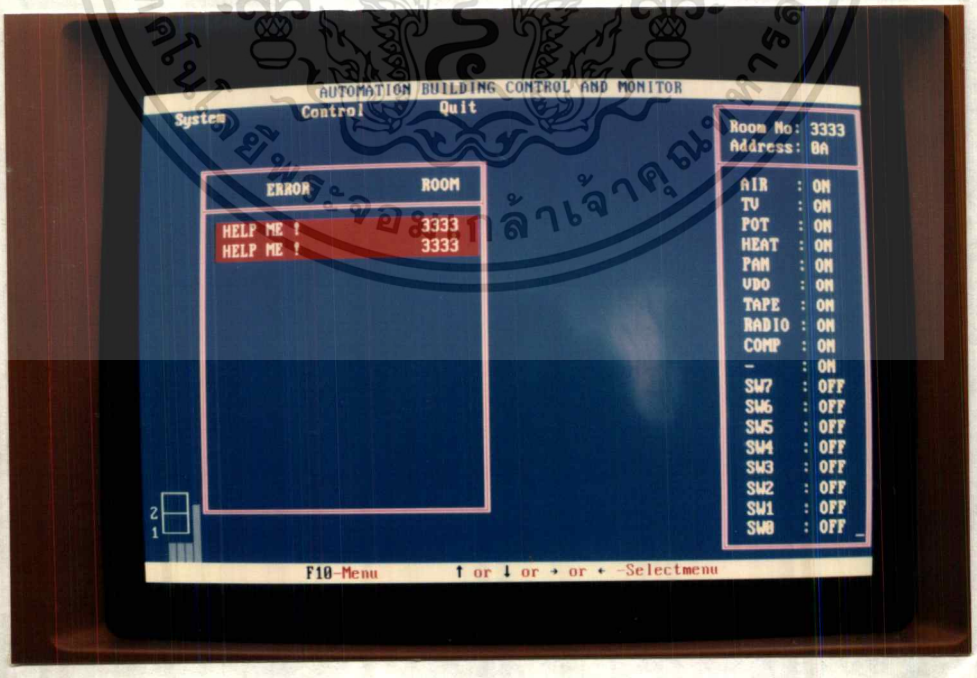
รูปที่ 4.4 แสดงการอ่านสถานะของเครื่องใช้ไฟฟ้าซึ่งปรับ Dip S.W. เป็น $0D_H$



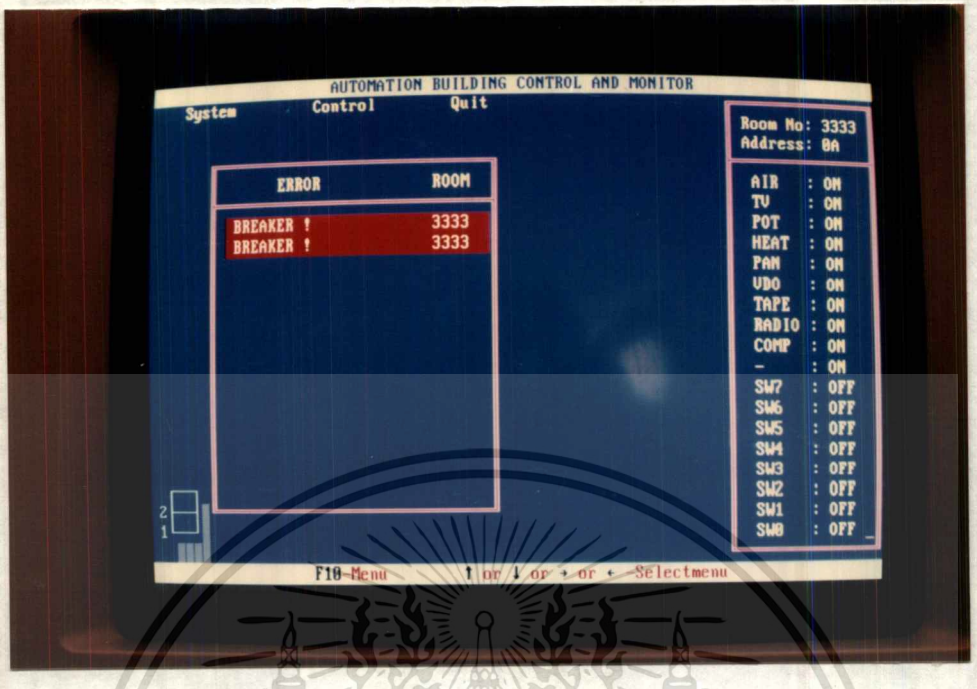
รูปที่ 4.5 แสดงการแจ้งเหตุการณ์ฉุกเฉินโดยให้ PC7 ON (FIRE)



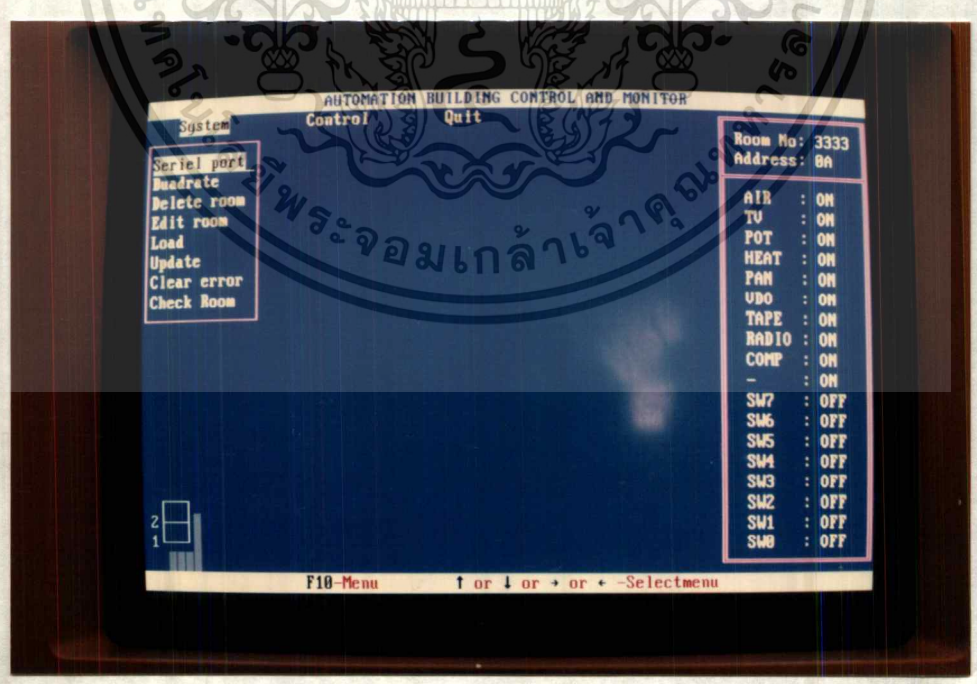
รูปที่ 4.6 แสดงการแจ้งเหตุการณ์ฉุกเฉินโดยให้ PC6 ON (ROBBER)



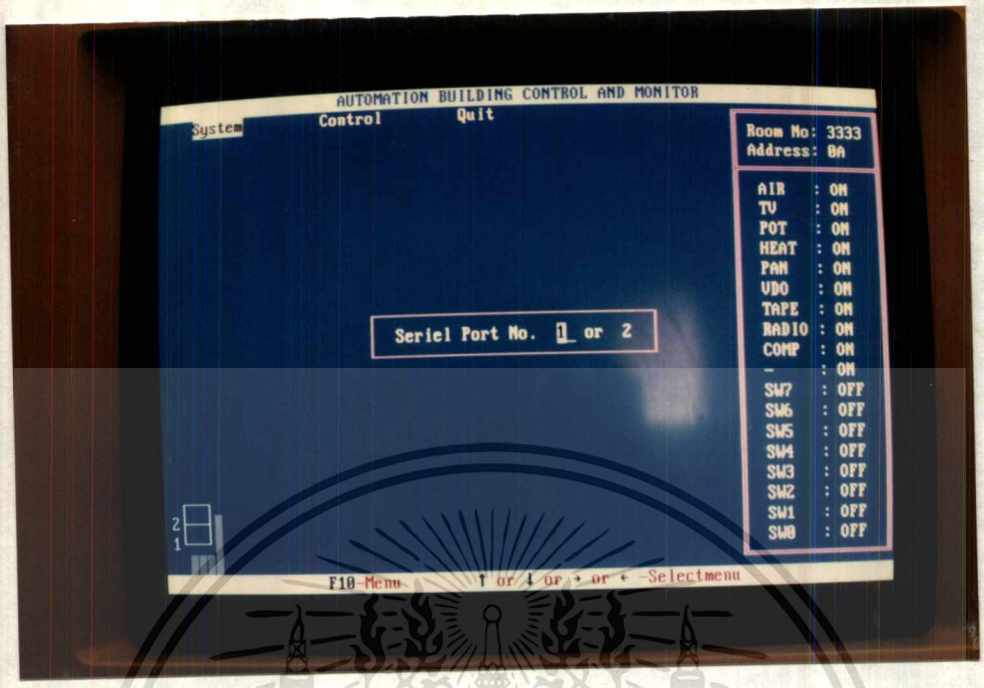
รูปที่ 4.7 แสดงการแจ้งเหตุการณ์ฉุกเฉินโดยให้ PC5 ON (HELP ME)



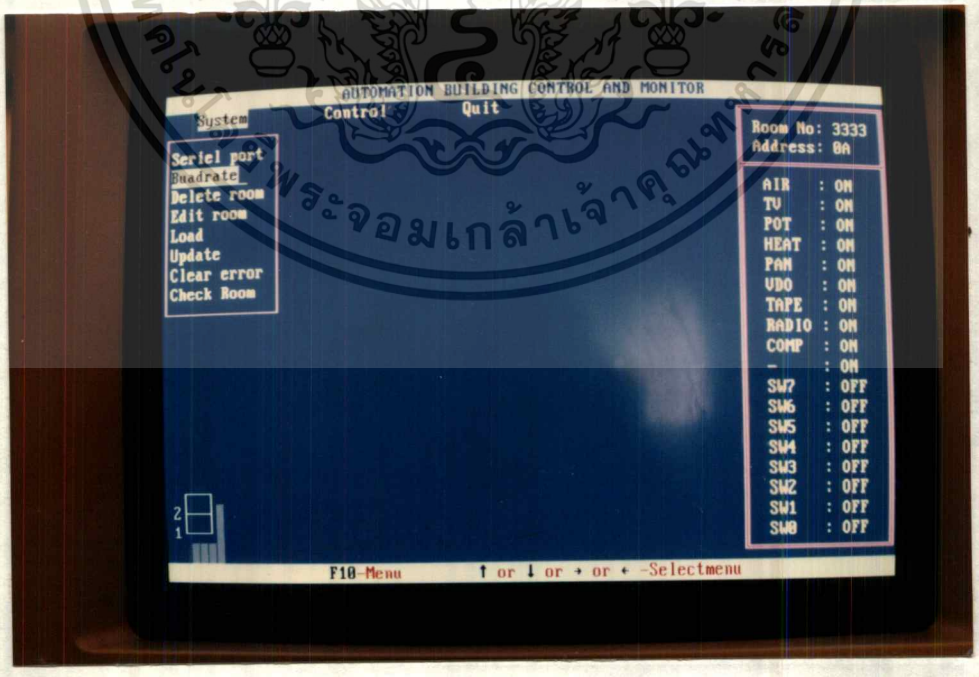
รูปที่ 4.8 แสดงการแจ้งเหตุการณ์ฉุกเฉินโดยให้ PC4 ON (BREAKER)



