การประยุกต์ใช้งานระบบไอเอสดีเอ็นในงานควบคุมการให้บริการคอมพิวเตอร์

THE ISDN APPLICATION FOR CONTROLLING COMPUTER SERVICE SYSTEM

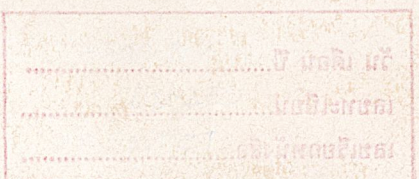
โดย

นาย ศักดิ์สิทธิ์ สุขาภิบาล	34107370
นาย สมพงษ์ สาระโภค	34107395
นาย สุทธิ จรรยาสุทธิวงศ์	34108428

อาจารย์ที่ปรึกษา

.....

(อ.ขนิษฐา แซ่ตั้ง)



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเลคทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาอิเลคทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งานระบบไอเอสดีเอ็นในงานควบคุมการให้บริการคอมพิวเตอร์

ผู้จัดทำ

นาย ศักดิ์สิทธิ์ สุขาภิบาล	34107370
นาย สมพงษ์ สาระโภค	34107395
นาย สุทธิ จรรยาสุทธิวงศ์	34108428

.....อาจารย์ที่ปรึกษา

(อ.ขนิษฐา แซ่ตั้ง)

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการ	5
บทที่ 3 การสร้างระบบให้บริการคอมพิวเตอร์	34
บทที่ 4 การทดลองและผลการทดลอง	69
บทที่ 5 บทสรุปและวิจารณ์	75
ภาคผนวก	
บรรณานุกรม	
กิตติกรรมประกาศ	

The ISDN application for controlling computer service system

Mr.Saksit Sukapibarn 34107370

Mr.Somphong Sarapoke 34107395

Mr.Sutthi Janyasutthiwong 34108428

Miss.Kanittha Sae-tang Advisor

1994

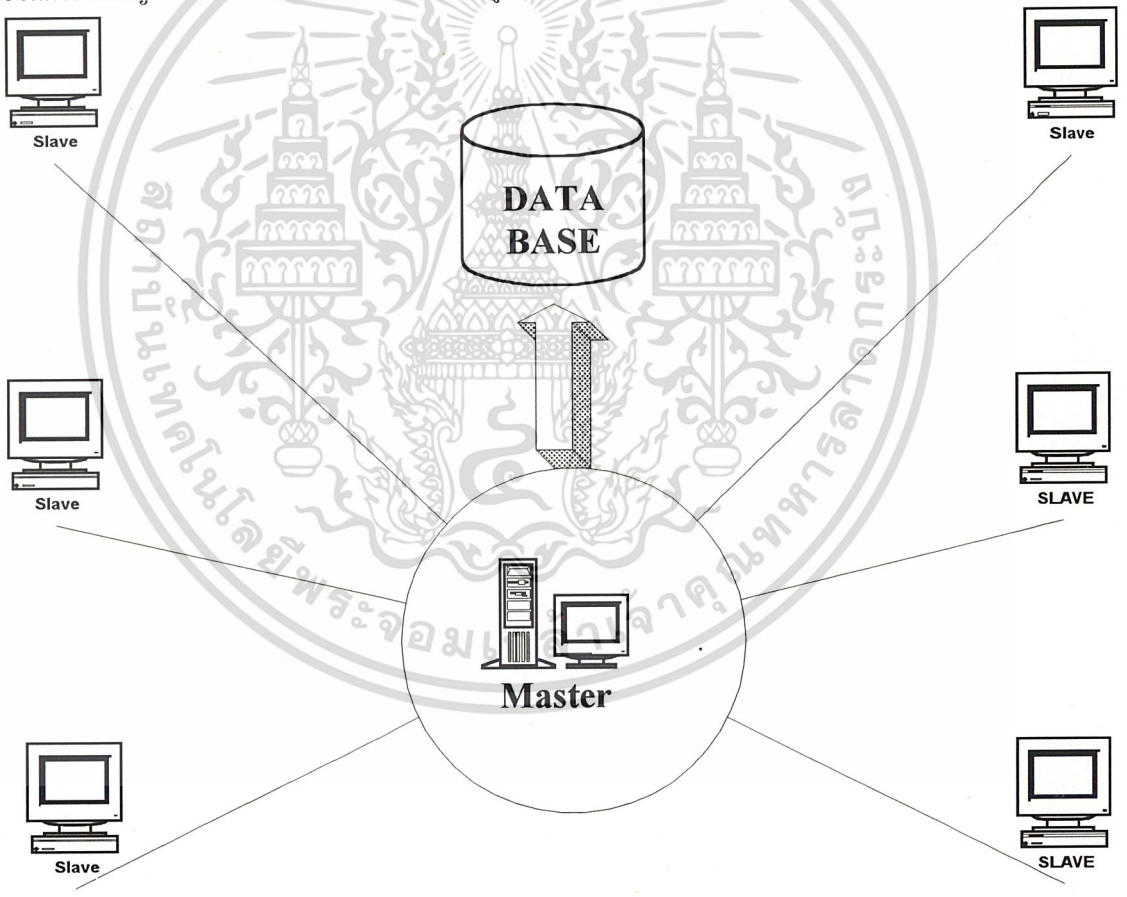
Abstract

The ISDN application for controlling computer service system consists of barcode decoder,database and ISDN communication device.Student can use computer by swifting a student-card ,with has student identity barcode, throughs a barcode reader.Output signal from barcode reader is converted to ascii code and transmitted via the ISDN device to central processing unit,a personal computer which has database.The central processing unit checks the student identity as it receives.It is responsible of student to use a computer .If he is permitted, the power line will connect to computer so that the computer is enable.

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันมีการนำระบบคอมพิวเตอร์มาใช้ในการควบคุมการบริการต่างๆมากมาย ซึ่งเอื้ออำนวยประโยชน์มากมายทั้งในด้านค่าใช้จ่ายในระยะยาวและความสะดวกต่างๆ คณะผู้จัดทำได้เล็งเห็นว่าการให้บริการคอมพิวเตอร์ของสำนักวิจัยและบริการคอมพิวเตอร์ของสถาบันเทคโนโลยีพระจอมเกล้าลาดกระบังนั้นประสบปัญหาจุกจิกมากมายในการให้บริการ อาทิเช่น การเปิดใช้หลายๆ เครื่องโดยผู้ใช้บริการคนเดียวทำให้ผู้ใช้บริการรายอื่นเสียโอกาส, บุคคลผู้ไม่มีสิทธิ์ใช้เครื่องมาแอบเปิดใช้, การเปิดเครื่องทิ้งไว้หลังการใช้งาน ความไม่พอใจของนักศึกษาที่มีต่อการให้บริการคอมพิวเตอร์ ฯลฯ คณะผู้จัดทำจึงได้มีความคิดที่จะประยุกต์ระบบการสื่อสารข้อมูลแบบ ISDN ผนวกเข้ากับระบบบาร์โค้ด(barcode) ของนักศึกษาที่มีอยู่แล้วและโปรแกรมฐานข้อมูลมาใช้เพื่อจัดการกับปัญหาดังกล่าว ระบบโดยรวมแสดงดังรูป



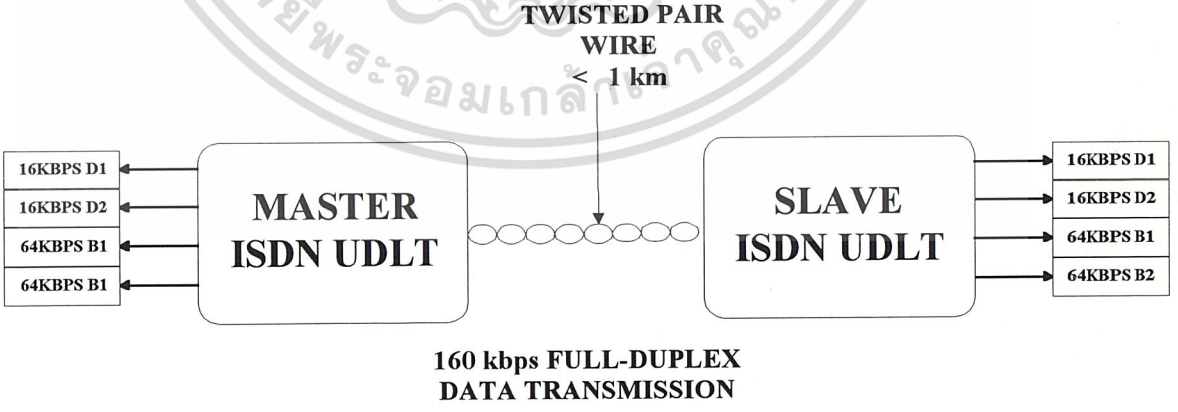
รูปที่ 1.1 แสดงโครงสร้างของโครงงาน

1.2 หลักการทำงาน

โครงงานนี้จะประกอบด้วยสามส่วนหลัก คือ โปรแกรมฐานข้อมูลในการประมวลผล, ระบบการสื่อสารแบบ ISDN และ ระบบบาร์โค้ด ซึ่งจะแจกแจงรายละเอียดในแต่ละส่วนในภายหลัง ซึ่งในบทนี้จะแจกแจงแต่ละส่วนโดยสรุป ดังนี้

1.2.1 โปรแกรมฐานข้อมูล ในส่วนนี้จะมีหน้าที่ในการจัดการระบบการให้บริการคอมพิวเตอร์ จะทำหน้าที่พิจารณาว่าผู้ใช้บริการที่เข้ามาขอใช้บริการนั้นมีสิทธิ์ในการใช้หรือเปล่า โดยรับข้อมูลจากระบบบาร์โค้ดมาแล้วทำการประมวลผลวิเคราะห์ว่าผู้ใช้บริการ ที่เข้ามาขอใช้นั้นสามารถใช้เครื่องได้หรือไม่ เมื่อทำการประมวลผลแล้วจะส่งข้อมูลที่ได้นั้นผ่านระบบการสื่อสารแบบ ISDN ไปยังหน่วยย่อยต่าง ๆ เพื่อทำการควบคุมการให้อนุญาตเปิดเครื่องคอมพิวเตอร์ต่อไป ดังนั้นโปรแกรมดังกล่าวจึงต้องมีฐานข้อมูลที่เข้ามาจัดการข้อมูลของผู้ใช้บริการแต่ละคน ซึ่งนอกจากมันจะทำงานโดยอัตโนมัติแล้ว ยังสามารถให้เจ้าหน้าที่ควบคุมด้วยตนเองได้อีกด้วย และสามารถทำบันทึกสถิติของการให้บริการและจัดพิมพ์รายงานผลการให้บริการออกทางเครื่องพิมพ์ได้อีกด้วย ในการต่อร่วม(interface) นั้นในส่วนนี้จะส่งข้อมูลผ่านพอร์ทอนุกรม(serial port)ไปยังมาสเตอร์ของส่วนการสื่อสารแบบ ISDN

1.2.2 ระบบการสื่อสารข้อมูล ในส่วนนี้จะเป็นการสื่อสารข้อมูลแบบ ISDN(Integrated Service Digital Network)โดยรับข้อมูลแบบอะซิงโครนัส(asynchronous)จากพอร์ทของคอมพิวเตอร์ใช้ไมโครโปรเซสเซอร์(microprocessor) Z80 ทำการแปลงให้เป็นข้อมูลแบบซิงโครนัส(synchronous)เพื่อที่จะส่งไปยัง IC เบอร์ MC145421 ซึ่งเป็น ISDN Universal Digital Loop Tranceiver II (UDLTII) โดย IC ตัวนี้เป็นตัวมาสเตอร์(master)จะส่งต่อไปยัง IC สลาฟ(slave)คือเบอร์ MC145425 ซึ่งมีความสามารถในการส่งข้อมูลได้ 160 กิโลบิทต่อวินาที(kbps) ฟูลดูเพล็กซ์ (full duplex data) โดยทำการส่งผ่านสายคู่ตีเกลียว(Twisted -pair cable) เบอร์ 26 awg ดังรูป



รูปที่ 1.2 แสดงโครงสร้างการส่งข้อมูลแบบ ISDN ที่จะใช้ในโครงงานนี้

หลังจากไอซีสลาฟได้รับข้อมูลจากมาสเตอ/ร์ก็จะนำไปประมวลผลข้อมูลที่ได้โดยไมโครคอนโทลเลอร์ (microcontroller)เบอร์ MCS8749 ในทางกลับกันวงจรส่วนที่เป็นสลาฟก็สามารถรับข้อมูลรหัสผู้ใช้บริการจากส่วนของบาร์โค้ดเพื่อส่งไปยัง MCS8749 ทำการส่งข้อมูลไปให้ไอซีสลาฟส่งข้อมูลขนาด 64 กิโลบิทต่อวินาที ผ่านสายคู่ตีเกลียวเบอร์ 26 awg ไปยังวงจรส่วนของมาสเตอร์ ไอซีมาสเตอร์จะทำการดีมอดดูเลต(demodulate)ข้อมูลออกมาให้เป็นข้อมูลทางดิจิตอล(digital)ให้ Z80 ทำการแปลงสัญญาณจากซิงโครนัสให้เป็นสัญญาณอะซิงโครนัส แล้วจึงส่งสัญญาณดังกล่าวไปยังพอร์ทอนุกรมของไมโครคอมพิวเตอร์ต่อไป

1.2.3 ระบบบาร์โค้ด ในส่วนนี้จะมีหน้าที่รับข้อมูลของนักศึกษาหรือผู้ใช้บริการที่อยู่ในรูปของบาร์โค้ด โดยจะทำาการแปลงรหัสแถบ 39(ซึ่งคณะใช้งานอยู่) ให้เป็นข้อมูลส่งไปให้ส่วนของวงจรสลาฟอีกที

1.3. ประโยชน์ที่คาดว่าจะได้รับ

3.1 ช่วยปรับปรุงระบบการให้บริการของสำนักวิจัยและบริการคอมพิวเตอร์

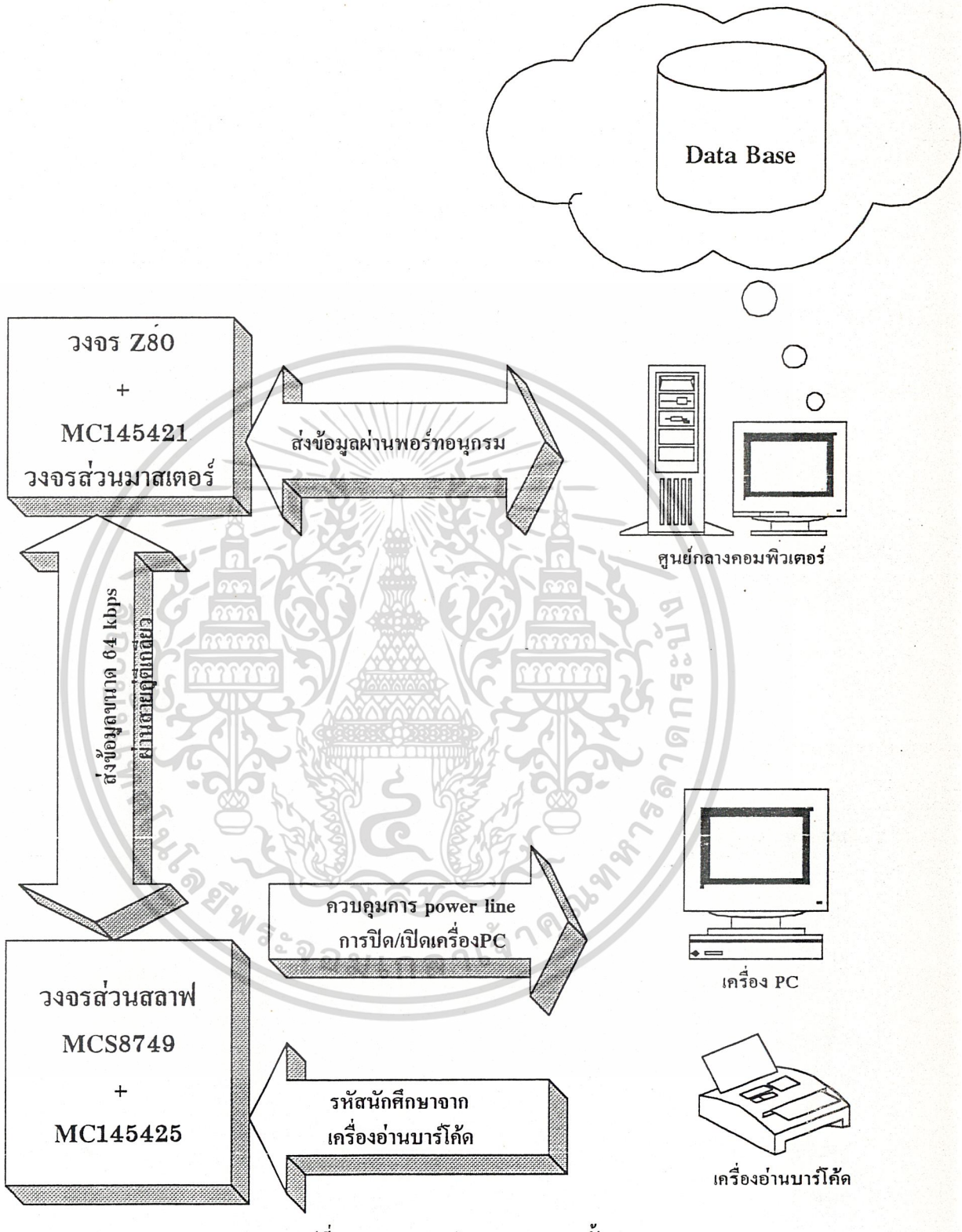
3.2 สามารถนำข้อมูลที่ได้จากสถิติการใช้งานเครื่องคอมพิวเตอร์ของนักศึกษาสถาบันเทคโนโลยีพระจอมเกล้าลาดกระบัง มาเป็นข้อมูลเพื่อใช้ในการวางแผนการขยายงานด้านบริการเครื่องคอมพิวเตอร์ให้สอดคล้องกับปริมาณการใช้บริการของนักศึกษา

3.3 ช่วยเพิ่มโอกาสในการใช้เครื่องคอมพิวเตอร์ของนักศึกษาให้สามารถบริการได้ทั่วถึงกันได้ เพราะระบบจะจัดการให้ใช้หนึ่งบัตรต่อหนึ่งเครื่อง ทำให้ไม่มีการใช้คนเดียวเปิดหลายเครื่อง

3.4 เป็นระบบรักษาความปลอดภัยทางอ้อม กล่าวคือ จะให้บริการกับนักศึกษารุ่นปัจจุบันเท่านั้น

3.5 ช่วยลดจำนวนของเจ้าหน้าที่ที่จะต้องมาคอยควบคุมการให้บริการคอมพิวเตอร์ อีกทั้งยังเพิ่มความสะดวกสบายแก่เจ้าหน้าที่อีกด้วยทางหนึ่ง

3.6 ช่วยส่งเสริมให้นักศึกษาที่จะทำโครงงานเกี่ยวกับการสื่อสารข้อมูลแบบ ISDN ให้มีความเข้าใจมากยิ่งขึ้นเนื่องจากระบบดังกล่าวสามารถส่งข้อมูลได้มาก อีกทั้งยังสามารถส่งสัญญาณทางดิจิตอล, สัญญาณเสียงและสัญญาณภาพได้ในเวลาเดียวกัน และกำลังมีใช้ในประเทศไทยได้ไม่นาน



รูปที่ 1.3 แสดงการทำงานของระบบทั้งหมด

บทที่ 2

ทฤษฎีและหลักการ

2.1 ระบบการสื่อสารข้อมูล(DATA COMMUNCATION SYSTEM)

การสื่อสารข้อมูลเกี่ยวข้องกับการส่งรหัสเลขฐานสอง (Binary code) ซึ่งเป็นรหัสที่สร้างและดำเนินการโดยคอมพิวเตอร์ การติดต่อสื่อสารข้อมูลมีลักษณะเชิงดิจิตอลที่สามารถกำหนดสภาวะได้ 2 สถานะ คือ ค่าตรรก (logic)เท่ากับ 1 หรือ 0 (ส่วนเชิงอนาลอกมีได้ไม่จำกัดสถานะ)

กำหนดให้การใช้ข้อมูลแทนข้อความ,กราฟฟิก(graphics) เป็นรหัสขนาด n บิทที่สามารถแทนจำนวนข้อมูลได้ 2^n ตัว สำหรับรหัสที่ใช้แทนตัวอักษร , ตัวเลข หรือสัญลักษณ์พิเศษ เรียกว่าตัวอักขระ (Alphanumeric)

ก่อนที่จะกล่าวถึงชนิดของรหัสที่มีใช้กัน ขออธิบายถึงสาเหตุในการที่ต้องสร้างหรือคิดค้นรหัสขึ้น ขอให้พิจารณาในประเด็นที่เป็นการสื่อสารระหว่างเครื่องจักรด้วยกัน พบว่าเครื่องจักร เช่น คอมพิวเตอร์ หรือโทรพิมพ์ ไม่มีความสามารถเข้าใจถึงความหมายของตัวหนังสือได้จึงต้องมีการแปลงความหมายให้เป็นแบบที่เครื่องจักรสามารถตีความได้ คือในสถานะเลขฐานสอง ดังนั้นจึงต้องมีอุปกรณ์ทำหน้าที่เข้ารหัส (Encoder) และถอดรหัส(Decoder)มาใช้ในการรับ/ส่ง ข้อมูลระหว่างเครื่องจักรด้วยกัน

ความหมาย บิท (Bit) ในการสื่อสารแล้ว คือ หน่วยที่เล็กที่สุดในระบบคอมพิวเตอร์ มีค่าข้อมูลเป็นตัวเลขระบบเลขฐานสอง คือ 0 หรือ 1 ย่อมาจาก **BI**nary digi**T**

2.1.1 วิธีการเข้ารหัสข้อมูล

สำหรับรหัสที่นิยมกันมากในการสื่อสารข้อมูลมีหลายแบบ ในที่นี้จะกล่าวถึง 3 ชนิด คือ

2.1.1.1 รหัสแบบบอโด (Baudot Code) เป็นรหัสมาตรฐานในการส่งโทรเลขประกอบด้วยบิทจำนวน 5 บิท ใช้แทนตัวเลขหรือตัวอักษร หรือเครื่องหมายพิเศษอย่างใดอย่างหนึ่ง

สามารถเข้ารหัสได้ทั้งหมด 32 ตัว (2^5) และส่งในแบบอนุกรมไม่สัมพันธ์ (seriai asychronus) และพัลซ์ที่สิ้นสุดหรือบิทจบ (stop bit) มักใช้ขนาด 1.5 บิท(ผิดกับแบบแอสกี้คีที่ใช้ขนาด 1-2 บิท)

2.1.1.2 รหัสเอบซีดิก (EBCDIC :Extend Binary Code Decimal Interchage Code) เป็นรหัสตัวอักษรแบบ 8 บิท ใช้กันมากในระบบสื่อสารของคอมพิวเตอร์ ตามมาตรฐาน IBM และมีบิทที่ 9 เพิ่มเติมในกำหนดพาริตี้บิท

2.1.1.3 รหัสแอสกี้ (ASCII) ซึ่งย่อมาจาก **American Standard Code Infor-mation** จัดเป็นรหัสแบบ 7 บิทพร้อมกับรวมบิทตรวจสอบอีก 1 บิท เป็นทั้งหมด 8 บิทดูได้จากตาราง 1.1 ที่แสดงบิทข้อมูล 7 บิทของแอสกี้ ส่วนมากไมโครคอมพิวเตอร์ จะใช้ 8 บิท ในการเข้ารหัสอักขระ

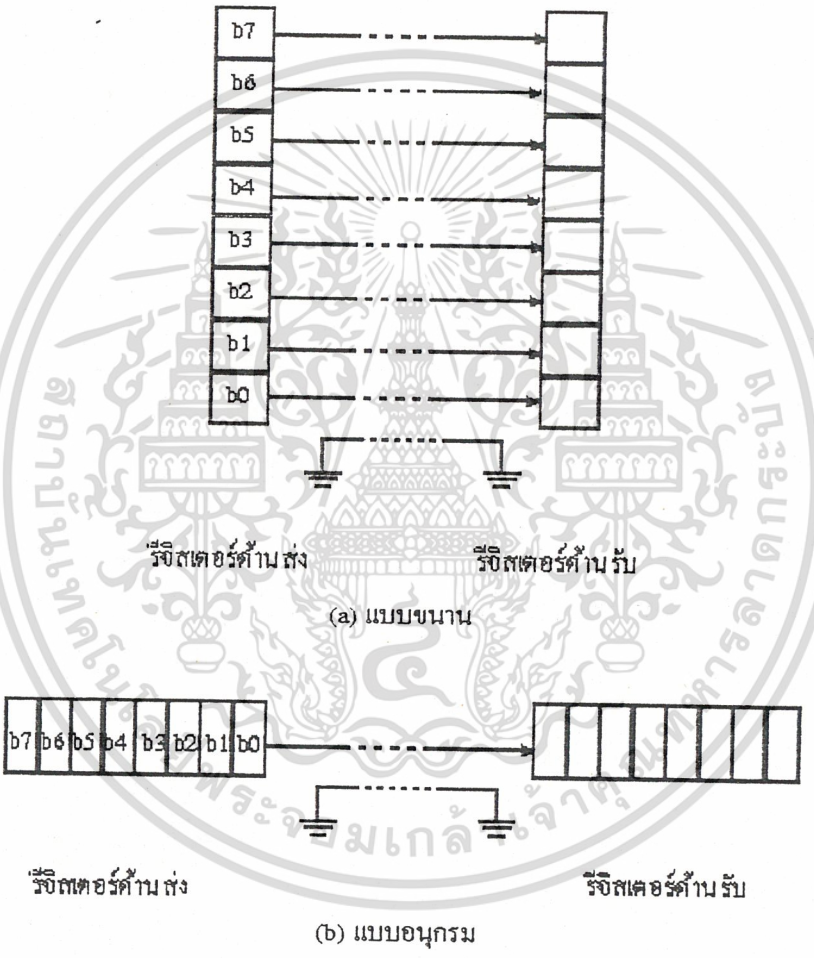
ใน 128 ตัวแรกของรหัส ASCII ได้รับการกำหนดมาตรฐานเอาไว้ใน ANSIX3.4 ปี ค.ศ. 1977 ส่วน 128 ตัวหลัง ซึ่งมีบิทที่ 8 เป็น 1 นั้น ไมโครคอมพิวเตอร์ส่วนมากใช้เป็นรหัสสำหรับสัญลักษณ์ทางกราฟฟิก

ซึ่งไมโครคอมพิวเตอร์ในแต่ละยี่ห้อจะไม่เหมือนกัน สำหรับคนไทยนำเอา 128 ตัวหลังมาเข้ารหัสอักขระทางภาษาไทย

2.1.2 การส่งข้อมูลแบบอนุกรม และ แบบขนาน (Serial and Parallel Transmission)

กรณีแบบขนาน ทุกบิทของข้อมูลถูกส่งแยกตามทางติดต่อที่แยกตามมาแต่ละบิทในเวลาเดียวกัน กรณีแบบอนุกรม แต่ละบิทของข้อมูลถูกส่งไปบนทางติดต่อ 1 ช่องและครั้งละหนึ่งบิท

พิจารณาจากรูปที่ 2.1

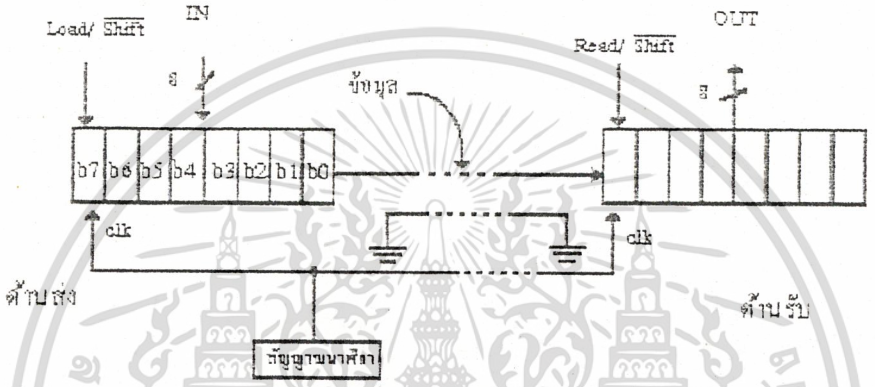


รูปที่ 2.1 แสดงการส่งข้อมูลแบบขนานและแบบอนุกรม

เปรียบเทียบทั้งสองวิธีพบว่าแบบขนานใช้ค่าใช้จ่ายสูงกว่าแบบอนุกรม เนื่องจากกรณีแบบขนานต้องใช้สายนำข้อมูลจำนวนมาก และยังต้องมีความเร็วในการส่งสูงกว่าด้วย เพราะทุกบิทส่งในเวลาเดียวกัน มีผลให้การใช้งานการส่งแบบขนาน เหมาะสำหรับกรณีด้านรับติดตั้งใกล้กับด้านส่ง เช่น ภายในบริเวณห้องเดียวกัน

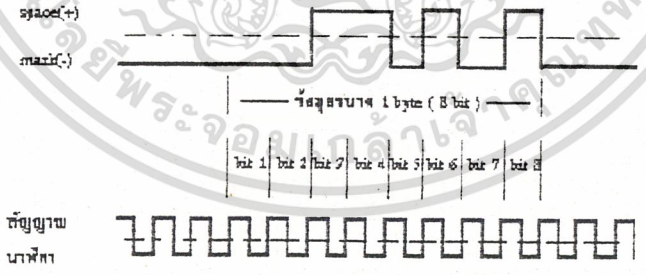
2.1.3 การส่งข้อมูลแบบซิงโครนัสและอะซิงโครนัส

2.1.3.1 การส่งข้อมูลแบบซิงโครนัส หมายถึง การที่ด้านรับอ่านข้อมูลเข้ามาในจังหวะเดียวกับด้านส่ง โดยใช้สัญญาณนาฬิกาเป็นตัวกำหนดจังหวะการทำงานของรีจิสเตอร์ทั้งสองให้ทำงานสัมพันธ์กัน (วงจรกำเนิดสัญญาณนาฬิกาจะติดตั้งภายในด้านส่ง นอกจากนี้เมื่อจังหวะเวลาถูกติดตั้งให้ซิ้งกับด้านรับได้แล้ว ข้อมูลจะถูกส่งไปบนทางติดต่อในแบบบิทต่อบิทต่อเนื่องกันไปอาศัยช่วงเวลาระหว่างบิทต่อบิทมีค่าเท่ากันโดยไม่ต้องมีบิทเริ่มส่งหรือบิทจบคอยกำกับ ทำให้ความเร็วในการส่งข้อมูลมีสูง พิจารณารูปที่ 2.2



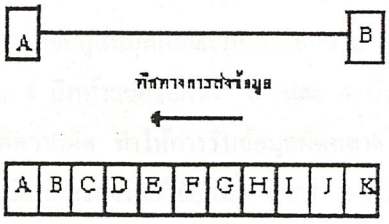
รูปที่ 2.2 แสดงการส่งข้อมูลแบบซิงโครนัส

ข้อเสียของการส่งแบบซิงโครนัส คือ การที่ต้องมีสัญญาณนาฬิกาขนานไปกับข้อมูล ทำให้ต้องการทางติดต่อช่องที่สองเพิ่ม โดยเฉพาะกรณีระยะทางไกล ๆ เป็นการยากมากที่จัดหาทางติดต่อแยกต่างหากสำหรับสัญญาณนาฬิกา อาจจะกล่าวได้ว่า การส่งแบบซิงโครนัส มีค่าใช้จ่ายสูงกว่าแบบอะซิงโครนัส แสดงตัวอย่างของข้อมูลที่ส่งแบบซิงโครนัส พร้อมสัญญาณนาฬิกาได้ดังรูป 2.3



รูปที่ 2.3 แสดงรูปแบบข้อมูลชนิดซิงโครนัส

ข้อมูลในแบบซิงโครนัสนี้จะถูกจัดอยู่ในรูปของชุดข้อมูล (block of data) ที่มีลักษณะพิเศษ คือ ช่วงระยะเวลาระหว่างตัวอักษรด้วยกันจะไม่มี ทำให้การส่งข้อมูลเป็นไปอย่างต่อเนื่อง ซึ่งตัวอักษรจะแทนด้วยรหัสเลขฐานสอง เช่น รหัสแอสกี้ ดังรูป 2.4



รูปที่ 2.4 แสดงการส่งข้อมูลแบบ ซิงโครนัส

จากรูปที่ 2.4 เป็นตัวอย่างที่อธิบายได้อย่างดี โดยเราสรุปประเด็นสำคัญจากภาพได้ดังนี้

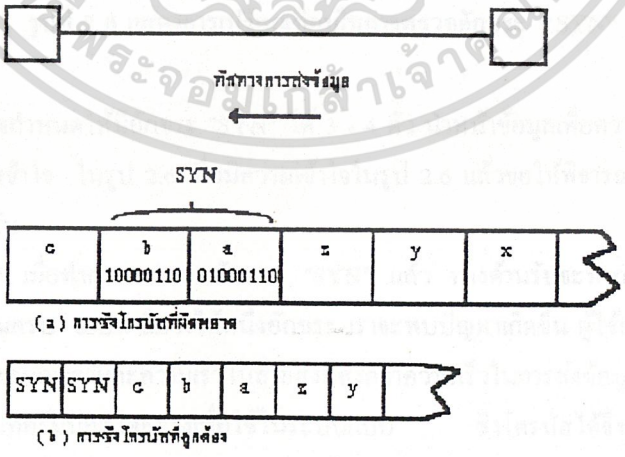
ตัวอักษรจาก A - K ถูกส่งไปอย่างต่อเนื่อง โดยที่ช่วงเวลาระหว่างตัวอักษร มีค่าเท่ากับศูนย์ ทางด้านรับ (ด้านเอ) จะต้องทราบตำแหน่งบิทแรกของตัวอักษรตัวแรกสุด รวมทั้งรู้ขนาดของตัวอักษรและความเร็วในการส่งด้วย อย่างเช่น ในกรณีตัวอักษรเป็นรหัสแอสกี้ พบว่าแต่ละตัวอักษรมีขนาด 8 บิท เมื่อถึงจุดนี้พบว่าปัญหาที่เกิดขึ้นคือ "ทำอย่างไรจึงจะรู้ตำแหน่งบิทแรกของตัวอักษรตัวแรกได้" มีการแก้ปัญหาโดยกำหนดตัวอักขระพิเศษที่ใช้เฉพาะทำหน้าที่การซิงโครนัสเท่านั้น เรียกว่า อักขระควบคุมการซิงโครนัส (เขียนย่อว่า SYN)

การที่มีอักขระ "SYN" เพื่อแจ้งให้ด้านรับทราบว่า ข้อมูลหลังจากอักขระ SYN คือ ข่าวสารที่ต้องการติดต่อให้รับได้เลย ส่วนทางด้านรับจะมีขั้นตอนทำงาน ดังนี้

1.ด้านรับทำการตรวจสอบ (detect) อักขระ SYN ในสายข้อมูลให้พบก่อนโดยมีกำหนดรูปแบบบิทของอักขระ "SYN" คือ 00010110 (ใช้พาริตี้คี่)

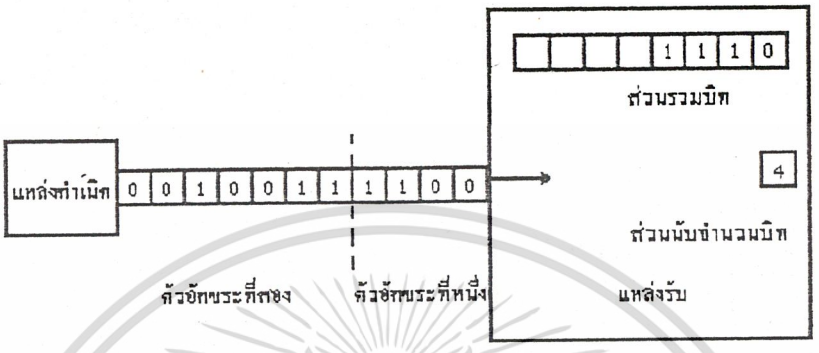
2.ตำแหน่งที่ด้านรับกำลังตรวจสอบ จะนำข้อมูลขณะนั้นไปเปรียบเทียบกับอักขระ "SYN" ที่ด้านรับเก็บไว้ ถ้าตรงกันแสดงว่าพบอักขระ "SYN" แล้ว และสามารถรับอักษรหลังจากนั้นไปได้อย่างต่อเนื่อง

อย่างไรก็ตาม ก็ยังมีปัญหาเกิดขึ้นตามมา ขอให้พิจารณาตัวอย่างในรูป 2.5 จากรูปพบกรณีที่หน้าสนใจ 2 กรณีแยกอธิบายดังนี้



รูปที่ 2.5 แสดงตัวอย่างของการหาอักขระซิงโครนัส

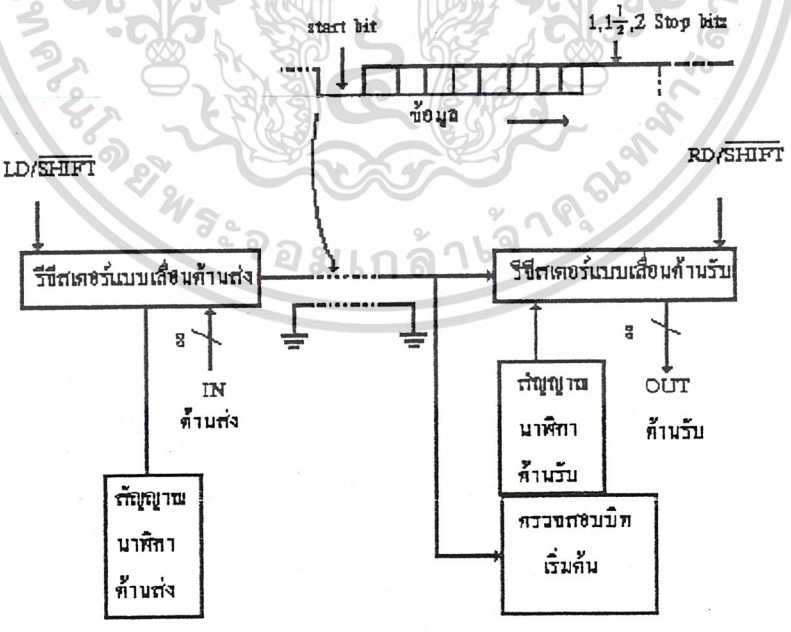
สำหรับอุปกรณ์รอบนอกที่มีความเร็วในการทำงานต่ำกว่าหน่วยประมวลผลกลางเช่น เครื่องพิมพ์ เพื่อให้การทำงานได้อย่างต่อเนื่อง เห็นได้ว่าส่วนบัฟเฟอร์ทำการเก็บข้อมูลจนเต็มหน่วยความจำแล้ว จึงส่งไปอย่างต่อเนื่องบนสายส่ง เพื่อรักษาการซิงโครนัสของข้อมูลได้



รูปที่ 2.7 แสดงการทำงานภายในด้านรับ

2.1.3.2 การส่งข้อมูลแบบอะซิงโครนัส

พิจารณาจากรูป 2.8 จากรูปพบว่าวิธีนี้ไม่จำเป็นต้องมีการซิงโครนัสกันตลอดเวลาบนทางติดต่อข้อมูล โดยจะมีการซิ้งก์ต่อเมื่อมีข้อมูลที่จะรับ/ส่ง เท่านั้น จึงต้องมีการใช้บิทเริ่มต้น (start bit) เพื่อให้ทางด้านรับตรวจ



รูปที่ 2.8 แสดงการส่งข้อมูลแบบอะซิงโครนัส

จับการเริ่มส่งของข้อมูล และมีผลให้วงจรกำเนิดสัญญาณนาฬิกาภายในด้านรับทำงานเพื่อเกิดการซิงโครนัสกันในการรับ/ส่ง ข้อมูล สำหรับความถี่ในการทำงานของวงจรกำเนิดสัญญาณนาฬิกา ทั้งด้านส่งและด้านรับมีค่าเท่ากันและขึ้นอยู่กับอัตราบิท (bit rate) ที่ใช้ด้วย นอกจากนี้มีการเพิ่มบิทจบ (stop bit) เพื่อบอกการสิ้นสุดของการส่งข้อมูลที่รับ/ส่งโดยมีขนาด 1 - 2 บิท เนื่องจากมีการเพิ่มบิทลงในข้อมูล ทำให้ความเร็วในการส่งข้อมูลช้ากว่าแบบ ซิงโครนัส แต่ค่าใช้จ่ายจะต่ำกว่า วิธีส่งแบบอะซิงโครนัสนี้เหมาะสำหรับงานประเภทการป้อนข้อมูลเพื่อส่งมีลักษณะไม่ต่อเนื่อง ส่วนการส่งไฟล์ข้อมูลที่มีขนาดใหญ่มาก ๆ ควรใช้วิธีแบบซิงโครนัส