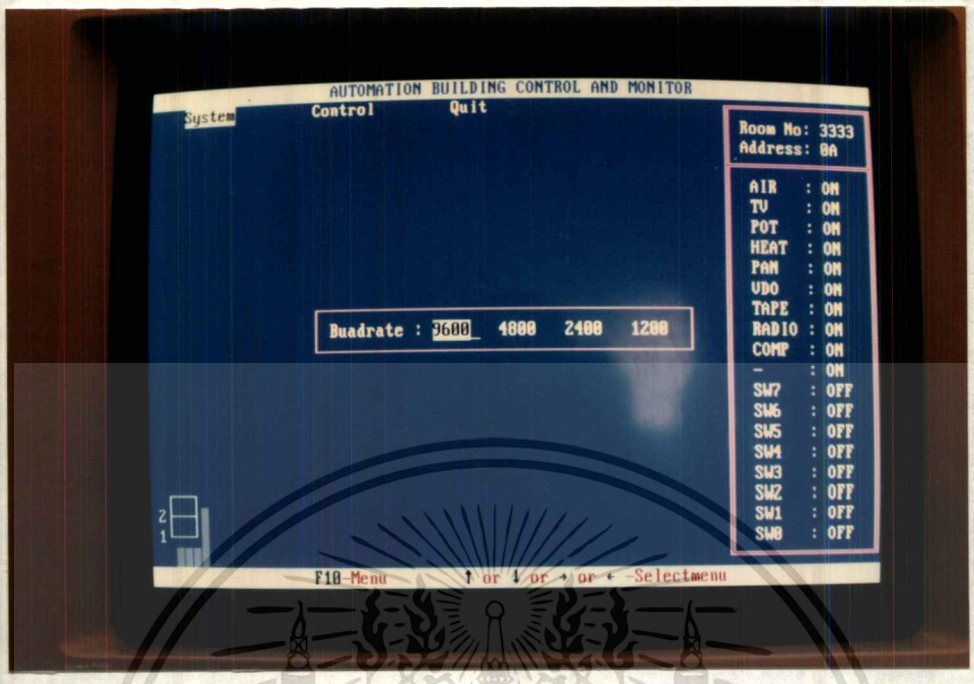
รูปที่ 4.9 แสดงหน้าจอขณะมีการเลือก Menu หลัก System



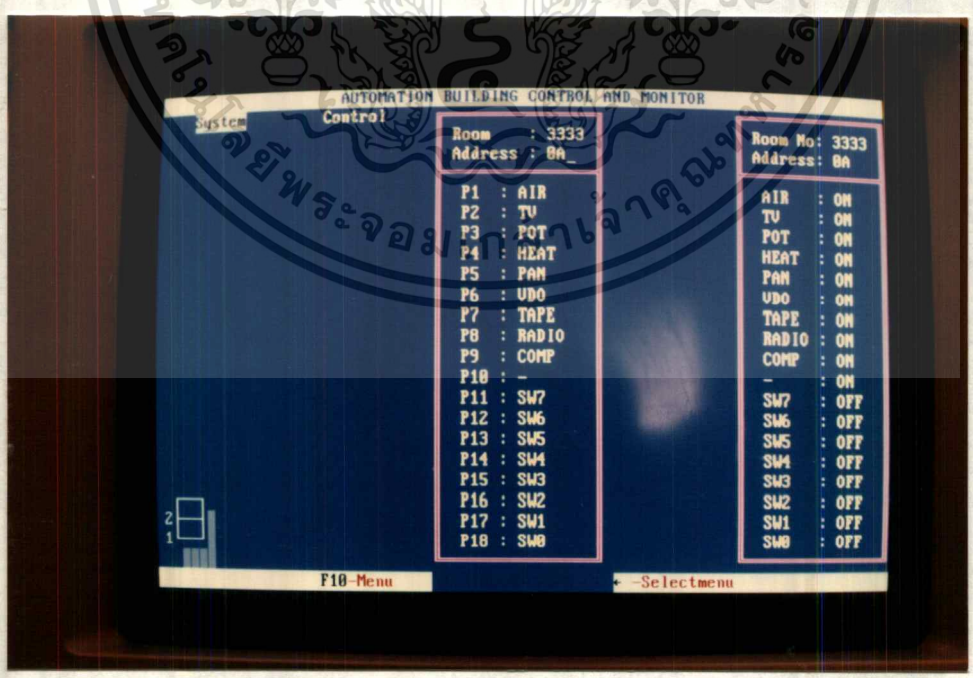
รูปที่ 4.10 แสดงหน้าจอขณะเลือก Serial Port



รูปที่ 4.11 แสดงการเลือก Menu Baudrate

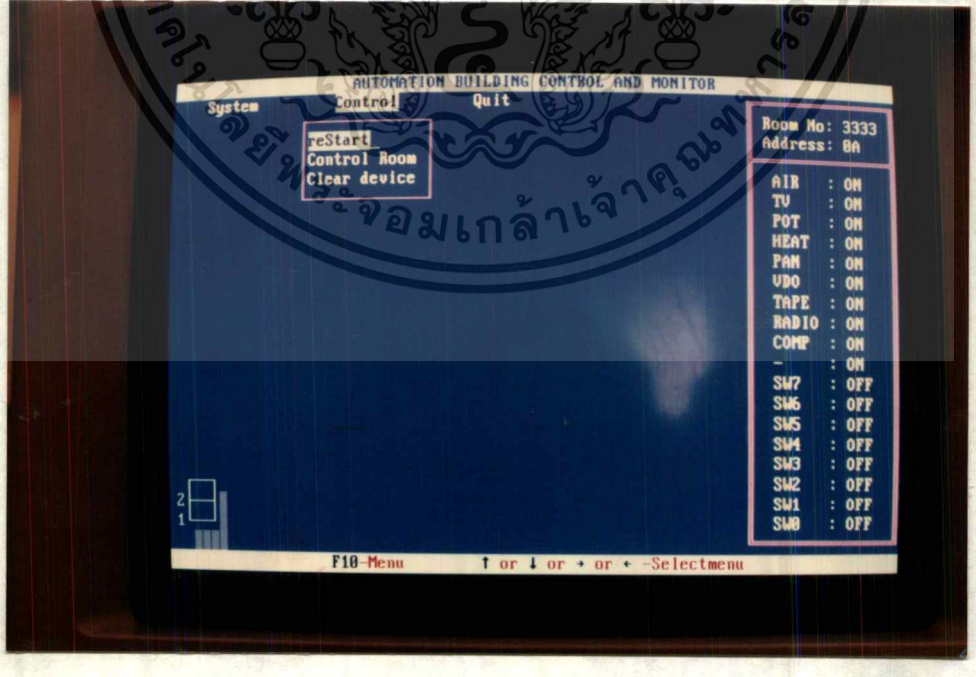


รูปที่ 4.12 แสดงหน้าจอในขณะเลือก Menu ย่อย Baudrate แล้ว

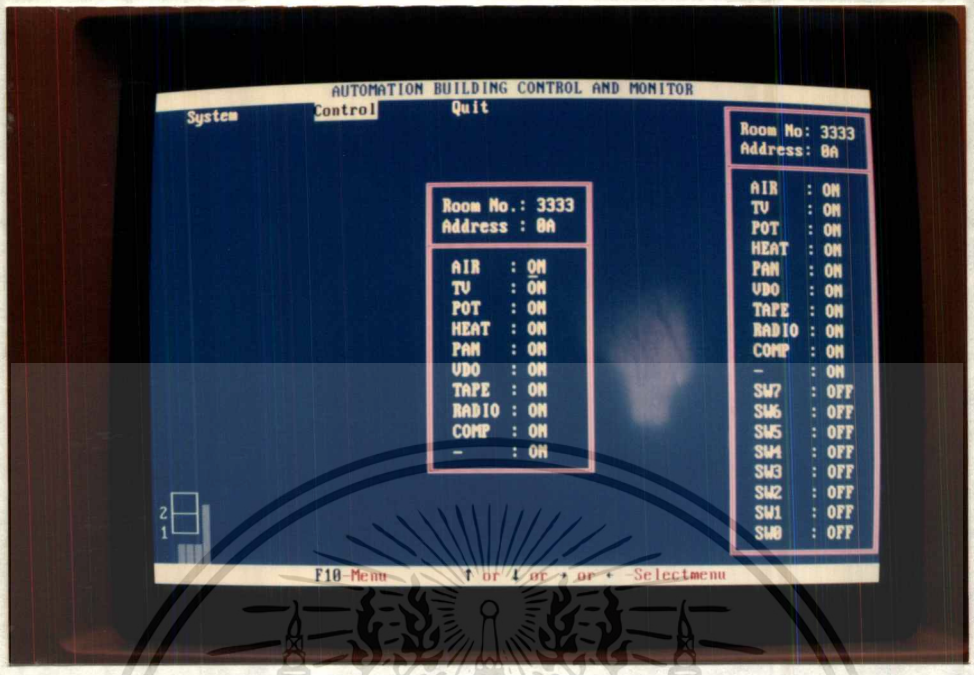


รูปที่ 4.13 แสดงหน้าจอที่เกิดจากการเลือก Edit

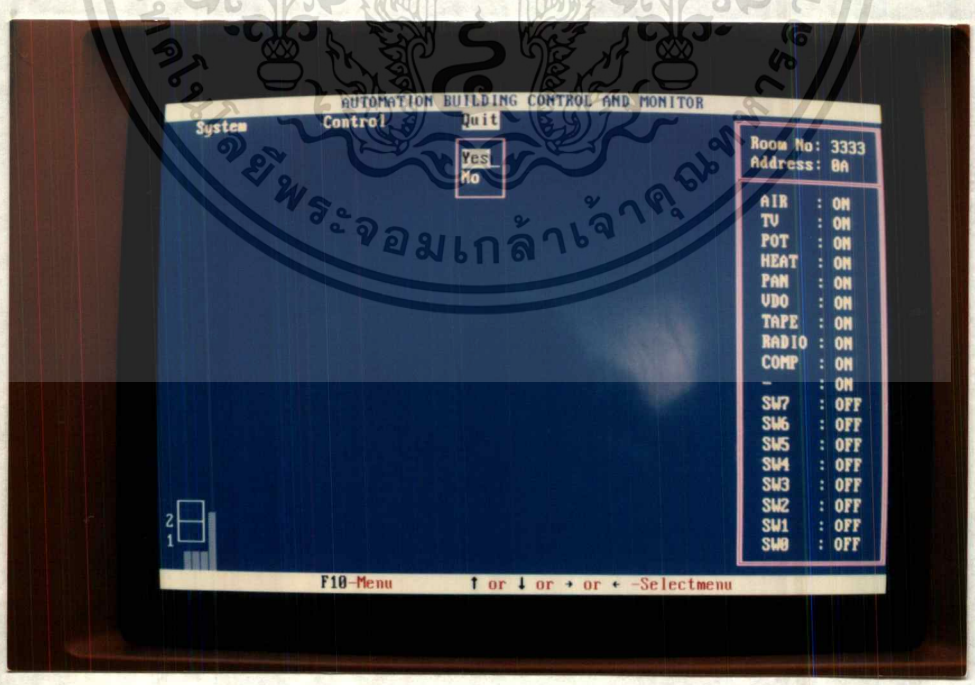
รูปที่ 4.14 แสดงหน้าจอที่เกิดจากการเลือก Menu หลัก Control



รูปที่ 4.15 แสดงหน้าจอที่เกิดจากการเลือก Menu ย่อย Restart



รูปที่ 4.16 แสดงหน้าจอเมื่อมีการเลือก Menu Control Room



รูปที่ 4.17 แสดงหน้าจอเมื่อต้องการออกจากโปรแกรม

บทที่ 5
สรุปผลของโครงงานและข้อเสนอแนะ

5.1 คำนำ

โครงงานการพัฒนาโปรแกรมสำหรับตรวจจับและควบคุมระบบในอาคาร ได้จัดทำขึ้นเป็นไปตามลำดับและขั้นตอนการดำเนินงาน จนสำเร็จลุล่วงตามเป้าหมายที่วางไว้และสามารถนำโครงงานนี้ไปประยุกต์ใช้ควบคุมและตรวจจับระบบอื่น ๆ ได้ตามความเหมาะสม

5.2 สรุปผลที่ได้รับจากโครงงาน

จากการที่ได้จัดสร้างโครงงานพัฒนาโปรแกรมสำหรับตรวจจับและควบคุมระบบในอาคาร เพื่อใช้ในงานอุตสาหกรรม ทำให้ได้เครื่องมือที่ช่วยในการตรวจสอบและควบคุมสภาวะการทำงานของเครื่องจักรในโรงงาน หรือเครื่องใช้ไฟฟ้าภายในอาคาร โดยใช้ควบคุมกับระบบ Single Phase ซึ่งการตรวจสอบและควบคุมนี้จะทำผ่านเครื่อง PC โดยจะติดต่อกับ Microcontroller ผ่านระบบ RS - 485 การตรวจสอบและควบคุมนั้นจะกระทำในลักษณะข้อมูลที่เป็นแบบ HDLC

ผลจากโครงงานนี้ นอกจากจะใช้ควบคุมและตรวจสอบสภาวะการทำงานของเครื่องจักร และยังสามารถประยุกต์ใช้ในการตรวจสอบและควบคุม เครื่องใช้ไฟฟ้าอื่น ๆ ที่ติดตั้งบริเวณกว้างๆ ได้

5.3 ข้อเสนอแนะในการพัฒนาโครงงาน

5.3.1 ไมโครคอนโทรลเล่อร์ที่สร้างขึ้นสามารถที่จะรับสัญญาณ Sensor Unit ได้ 12 ชนิด และสามารถควบคุมได้ 10 ชนิด โดยสามารถขยายอินพุทและเอาท์พุท ได้อีก 2 - 3 Unitต้องพัฒนา Software ที่ใช้ควบคุมไมโครลคอนโทรลเลอร์

5.3.2 การที่จะพัฒนาระบบการรับส่งข้อมูลด้วย HDLC จะให้มีประสิทธิภาพนั้นควรจะเปลี่ยนใช้ CPU ที่มีคุณภาพสูงขึ้น เช่น 8085 ร่วมกับ 8250 หรือสูงขึ้นใช้ IBM PC มาทำงานแทน

5.3.3 การที่จะขยายข่ายการสื่อสารข้อมูลควรจะใช้เครื่องคอมพิวเตอร์ระดับ PC หลาย ๆ เครื่องเป็นหน่วยควบคุมย่อยๆ แล้วส่งข้อมูลมาให้ศูนย์กลางอีกทีหนึ่ง จะทำให้ประสิทธิภาพการใช้งานดีขึ้น

5.3.4 ควรพัฒนาด้าน Software ได้ดียิ่งขึ้นควบคู่กันไป

5.4 <u>ปัญหาและอุปสรรคในการจัดทำโครงงาน</u>

5.4.1 ระบบสื่อสารด้วย HDLC Frame Format เป็นระบบสื่อสารข้อมูลที่มีประสิทธิภาพของ Hardware กล่าวคือ 8031 ทำการรับส่งข้อมูลได้ช้ามาก เมื่อเปรียบเทียบกับคอมพิวเตอร์ที่ใช้ในระบบ HDLC เต็มรูปแบบ (หรือ PC)

5.4.2 การพัฒนาโปรแกรมบน 8031 ต้องใช้ภาษาแอสเซมบลี้ที่มีคำสั่งในการใช้งานน้อย ยุ่งยากต่อการเขียน software ควบคุม นอกจากนี้เครื่องมือในการพัฒนามีน้อย คุณภาพต่ำซึ่งต่างกับการพัฒนาบน PC ที่มีเครื่องมืออำนวยความสะดวกมากมายและมีประสิทธิภาพสูง

5.4.3 รายละเอียดเกี่ยวกับตัว CPU 8031 ค่อนข้างหาได้ยาก ไม่เป็นที่แพร่หลาย จึงต้องใช้เวลาในการศึกษามากซึ่งมีเวลาจำกัด

5.4.4 การสร้าง Hardware ต้องใช้แผ่นพิมพ์วงจรแบบ Plate Throught Hold ซึ่งต้องจ้างร้านทำ ซึ่งราคาสูง มีร้านรับทำน้อยและใช้เวลาทำนาน

ภาคผนวก

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>

#define BORDER 1
#define ESC 27
#define REV_VID 0
#define NORM_VID 1
#define MAXX 80
#define MAXY 25
#define MAXstring 15
#define MAXDEVICE 18
#define TEXTCOLOR YELLOW
#define BORDERCOLOR LIGHTMAGENTA
#define BACKGROUND BLUE
#define BUILDCOLOR LIGHTGRAY
#define HOTKEYCOLOR RED
#define maxroom 160
#define maxlet 7
#define maxport 24
#define useport 18
#define INTR 0X1C

char *system_[]=
  "Seriel port",
  "Buadrate",
  "Delete room",
  "Edit room",
  "Load",
  "Update",
  "Clear error"
;

char *control[]=
```

```
 "reStart",
 "control Room",
 "Clear device"
;

char *quit[]=
 "Yes",
 "No"
;

char *mainmenu[] =
   "System",
   "Control",
   "Quit"
;

 char *serielport[] =
    "1",
    "2"
 ;

  char *BUADRATE[] =
   "9600",
   "4800",
   "2400",
   "1200"
  ;

   char *ERROR[] =
    "",
    " FIRE !",
    " ROBBER !",
    " HELP ME !",
    " BREAKER !",
    " COMMUNICATE FAIL !",
    " DATA ERROR !",
    " FILE NOT FOUND !",
    " ROOM NON SPECIFY !",
    " DISK NOT READY !"
```

```
;

char *device[] = "P1  :","P2  :","P3  :","P4  :","P5  :","P6  :","P7  :",
       "P8  :","P9  :","P10 :","P11 :","P12 :","P13 :","P14 :",
       "P15 :","P16 :","P17 :","P18 :" ;

char fname[MAXstring] = "CONFIG.BLD";
unsigned int  dataIn[7] ;
unsigned int  dataOut[7];
char address[maxroom][maxlet];
char room[maxroom][maxlet];
char port[maxroom][maxport][maxlet];
char statusA[8][4], statusB[8][4], statusC[8][4]; /* port A B C */
char show1[20][maxlet], show2[20][maxlet];
char Speed[5]= "9600";
char Comport[2] = "2";
int  indexData = 0;   /* total room in table */
int  Error = 0;
int  currentIndex = 0;
char roomNo[maxlet*2+1];
int  Maxt = 20;
int  Rxndx, Txndx ;
int  Count = 0,fclrdevice=0;
unsigned int  Cardport = 0;
int Repeat = 0,ERRORselect=0,lineerror;
int QUIT = 0;
char p[80*25*2];

void initialscreen();
void drawbuilding(int build);
void cursor_off(void);
void cursor_on(void);
int popup(char *menu[], char *keys, int count, int x, int y, int border);
void save_video(int startx, int endx, int starty, int endy,
   char *buf_ptr);
void restore_video(int startx, int endx, int starty, int endy,
      char *buf_ptr);
void write_video(int x, int y, char *p, int attrib);
void display_menu(char *menu[], int x, int y, int count);
```

```
void draw_border(int startx, int starty, int endx, int endy, int thickness);
int get_resp(int x, int y, int count, char *menu[], char *keys);
int readkey(void);
void draw_ch_line(int x,int y,int x1,int y1,char ch);
void hotkey_menu();
int get_resp_VER(int x, int y, int count, char *menu[], char *keys, int wide);
int selectVER(char *menu[], char *keys, int count, int x, int y, int wide);
void seriel_port(char PORT[2]);
void room_menu(char roomstring[15]);
void buadrate_menu(char BUADRATE[5]);
void filename_menu(char fname[MAXstring]);
void input_name_device();
void show_state();
void control_status();
void readchar(int x, int y, int num, char data[MAXstring]);
void error_menu(int error);
void beep(void);
void interrupt (*oldhandler)(void);


 void interrupt handler(void)

    Count++;
    oldhandler();


/*****************************************************************/

initial()
   int i;
    int comport, baurdrate;
    unsigned int speedmsb,speedlsb,RS232port;
    comport = atoi(Comport);
    switch (comport)

  case 1: RS232port = 0x3f0; break;
  case 2: RS232port = 0x2f0;

    baurdrate = atoi(Speed);
```

```
  switch (baurdrate)

     case  9600 :  speedlsb = 0x0c; speedmsb = 0x00; break;
case  4800 :  speedlsb = 0x18; speedmsb = 0x00; break;
case  2400 :  speedlsb = 0x30; speedmsb = 0x00; break;
case  1200 :  speedlsb = 0x60; speedmsb = 0x00; break;
case   600 :  speedlsb = 0xc0; speedmsb = 0x00; break;

   outportb(RS232port+0x00b, 0x80);
   outportb(RS232port+0x008, speedlsb);
   outportb(RS232port+0x009, speedmsb);
   outportb(RS232port+0x00b, 0x07);   /* data 8 bit 2 stop No parity */

   return 0;


chkcard()
   unsigned int k=0,i;
    for (i=0x300;i<=0x30f;i++)
        outportb(i,0xaa);
  k = inportb(i);
  if(k == 0x5f)
     outportb(i,0xbb);
      k = inportb(i);
      if(k == 0x4f)
        Cardport = i;break;


   return 0;


communicate()

   int i,j,n;
   char k = 0;
   int repeat = 0;
   unsigned char Tmsg[8];
   unsigned char Rmsg[8];
```

```
    Rmsg[1] = Tmsg[1];
  break;

    k=0;

   outportb(Cardport,3);        /* close the door for recieve */
   if(Error==0) break;
  /* while repeat <3 */
  for(i=0;i<7;i++) dataIn[i] = Rmsg[i];
  return;



/*************************************************************************/

function1(int code)

{
  int i;
  switch(code)

    case 0:
          checksum();
          if(Error == 6)
            dataIn[1]=dataOut[1];
             searchAddr();
             error_menu(6);
             break;

          convDataStr();
       show_state();
          break;
    case 1: searchAddr();  /*find and show*/
          error_menu(5);
          break;
    case 2:
          error_menu(6);
          break;

   return;
```

```
unsigned int RS232port, comport;

for(i=0;i<=6;i++) Tmsg[i] = dataOut[i];
comport = atoi(Comport);
switch (comport)

case 1: RS232port = 0x3f0; break;
case 2: RS232port = 0x2f0;

            /***Tx 6 byte ***/
while(repeat<3)

  for (i=0;i<7;i++)
     while (!(k & 0X40))  k = inportb(RS232port+0x0d);
outportb(RS232port+0x08,Tmsg[i]);     /* send data*/
k=0;
delay(2);

if (Tmsg[2]==0x3C||Tmsg[2]==0x6C||Tmsg[2]==0x5C) return; /**Not want Request*/


  Rxndx = 0;
  k = Count = Error = 0;

            /***Rx 6 byte ***/
outportb(Cardport,0);          /* open the door for recieve */
while ((Count < Maxt) &&(Rxndx <7))

  while (!(k & 0X03) && (Count < Maxt))  k = inportb(RS232port+0x0d);
   if(k & 0x01)