2.1.4ระบบการสื่อสารข้อมูลแบบ ISDN

2.1.4.1การพัฒนาไปสู่ ISDN

ISDN ย่อมาจาก Integrated Service Digital Network หรือ เครือข่ายการบริการเชิงร่วมแบบดิจิตอล หลักการของ ISDN ได้ถูกเสนอแนะนำเป็นครั้งแรก ในการประชุมทั่วไปครั้งที่ 5 ของ CCITT ซึ่งจัดที่กรุงเจนีวา ในปี ค.ศ. 1972 ในครั้งนั้น ทาง CCITT ได้กำหนดข้อเสนอเกี่ยวกับการจัดกลุ่มระดับที่ 1 และ 2 ของ PCM รวมทั้งให้เริ่มต้นศึกษาวิจัยเกี่ยวกับการประยุกต์ ระบบสวิตช์วงจร,การกำเนิดสัญญาณต่างๆ และการส่งข้อมูลให้ใช้ งานได้ในเครือข่ายร่วมแบบดิจิตอล (Integrated DIgital Network:IDN) ในการประชุมครั้งนั้น ISDN ถูกเสนอด้วยแนวคิดว่า ถ้าอุปกรณ์ในระบบสวิตช์วงจรและระบบส่งสัญญาณ มีการทำงานแบบดิจิตอลแล้ว จะสามารถใช้สัญญาณเสียงโทรศัพท์แบบ PCM ได้ รวมทั้งใช้งานบริการอื่นได้ด้วยเช่น บริการสื่อสารข้อมูลคอมพิวเตอร์ , โทรสาร ฯลฯ เครือข่ายบริการสื่อสารร่วมระบบดิจิตอล ทำให้สามารถถ่ายทอดสัญญาณดิจิตอลได้อย่างสมบูรณ์โดยไม่มีการเปลี่ยนแปลง โดยอาศัยอุปกรณ์มาตรฐานชุดหนึ่ง ซึ่งสามารถให้บริการได้หลายรูปแบบ ระบบไอเอสดีเอ็นมีการพัฒนาจากยุคเริ่มต้นที่เรียกว่า Narrow Band ISDN (ISDN-N) ยุคต่อมาคือ ยุคของ Broad Band ISDN (ISDN-B) และเข้ามาสู่ยุคล่าสุดคือ ยุค Universal ISDN (ISDN-U)

2.1.4.2หลักการในการพัฒนา ISDN

ประเด็นหลักของ ISDN คือสามารถให้บริการในงานที่เกี่ยวกับเสียงและไม่เกี่ยวกับเสียงได้ในเครือข่ายเดียวกัน จุดสำคัญของการให้บริการร่วมในแบบ ISDN คือการมีการอินเตอร์เฟสระหว่างผู้ใช้และโครงข่ายได้หลายความต้องการ รวมทั้งจำกัดรูปแบบการส่งในเครือข่าย ISDN ให้มีน้อยลง ขอให้พิจารณารูป 2.9 ที่แสดงการพัฒนาเครือข่ายโทรคมนาคมชนิดต่างๆให้เป็นแบบ ISDN

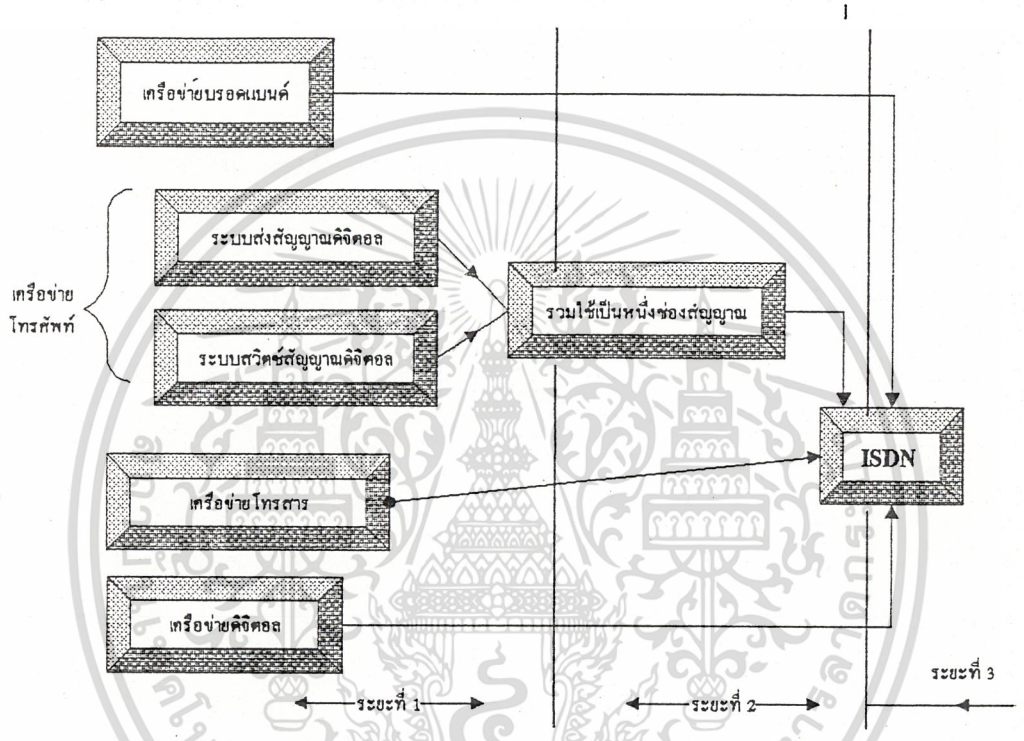
2.1.4.3 รายละเอียดต่าง ๆของเครือข่าย ISDN

จากที่ทุกวันนี้เริ่มเป็นสังคมแบบสาระสนเทศมากขึ้น ทำให้การบริการทางสื่อสารต้องรวดเร็วมาก ซึ่งในโครงข่ายการสื่อสารที่ได้พัฒนาขึ้นมาใหม่ มีความแตกต่างอย่างมากกับเครือข่ายที่ใช้กันอยู่ในปัจจุบันซึ่งพอจะสรุปเป็นประเด็นได้ดังนี้

-บริการสื่อสารที่เกิดขึ้นใหม่ ไม่นานมานี้การสื่อสารข้อมูลได้มีใช้อย่างมาก และคาดว่าจะมีบทบาทเด่นเข้ามาแทนที่ การสื่อสารทางโทรศัพท์ ขณะเดียวกัน บริการสื่อสารประเภทเอกสารอย่างเช่น เทเล็กซ์หรือบริการสื่อสารทางภาพ เช่น โทรสาร , วิดีโอเท็กซ์ หรือการประชุมทางจอทีวี สิ่งเหล่านี้คาดว่าที่การใช้มากขึ้น

รวมทั้งมีบริการใหม่ที่รวมเอาการรับรู้ทางตาและหูมาใช้ให้เป็นประโยชน์ และบริการเหล่านี้ไม่เพียงใช้ระหว่างคนด้วยกันเท่านั้น หากแต่ยังใช้ระหว่างคนกับเครื่องจักร หรือระหว่างเครื่องจักรด้วยกัน

-ความก้าวหน้าในการจัดการสื่อสาร ขบวนการต่างๆที่เกิดขึ้นในเครือข่ายมีไว้เพื่อความสะดวกและความง่ายในพัฒนาบริการสื่อสารที่มีอยู่โดยไม่ให้ข้อมูลที่ต้องการส่งมีผลกระทบแต่อย่างใดทั้งสิ้น



อ

รูปที่ 2.9 แสดงการพัฒนาไปสู่แบบ ISDN

2.1.5 สายส่งในงานโทรคมนาคม

2.1.5.1 ทฤษฎีพื้นฐานเกี่ยวกับสายส่ง

ถ้ามีการแยกสเปกตรัมของความถี่ในคลื่นสี่เหลี่ยมมุมฉาก(Square wave) พบว่าประกอบด้วยฮาร์โมนิกอันดับคี่เท่านั้น โดยเราวิเคราะห์คลื่นสี่เหลี่ยมมุมฉากได้ว่าประกอบด้วย

1.คลื่นรูปซายน์ ที่มีความถี่พื้นฐาน (Fundamental frequency) มีค่าเท่ากับ $f_1 = 1/T$ Hz โดยที่ T คือคาบเวลาที่ใช้วินาที

2.คลื่นรูปซายน์ ที่มีความถี่ฮาร์โมนิกอันดับคี่ (ตั้งแต่อันดับ 3 ขึ้นไป) มีค่าเท่ากับ

$$f_m = n/T \text{ Hz } (n = 3, 5, 7, ...)$$

โดยที่ถ้าคลื่นที่ความถี่พื้นฐานมีคลื่นที่ความถี่ฮาร์โมนิกอันดับคี่มารวมกันทางคณิตศาสตร์ตามสมการฟูเรียร์มากเท่าใด จะมีผลทำให้คลื่นผลลัพธ์ที่ได้มีรูปร่างใกล้เคียงคลื่นสี่เหลี่ยมมุมฉากตามทฤษฎีมากขึ้นเท่านั้น

สมการฟูเรียร์ของคลื่นสี่เหลี่ยมมุมฉากคือ

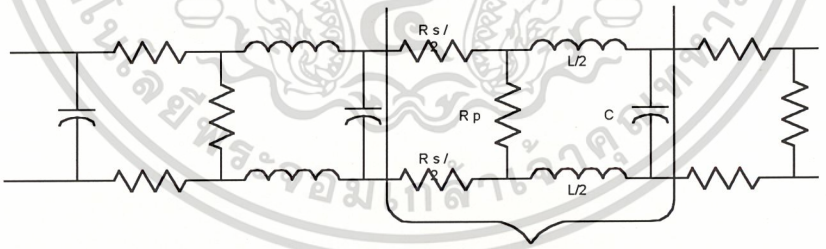
$$V_{(t)} = \frac{4}{\pi} Sin(2\pi f_1 t) + \frac{4}{3\pi} Sin(2\pi f_3 t) + \frac{4}{5\pi} Sin(2\pi f_5 t) + \ldots \quad \textbf{(2.1)}$$

สำหรับสัญญาณดิจิตอลก็มีรูปเป็นสี่เหลี่ยมมุมฉากเช่นกัน ถึงแม้ว่าบางครั้งช่วงดิวตี้ไซเคิลจะไม่เท่ากับ 50% แน่นอนก็ตาม แต่การที่รูปคลื่นมีช่วงเวลาขอบขาขึ้นและขอบขาลงน้อยมากนั้นเป็นเหตุที่พอชี้ได้ว่าประกอบด้วยฮาร์โมนิกลำดับสูงมากได้เช่นกัน ส่วนปัญหาในการนำรูปสัญญาณดิจิตอลนี้ ไปบนสายส่งก็มีประเด็นที่น่าสนใจอยู่สองอย่างคือ ค่าการลดทอนในสาย (attenuation) และ ค่าหน่วงเวลาสัญญาณ (delay) มีค่าเท่ากันทุกจุดไปตลอดสายส่ง เรากล่าวได้ว่าสัญญาณที่รับได้ทางด้านรับจะมีรูปคลื่น (waveform) เช่นเดียวกับที่ส่งมาจากด้านส่ง (ถึงแม้จะมีขนาดเล็กลง แต่ก็สามารถนำมาขยายได้) ถ้าเราพิจารณาเงื่อนไขข้างต้นในเทอมของตัวแปร (parameter) ของสายส่งพบว่าตัวแปรที่เกี่ยวข้องด้วยสองตัว คือ ค่าการลดทอนที่คงที่ (Attenuation constant) และ ค่าความเร็วเฟส (Phase Velocity) ทั้ง 2 เทอมนี้ ต้องมีค่าคงที่ในตลอดช่วงความกว้างแถบของสัญญาณ มิฉะนั้นจะเกิดความสูญเสียของรูปสัญญาณได้สำหรับตัวแปรต่างๆในสายส่งจะอธิบายในตอนต่อไปนี้

2.1.5.2ค่าของตัวแปรต่าง ๆ ในสายส่ง

2.1.5.2.1วงจรสมมูลในสายส่ง (Equivalent circuit)

ในสายส่งแบบลวดตัวนำ 2 เส้น (2-wire) เราสามารถแยกคุณสมบัติได้ในเทอมของตัวแปรดังนี้ ค่าความต้านทาน (R), ค่าความจุไฟฟ้า (C)และค่าความเหนี่ยวนำไฟฟ้า (L) จากการวิเคราะห์พบว่าตลอดความยาวของสายที่มีค่า **R , L ,C** จัดเรียงตัวเป็นรูปแบบที่แน่นอน และสามารถเขียนแทนด้วยวงจรไฟฟ้าได้ดังรูป 2.10



แทนวงจร 1ลูป

รูปที่ 2.10 แสดงวงจรไฟฟ้าของสายส่ง

เพื่อความง่ายจึงคิดวงจร 1 ลูปต่อความยาวสายส่ง 1 เมตร โดยกำหนดให้

$\mathbf{R_S}$ แทน ความต้านทานที่ต่ออนุกรมกับลวดตัวนำ

R_p แทน ความต้านทานของฉนวนที่หุ้ม

C แทน ความจุไฟฟ้า ระหว่างลวดตัวนำ 2 เส้น

L แทน ความเหนี่ยวนำไฟฟ้าของลวดตัวนำ

โดยทั่วไปค่า R , C , L ในสายแต่ละชนิด จะมีค่าแตกต่างกันไปขึ้นอยู่กับขนาดของลวดตัวนำ ช่องห่างระหว่างลวดตัวนำ และชนิดของฉนวนที่ใช้ ยกตัวอย่างสายคู่ตีเกลียว **(twisted-pair line)** ขนาด 22 เกจ มีค่าดังนี้

L	= 2	mH
C	= 50	pF/m
R_S	= 0.1	Ω/m
R_p	= 800	MΩ/m

แต่ในการใช้งานจริงปัจจัยอื่นมีผลกระทบต่อค่าเหล่านี้ได้เช่นกัน ยกตัวอย่างเช่น อุณหภูมิ , ความชื้น และ ความถี่

โดยเฉพาะปรากฏการที่เรียกว่า "ผลกระทบจากผิว " (Skill Effect) ที่ทำให้ค่า R_s สูงมาก เป็นสัดส่วนโดยตรงกับค่ารากที่สองของความถี่ที่ใช้งานได้ เห็นได้ชัดมากในการใช้งานความถี่สูง

ถ้าลวดตัวนำมีรอยปริแตกหรือหักจะทำให้เกิดการรั่วไหลของฟลักแม่เหล็กจากแกนกลางของลวดตัวนำ มีผลทำให้ไหลย้อนของสนามแม่เหล็กจำนวนมากทำให้กระแสที่ไหลอยู่หลุดจากผิวตัวนำไป

2.1.5.2.2 ประสิทธิภาพของสายส่ง

การพิจารณาประสิทธิภาพดูจากตัวแปร 3 ตัว ดังนี้

-ความเร็วเฟส (Phase Velocity : V_p)

ค่านี้คิดจากความเร็วในการแพร่กระจายคลื่นในสายส่งในหน่วย เมตร/วินาที ในทางทฤษฎีแล้ว เรากำหนดให้ค่า V_p มีค่าคงที่ตลอดทุกความถี่ที่ส่งในสาย ซึ่งการมีค่าคงที่นี้สามารถทำให้องค์ประกอบของสัญญาณให้มีเฟสสัมพันธ์กับด้านส่งได้ เป็นการลดการสูญเสียทางเฟส

-ค่าการลดทอนคงที่ (Attenuation :a)

ค่านี้คิดจากกำลังที่สูญเสียในสายต่อความยาว 1 เมตรของสายส่งและเหมือนกันกับค่า V_p เราก็กำหนดให้ค่า a มีค่าคงที่ (ในทางทฤษฎี) เพื่อป้องกันการสูญเสียทางแอมปลิจูดหรือขนาดของสัญญาณ (amplitude distortion)

ค่า a อธิบายในหน่วย เนเปอร์ ต่อเมตร (nepers per meter) ซึ่งค่าเนเปอร์แทนค่าลอกการิธึมของกำลังโดยใช้ฐาน " e " ที่ไม่เหมือนกับค่าเดซิเบลที่ใช้ฐานเป็น 10log ถ้าต้องการเปลี่ยนค่าจากเนเปอร์ไปเป็นเดซิเบลให้คูณด้วย 8.686

-ค่าอิมพีแดนซ์ของสาย (Zo)

ค่านี้คิดจากอิมพีแดนซ์ภายในสายส่งเอง มีหน่วยเป็นโอห์ม และสามารถแสดงเป็นสมการได้ว่า

$$Zo=Ro+jXo \qquad (2.2)$$

ถ้าต้องการให้ไม่มีการสูญเสียกำลังเนื่องจากการสะท้อนกลับ (reflection loss) จากด้านรับหรือโหลด ต้องกำหนดเงื่อนไขว่า

" ค่าอิมพีแดนซ์ของแหล่งส่งและแหล่งรับต้องมีค่าเท่ากับ **Zo** "ภายใต้เงื่อนไขนี้ทำให้สัญญาณส่งผ่านได้ตลอดโดยมีการสะท้อนกลับมาบางส่วน ในทางทฤษฎีแล้ว มีการกำหนดให้ค่า **Zo = Ro** (ให้ **Xo = 0** W) และมีค่าคงที่ตลอดช่วงความกว้างแถบของสัญญาณ

2.1.5.2.3 ค่าการสูญเสียของสัญญาณ (Distortion)

ปรากฏการณ์สูญเสียที่เกิดขึ้นกับสัญญาณที่รับได้มีสาเหตุมาจากค่าความเร็วเฟสไม่คงที่ ทำให้เกิดการแยกกระจายของสัญญาณเกิดขึ้น (dispersion) ซึ่งทำให้เกิดสูญเสียทางเฟสและแอมปลิจูด

2.1.5.3 สายนำสัญญาณ (Transmission lines)

ในที่นี้ หมายถึง ตัวกลางที่เป็นสายส่งโดยจะแยกอธิบายแต่ละแบบดังนี้

ก. สายคู่ตีเกลียว (Twisted - pair cable)

จัดว่ามีราคาถูกกว่าแบบอื่น ๆ ประกอบด้วยลวดตัวนำ 2 เส้น พันระหว่างกันสามารถส่งข้อมูลแบบอนาลอก หรือดิจิตอลได้ แต่ไม่สามารถใช้กับการส่งข้อมูลที่มีความเร็วสูงได้ เพราะมีค่าความกว้างแถบแพียง 3.1 KHz เท่านั้น

เรานิยมใช้สายนี้โดยจัดเป็นกลุ่ม ที่อาจมีคู่สายได้จาก 4-3000 คู่ ซึ่งในระบบข่ายงานบริเวณเฉพาะที่ (LAN) นิยมใช้แบบ 25 คู่

เราแบ่งรูปแบบตามลักษณะของสายได้ 2 ประเภท

- สายแบบ STP (Shield twisted pair)

สายนี้มีการใช้อยู่ในวงจำกัดเมื่อเทียบกับอีกแบบ แต่ถ้าคิดในด้านคุณภาพแล้วดีกว่าแน่ ตัวอย่างที่มีการใช้สายแบบ STP คือในระบบ LAN ของ **AT & T** ที่เรียกกันว่า **Starlan** มีการระบุให้ใช้สาย STP โดยคู่หนึ่งใช้ส่งข้อมูล อีกคู่หนึ่งใช้รับข้อมูล

สำหรับระยะทางการส่งข้อมูลใช้ได้ไกลถึง 300 เมตร

- สายแบบ UTP (Unshield twisted pair)

สายชนิดนี้ก็คือ สายโทรศัพท์ธรรมดาที่เราใช้กันทั่วไป มีการใช้งานอย่างแพร่หลาย โดยเฉพาะในงานระบบแลน ด้วยเหตุว่า

มีความสะดวกในการใช้งานมากกว่าสายโคแอกเชียลเคเบิล เพราะสามารถเข้าได้กับแลนเกือบทุกระบบ และมีความประหยัดกว่าเพราะสามารถใช้เป็นสายโทรศัพท์ได้ด้วย

แต่ข้อเสียของสายโทรศัพท์แบบ UTP คือ ระยะทางในการส่งข้อมูลมีการจำกัดได้เพียง 100 เมตรเท่านั้น

ข. สายโคแอกเชียล (Coaxial Cable)

สามารถทำงานในย่านความถี่สูงได้ และค่าความกว้างแถบมากกว่าสายโทรศัพท์ มีผลทำให้ความเร็วในการส่งข้อมูลสูงกว่าแบบคู่ตีเกลียวได้ ประเภทของสายโคแอกเชียล แบ่งได้ 2 ประเภท ตามค่าอิมพีแดนซ์ภายในสาย

- สายประเภท 50 โอมห์ มีการใช้งานในระบบ LAN โดยนิยมใช้เป็นสายหลักของเครือข่าย

- สายประเภท 75 โอมห์ มีการใช้งานในงานสายเคเบิลทีวี (**CATV cable**) และสามารถใช้ในงานระดับแลนได้เช่นกัน

ค. สายใยแสง (Fiber-optic cable)

จัดเป็นเทคโนโลยีในด้านสายส่งที่กำลังมีบทบาทอย่างมากในงานวิศวกรรม เนื่องจากเป็นการใช้หลักการแปลงรูปสัญญาณในทางไฟฟ้าให้เป็นสัญญาณแสง และมีความกว้างแถบที่กว้างมาก ทำให้สามารถส่งข้อมูลที่อัตราเร็วสูงมากได้ ข้อดีที่เด่นชัดของการใช้เส้นใยแสง คือขนาดเล็กอย่างเห็นได้ชัด เมื่อเปรียบเทียบกับสายเคเบิลแบบอื่น

2.1.5.4 การมอดดูเลทสัญญาณทางดิจิตอล

ในการมอดดูเลทสัญญาณทางดิจิตอลนั้น มี 3 วิธี แต่ที่จะเน้นหนักคือวิธีสุดท้าย เพราะว่าวิธีดังกล่าวใช้ในโครงงานชิ้นนี้ รายละเอียดจะกล่าวในภายหลัง

1. การมอดดูเลทดิจิตอลทางขนาด (Amplitude Shift Keying :ASK)

กำหนดรูปสมการได้ตามสถานะของบิท คือ

(คลื่นพาหะ) $a_c = A_c \sin 2\pi f_c t$ เมื่อแสดงสถานะของบิทเป็นหนึ่ง

~ $= 0$ เมื่อแสดงสถานะของบิทเป็นศูนย์

2. การมอดดูเลทดิจิตอลทางความถี่ (Frequency Shift Keying :FSK)

กำหนดรูปสมการได้ตามสถานะของบิท คือ

(คลื่นพาหะ) $a_c = A_c \sin 2\pi f_1 t$ เมื่อแสดงสถานะของบิทเป็นหนึ่ง

~ $= A_c \sin 2\pi f_2 t$ เมื่อแสดงสถานะของบิทเป็นศูนย์

3. การมอดดูเลทดิจิตอลทางเฟส (Phase Shift Keying :PSK)

ถ้าการมอดดูเลทชันทางเฟสถูกใช้ในการส่งสัญญาณทางดิจิตอล มักจะรู้จักกันในนาม phase shift keying (PSK) ถ้าสัญญาณนั้นเป็นสัญญาณไบนารี เฟสของสัญญาณพาหะ (Carreir) จะถูก shift ระหว่างสองตำแหน่งซึ่งแตกต่างกัน 180° ซึ่งบางครั้งเรียกว่า phase reversal keying (PRK)

สัญญาณแถบไบนารี(Baseband) Vm ด้วยค่า -1 และ +1 นำมารวมกับมอดดูเลเทอร์ผลิต Double Sideband Suppressed Carrier (DSBSC) เป็นสัญญาณ Vc ดังสมการ (คลื่นพาหะ)

$v_c = V_c \cos w_c t$ เมื่อแสดง Vm = +1

$v_c = V_c \cos (w_c t + p)$ เมื่อแสดง Vm = -1

PSK ใช้ DSBSC amplitude modulation ซึ่งต้องการแบนวิดธ์เดียวกันกับ ASK และน้อยกว่า FSK นอกจากนี้ PSK ยังเป็น suppressed carrier system กำลังการส่งข้อมูล (transmitted power) อยู่ในช่วง sideband เมื่อเปรียบเทียบกับ FSK และ ASKแล้วจะใช้เพียงครึ่งเดียวของกำลังทั้งสองแบบที่อยู่ในช่วง sideband ซึ่งน้อยกว่าประมาณ 3 dB ดังนั้น ณ. จุดการกำลังการส่งเดียวกัน PSK จะมีค่าน้อยกว่า FSK หรือ ASK

สัญญาณ PSK มีได้มากกว่าสองระดับ ตัวอย่างเช่น 4-ระดับ PSK ใช้สี่แคร์เรียเฟส แยกกัน 90 องศาซึ่งบางครั้งรู้จักกันในนาม quadrature phase shift keying (QPSK) จะใช้ส่งสัญญาณแถบไบนารีเป็นคู่มี 4 ระดับคือ 00, 01, 10, 11 ตามเฟสทั้ง4 ซึ่งทำให้อัตราการส่งสูงขึ้น

ในการส่งข้อมูลที่มีอัตราการส่งข้อมูลที่สูงขึ้นนี้ อาจใช้มัลติเฟส PSK ซึ่งจะช่วยลดสัญญาณรบกวนที่ดีกว่า สำหรับสองเฟส PSK ต้องการเปลี่ยนแรงดัน Vc ซึ่งช่วงขณะทำการเปลี่ยนจะมีการผิดพลาดในการตรวจจับสัญญาณ 1 และ 0 อันเนื่องมาจาก noise ที่มารบกวน แสดง PSK แบบ 4 เฟส ค่าผิดพลาดในการส่งจะลดลงประมาณ 0.707 เท่าหรือ 3dB จะเห็นว่าช่วยการในเรื่องของ noise และข้อดีอีกประการหนึ่งก็คือ สี่เฟส PSK ต้องการแบนวิดธ์เพียงครึ่งหนึ่งของสองเฟส PSK กำลังของ noise ลดลง 3dBแต่ค่าความความผิดพลาดยังคงเดิม

2.2 รหัสแถบ (BARCODE)

2.2.1 ความเป็นมาของรหัสแถบ

สหรัฐอเมริกาได้ออกสิทธิบัตรรับรองรหัสแถบ ขึ้นเมื่อปี ค.ศ.1949 ในแบบที่เรียกว่า Circular Bar Code ต่อมาในปี ค.ศ.1960 ก็มีการรองรับรหัสแถบแบบที่เรียกว่า Rail identification symbol หลังจากนั้นเป็นต้นมาเทคนิคของรหัสแถบรูปแบบต่างๆ ก็มีมากขึ้น และเริ่มใช้งานอย่างจริงจังเมื่อปี ค.ศ.1970 เมื่อคณะกรรมการบริหารด้านห้างสรรพสินค้าของสหรัฐอเมริกา ได้นำรหัสที่เรียกว่า UPC (Universal Product Code) ซึ่งเป็นรหัสที่ใช้กันมากในสินค้า ออกเผยแพร่และใช้กันอย่างแพร่หลายในสหรัฐอเมริกา และยุโรป ตั้งแต่ปี ค.ศ.1973 และ ค.ศ.1977 ตามลำดับ

การใช้งานในด้านอื่น ๆ เริ่มตั้งแต่ปี ค.ศ.1980 เป็นต้นมา แต่คนส่วนใหญ่เริ่มคุ้นเคยกับรหัสแถบเป็นอย่างจากรหัสสินค้า และการชำระเงินที่คอมพิวเตอร์รวมออกมาจากการอ่านรหัสแถบบนสินค้าเหล่านั้น จากความสะดวกเหล่านี้ สามารถลดพนักงาน ณ จุดนี้ลงได้

ในปี ค.ศ.1981 มีห้างสรรพสินค้ามากกว่า 4,000 แห่ง ในสหรัฐอเมริกาและแคนาดาใช้รหัสแถบในธุรกิจนี้ นอกจากนี้ยังใช้กับ กิจการอื่น เช่น ห้องสมุด บริการสุขภาพงานบริหาร การผลิตสินค้าเป็นต้น

2.2.2 หลักการของรหัสแถบ

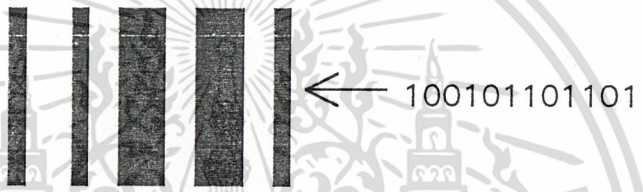
รหัสแถบเป็นการแทนข้อมูลที่เป็นรหัสเลขฐานสอง ในรูปแบบของแถบขาวดำ ที่มีความกว้างแคบต่างกัน วางเรียงขนานสลับกัน ด้วยจำนวนของแถบสลับกันไป ขึ้นอยู่กับข้อมูลที่ต่างกันและชนิดของรหัสแถบที่เลือกใช้ต่างกันด้วย

การเข้ารหัสของรหัสแถบ แบ่งเป็น 2 วิธีการคือ

1 . จะใช้สีของรหัสแถบนำเข้ามาเข้ารหัสโดยใช้สีดำแทน "1" และแถบสีขาวหรือแถบว่างเป็น "0" ลักษณะเช่นนี้เรียกว่า "เดลต้าโคด" (Delta Code)

2.จะใช้ความกว้างของรหัสแถบนำมาเข้ารหัส โดยถ้าเป็นแถบกว้างจะเป็น "1" และแถบแคบจะเป็น "0" ลักษณะเช่นนี้เรียกว่า "วิดท์โคด" (Width Code) การเข้ารหัสชนิดนี้จะไม่สนใจสีของแถบเลย
แถบขาวดำ มีลักษณะและชื่อที่ใช้คือ

- แถบสีที่มีความกว้างมากเรียกว่า Wide Bar
- แถบสีที่มีความกว้างน้อยเรียกว่า arrow Bar
- ช่องว่าง(แถบสีขาว) ที่มีความกว้างมาก เรียกว่า Wide Space
- ช่องว่าง (ช่องสีขาว) ที่มีความกว้างน้อย เรียกว่า Narrow Space



รูปที่ 2.11 การแทนค่าเลขฐานสองของแถบต่างๆ

2.2.3 การอ่านรหัสแถบ

ในการอ่านรหัสแถบใช้หลักการเปลี่ยนรหัสแถบให้เป็นรหัสแอสกี้ โดยอาศัยความแตกต่างกันระหว่างแถบเข้มกับพื้นที่ว่าง โดยพื้นที่ว่าง (ปกติจะเป็นสีขาวหรือสีอ่อน) จะมีการสะท้อนกลับของแสงได้มากกว่าบริเวณที่เป็นแถบเข้ม (ซึ่งใช้สีดำหรือสีอื่นที่มีความเข้มมาก) หัวอ่าน (Barcode Reader) จะประกอบด้วยด้วยตัวกำเนิดแสงที่ส่งผ่านเลนส์ออกมา โดยถูกบังคับทิศทางให้มีจุดรวมแสงเล็กที่สุด กับตัวรับแสงที่มีความไวสูง ทั้งสองอย่างนี้จะบรรจุไว้ในหัวอ่านเดียวกัน ที่มีหลายรูปแบบ แต่แบบที่เป็นพื้นฐานที่สุดอยู่ในรูปคล้ายปากกาขนาดใหญ่ (Wand type)

หัวอ่านจะสแกนผ่านรหัสแถบในขณะที่ตัวกำเนิดแสง จะทำให้เกิดแสงผ่านเลนส์ไปกระทบบนรหัสแถบและสะท้อนกลับจากแถบกลับไปยังตัวรับแสง (Photosensor) ที่เกิดค่าความแตกต่างขึ้นตามหลักการสะท้อนกลับในแต่ละแถบ ทำให้เกิดสภาวะลอจิก "1" และสภาวะลอจิก "0"ขึ้นตามที่กล่าวมาแล้วข้างต้น ซึ่งเมื่อรวมสภาวะลอจิก "1"และ "0" ทั้งหมด ตลอดความกว้างของทุกแถบแล้วจะตรงกับรูปแบบที่กำหนดไว้แล้ว ในหัวอ่านรหัสแถบจะใช้ตัวกำเนิดแสงสีแดงหรือสีขาว แต่ส่วนใหญ่จะใช้สีแดงเนื่องจากแสงสีขาวต้องการพลังงาน และความเข้มของแสงสูงกว่าแสงสีแดง แสงสีแดงสามารถอ่านรหัสที่พิมพ์ด้วยสีต่าง ๆ ได้ทุกสี ยกเว้นรหัสที่พิมพ์ด้วยสีแดง

องค์ประกอบที่สำคัญสองประการที่จำเป็นอย่างมากในการอ่ารหัสแถบได้ถูกต้อง

ประการแรกคือ พื้นที่ภายในแถบและช่องว่าง จะต้องทำให้เกิดความแตกต่างของการสะท้อนกลับอย่างมาก (contrast) เช่นแถบสีดำและช่องว่างสีขาว เผื่นต้น ซึ่งปรกติความแตกต่างนี้จะต้องอยู่ในช่วงระหว่างอัตรา 80-90 % ขึ้นไป

ประการที่สองคือ ความกว้างระหว่างแถบกว้าง หรือช่องว่างต่อแถบแคบ หรือช่องว่างแคบจะเป็นอัตราส่วน 2:0.5, 2:1 และ 3:1

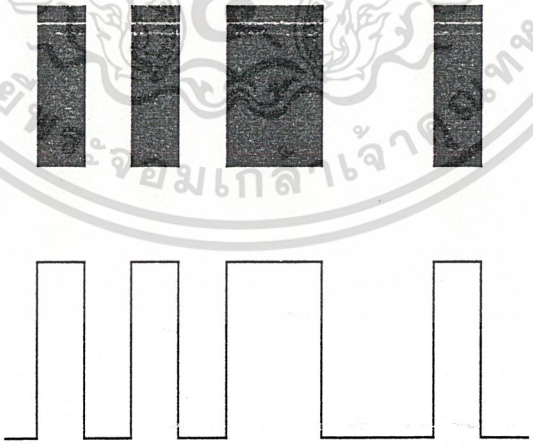
ตามหลักของรหัสแถบแล้ว สัญญาณที่อ่านได้จากหัวอ่านจะไม่ขึ้นกับชนิดของรหัสแถบ แต่จะขึ้นกับรหัสแถบขาว-ดำ ที่รูดผ่าน คือถ้าเป็นแถบดำสัญญาณที่อ่านได้จากหัวอ่านจะเป็น"1" ถ้าเป็นแถบขาวสัญญาณที่อ่านได้จะเป็น "0" ความกว้างของสัญญาณที่อ่านได้จะเท่ากับความกว้างของแถบขาว-ดำ

2.2.4 ชนิดของรหัสแถบ

ปัจจุบันชนิดของรหัสแถบที่นิยมใช้กันแพร่หลายแบ่งได้เป็น

1. ชนิดรหัส 2 ใน 5 (2 of 5 Code)
2. ชนิด 2 ใน 5 แบบสอดแทรก (interleaved 2 of 5)
3. ชนิด 3 ใน 9 (3 of 9 or 39 Code)
4. ชนิดรหัส Codabar
5. ชนิดรหัส UPC (Universal Product Code)
6. ชนิดรหัส EAN (European Article Numbering)
7. ชนิดรหัส Code 93

ซึ่งในการทำโปรเจคจะใช้เฉพาะ รหัสแถบชนิด 3 ใน 9 หรือรหัส 39 เท่านั้นจึงขออธิบายรายละเอียดเฉพาะ รหัสชนิด 3 ใน 9 เท่านั้น



รูปที่ 2.12 แสดงสัญญลักษณ์ที่อ่านได้จากหัวอ่าน

2.2.5 รหัสแถบชนิด 3 ใน 9

ชื่อของรหัสแถบชนิดนี้บอกถึงโครงสร้างของรหัสว่า 3 ใน 9 ส่วนตัวอักษรจะเป็นแถบกว้าง และที่เหลือเป็นแถบแคบอีก 6 แถบ แต่ละตัวอักษรใน รหัส 39 จะแสดงเป็นกลุ่มของแถบดำ 5 แถบ และเป็นแถบว่าง 4 แถบแต่ละตัวอักษรที่สมบูรณ์จะต้องรวมตัวอักษร เริ่ม(start) และตัวอักษรหยุด (stop) เข้าไปด้วย โดยใช้เป็นตัว Asterisk (*) ข้อมูลทั้ง 43 ตัวอักษร ประกอบด้วยตัวเลข 10 ตัว คือ 0-9 ตัวอักษร 26 ตัวอักษร 26 ตัว A-Z space และเครื่องหมายอีก 6 ตัว (-,.,$,/,+,%)

คุณสมบัติที่สำคัญของ รหัส 39 คือมี การตรวจสอบความถูกต้องในตัวเอง (self-checking) ซึ่งทำให้ข้อมูลมีปลอดภัยสูง และถ้ามีเครื่องอ่านที่ดี และมีรหัสแถบที่มีคุณภาพสูง จะพบข้อผิดพลาด เพียง 1 ใน 70 ล้านครั้งของการอ่าน และถ้ารหัวแถบมีการพิมพ์มาอย่างดี(ใช้เครื่องพิมพ์ที่มีคุณภาพสูงกว่า เครื่องพิมพ์แบบหัวเข็ม) จะพบข้อผิดพลาดเพียง 1 ครั้ง ใน 3 ล้านครั้ง

รหัส 39 เป็นรหัสที่มีความยาวเปลี่ยนแปลงได้ ขึ้นอยู่กับเครื่องอ่านที่ใช้ และ รหัส 39 มีการตรวจสอบความถูกต้องในตัวเอง จึงไม่ต้องมีอักขระสำหรับตรวจสอบ รหัส 39 เป็นอักษรชนิดไม่ต่อเนื่อง โดยจะมีช่องว่างระหว่างตัวอักขระ และเป็นรหัสที่อ่านได้สองทาง คือสามารถอ่านจากซ้ายไปขวาหรืออ่านจากขวามาซ้ายก็ได้เช่นเดียวกัน ขนาดของรหัส 39 จะเปลี่ยนไปตามความกว้าง สำหรับรหัส 39 ที่มีความหนาแน่นสูงจะมี 9.4 ตัวอักษรต่อนิ้ว และรหัส 39 ที่มีความหนาแน่นต่ำจะมี 1.4 ตัวอักษรต่อนิ้ว

2.2.6 สรุปคุณลักษณะของรหัส 39

รูปแบบตัวอักษรทั้งหมด อักษรภาษาอังกฤษตัวใหญ่ 26 ตัว (A-Z)
ตัวเลข 10 ตัว
อักษรพิเศษ 7 ตัว

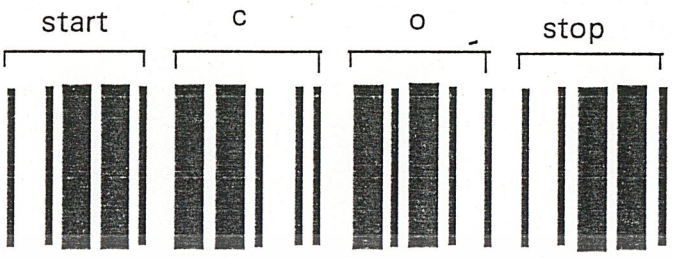
- สามารถขยายได้เป็น 128 ตัวอักษรตาม รหัสแอส ถ้าใช้อักษรนำหน้ารหัสต่างๆ 2 ตัว
- ความยาวของแถบรหัสเปลี่ยนแปลงได้
- อักษรที่ใช้ตรวจสอบความผิดพลาด เป็นส่วนที่มีหรือไม่มีก็ได้
- อักษร overhead 2 ตัว ต่อ 1 แถบรหัส
- ความหนาแน่นมากที่สุดได้ 9.8 หลักต่อ 1 นิ้ว เมื่อพิมพ์ 7.5 mil xdimention

2.2.7 คำอธิบายของรหัสชนิด 39

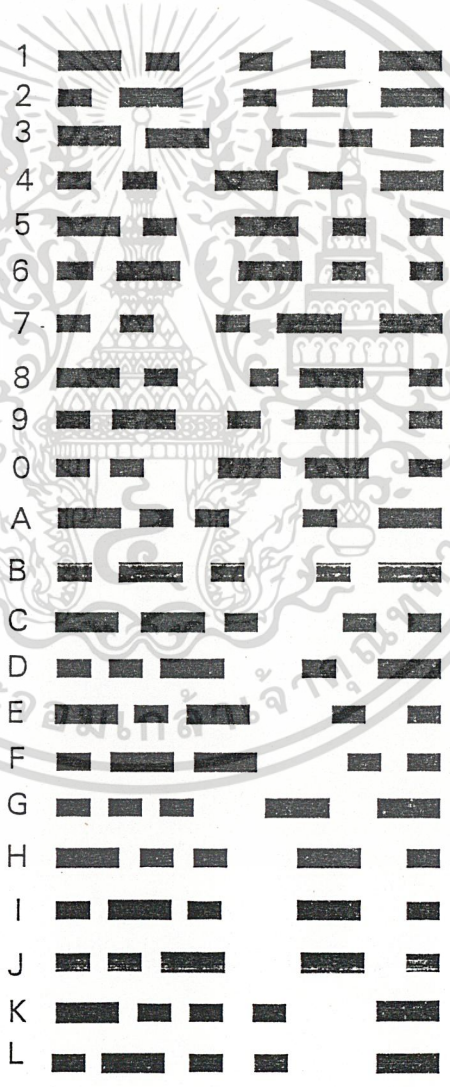
ทุก ๆ แถบของรหัสแถบชนิด 39 จะประกอบด้วย

1. นำด้วย quiet zone
2. รูปแบบตัวอักษรเริ่มต้น
3. อักษรข้อมูล
4. รูปแบบตัวอักษรสิ้นสุด
5. ปิดท้ายด้วย quiet zone

ซึ่งมีลักษณะโครงสร้างดังรูปที่ 2.3



รูปที่ 2.13 แสดงโครงสร้างของรหัส 39



รูปที่ 2.14 แสดงถึงตัวอย่างแบบแต่ละตัวอักษรของรหัส 39

ASCI	CODE 39	ASCII	CODE39	ASCII	CODE39	ASCII	CODE39
NUL	%U	SP	Space	@	%V	‘	%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	“	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	‘	/G	G	G	g	+G
BS	$H	(	/H	H	H	h	+H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	‘	/L	L	L	l	+L
CR	$M	-	-	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	O	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W

CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<	%G	\	%L	|	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	↑	%N		%S
US	%D	?	%J	_	%O	DEL	%T ,%X ,%Y,%Z

ตารางที่ 2.2 อักษรของรหัส 39 เทียบกับค่าของรหัสแอสกี้

2.2.8 การตรวจสอบความถูกต้องของรหัส 39

อักขระตรวจสอบใช้ช่วยตรวจสอบความถูกต้องของข้อมูล อักขระที่ใช้ตรวจสอบนี้จะใส่เป็นหลักของข้อมูล หลังจากที่ถูกเข้ารหัสแล้ว โดยค่าของหลักที่ใช้ตรวจนี้ จะคำนวณได้ตามนี้

1. ใช้ตารางที่ 2.2 ให้ค่าแก่รหัสข้อมูลต่างๆ
2. รวมผลรวมของทุก ๆ ข้อมูล
3. หาเศษที่ได้จากข้อ 3 ไปเทียบ กับตารางที่ 2 ว่าเป็นอักขระอะไร อักษรนั้นจะเป็นอักษรตรวจสอบ

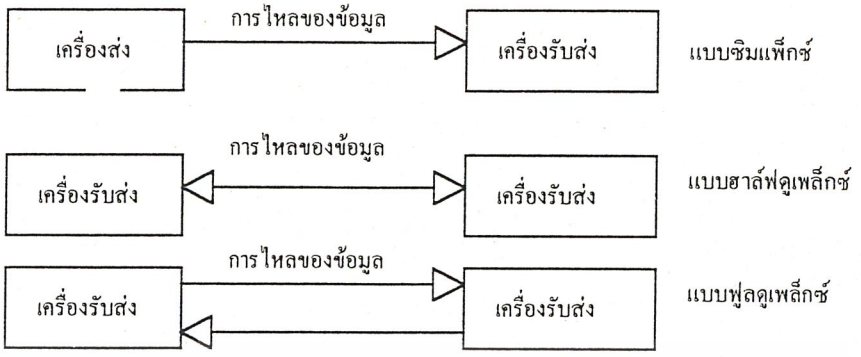
ตัวอย่างเช่นแถบรหัสมีข้อมูล "TEST" ผลรวมจะเป็น 29+14+28+29=100 ผลหารของ 100 ด้วย 43 คือ 2 เศษ 4 ฉะนั้นอักษรตรวจสอบคือ "E"

2.3 การรับส่งข้อมูลผ่านพอร์ตอนุกรม (SERIAL PORT TRANSMISSION)

พอร์ตอนุกรม เป็นอุปกรณ์ที่ใช้สำหรับรับส่งข้อมูลซึ่งมีการใช้งานกันอย่างแพร่หลาย โดยปกติคอมพิวเตอร์ทุก ๆ เครื่องจะมีพอร์ตอนุกรมมากับเครื่องอยู่แล้วซึ่งจะเป็นพอร์ตอนุกรม RS-232C โดยสมาคมผู้ผลิตอุปกรณ์อิเลคโทรนิกส์ของสหรัฐอเมริกา (EIA) ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมเอาไว้ภายใต้ชื่อว่า RS-232C และพอร์ตอนุกรม RS-232C นี้เป็นที่นิยมใช้งานมากที่สุด

2.3.1 รูปแบบของการสื่อสารแบบอนุกรม

การติดต่อแบบอนุกรมอาจจะแบ่งได้ 3 แบบ ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 รูปแบบการติดต่อสื่อสารข้อมูลแบบอนุกรม

1.แบบซิมเพล็กซ์ (simplex) ข้อมูลส่งได้ในทางเดียวเท่านั้น บางครั้งก็เรียกว่าการส่งทิศทางเดียว (Unidirectional data bus) ตัวอย่างการสื่อสารแบบนี้ได้แก่ การส่งกระจายเสียงวิทยุ โทรทัศน์(Radio & Television Broadcasting)

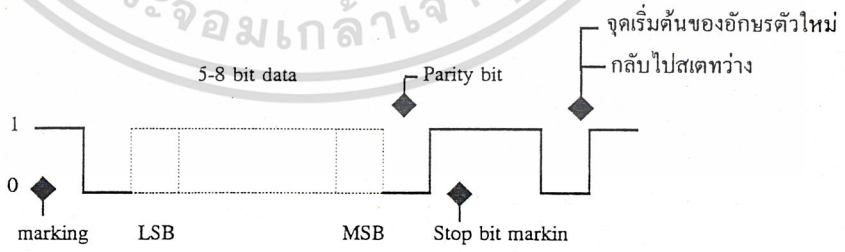
2.แบบฮาร์ฟดูเพล็กซ์(half duplex) ข้อมูลสามารถส่งได้ทั้งสองทาง แต่จะต้องผลัดกันส่งและผลัดกันรับ จะส่งและรับพร้อมกันไม่ได้ ตัวอย่างการสื่อสารแบบนี้ได้แก่ วิทยุสื่อสาร วิทยุสมัครเล่น

3.แบบฟูลดูเพล็กซ์(full duplex) ข้อมูลสามารถส่งได้ทั้งสองทาง และสามารถส่งได้พร้อมกันทั้งสองทิศทาง ตัวอย่างการสื่อสารแบบนี้ได้แก่ โทรศัพท์

การส่งข้อมูลแบบฟูลดูเพล็กซ์ และฮาร์ฟดูเพล็กซ์ ไม่ขึ้นอยู่กับจำนวนของสายในการติดต่อ บางครั้งคำว่า two wire หรือ 2 เส้น และ four wire หรือ 4 เส้น นั้นใช้ในการบรรยายถึงลักษณะการสื่อสาร

2.3.2 การรับส่งข้อมูลแบบอนุกรมอะซิงโครนัส

เป็นการสื่อสารที่พัฒนามาจากการส่งโทรพิมพ์ การสื่อสารแบบอะซิงโครนัสนั้นข้อมูลจะถูกส่งผ่านพอร์ตอนุกรมครั้งละ 1 บิท ซึ่งจะแตกต่างจากการส่งแบบขนานที่จะส่งครั้งละ 1 ไบต์เลย และระยะเวลาที่ใช้ในการส่งข้อมูลแบบอะซิงโครนัสในแต่ละไบต์นั้นไม่จำเป็นต้องเท่ากันเสมอจึงได้ชื่อว่าการรับส่งข้อมูลแบบอะซิงโครนัส ซึ่งมีลักษณะสัญญาณแสดงดังรูปที่ 2.16



รูปที่ 2.16 แสดงรูปแบบของข้อมูลแบบอะซิงโครนัส

ในการรับส่งข้อมูลผ่านพอร์ตแบบอนุกรมแบบอะซิงโครนัสนั้น ข้อมูลแต่ละไบต์ประกอบด้วย

1. บิทแสดงการเริ่มต้น(start bit)

2. บิทข้อมูล(data bit)

3. พาริตี้บิท(parity bit)

4. บิทแสดงการสิ้นสุด(stop bit)

ในขณะที่เครื่องไม่มีข้อมูลส่งออกมา เรียกว่า สถานะการส่งว่าง(idle) จะมีสัญญาณหรือแรงดันตลอดเวลา เพื่อความแน่ใจว่าฝ่ายรับยังติดต่อกับฝ่ายส่งอยู่ เมื่อเริ่มต้นส่งข้อมูล สัญญาณอะซิงโครนัสจะเป็น "0" หนึ่งช่วงสัญญาณนาฬิกา บิทนี้เรียกว่าบิทเริ่มต้น ต่อจากบิทเริ่มต้นก็เป็นข้อมูล 1 ตัวอักษร ซึ่งอาจจะมีตั้งแต่ 5-8 บิท โดยบิทต่ำ(LSB) จะถูกส่งออกก่อนไปจนถึงบิทสูงสุด(MSB) ต่อจากข้อมูลก็เป็นบิทพาริตี้ ซึ่งอาจจะใช้หรือไม่ก็ได้ โดยมีหน้าที่ตรวจสอบสัญญาณที่รับได้ อาจเป็นพาริตี้คู่ (Even Parity)หรือพาริตี้คี่ (Odd parity) ซึ่งบิทพาริตี้นี้ทางฝ่ายผู้ส่งจะต้องตรวจสอบข้อมูลแล้วใส่พาริตี้ เมื่อฝ่ายรับรับได้แล้วก็จะตรวจสอบว่าข้อมูลที่รับได้มีพาริตี้เป็นจริงตามที่ฝ่ายส่งบอกมาหรือไม่ ต่อจากบิทพาริตี้ก็เป็นบิทสิ้นสุด ซึ่งมีความกว้างเป็น 1,1.5,2 พัลส์(pulse) ของสัญญาณนาฬิกา แล้วแต่ผู้รับและผู้ส่ง ดังนั้นการเริ่มใช้พอร์ตอนุกรม จึงต้องตั้งค่าพารามิเตอร์ต่าง ๆ ซึ่งจำเป็นสำหรับการส่งข้อมูลแบบอนุกรม อันได้แก่

1. ความเร็วในการส่งข้อมูล

2. ความยาวรหัส 1 อักขระ

3. บิทพาริตี้

4. จำนวนบิทสิ้นสุด

การส่งแบบอะซิงโครนัสนี้มีลักษณะส่งไปทีละอักขระ จำนวนพัลส์ของสัญญาณที่ส่งออกมายังมีบางส่วนใช้ในการควบคุมการส่งได้แก่ บิทเริ่มต้น พาริตี้บิท และบิทสิ้นสุด ทำให้เกิดการส่งอักขระต่อวินาทีน้อยลงการส่งสัญญาณด้วยความเร็ว 300 baud สำหรับความยาวรหัส 7 บิทต่อ 1 อักขระ ไม่ได้หมายความว่าส่งได้ 300/7 อักขระต่อวินาที

การบอกความเร็วในการส่งข้อมูลแบบอนุกรม บอกได้ 2 แบบคือ

1.อัตราบิท(bit rate) คือ จำนวนบิทข้อมูลที่สามารถส่งไปได้ใน 1 วินาที หน่วย บิทต่อวินาที(bps)

2.อัตราบอด(baudrate) คือ จำนวนครั้งของการเปลี่ยนแปลงสัญญาณในเวลา 1 วินาที

อัตราบิท=อัตราบอด*บิทใน 1 บอด

2.3.3 การควบคุมการส่งเมื่อความเร็วในการทำงานของฝ่ายรับและฝ่ายส่งไม่เท่ากัน

เนื่องจากการใช้ภาษาในระดับสูงเขียนเป็นโปรแกรมควบคุมการทำงานของการถ่ายโอนข้อมูลแบบอนุกรม ฝ่ายรับอาจจะใช้เวลาในการรับข้อมูลเข้ามามากกว่าที่ทางฝ่ายส่งใช้ในการส่ง ดังนั้นเราจึงจำเป็นจะต้องมีวิธีการควบคุมไม่ให้เกิดการสูญหายของข้อมูลที่เราต้องการส่ง วิธีการควบคุมดังกล่าวมีอยู่หลายวิธีดังนี้

1. การมีบัฟเฟอร์ในการสื่อสารข้อมูล

บัฟเฟอร์สำหรับการสื่อสารก็คือหน่วยความจำในคอมพิวเตอร์ซึ่งแบ่งแยกออกมาจากหน่วยความจำหลัก สำหรับเก็บพักข้อมูลชั่วคราวระหว่างการติดต่อสื่อสาร บัฟเฟอร์สำหรับการสื่อสารนี้ส่วนมากใช้สำหรับฝ่ายรับเท่านั้น เนื่องจากฝ่ายรับจะต้องรับข้อมูลจากฝ่ายส่งให้ทัน การรับข้อมูลจะดำเนินไปจนกระทั่งบัฟเฟอร์เต็ม เมื่อบัฟเฟอร์เต็ม การรับข้อมูลจะหยุดลงจนกระทั่งข้อมูลในบัฟเฟอร์ถูกส่งหรือถูกอ่านออกไปจนหมด ในลักษณะเข้าก่อนออกก่อน (queue) ข้อมูลจึงจะถูกรับเข้ามาใส่ในบัฟเฟอร์อีกครั้ง

2. การควบคุมโดยใช้ XON/XOFF

ในบางครั้งการส่งถ่ายข้อมูลด้วยความเร็วสูงและขนาดของข้อมูลมีขนาดใหญ่ ซึ่งอาจจะมีบัฟเฟอร์ไม่พอ ถ้าหากบัฟเฟอร์เต็ม ก็จำเป็นต้องควบคุมการส่งข้อมูลโดยการบอกให้ฝ่ายส่งหยุดชั่วคราว(XOFF) จนกว่าฝ่ายรับจะจัดการเอาข้อมูลออกจากบัฟเฟอร์สื่อสารหมดเสียก่อน จึงบอกให้ฝ่ายส่ง ส่งข้อมูลต่อไป(XON)

3. การใช้โปรโตคอล(Protocol)

เทคนิคอีกอย่างหนึ่งในการควบคุมการส่ง คือการใช้โปรโตคอล เทคนิคนี้ทั้งฝ่ายรับและฝ่ายส่งจะต้องเหมือนกัน โดยการใช้อักขระควบคุมตามตารางระหัสแอสกี้ สำหรับการควบคุมการส่งข้อมูลออกมาเป็นกลุ่มที่มีขนาดคงที่ อักขระที่ใช้เป็นโปรโตคอล ในการควบคุมการส่งจากตารางแอสกี้ มีดังต่อไปนี้

ETB = END OF TRANSMISSION BLOCK

มีค่าเป็น 23 เป็นการบอกฝ่ายรับว่าขณะนี้สิ้นสุดการส่งข้อมูลกลุ่มหนึ่งแล้ว

ETX = END OF TEXT

มีค่าเป็น 03 เป็นการบอกฝ่ายรับว่ขณะนี้สิ้นสุดการส่งข้อมูลแล้ว

ENQ = ENQUIRY

มีค่าเป็น 05 เป็นอักขระที่ส่งมาจากฝ่ายรับบอกให้ฝ่ายส่ง ส่งข้อมูลได้

NACK = NEGATIVE ACKNOWLEDGE

มีค่าเป็น 21 เป็นการบอกฝ่ายส่งให้รู้ว่าข้อมูลที่รับมาผิดพลาด

ACK = ACKNOWLEDGE

มีค่าเป็น 06 เป็นการบอกฝ่ายส่งให้รู้ว่าข้อมูลได้รับถูกต้องแล้ว

2.3.4 พอร์ตมาตรฐาน RS-232C

พอร์ตอนุกรม RS-232C นี้เป็นฮาร์ดแวร์ที่ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรม ซึ่งเป็นที่นิยมใช้งานกันอย่างแพร่หลาย สำหรับเครื่องไมโครคอมพิวเตอร์ทุกเครื่องจะมีพอร์ตอนุกรม RS-232C นี้มากับเครื่องทุกเครื่อง มาตรฐานตัวนี้ย่อมาจาก

RS ย่อมาจาก Recommened Standard

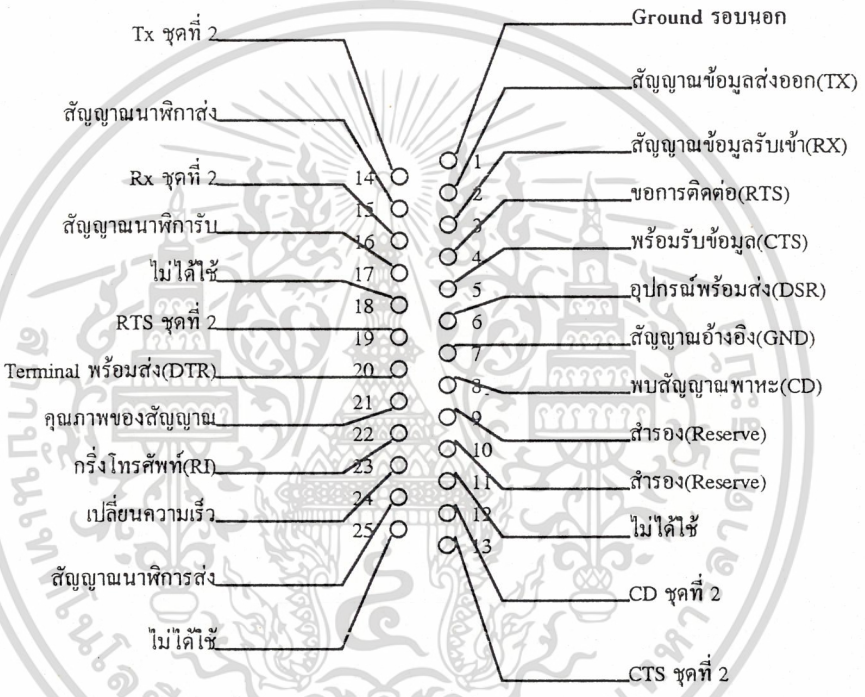
232 เป็นหมายเลขบ่งบอกถึงมาตรฐานตัวนี้

C เป็นการแก้ไขฉบับสุดท้ายของมาตรฐานนี้

จุดประสงค์ของมาตรฐานตัวนี้คือ เพื่อใช้บรรยายลักษณะการเชื่อมต่อระหว่างอุปกรณ์รับส่งข้อมูลปลายทาง (DTE) กับอุปกรณ์สื่อสารข้อมูล(DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DCE จะหมายถึงโมเด็ม,อุปกรณ์อื่น ๆ เช่น เครื่องพิมพ์ ในการเชื่อมต่อ RS-232C สามารถส่งถ่ายข้อมูลด้วยความเร็ว ตั้งแต่ 110-19200 บอด(baud) ความเร็วในการส่งถ่ายข้อมูลสูงสุดถึง 115200 บิทต่อวินาที ที่อัตราบอด 19200 ซึ่งเป็นความเร็วที่เพียงพอสำหรับเครื่องไมโครคอมพิวเตอร์ ความยาวของสายที่ใช้ในการเชื่อมต่อโดยมาตรฐานของ RS-232C ต้องยาวไม่เกิน 50 ฟุต

2.3.5 การกำหนดจุดต่อของ RS-232C

คอนเน็คเตอร์ซึ่งมาตรฐาน RS-232C กำหนดไว้จะเป็นแบบ DB-25 ลักษณะของคอนเน็คเตอร์พร้อมด้วยหน้าที่ของแต่ละขาของคอนเน็คเตอร์แสดงไว้ดังรูปที่ 2.17



รูปที่ 2.17 แสดงจุดต่อของมาตรฐาน RS-232C

อย่างไรก็ตามผู้ผลิตไมโครคอมพิวเตอร์อาจจะใช้ข้อต่อชนิดอื่น ๆ นอกเหนือจาก DB-25 นี้ ยกตัวอย่างเช่น Fujitsu F-8 หรือ IBM AT,IBM JR เป็นต้น ตัวเมียควรจะอยู่ที่ตัวโมเด็ม ขณะที่ตัวผู้ควรจะอยู่ที่คอมพิวเตอร์ (Asynchronous communication adupter) สำหรับในไมโครคอมพิวเตอร์จะใช้เพียงขาสัญญาณต่าง ๆแค่ 9 ขาเท่านั้น ดังจะกล่าวโดยละเอียดว่าสัญญาณต่าง ๆ แต่ละสัญญานมีหน้าที่อย่างไร

1. Protective ground (PG) ขาที่ 1

เป็นขาที่ต่อกับตัวถังของอุปกรณ์ซึ่งโดยปกติแล้วจะต่อกับกราวด์ภายนอก และต่ออยู่กับสายชิลด์ของสายเคเบิลเชื่อมต่อระหว่าง DTE กับ DCE เพื่อป้องกันไม่ให้ไฟฟ้าสถิตย์และสนามแม่เหล็กใด ๆ ที่เกิดขึ้นมารบกวนต่อการส่งสัญญาณผ่านสายสัญญาณ การต่อกับตัวถังนี้ควรจะต่อกับตัวถังเพียงด้านเดียวเท่านั้น

2. Transmit data (TD) ขาที่ 2

เป็นข้อมูลที่ส่งออกจาก DTE ไปยังโมเด็มหรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น ๆ หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานะของขานี้จะเป็น ลอจิก"1" หรือเทียบเท่ากับบิทแสดงการสิ้นสุด(stop bit)

3.Receive data (RD) ขาที่ 3

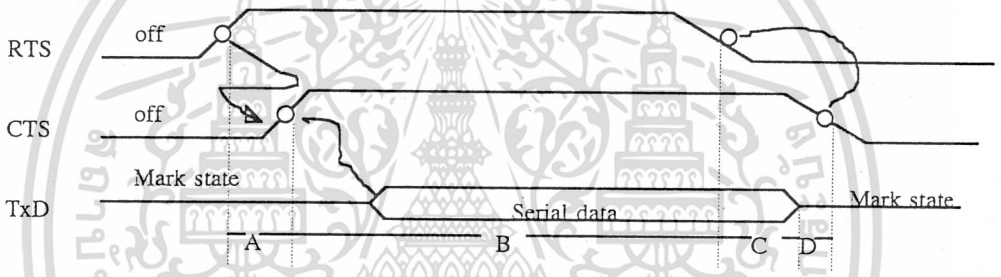
เป็นขาสัญญาณข้อมูลเข้าไปยังอุปกรณ์ DTE หรือไมโครคอมพิวเตอร์เมื่อไม่มีสัญญาณรับเข้ามาขานี้จะมีสถานะเป็น "1"

4. Request to send (RTS) ขาที่ 4

เป็นขาที่ใช้สำหรับส่งสัญญาณไม่ยังโมเด็มหรือเครื่องพิมพ์ เพื่อร้องขอที่จะส่งข้อมูลออกไปยังตัวรับ(ตัวส่งต้องการส่งข้อมูลทางขา 2) สัญญาณนี้จะใช้คู่กับสัญญาณ CTS(Clear to send) อุปกรณ์รับหากได้รับสัญญาณ RTS นี้ก็จะตรวจสอบว่าตัวเองพร้อมจะรับสัญญาณหรือไม่ ถ้าหากพร้อมก็ส่งสัญญาณออกไปที่ขา CTS

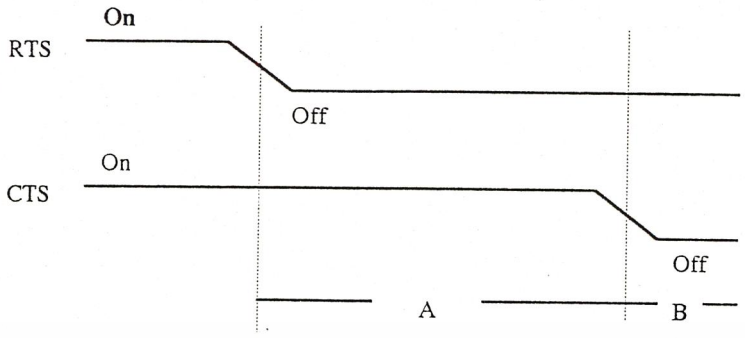
5.Clear to send (CTS) ขาที่ 5

ดังอธิบายไว้ใน สัญญาณ RTS เมื่อสัญญาณนี้เป็นลอจิก "1" หมายความว่า อุปกรณ์รับพร้อมที่จะรับข้อมูล



รูปที่ 2.18 แสดง Timing Diagram ของการรับส่งข้อมูลทางขา RTS และ CTS

จากรูปจะเห็นว่าการทำ handshake ของสัญญาณ RTS และ CTS ในช่วง *A* DTE จะป้อนสัญญาณ RTS แสดงให้ DCE ทราบว่า DTE ต้องการจะส่งข้อมูลซึ่งจะเกิดขั้นตอนเหล่านี้คือ DCE จะจัดตั้งช่องทางการสื่อสารและป้อนสัญญาณ CTS-ON ซึ่งแสดงให้ DTE รู้ว่าเริ่มส่งข้อมูลได้แล้ว แต่ TXD จะยังอยู่ในสถาวะ Mark อยู่ในช่วง *B* ข้อมูลจะถูกส่งผ่านทางวงจร Transmitted data เมื่อข้อมูลถูกส่งออกไปจนหมดแล้ว DTE จะออฟสัญญาณ RTS เพื่อบอกให้DCE รู้ว่า DTE ไม่ต้องการส่งข้อมูลอีกต่อไปแล้ว ในช่วง *C* เมื่อ DTE ส่งข้อมูลทั้งหมดออกไปที่ communication Link เสร็จแล้ว วงจร TXD จะกลับเข้าสู่สภาวะ Mark ในช่วง *D* DCE แจ้งให้ DTE ทราบว่า DCE พร้อมแล้วที่จะรับข้อมูลชุดใหม่เพื่อส่งออกไปโดยการออฟสัญญาณCTS



รูปที่ 2.19 Timing Diagram ที่แสดงความสัมพันธ์ของ RTS และ CTS

จากรูปจะเห็นว่าในช่วง A DTE ไม่สามารถ ON สัญญาณ RTS ใหม่อีกครั้งหนึ่งได้DTE ต้องรอจนกว่า DCE ส่งข้อมูลที่เหลือออกไปจนหมดก่อน โดย DCE จะปิดสัญญาณ CTS เพื่อแสดงให้ DTE รู้ว่ามันพร้อมที่จะรับข้อมูลชุดใหม่แล้ว ในช่วง *B* DCE สามารถ ONสัญญาณ RTS ใหม่เมื่อใดก็ได้เนื่องจาก CTS มีสภาวะเป็น OFF การทำเช่นนี้เป็นการป้องกันไม่ให้เกิด overrun error ขึ้น (overrun error คือการที่ DTE ทำการส่งข้อมูลชุดใหม่มาอีกในขณะที่ DCE ยังส่งข้อมูลชุดเก่าไม่เสร็จ)

6.Data set ready (DSR) ขาที่ 6

เมื่อสัญญาณสายนี้อยู่ในสถานออน(ลอจิก "0") เป็นการบอกไมโครคอมพิวเตอร์ หรือฝ่ายส่งว่าโมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้ว และพร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณผ่านสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่าต่อกับโทรศัพท์เรียบร้อยแล้ว

7.Signal Ground (SG) ขาที่ 7

Signal Ground ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุก ๆ สายของสัญญาณ จะมีแรงดันเป็น"0" เมื่อเทียบกับสัญญาณตัวอื่น

8.Carrier detect (CD) ขาที่ 8

โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน(ลอจิก "0") ไปบอกไมโครคอมพิวเตอร์เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุด LED ให้สว่าง(ซึ่งจะอยู่ที่หน้าปัดของโมเด็ม) เพื่อแสดงว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้ว

9.Data terminal ready (DTR) ขาที่ 20

คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน เมื่อพร้อมที่จะติดต่อกับโมเด็ม

10.Ring indicator (RI) ขาที่ 22

สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบรับอัตโนมัติ สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งเข้ามา และออฟระหว่างเสียงดังของกระดิ่ง

ระดับสัญญาณของ RS-232C

สัญญาณที่ขาทุกขาที่คอนเนคเตอร์ของ RS-232C จะเป็นสภาวะใดสภาวะหนึ่ง ในแต่ละคู่ของคู่ต่อไปนี้

Mark / space

on / off

logic 0 / logic 1

2.4 ระบบฐานข้อมูล (DATABASE)

ระบบจัดการฐานข้อมูลได้ทวีความสำคัญขึ้นอย่างรวดเร็ว สำหรับรูปของข้อมูลมีหลากหลาย เช่น รายการบัญชี ข้อมูลของนักศึกษาข้อมูลการลงทะเบียนของนักศึกษาแต่ละคน ลักษณะของระบบฐานข้อมูลจะเป็นการนำเอาข้อมูล ซึ่งเป็นวัตถุดิบมาจัดเก็บให้เป็นระเบียบ มีการจัดเรียงลำดับแยกประเภท จัดให้อยู่ในรูปแบบที่สามารถนำไปใช้งานได้อย่างสะดวก

2.4.1 การออกแบบระบบฐานข้อมูล

การออกแบบระบบฐานข้อมูลจะจำแนกออกได้เป็น 2 วิธีใหญ่ ๆ คือ

1.วิธีการอุปนัย(Bottom-up หรือ Inductive approach) เป็นการออกแบบสร้างฐานข้อมูลโดยอาศัยวิธีการรวบรวมข้อมูล หรือ โปรแกรมที่มีอยู่แล้วจากหลาย ๆ หน่วยงานแนวคิดพื้นฐานของการออกแบบประเภทนี้ก็คือ ถือว่าลักษณะงานของแต่ละหน่วยงานมีความซับซ้อน สมบูรณ์แตกต่างกัน ดังนั้น รูปแบบของฐานข้อมูลที่ดีจึงต้องเป็นรูปแบบที่รวบรวมเอาข้อดีของโปรแกรมทั้งหมดที่มีอยู่ในหน่วยงาน มารวบรวมเป็นโปรแกรมขนาดใหญ่ แต่ข้อเสียของวิธีการนี้ก็คือการจะรวมเอาวิธีการย่อย ๆ เข้าด้วยกันทำได้ไม่ง่ายนัก

2.วิธีการนิรภัย(Top-down หรือ deductive approach) คือเลือกเอาผู้ที่เข้าใจระบบทั้งหมด ศึกษาว่าองค์การมีข้อมูลอะไรบ้าง ต้องการอะไรบ้าง แล้วจึงนำมาออกแบบเป็นโครงสร้างทั้งหมดของระบบฐานข้อมูล ซึ่งวิธีการนี้เป็นวิธีการที่เหมาะสมสำหรับการจัดวางระบบที่มีความยุ่งยากซับซ้อน

2.4.2 ขั้นตอนการออกแบบระบบฐานข้อมูล

การออกแบบระบบฐานข้อมูลจะประกอบด้วยขั้นตอน 4 ลำดับดังต่อไปนี้

1.รวบรวมข้อมูล ผู้ออกแบบต้องเข้าใจว่าในระบบจะมีข้อมูลอะไรอยู่บ้าง และต้องผลลัพธ์หรือเอาท์พุทแบบใดบ้าง

2.ทำความเข้าใจกับข้อมูลและการประยุกต์ใช้งาน โดยสรุปผู้ออกแบบจะต้องพยายามทำความเข้าใจเกี่ยวกับประเด็นต่าง ๆ ดังต่อไปนี้

- แหล่งข้อมูล จะไปรวบรวมข้อมูลจากที่ใด
- ข้อมูลเหล่านี้จะคงค่าไว้นานเท่าใดจึงจะมีการลบทิ้ง
- ข้อมูลแต่ละชุดมีความสำคัญกับข้อมูลอื่นอย่างไร
- ข้อมูลมีโอกาสถูกเปลี่ยนแปลงมากน้อยแค่ไหน
- มีกฎเกณฑ์ในการตรวจสอบความถูกต้องของข้อมูลได้อย่างไร

3.กำหนดรูปแบบของระบบขั้นต้น หลังจากที่ได้ข้อสรุปในเรื่องของฟิลด์และไฟล์(table) ที่จะใช้ภายในระบบ ทีมงานจะต้องกำหนดร่างของระบบคร่าว ๆ ว่าภายในระบบจะมีการส่งผ่านข้อมูล และแสดงผลเป็นเอาท์พุทในลักษณะใด

4.วิเคราะห์และลงมือสร้างระบบที่ต้องการ หลังจากที่ได้ดูรูปแบบของระบบขั้นต้นแล้ว จะต้องศึกษาอย่างละเอียดว่าจะต้องมีการจดการอย่างไรตามลำดับขั้นตอน แล้วจึงวางรูปแบบของระบบในรายละเอียดเป็นขั้นตอนสุดท้าย

2.4.3 โครงสร้างของหน่วยจัดเก็บข้อมูล

1.เรคอร์ด(record) เป็นหน่วยหลักที่กำหนดการจัดเก็บข้อมูลให้เป็นหมวดหมู่ หน่วยที่ใช้จัดเก็บเป็นเรคอร์ดจะเป็นตัวนักศึกษาแต่ละคน หรือ หน่วยงานหนึ่ง ๆ แล้วแต่ระบบฐานข้อมูล

2.ฟิลด์(field or data item) เป็นหน่วยของข้อมูลย่อยแต่ละตัวที่จัดเก็บไว้ภายในเรคอร์ด เช่น ชื่อ นามสกุล ชั้นปี อายุ วันทำบัตร

3.แวลลู(Value) คือค่าของข้อมูลในแต่ละฟิลด์ที่ถูกจัดเก็บอยู่ในแต่ละเรคอร์ด

ในการสร้างไฟล์ฐานข้อมูลเพื่อเป็นหน่วยจัดเก็บข้อมูล ผู้ออกแบบระบบอาจกำหนดโครงสร้างของการจัดรูปแบบแรคอร์ด และฟิลด์ในไฟล์ได้แตกต่างกันในหลายลักษณะแต่อย่างไรก็ตาม รูปแบบของการจัดดังกล่าวนี้ อาจแบ่งได้เป็น 2 ประเภทใหญ่ ๆ ด้วยกัน

1.record-based model(group items) จัดเก็บชุดข้อมูลที่สัมพันธ์กันในลักษณะที่เป็นฟิลด์ต่าง ๆ ภายในแต่ละเรคอร์ดของไฟล์ อาจจัดเก็บเป็นไฟล์ใหญ่เดี่ยว 1 ไฟล์(single file) หรือ เป็นไฟล์ย่อยที่สัมพันธ์กัน (multifile structure) ก็ได้

Single file เป็นไฟล์เดี่ยวที่มีความเป็นอิสระในตัวเองไม่เกี่ยวข้องกับไฟล์อื่น ๆ ในระบบฐานข้อมูลทั้งหมด(ไม่สัมพันธ์กันโดยระบบ แต่ผู้ใช้ระบบอาจโยงผลลัพธ์ที่ได้จากการประมวลผลแต่ละไฟล์เข้าด้วยกันได้) วิธีการจัดเก็บข้อมูลใน single file ยังสามารถจำแนกได้หลายประเภทเช่น

-single flat file

-composit flat file

-single hierachical file(1 path)

-single multipath("branching")

Multifile structure เป็นการจัดสร้างไฟล์ข้อมูลหลายไฟล์ โดยกำหนดความสัมพันธ์ระหว่างไฟล์เอาไว้ ระบบสามารถจะเชื่อมโยงไฟล์ต่าง ๆ เข้าด้วยกันได้เมื่อผู้ใช้สั่งการให้เชื่อมโยงไฟล์เข้าด้วยกัน ระบบของฐานข้อมูลส่วนใหญ่ จะมีลักษณะเป็นแบบ multifile structure สำหรับไฟล์ย่อยนั้นอาจจะเป็น flat file หรือ hierachical ก็ได้

2.Object-relation strucure model(ungroup items) เป็นการจัดเก็บข้อมูลในลักษณะเชิงความสัมพันธ์ ไม่กำหนดเป็นรูปแบบความสัมพันธ์ของฟิลด์และเรคอร์ดที่สมบูรณ์ เช่นเดียวกับแบบที่ 1 แต่จะเป็นการจัดที่ตั้งอยู่บนพื้น

ฐานแนวความคิดในการเชื่อมโยงข้อมูลแต่ละคู่ที่สัมพันธ์กันพยายามทำให้เสมือนกับเป็นวิธีธรรมชาติที่คนมองเห็นคู่ความสัมพันธ์ของข้อมูลเป็นชุด ๆ การจัดโครงสร้างในลักษณะนี้ไม่เป็นที่นิยมแพร่หลายนัก

2.4.4 วิธีการจัดการไฟล์ฐานข้อมูล

หลังจากที่แฟ้มข้อมูลถูกสร้างขึ้นมาแล้ว เราจะต้องสามารถดึงเอาข้อมูลที่มีอยู่ในเรคอร์ดต่าง ๆ ออกมาใช้งาน วิธีการที่จะเข้าถึง(access) ข้อมูลที่ต้องการ มีด้วยกันหลายวิธีดังนี้

1.Sequential processing เป็นการประมวลเรคอร์ดต่าง ๆ ที่อยู่ในไฟล์ที่ละเรคอร์ดไปตามลำดับของเรคอร์ดที่ถูกจัดเก็บอยู่ภายในไฟล์ วิธีการที่จะเข้าถึงเรคอร์ดจะอาศัยหลักการว่า "ไปที่เรคอร์ดถัดไป" ดังนั้น ถ้าต้องการจะดูข้อมูลที่มีอยู่ในเรคอร์ดที่4 ก็จะต้องเริ่มอ่านข้อมูลที่อยู่ในเรคอร์ดที่ 1 ในไฟล์ก่อน แล้วจึงอ่านไปเรื่อย ๆ ตามลำดับจนกว่าจะถึงเรคอร์ดที่ต้องการ แล้วจึงแสดงผลขึ้นมา

2.Sorted order processing เป็นการจัดเรียงลำดับเรคอร์ดเสียใหม่ตามเงื่อนไขที่ต้องการ นั่นคือต้องการเปลี่ยนจากไฟล์ของเรคอร์ดที่จัดลำดับตามแบบเดิม และสร้างเป็นไฟล์ใหม่เกิดขึ้นมา โดยที่ภายในไฟล์ใหม่จะมีเรคอร์ดที่จัดเรียงลำดับของเรคอร์ดเสียใหม่ตามรูปแบบที่ต้องการ และจะยังคงไฟล์ที่จัดแบบเดิมเอาไว้หรือจะลบทิ้งเสียก็ได้

3.direct access/relative file structure การเก็บเรคอร์ดไว้ในไฟล์ฐานข้อมูลบตแผ่นดิสก์จะทำให้เราสามารถเข้าถึเรคอร์ดได้โดยตรง จึงเรียกว่าเป็น direct access หรือบางทีจะเรียก random access เพราะจะสุ่มกระโดดไปที่เรคอร์ดใดก็ได้ในไฟล์ โดยไม่ต้องเรียงไปตามลำดับตั้งแต่เรคอร์ดที่ 1 เป็นต้นไป โดยปกติซอฟต์แวร์ที่จัดสร้างแบะจัดการไฟล์ฐานขอ้มูลจะใช้วิธีการนี้ เราจึงสามารถสั่งให้ไปยังตำแหน่งของเรคอร์ดหนึ่ง ๆ ได้โดยตรง

4.Index structure ลำดับที่ของเรคอร์ดในไฟล์เป็นตำแหน่งที่เรคอร์ดนั้น ๆ เรียงอยู่ในไฟล์ฐานข้อมูลจริง เทคนิคการเรียงลำดับเรคอร์ดแบบวิธีการ sort จะเป็นการจดเรียงลำดับที่เรคอร์ดใหม่ คือเปลี่ยน physical record order เสียใหม่ทั้งหมดแต่ยังมีอีกวิธีหนึ่งที่เราจะยังคงตำแหน่งลำดับที่ของเรคอร์ดเดิมอยู่โดยจะกำหนดดัชนี(index)เป็นตัวควบคุมชี้ตำแหน่งของเรคอร์ดตามต้องการ เรียกว่าเป็น logical record order นั่นคือเราไม่ได้เอาเรคอร์ดมาสลับที่ใหม่จริง ๆ แต่เราจะกำหนดตัวชี้เพื่อชี้กำหนดวิธีเรียงลำดับตามรูปแบบที่ต้องการ โดยที่ยังคงไฟล์ฐานข้อมูลเดิม ที่มี physical recordorder เดิมอยู่ได้

ข้อดีของการใช้ดัชนีมี 2 ประการด้วยกันคือ

(1) จะช่วยให้เราสามารถใช้วิธีการเข้าถึงข้อมูลแบบ direct access ได้ และ

(2) จะทำให้เราสามารถเข้าถึงข้อมูลตามลำดับเงื่อนไขใด ๆ ก็ได้ โดยไม่ต้องเป็นการเข้าถึงตามเงื่อนไขหมายเลขของเรคอร์ดอย่างเช่นวิธีการแบบrelative file structure

2.4.5 เทคนิคการพัฒนาโปรแกรม

ขั้นที่ 1 วางแผนแบ่งส่วนย่อยของระบบ

วิธีออกแบบระบบโปรแกรมที่นิยมใช้กันมากที่สุดวิธีหนึ่งคือ วิธีการแก้ปัญหาแบบ top-down design ซึ่งเป็นวิธีการแบ่งปัญหาทั้งหมดออกเป็นปัญหาย่อย ๆ และเมื่อออกแบบรายละเอียดการแก้ปัญหาในแต่ละส่วนย่อย ๆ แล้ว