  Rmsg[Rxndx] = inportb(RS232port+0x08);
  Rxndx++;
  Maxt = Count + 5;

   else if((Count >= Maxt)&&(Rxndx < 7))

     repeat++;
  Error = 5;
```

```
    Comport[0] = *rp;
    rp++;


count = 1;

    else

i = 0;
while(*rp != '\0' && rp<(strlen(read)+bp))    /* not end line*/

  while(*rp == ' ' || *rp == '\t')
    rp++;
  while(*rp != ' ' && *rp != '\0' && *rp != '\t')

    room[indexData][i] = *rp;
    i++; rp++;
    room[indexData][i] = NULL;

  i = 0;
  while(*rp == ' ' || *rp == '\t')
    rp++;
  while(*rp != ' ' && *rp != '\0' && *rp != '\t')

    address[indexData][i] = *rp;
    i++; rp++;
    address[indexData][i] = NULL;

  i = 0;
  for(nump=0;nump<maxport;nump++)

     while(*rp == ' ' || *rp == '\t')
  rp++;
   while(*rp != ' ' && *rp != '\0' && *rp != '\t' && rp<(strlen(read)+bp))

  port[indexData][nump][i] = *rp;
  i++; rp++;
  port[indexData][nump][i] = NULL;
```

```
        i = 0;
        if(*rp == '\0' *rp == EOF)
       break;
       /* end for port */
   if (strlen(read)>4)   /*  protect */
        indexData++;
        /* end line */

  /* end eof*/
  fclose(fp);
  return;


checksum()

   int sum=0,index=0;
   if(dataIn[0] != 0x7e)
         Error = 6; return;

   for(index=0;index<5;index++)
      sum = sum+dataIn[index];
   sum = sum+dataIn[6];      /* add with stop bit */
   sum = sum%0xff;
   if(sum == dataIn[5])     /* summing O.K. */
      return;
   else                     /* summing Error */
   Error = 6;       /* check summing Error*/
   return;


 convDataStr()
 {
    int oper=0x80,pos=0,count,temp,ander,result;
    char status[4];
    for(count=2;count<5;count++)

       temp = dataIn[count];
       for(pos=0;pos<8;pos++)
```

```
ander = oper >> pos;      /* shift bit */
result = temp&ander;
if(result==0)
  strcpy(status,"OFF");
else
  strcpy(status,"ON");
switch(count)

 case 2:
strcpy(statusC[pos],status);
break;
 case 3:
strcpy(statusA[pos],status);
break;
 case 4:
strcpy(statusB[pos],status);
break;


searchAddr();

 for(count=0;count<4;count++)

   if (strcmp(statusC[count],"ON")==0)
temp = 1;
   else
temp = 0;
   switch(count)

case 0:
if (temp == 1)
   error_menu(1);  /* fire */
else
 Error = 0;
       break;
case 1:
if (temp == 1)
 error_menu(2);  /* robber */
 else
```

```
      Error = 0;
            break;
     case 2:
     if (temp == 1)
      error_menu(3);  /* Help! */
     else
      Error = 0;
            break;
     case 3:
      if (temp == 1)    /* normal error */
      error_menu(4);
      else
      Error = 0;
      break;

 if (temp == 1 )
    return;

    return;


sortTable()

   int i,j,change;
   char tempR[maxlet],tempA[maxlet];
   char tempP[maxlet];
   do

      change = 0;
      for (i=0;i<indexData-1;i++)
  if (strcmp(&room[i][0],&room[i+1][0])>0)

       strcpy(tempR,&room[i][0]);
       strcpy(&room[i][0],&room[i+1][0]);
       strcpy(&room[i+1][0],tempR);
       strcpy(tempA,&address[i][0]);
       strcpy(&address[i][0],&address[i+1][0]);
       strcpy(&address[i+1][0],tempA);
       for (j=0;j<useport;j++)
```

```
    strcpy(tempP,&port[i][j][0]);
    strcpy(&port[i][j][0],&port[i+1][j][0]);
    strcpy(&port[i+1][j][0],tempP);

    change = 1;

    while (change);
    return;

searchAddr()

    int i,j,count;
    char temp[3];

    sprintf(temp,"%02X",dataIn[1]);   /* int to string */
    for(i=0;i<indexData;i++)

      if((strcmp(temp,address[i]))==0)

        strcpy(show1[0],room[i]);
        strcpy(show2[0],address[i]);
    j = 6;
        for (count=1;count<=18;count++)

      strcpy(show1[count],port[i][count-1]);
      if (count>0 && count<=2)
        strcpy(show2[count],statusC[j]);
      if (count>2 && count<=10)
        strcpy(show2[count],statusA[j]);
      if (count>10)
        strcpy(show2[count],statusB[j]);
          j++;
      if (count==2 || count==10)
            j = 0;

    return;   /* find only one times */
  }
```

```
  return;


/*-----------End function1-----------*/
function2(int code)

   switch(code)

     case 0:
       sequence();
       break;
     case 1:
       outToPort();
       break;
     case 2:
       sortTable();
       saveFile();
       break;
     case 3:
       openfile();
       break;
     case 4:
       sortTable();
       delete();
     textbackground(BACKGROUND);
     window(1,3,63,MAXY-1);
     clrscr();
     window(1,1,80,25);
     ERRORselect = 0;
     drawbuilding(indexData);
       break;
     case 5:
       edit();
       sortTable();
       break;
     case 6:
       readTable();
```

```
      break;
    case 7:
      clrDevice();
      break;
    case 8:
      allWork();
      break;
    case 9:
      reStart();
      break;
    case 10:
      workGroup();
      break;

  return;

sequence()

  char *b;

  dataOut[0] = dataOut[6] = 0x7e;
  dataOut[1] = strtol(address[currentIndex],&b,16);
  dataOut[2] = 0x2c;    /* read port to PC */
  dataOut[3] = dataOut[4] = 0;
  createSum();
  currentIndex = currentIndex+1;
  if (currentIndex >= indexData)
   currentIndex = 0;
  return;

createSum()

  int i,sum=0;
  for(i=0;i<5;i++)
    sum = sum+dataOut[i];
```

```
  sum = sum+dataOut[6];
  sum = sum%0xff;
  dataOut[5] = sum;
  return;


saveFile()

  int i=0,j,k,l;
  FILE *fp;

  if((fp=fopen(fname,"wt"))==NULL)
  {
    error_menu(9);
    return;

  fwrite(Speed,strlen(Speed),1,fp);
  fprintf(fp," ");
  fwrite(Comport,strlen(Comport),1,fp);
  fprintf(fp,"");
  for(i=0;i<indexData;i++)

   if(!isdigit(room[i][0]))

    strcpy(room[i],"NULL");
    strcpy(address[i],"NULL");
    for (j=0;j<useport;j++)
    strcpy(port[i][j],"NULL");
    indexData = indexData-1;
    fclose(fp);
    return;

  strupr(room[i]);
  strupr(address[i]);
  for(j=i+1;j<indexData;j++)
  if (strcmp(room[j],room[i])==NULL)

    for(k=j;k<indexData;k++)
```

```
  while (*pt !='-')

  room1[i] = *pt;  i++;
  room1[i] = NULL;
  pt++;


else
   strcpy(room1,roomNo);
for(i=0;i<indexData;i++)
  if (strcmp(room1,room[i])==NULL)

top = end = i;
if (group == 1)

   for(n=top+1;n<indexData;n++)
     if (strstr(back,room[n])!=NULL)

   end = n;
   break;


ndelete = end-top+1;
indexmove = end+1;
loop = indexData-indexmove;
for(j=end+1;j<(indexmove+loop);j++,top++)

   strcpy(&room[top][0],&room[j][0]);
   strcpy(&address[top][0],&address[j][0]);
   for(n=0;n<useport;n++)
     strcpy(&port[top][n][0],&port[j][n][0]);

for(i=indexData-ndelete;i<indexData;i++)

   strcpy(&room[i][0],NULL);
   strcpy(&address[i][0],NULL);
   for(n=0;n<useport;n++)
     strcpy(&port[i][n][0],NULL);
```

```
strcpy(room[k],room[k+1]);
strcpy(address[k],address[k+1]);
for(l=0;l<useport;l++)
strcpy(port[k][l],port[k+1][l]);

strcpy(room[k],"NULL");
strcpy(address[k],"NULL");
for(l=0;l<useport;l++)
strcpy(port[k][l],"NULL");
indexData = indexData-1;

fwrite(room[i],strlen(room[i]),1,fp);
fprintf(fp," ");
fwrite(address[i],strlen(address[i]),1,fp);
for(j=0;j<useport;j++)

  fprintf(fp," ");
  strupr(port[i][j]);
  if (!strcmp(port[i][j],""))
     strcpy (port[i][j]," -");
  fwrite(port[i][j],strlen(port[i][j]),1,fp);

 if (i!=indexData-1)
fprintf(fp,"");

fclose(fp);
return;


delete()

  int i=0,j,n,loop,group=0,end=0,top=0;
  int ndelete=0,indexmove=0;
  char *back,*pt,room1[maxlet];

  if(strstr(roomNo,"-")!=NULL)
     group = 1;
     pt = roomNo;
     back = strstr(roomNo,"-");   /* back room */
```

```
for(j=0;j<useport;j++)
   strcpy(port[i][j],show1[j+1]);
return;

if (found == 0)

   strcpy(room[indexData],show1[0]);
 strcpy(address[indexData],strupr(show2[0]));
   for(j=0;j<useport;j++)
 strcpy(port[indexData][j],show1[j+1]);
 indexData = indexData+1;

   return;

 return;


outToPort()     /* convStrData for write to port */

 int i,j=2,data=0,bit,count=1,temp=0;
 char *b;

 dataOut[0] = dataOut[6] = 0x7e;  /* start-stop bit */
 dataOut[4] = 0;   /* port B no data */
 for(i=0;i<indexData;i++)
   if (strcmp(room[i],show1[0])==NULL)

     strcpy(show2[0],address[i]);
     dataOut[1] = strtol(address[i],&b,16); /* address */
     break;

 if(fclrdevice==0)

 for(i=1;i<=10;i++)          /* data */

     if (strcmp(show2[i],"ON")==0)
       bit = 1;
     else
       bit = 0;
```

```
  indexData = indexData-ndelete;

  return;


readTable()

  int i,count,found=0;
  if (strlen(show1[0])>maxlet)   /* not group */
     return;
  for(i=0;i<indexData;i++)
     if (strcmp(show1[0],room[i])==NULL)

  found = 1;
        strcpy(show1[0],room[i]);
        strcpy(show2[0],address[i]);
  for (count=1;count<=useport;count++)
     strcpy(show1[count],port[i][count-1]);
  return;

  if (found == 0)

  strcpy(show2[0],NULL);
  for (count=1;count<=useport;count++)
     strcpy(show1[count],NULL);

  return;


edit()

  int i,j,found=0;
  if (strlen(show1[0])>maxlet)   /* not group */
     return;
  for(i=0;i<indexData;i++)
     if (strcmp(room[i],show1[0])==NULL)

  found = 1;
  strcpy(address[i],strupr(show2[0]));
```

```
        temp = bit<<count;
        data = tempIdata;
        if (count==0)

          dataOut[j] = data;
          j++;
      data = 0;
      temp = 0;
      count = 8;

        count--;

      dataOut[2] = dataOut[2]|0x1c;
      createSum();

  return;

clrDevice()

    int i;
    char *b;

    fclrdevice = 1;
    if (strstr(roomNo,"-")!=NULL)

      workGroup();

    else
      outToPort();
    dataOut[2] = 0x5c;  /* command */
    createSum();
    fclrdevice = 0;
    return;