ก็จะสามารถแก้ไขปัญหาใหญ่ทั้งหมดได้ เพื่อที่จะทำให้มองเห็นโครงสร้างของปัญหาทั้งหมด ว่าประกอบขึ้นมาจากปัญหาย่อย ๆ อะไรบ้างผู้ออกแบบมักจะวางแผนผังของระบบทั้งหมดว่าประกอบขึ้นด้วยส่วนย่อย ๆ อะไรบ้าง (block diagram of the system)

ขั้นที่ 2 วางแผนแบ่งส่วนย่อยของโปรแกรม

เมื่อออกแบบผังการทำงานออกเป็นส่วนได้ตามที่ต้องการแล้ว ขั้นตอนถัดไปคือ การวางแผนเขียนโปรแกรมโดยใช้เทคนิคของการเขียนโปรแกรมแบบ modular programming ซึ่งเป็นวิธีการแบ่งโปรแกรมออกเป็นหน่วยย่อย ๆ (module) เทคนิคนี้มักจะกำหนดให้เขียนเป็นตัวโปรแกรมได้ไม่เกิน 2 ถึง 3 หน้าต่อหนึ่งหน่วย จุดมุ่งหมายหลักของการเขียนโปรแกรมในรูปของหน่วยย่อย ๆ แบ่งได้เป็น 2 ประเด็น คือ

1.เพื่อเป็นการแบ่งแยกโครงสร้างของโปรแกรมที่ซับซ้อนให้เป็นหน่วยย่อย ๆ และจัดให้เป็นระเบียบ (decomposition of programming tasks) ซึ่งจะเป็นการช่วยให้ผู้เขียนโปรแกรมเขียนโปรแกรม แลทดสอบส่วนต่าง ๆ ของโปรแกรมได้สะดวกขึ้น โปรแกรมย่อยในลักษณะนี้มักจะเรียกว่าเป็น subprogram หรือ subroutines คือเป็นส่วนของโปรแกรมที่ทำหน้าที่เฉพาะอย่าง

2.เพื่อใช้กับส่วนของโปรแกรมที่ต้องถูกเรียกใช้งานซ้ำกันบ่อย ๆ และ เป็นส่วนที่สามารถนำไปใช้งานร่วมกันระหว่างโปรแกรมต่าง ๆ ได้ (reusable program elements) โปรแกรมย่อยในลักษณะนี้มักจะเรียกกันว่าเป็น procedural module คือเป็นโปรแกรมย่อย ๆ ที่นำไปใช้ประกอบการทำงานของโปรแกรมอื่น ๆ ตัวอย่างเช่น โปรแกรมจัดรูปแบบหน้าจอ

หลักการพื้นฐานที่ผู้ออกแบบเขียนโปรแกรมจะยึดเป็นหลักปฏิบัติในการออกแบบ modular programming มี 2 ประการคือ

1. โปรแกรมย่อยแต่ละโปรแกรมจะเป็นอิสระด้วยตัวเอง ทำงานเฉพาะอย่าง อย่างใดอย่างหนึ่ง และมีความสมบูรณ์ในตัวเอง

2. โปรแกรมย่อยแต่ละโปรแกรมควรจะมีจุดเข้า และจุดออกจากโปรแกรมเพียงจุดเดียว ไม่ควรมีการหยุดโปรแกรมตรงกลางโปรแกรมแล้วให้อีกโปรแกรมหนึ่งมารับงานต่อ โดยที่ยังทำโปรแกรมเดิมนั้นไม่เสร็จ

ขั้นที่ 3 เขียน flow chart

หลังจากที่ได้กำหนดส่วนย่อยต่าง ๆของโปรแกรม โดยแบ่งเป็น module ต่าง ๆ ตามที่ผู้ออกแบบโปรแกรมเห็นสมควรแล้ว ก็จะมาออกแบบในรายละเอียด โดยการเขียน flow chart แสดงขั้นตอนว่าจะต้องทำอะไรก่อนหลัง

ขั้นที่ 4 ลงมือเขียนโปรแกรม

ลงมือเขียนโปรแกรมตาม flowchart ที่กำหนดไว้ วิธีการเขียนโปรแกรมที่นิยมกันก็คือเขียนจากล่างขึ้นบน (Bottom Up) คือเขียนโปรแกรมย่อยที่มีความเป็นอิสระไม่ขึ้นกับโปรแกรมอื่นก่อน จากนั้นจึงค่อยเขียนโปรแกรมที่เรียกใช้โปรแกรมย่อยนั้นขึ้นไปเรื่อย ๆ

ข้อควรปฏิบัติในการเขียนโปรแกรมก็คือ ควรจะมีคำอธิบายตัวแปรและการทำงานส่วนต่าง ๆ ในโปรแกรมไว้ด้วย

บทที่ 3

การสร้างระบบการให้บริการคอมพิวเตอร์

3.1 การโปรแกรมใช้งานพอร์ตอนุกรมและฐานข้อมูล

3.1.1 โปรแกรมใช้งานพอร์ตอนุกรมในการสื่อสารข้อมูล

ในการโปรแกรมใช้งานพอร์ตอนุกรมในการสื่อสารนั้นอาจจะทำได้ 2 วิธีคือ

1.ใช้ฟังก์ชันที่มีให้ใช้อยู่แล้วในระบบปฏิบัติการ(OS)

2.โดยการเขียนโปรแกรมควบคุม hardware โดยตรง

3.1.1.1 การเขียนโปรแกรมใช้งานพอร์ตอนุกรมโดยใช้ฟังก์ชันที่มีอยู่แล้วในระบบปฏิบัติการ

ระบบปฏิบัติการ(OS) จะมีฟังก์ชันเพื่อสนับสนุนการใช้งานพอร์ตอนุกรมอยู่ 2 แบบ คือฟังก์ชันของ DOS และ ฟังก์ชันของ BIOS

ก.ฟังก์ชันของ DOS

DOS จะมีฟังก์ชันเพื่อสนับสนุนการสื่อสารผ่านพอร์ตอนุกรมอยู่ 2 ฟังก์ชัน ซึ่งเราสามารถเรียกใช้งานได้โดยการเรียกใช้ int 21H ฟังก์ชัน 3 และ 4 โดยเซตค่าในรีจิสเตอร์ AH ให้เป็น 3 หรือ 4

ฟังก์ชัน 3 จะทำหน้าที่อ่านอักขระจาก Asynchronous port ค่าที่อ่านเข้ามาจะถูกเก็บอยู่ในรีจิสเตอร์ AL

ฟังก์ชัน 4 ทำหน้าที่ส่งอักขระในรีจิสเตอร์ DL ไปยัง Asynchronous port

สำหรับข้อผิดพลาดที่เกิดขึ้นจะรายงานกลับมา ซึ่งมีดังต่อไปนี้

บิท

15 Time out — รอจนหมดเวลา

14 Transmit shift register — เมื่อรีจิสเตอร์ข้อมูลว่างลง จะมีสถานะ empty เป็น "1" และจะมีสถานะเป็น "0" เมื่อมีการถ่ายข้อมูลจากรีจิสเตอร์เก็บรักษาข้อมูลไปยังรีจิสเตอร์ข้อมูลส่ง

13 Transmit holding register — เมื่อรีจิสเตอร์รักษาข้อมูลส่งว่างลงจะempty มีค่าเป็น "1"

12 Break detect — เป็นบิทที่บอกว่าสายข้อมูลรับมีสถานะเป็น"0"นานเกินไปกว่าข้อมูล1 ตัว

11 Framing error — เป็นบิทที่บอกว่า บิทสิ้นสุดที่รับได้นั้นไม่ถูกต้อง(ไม่เป็น "0")

10 Parity error — เป็นบิทที่บอกว่า ข้อมูลที่รับได้มีพาริตี้ผิด

9 Overrun error — เป็นบิทที่บอกว่าคอมพิวเตอร์ไม่ได้อ่านข้อมูลตัวที่แล้วไปจากรีจิสเตอร์บัฟเฟอร์

8 Data ready — เป็นบิทที่บอกว่าพร้อมจะรับข้อมูลแล้ว

7 Receive line signal detect — บิทนี้จะมีสถานะตรงข้ามกับขา RLSD

6 Ring indicator — บิทนี้จะมีสถานะตรงข้ามกับขา RI

5 Data set ready — บิทนี้จะมีสถานะตรงข้ามกับขา DSR

4 Clear to send — บิทนี้จะมีสถานะตรงข้ามกับขา CTS

3 Delta receive line signal — เป็นบิทที่บอกว่า RLSD เปลี่ยนสถานะ detect แล้ว

2 Trailing edge ring — เป็นบิทที่บอกว่า RI เปลี่ยนสถานะแล้ว indicator

1	Delta data set ready	เป็นบิทที่บอกว่า DSR เปลี่ยนสถานะแล้ว
0	Delta clear to send	เป็นบิทที่บอกว่า CTS เปลี่ยนสถานะแล้ว

ข.ฟังก์ชัน BIOS

bios มีฟังก์ชันเพื่อสนับสนุนการใช้งานพอร์ตอนุกรมซึ่งสามารถทำให้เราใช้งานพอร์ตอนุกรมได้อย่างมีประสิทธิภาพกว่าการใช้ฟังก์ชันของ DOS ซึ่งเราสามารถเรียกใช้ได้โดยเรียก อินเทอร์รัพต์ 14H หมายเลขฟังก์ชันจะระบุในรีจีสเตอร์ AH ฟังก์ชันต่าง ๆอธิบายได้ดังนี้

ฟังก์ชัน 1 เป็นการส่งอักขระในรีจีสเตอร์ AL ไปในสายสื่อสารจะรายงานสถานะมาในรีจีสเตอร์ AH ถ้าหากการส่งทำได้สมบรูณ์ บิทที่ 7 จะเป็น "1" บิทอื่นจะแสดงสถานะของสายสื่อสาร

ฟังก์ชัน 2 เป็นการอ่านอักขระข้อมูลจากพอร์ตอนุกรม ข้อมูลที่อ่านเข้ามาจะถูกนำไปเก็บที่รีจีสเตอร์ AL และจะรายงานข้อผิดพลาดมาในรีจีสเตอร์ AH

ฟังก์ชัน 3 เป็นการตรวจสอบสถานะของพอร์ตอนุกรม โดยจะรายงานผลมาในรีจีสเตอร์ AH ความหมายแต่ละบิทเป็นดังนี้

บิท	7	=	Timeout
บิท	6	=	Transfer shift register empty
บิท	5	=	Transfer register empty
บิท	4	=	Break detect
บิท	3	=	Framing error
บิท	2	=	Parity error
บิท	1	=	Overrun error
บิท	0	=	Data ready

3.1.1.2 การเขียนโปรแกรมควบคุม hardware โดยตรง

ในกรณีที่ต้องการจะควบคุมการทำงานของ Asynchronous Adapter โดยตรงสิ่งที่เราจำเป็นจะต้องรู้คือการทำงานของ 8250 และตำแหน่งที่อยู่ของรีจิสเตอร์ต่าง ๆ ของ 8250 ซึ่งตำแหน่งที่อยู่ของรีจีสเตอร์ใน 8250 และการควบคุมเป็นดังนี้

3F8 TX Buffer	Dlab = 0 (write)
3F8 RX Buffer	Dlab = 0 (read)
3F8 Divisor Latch LSB	Dlab = 1
3F9 Divisor Latch MSB	Dlab = 1
3F9 Interupt Enable Register	Dlab = 0
3FA Interupt Identification Regiser	
3FB Line Control Register	
3FC Modem Control Register	
3FD Line Status Register	
3FE Modem Status Register	

ตำแหน่งที่อยู่บางตัวจะใช้หลายหน้าที่ ซึ่งควบคุมโดยสถานภาพของ FLIP-FLOP ตัวหนึ่งซึ่งมีชื่อว่า Dlab (Divisor Latch Address Bit) ซึ่งอยู่ในการควบคุมของ Line Control Regiser ซึ่งสามารถสรุปหน้าที่ควบคุมรีจีสเตอร์ต่าง ๆ ได้ดังนี้

1.รีจีสเตอร์บัฟเฟอร์สำหรับตัวรับข้อมูล (RBR-Receiver Buffer Regiser)

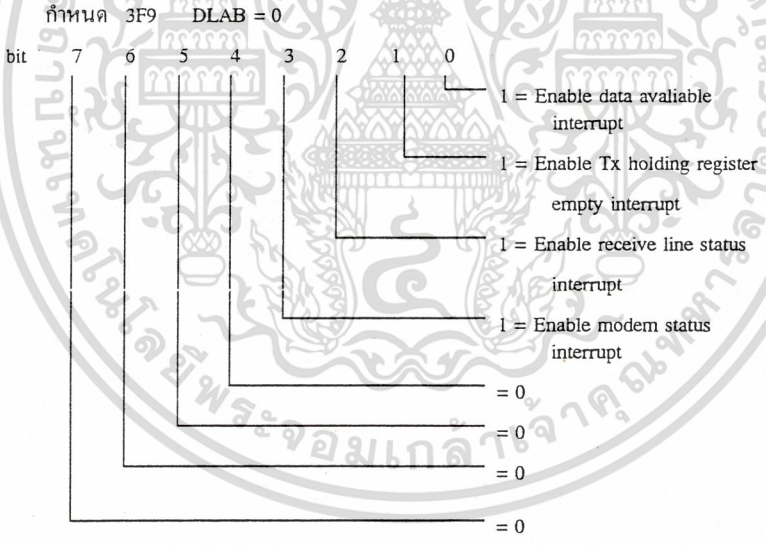
เป็นรีจีสเตอร์ที่บรรจุอักขระที่รับมา จากสายการสื่อสารสัญญาณพอร์ตที่กำหนดคือหมายเลขแอดเดรส 3F8 ขณะที่ Dlab=0 หาก CPU อ่านข้อมูลที่รีจีสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

2.รีจีสเตอร์โฮลดิ้งสำหรับตัวส่งข้อมูล (THR-Transmitter Holding Register)

เป็นรีจีสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจีสเตอร์นี้จะรับขอ้มูลจาก CPU ดดยกำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ Dlab=0 ข้อมูลที่ CPU เอาท์พุทมาที่พอร์ตนี้ก็เพื่อจะส่งต่อออกไปยังสายสื่อสารข้อมูล

3.รีจีสเตอร์อนุญาตอินเทอร์รัพต์ (IER-Interrupt Enable Register)

ใน COM1 เมื่อ Dlab=0 พอร์ต 3F9 จะเป็นรีจีสเตอร์อนุญาตอินเทอร์รัพต์ผู้ใช้สามารถกำหนดให้เกิดอินเทอร์รัพต์หรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจีสเตอร์นี้ การอินเทอร์รัพต์ของ 8250 นี้มี 4 แบบ ดังนั้นจึงต้องกำหนดการอนุญาตได้ทั้ง 4 แบบ ดังนั้น โดยการใช้ข้อมูลแต่ละบิทของรีจีสเตอร์นี้เพื่อการกำหนดการอนุญาต ข้อมูลที่อยู่ในรีจีสเตอร์นี้มีความหมายดังนี้



bit 0 บิทนี้ควบคุมให้มีการอินเทอร์รัพต์ จากความพร้อมของข้อมูลฝ่ายรับ เมื่อบิท 0 มีค่าเป็น "1"

bit 1 บิทนี้ควบคุมให้มีการอินเทอร์รัพท์จากการที่ Transmitter holding register ว่างลง เมื่อบิท 1 มีค่าเป็น "1"

bit 2 บิทนี้ควบคุมให้มีการอินเทอร์รัพต์ จากสถานะของสายรับ เมื่อบิทนี้มีค่าเป็น "1"

bit 3 บิทนี้ควบคุมให้มีการอินเทอร์รัพต์ จากสถานะของโมเด็ม เมื่อบิท 3 นี้มีค่าเป็น "1"

4.การโปรแกรมอัตราบอด (Programmable Baud Rate Generator)

อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1 - (2^{16}–1) ค่าความถี่ที่ออกจากตัวกำหนดอัตราเร็วจะมีค่าเป็น 16 เท่าของอัตราเร็วในการรับส่งข้อมูล ดังนั้น

ตัวหาร =ความถี่สัญญาณนาฬิกา/(อัตราบอด*16)

การกำหนดอัตราบอดด้วยการกำหนดตัวหารนี้ตัวหารจึงเป็นค่าที่กำหนดในรีจิสเตอร์ 2 ตัว ตัวหารนี้จะถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ Dlab=1 แล้วให้กำหนดลงมาในรีจิสเตอร์ 3F8 ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F9 เมื่อ Dlab=1 จะมีค่าเป็นตัวหารของ MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตาราง

5.รีจิสเตอร์กำหนดอินเทอร์รัพต์ (IIR-Interrupt Identification Register)

การอินเทอร์รัพต์ของ 8250 ทำให้สามารถเชื่อมโยง 8250 เข้ากับไมโครคอมโพรเซสเซอร์ที่ใช้อยู่โดยทั่วไปได้ซึ่งระดับการอินเทอร์รัพต์ของ 8250 ถูกจัดเป็น 4 ระดับตามความสำคัญจากสาเหตุต่าง ๆ คือ

ระดับที่ 1 จากสถานะสายตัวรับ

ระดับที่ 2 จากความพร้อมของข้อมูล

ระดับที่ 3 จากรีจิสเตอร์รักษาข้อมูลส่งว่าง

ระดับที่ 4 จากสถานะโมเด็ม

ในขณะที่มีความต้องอินเทอร์รัพต์หลายระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่ารอไว้ก่อน โดยเก็บสถานะการอินเทอร์รัพต์นี้ไว้ในรีจิสเตอร์กำหนดอินเทอร์รัพต์

ความหมายของรีจิสเตอร์มีดังนี้

บิท 0 เป็นบิทที่ใช้แสดงว่ามีอินเทอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถได้ซีพียู ตรวจสอบดูด้วยวิธีการ polling ได้ ถ้าบิทนี้เป็น "1" หมายถึง ไม่มีการอินเทอร์รัพต์เกิดขึ้น

บิท 1-2 เป็นบิทที่แสดงความหมายบอกว่าการอินเทอร์รัพต์ที่เกิดขึ้นนั้นมาจาก อินเทอร์รัพต์ฟังก์ชันใด

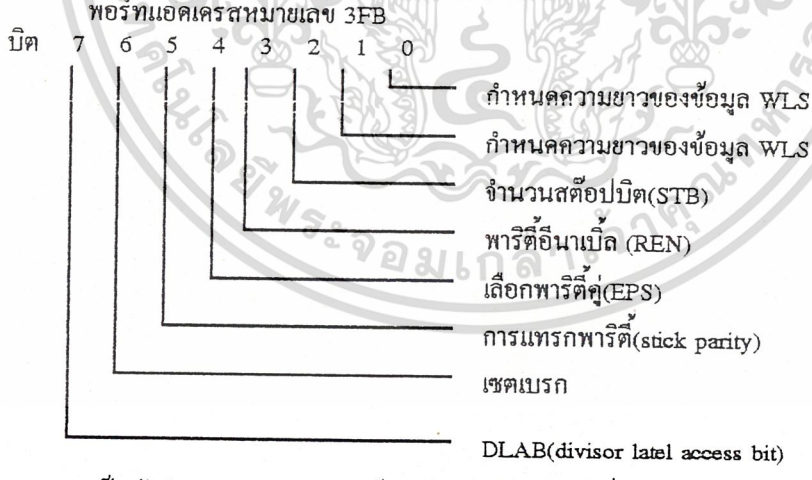
บิท 3-7 มีค่าเป็น "0"

ตารางที่ 3.1 แสดงค่าตัวหารสำหรับการกำหนด baud rate

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	60	-
1800	64	40	-
2000	58	3A	0.69
2400	48	30	-
3600	32	20	-
4800	24	18	-
7200	16	10	-
9600	12	0C	-

6.รีจีสเตอร์ควบคุมสายสื่อสาร (LCR-Line Control Register)

ในการควบคุมรูปแบบของข้อมูลแบบอะซิงโครนัสนั้น ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจีสเตอร์ควบคุมสายสื่อสาร รีจีสเตอร์ตัวนี้มี 8 บิท โดยแต่ละบิทมีความหมายดังนี้



บิท 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	ความหมาย
0	0	หมายถึงข้อมูลขนาด 5 บิต
0	1	หมายถึงข้อมูลขนาด 6 บิต
1	0	หมายถึงข้อมูลขนาด 7 บิต
1	1	หมายถึงข้อมูลขนาด 8 บิต

บิท 2 เป็นบิทที่ใช้ในการกำหนดจำนวนบิทสิ้นสุด ถ้าเป็น "0" หมายถึง ใช้บิทสิ้นสุด 1 บิท ถ้าเป็น "1" ถ้าในกรณีการส่งแบบ 5 บิทจะมี ความยาวบิทสิ้นสุดเป็น 1.5 บิท แต่ถ้าเป็นการส่งแบบ 6,7,8 บิท ความยาวของบิทสิ้นสุดจะเป็น 2

บิท 3 บิทนี้เป็นบิทแสดงการอนุญาตให้มีการตรวจสอบพาริตี้ โดยถ้าบิทนี้มีค่าเป็น "1" จะมีการเพิ่มพาริตี้

บิท 4 ถ้าบิทนี้มีค่าเป็น "0" และ บิท 3 มีค่าเป็น "1" จะหมายถึงการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิทนี้มีค่าเป็น "1" จะเป็นพาริตี้คู่

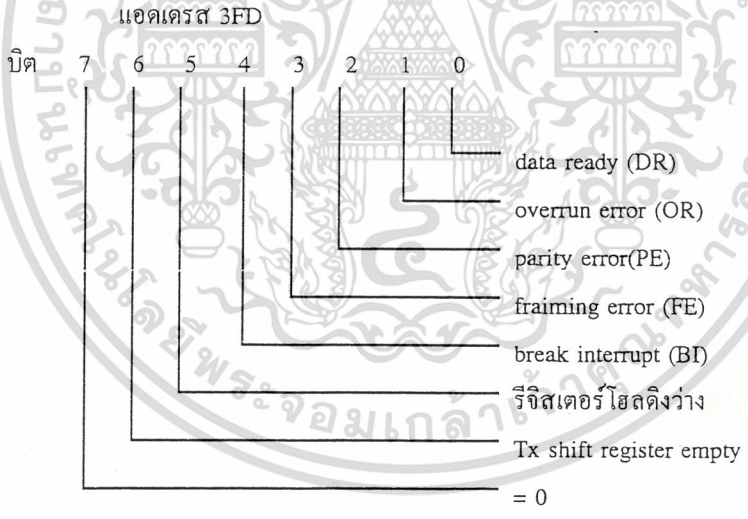
บิท 5 เมื่อบิทที่ 3 มีค่าเป็น "1" และบิท 5 มีค่าเป็น "1"และบิท 4 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี้(stick parity) ด้วยเงื่อนไขกำหนดให้เป็น "0" และ ถ้าบิท 4 มีค่าเป็น "0" และ บิท 3มีค่าเป็น "1" และ บิท 5 มีค่าเป็น "1" จะมีการกำหนดพาริตี้เป็น 1

บิท 6 เป็นบิทที่ควบคุมการเบรก เมื่อบิท 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด

บิท 7 บิทนี้ทำหน้าที่เป็น Dlab เป็นบิทที่มีผลต่อการแลตช์ตัวหาร

7.รีจีสเตอร์แสดงสถานะสายสื่อสาร (LSR-Line Status Register)

รีจีสเตอร์ตัวนี้ เป็นรีจีสเตอร์ที่จะให้ข้อมูลแก่ CPU ที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบิทต่าง ๆ ในรีจีสเตอร์นี้เป็นดังนี้



บิท 0 บิทนี้เป็นบิทที่บอกสถานะการรับข้อมูล ถ้าบิทนี้เป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิทแล้ว บิทนี้จะได้รับการรีเซตให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้วหรือจะให้ซีพียูเขียนข้อมูลกลับมายังรีจีสเตอร์นี้ก็ได้

บิท 1 บิทนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด overrun error(OR) กล่าวคือ ขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไป ปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับบนบัฟเฟอร์นี้ บิทนี้จะรีเซตโดยซีพียูเมื่อซีพียูอ่านค่าจากรีจีสเตอร์นี้ไปแล้ว

บิท 2 บิทนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด parity error(PE) กล่าวคือ ถ้ามีการตรวจสอบบิทพาริตี้แล้ว ไม่เป็นตามที่กำหนดไว้ บิทนี้จะได้รับการเซตโดยซีพียู เมื่อซีพียูอ่านค่าจากรีจีสเตอร์ไปแล้ว

บิท 3 บิทนี้มีค่าเป็น "1" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด

บิท 4 บิทนี้เรียกว่า break interrupt (BI) บิทนี้จะได้รับการเซตให้มีค่าเป็น "1" ถ้าหากว่ารับข้อมูลอิน พุตเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิท 5 บิทนี้เป็นบิทที่บอกว่า 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสาร บิทนี้จะได้รับการเซตให้มีค่าเป็น "1" บิทนี้ยังคงสร้างสัญญาณอินเทอร์รัพต์เพื่อส่งไปบอกซีพียูด้วย บิทนี้จะมีสถานะเซตเมื่อมีการ ส่งถ่ายข้อมูลจากโฮลรีจีสเตอร์ไปยังชิฟรีจีสเตอร์ส่ง

บิท 6 เป็นบิทที่บอกว่าชิฟรีจีสเตอร์ว่างเปล่า บิทนี้ว่างเปล่า บิทนี้ได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิท 7 จะเป็น "0" ตลอด

3.1.1.3 ขั้นตอนการทำฮาร์ดแวร์อินเทอร์รัพต์

ไอบีเอ็มพีซีมีสัญญาณการอินเทอร์รัพต์ที่สามารถเข้าไปขัดจังหวะการทำงานของซีพียูได้ 8 ช่องทาง โดยใช้ไอซี 8259 สำหรับควบคุมอินเทอร์รัพต์ดัง นี้

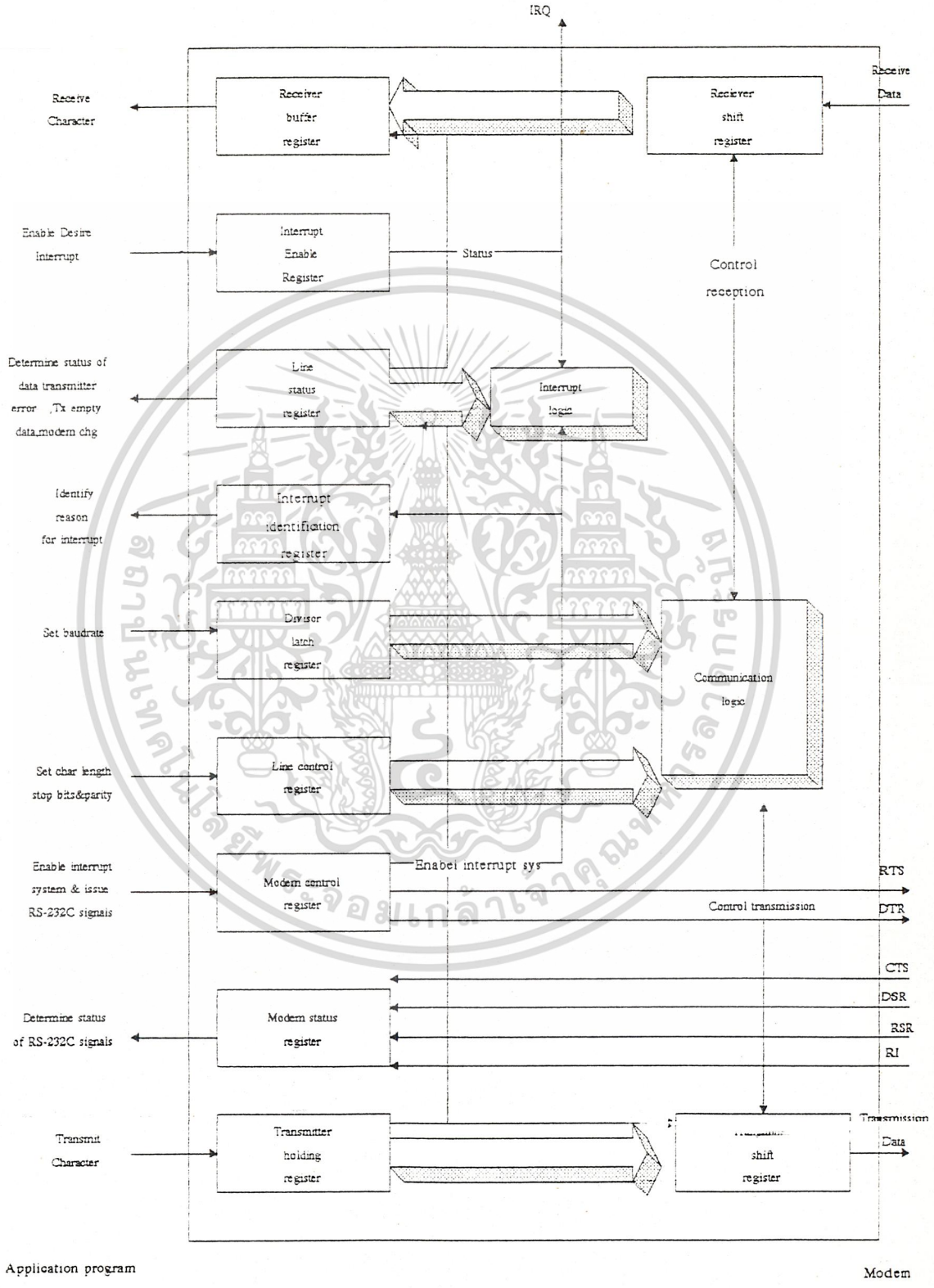
IRQ 0 สำหรับการอินเทอร์รัพต์ Timer

IRQ 1 สำหรับการอินเทอร์รัพต์ Keyboard

IRQ 2 สำรองไว้

IRQ 3 สำหรับพอร์ตอนุกรมหมายเลข 1

IRQ 4 สำหรับพอร์ตอนุกรมหมายเลข 0



รูปที่ 3.1 แสดงการส่งผ่านข้อมูลและสัญญาณต่างออกไปที่พอร์ตอนุกรม

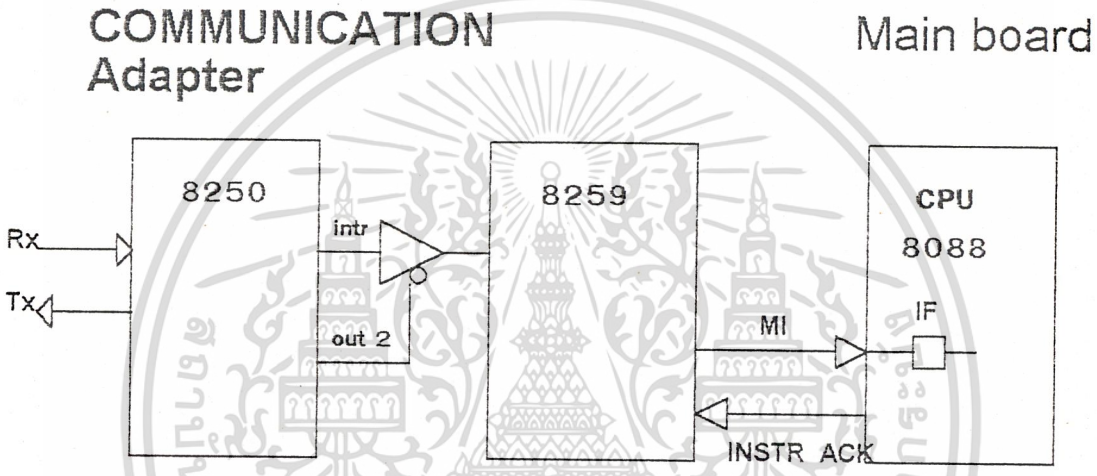
IRQ 5 สำหรับดิสเกตส์คอนโทรลเลอร์

IRQ 6 สำหรับดิสเกตส์คอนโทรลเลอร์

IRQ 7 สำรองไว้

สัญญาณการอินเทอร์รัพต์ทั้ง 8 จะทำให้เกิดการอินเทอร์รัพต์ที่ 8 บวกด้วยหมายเลขของ IRQ ดังนั้น การอินเทอร์รัพต์จาก COM1 ซึ่งผ่านทาง IRQ 4 จึงทำให้เกิด Interrupt Vectorที่ 0CH

ขั้นตอนของการยอมให้สัญญาณการขัดจังหวะผ่านเข้าไปถึงซีพียูแสดงได้ดังรูปที่ 2.21



รูปที่ 3.2 แสดงขั้นตอนการไหลของสัญญาณอินเทอร์รัพต์จากพอร์ตอนุกรม

- พอร์ต XF9 กำหนดชนิดของการเกิดการอินเทอร์รัพต์
- พอร์ต XFC เปิด OUT2 ให้สัญญาณการอินเทอร์รัพต์(INT)ส่งต่อไปที่ 8259
- พอร์ต 21H Interrupt Mask ยอมให้สัญญาณ IRQ 4 ส่งต่อไปที่ 8088 ถ้าเซตบิทที่ 4 เป็น 0
- พอร์ต 20H บอกว่า การอินเทอร์รัพต์ IRQ 4 กำลังทำงานอยู่หรือไม่
- I Flag(interrupt flag) สำหรับรับหรือไม่รับสัญญาณการขัดจังหวะ คำสั่งในการเซตและรีเซตแฟลกคือ STI และ CLI

3.1.1.4 ขั้นตอนการขอการอินเทอร์รัพต์จากพอร์ตอนุกรมอธิบายได้ดังนี้

1.เมื่อมีเหตุการณ์เกิดขึ้นที่ 8250 ตามที่เรากำหนด เช่น ตั้งค่าไว้ที่ XF9 เป็น 01สัญญาณ Int จะออกมาจาก 8250 เมื่อ 8250 รับข้อมูลครบ 1 อักขระ

2.สัญญาณ Int ที่ออกมาจาก 8250 จะถูกควบคุมด้วยเกตตัวหนึ่ง ซึ่งสามารถควบคุมได้โดยพอร์ต XFC บิทที่ 3 (สัญญาณ OUT 2)

3.เมื่อ 8259 รับสัญญาณ IRQ 4 จะตรวจสอบดู IMR(Interrupt Mask Register) ว่าจะรับหรือไม่ 8259 จะรับสัญญาณ IRQ ก็ต่อเมื่อบิทที่ตรงกับหมายเลขของ IRQ ใน IMR เป็น "0" เท่านั้น จะเข้าไปควบคุม IMR ได้โดยช่องทางพอร์ต 21H

4.ถ้า 8259 รับสัญญาณ IRQ ก็จะตรวจสอบลำดับความสำคัญของสัญญาณ IRQ ว่าขณะนี้ซีพียูกำลังทำงานให้กับการอินเทอร์รัพต์ไหนบ้าง โดยตรวจสอบดูที่ ISR(Interrupt Service Register) ถ้าซีพียูยังไม่ได้ให้บริการการอินเทอร์รัพต์ใด ๆ ที่สำคัญกว่า 8259 จะส่งสัญญาณการอินเทอร์รัพต์ไปยัง 8088 ผ่านทาง Maskable Interrupt(MI)

5.ทุก ๆคำสั่งของ 8088 จากการปฏิบัติแล้วจะตรวจดูสัญญาณการขัดจังหวะที่ขา NMI(Non Maskable Interrupt) และ MI สำหรับสัญญาณ MI จะมีผลให้ 8088 ให้ความสนใจก็ต่อเมื่ออินเทอร์รัพต์แฟลกเป็น 1

6.เมื่อ 8088 รับสัญญาณ MI จะส่งสัญญาณไปบอก 8259 ว่า ตกลงรับการร้องขออินเทอร์รัพต์ (Interrupt Acknowledge) กลับไปยัง 8259

7.8259 จะเอาหมายเลขทิศทางการอินเทอร์รัพต์ (ในที่นี้เท่ากับ 0CH) ส่งให้ 8088 ทาง data bus พร้อมกับตั้งค่า ISR บิทที่ตรงกับหมายเลขสัญญาณ IRQ ให้เป็น 1 เพื่อเป็นการเตือนตัวเองว่าซีพียูกำลังให้บริการการอินเทอร์รัพต์ไหนอยู่

8.8088 รับเอาหมายเลขทิศทางการอินเทอร์รัพต์(Interrupt Vector) จาก data bus แล้วจะเก็บค่าภาวะแฟลกและเซกเมนต์ของโปรแกรมและโปรแกรมเคาน์เตอร์ขณะนั้นลงในสแตกซ์ พร้อมกับกระโดดไปทำงานยังอินเทอร์รัพต์รูทีนซึ่งชี้โดย Interrupt Vector Table(ในที่นี้คือในหน่วยความจำ ตำแหน่งที่ 0000:0030H) พร้อมกับเคลียร์ IF flag ให้เป็น "0"ไม่ยอมให้เกิด Maskable Interrupt อีกต่อไป

9.เป็นหน้าที่ของอินเทอร์รัพต์รูทีนที่จะต้องจัดการต่อไป

3.1.1.5 เทคนิคการเขียนอินเทอร์รัพต์รูทีนสำหรับการใช้งานพอร์ตอนุกรม

1.ตั้งพารามิเตอร์ต่าง ๆ สำหรับการสื่อสาร

2.กำหนดชนิดของการเกิดสัญญาณการขัดจังหวะใน 8250

3.กำหนด Vector table สำหรับการอินเทอร์รัพต์ของพอร์ตอนุกรม

(ที่ 0CH สำหรับ IRQ 4 และ 0BH สำหรับ IRQ 3)

4.ตั้งค่า IMR ใน 8259 โดยผ่านทางพอร์ต 21H

5.เปิดอินเทอร์รัพต์เกตใน 8250 โดยให้บิทที่ 3 ของ MCR เป็น 1

6.ต้องไม่ลืมเอาอินเทอร์รัพต์เวคเตอร์เก่ากลับคืนมาเมื่อเราเอาอินเทอร์รัพต์รูทีนออกจากหน่วยความจำหรือเลิก ใช้โปรแกรมนี้แล้ว

ตัวรูทีนที่ให้บริการสำหรับการอินเทอร์รัพต์สำหรับพอร์ตอนุกรม ควรจะมีลักษณะดังนี้

1.เซตค่า IF flag ให้เป็น 1 กลับคืนถ้าหากต้องการอินเทอร์รัพต์ด้วยเหตุใด ๆ ในกรณีที่เราตั้ง อินเทอร์รัพต์ตัวอื่นที่มีความสำคัญกว่า

2.เก็บค่ารีจิสเตอร์ต่าง ๆ ที่ต้องการใช้ลงในสแต็กให้หมด

3.อ่านดูรีจิสเตอร์ IIR ใน 8250 ว่าเป็นการเกิดการอินเทอร์รัพต์ด้วยเหตุใด ในกรณีที่เราตั้งค่าให้เกิดอินเทอร์รัพต์หลายแบบ

4.อ่านอินพุทมาจาก 8250 และจัดการตามขบวนการ

5.เอาค่ารีจิสเตอร์ต่าง ๆที่เก็บไว้ในสแต็กกลับคืนมา

6.ส่งเอาต์พุตไปบอก 8259 ว่า สิ้นสุดการทำงานการขัดจังหวะที่ขอมาแล้ว โดยส่งไปที่พอร์ต 20H ด้วยค่า 20H บวกหมายเลข IRQ ที่ 8259 รับเข้ามา เช่นถ้าเป็น IRQ4 ก็ให้ส่งค่า 24H ออกไปที่พอร์ต 20H

7.คำสั่ง IRET กระโดดกลับไปที่โปรแกรมหลัก

3.1.1.6 โปรแกรมใช้งานพอร์ตอนุกรมภาษาซี

ก.โปรแกรมส่วนอินเทอร์รัพต์

โปรแกรมส่วนนี้จะอ่านข้อมูลซึ่งถูกส่งเข้ามาทางพอร์ตอนุกรมเข้าไปเก็บในบัฟเฟอร์เพื่อรอให้ข้อมูลถูกอ่านไปใช้งาน ถ้าหากบัฟเฟอร์เต็มข้อมูลเดิมจะถูกทับ

```
void interrupt com1(void)
{
 CommBuf[ CommDataPos++ ] = inportb( CommAddr );
 if ( CommDataPos == CommBufSize )
        CommDataPos = 0;
 outportb( 0x20 , EOI );
}
```