workGroup()
```

```
int i=0,n,top=0,end=0;
int count=1,j=2,temp=0,bit=0,data=0;
char *pt,room1[maxlet],*b;
if(strstr(roomNo,"-")!=NULL)

   pt = roomNo;
   while (*pt !='-')

  room1[i] = *pt;  i++;
  room1[i] = NULL;
 *pt++;


else

   outToPort();
   return;

for(i=0;i<indexData;i++)
   if (strcmp(room1,room[i])==NULL)

top = end = i;
for(n=top+1;n<indexData;n++)
   if (strstr(pt,room[n])!=NULL)
   {
      end = n;
      break;

break;

 dataOut[0] = dataOut[6] = 0x7e;
 strcpy(show2[0],address[top]);
 dataOut[1] = strtol(address[top],&b,16);
 dataOut[4] = strtol(address[end],&b,16);
 if(fclrdevice==0)

   for(i=1;i<=10;i++)          /* data */

      if (strcmp(show2[i],"ON")==0)
```

```
        bit = 1;
      else
        bit = 0;
      temp = bit<<count;
      data = tempdata;
      if (count==0)

        dataOut[j] = data;
        j++;
        data = 0;
     temp = 0;
     count = 8;

      count--;

    dataOut[2] = dataOut[2]|0x3c;
    createSum();

  return;


allWork()

   int i,j=2,count=1,data=0,bit=0,temp=0;
   dataOut[0] = dataOut[6] = 0x7e;
   dataOut[1] = 0;
   dataOut[4] = 0;
  for(i=1;i<=10;i++)          /* data */

     if (strcmp(show2[i],"ON")==0)
        bit = 1;
     else
        bit = 0;
     temp = bit<<count;
     data = tempdata;
   if (count==0)

        dataOut[j] = data;
        j++;
```

```
data = 0;
temp = 0;
count = 8;

    count--;

dataOut[2] = dataOut[2]|0x6c;
createSum();
return;


reStart()

int i;
if (strlen(show1[0])>maxlet)
    return;
for(i=0;i<indexData;i++)
    if (strcmp(show1[0],room[i])<=0)
        strcpy(show1[0],room[i]);
    currentIndex = i;
        break;

return;


void initialscreen()
int n,loop;

    gotoxy(1,1);
textbackground(LIGHTGRAY);
clrscr();
textcolor(BLUE);
cprintf("                AUTOMATION BUILDING CONTROL AND MONITOR");
hotkey_menu();
window(1,2,80,24);
textbackground(BACKGROUND);
clrscr();
window(1,1,80,25);
    textcolor(TEXTCOLOR);
```

```
    for(n=5,loop=0;loop < 3;loop++,n=n+15)
{
        gotoxy(n,2);
         cprintf(mainmenu[loop]);



void hotkey_menu()
  struct text_info initial_info;

 gettextinfo(&initial_info);
 textcolor(HOTKEYCOLOR);
 gotoxy(23,MAXY);       cprintf("-Menu");
 gotoxy(38,MAXY);       cprintf("or  or  or  -Selectmenu");
 textcolor(BLACK);
 gotoxy(20,MAXY);       cprintf("F10");
   gotoxy(38-2,MAXY);     putch(24);
   gotoxy(38+3,MAXY);     putch(25);
   gotoxy(38+3+5,MAXY);   putch(26);
   gotoxy(38+3+5+5,MAXY); putch(27);
   textcolor(initial_info.attribute);



void drawbuilding(int build)

   int i,x,y,loopmax,loopmin,wide=2,high,roomnum=1,length;
   int loop[8],color;
   char nroom[8];
   struct text_info initial_info;

   gettextinfo(&initial_info);
   textcolor(BUILDCOLOR);
   textbackground(BACKGROUND);

   if (build > 0 && build <= maxroom)

     loopmax = build/20;
     loopmin = build%20;
   for (i=0; i<=8; i++)
```

```
loop[i] = 0;

for (i = 0;i < 8 && loopmax > 0; i++,loopmax--)

loop[i] = 20;

  if (loopmin > 0 )

loop[i] = loopmin;

for (i = 0,x = 3; loop[i] != 0 && i < 8; i++,x = x+8)

 y=MAXY-2;
 draw_ch_line( x+1, y+1, x+wide+2, y+1, 'ฑ');
 draw_ch_line( x+1, y, x+wide+2, y, 'ฑ');
 gotoxy( x, y);        putch('ภ');
 gotoxy( x+wide+1, y); putch('');
 draw_ch_line( x+1, y, x+wide, y, 'ฦ');
 itoa(roomnum,nroom,10);
 gotoxy( x-strlen(nroom), y);
 cputs(nroom);
 roomnum++;
 for ( y=MAXY-3; loop[i] >1; loop[i]--,y--)

  draw_ch_line( x+1, y, x+wide+2, y, 'ฑ');
  gotoxy( x, y);        putch('ว');
  gotoxy( x+wide+1, y); putch('ค');
  draw_ch_line( x+1, y, x+wide, y, 'ฦ');
  itoa( roomnum, nroom, 10);
  gotoxy( x-strlen(nroom), y);
  cputs(nroom);
  roomnum++;

  gotoxy( x, y);        putch('');
  gotoxy( x+wide+1, y); putch('ท');
  draw_ch_line( x+1, y, x+wide, y, 'ฦ');


  textcolor(initial_info.attribute);
```

```
void draw_ch_line(int x,int y,int x1,int y1,char ch)
 for (; y <= y1;y++)
  for (; x <= x1; x++)
     gotoxy(x,y);  putch(ch);



/* Turn off the cursor*/
void cursor_off(void)

 union REGS r;

 r.h.ah = 1;      /* remove the cursor */
 r.h.ch = 8;
 r.h.cl = 8;
 int86(0x10,&r,&r);



/* Turn on the cursor */
void cursor_on(void)

 union REGS r;

 r.h.ah = 1;  /* show the cursor */
 r.h.ch = 7;
 r.h.cl = 7;
 int86(0x10,&r,&r);



 /* Display a pop-up menu and return selection.
    This function returns -2 if menu cannot be constructed;
    it returns -1 if user hits escape key;
    otherwise the item number is returned starting
    with 0 as the first (topmost) entry.
 */
 int popup(
   char *menu[], /* menu text */
```

```
  char *keys,   /* hot keys */
  int count,    /* number of menu items */
  int x, int y, /* X,Y coordinates of left hand corner */
  int border    /* no border if 0 */
)

  int i, len;
  int endx, endy, choice;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textcolor(TEXTCOLOR);
  textbackground(BACKGROUND);
  len = 0;
  for(i=0; i<count; i++)
   if(strlen(menu[i]) > len) len = strlen(menu[i]);
  endx = len + 2 + x;
  endy = count + 1 + y;

  /* save what is currently on the screen */
  save_video(x, endx+1, y, endy+1, p);

  if(border) draw_border(x, y, endx, endy,1);

  /* display the menu */
  display_menu(menu, x+1, y+1, count);

  /* get the user's response */
  choice = get_resp(x, y, count, menu, keys);

  /* restore the original screen */
  restore_video(x, endx+1, y, endy+1, (char *) p);

  textcolor(initial_info.attribute);
  return choice;


/* Display the menu in its proper location. */
void display_menu(char *menu[], int x, int y, int count)
```

```
  register int i;
  for(i=0; i<count; i++, y++)
    gotoxy(x, y);
  cprintf(menu[i]);



/* Draw a border around the menu. */
void draw_border(int startx, int starty, int endx, int endy, int thickness)

  int i,lineHOR,lineVER,LEFT_TOP,LEFT_BOTTOM,RIGHT_TOP,RIGHT_BOTTOM,oldcolor;
  struct text_info initial_info;

  if (thickness == 1)
     lineHOR      = 'ฤ';
     lineVER _    = 'ณ';
     LEFT_TOP _   = '';
     LEFT_BOTTOM  = 'ภ';
     RIGHT_TOP    = 'ฬ';
     RIGHT_BOTTOM = '';

  else if (thickness == 2)
   lineHOR      = 'ธ';
   lineVER      = 'บ';
   LEFT_TOP     = 'ศ';
   LEFT_BOTTOM  = 'ฬ';
   RIGHT_TOP    = 'ป';
   RIGHT_BOTTOM = 'ผ';


   gettextinfo(&initial_info);
   textcolor(BORDERCOLOR);
   /* draw vertical lines */
   for(i=starty+1; i<endy; i++)
    gotoxy(startx, i) ;
     putch(lineVER);
     gotoxy(endx, i);
     putch(lineVER);
```

```
/* draw horizontal lines */
for(i=startx+1; i<endx; i++)
  gotoxy(i, starty);
  putch(lineHOR);
  gotoxy(i, endy);
  putch(lineHOR);

/* draw the corners */
gotoxy(startx, starty); putch(LEFT_TOP);
gotoxy(startx, endy); putch(LEFT_BOTTOM);
gotoxy(endx, starty); putch(RIGHT_TOP);
gotoxy(endx, endy); putch(RIGHT_BOTTOM);
textcolor(initial_info.attribute);

/* Input user's selection. */
int get_resp(int x, int y, int count, char *menu[], char *keys)

  union inkey
    char ch[2];
    int i;
  c;
  int arrow_choice=0;
  char *key_choice;

  x++;
  y++;

  /* highlight the first selection */
  gotoxy( x, y);
  write_video(x, y, menu[0], REV_VID); /* reverse video */

  for(;;)
    c.i = readkey();   /* read the key */

    /* reset the selection to normal video */
```

```
  gotoxy(x, y+arrow_choice);
  write_video(x, y+arrow_choice,
              menu[arrow_choice], NORM_VID); /* redisplay */

 if(c.ch[0])  /* is normal key */
    /* see if it is a hot key */
    key_choice = strchr(keys, tolower(c.ch[0]));
    if(key_choice) return key_choice-keys;

    /* check for ENTER or space bar */
    switch(c.ch[0])
      case '': return arrow_choice;
      case ' ' : arrow_choice++;
       break;
      case ESC : return -1; /* cancel */


   else   /* is special key */
     switch(c.ch[1])
       case 72: arrow_choice--; /* up arrow */
     break;
       case 80: arrow_choice++; /* down arrow */
     break;
   case 75: return -2;
   case 77: return -3;


    if(arrow_choice==count) arrow_choice = 0;
    if(arrow_choice<0) arrow_choice = count-1;

    /* highlight the next selection */
    gotoxy(x, y+arrow_choice);
    write_video(x, y+arrow_choice, menu[arrow_choice], REV_VID);



/* Display a string with specified attribute. */
void write_video(int x, int y, char *p, int attrib)
```

```
union REGS r;
int i;
struct text_info initial_info;

gettextinfo(&initial_info);
gotoxy(x,y);
if (attrib == REV_VID)
   textcolor(BLACK);
 textbackground(LIGHTGRAY);

if (attrib == NORM_VID)
  textcolor(TEXTCOLOR);
  textbackground(BACKGROUND);

 cprintf("%s",p);
 textcolor(initial_info.attribute);
 textbackground(initial_info.attribute);



/* Save a portion of the screen. */
void save_video(int startx, int endx, int starty, int endy,
   char *buf_ptr)

  int i,j;

  gettext(startx,starty,endx,endy,buf_ptr);
  textbackground(BACKGROUND);
  window(startx,starty,endx,endy);
  clrscr();
  window(1,1,80,25);



 /* Restore a portion of the screen. */
 void restore_video(int startx, int endx, int starty, int endy,
   char *buf_ptr)

   puttext(startx,starty,endx,endy,buf_ptr);
```

```
/* Return the 16-bit scan code from the keyboard. */
readkey(void)

  union REGS r;

  r.h.ah = 0;
  return int86(0x16, &r, &r);


/* Input user's selection. */
int get_resp_VER(int x, int y, int count, char *menu[], char *keys, int wide)

  union inkey
  char ch[2];
  int i;
   c;
  int arrow_choice=0;
  char *key_choice;

  /* highlight the first selection */
  gotoxy( x, y);
  write_video(x, y, menu[0], REV_VID); /* reverse video */

  for(;;)
    c.i = readkey();    /* read the key */

    /* reset the selection to normal video */
    gotoxy(x+(arrow_choice*wide), y);
    write_video(x+(arrow_choice*wide), y,
                menu[arrow_choice], NORM_VID); /* redisplay */

    if(c.ch[0])  /* is normal key */
      /* see if it is a hot key */
      key_choice = strchr(keys, tolower(c.ch[0]));
      if(key_choice) return key_choice-keys;

      /* check for ENTER or space bar */
      switch(c.ch[0])
        case '': return arrow_choice;
```

```
    case ' ' : arrow_choice++;
      break;
    case ESC : return -1; /* cancel */


  else  /* is special key */
    switch(c.ch[1])
case 75: arrow_choice--; /* left arrow */
      break;
case 77: arrow_choice++; /* right arrow */
      break;


  if(arrow_choice==count) arrow_choice = 0;
  if(arrow_choice<0) arrow_choice = count-1;

  /* highlight the next selection */
  gotoxy(x, y+arrow_choice);
  write_video(x+(arrow_choice*wide), y, menu[arrow_choice], REV_VID);



int selectVER(
  char *menu[], /* menu text */
  char *keys,   /* hot keys */
  int count,    /* number of menu items */
  int x, int y, /* X,Y coordinates of left hand corner */
  int wide      /* distance between startmenu1 to startmenu2 */
)

  int i, len, endx, endy, choice, n;
  char *p;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textcolor(TEXTCOLOR);
  textbackground(BACKGROUND);
  /* display the menu */
  for(i=0, n=x; i<count; i++, n=n+wide)
```

```
    gotoxy(n, y);
  cprintf(menu[i]);

  /* get the user's response */
  choice = get_resp_VER(x, y, count, menu, keys, wide);
  textcolor(initial_info.attribute);
  textbackground(initial_info.attribute);
  return choice;


/*  select number seriel port to tranfer data */
void seriel_port(char PORT[2])
  int port;
   int x = 25,
       y = 12,
       endx = 55,
     endy = 14;
   struct text_info initial_info;

   gettextinfo(&initial_info);
   textcolor(TEXTCOLOR);
   textbackground(BACKGROUND);

   /* save what is currently on the screen */
   save_video(x, endx+1, y, endy+1, p);

   draw_border(x, y, endx, endy,1);
   gotoxy(x+3,y+1); cprintf("Seriel Port No.     or");
   port = selectVER(serielport,"12",2,x+20,y+1,7);
   if (port == 0)  strcpy(PORT,"1");
   if (port == 1)  strcpy(PORT,"2");
   restore_video(x, endx+1, y, endy+1, (char *) p);
   textcolor(initial_info.attribute);
   textbackground(initial_info.attribute);


 void room_menu(char roomstring[15])
   int x = 28,
       y = 12,
```

```
    endx = 52,
    endy = 14;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textcolor(TEXTCOLOR);
  textbackground(BACKGROUND);

  /* save what is currently on the screen */
  save_video(x, endx+1, y, endy+1, p);

  draw_border(x, y, endx, endy,1);
  gotoxy(x+2,y+1); cprintf("Room No.:");
  readchar(x+12,y+1,11,roomstring);

  restore_video(x, endx+1, y, endy+1, (char *) p);
  textcolor(initial_info.attribute);
  textbackground(initial_info.attribute);


void buadrate_menu(char SPEED[5])

  int buadrate;
  int x = 20,
      y = 12,
    endx = 60,
    endy = 14;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textcolor(TEXTCOLOR);
  textbackground(BACKGROUND);

  /* save what is currently on the screen */
  save_video(x, endx+1, y, endy+1, p);

  draw_border(x, y, endx, endy,1);
  gotoxy(x+2,y+1); cprintf("Buadrate :");
  buadrate = selectVER(BUADRATE,"9421",4,x+13,y+1,7);
```

```
switch (buadrate)
 case 0: strcpy(SPEED,"9600"); break;
case 1: strcpy(SPEED,"4800"); break;
case 2: strcpy(SPEED,"2400"); break;
case 3: strcpy(SPEED,"1200"); break;

initial();
restore_video(x, endx+1, y, endy+1, (char *) p);
textcolor(initial_info.attribute);
textbackground(initial_info.attribute);


void filename_menu(char fname[MAXstring])
int x = 26,
    y = 12,
  endx = 54,
  endy = 14;
 struct text_info initial_info;

 gettextinfo(&initial_info);
 textcolor(TEXTCOLOR);
 textbackground(BACKGROUND);

 /* save what is currently on the screen */
 save_video(x, endx+1, y, endy+1, p);

 draw_border(x, y, endx, endy,1);
 gotoxy(x+2,y+1); cprintf("File Name :");
 readchar(x+14,y+1,12,fname);

 restore_video(x, endx+1, y, endy+1, (char *) p);
 textcolor(initial_info.attribute);
 textbackground(initial_info.attribute);



/* input name device of room */
void input_name_device()
 char ch;
```

```
int i,loop,count=0,pr,pl,num,FOUND;
int x = 32,
 y = 2,
 endx = 49,
 endy = 24;
struct text_info initial_info;

gettextinfo(&initial_info);
for(;;)
  room_menu(show1[0]);
   if (strlen(show1[0]) > 5)
       beep();
        error_menu(8);

   else break;

function2(6);                    /* READ TABLE */
textcolor(TEXTCOLOR);
textbackground(BACKGROUND);

/* save what is currently on the screen */
save_video(x, endx+1, y, endy+1, p);

draw_border(x, y, endx, endy,2);
textcolor(BORDERCOLOR);
gotoxy( x, y+3);    putch('Ì');
gotoxy( endx, y+3); putch('¹');
draw_ch_line( x+1, y+3, endx-1, y+3, 'Í');
textcolor(TEXTCOLOR);
gotoxy(x+1,y+1);  cprintf(" Room    : %s",show1[0]);
gotoxy(x+1,y+2);  cprintf(" Address : %s",show2[0]);
for (i = y+4,loop=0; loop < MAXDEVICE; i++,loop++)

gotoxy( x+3, i);  cprintf("%s %s",device[loop],show1[loop+1]);

 for( i=0,num=2;i != -1;)

gotoxy(x+12+strlen(show2[i]),y+i+2);
count = strlen(show2[i]);
```

```
{
for(;;)
ch = getch();
if ((ch >= 'a' && ch <= 'f') || (ch >= 'A' && ch <= 'F') ||
 (ch >= '0' && ch <= '9') )
 if (strlen(show2[i]) < num)
    for (pl = strlen(show2[i]),pr = strlen(show2[i])+1;pr >= 0;pr--,pl--)
     if (pr > count) show2[i][pr] = show2[i][pl];
    else _show2[i][pr] = ch; break;

   show2[i][strlen(show2[i])] = '';
   gotoxy(x+12,y+i+2);
   cprintf("%s",show2[i]);
   count++;
   gotoxy(x+12+strlen(show2[i]),y+i+2);

  else_beep();
 else if (ch == '' && count > 0)
     count--;
    for (pl = count,pr = count+1;pl < strlen(show2[i]);pl++,pr++)
     show2[i][pl] = show2[i][pr];
    show2[i][strlen(show2[i])] = '';
    gotoxy(x+12+strlen(show2[i]),y+i+2);
    putch('');
    gotoxy(x+12,y+i+2);
    cprintf("%s",show2[i]);
    gotoxy(x+12+count,y+i+2);

  else if (ch == ESC)
    i = -1;
    break;

 else if (ch == '')
     ch = getch();
  /* LEFT */ if ((ch == 75) && (count-1 >= 0))
     count--;
     gotoxy(x+12+count,y+i+2);

  /* RIGHT */ else if ((ch == 77) && (count+1 <= num) && (count+1 <= strlen(show2[i])))
     count++;
```

```
    gotoxy(x+12+count,y+i+2);

/* DEL */ else if ((ch == 80) && (i <= MAXDEVICE))
      if (i == MAXDEVICE) i = 0;
    break;

   else beep();

 else if (ch == '')  i = -1; break;
 else beep();


 for( i=1,num=5; i <= MAXDEVICE; i++)

gotoxy(x+9+strlen(show1[i]),y+i+3);
count = strlen(show1[i]);
 for(;;)
 ch = getch();
 if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') ||
  (ch >= '0' && ch <= '9') || ch == '.' || ch == '_' ||
   ch == '&' || ch == '#' || ch == '@' || ch == '!' ||
   ch == '(' || ch == ')' || ch == '^' || ch == '%' ||
   ch == '-')
  if (strlen(show1[i]) < num)
     for (pl = strlen(show1[i]),pr = strlen(show1[i])+1;pr >= 0;pr--,pl--)
      if (pr > count) show1[i][pr] = show1[i][pl];
     else _show1[i][pr] = ch; break;

    show1[i][strlen(show1[i])] = '';
    gotoxy(x+9,y+i+3);
    cprintf("%s",show1[i]);
    count++;
    gotoxy(x+9+strlen(show1[i]),y+i+3);

   else_beep();
 else if (ch == '' && count > 0)
     count--;
    for (pl = count,pr = count+1;pl < strlen(show1[i]);pl++,pr++)
     show1[i][pl] = show1[i][pr];
```

```
show1[i][strlen(show1[i])] = '';
gotoxy(x+9+strlen(show1[i]),y+i+3);
putch('');
gotoxy(x+9,y+i+3);
cprintf("%s",show1[i]);
gotoxy(x+9+count,y+i+3);

else if (ch == ESC)
 i = MAXDEVICE;
 break;

else if (ch == '')
  ch = getch();
 if ((ch == 75) && (count-1 >= 0))
   count--;
  gotoxy(x+9+count,y+i+3);

  else if ((ch == 77) && (count+1 <= num) && (count+1 <= strlen(show1[i])))
   count++;
   gotoxy(x+9+count,y+i+3);

  else if (ch == 83 && count < strlen(show1[i]))
   for (pl = count,pr = count+1;pl < strlen(show1[i]);pl++,pr++)
   show1[i][pl] = show1[i][pr];
            show1[i][strlen(show1[i])] = '';
   gotoxy(x+9+strlen(show1[i]),y+i+3);
   putch('');
   gotoxy(x+9,y+i+3);
   cprintf("%s",show1[i]);
   gotoxy(x+9+count,y+i+3);

  else if ((ch == 72) && (i >= 1))
    if (i == 1)  i = MAXDEVICE; i--;
   else i = i-2;
   break;

  else if ((ch == 80) && (i <= MAXDEVICE))
    if (i == MAXDEVICE) i = 0;
   break;
```