ข.โปรแกรมส่วนติดตั้งพอร์ตอนุกรม

โปรแกรมส่วนนี้จะทำการติดตั้งพอร์ตอนุกรม โดยจะตั้งค่าพารามิเตอร์ หมายเลขพอร์ตที่จะใช้งาน, อัตราบอด,จำนวนบิตข้อมูลต่อ 1 ตัวอักษร,จำนวนบิทสิ้นสุด,พาริตี้บิทเป็นแบบใด โดยจะทำการเก็บสถานะของรีจิสเตอร์ต่าง ๆ ก่อนจะทำการติดตั้ง

```
int InitSerialComm(int comno,int baudrate,int databit,int stopbit,int parity)
{
 char low_divisor,high_divisor,comm_spec;
 if ( installed )
        return( E_INSTALLED );
 if ( comno != COM1 && comno != COM2 )
        return( E_COMN );
 if ( baudrate < B1200 || baudrate > B19200 )
        return( E_BAUD );
 if ( databit != D7 && databit != D8 )
        return( E_DATABIT );
 if ( stopbit != S1 && stopbit != S2 )
        return( E_STOPBIT );
 if ( parity < PNONE || parity > PEVEN )
        return( E_PARITY );
 switch( comno )
```

```
{
        case COM1 : CommAddr = COM1PORT; break;
        case COM2 : CommAddr = COM2PORT; break;
}
CommBuf = (char *) malloc(CommBufSize);
if ( CommBuf == NULL )
        return( E_MEMORY );
low_divisor = baud[baudrate];
high_divisor = 0;
comm_spec = Data[databit] | stop[stopbit] | pari[parity];
outportb( CommAddr + LCR , comm_spec | 0x80 );
outportb( CommAddr + DLL , low_divisor      );
outportb( CommAddr + DLH , high_divisor     );
outportb( CommAddr + LCR , comm_spec & 0x7F );
outportb( CommAddr + IER , 1                ); /* enable int : Rx Ready */
outportb( CommAddr + MCR , 0x0B             ); /* OUT2(3),RTS(1) */
inportb( CommAddr + LSR );
inportb( CommAddr + MSR );
inportb( CommAddr + IIR );
switch( comno )
{
        case COM1 :
        {       old1 = getvect( COM1INT );
                setvect( COM1INT , com1 );
                outportb( IMR , inportb(IMR) & (~IRQ4FLAG) );
        } break;
        case COM2 :
        {       old2 = getvect( COM2INT );
                setvect( COM2INT , com2 );
                outportb( IMR , inportb(IMR) & (~IRQ3FLAG) );
        } break;
}
installed = comno;
return( PASS );
}
```

ค.โปรแกรมส่วนยกเลิกใช้งานพอร์ตอนุกรม

โปรแกรมส่วนนี้จะทำการคืนค่าสถานะของรีจีสเตอร์ต่าง ๆ ก่อนที่เราจะทำการติดตั้งโปรแกรมใช้งานพอร์ตอนุกรมของเรากลับคืนให้แก่ระบบ

```
void QuitSerialComm(void)
{
 if ( ! installed )   return;
 switch( installed )
 {
        case COM1 :
        {
                setvect( COM1INT , old1 );
                outportb( IMR , inportb(IMR) | IRQ4FLAG );
        } break;
        case COM2 :
        {
                setvect( COM1INT , old2 );
                outportb( IMR , inportb(IMR) | IRQ3FLAG );
        } break;
 }
 free(CommBuf);
}
```

ง.โปรแกรมส่งข้อมูลออกไปยังพอร์ตอนุกรม

โปรแกรมนี้จะส่งข้อมูลออกไปทางพอร์ตอนุกรมทีละ 1 ตัวอักษร

```
int Transmit(unsigned char ch)
{
 int i;
 for ( i=0; i<TXCOUNT; i++ )
        if ( inportb( CommAddr + LSR ) & 0x20 ) /* THR empty? */
        {
                outportb( CommAddr , ch );
                return( PASS );
        }
 return(FAIL); /* TXCOUNT too small or Serial Comm. Port error */
}
```

3.1.2 โครงสร้างของหน่วยจัดเก็บข้อมูล

ในระบบจะมีฐานข้อมูลทั้งหมด 3 ชุด ดังนี้

1. ฐานข้อมูลที่เก็บข้อมูลของนักศึกษาที่ทำบัตรเพื่อขอใช้เครื่องคอมพิวเตอร์ มีโครงสร้างดังนี้

ก. ชื่อนักศึกษา

ข. รหัสนักศึกษา

ค. คณะ

ง. ชั้นปี

จ.วันที่เริ่มทำบัตร และ วันหมดอายุ

2.ฐานข้อมูลที่เก็บข้อมูลของนักศึกษาที่ลงทะเบียนเรียนในวิชาที่จองเวลาขอใช้เครื่องคอมพิวเตอร์ซึ่งกรณีนี้ไม่จำเป็นต้องมีบัตรก็สามารถใช้บริการได้ มีโครงสร้างดังนี้

ก. รหัสนักศึกษา

ข. รหัสวิชา

3. ฐานข้อมูลที่เก็บข้อมูลการจองเวลาขอใช้ห้องคอมพิวเตอร์สำหรับใช้เป็นห้องเรียน มีโครงสร้างดังนี้

ก. ชื่อวิชา

ข. รหัสวิชา

ค. ชื่ออาจารย์ผู้สอน

ง. วันที่จะจอง หรือ วันที่เรียน

จ. ช่วงเวลาที่ต้องการจอง

ช. จำนวนนักศึกษา โดยรายชื่อนักศึกษาจะเก็บในฐานข้อมูลตามข้อ 2

3.1.3 การเรียงข้อมูลด้วยวิธี Quick sort

สำหรับวิธีในการเรียงข้อมูลมีหลายวิธี ได้แก่ Selection sort,Bubble sort,Merge sort,Heap sort,Quick sort,Radix sort การจะเลือกใช้อัลกอริทึมใด ก็ขึ้นอยู่กับปัจจัยต่าง ๆ ดังนี้

1.จำนวนข้อมูลในการเรียงข้อมูลถ้าหากข้อมูลมีไม่มากเราก็อาจจะใช้วิธีง่าย เช่น selection sort หรือ bubble sort แต่ถ้าหากข้อมูลมีจำนวนมาก 800-1000 ตัวขึ้นไปก็ควรจะใช้ quick sort

2.ขนาดของข้อมูล ซึ่งจะมีผลในการบรรจุลงบนหน่วยความจำ ซึ่งถ้าหากข้อมูลมีขนาดใหญ่เราก็ควรจะใชพ้อยน์เตอร์ในการจัดเรียงให้แทน

3.ลักษณะของข้อมูลก่อนการจัดเรียง ถ้าหากเป็นข้อมูลแบบสุ่มก็ควรจะเลือกใช้วิธี quick sort แต่ถ้าหากข้อมูลอยู่ในลักษณะเรียงตรงกันข้ามก็ควรจะเลือกใช้วิธี heap sort

ในที่นี้จะกล่าวถึงเฉพาะวิธี Quick sort ซึ่งในโครงงานนี้เลือกใช้ ด้วยเหตุผล 2 ประการ คือ

1.ข้อมูลนักศึกษาที่ได้รับเข้ามามีลักษณะสุ่มไม่มีรูปแบบการเรียงลำดับที่แน่นอน รหัสนักศึกษาจะเป็นอะไรก็ได้ตามลำดับของผู้มาสมัครเป็นสมาชิก ไม่ได้เรียงลำดับตามคณะ หรือว่าชั้นปี

2.ข้อมูลนักศึกษามีจำนวนมากการเลือกใช้วิธี Quick sort จะสามารถเรียงข้อมูลได้อย่างรวดเร็ว

หลักการอย่างง่าย ๆ ของวิธีการเรียงข้อมูลแบบ Quick sort คือ ใช้เรคอร์ดหรือคีย์พิเศษตัวหนึ่งเป็นตัวแบ่ง ข้อมูลออกเป็น 2 ส่วน ส่วนแรกจะมีค่ามากกว่าคีย์พิเศษจะอยู่ทางขวา และอีกส่วนจะมีค่าน้อยกว่าคีย์พิเศษ

จะอยู่ทางซ้ายจากนั้นในแต่ละส่วนก็จะถูกแบ่งออกเป็น 2 ส่วนย่อยเช่นนี้อีก ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งไม่สามารถแบ่งข้อมูลต่อไปได้อีก ก็จะได้ข้อมูลที่เรียงเสร็จเรียบร้อย

อัลกอริทึมสำหรับการเขียนโปรแกรมเป็นดังนี้

1.แบ่งตารางออกเป็น 2 ส่วนย่อยโดยมีคีย์พิเศษเป็นตัวแบ่ง

2.เรียกใช้ตัวเองกับส่วนย่อยที่อยู่ทางขวา

3.เรียกใช้ตัวเองกับส่วนย่อยที่อยู่ทางซ้าย

เวลาโดยเฉลี่ยที่จะถูกใช้โดยวิธี Quick sort เท่ากับ Olog(nxlog,n) แต่ถ้าข้อมูลอยู่ในลักษณะที่แย่ที่สุดจะใช้เวลามากที่สุดเท่ากับ $\frac{n^2}{2}$ เมื่อ n คือ จำนวนข้อมูล

3.1.3.1 โปรแกรมการเรียงข้อมูลโดยใช้วิธี Quick sort

โปรแกรมนี้จะเรียงข้อมูลที่อยู่ในดิสก์โดยใช้วิธี Quick sort โดยใช้อัลกอริทึมตามที่ได้กล่าวมาแล้วในหัวข้อ 3.1.3 การเรียงข้อมูลจะเรียงตามรหัสนักศึกษา

ก.โปรแกรมหลักทำหน้าที่เรียงข้อมูล

```
void QuickDisk(FILE *fp,unsigned long int TotalRecord)
{
        Qsdisk(fp,0L,TotalRecord-1);/* count-1 is a number of record begin at '0'*/
}
void Qsdisk(FILE *fp,unsigned long int left,unsigned long int right)
{
        unsigned long int i,j;
        char Id[9];

        i=left;     j=right;
        /*get the middle StudentId*/
        strcpy(Id,GetStudentId(fp,(long)(i+j)/2));
        do
        {
                while( strcmp(GetStudentId(fp,i),Id)<0 && i<right ) i++;
                while( strcmp(GetStudentId(fp,j),Id)>0 && j>left )  j--;
                if(i<=j)
                {
                        Swapfield(fp,i,j);
                        i++;        j--;
                }
        } while(i<=j);
```

```
        if(left<j)  Qsdisk(fp,left,j);
        if(i<right) Qsdisk(fp,i,right);
    }
```

ข.โปรแกรมทำหน้าที่สลับข้อมูลระหว่างเรคอร์ดที่น้อยกว่ากับเรคอร์ดที่มากกว่า

```
    void Swapfield(FILE *fp,unsigned long int i,unsigned long int j)
    {
        char a[sizeof(StudentRecord_t)],b[sizeof(StudentRecord_t)];
        /*first read in record i and j*/
        fseek(fp,13L+sizeof(StudentRecord_t)*i,SEEK_SET);
        fread((char *)a,sizeof(StudentRecord_t),1,fp);
        fseek(fp,13L+sizeof(StudentRecord_t)*j,SEEK_SET);
        fread((char *)b,sizeof(StudentRecord_t),1,fp);
        /*then write them back in opposite slots*/
        fseek(fp,13L+sizeof(StudentRecord_t)*i,SEEK_SET);
        fwrite((char *)b,sizeof(StudentRecord_t),1,fp);
        fseek(fp,13L+sizeof(StudentRecord_t)*j,SEEK_SET);
        fwrite((char *)a,sizeof(StudentRecord_t),1,fp);
    }
```

ค.โปรแกรมทำหน้าที่อ่านข้อมูลรหัสนักศึกษาขึ้นมาจากไฟล์

```
    char *GetStudentId(FILE *fp,unsigned long int record)
    {
        StudentRecord_t *p;
        p=&StudentRecord;
        fseek(fp,13L+record*sizeof(StudentRecord),SEEK_SET);
        fread(p,sizeof(StudentRecord),1,fp);
        return StudentRecord.StudentId;
    }
```

3.1.4 การค้นหาข้อมูล

3.1.4.1 การค้นหาข้อมูลแบบซีเควนเชียล

เป็นเทคนิคพื้นฐานที่ง่ายที่สุดในการค้นหาข้อมูล โดยจะอ่านค่าจากเรคอร์ดแล้วทำการเปรียบเทียบข้อมูลนั้นว่าตรงกับข้อมูลที่ต้องการหาหรือไม่ โดยจะทำการไล่ไปทีละ 1 ข้อมูลอย่างมีลำดับจนกว่าจะพบข้อมูลตามที่ต้องการ

3.1.4.2 การค้นหาข้อมูลแบบไบนารีทรี

การค้นหาข้อมูลวิธีนี้เราจะนำข้อมูลที่เรียงแล้วมาทำการแบ่งออกเป็น 2 ส่วน แล้วเอาค่าที่อยู่ตรงกลางนั้นมาเปรียบเทียบกับข้อมูลที่ต้องการ ถ้าหากค่าที่อยู่ตรงกลางมีค่ามากกว่าก็แสดงว่าข้อมูลที่ต้องการจะต้องอยู่ก่อนค่ากลางนี้(ครึ่งแรกนั่นเอง) จากนั้นเราจะนำข้อมูลในครึ่งแรกมาแบ่งเป็น 2 ส่วนอีก และนำค่ากลางมาเปรียบเทียบอีก ทำเช่นนี้เรื่อยไป เราก็จะค้นหาข้อมูลที่เราต้องการได้ถ้าหากมีข้อมูลอยู่ในตารางนั้น ๆ

อัลกอริทึมสำหรับเขียนโปรแกรมเป็นดังนี้

1. หาตำแหน่งตรงกลางของข้อมูลทั้งหมด
2. เปรียบเทียบข้อมูลที่ต้องการค้นหากับข้อมูลที่อยู่ตรงกลางข้อมูลทั้งหมด
3. ถ้าหากข้อมูลที่ต้องการค้นหามีค่ามากกว่าข้อมูลที่อยู่ตรงกลาง เรียกตัวเองกับข้อมูลครึ่งหลัง
4. ถ้าหากข้อมูลที่ต้องการค้นหามีค่าน้อยกว่าข้อมูลที่อยู่ตรงกลาง เรียกตัวเองกับข้อมูลครึ่งแรก

เวลามากที่สุดที่ใช้ในการค้นหาแบบไบนารีเท่ากับ Olog,n เมื่อ n คือ จำนวนข้อมูล

3.1.4.3 โปรแกรมภาษาซีทำงานค้นหาข้อมูลแบบไบนารีทรี

โปรแกรมจะค้นหาข้อมูลถ้าหากพบจะให้ตำแหน่งของข้อมูลกลับมาถ้าหากไม่พบจะให้ค่า NOTFOUND กลับมา

```
long int Search(FILE *fp,char *StudentId,unsigned long int totalrecord)
{
    unsigned long int low,high,middle;
    low=0;   high=totalrecord-1;
    while(low<=high)
    {
        middle=(low+high)/2;
        if( strcmp(StudentId,GetStudentId(fp,middle)) < 0 ) high=middle-1;
        else if( strcmp(StudentId,GetStudentId(fp,middle)) > 0 ) low =middle+1;
        else return middle; /*found*/
    }
    return NOTFOUND;
}
```

3.2 ส่วนประกอบทางด้านฮาร์ดแวร์ของมาสเตอร์

วงจรทางด้านมาสเตอร์จะมีหน้าที่ที่จะแปลงการส่งข้อมูลแบบอะซิงโครนัสจากหน่วยประมวลผลฐานข้อมูล เป็นการส่งข้อมูลแบบซิงโครนัสผ่านไปในระบบการส่งข้อมูลแบบ ISDN และในทางกลับกันก็ทำการรับข้อมูลจากสลาฟซึ่งเป็นแบบซิงโครนัสไปเป็นการส่งข้อมูลแบบอะซิงโครนัสส่งต่อไปยังหน่วยประมวลผลฐานข้อมูล นอกจากนี้จะต้องทำการจัดลำดับการส่งข้อมูลของสลาฟที่จะส่งเข้ามา และเป็นบัฟเฟอร์ในการส่งข้อมูลของฐานข้อมูลอีกทางหนึ่งด้วย

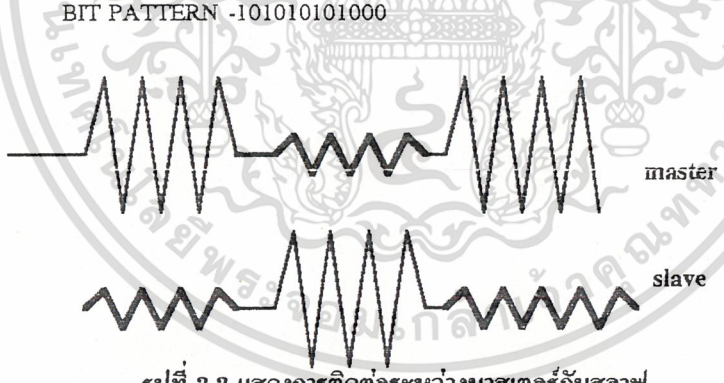
3.2.1 ISDN Universal Digital Loop Transceivers II (UDLT II)

3.2.1.1 รายละเอียดของอุปกรณ์

อุปกรณ์ที่สำคัญสำหรับโครงงานนี้คือ ไอซีเบอร์ MC145421 และ MC145425 สามารถส่งข้อมูลได้ 160 กิโลบิทต่อวินาที ฟูลดูเพล็กส์ไปบนสายคู่ตีเกลียว(twisted pair) ขนาด 26 AWG ในระยะทาง 1 กิโลเมตร โดยจะมีช่องทาง(channel) ในการส่ง 4 ช่องทาง คือ ช่องทาง B ขนาด 64 กิโลบิทต่อวินาที 2 ช่องทาง และ ช่องทาง D ขนาด 16 กิโลบิทต่อวินาที อีก 2 ช่องทาง โดยจะใช้เทคนิคมอดดูเลทแบบ 512 kilobaud MDPSK สามารถส่งโปรโตคอลได้อิสระ ดังนั้นจึงเหมาะกับโครงงานนี้เป็นอย่างยิ่ง

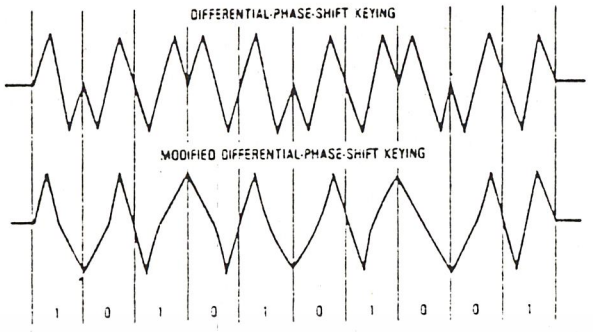
โดยโครงสร้างของไอซีเป็นเทคโนโลยีซีมอส(CMOS) แบบแอลเอสไอ(LSI) โดย MC145421 จะเป็นตัวมาสเตอร์ และ MC145425 เป็นตัวสลาฟ ถูกออกแบบมาให้ทำงานรวมกับ PABX และ PBX แต่ในโครงงานนี้จะใช้ไมโครโปรเซสเซอร์ Z80 เป็นตัวควบคุมมาสเตอร์ และ ไมโครชิพ PIC เป็นตัวควบคุมสลาฟ

การส่งข้อมูลจะส่งเป็นชุด โดยจะส่งข้อมูลเป็นเฟรม ภายใต้สัญญาณนาฬิกาขนาด 8 กิโลเฮอร์ซ กิโลเฮิร์ท หรือ 125 ไมโครวินาที(μs)ต่อหนึ่งเฟรม(frame) ในส่วนของมาสเตอร์แต่ละช่วงของเฟรมจะขึ้นอยู่กับสัญญาณขาขึ้นของสัญญาณ MASTER SYNC INPUT (MSI) ในส่วนของสลาฟแต่ละช่วงของเฟรมจะเริ่มขึ้นหลังจากการดีมอดดูเลทกลุ่มสัญญาณ(burst) จาก มาสเตอร์ ซึ่งจะมีขนาด 20 บิท แบ่งเป็น B สองช่องทาง(8 บิท/ช่องทาง) และ D สองช่องทาง (2 บิท / ช่องทาง) การส่งและรับจะโต้ตอบกันเป็นแบบ PING-PONG fashion ดังในรูป 3.3



รูปที่ 3.3 แสดงการติดต่อระหว่างมาสเตอร์กับสลาฟ

รูปแบบของการมอดดูเลทภายในไอซีเป็นแบบ MDPSK นั้นจะมีข้อที่แตกต่างจาก DPSK ที่ว่า Differential - Phase Shift Keying เข้ารหัส(encode) เฟสจากสัญญาณพาหะ(Carrier) 512 กิโลเฮิร์ท เลข "0" จะบ่งชี้ด้วย 180^0 เฟสเลื่อนไปในระหว่างขอบเขตของบิท ในขณะที่ "1" จะถูกชี้ด้วยสัญญาณที่เป็นเฟสเดียวกัน MPDSK จะมีการเลื่อนเฟสในลักษณะเดียวกับ DPSK แต่จะมีการเลื่อนเฟสทุกช่วงครึ่งหนึ่งของสัญญาณพาหะ 512 กิโลเฮิร์ท คือที่ 256 กิโลเฮิร์ท ดังแสดงในรูปที่ 3.4



รูปที่ 3.4 แสดงเปรียบเทียบระหว่าง MDPSK กับ DPSK

สำหรับการอินเตอร์เฟสไอซีกับสายคู่ตีเกลียวนั้นจะต้องมีหม้อแปลงเพื่อที่จะทำอิมพีแดนซ์แมทชิ่งเป็นตัวจำกัดแบนวิดท์ เพิ่มแรงดันขารับให้ถึงระดับขีดเริ่ม(Threshold) และเป็นตัวป้องกันอินพุท สำหรับขา LO1 และ LO2 ใช้ตัวต้านทาน 440 โอห์ม โดยจะใช้ด้านละ 220 โอห์ม เมื่อผ่านหม้อแปลงที่มีอัตราส่วนจำนวนรอบเท่ากับ 2:1 ความต้านทานเสมือนก็จะเหลือ 110 โอห์มซึ่งก็จะแมท(match)กับสายคู่ตีเกลียวขนาด 26 AWG ที่มีการลดทอน 18 เดซิเบลต่อ 1 กิโลเมตร

ระดับแรงดันที่ LO1 และ LO2 เป็น 2.25 โวลท์พีค ครึ่งหนึ่งของแรงดันจะตกคร่อมที่ความต้านทาน 220 โอห์มให้เหลือ 1.12 โวลท์ แล้วก็จะถูกลดลงอีกทีด้วยค่าอัตราส่วนของหม้อแปลง 2:1 รอบ เหลือ 0.56 โวลท์พีค ที่สายขนาด 26 AWG สัญญาณจะถูกลดลงด้วยค่า 18 เดซิเบลต่อกิโลเมตรที่ line side ของด้านรับจะเหลือสัญญาณขนาด 8.9 มิลลิโวลท์พีคจะถูกขยายด้วยหม้อแปลงอัตราส่วน 1:4 ให้เป็น 35.6 มิลลิโวลท์พีค แล้วจะถูกลดลงด้วยค่าความต้านทานภายใน ให้เหลือ 25 มิลลิโวลท์พีค ซึ่งจะเป็นระดับต่ำสุดเข้าไปที่ขา LI

ค่าแรงดันสูงสุดที่ไอซี UDLTs คือ 3.0 โวลท์พีค เพราะถูกลดค่าด้วยความต้านทานของที่แหล่งกำเนิดและขดลวดหม้อแปลง สัญญาณที่ line side ของหม้อแปลงจะมีค่า 0.75 โวลท์พีค เมื่อผ่านหม้อแปลงอัตรารอบ 1:4สัญญาณจะมีระดับเป็น 3 โวลท์พีค ซึ่งจะเกินกว่าค่าอินพุทสูงสุดของ LI ที่จะรับไว้ได้ ค่าแรงดันสูงสุดคือ 2.5 โวลท์พีค ส่วนระดับแรงดันที่เกินมานั้นจะอยู่ในรูปของกระแสจ่ายเข้าไปในซับสเตนของไอซี UDLT ทำให้ไปรบกวนการมอดดูเลชั่นส่งผลให้เกิดการผิดพลาด (Bit Error)ได้ การป้องกันก็จะใช้ตัวต้านทานและไดโอด แคลมป์ที่อินพุท LI ดังในรูปที่ 3.5 แต่ต้องคำนึงถึงไดโอดภายในไอซีด้วยเป็นสำคัญ ไดโอดที่ต่ออยู่ภายนอกควรได้รับไบอัสให้ ทำงาน(turn on) ก่อน จึงต้องต่อตัวต้านทานขนาด 5 กิโลโอห์มเพื่อให้ไดโอดทำงาน

พาว์เวอร์แบนด์วิธ (Power Bandwidth) ที่ UDLTสามารถส่งได้กว้างที่สุดคือ 8 - 512 กิโลเฮิร์ท แต่เพื่อชดเชยการลดทอนต่างๆจะใช้ในช่วง 20 - 512 กิโลเฮิร์ท จึงเป็นการกำหนดขนาดแบนวิธของหม้อแปลงไปในตัว ที่ค่า ช่วงความถี่ต่ำ(lower bandwidth) 20 กิโลเฮิร์ท จะถูกกำหนดด้วยค่าความเหนี่ยวนำที่ขดลวดของหม้อแปลง 1.75 มิลลิเฮนรี่ ส่วนที่ช่วงความถี่สูง(upper bandwidth) 512 กิโลเฮิร์ทจะถูกกำหนดด้วยค่าคาปาซิเตอร์ที่ 0.001 ไมโครเฮนรี่โดยจะต่อขนานกับแท้ปหม้อแปลงดังในรูปที่ 3.5

3.2.1.2 การประยุกต์การใช้งาน

ในการออกแบบนั้นเราต้องพิจารณาสัญญาณควบคุมต่างๆให้ครบ ซึ่งรายละเอียดเกี่ยวกับสัญญาณควบคุมต่างๆจะอยู่ในภาคผนวก เราจะแยกกล่าวในรายละเอียดของสัญญาณดังต่อไปนี้

- CCI (high speed clock input) จะใช้วงจรออสซิลเลเตอร์ซึ่งประกอบด้วยไอซี 74HC04 หรือ CD4069 ,คริสตอลขนาด 8 เมกกะเฮิร์ท ,ตัวเก็บประจุ 20 พิโกฟารัด และความต้านทานขนาด 10 เมกกะโอห์มต่อกันดังรูป 3.7 เพื่อผลิตสัญญาณนาฬิกาขนาด 8 เมกกะเฮิร์ท

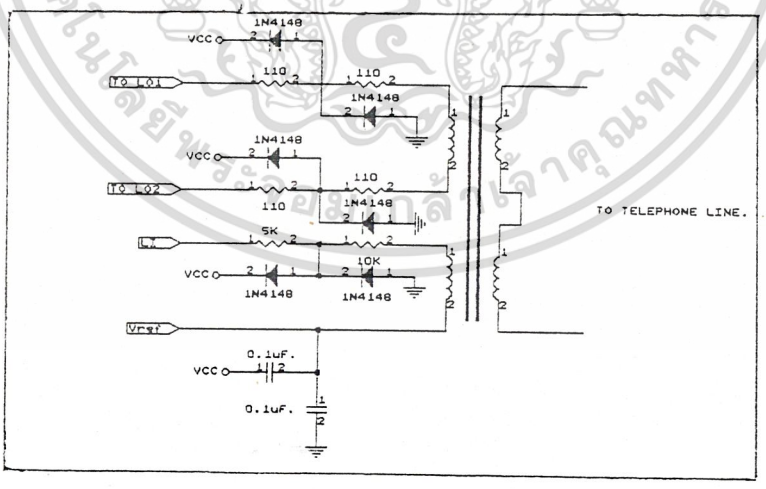
- MSI (master sync input) ,TDC/RDC (transmit/recieve data clock input) จะใช้ไอซีดีฟลิปฟลอบและไอซีตัวหารต่อจากสัญญาณนาฬิกาขนาด 8 เมกกะเฮิร์ทที่ผลิตให้ CCI มาหารความถี่ให้ได้สัญญาณนาฬิกาขนาด 8 กิโลเฮิร์ท(สำหรับ MSI)และสัญญาณนาฬิกาขนาด 256 กิโลเฮิร์ท(สำหรับ TDC/RDC) โดยใช้ 74HC74 หารสอง 2 ครั้ง แล้ว ผ่านเข้าไปที่ 74HC393 เพื่อผลิตความถี่ต่อไป การต่อวงจรจะแสดงในรูปที่ 3.7

-TE1,TE2,RE1,RE2 เป็นสัญญาณอนุญาตในการส่งหรือรับข้อมูล จะนำอินเวอร์เตอร์(inverter) และแอนเกท (Ands gate) มาประยุกต์ในการสร้างสัญญาณควบคุมดังรูปที่ 3.7 ส่วนแผนผังสัญญาณนาฬิกา (timming diagram) ของสัญญาณควบคุมที่กล่าวมาทั้งหมดจะแสดงในรูปที่ 3.10

- V_{REF} ,LI , LO1 ,LO2 จะมีการต่อดังรูปที่ 3.5 (ส่วนนี้ได้อธิบายไปแล้วในหัวข้อ รายละเอียดของอุปกรณ์)

- SE , LB , PD จะต่อกับ +5 โวล์ท

ส่วนขา Tx , Rx และ VD จะกล่าวอีกทีในการต่อกับ Z80 PIO



TRANFORMER PARA METERS

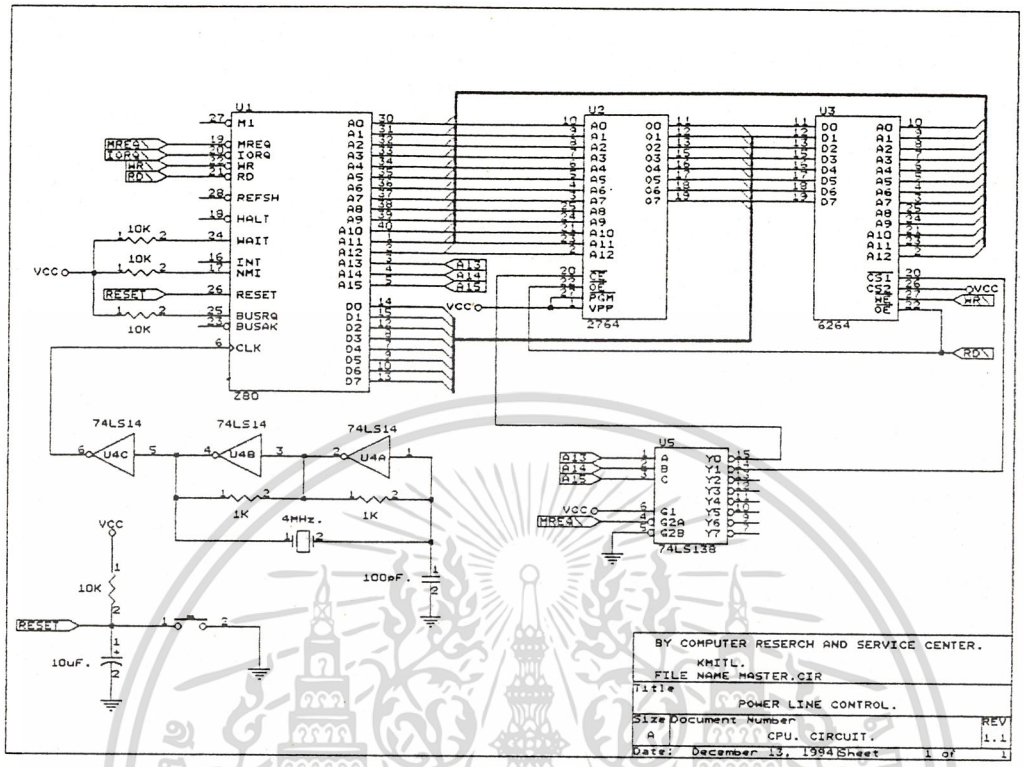
INDUCTANCE OF Tx WINDING 1.75 mH

TURN RATIO: Tx:L1+L2 2:1

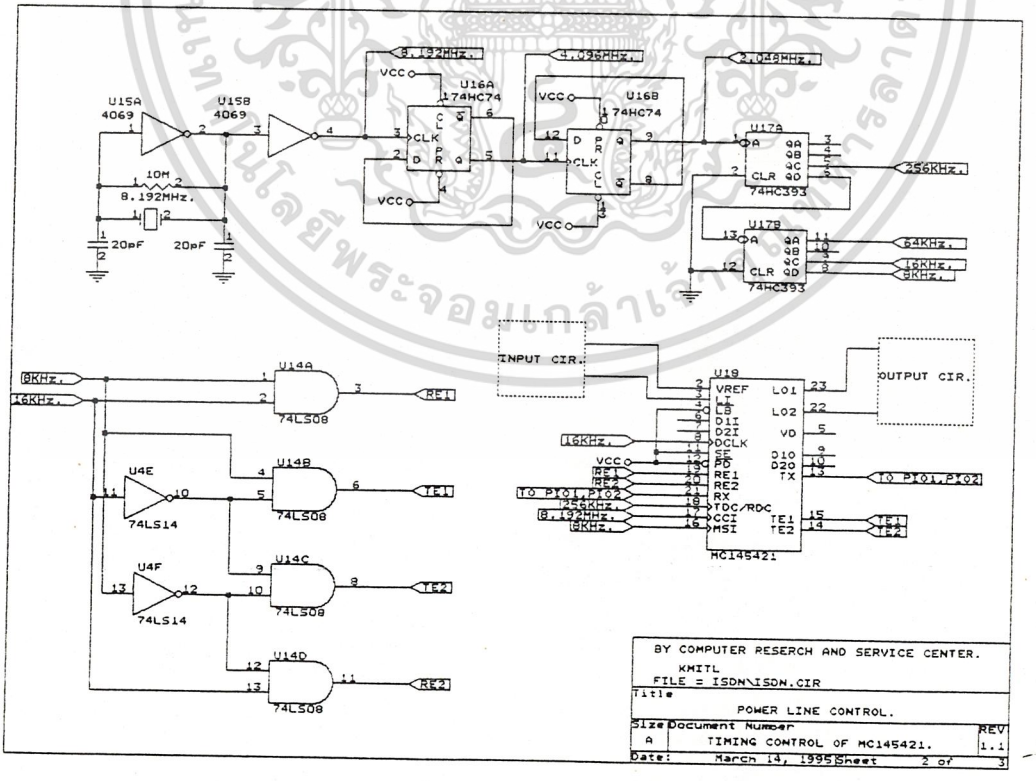
TURN RATIO: Rx:L1+L2:4:1

DIODE: 1N4148 OR EQUIVALENT

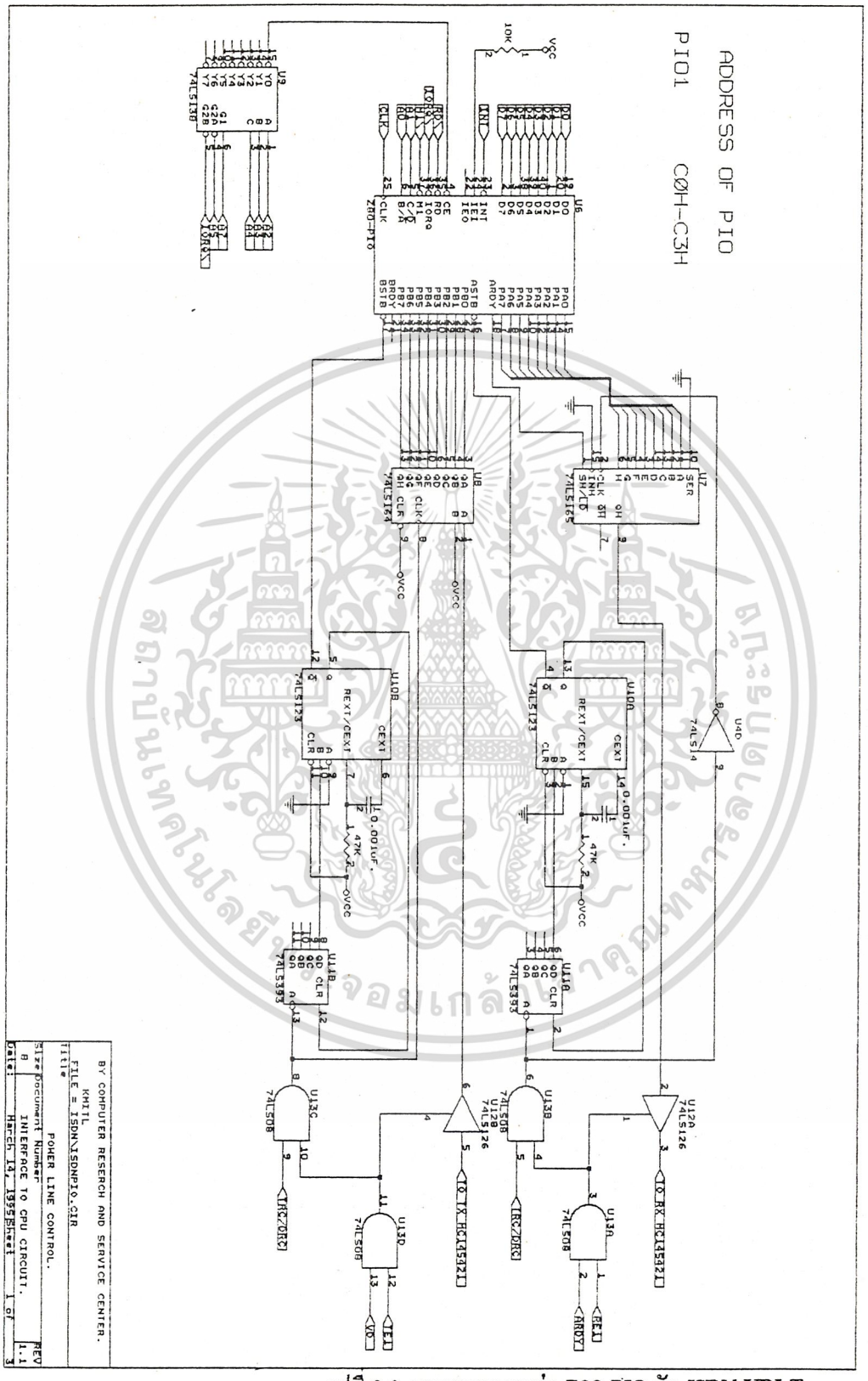
รูปที่ 3.5 การอินเตอร์เฟสกับสายคู่ตีเกลียว



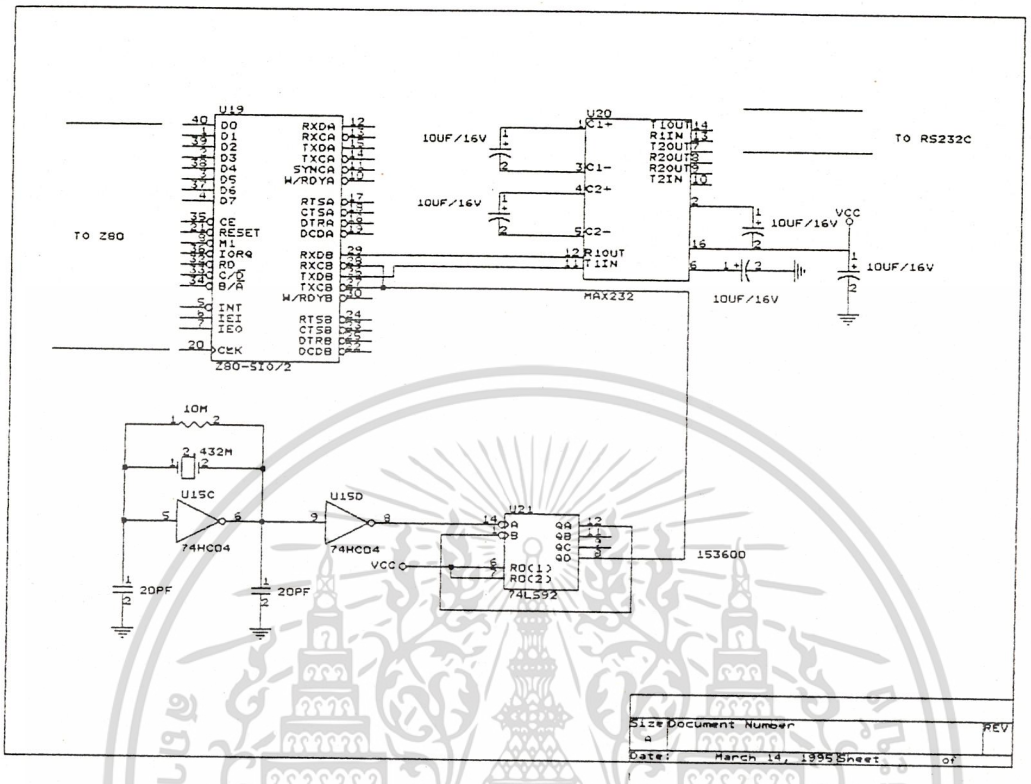
รูปที่ 3.6 วงจรมาสเตอร์ส่วนที่ใช้ Z80 ควบคุม



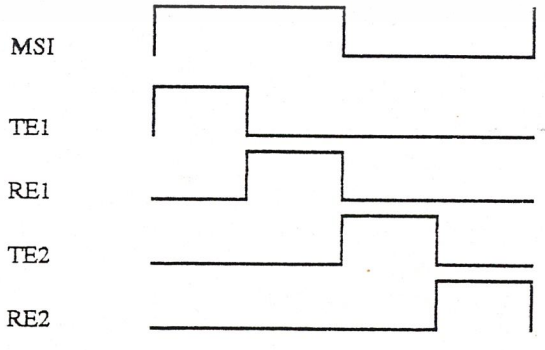
รูปที่ 3.7 วงจรส่วนสัญญาณนาฬิกาและสัญญาณควบคุม ISDN