```
    else beep();

  else if (ch == '') break;
  else beep();


  function2(5);                                  /* EDIT */
  restore_video(x, endx+1, y, endy+1, (char *) p);
  textcolor(initial_info.attribute);
  textbackground(initial_info.attribute);


void show_state()
  int i;
   int x = 64,
    y = 2,
    endx = MAXX,
    endy = MAXY-1;
  struct text_info initial_info;

    gettextinfo(&initial_info);
    textcolor(BORDERCOLOR);
    textbackground(BACKGROUND);
    draw_border(x, y, endx, endy,2);
    gotoxy( x, y+3);    cputs("Ì");
    gotoxy( endx, y+3); cputs("¹");
    draw_ch_line( x+1, y+3, endx-1, y+3, 'Í');
    textcolor(TEXTCOLOR);
    gotoxy(x+1,y+1);  cprintf(" Room No: %s",show1[0]);
    gotoxy(x+1,y+2);  cprintf(" Address: %s",show2[0]);
    for ( i=1; i<=MAXDEVICE; i++)
     gotoxy( x+3,y+3+i);  cprintf("     ");
       gotoxy( x+3,y+3+i);  cprintf("%s ",show1[i]);
     gotoxy( x+9,y+3+i);  cprintf(":");
     gotoxy( x+11,y+3+i); cprintf("   ");
            gotoxy( x+11,y+3+i); cprintf("%s ",show2[i]);


     textcolor(initial_info.attribute);
```

```
void control_status()
 char ch,room1[maxlet],*pt;
  int x = 32,
   y = 6,
   endx = 49,
   endy = 20,
   i=0;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textcolor(BORDERCOLOR);
  textbackground(BACKGROUND);

  /* save what is currently on the screen */
  save_video(x, endx+1, y, endy+1, p);

  draw_border(x, y, endx, endy,2);
  gotoxy( x, y+3);   cputs("Ã");
  gotoxy( endx, y+3); cputs("´");
  draw_ch_line( x+1, y+3, endx-1, y+3, 'Ä');
  textcolor(TEXTCOLOR);
  if (strstr(roomNo,"-")!=NULL)

    pt = roomNo;
    while (*pt != '-')

      room1[i] = *pt;
      i++;
      room1[i] = NULL;
      pt++;

     strcpy(show1[0],room1);

    else
     strcpy(show1[0],roomNo);
    function2(6);              /* READ TABLE */
    if ((strcmp(show1[0],"all") == 0) || (strcmp(show1[0],"ALL") == 0))
```

```
for(i = 1;i <= 10;i++)
sprintf(show1[i],"P%d",i);


gotoxy(x+1,y+1);  cprintf(" Room No.: %s",show1[0]);
gotoxy(x+1,y+2);  cprintf(" Address : %s",show2[0]);
for ( i=1; i<=10; i++)
 gotoxy( x+3,y+3+i);  cprintf("%s",show1[i]);
gotoxy( x+9,y+3+i);  cprintf(":");
gotoxy( x+11,y+3+i);  cprintf("%s",show2[i]);


for ( i=1; i<=10; i++)
    gotoxy(x+11,y+3+i);
   ch = getch();
if (ch == ESC)
  i = 10; break;
 else if (ch == '')
  gotoxy(x+11,y+3+i);
  if (!strcmp(show2[i],"ON")) strcpy(show2[i],"OFF");
   else strcpy(show2[i],"ON");
   cprintf("%s ",show2[i]);
  gotoxy(x+11,y+3+i);

 else if (ch == '')
  ch = getch();
  if ((ch == 72) && (i >= 1))
     if (i == 1) i = 10-1;
    else i = i-2;

   else if ((ch == 80) && (i <= 10))
    if (i == 10) i = 0;
   else  i--; beep();

   else  i--; beep();

   restore_video(x, endx+1, y, endy+1, (char *) p);
   textcolor(initial_info.attribute);
   textbackground(initial_info.attribute);
   if (strstr(roomNo,"-") != NULL)
```

```
    function2(10);        /* WORK GROUP */
     communicate();

   else
   if ((strcmp(show1[0],"ALL") == 0) || (strcmp(show1[0],"all") == 0))
     function2(8);___/* ALL WORK */
   communicate();

   else
     function2(1);        /* OUT PORT */
   communicate();
   if(Error == 5) error_menu(5);
   else  function1(0);         /* DETECT */




void readchar(int x, int y, int num, char data[MAXstring])
 char ch;
 int count = 0,pr,pl;
 struct text_info initial_info;

 gettextinfo(&initial_info);
 textcolor(TEXTCOLOR);
 count = strlen(data);
 gotoxy(x,y);
 cprintf("%s",data);
 gotoxy(x+count,y);
 for(;;)
  ch = getch();
  if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') ||
   (ch >= '0' && ch <= '9') || ch == '.' || ch == '_' ||
    ch == '&' || ch == '#' || ch == '@' || ch == '!' ||
    ch == '(' || ch == ')' || ch == '^' || ch == '%' ||
                ch == ':' || ch == '-')
   if ((count < num) && (strlen(data) < num))
     for (pl = strlen(data),pr = strlen(data)+1;pr >= 0;pr--,pl--)
       if (pr > count) data[pr] = data[pl];
      else _data[pr] = ch; break;
```

```
data[strlen(data)] = '';
        gotoxy(x,y);
cprintf("%s",data);
count++;
        gotoxy(x+count,y);

else beep();
else if (ch == '' && count > 0)
  count--;
for (pl = count,pr = count+1;pl < strlen(data);pl++,pr++)
 data[pl] = data[pr];
 data[strlen(data)] = '';
 gotoxy(x+strlen(data),y);
 putch('');
 gotoxy(x,y);
 cprintf("%s",data);
 gotoxy(x+count,y);

else if (ch == '')
  ch = getch();
 if ((ch == 75) && (count-1 >= 0))
   count--;
  gotoxy(x+count,y);

 else if ((ch == 77) && (count+1 <= num) && (count+1 <= strlen(data)))
  count++;
  gotoxy(x+count,y);

 else if (ch == 83 && count < strlen(data))
  for (pl = count,pr = count+1;pl < strlen(data);pl++,pr++)
  data[pl] = data[pr];
        data[strlen(data)] = '';
  gotoxy(x+strlen(data),y);
  putch('');
  gotoxy(x,y);
  cprintf("%s",data);
  gotoxy(x+count,y);
```

```
    else beep();

    else if (ch == '') break;
    else if (ch == ESC) break;
    else beep();

  textcolor(initial_info.attribute);


void beep(void)
  sound(400);
  delay(100);
  nosound();


void beep2(void)
  sound(800);
  delay(200);
  nosound();



void error_menu(int error)
  int x = 17,
      y = 5,
      endx = 63,
      endy = 22;
  char ch;
  struct text_info initial_info;

  gettextinfo(&initial_info);
  textbackground(RED);
  textcolor(TEXTCOLOR);
  if(ERRORselect == 0)
    textcolor(BORDERCOLOR);
  textbackground(BACKGROUND);
  window(x,y,endx,endy);
  clrscr();
  window(1,1,80,25);
```

```
draw_border(x, y, endx, endy,2);
gotoxy( x, y+2);    cputs("ท");
gotoxy( endx, y+2); cputs("น");
draw_ch_line( x+1, y+2, endx-1, y+2, 'อ');
textcolor(TEXTCOLOR);
gotoxy(x+1,y+1);  cprintf("          ERROR                ROOM");
lineerror = y+3;

ERRORselect=1;
if (error <= 4)
  textbackground(RED);
textcolor(WHITE + BLINK);
beep2();

 if (error > 4)
   textbackground(BACKGROUND);
 textcolor(WHITE);
 beep();

 if (lineerror > endy-1)
  lineerror = y+3;
 gotoxy(x+2,lineerror);
  cprintf("                                  ");
  gotoxy(x+2,lineerror);  cprintf("%s",ERROR[error]);
  gotoxy(x+35,lineerror);  cprintf("%s",show1[0]);
  lineerror++;

  textcolor(initial_info.attribute);
  textbackground(initial_info.attribute);
}

void menu(void)

{
 int i,choice,systemselect,controlselect,quitselect,wide=15,oldchoice;
 int FOUND;
 char name[20],ch;

 cursor_off();
 choice = selectVER(mainmenu,"scq",3,5,2,wide);
```

```
for (;choice != -1;)
  switch (choice)
   case 0: oldchoice = choice;
    write_video(5+(choice*wide), 2, mainmenu[choice], REV_VID);
    systemselect = popup(system_, "sbdeluc", 7, 1, 3, BORDER);
    switch (systemselect)
      case -1:choice = -1;
       break;
     case 0: seriel_port(Comport);
           choice = -1;
       break;
     case 1: buadrate_menu(Speed);
       choice = -1;
       break;
     case 2: for(FOUND = 0;FOUND != 1;)
        room_menu(roomNo);
        if (strlen(roomNo) > 11)
         beep();
          error_menu(8);

        for (i=0;i<indexData;i++)
         if (strstr(roomNo,room[i]) != NULL)
           FOUND=1;
        if (FOUND == 0)
          beep();
           error_menu(8);

       function2(4);  /* DELETE */
       choice = -1;
       break;
     case 3: input_name_device(); /* edit */
       textbackground(BACKGROUND);
       window(1,3,63,MAXY-1);
       clrscr();
       window(1,1,80,25);
       ERRORselect = 0;
       drawbuilding(indexData);
       choice = -1;
```

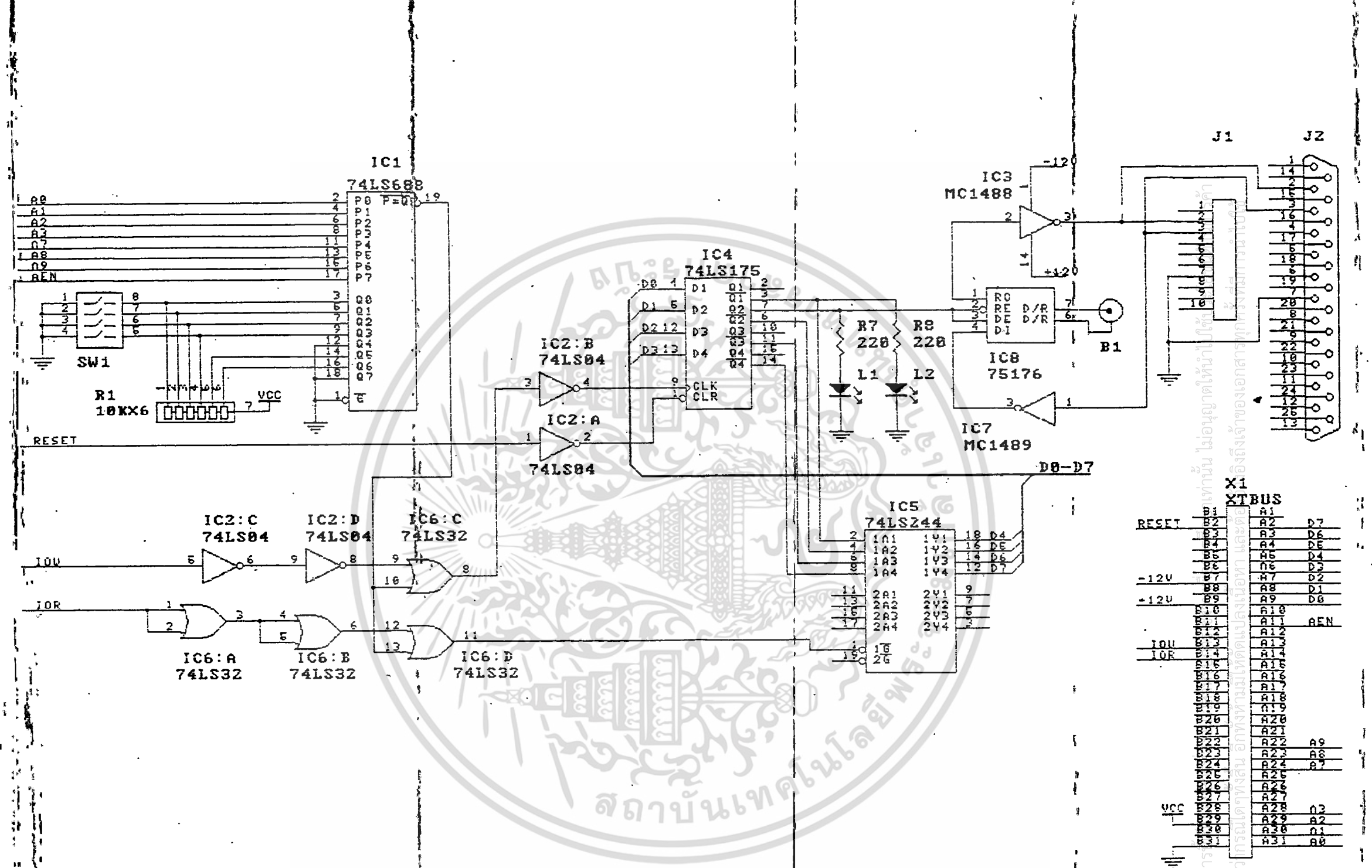
```
    break;
  case 4: for (;;)
     filename_menu(fname);
     if (strlen(fname) <= 12)
       function2(3);  /* OPEN FILE */
     textbackground(BACKGROUND);
     window(1,3,63,MAXY-1);
     clrscr();
     window(1,1,80,25);
     ERRORselect = 0;
     drawbuilding(indexData);
     choice = -1;
     break;


    break;
  case 5: for (;;)
     filename_menu(fname);
     if (strlen(fname) <= 12)
       function2(2);  /* UPDATE FILE */
     choice = -1;
     break;


                    case 6: if (ERRORselect == 1)
   ERRORselect = 0;
 textbackground(BACKGROUND);
  window(1,3,63,MAXY-1);
 clrscr();
 window(1,1,80,25);
            drawbuilding(indexData);