รูปที่ 3.8 วงจรแสดงการต่อ Z80 PIO กับ ISDN UDLTs



รูปที่ 3.9 แสดงการเชื่อมต่อ Z80 SIO กับ RS232C

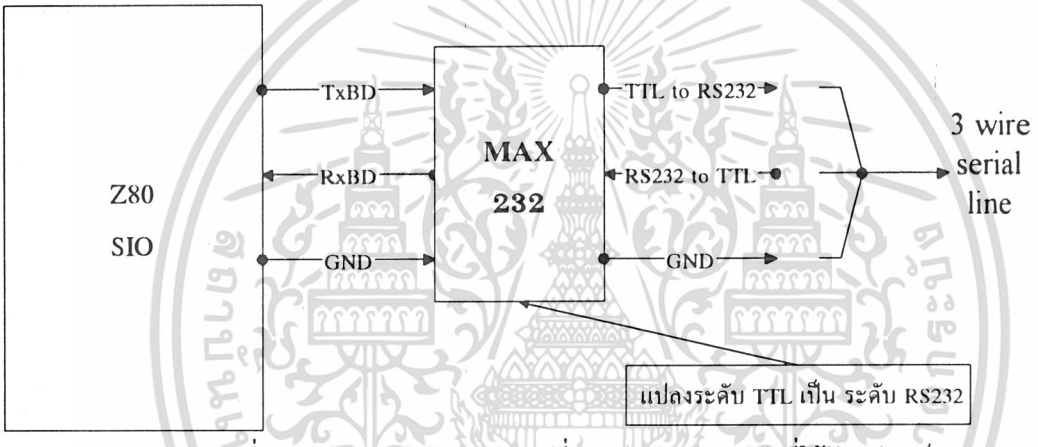


รูปที่ 3.10 แสดงแผนผังสัญญาณควบคุม MSI , TE1 , TE2 , RE1 , RE2

3.2.2 การใช้งาน Z80 SIO ในการติดต่อสื่อสารผ่านพอร์ทอนุกรม

3.2.2.1 รายละเอียดของอุปกรณ์

ในการติดต่อสื่อสารระหว่างบอร์ดมาสเตอร์กับเครื่องไมโครคอมพิวเตอร์นั้น ในโครงงานนี้จะใช้ Z80 Serial Input/Output Controller (SIO) ทำการส่งและรับข้อมูลแบบอะซิงโครนัส(ซึ่งรายละเอียดเกี่ยวกับ Z80 SIO โดยละเอียดจะอยู่ในภาคผนวก) และ Z80 SIO ที่จะนำมาใช้นั้นเป็น Z80 SIO/2 การเชื่อมต่อระหว่าง Z80 SIO กับพอร์ท RS-232C นั้นจำเป็นที่จะต้องคำนึงถึงรูปแบบมาตรฐานของ RS-232C ด้วย เนื่องจาก RS-232C นั้นไม่ได้ระดับแรงดันปกติของ ทีทีแอล(TTL) แต่จะใช้ระดับแรงดันในสายส่งประมาณ ±12 โวลท์ ดังนั้นในการอินเตอร์เฟสระหว่าง Z80 SIO กับ RS-232C นั้นจะต้องทำการเปลี่ยนระดับสัญญาณ TTL ไปเป็นระดับสัญญาณมาตรฐานของ RS-232C ในการส่งข้อมูล และเปลี่ยนจากระดับสัญญาณมาตรฐานของ RS-232C ไปเป็นระดับสัญญาณ TTL ในการรับข้อมูล อุปกรณ์ที่มาหน้าที่ทั้งสองอย่างนี้คือ IC เบอร์ MAX232 ของMAXSIM การเชื่อมต่อระหว่างอุปกรณ์ดังกล่าวจะแสดงดังในรูปที่ 3.11



รูปที่ 3.11 แสดงแผนผังของการแปลี่ยนแปลงระดับแรงดันที่ใช้กับอุปกรณ์
พวก TTL ซึ่งเป็นโครงสร้างภายในของ SIO (5 โวลท์)กับบัสของ RS-232C (12 โวลท์)

สัญญาณนาฬิกาที่ใช้จะสร้างขึ้นจากการหารความถี่ 1.8632 เมกกะเฮิร์ทด้วยไอซีตัวนับ 74LS92 โดยหารให้ด้วย 12 จะเหลือความถี่ 153,600 เฮิร์ท จากนั้นก็จะทำการหารโดยโปรแกรมด้วยค่า 16 ก็จะได้ความถี่สัญญาณอัตราบอดเท่ากับ 9,600 เพื่อเป็นสัญญาณนาฬิกาที่ใช้ในการติดต่อ RS-232C

3.2.2.2 การประยุกต์การใช้งาน

ในส่วนของวงจรจะแสดงดังในรูปที่ 3.8 จะต้องทำการ Innitialize SIO ก่อน โดยใส่ค่าเริ่มต้นให้กับ WRITE REGISTER ต่างๆของ SIO เพื่อให้ทำงานตามที่เราต้องการ ในโครงงานนี้ใช้ช่องทาง B ซึ่งจะมี 8 WRITE REGISTER (WR0-WR7) มี 3 READ REGISTER (RR0-RR2) โดยที่ REGISTER แต่ละตัวจะมีความหมายดังต่อไปนี้

WR0 รีเซ็ต SIO

WR4 กำหนดตัวหารความถี่สัญญาณนาฬิกา , จำนวนบิทสิ้นสุด ,รูปแบบพาริตี้บิท

WR3 จำนวนบิทต่อหนึ่งอักขระในโหมดการรับ, เลือก Auto Enable ,เลือก recieve enable bit

WR5 จำนวนบิทต่อหนึ่งอักขระ ในโหมดการส่ง,เลือก DTR และ RTS enable ,เลือก transmitter enable

WR2 กำหนดอินเตอร์รัพท์เวคเตอร์ (สำหรับโหมดอินเตอร์รัพท์)

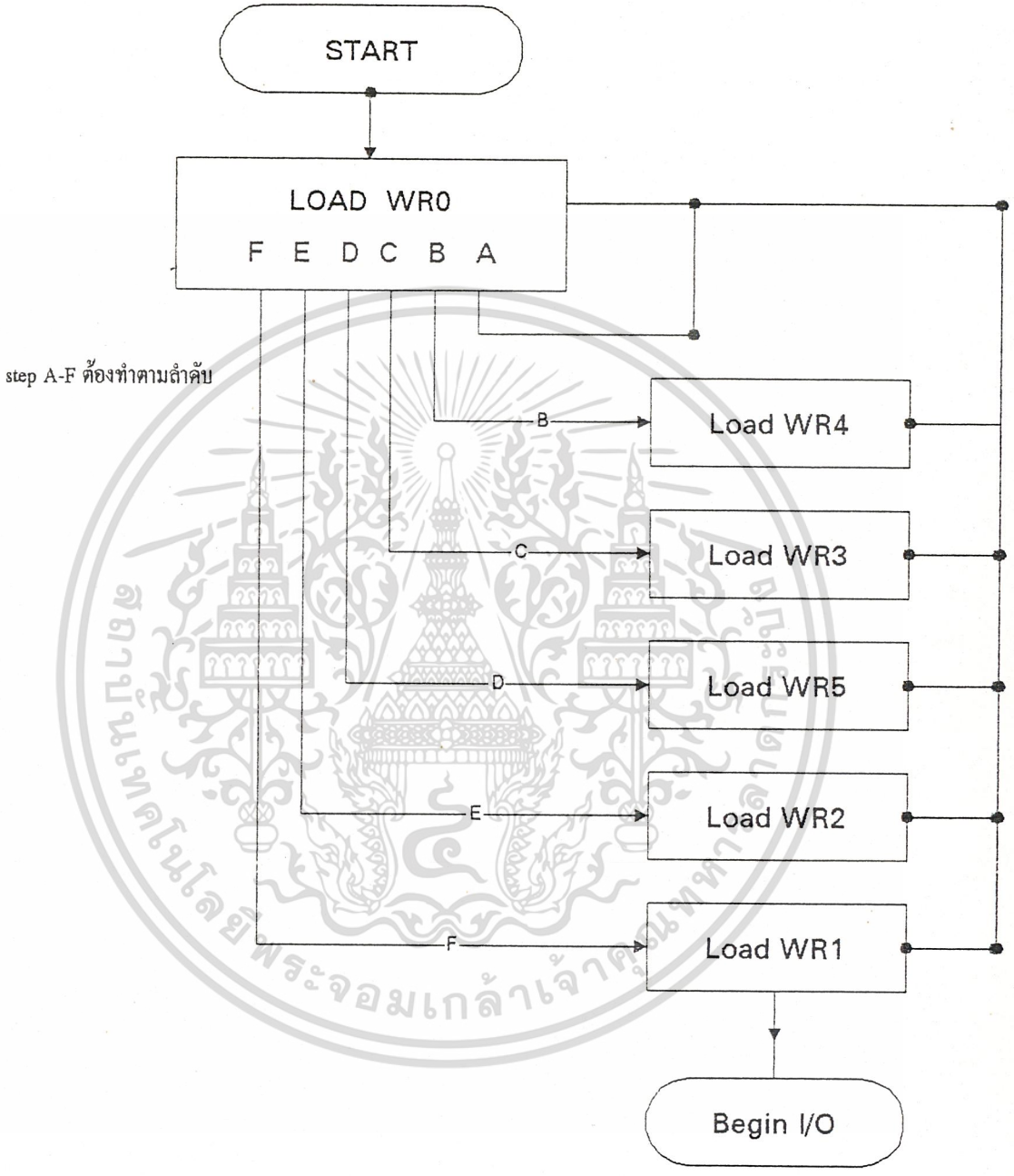
WR1 กำหนดรูปแบบการอินเตอร์รัพท์

ควรที่จะเรียงการเซ็ตตามลำดับข้างบนซึ่งอาจเขียนเป็นอัลกอริทึมได้ดังนี้

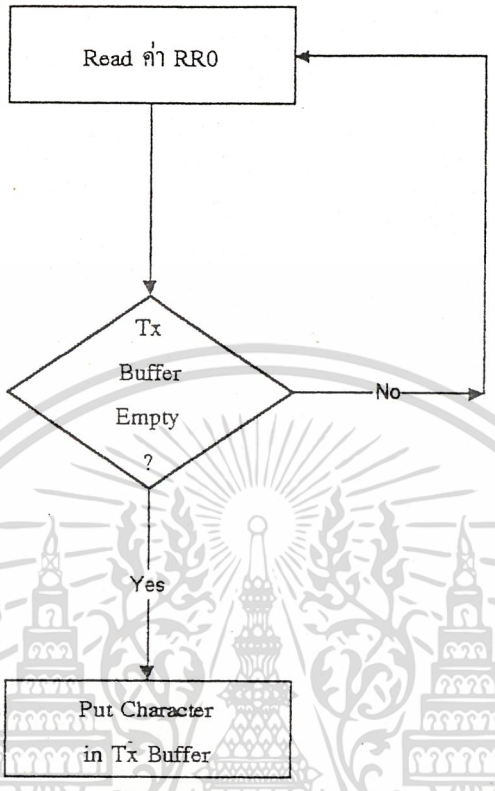
สำหรับโครงงานนี้จะใช้ อัตราบอด 9600, 8 บิทต่อตัวอักษร, 2 บิทสิ้นสุด, พาริตี้ค่โดยขณะรับข้อมูลจะอยู่ในโหมดอินเทอร์รัพต์ ส่วนการส่งข้อมูลจะโปรแกรมให้อยู่ในโหมดพอลลิ่ง(POLLING) ซึ่งก็จะมีขั้นตอนของอัลกอริทึมตาม แผนผังการทำงาน

หลังจาก initialized SIO จะทำการตรวจสอบบิท 2 ของ RR0 ถ้าเป็น 0 แสดงว่าบัฟเฟอร์ไม่ว่าง ตัวอักษร ตัวใหม่ก็จะไม่ทำการส่ง ช่วงนี้ซีพียูจะไปทำงานอื่นก่อนหรือจะตรวจสอบรอสัญญาณวนรอไปเรื่อยๆ ถ้าบิท 2 ของ RR0 เป็น 1 แสดงว่า Buffer Empty CPU ก็จะทำการส่งข้อมูลออกไป สำหรับ DTR , DTE ,RTS และ CTS ในโครงงานนี้ไม่ได้ใช้ โปรมแกรมข้างล่างจะแสดงการ initial Z80 SIO

```
; program for initial asynchornous  with Z80SIO
        CPU     "Z80.TBL"
        HOF     "BIN8"
        ORG     0000H
ADATA:EQU       80H             ;CHANNEL A DATA
BDATA:EQU       81H             ;CHANNEL B DATA
ACONT:EQU       82H             ;CHANNEL A CONTROL
BCONT: EQU      83H             ;CHANNEL B CONTROL
;INITIAL SIO INTERRUPT
SIOINI:  DI
        LD      HL,TABLE1
SINIT1: LD      A,(HL)
        CP      0FFH
        JR      Z,SINIT2
        OUT     (BCONT),A
        INC     HL
        JR      SINIT1
;TABLE FOR INITIAL SIO INTERRUPT
```



รูปที่ 3.12 แสดง แผนผังการทำงาน การ initialize Z80 SIO



รูปที่ 3.12 แสดง แผนผังการทำงาน ของ Polled Transmit

```
TABLE1:DFB    18H        ;WR0 CHANNEL RESET
       DFB    02H        ;ชี้ไปที่ WR2
       DFB    00H        ;WR2 INT VECT = 00H
       DFB    03H        ;ชี้ไปที่ WR3
       DFB    0C1H       ;8 BIT REC ,RX ENABLE
       DFB    04H        ;ชี้ไปที่ WR4
       DFB    4FH        ;WR4,X16 CLK ,2 STOP,EVEN PARITY
       DFB    05H        ;ชี้ไปที่ WR5
       DFB    68H        ;8 BIT TRAN,TX ENABLE
       DFB    01H        ;ชี้ไปที่ WR1
       DFB    14H        ;ENABLE INTERRUPT
       DFB    0FFH
```

3.2.3 การใช้งาน Z80 PIO

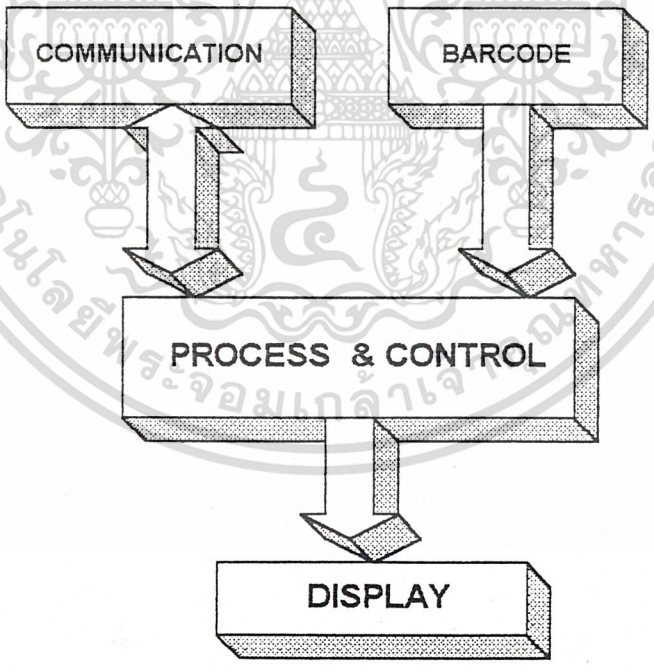
3.2.1.1 รายละเอียดของอุปกรณ์

ในการส่งข้อมูลแบบซิงโครนัสจะใช้ Z80 PIO ทำงานร่วมกับไอซีชิฟท์รีจีสเตอร์ (shift register) 74LS165 (ขนาน แปลงเป็น อนุกรม) 74LS164 (อนุกรม แปลงเป็น ขนาน) เพื่อทำการส่งข้อมูลแบบซิงโครนัสให้กับ ISDN Z80 PIO จะประกอบด้วยพอร์ตขนานสองพอร์ตสามารถต่อกับ Z80 โดยมีการอินเทอร์รัพท์ในโหมดสอง รายละเอียดและแผนผังสัญญาณนาฬิกาจะอยู่ในภาคผนวกโดยละเอียด

3.2.1.2 การประยุกต์การใช้งาน

วงจรแสดงในรูปที่ 3.8 ในโครงงานนี้จะใช้พอร์ต A เป็น พอร์ตเอาท์พุทต่อกับ 74LS165 ส่งข้อมูลผ่าน 74LS126 โดยมีสัญญาณ RE1 และ AREADY เป็นสัญญาณควบคุมการส่งข้อมูล ส่วน 74LS393 และ 74LS123 เป็นตัวควบคุมการเลื่อนข้อมูลของ 74LS165 และคอยส่งสโตรบให้ ASTB เพื่อบอกให้รู้ว่าอุปกรณ์พร้อมรับข้อมูลจาก Z80 PIO ดังนั้นการทำงานของพอร์ต A จะต้องเป็นโหมด 0 ส่วนการรับข้อมูลจาก ISDN จะใช้พอร์ต B กับ 74LS164 โดยมีสัญญาณจาก VD และ TE1 เป็นตัวควบคุมการไหลของข้อมูล มี 74LS393 และ 74LS123 เป็นตัวควบคุมการเลื่อนข้อมูลของ 74LS164 และคอยส่งสโตรบให้ BSTB เพื่อบอกให้ PIO รับข้อมูลจาก 74LS164

3.3 การทำงานของวงจรของบอร์ดควบคุมเทอร์มินอล (บอร์ดสเลฟ)



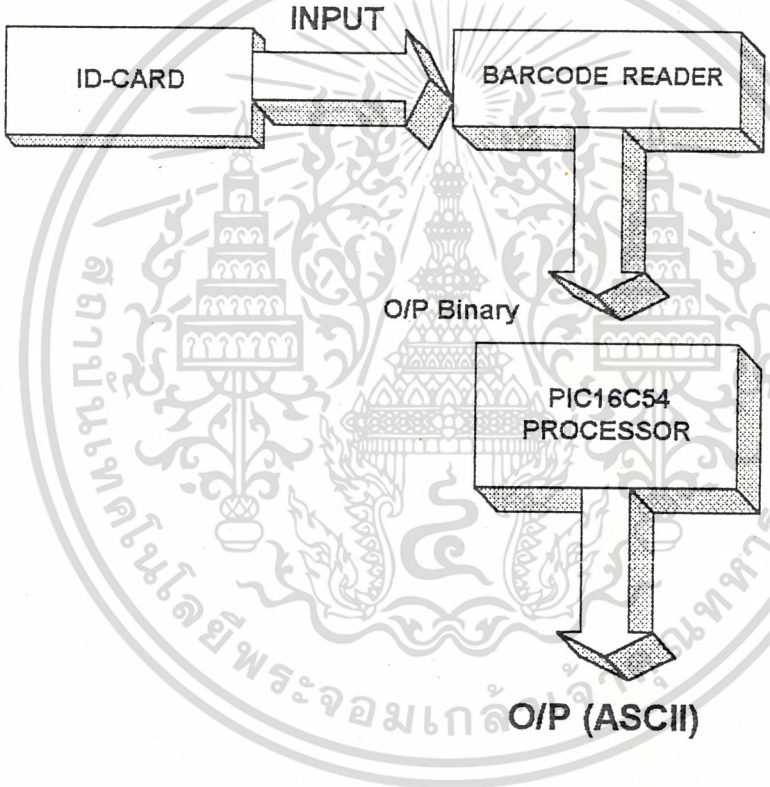
รูปที่ 3.13 บล็อคไดอะแกรมบอร์ดเทอร์มินอล

3.3.1 ส่วนประกอบหลักของบอร์ดเทอร์มินอล

บอร์ดสำหรับควบคุมเทอร์มินอลนี้จะประจำอยู่ที่เครื่องเทอร์มินอลของระบบแลน โดยเทอร์มินอลหนึ่งเครื่องจะใช้บอร์ดควบคุมเทอร์มินอลหนึ่งบอร์ด หน้าที่หลักของบอร์ดนี้คือ การติดต่อระหว่างยูสเซอร์และหน่วยประมวลผลกลาง ซึ่งมีส่วนประกอบหลัก 3 ส่วนดังรูปที่ 1

ซึ่งหน้าที่ของแต่ละส่วนเป็นดังนี้

1. Barcode (รหัสแถบ) ส่วนนี้จะทำหน้าที่รับอินพุทรหัสแถบรหัสนักศึกษา แล้วนำมาประมวลผลถอดรหัส แล้วแปลงเป็นรหัสแอสกี้ แล้วทำการส่งข้อมูลให้ส่วนประมวลผลและควบคุม (Process & Control) ต่อไป ส่วนของรหัสแถบนี้จะประกอบด้วยไอซีไมโครคอนโทรลเลอร์ เบอร์ PIC16C54 ทำหน้าที่ถอดรหัสของรหัสแถบและเครื่องอ่านรหัสแถบแบบรูด 1 ชุด ซึ่งทำหน้าที่รับอินพุทข้อมูลจากรหัสแถบของรหัสนักศึกษา และได้เอาท์พุทเป็นเลขฐานสอง แล้วจึงส่งให้ ไอซี PIC16C54 ทำการถอดรหัสต่อไป

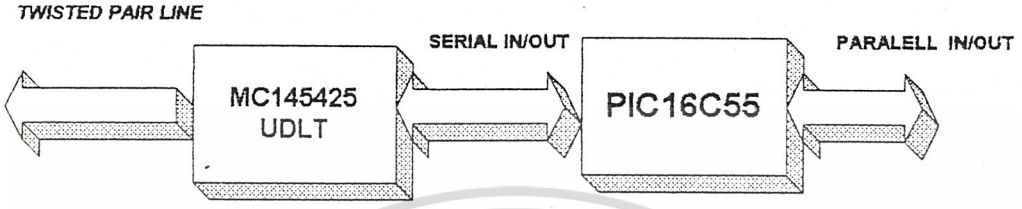


รูปที่ 3.14 บล็อคไดอะแกรม ของวงจรส่วนถอดรหัสบาร์โค้ด

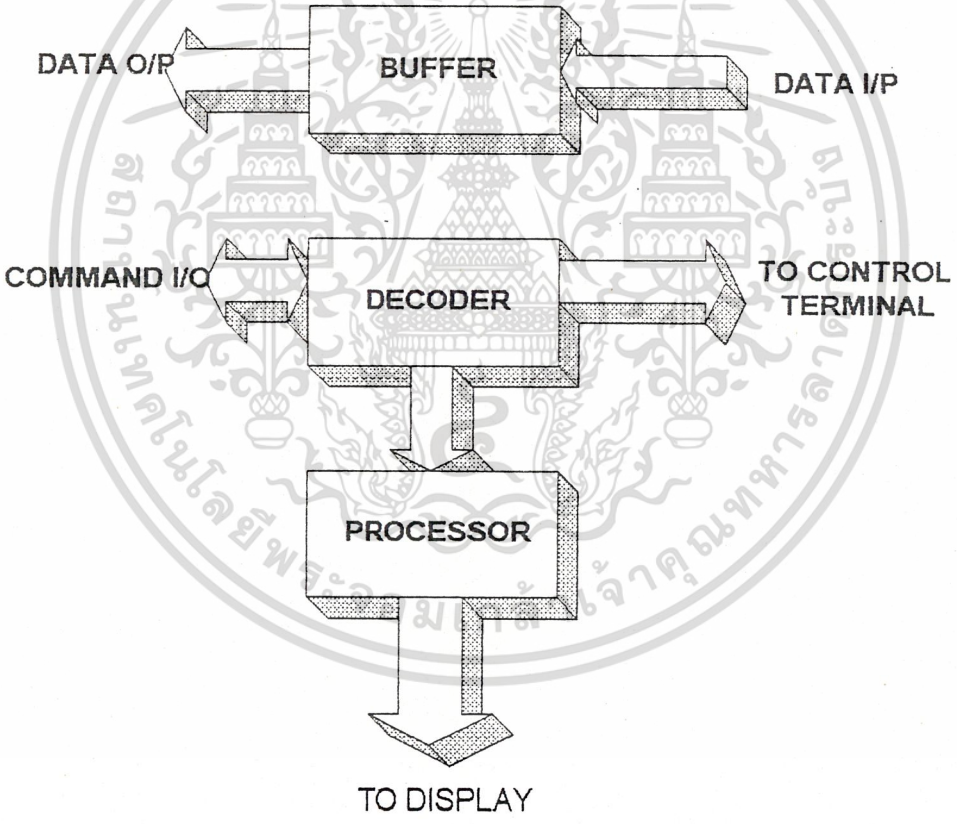
2. วงจรส่วน communication ประกอบด้วยไมโครคอนโทรลเลอร์ PIC16C55 ทำหน้าที่เชื่อมต่อระหว่างบอร์ดสเลฟ ควบคุมเครื่องเทอร์มินอลกับหน่วยประมวลผลกลาง เนื่องจากระหว่างบอร์ดควบคุมเทอร์มินอลกับหน่วยประมวลผลกลาง เชื่อมต่อผ่านสาย twisted pair ซึ่งเป็นการสื่อสารแบบอนุกรม แต่ส่วน ประมวลผลและควบคุม จะทำการสื่อสารแบบขนานและมีความเร็วต่ำ ดังนั้นจึงต้องมี PIC16C55 คอยทำการควบคุมช่วยตรงส่วนนี้คือจะแปลงข้อมูลจาก อนุกรมเป็นขนาน และ แปลงจากขนานเป็นอนุกรม และใช้เป็นที่พักข้อมูล (Buffer) ด้วย

นอกจาก PIC16C55 แล้วยังมีไอซี UDLT ของโมโตโรล่าเบอร์ MC145425 ซึ่งใช้ทำหน้าที่ มอดูเลตข้อมูลแล้วส่งผ่านแชนเนล B และ D แบบอนุกรมด้วยความเร็ว 160 กิโลบิตต่อนาที ซึ่งเป็นความเร็วของการส่ง

ข้อมูลในระบบ ISDN เนื่องจากเหตุผลทางด้านความเร็ว จึงนำ IC MC145425 มาใช้งาน นอกจากนี้ยังสามารถส่งข้อมูลที่เป็นเสียงได้อีกด้วย ซึ่งเหมาะในการที่จะพัฒนาระบบต่อไปในอนาคต



รูปที่ 3.15 บล็อคไดอะแกรมวงจรส่วน communication



รูปที่ 3.16 บล็อคไดอะแกรมแสดงหน้าที่ของวงจรส่วน Process & control

3. วงจรส่วน process & control วงจรส่วนนี้ถือว่าเป็นหัวใจของบอร์ดนี้เลยก็ว่าได้ ซึ่งวงจรส่วนนี้ประกอบด้วยไอซีไมโครคอนโทรลเลอร์ intel 8749 ทำหน้าที่รับข้อมูลจากวงจรส่วนรหัสแถบซึ่งเป็นรหัสแอสกี้และส่งต่อให้วงจรส่วน communication ทำการถอดรหัสคำสั่งที่ได้รับจากหน่วยประมวลผลกลางเพื่อนำไปควบคุมเครื่องเทอร์มินอลต่อไป และนอกจากนี้ไอซี 8749 ยังทำการควบคุมการแสดงผลของการทำงานผ่านจอแอลซีดี เพื่อให้การติดต่อกับผู้ใช้เกิดประสิทธิภาพมากที่สุด

3.3.2 การทำงานของวงจรบอร์ดควบคุมเทอร์มินอล

จะขอเริ่มที่วงจรส่วนรหัสแถบก่อนโดยไอซี PIC16C54 ซึ่งใช้สัญญาณนาฬิกาความถี่ 4 เมกะเฮิร์ท ใช้ตัวกำเนิดความถี่แบบ RC ต่อเข้าที่ขา osc1 และ osc2 ตามลำดับ ที่ขา MCLR จะต่อวงจรพาวเวอร์ออนรีเซ็ต พอร์ต RB0...RB7 จะเป็นบัสข้อมูลสำหรับส่งข้อมูลรหัสนักศึกษาที่อ่านได้จากวงจรส่วนรหัสแถบให้กับ ไอซี 8749 ส่วนพอร์ต RA จะใช้เป็นสัญญาณควบคุมและใช้สำหรับเป็นขาอินพุทคาต้าที่เข้ามาเครื่องอ่านรหัสแถบ

เมื่อนักศึกษานำบัตรนักศึกษาที่มีรหัสแถบมารูดผ่านเครื่องอ่านรหัสแถบ ที่ขาอินพุท RA0 จะมีข้อมูลเลขฐานสองเข้ามา จากนั้นไอซี PIC16C54 ซึ่งมีโปรแกรมสำหรับถอดรหัสข้อมูลของรหัสแถบ จะทำการถอดรหัสออกมาเป็นรหัสนักศึกษา แล้วจึงแปลงเป็นรหัสแอสกี้อีกที หลังจากเสร็จขบวนการนี้แล้ว จึงจะส่งให้ไอซี 8749 ผ่านทางพอร์ต RB0...RB7

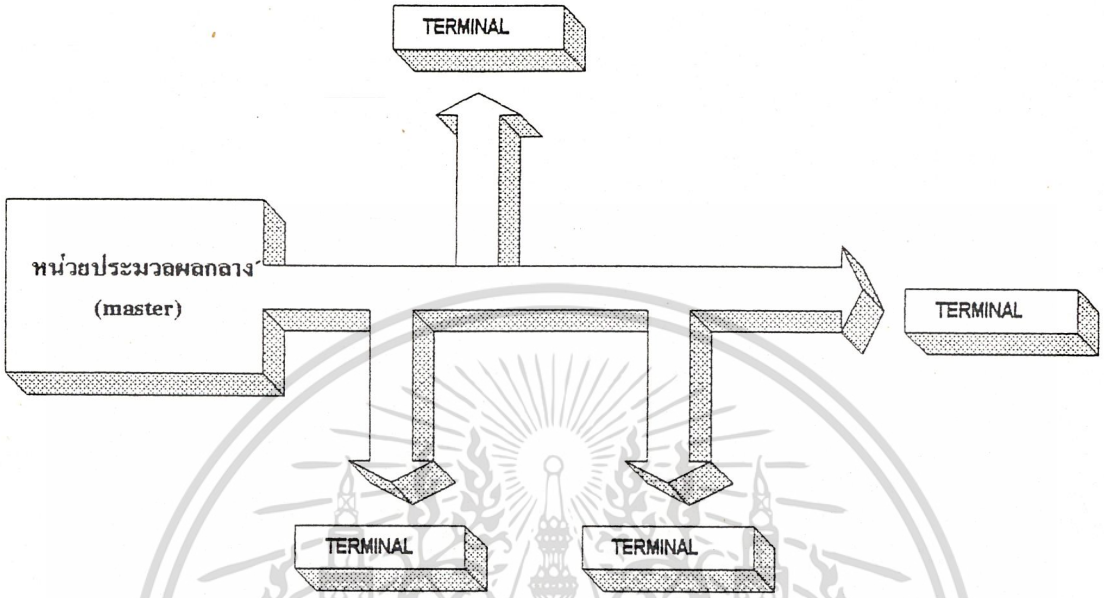
เมื่อไอซี 8749 ได้รับรหัสนักศึกษาแล้วก็จะทำการส่งผ่านบัสข้อมูลเข้าเป็นอินพุทของพอร์ต RC0...RC7 ของ ไอซี PIC16C55 วงจรส่วนนี้จะทำการแปลงข้อมูลแบบให้เป็นแบบอนุกรม หลังจากนั้นก็จะรอสัญญาณเรียกจากหน่วยประมวลผลกลาง เมื่อได้รับสัญญาณเรียกจากหน่วยประมวลผลกลางแล้วก็จะส่งข้อมูลผ่านพอร์ต RB7 ไปให้ ไอซี MC145425 ซึ่งจะทำหน้าที่มอดดูเลตและส่งข้อมูลด้วยความเร็วสูงผ่านสายโทรศัพท์ไปให้หน่วยประมวลผลกลางต่อไป ก่อนที่ไอซี PIC16C55 จะส่งข้อมูลออกเอาท์พุท จะทำการต่อขาเอาท์พุทของไอซี MC145425 เข้ากับสายโทรศัพท์ ด้วยการใช้รีเลย์ทำการตัดต่อ แล้วจึงส่งข้อมูลให้ไอซี MC145425 ต่อไป เหตุที่ต้องมีการต่อรีเลย์เพื่อทำการสวิตช์นั้น เนื่องจากบอร์ดควบคุมเทอร์มินอลจะมีการเชื่อมต่อหน่วยประมวลผลกลางแบบบัสหรือแบบมัลติดรอป(Multidrop) นั่นเอง ดังรูป 3.17

เมื่อ ไอซี MC145425 ได้รับข้อมูลแล้วก็จะทำการมอดดูเลต และส่งออกเอาท์พุทที่ขา Tx ผ่านสายโทรศัพท์ต่อไป การทำงานของวงจรเมื่อมีข้อมูลส่งมาจากหน่วยประมวลผลกลาง ขั้นตอนการทำงานก็จะคล้าย ๆ กับตอนส่งข้อมูลออกเอาท์พุทของบอร์ดควบคุมเทอร์มินอล เพียงแต่มีทิศทางตรงกันข้ามเท่านั้นเอง และข้อมูลที่ส่งจากหน่วยประมวลผลกลางจะเป็นไปในรูปแบบของคำสั่ง (command) ส่งมาให้กับบอร์ดควบคุมเทอร์มินอล เพื่อที่จะทำการคอนโทรลและติดต่อกับผู้ใช้ โดยที่ไอซี 8749 จะทำการถอดรหัสคำสั่งและปฏิบัติตามคำสั่งที่ได้รับมา ตัวอย่างเช่นอาจสั่งให้ ต่อไฟ 220 โวลต์ ให้เครื่องเทอร์มินอล หรือทำการตัดไฟ 220 โวลต์จากเครื่องเทอร์มินอล หรือส่งคำสั่งออก ที่หน้าจอแสดงผลเป็นการแจ้งให้ผู้ใช้ได้ทราบว่าขณะนี้มีการเรียนการสอนไม่อนุญาตให้บุคคลภายนอกใช้งานได้ เป็นต้น โดยคำสั่งนี้มีความยืดหยุ่นมากเพราะเป็นซอฟท์แวร์จึงทำให้สามารถปรับเปลี่ยนได้โดยง่าย หน้าที่ของบอร์ดนี้ถ้าจะพูดง่าย ๆ ก็คือ คอยรับคำสั่งจากหน่วยประมวลผลกลางว่าจะให้ทำการต่อไฟ 220 โวลต์ เข้ากับเครื่องเทอร์มินอลหรือไม่เท่านั้นเอง

3.3.3 การสร้างและการคำนวณ

ไอซี PIC16C54 ใช้สัญญาณนาฬิกา 4 เมกะเฮิร์ท ต่อเข้าที่ขา osc1 และ osc2 ที่ขา 4 (MCLR) จะต่อวงจรพาวเวอร์ออนรีเซ็ต ที่ขา RTCC จะต่อลงกราวด์ พอร์ต RB0..RB7 ใช้เป็นช่องทางติดต่อข้อมูลกับ ไอซี 8749 ส่วนพอร์ต RA ใช้เป็นขาสำหรับรับสัญญาณอินพุทจากเครื่องอ่านรหัสแถบ และใช้สำหรับเป็นสัญญาณควบคุมด้วย

สำหรับไอซี 8749 จะใช้สัญญาณนาฬิกาที่ความถี่ 12 เมกะเฮิร์ท พอร์ต P10..P17 ต่อเข้ากับจอแสดงผลแอลซีดี พอร์ต P20..P27 ใช้เป็นสัญญาณควบคุม ส่วนพอร์ต DB0..DB7 จะใช้เป็นพอร์ตขนาน 8 บิท สำหรับส่งและรับข้อมูลกับอุปกรณ์อื่น ๆ

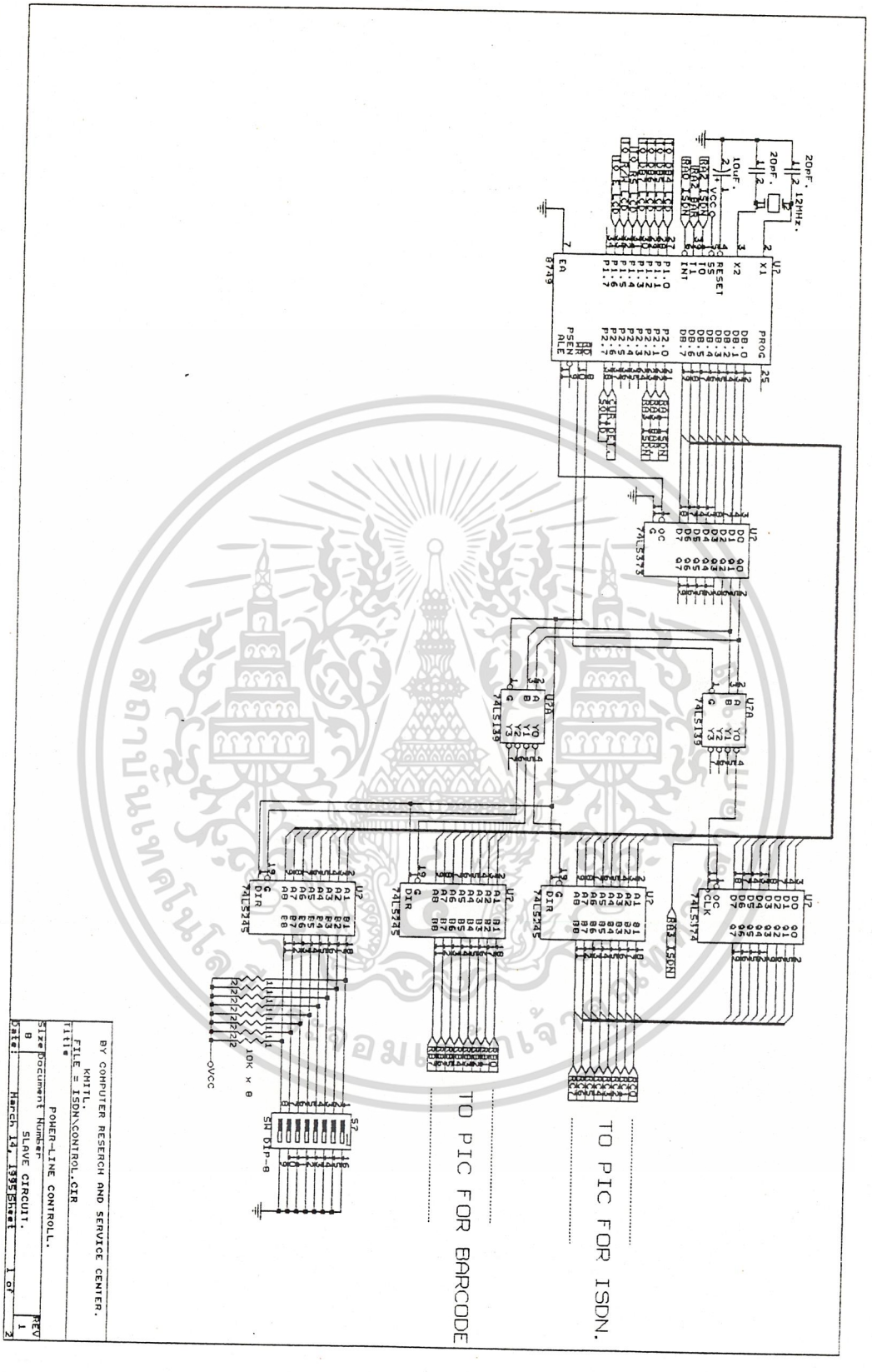


รูปที่ -3.17 แสดงการเชื่อมต่อระหว่าหน่วยประมวลผลกลางกับบอร์ดควบคุมเทอร์มินอล

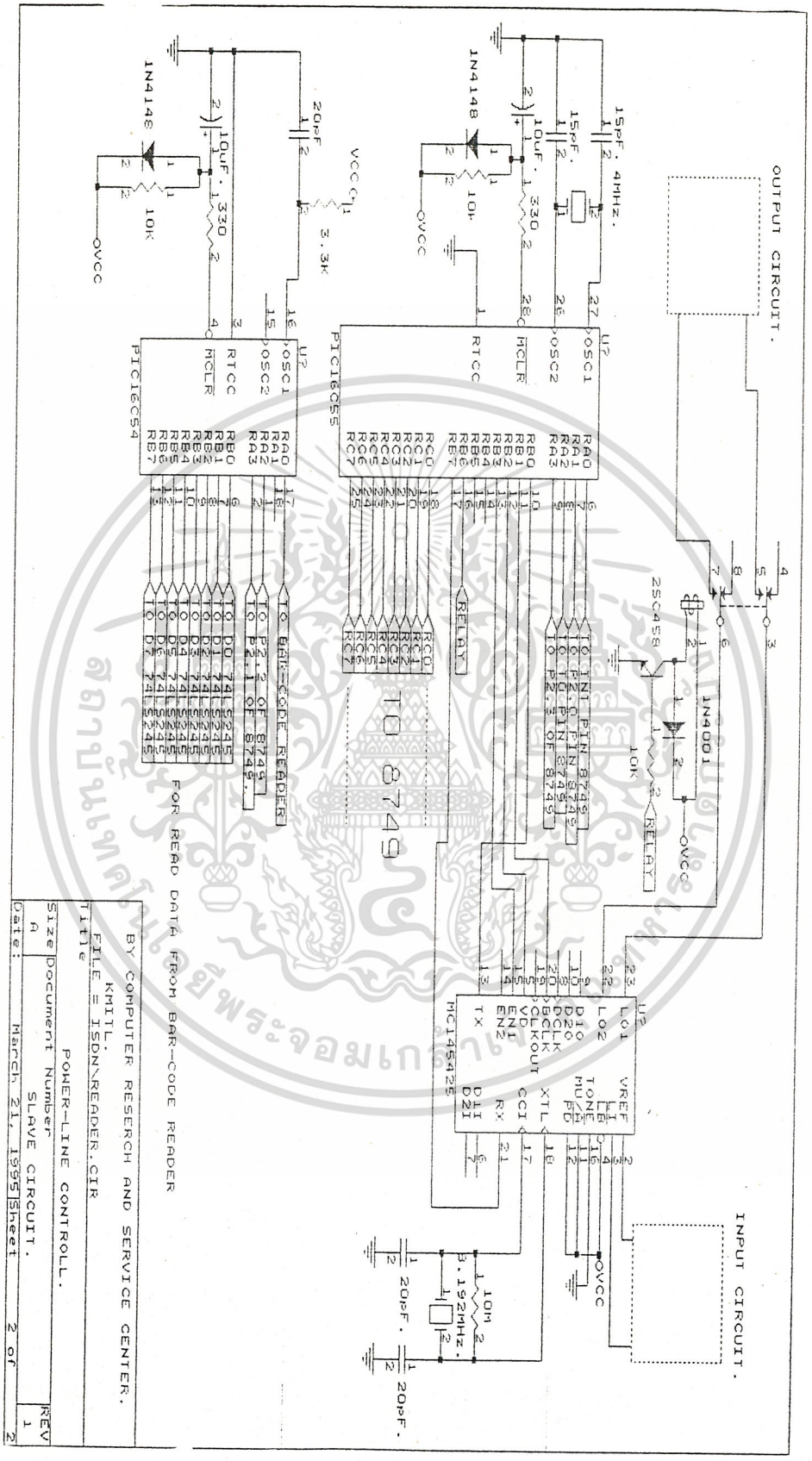
ไอซี PIC16C55 ซึ่งใช้เป็นไอซีสำหรับจัดการเกี่ยวกับแปลงการสื่อสารระหว่างอนุกรมและขนาน จะใช้สัญญาณนาฬิกาความถี่ 4 เมกเฮิร์ท เช่นเดียวกัน โดยที่พอร์ต RA จะใช้เป็นสัญญาณควบคุม ส่วนพอร์ต RB กับไอซี สเลฟ MC145425 ส่วนอีกพอร์ตที่เหลือคือพอร์ต RC จะใช้เป็นพอร์ตขนาน 8 บิท สำหรับติดต่อกับไอซี 8749

ไอซี MC145425 UDLT สำหรับเป็นตัวจัดการการสื่อสารข้อมูลผ่านสายโทรศัพท์ ใช้ตัวกำเนิดความถี่แบบแร่ความถี่ 8 เมกะเฮิร์ท ที่ขา LO1 และ LO2 จะต่อกับรีเลย์ก่อน แล้วจึงต่อกับวงจรที่ใช้เชื่อมต่อกับสายโทรศัพท์ ส่วนขา LI และ Vref จะต่อกับวงจรเชื่อมต่อสายโทรศัพท์เลย สายโทรศัพท์ที่ใช้ในการเชื่อมต่อ จะใช้ขนาด 26 AWG ซึ่งเป็นสายคู่ตีเกลียว ขนาดยาวไม่เกินหนึ่งกิโลเมตร ส่วนจอแสดงผลใช้แบบแอลซีดี ขนาด 20 ตัวอักษร 2 แถว สำหรับติดต่อกับผู้มาขอใช้บริการ

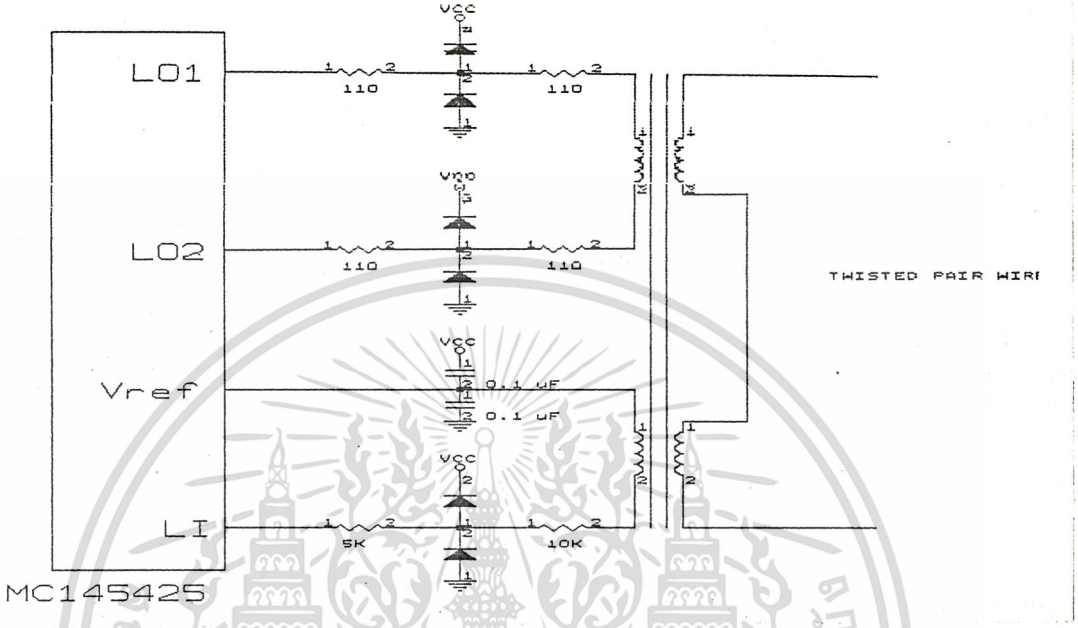
สำหรับอุปกรณ์ที่ใช้สำหรับตัดต่อไฟ 220 โวลต์ เข้ากับเครื่องเทอร์มินอล เป็นโซลิดสเตทรีเลย์ซึ่งควบคุมการตัดต่อโดยไอซี 8749 อีกที



รูปที่ 3.18 วงจรควบคุมบอร์ดสลาฟที่มี 8749 เป็นตัวควบคุม



รูปที่ 3.19 วงจรแสดงการนำ PIC16C55 ต่อกับ MC145425 และการต่อ PIC16C55 กับเครื่องอ่านรหัสแถบ



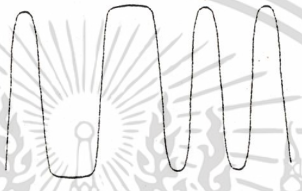
รูปที่ 3.20 วงจรเชื่อมต่อสายคู่ตีเกลียว

บทที่ 4

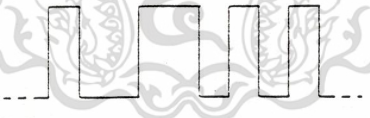
การทดลองและผลการทดลอง

4.1 การทดลอง

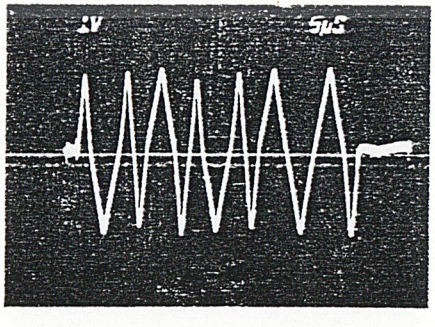
ในการทดลองตรวจสัญญาณอินพุทและสัญญาณเอาท์พุท จากขาสัญญาณต่างๆ ได้ผลการทดลองดังรูป



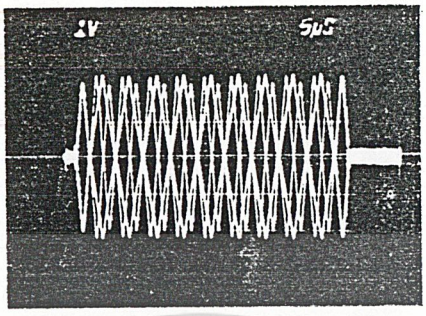
รูป 4.1 รูปสัญญาณที่ได้จากหัวอ่านรหัสแถบ



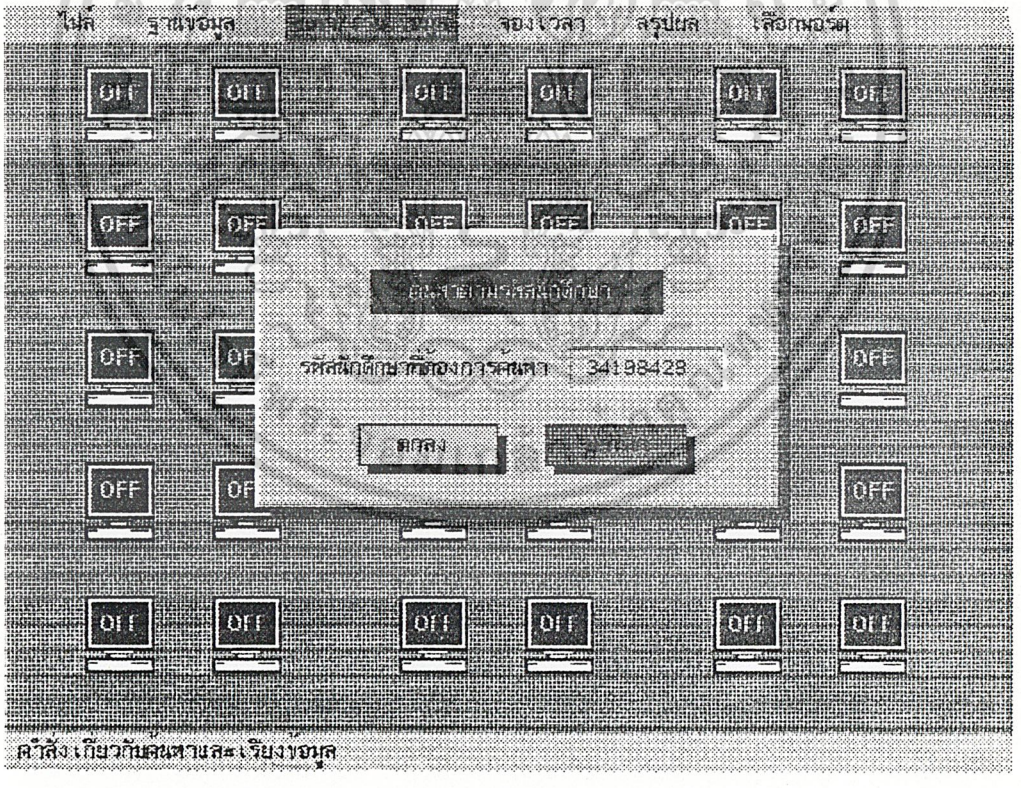
รูป 4.2 รูปสัญญาณที่ได้จากหัวอ่านหลังจากผ่านวงจรเปรียบเทียบแรงดัน



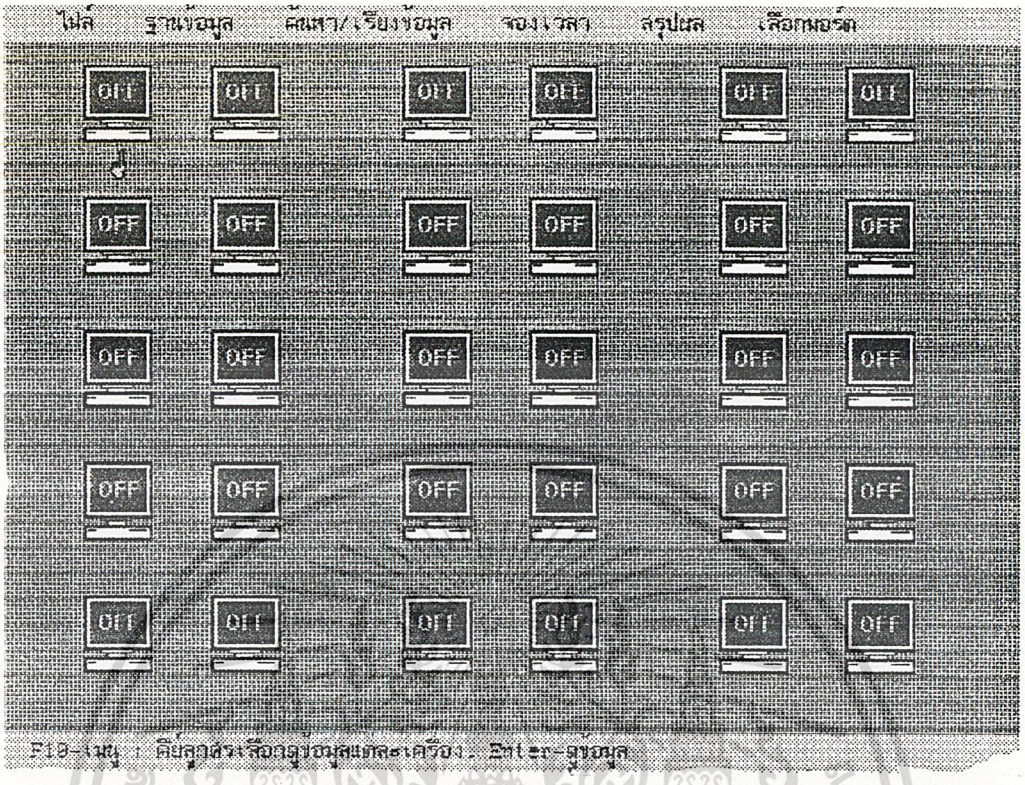
รูป 4.3 รูปสัญญาณที่ขา LI ของ ไอซี MC145425



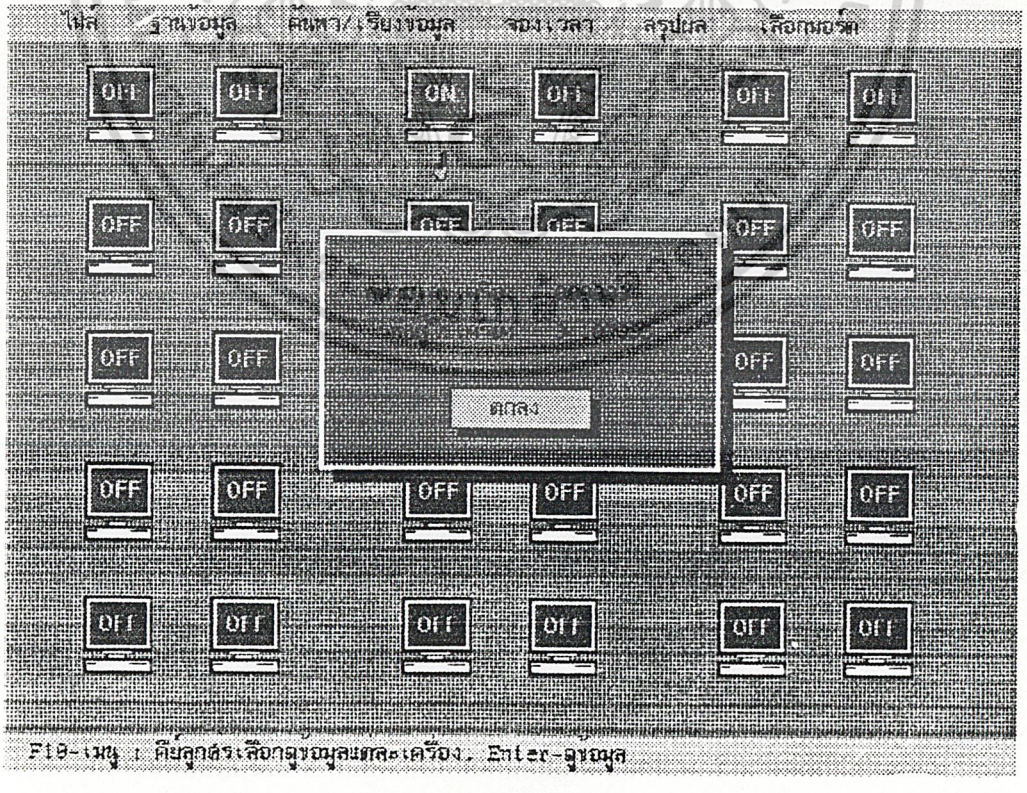
รูปที่ 4.4 สัญญาณที่ขา LO1 ไอซี MC145421



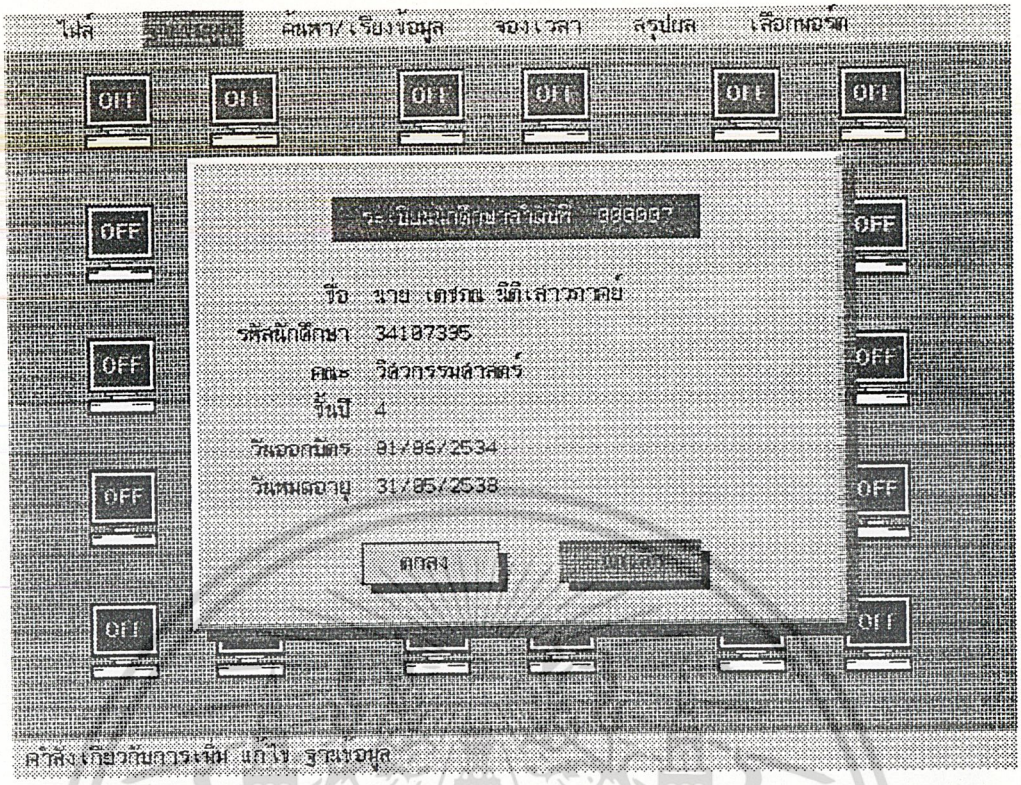
รูปที่ 4.5 หน้าจอขณะทำการป้อนข้อมูลเพื่อค้นหาในฐานข้อมูล



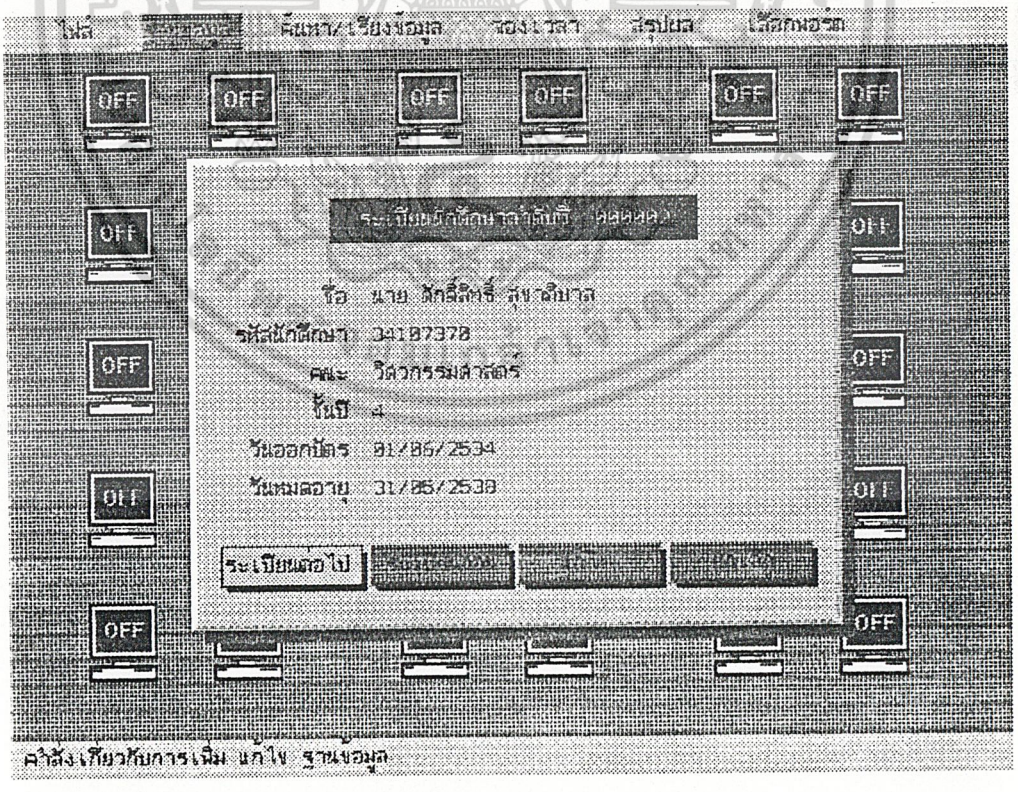
รูปที่ 4.6 แสดงตำแหน่งและสถานะของเครื่องคอมพิวเตอร์แต่ละเครื่อง



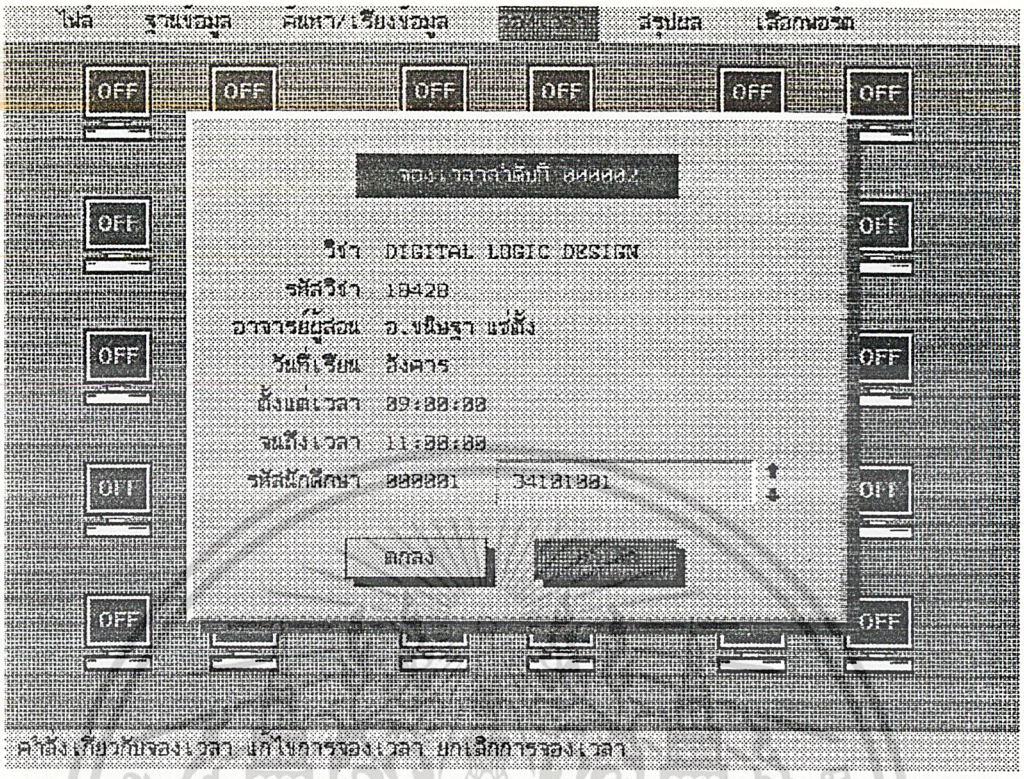
รูปที่ 4.7 แสดงข้อมูลการใช้งานเครื่องคอมพิวเตอร์แต่ละเครื่อง



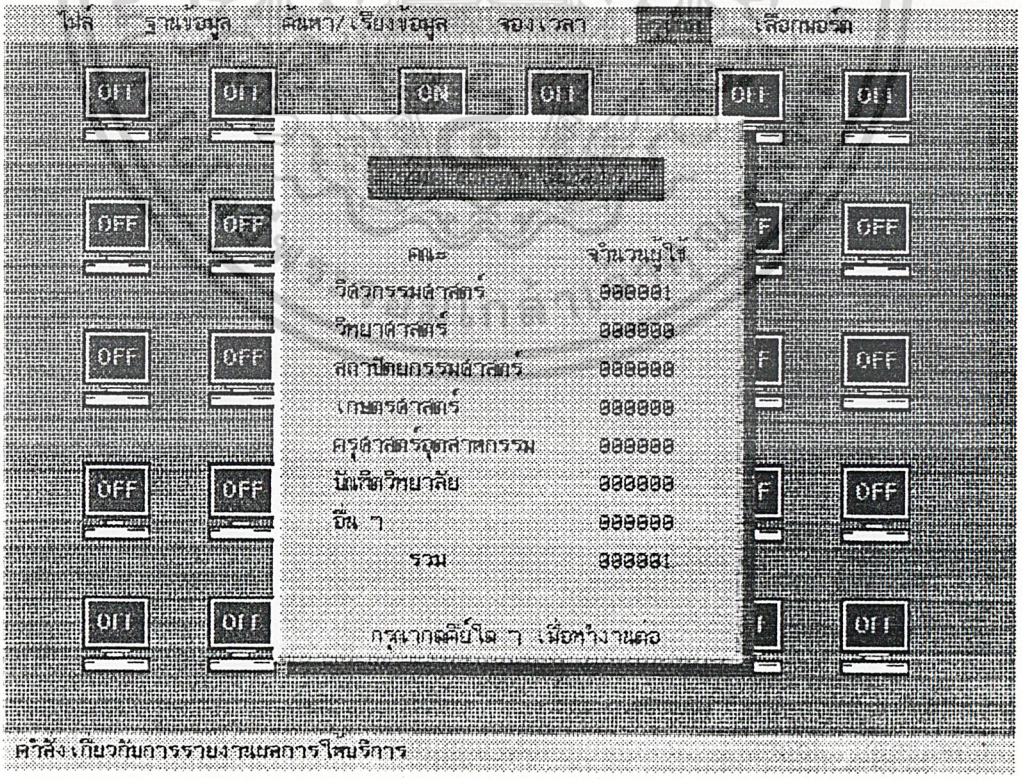
รูปที่ 4.8 แสดงการป้อนข้อมูลนักศึกษาสำหรับเก็บในฐานข้อมูล



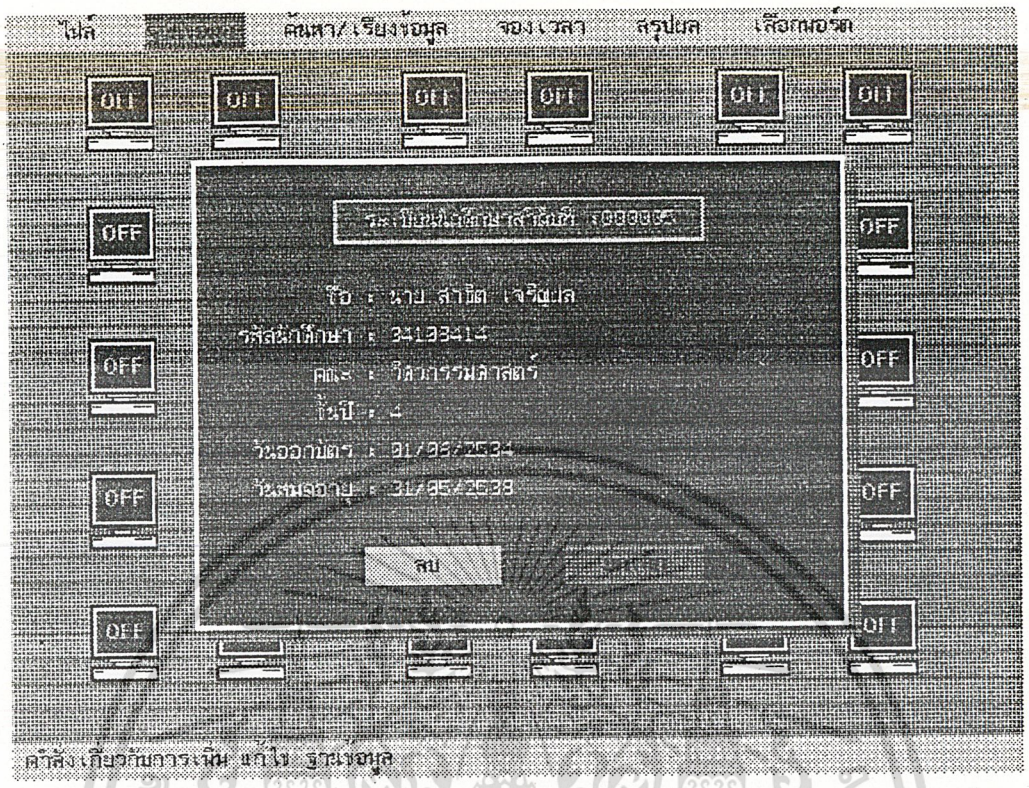
รูปที่ 4.9 แสดงการแก้ไขข้อมูลนักศึกษาในฐานข้อมูล



รูปที่ 4.10 แสดงข้อมูลการจองเวลาเพื่อใช้ห้องคอมพิวเตอร์สำหรับคาบเรียน



รูปที่ 4.11 แสดงรายงานผลการให้บริการเครื่องคอมพิวเตอร์



รูปที่ 4.12 หน้าจอขณะจะทำการลบข้อมูลในฐานข้อมูล

บทที่ 5

บทสรุป ปัญหาและข้อเสนอแนะ

5.1 บทสรุป

ระบบการให้บริการคอมพิวเตอร์ที่สร้างขึ้นมาสามารถทำงานได้ผลในระดับที่น่าพอใจ สามารถจัดการระบบการให้บริการซึ่งจะมีรูปแบบการให้บริการ เป็นคาบเรียน การให้บริการปกติสำหรับนักศึกษาทั่วไป และการให้บริการเป็นกรณีพิเศษนอกเหนือจากเวลาราชการสำหรับการฝึกอบรมบุคลากร

5.2 ปัญหา

1.ด้านฮาร์ดแวร์

1.1 ปัญหาเกี่ยวกับการใช้งานไอซี ISDN ซึ่งยังเป็นสิ่งที่ใหม่ยังไม่เคยมีผู้ศึกษามาก่อนทำให้ต้องเสียเวลาในการศึกษาการทำงานของไอซีพอสมควร และสิ่งที่เป็นปัญหามากคือการประยุกต์ใช้งานไอซี ISDN เพื่อนำมาต่อในรูปแบบของมัลติดรอป ซึ่งมีความยุ่งยากมาก ผู้จัดทำได้แก้ปัญหาโดยการใช้รีเลย์ในภาคสลาฟเข้ามาช่วยในการเชื่อมต่อซึ่งยังมีข้อเสียคือทำให้การทำงานช้าลง ซึ่งตรงนี้ควรจะได้รับการปรับปรุงค้นหาวิธีการอื่นเพื่อให้สามารถทำงานได้เร็วขึ้น

1.2 ปัญหาการไม่ตรงกันของสัญญาณนาฬิกาซึ่งใช้ในการรับส่งข้อมูลซึ่งทำให้สัญญาณที่รับได้มีความผิดพลาด ซึ่งต้องเสียเวลาในการส่งทวนเพื่อแก้ปัญหาข้อผิดพลาดที่เกิดขึ้น ทั้งในแบบซิงโครนัสและแบบอะซิงโครนัส

1.3 ในส่วนของการควบคุมบอร์ดสลาฟ ซึ่งใช้ MCS 8749 ในการควบคุมนั้น จะใช้การควบคุมแอลซีดีแบบสี่บิททำให้การควบคุมค่อนข้างมีปัญหา

1.4 เกี่ยวกับโปรแกรมแอสเซมบลี้ของไอซีไมโครคอนโทรลเลอร์ PIC16C54 และ PIC16C55 ซึ่งค่อนข้างเขียนยากพอสมควร เนื่องจากเป็นไอซีใหม่ ยังไม่เคยนำมาใช้งานด้านนี้มาก่อน โปรแกรมบางส่วนอาจจะยังทำงานไม่สมบูรณ์ คงต้องใช้เวลาพอสมควร เพื่อแก้ไข ให้เสร็จสมบูรณ์มากกว่านี้

1.5 เนื่องจากเป็นระบบที่เพิ่งนำมาติดตั้ง ให้ใช้งานในการให้บริการคอมพิวเตอร์ ผู้มาขอใช้บริการบางคนอาจจะยังไม่คุ้นเคย จึงทำให้การให้บริการมีความล่าช้า ไม่เป็นไปตามที่วางแผนเอาไว้

1.6 บัตรนักศึกษาที่มีรหัสแถบไม่ตรงกับตำแหน่งของเครื่องอ่านรหัสแถบทำให้การอ่านรหัสนักศึกษาผิดพลาด

2.ด้านซอร์ฟแวร์

2.1 ปัญหาเกี่ยวกับการจัดการหน่วยความจำ บ่อยครั้งที่การทำงานของโปรแกรมผิดพลาดเนื่องจากการจัดการหน่วยความจำไม่ดี ทำให้เกิดการซ้อนทับกันของตำแหน่งหน่วยจำของตัวแปร

2.2 เกี่ยวกับโปรแกรมแอสเซมบลี้ของไอซีไมโครคอนโทรลเลอร์ PIC16C54 และ PIC16C55 ซึ่งค่อนข้างเขียนยากพอสมควร เนื่องจากเป็นไอซีใหม่ ยังไม่เคยนำมาใช้งานด้านนี้มาก่อน โปรแกรมบางส่วนอาจจะยังทำงานไม่สมบูรณ์ คงต้องใช้เวลาพอสมควร เพื่อแก้ไข ให้เสร็จสมบูรณ์มากกว่านี้

5.3 ข้อเสนอแนะ

1.ด้านฮาร์ดแวร์

1.1 การส่งข้อมูลระหว่างมาสเตอร์กับสลาฟจะใช้ลักษณะการพอลลิ่งเพื่อแก้ปัญหามัลติดรอปที่เกิดขึ้นให้น้อยลง

1.2 การควบคุมข้อผิดพลาดในการส่งข้อมูลทั้งแบบซิงโครนัสและแบบอะซิงโครนัส สามารถแก้ไขโดยกำหนดให้มีการเพิ่มการตรวจสอบผลรวม (check sum) ในโปรโตคอลที่กำหนดขึ้นมา หากพบว่ามีการส่งข้อมูลที่ผิดพลาด ก็ให้ส่งไปฝ่ายส่งส่งมาอีกครั้ง

1.3 การติดรหัสแถบของนักศึกษาควรจะให้เป็นมาตรฐานเดียวกันทั้งหมด

1.4 การเชื่อมต่อ โซลิดสเตทรีเลย์ กับสายไฟ 220 โวลต์ และ กับวงจรขับรีเลย์ ควรตรวจสอบให้ดี เพราะอาจจะทำให้เกิดปัญญาภายหลังได้

2.ด้านซอร์ฟแวร์

2.1 ในการใช้งานพ้อยเตอร์ต้องใช้อย่างระมัดระวัง บ่อยครั้งที่พบว่ามีการให้ค่าแก่ตัวแปรพ้อยเตอร์ผิดพลาดทำให้การทำงานของโปรแกรมผิดพลาด โดยเฉพาะถ้าหากเป็นตัวแปรพ้อยเตอร์ชนิด Far ถ้าหากโชคไม่ดีเกิดค่าในพ้อยเตอร์นั้นชี้ไปยังหน่วยความจำส่วนที่ระบบปฏิบัติการใช้งานอยู่และมีการเขียนข้อมูลลงไปในหน่วยความจำส่วนนั้นจะทำให้เครื่องหยุดการทำงานได้

2.2 ในการโปรแกรมใช้งานอินเทอร์รัพต์นั้น ผู้โปรแกรมต้องกระทำอย่างระมัดระวัง เนื่องจากต้องมีการเข้าไปแก้ไขอินเทอร์รัพต์เวกเตอร์ในตารางอินเทอร์รัพต์ของระบบปฏิบัติการซึ่งต้องระวังอย่างมากถ้าหากผิดพลาดจะทำให้ระบบเกิดหยุดการทำงานได้