                              choice = -1;
                              break;
    default:if (systemselect == -2) choice = 2;
      if (systemselect == -3) choice = 1;


    write_video(5+(oldchoice*wide), 2, mainmenu[oldchoice], NORM_VID);
    break;
```

```
case 2: for(FOUND = 0;FOUND != 1;)
    room_menu(roomNo);
    if (strlen(roomNo) > 11)
     beep();
     error_menu(8);

    for (i=0;i<indexData;i++)
     if (strstr(roomNo,room[i]) != NULL)
      FOUND=1;
    if (FOUND == 0)
     beep();
      error_menu(8);


    function2(7);       /* CLEAR DEVICE */
    communicate();
    choice = -1;
    break;
  default:if (controlselect == -2) choice = 0;
    if (controlselect == -3) choice = 2;

 write_video(5+(oldchoice*wide), 2, mainmenu[oldchoice], NOP.M_VID);
 break;
case 2: oldchoice = choice;
 write_video(5+(choice*wide), 2, mainmenu[choice], REV_VID);
 quitselect = popup(quit, "yn", 2, 34, 3, BORDER);
 switch (quitselect)
   case -1:choice = -1;
    break;
  case 0: cursor_on();
    textcolor(WHITE);
    textbackground(BLACK);
    clrscr();
    QUIT = 1;
    return;
  case 1: choice = -1; break;
  default:if (quitselect == -2) choice = 1;
        if (quitselect == -3) choice = 0;
```

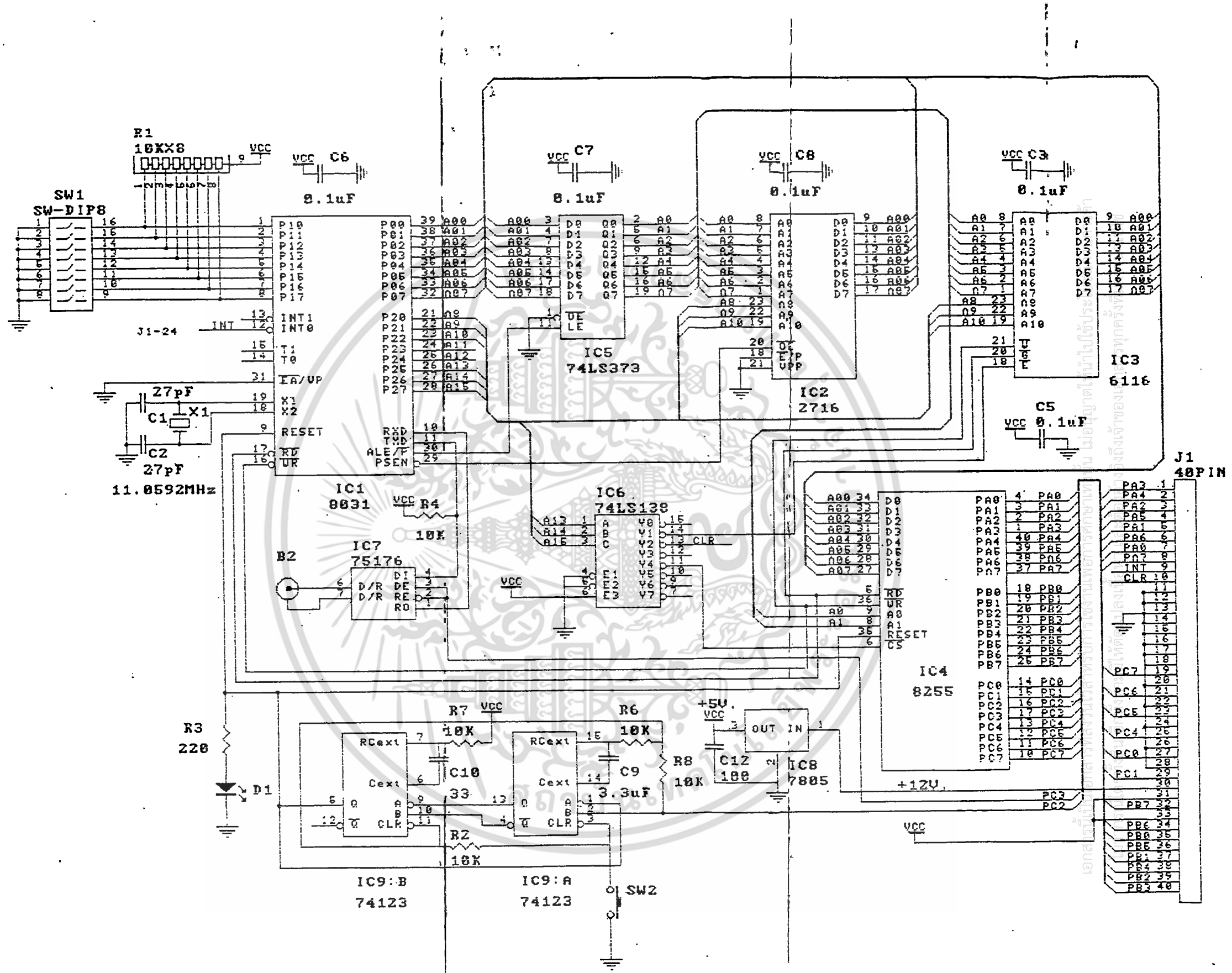
```
        write_video(5+(oldchoice*wide), 2, mainmenu[oldchoice], NORM_VID);
        break;



main()

 int i;
 chkcard();
  if(Cardport == 0)
     clrscr();
      printf("RS-485 Interface Not Install  Program may be HANG OVER !");
      getch();

 initialscreen();
 chkcard();
 openfile();
 initial();
 oldhandler = getvect(INTR);
 setvect(INTR, handler);
 drawbuilding(indexData);

 while(1)
    Error = 0;
  if(kbhit())
    menu();
    if(QUIT) break;
    function2(0);  /*sequence*/
    communicate();
    if(Error==5)
   function1(1);
  else function1(0);

    if(currentIndex %4 == 0)
    dataOut[0] = dataOut[6] = 0x7e;
     dataOut[2] = 0X4C;
     createSum();
     communicate();
```

```
case 1: oldchoice = choice;
  write_video(5+(choice*wide), 2, mainmenu[choice], REV_VID);
  controlselect = popup(control, "src", 3, 16, 3, BORDER);
  switch (controlselect)
    case -1:choice = -1;
     break;
 /* Restart */__  case 0:
     _room_menu(show1[0]);
   if (strlen(show1[0]) > 5)
    beep();
     error_menu(8);

                                function2(9);   /* Rstart */
                                choice = -1;
   break;

                          case 1: FOUND = 0;
      room_menu(roomNo);
      if (strlen(roomNo) > 11)
       beep();
        error_menu(8);

      else
       for (i=0;i<indexData;i++)
      if ((strstr(roomNo,room[i]) != NULL) ||
       (strcmp(roomNo,"all") == 0) ||
       (strcmp(roomNo,"ALL") == 0))
        FOUND=1;
       break;

        if (FOUND == 0)
          strcpy(show1[0],roomNo);
           error_menu(8);

        if (FOUND == 1)
       control_status();

       choice = -1;
       break;
```

```
}
 if(Error==5)  Error = 0;
 else

 function1(0);


/*  while(1) */
setvect(INTR, oldhandler);
return 0;
```



RS-485 Interface Circuit



8031 Microcontroller circuit

กิตติกรรมประกาศ

โครงงานระบบควบคุมภายในอาคาร ต้องจะไม่สามารถสำเร็จลุล่วงตามจุดประสงค์ที่ต้องการได้ หากขาดผู้สนับสนุนในด้านต่างๆ ทางคณะผู้จัดทำโครงงานนี้ จึงขอขอบคุณผู้สนับสนุน โครงงานนี้ จึงขอขอบคุณผู้สนับสนุนโครงงานนี้ คือ ผศ. วิทยา ทิพย์สุวรรณพร คุณเทิดทูล ภู่หลง คุณ นพเก้า อ่วมกระทุ่ม การสื่อสารแห่งประเทศไทย ที่เอื้อเฟื้ออุปกรณ์ที่จำเป็น เช่น PRINTER PLOTTER และบุคคลากรที่มีความสามารถมาแนะนำและช่วยเหลือตลอดการปฏิบัติงาน อีกทั้งยังให้คำแนะนำและปรึกษาเกี่ยวกับปัญหาต่างๆ ที่เกิดขึ้นตลอดช่วงของการดำเนินงานจัดสร้างโครงงานนี้ และขอขอบคุณท่าน บิดา มารดา ที่ให้กำเนิดเกิดมาและให้การสนับสนุนทุกสิ่งทุกอย่าง จนโครงงานนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ตามจุดประสงค์ที่ตั้งไว้ทุกประการ

เอกสารอ้างอิง

1. ยืน ภู่วรวรรณ. การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวร์ค.
พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : ซีเอ็ดยูเคชั่น , 2532

2. สมพันธุ์ ชาญศิลป์. เอกสารประกอบการสอนวิชาภาษา C.
พิมพ์ครั้งที่ 1. ขอนแก่น : หน่วยงานสารบรรณ งานบริหารและธุรการ
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น , 2534

3. ยืน ภู่วรวรรณ. เทคโนโลยีฮาร์ดแวร์ IBM PC.
พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : ซีเอ็ดยูเคชั่น , 2533

4. ชูชัย ธนสารรั้งเจริญ และ ทินกร ดุ๊ก. การใช้งาน Z 80.
พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : ฟิสิกส์เซ็นเตอร์

5. ศิวัฒน์ ศิวะบวร และ พรชัย จักรธำรงค์. การประยุกต์ใช้งานภาษา C.
พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : ซีเอ็ดยูเคชั่น , 2535

6. คู่มือไมโครโปรเซสเซอร์ไมโครคอนโทรลเลอร์.
พิมพ์ครั้งที่ 1. บรษัท อีทีที จำกัด.กรุงเทพมหานคร , 2534

7. KENNETH J.AYALA. THE 8051 MICROCONTROLLER ARCHITECTURE,
PROGRAMMING,AND APPLICATION. WEST PBLISHING COMPANY,
U.S.A. , 1991