ภาคผนวก

```
#include <graphics.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <ctype.h>
#include <math.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
/*pulldown menu colors*/
#define RIMCOLOR     GREEN
#define HOTKEY       RED
#define HOTKEY_R     LIGHTRED
#define NORLMENU     BLACK
#define NORLMENU_R BLACK
#define  MAINBKG_1      LIGHTGRAY
#define MAINBKG_2      LIGHTCYAN
#define MAINBKG_R      LIGHTGREEN
#define DESATT         BLUE
#define DESATT_R       RED
#define  BKGPULLD      WHITE
#define BKGPULLD_R LIGHTGREEN
/*popup window colors*/
#define MENUSHA        DARKGRAY
#define POPUPBKG       LIGHTGRAY
#define  RIMUPPER      WHITE
#define RIMLOWER     DARKGRAY
#define POPUPSHA       LIGHTMAGENTA
#define ICONSHA      BLACK
#define ICONORM        LIGHTGREEN
#define  ICONORIM      GREEN
#define ICONTEXT       BLUE
#define ICONACTIVE   YELLOW
#define ICONACRIM    RED
#define BKSP           8
#define TABKEY         9
#define ENTER          13
#define SHFTAB         15
#define ESC            27
#define HOME           71
#define UP             72
#define  LEFT          75
#define RIGHT          77
#define END            79
#define DOWN           80
#define  DELETE        83
#define Alt_A     0x1E
#define Alt_K     0x25
#define Alt_S     0x7F
#define Alt_0     0x81
#define Alt_R   0x13
#define Alt_V   0x2F
#define Alt_X   0x2D
#define F10     0x44
#define ENG     0
#define THA     1
#define FAIL    0
#define PASS      1
#define FALSE 0
#define TRUE     1
#define NOTFOUND    (-1)
#define USERCANCEL (-2)
#define MAX_MENU    6
#define DELCODE    'D'
int AppendData(void);
void Border(int leftx,int lefty,int rightx,int righty,int thick,int color);
void Blockfill(int leftx,int lefty,int rightx,int righty,int color);
char *Convertdate(date_t *srcdate);
char *CopyFromPos(int npos,unsigned char *s);
int Creatfile(FILE **fptr,char *fname,char *Textheader);
void DatabaseDialogue(void);
int Daytoint(char *d);
void Definemenu(void);
void Dimension(int leftx,int lefty,int rightx,int righty,int thick,int color1,int
color2);
void DisplayChar(int col,int row,unsigned unsigned char ascii,int color);
void DisplayIdRsvnumber(unsigned long int recnum);
void DisplayPulldown(int num);
void DisplayString(int x,int y,unsigned char *text,int color1,int color2);
void DisplayStudentData(void);
void DosShell(void);
void DrawReserveDialogue(void);
void ErrorDialogue(char *errmessage);
void ExitProgram(void);
void FillLine(int leftx,int lefty,int rightx,int righty,int color);
void getcurpath(void);
int Getdate(int xpos,int ypos,char inputdate[8]);
char *GetEngString(int startx,int starty,int MaxStrLen,int LineLength,int
Tcolor,int Bcolor);
int GetmodifyRecord(void);
int GetThaiString(int startx,int starty,int MaxStrLen,int LineLength,int
Tcolor,int Bcolor);
int GetReserveMember(FILE *memberftemp,long *netstudent,long *
totalstudent);
char *GetStudentId(FILE *fp,unsigned long int record);
int Gettime(int xpos,int ypos,char inputtime[9]);
void Getvideo(int leftx,int lefty,int rightx,int righty,char far *buffer);
unsigned unsigned char GetThaikey(unsigned unsigned char ch);
void Initialize(void);
void InitialStudData(void);
int InitSerialComm(int,int,int,int,int);
void interrupt (*old1)(),interrupt (*old2)();
void inttoDay(int d);
int is_in(unsigned char *s,unsigned char c);
```

```
void LogMensuBuffer(void);
void MakeMainmenu(int num,unsigned char *menu[],unsigned char *
keys,int count,int startx,int starty);
void MakeOneLineChar(int length,unsigned char *str);
void MenuManage(void);
int ModifyData(void);
int Openfile(int mode,unsigned long int *Nrecord);
void OpenDatafile(void);
void PutCenterString(int startx,int endx,int starty,unsigned char *s,int color);
void Putvideo(int leftx,int lefty,int rightx,int righty,char far *buffer);
int QuickDisk(FILE *fp,unsigned long int totalrecord);
int Qsdisk(FILE *fp,unsigned long int left,unsigned long int right);
void QuitSerialComm(void);
void ResetBuffer(void);
void ReadPont(void);
void ReserveTime(void);
long int Search(FILE *fp,char *StudentId,unsigned long int totalrecord);
void SearchDialogue(void);
void Shadow(int leftx,int lefty,int rightx,int righty,int bthick,int thick,int
color);
void SortDialogue(void);
int Sort(FILE *fp,unsigned long int totalrecord);
int Stringlength(unsigned char *text);
void Swapfield(FILE *fp,unsigned long int i,unsigned long int j);
void WritePixel(int x,int y,int color);
int xmit(unsigned char);
typedef struct {
   char day;
   char month;
   int year;
   } date_t;
typedef struct {
   char Name[30];
   char StudentId[9];
   char Faculty[30];
   char Year;
   date_t RegisDate;   //dd-mm-yy
   date_t ExpirDate;
   } StudentRecord_t;
typedef struct {
   char SubjectId[9];
   char TeacherName[40];
   char Day;
   char BeginTime[9];
   char EndTime[9];
   } Reserveheader_t;
typedef struct {
   char StudentId[9];
   char SubjectId[9];
   } Reservedata_t;
struct menu_info {
   int startx,starty,endx,endy;
   unsigned unsigned char *p;
   unsigned char **menu;
   unsigned char *keys;
   int count;
   } menu_num[MAX_MENU];
FILE *StudentIdFile;
FILE *ReserveHeaderFile;
FILE *ReserveIdFile;
StudentRecord_t StudentRecord;
Reserveheader_t ReserveHeader;
Reservemember_t ReserveMember;
char    NodeID;
char    *CommBuf;
int     CommAddr;
int     installed = 0;
unsigned CommDataPos = 0;
unsigned ReceiveFlag = 0; //OFF
unsigned baud[5] = { 96,48,24,12,6 }; /* for 1200,2400,4800,9600 &
19200 bps */
char Data[2] = { 0x03,0x02 };        /* feature bit of 8-bit , 7-bit */
char stop[2] = { 0x00,0x04 };        /* feature bit of 1-bit , 2-bit */
char pari[3] = { 0x00,0x08,0x18 };   /* feature bit of none,odd,even */
char rtime[9]=" : : ",xtime[9]=" : : ";
char rdate[11]=" / /   ",xdate[11]=" / /   ";
unsigned long int TotalRecord=0L;
unsigned long int TotalReserve=0L;
unsigned long int TotalReserveStudent=0L;
unsigned long int CurrentRecord=0L;
unsigned long int CurrentReserve=0L;
char std_datfile[127];
char rsv_headfile[127];
char rsv_stdfile[127];
unsigned char far *vidmem=(unsigned char far *)0xA0000000;
unsigned char Font[5376];
unsigned char InputString[127]="";
unsigned char CurrentPath[127]="";
int Language=ENG;
int main_menu_startx[MAX_MENU]={4,11,22,39,50,63};
int main_menu_length[MAX_MENU]={3,7,13,7,9,8};
unsigned char far *mainmenu_buf=NULL;
unsigned char *main_menu[]={
   "ไ%ฟล์",
   "ฐา%นข้อมูล",
   "%ค้นหา/เรียงข้อมูล",
   "%จองเวลา",
   "%พิมพ์รายงาน",
   "เส%อกพอร์ต",
   };
unsigned char *mainkey="ฟ่าคงพอ";
unsigned char *File[]={
   " %บันทึกข้อมูลในชื่ออื่น ",
   " %ออกไปดอสชั่วคราว ",
   " เ%ลิกการทำงาน   Alt+X ",
   };
```

```
unsigned char *Database[]={
   " เพิ่มระเบียนใหม่ ",
   "แก้ไขข้อมูลเก่า ",
   " ลบข้อมูลเดิม ",
   };
unsigned char *Search_sort[]={
   " ค้นหาข้อมูลจากแฟ้มข้อมูล ",
   " เรียงข้อมูลในแฟ้มข้อมูล ",
   };
unsigned char *Reserve[]={
   " จองเวลา",
   " ข้อมูลเกี่ยวกับการจอง ",
   " ยกเลิกการจอง ",
   };
unsigned char *Report[]={
   " พิมพ์รายงาน ",
   };
unsigned char *Setserial[]={
   "ใช้พอร์ตหนึ่ง ",
   "ใช้พอร์ตสอง ",
   };
unsigned char *describ[]={
   " คำสั่งเกี่ยวกับการจัดการไฟล์และระบบ",
   " คำสั่งเกี่ยวกับการเพิ่มแก้ไขฐานข้อมูล",
   " คำสั่งเกี่ยวกับการค้นหาและเรียงข้อมูล",
   " คำสั่งเกี่ยวกับการจองเวลา แก้ไขเวลาการจองเวลา",
   " คำสั่งเกี่ยวกับการพิมพ์รายงาน",
   " เลือกพอร์ตอนุกรมที่จะใช้งาน",
   };
void Initialize(void)
{
   int gdriver, gmode, errorcode;
   detectgraph(&gdriver,&gmode);
   if( gdriver != 9)
   {
      printf("This program must run in 640x480 VGA mode\n");
      exit(0);
   }
   initgraph(&gdriver, &gmode,"");
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      exit(1);
   }
   ReadFont();
}
void ExitProgram(void)
{
   farfree(mainmenu_buf);
   fseek(StudentIdFile,9,SEEK_SET);
   fwrite(&TotalRecord,sizeof(unsigned long),1,StudentIdFile);
   fclose(StudentIdFile);
   fseek(ReserveHeaderFile,9,SEEK_SET);
   fwrite(&TotalReserve,sizeof(unsigned long),1,ReserveHeaderFile);
   fclose(ReserveHeaderFile);
   fseek(ReserveIdFile,9,SEEK_SET);
   fwrite(&TotalReserveStudent,sizeof(unsigned long),1,ReserveIdFile);
   fclose(ReserveIdFile);
   closegraph();
}
void ReadFont(void)
{
   FILE *font_fp;
   if( (font_fp=fopen("font1.fon","rb"))==NULL )
   {
      printf("Cannot open file font1.fon\n");
      exit(1);
   }
   fseek(font_fp,768L,SEEK_SET);
   fread((unsigned char *)Font,sizeof(Font),1,font_fp);
   fclose(font_fp);
}
void Getvideo(int leftx,int lefty,int rightx,int righty,char far *buffer)
{
   char far *t;
   register unsigned int x,y,plane;
   if(leftx<0||lefty<0||rightx>79||righty>19) return;
   lefty*=24;righty=(righty*24)+23;
   outportb(0x3CE,0x04);//select control register to read
   for(y=lefty;y<=righty;y++)
   {
      t=vidmem+y*80+leftx;
      for(x=leftx;x<=rightx;x++)
      {
         outportb(0x3CF,0x00);//select plane 0
         *buffer++ = *t;
         outportb(0x3CF,0x01);//select plane 1
         *buffer++ = *t;
         outportb(0x3CF,0x02);//select plane 2
         *buffer++ = *t;
         outportb(0x3CF,0x03);//select plane 3
         *buffer++ = *t++;
      }
   }
}
void Putvideo(int leftx,int lefty,int rightx,int righty,char far *buffer)
{
   char far *t;
   register unsigned int x,y,plane;
   if(leftx<0||lefty<0||rightx>79||righty>19) return;
   lefty*=24;righty=(righty*24)+23;
   outportb(0x3C4,0x02);//select sequencial register to write
   for(y=lefty;y<=righty;y++)
   {
      t=vidmem+y*80+leftx;
```

```
    for(x=leftx;x<=rightx;x++)
    {
      outportb(0x3C5,0x01);//select plane 0
      *t=*buffer++;
      outportb(0x3C5,0x02);//select plane 1
      *t=*buffer++;
      outportb(0x3C5,0x04);//select plane 2
      *t=*buffer++;
      outportb(0x3C5,0x08);//select plane 3
      *t++=*buffer++;
    }
  }
}
void WritePixel(int x,int y,int color)
{
  unsigned char far *t;
  register int plane,bit_position;
  t=vidmem+y*80+x/8;
  bit_position=0x01<<(7-(x%8));
  outportb(0x3CE,0x04);//select control register
  outportb(0x3C4,0x02);//select sequencial register
  for (plane=0;plane<=3;plane++)
  {
    outportb(0x3CF,plane); //select plane 0-3 to read;
    outportb(0x3C5,0x01<<plane); //select plane 0-3 to write;
    if (color&0x01)
      *t=((*t)|bit_position);
    else  *t=((*t)&~bit_position);
    color=color>>1;
  }
}
/*** top left of screen is (0,0)      top right of screen is (79,0)
    bottom left of screen is (0,19) bottom right of screen is (79,19) ***/
void DisplayChar(int col,int row,unsigned unsigned char ascii,int color)
{
  int x,y;
  unsigned unsigned char bitset=0x01;
  if( col>79 || row > 19 || ascii<32 )
    return;
  else ascii-=32;
  col*=8;row*=24;
  for(y=0;y<24;y++)
  {
    bitset=0x01;
    if(Font[ascii*24+y])
    {
      for(x=8;x>0;x--)
      {
        if( Font[ascii*24+y] & bitset )
        {
          WritePixel(x+col,y+row,color);
        }
        bitset<<=1;
      }
    }
  }
}
/*** top left of screen is (0,0)    top right of screen is (79,0)
    bottom left of screen is (0,19)  bottom right of screen is (79,19) ***/
void DisplayString(int x,int y,unsigned char *text,int color1,int color2)
{
  int color;
  while(*text)
  {
    if( *text=='%' )
    {
      text++; color=color2;
    }
    else color=color1;
    switch(*text)
    {
      /*case use unsigned char*/
      case 215: case 216: case 217: case 218: case 219:
      case 220: case 221: case 222: case 223: case 224:
      case 225: case 226: case 227:
      case 228: --x;break;
    }
    DisplayChar(x++,y,*text++,color);
    if(x==80) return;
  }
}
int Stringlength(unsigned char *text)
{
  unsigned int length=0;
  while(*text)
  {
    switch(*text++)
    {
      case 215: case 216: case 217: case 218: case 219:
      case 220: case 221: case 222: case 223: case 224:
      case 225: case 226: case 227:  case 228:
      case '%': break;
      default : length++;
    }
  }
  return length;
}
void PutCenterString(int startx,int endx,int starty,unsigned char *s,int color)
{
  int len;
  len=Stringlength(s)/2;
  endx=(endx+startx)/2;
  endx-=len;
  startx=endx;
  DisplayString(startx,starty,s,color,color);
}
```

```
/*** this function accept parameter with 640x480 direct value ***/
void FillLine(int leftx,int lefty,int rightx,int righty,int color)
{
  register int x,y;
  if(leftx<0||lefty<0||rightx>639||righty>479) return;
  for(y=lefty;y<righty;y++)
    for(x=leftx;x<rightx;x++)
      WritePixel(x,y,color);
}
/*** top left of screen is (0,0)     top right of screen is (79,0)
     bottom left of screen is (0,19)  bottom right of screen is (79,19) ***/
void Blockfill(int leftx,int lefty,int rightx,int righty,int color)
{
  unsigned char far *t;
  register int x,y,i;
  if(leftx<0||lefty<0||rightx>79||righty>19) return;
  lefty*=24;righty=(righty*24)+24;
  color=color&0x0F;
  outportb(0x3C4,0x02); //select sequencial write plane reg.
  outportb(0x3C5,0x0F); //enable to write plane 0-3
  for(y=lefty;y<righty;y++)
  {
    t=vidmem+y*80+leftx;
    for(x=leftx;x<=rightx;x++)
    {
      outportb(0x3CE,0x01); //select set/reset enable register.
      outportb(0x3CF,0x0F); //set plane enable in set/reset enable reg.
      outportb(0x3CE,0x00); //select set/reset register.
      outportb(0x3CF,color&0x0F);//load color into set/reset reg.
      *t=*t++;
    }
  }
  outportb(0x3CE,0x01); //select set/reset enable register.
  outportb(0x3CF,0x00); //disable all plane for set/reset operation.
}
void Dimension(int leftx,int lefty,int rightx,int righty,int thick,int color1,int
color2)
{
  register int x,y,i;
  leftx=leftx*8-thick;rightx=rightx*8+7+thick;
  lefty=lefty*24-thick;righty=(righty*24)+23+thick;
  if(leftx<0||lefty<0||rightx>=640||righty>=480) return;
  color1=color1&0x0F; color2=color2&0x0F;
  /*upper x axis*/
  for(i=0;i<thick;i++)
    for(x=leftx;x<=rightx-i;x++)
      WritePixel(x,lefty+i,color1);
  /*lower x axis*/
  for(i=0;i<thick;i++)
    for(x=leftx+i;x<=rightx;x++)
      WritePixel(x,righty-i,color2);
  /*left y axis*/
  for(i=0;i<thick;i++)
    for(y=lefty;y<=righty-i;y++)
      WritePixel(leftx+i,y,color1);
  /*right y axis*/
  for(i=0;i<thick;i++)
    for(y=lefty+i;y<=righty;y++)
      WritePixel(rightx-i,y,color2);
}
void Border(int leftx,int lefty,int rightx,int righty,int thick,int color)
{
  register int x,y,i;
  leftx=leftx*8-thick;rightx=rightx*8+7+thick;
  lefty=lefty*24-thick;righty=(righty*24)+23+thick;
  if(leftx<0||lefty<0||rightx>=640||righty>=480) return;
  color=color&0x0F;
  for(x=leftx;x<=rightx;x++)
  {
    for(i=0;i<thick;i++)
    {
      WritePixel(x,lefty+i,color);
      WritePixel(x,righty-i,color);
    }
  }
  for(y=lefty;y<=righty;y++)
  {
    for(i=0;i<thick;i++)
    {
      WritePixel(leftx+i,y,color);
      WritePixel(rightx-i,y,color);
    }
  }
}
void Shadow(int leftx,int lefty,int rightx,int righty,int bthick,int thick,int
color)
{
  unsigned char far *t;
  int x,y;

  if(leftx<0 || lefty<0 || rightx>79 || righty>19) return;
  leftx=leftx*8-bthick;
  rightx=rightx*8+7+bthick;
  lefty=lefty*24-bthick;
  righty=righty*24+23+bthick;
  for(x=leftx+thick+1;x<=rightx+thick;x++)
    for(y=1;y<=thick;y++)
      WritePixel(x,righty+y,color);
  for(y=lefty+thick+2;y<=righty+thick;y++)
    for(x=1;x<=thick;x++)
      WritePixel(rightx+x,y,color);
}
unsigned unsigned char GetThaikey(unsigned unsigned char ch)
{
  switch (ch)
  {
```

```
case 'a': ch=189; break;
case 'b': ch=217; break;
case 'c': ch=209; break;
case 'd': ch=161; break;
case 'e': ch=207; break;
case 'f': ch=178; break;
case 'g': ch=208; break;
case 'h': ch=225; break;
case 'i': ch=193; break;
case 'j': ch=224; break;
case 'k': ch=206; break;
case 'l': ch=199; break;
case 'm': ch=181; break;
case 'n': ch=220; break;
case 'o': ch=183; break;
case 'p': ch=192; break;
case 'q': ch=213; break;
case 'r': ch=188; break;
case 's': ch=200; break;
case 't': ch=204; break;
case 'u': ch=218; break;
case 'v': ch=202; break;
case 'w': ch=212; break;
case 'x': ch=185; break;
case 'y': ch=221; break;
case 'z': ch=186; break;
case 'A': ch=194; break;
case 'B': ch=222; break;
case 'C': ch=167; break;
case 'D': ch=173; break;
case 'E': ch=172; break;
case 'F': ch=210; break;
case 'G': ch=170; break;
case 'H': ch=223; break;
case 'I': ch=177; break;
case 'J': ch=227; break;
case 'K': ch=198; break;
case 'L': ch=197; break;
case 'M': ch=63; break;
case 'N': ch=228; break;
case 'O': ch=214; break;
case 'P': ch=171; break;
case 'Q': ch=129; break;
case 'R': ch=175; break;
case 'S': ch=164; break;
case 'T': ch=182; break;
case 'U': ch=226; break;
case 'V': ch=203; break;
case 'W': ch=34; break;
case 'X': ch=41; break;
case 'Y': ch=222; break;
case 'Z': ch=40; break;
case ',': ch=191; break;
case '.': ch=211; break;
case '/': ch=187; break;
case '<': ch=176; break;
case '>': ch=201; break;
case '?': ch=207; break;//unuse
case ';': ch=196; break;
case '\': ch=165; break;
case ':': ch=169; break;
case '"': ch=46; break;
case '[': ch=184; break;
case ']': ch=195; break;
case '{': ch=174; break;
case '}': ch=44; break;
case '1': ch=195; break;
case '2': ch=47; break;
case '3': ch=95; break;
case '4': ch=190; break;
case '5': ch=180; break;
case '6': ch=215; break;
case '7': ch=219; break;
case '8': ch=163; break;
case '9': ch=179; break;
case '0': ch=166; break;
case '-': ch=162; break;
case '=': ch=168; break;
case '\': ch=207; break;//unuse
case '!': ch=46; break;
case '@': ch=129; break;
case '#': ch=130; break;
case '$': ch=131; break;
case '%': ch=132; break;
case '^': ch=216; break;
case '&': ch=207; break;
case '*': ch=133; break;
case '(': ch=134; break;
case ')': ch=135; break;
case '_': ch=136; break;
case '+': ch=137; break;
case '|': ch=207; break;// unuse
}
return ch;
}
void Definemenu(void)
{
MakeMainmenu(0,File,"ไฟล์",3,4,1);
MakeMainmenu(1,Database,"ฐานข้อมูล",3,11,1);
MakeMainmenu(2,Search_sort,"ค้นหา",2,22,1);
MakeMainmenu(3,Reserve,"สำรองข้อมูล",3,39,1);
MakeMainmenu(4,Report,"รายงาน",1,50,1);
MakeMainmenu(5,Setserial,"ตั้งค่า",2,63,1);
//MAX_MENU=6
}
```

```
void MakeMainmenu(int num,unsigned char *menu[],unsigned char *
keys,int count,
      int startx,int starty)
{
  int i,len;
  int endx,endy;
  len=0;
  for(i=0;i<count;i++)
    if(len<Stringlength(menu[i])) len=Stringlength(menu[i]);
  endx=startx+len-1;
  endy=starty+count-1;
  if( (endy>19)||(endx>79) )
  {
    printf("Size of menu error\n");
    exit(1);
  }
  menu_num[num].startx=startx;
  menu_num[num].endx=endx;
  menu_num[num].starty=starty;
  menu_num[num].endy=endy;
  menu_num[num].menu=(unsigned char **)menu;
  menu_num[num].keys=keys;
  menu_num[num].count=count;
}
void LogMenuBuffer(void)
{
  int large_menu=0;
  unsigned long mainmenu_bsize=0;
  unsigned long i,j;
  for(i=0;i<MAX_MENU;i++)
  {
    if( ((menu_num[i].endx-menu_num[i].startx)*(menu_num[i].endy-
menu_num[i].starty)) >mainmenu_bsize )
    {
      mainmenu_bsize=((menu_num[i].endx-menu_num[i].startx)*
(menu_num[i].endy-menu_num[i].starty));
      large_menu=i;
    }
  }
  i=menu_num[large_menu].endx-menu_num[large_menu].startx+4;
  i*=8; i+=2;
  j=menu_num[large_menu].endy-menu_num[large_menu].starty+3;
  j *= 24;j += 2;
  mainmenu_bsize=i*j/2;
  mainmenu_buf=(void far *)farmalloc(mainmenu_bsize);
  if( !mainmenu_buf )
  {
    closegraph();
    exit(1);
  }
}
void DisplayPulldown(int num)
{
  unsigned char **m;
  int i;
  m=menu_num[num].menu;
  //Save video reserve for menu and shadow area(+2,+1)
  getimage(menu_num[num].startx*8-2,menu_num[num].starty*24-2,
      (menu_num[num].endx+2)*8,(menu_num[num].endy+2)*24,
mainmenu_buf);
  Blockfill(menu_num[num].startx,menu_num[num].starty,
      menu_num[num].endx,menu_num[num].endy,BKGPULLD);
  Border(menu_num[num].startx,menu_num[num].starty,
      menu_num[num].endx,menu_num[num].endy,2,RIMCOLOR);
  Shadow(menu_num[num].startx,menu_num[num].starty,
      menu_num[num].endx,menu_num[num].endy,2,6,MENUSHA);
  for(i=0;i<menu_num[num].count;i++)
    DisplayString(menu_num[num].startx,i+1,m[i],
NORLMENU,HOTKEY);
}
int is_in(unsigned char *s,unsigned char c)
{
  int i;
  for(i=0;*s;i++)
    if(*s++==c) return i+1;
  return 0;
}
char *CopyFromPos(int npos,unsigned char *s)
{
  int i=0;
  unsigned char temps[80]="";
  while( s[npos] ) temps[i++] = s[npos++];
  temps[i]='\x0';
  return (temps);
}
void MakeOneLineChar(int length,unsigned char *str)
{
  int len=0;
  while(*str)
  {
    if(len==length)   *str='\x0';
    else
    {
      if(is_in("ฺ็่้๊๋ัิีึืุู",*str++));
      else len++;
    }
  }
}
char *GetEngString(int startx,int starty,int MaxStrLen,int LineLength,int
Tcolor,int Bcolor)
{
  union inkey {
    char ch[2];
    int i;
    } c;
  unsigned char *tempstring="";
```

```
int leftx=0,curx=0,Stringlen=0,StrModPos=0;
int i;

MaxStrLen-=1;  LineLength-=1;
DisplayChar( (curx+startx),starty,230,Tcolor);
for(;;)
{
  while(!bioskey(1)) ;
  c.i=bioskey(0);
  if(c.ch[0])                                                    ;
  {
    if(Language==THA) c.ch[0]=GetThaikey(c.ch[0]);
    switch(c.ch[0])
    {
      case ESC:  InputString[0]='\x0';return(InputString);
      case ENTER:
      {
        DisplayChar( (curx+startx),starty,230,Bcolor);
        InputString[Stringlen]='\x0';                                Bcolor);
        return(InputString);
      }
      case BKSP :
      {
       if(curx==0)
       {
        sound(2000);delay(200);nosound();break;
       }
       else if( StrModPos<Stringlen )
       {
         InputString[Stringlen]='\x0';
         tempstring=CopyFromPos(StrModPos,InputString);
         InputString[StrModPos-1]='\x0';
         strcat(InputString,tempstring);
         Stringlen--;StrModPos--;

         Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
         if(leftx)
         {
           leftx--;
           tempstring=CopyFromPos(leftx,InputString);
           MakeOneLineChar(LineLength+1,(char *)tempstring);
           DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
           DisplayChar( (curx+startx),starty,230,Tcolor);
         }
         else
         {
           tempstring=CopyFromPos(0,InputString);
           MakeOneLineChar(LineLength+1,tempstring);
           DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
           DisplayChar( (curx+startx),starty,230,Bcolor);
           curx--;
           DisplayChar( (curx+startx),starty,230,Tcolor);
         }
       }
       else if( leftx )
       {
         Stringlen--;StrModPos--;
         InputString[Stringlen]='\x0';
         leftx--;
         Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
         tempstring=InputString+leftx;
         DisplayString(startx,starty,(char *)tempstring,Tcolor,Tcolor)

         DisplayChar( (curx+startx),starty,230,Tcolor);
       }
       else // leftx==0
       {
         Stringlen--;StrModPos--;
         DisplayChar( (curx+startx),starty,230,Bcolor);
         curx--;
         DisplayChar( (curx+startx),starty,InputString[Stringlen],

         DisplayChar( (curx+startx),starty,230,Tcolor);
       }
      }break;
      default:
      {
       if( Stringlen == MaxStrLen)
       {
         sound(2000);delay(100);nosound();
       }
       // used to press arrowkey
       else if( StrModPos<Stringlen )
       {
         //c.ch[0] doesn't in thai special character set
         InputString[Stringlen]='\x0';
         tempstring=CopyFromPos(StrModPos,InputString);
         InputString[StrModPos++]=c.ch[0];
         InputString[StrModPos]='\x0';
         strcat(InputString,tempstring);
         DisplayChar( (curx+startx),starty,230,Bcolor);
         Stringlen++;

         //display part
         if(Stringlen > LineLength)
         {
           if( curx > LineLength )
           {
             leftx++;
           }
           else curx++;
           Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
           tempstring=CopyFromPos(leftx,InputString);
           MakeOneLineChar(LineLength+1,tempstring);
           DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
           DisplayChar( (curx+startx),starty,230,Tcolor);
```

```
            }
            else
            {
              curx++;
              Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
              DisplayChar( (curx+startx),starty,230,Tcolor);
              DisplayString(startx,starty,InputString,Tcolor,Tcolor);
            }
          }
          else //never place arrowkey
          {
            InputString[Stringlen++]=c.ch[0]; StrModPos++;
            //display part
            if( curx > LineLength )
            {
              Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
              InputString[Stringlen]='\x0';
              leftx++;
              tempstring=InputString+leftx;
              DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
              DisplayChar( (curx+startx),starty,230,Tcolor);
            }
            else
            {
              DisplayChar( (curx+startx),starty,c.ch[0],Tcolor);
              DisplayChar( (curx+startx),starty,230,Bcolor);
              curx++;
              DisplayChar( (curx+startx),starty,230,Tcolor);
            }
          }
        }
      }
    }
    else if(c.ch[1])
    {
    switch(c.ch[1])
      {
       case LEFT:
       {
         if( curx==0 )
         {
           if( leftx==0 ) ;
             else
             {
               Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
               StrModPos--; leftx--;
               while(is_in("ิีึืุูัำ็่้๊๋",InputString[leftx])) leftx--;
               tempstring=CopyFromPos(leftx,InputString);
             MakeOneLineChar(LineLength+1,tempstring);
               DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
               DisplayChar( (curx+startx),starty,230,Tcolor);
             }
         }
         else
         {
           StrModPos--;
             while(is_in("ิีึืุูัำ็่้๊๋",InputString[StrModPos])) StrModPos-
-;

             DisplayChar( (curx+startx),starty,230,Bcolor);
             curx--;
             DisplayChar( (curx+startx),starty,230,Tcolor);
         }
       }break;
        case RIGHT:
        {
          if( StrModPos==Stringlen) ;
          else if( curx>LineLength )
          {
            Blockfill(startx,starty,startx+LineLength+1,starty,Bcolor);
            StrModPos++; leftx++;
            while(is_in("ิีึืุูัำ็่้๊๋",InputString[leftx])) leftx++;
            tempstring=CopyFromPos(leftx,InputString);
            MakeOneLineChar(LineLength+1,tempstring);
            DisplayString(startx,starty,tempstring,Tcolor,Tcolor);
            DisplayChar( (curx+startx),starty,230,Tcolor);
          }
          else
          {
            StrModPos++;
            while(is_in("ิีึืุูัำ็่้๊๋",InputString[StrModPos]))
StrModPos++;
            DisplayChar( (curx+startx),starty,230,Bcolor);
            curx++;
            DisplayChar( (curx+startx),starty,230,Tcolor);
          }
        }break;
      }
    }
  }
}
int GetThaiString(int startx,int starty,int MaxStrLen,int LineLength,int
Tcolor,int Bcolor)
{
  union inkey {
    char ch[2];
    int i;
    } c;
  unsigned char *tempstring="";
  int curx=0,Stringlen=0,StrModPos=0;

  if( LineLength<MaxStrLen ) return(ESC);
  MaxStrLen-=1;  LineLength-=1;
  Stringlen=strlen(InputString);
  curx=Stringlength(InputString);
  StrModPos=Stringlen;
  DisplayChar( (curx+startx),starty,230,Tcolor);
```

```
for(;;)
{
   while(!bioskey(1)) ;
   c.i=bioskey(0);
   if(c.ch[0])
   {
      if(Language==THA) c.ch[0]=GetThaikey(c.ch[0]);
      switch(c.ch[0])
      {
         case '':
         {
            if(Language==ENG) Language=THA;
            else Language=ENG;
         }break;
         case ESC:InputString[0]='\x0';return(ESC);
         case ENTER:
         {
            DisplayChar( (curx+startx),starty,230,Bcolor);
            InputString[Stringlen]='\x0';
            return(ENTER);
         }
         case TABKEY:
         {
            DisplayChar( (curx+startx),starty,230,Bcolor);
            InputString[Stringlen]='\x0';
            return(TABKEY);
         }
         case BKSP :
         {
            if(curx==0)
            {
               sound(2000);delay(200);nosound();
            }
            else
            {
               if( StrModPos<Stringlen )
               {
                  if( is_in("ัิีึืฺุู็่้๊๋์",InputString[StrModPos-1]) );
                  else
                  {
                     DisplayChar( (curx+startx),starty,230,Bcolor);
                     curx--;
                     DisplayChar( (curx+startx),starty,230,Tcolor);
                  }
                  InputString[Stringlen]='\x0';
                  tempstring=CopyFromPos(StrModPos,InputString);
                  InputString[StrModPos-1]='\x0';
                  strcat(InputString,tempstring);
                  //display part
                  Blockfill(startx,starty,startx+LineLength,starty,Bcolor);
                  DisplayString(startx,starty,InputString,Tcolor,Tcolor);
                  DisplayChar( (curx+startx),starty,230,Tcolor);
                  Stringlen--;StrModPos--;
               }
               else
               {
                  Stringlen--;StrModPos--;
                  if( is_in("ัิีึืฺุู็่้๊๋์",InputString[Stringlen]) )
                  {
                     DisplayChar( (startx+curx-1),starty,InputString
[Stringlen],Bcolor);
                  }
                  else
                  {
                     DisplayChar( (curx+startx),starty,230,Bcolor);
                     curx--;
                     DisplayChar( (curx+startx),starty,InputString
[Stringlen],Bcolor);
                     DisplayChar( (curx+startx),starty,230,Tcolor);
                  }
               }
            }
         }break;
         default:
         {
            if( Stringlen == MaxStrLen)
            {
               sound(2000);delay(100);nosound();
            }
            else
            {
               if( is_in("ัิีึืฺุู็่้๊๋์",c.ch[0]) )
               {
                  if( is_in("ะัาำิีึืฺุู็่้๊๋์ๆ",InputString[StrModPos-1]) )
                  {
                     sound(2000);delay(200);nosound();
                  }
                  else if( is_in("ัิีึืฺุู็่้๊๋์",InputString[StrModPos-1]) )
                  {
                     DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Bcolor);
                     InputString[StrModPos-1]=c.ch[0];
                     DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Tcolor);
                  }
                  else if( is_in("ุู",InputString[StrModPos-1]) )
                  {
                     if( is_in("ัิีึืฺ็่้๊๋์",InputString[StrModPos-2]) )
                     {
                        DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Bcolor);
                        InputString[StrModPos-2]=c.ch[0];
                        DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Tcolor);
                     }
                     else
```

```
{
  if( StrModPos==Stringlen )
      {
    InputString[StrModPos++]=c.ch[0];
      }
   else
   {
    tempstring=CopyFromPos(StrModPos,InputString);
     InputString[StrModPos++]=c.ch[0];
        InputString[StrModPos]='\x0';
     strcat(InputString,tempstring);
   }
      Stringlen++;
  DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
 }
  }
  else
 {
  if( StrModPos==Stringlen )
  {
    InputString[StrModPos++]=c.ch[0];
    }
  else
  {
    tempstring=CopyFromPos(StrModPos,InputString);
      InputString[StrModPos++]=c.ch[0];
    InputString[StrModPos]='\x0';
    strcat(InputString,tempstring);
    }
  Stringlen++;
  DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
  }
}
  else if( is_in("ิ",c.ch[0]) )
 {
  if( is_in("ีิัึืุู",InputString[StrModPos-1]) )
  {
    sound(2000);delay(200);nosound();
    }
  else if( is_in("ใไ",InputString[StrModPos-1]) )
    {
      DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Bcolor);
    InputString[StrModPos-1]=c.ch[0];
    DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Tcolor);
  }
    else if( is_in("แโ",InputString[StrModPos-1]) )
    {
      if( is_in("ใไ",InputString[StrModPos-2]) )
      {
        DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Bcolor);
    InputString[StrModPos-2]=c.ch[0];
    DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Tcolor);
  }
  else if( is_in("ิ",InputString[StrModPos-2]) )
  {
    sound(2000);delay(200);nosound();
  }
  else
  {
    if( StrModPos==Stringlen )
    {
      InputString[StrModPos++]=c.ch[0];
    }
    else
    {
      tempstring=CopyFromPos(StrModPos,InputString);
      InputString[StrModPos++]=c.ch[0];
      InputString[StrModPos]='\x0';
      strcat(InputString,tempstring);
    }
    Stringlen++;
    DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
  }
 }
 else
 {
  if( StrModPos==Stringlen )
  {
    InputString[StrModPos++]=c.ch[0];
  }
  else
  {
    tempstring=CopyFromPos(StrModPos,InputString);
    InputString[StrModPos++]=c.ch[0];
    InputString[StrModPos]='\x0';
    strcat(InputString,tempstring);
  }
  Stringlen++;
  DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
 }
}
else if( is_in("ิ",c.ch[0]) )
{
  if( is_in("ีิัึืุูใไแโ",InputString[StrModPos-1]) )
{
  sound(2000);delay(200);nosound();
}
else if( is_in("ิ",InputString[StrModPos-1]) )
  {
  DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Bcolor);
```

```
                InputString[StrModPos-1]=c.ch[0];
                DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Tcolor);
            }
            else if( is_in("แแไไ",InputString[StrModPos-1]) )
            {
                if( is_in("ก",InputString[StrModPos-2]) )
                {
                    DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Bcolor);
                InputString[StrModPos-2]=c.ch[0];
                DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-2],Tcolor);
                }
                else if( is_in("ูิีืึ",InputString[StrModPos-2]) )
                {
                    sound(2000);delay(200);nosound();
                }
                else
                {
                    if( StrModPos==Stringlen )
                    {
                        InputString[StrModPos++]=c.ch[0];
                    }
                    else
                    {
tempstring=CopyFromPos(StrModPos,InputString);
                        InputString[StrModPos++]=c.ch[0];
                        InputString[StrModPos]='\x0';
                        strcat(InputString,tempstring);
                    }
                    Stringlen++;
                    DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
                }
            }
            else
            {
                if( StrModPos==Stringlen )
                {
                    InputString[StrModPos++]=c.ch[0];
                }
                else
                {
                    tempstring=CopyFromPos(StrModPos,InputString);
                    InputString[StrModPos++]=c.ch[0];
                    InputString[StrModPos]='\x0';
                    strcat(InputString,tempstring);
                }
                Stringlen++;
                DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
            }
        }
        else if( is_in("ไใ",c.ch[0]) )
        {
            if( is_in("ีิำัๆ",InputString[StrModPos-1]) )
            {
                sound(2000);delay(200);nosound();
            }
            else if( is_in("แแไไใใ",InputString[StrModPos-1]) )
            {
                DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Bcolor);
                InputString[StrModPos-1]=c.ch[0];
                DisplayChar( (curx+startx-1),starty,InputString
[StrModPos-1],Tcolor);
            }
            else
            {
                if( StrModPos==Stringlen )
                {
                    InputString[StrModPos++]=c.ch[0];
                }
                else
                {
                    tempstring=CopyFromPos(StrModPos,InputString);
                    InputString[StrModPos++]=c.ch[0];
                    InputString[StrModPos]='\x0';
                    strcat(InputString,tempstring);
                }
                Stringlen++;
                DisplayChar( (curx+startx-1),starty,c.ch[0],Tcolor);
            }
        }
        else
        {
            if( StrModPos<Stringlen )
            {
                tempstring=CopyFromPos(StrModPos,InputString);
                InputString[StrModPos++]=c.ch[0];
                InputString[StrModPos]='\x0';
                strcat(InputString,tempstring);
                Stringlen++;
                Blockfill(startx,starty,startx+LineLength,starty,Bcolor)
;
                DisplayString(startx,starty,InputString,Tcolor,Tcolor);
                curx++;
                DisplayChar( (curx+startx),starty,230,Tcolor);
            }
            else
            {
                InputString[StrModPos++]=c.ch[0];
                Stringlen++;
                DisplayChar( (curx+startx),starty,c.ch[0],Tcolor);
                DisplayChar( (curx+startx),starty,230,Bcolor);
                curx++;
```

```
                DisplayChar( (curx+startx),starty,230,Tcolor);
              }
            }
          }
        }
      }
    }
    else if(c.ch[1])
    {
      switch(c.ch[1])
      {
        case SHFTAB:
        {
          DisplayChar( (curx+startx),starty,230,Bcolor);
          InputString[Stringlen]='\x0';
          return(SHFTAB);
        }
        case UP:
        {
          DisplayChar( (curx+startx),starty,230,Bcolor);
          InputString[Stringlen]='\x0';
          return(UP);
        }
        case DOWN:
        {
          DisplayChar( (curx+startx),starty,230,Bcolor);
          InputString[Stringlen]='\x0';
          return(DOWN);
        }
        case LEFT:
        {
          if( curx==0 )
          {
            sound(2000);delay(200);nosound();break;
          }
          else
          {
            StrModPos--;
            while(is_in("ฺ฿เแโใไๅ",InputString[StrModPos])) StrModPos-
-;
            DisplayChar( (curx+startx),starty,230,Bcolor);
            curx--;
            DisplayChar( (curx+startx),starty,230,Tcolor);
          }
        }break;
        case RIGHT:
        {
          if( StrModPos==Stringlen )
          {
            sound(2000);delay(200);nosound();break;
          }
          else
          {
            StrModPos++;
            while(is_in("ฺ฿เแโใไๅ",InputString[StrModPos])
StrModPos++;
            DisplayChar( (curx+startx),starty,230,Bcolor);
            curx++;
            DisplayChar( (curx+startx),starty,230,Tcolor);
          }
        }break;
      }
    }
  }
}
void MenuManage(void)
{
  int mode=0,choice=0,num=0;
  int leftx=0;
  int sel_choice=0;
  int user_select=FALSE;
  char far *append_buf;
  unsigned char **m;
  union inkey{
    unsigned char ch[2];
    int i;
    } c;

  Blockfill(0,0,79,1,MAINBKG_1);
  Blockfill(0,1,79,18,MAINBKG_2);
  Blockfill(0,19,79,19,MAINBKG_1);
  Defmenemu();
  LogMenuBuffer();

  for(num=0;num<MAX_MENU;num++)
    DisplayString(main_menu_startx[num],0,main_menu[num],
NORLMENU,HOTKEY);
  DisplayString(2,19,"%P%1%0-Menu | Arrow keys-Select. Enter-
Information.",DESATT,HOTKEY);
  OpenDatafile();
  num=0;
  for(;;)
  {
    while(!bioskey(1));
      c.i=bioskey(0);
    if(mode==0)
    {
      switch(c.ch[1])
      {
        case F10 :
        {
          mode=1;
          Blockfill(main_menu_startx[num],0,main_menu_length[num]
+main_menu_startx[num],0,MAINBKG_R);
          DisplayString(main_menu_startx[num],0,main_menu[num],
NORLMENU_R,HOTKEY_R);
```

```
            Blockfill(leftx,19,79,19,MAINBKG_1);
            DisplayString(leftx,19,describ[num],DESATT,HOTKEY);
          } break;
        }
      }
      else if (mode==1)
      {
        if(c.ch[0])
        {
          switch(c.ch[0])
          {
            case ENTER :
            {
              DisplayPulldown(num);
              m=menu_num[num].menu;
              Blockfill(menu_num[num].startx,choice+1,
                menu_num[num].endx,choice+1,BKGPULLD_R );
              DisplayString(menu_num[num].startx,choice+1,
                m[choice],NORLMENU_R,HOTKEY_R);
              mode=2;
            } break;
            case ESC :
            {
              mode=0;
              Blockfill(main_menu_startx[num],0,
                main_menu_length[num]+main_menu_startx[num],0,
MAINBKG_1);
              DisplayString(main_menu_startx[num],0,
                main_menu[num],NORLMENU,HOTKEY);
              Blockfill(leftx,19,79,19,MAINBKG_1);
              DisplayString(2,19,"%P%1%0-Menu | Arrow keys-Select.
Enter-Information.",DESATT,HOTKEY);
            }break;
          }
        }
        else
        {
          switch(c.ch[1])
          {
            case F10 :
            {
              mode=0;
              Blockfill(main_menu_startx[num],0,
                main_menu_length[num]+main_menu_startx[num],0,
MAINBKG_1);
              DisplayString(main_menu_startx[num],0,
                main_menu[num],NORLMENU,HOTKEY);
              Blockfill(leftx,19,79,19,MAINBKG_1);
              DisplayString(2,19,"%P%1%0-Menu | Arrow keys-Select.
Enter-Information.",DESATT,HOTKEY);
            } break;
            case Alt_A ://File
            {
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
              num=0;
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
              Blockfill(leftx,19,79,19,MAINBKG_1);
              DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
              DisplayPulldown(num);
              m=menu_num[num].menu;
              Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
              DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
              mode=2;
            } break;
            case Alt_K : //Database
            {
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
              num=1;
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
              Blockfill(leftx,19,79,19,MAINBKG_1);
              DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
              DisplayPulldown(num);
              m=menu_num[num].menu;
              Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
              DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
              mode=2;
            } break;
            case Alt_S : //search+sort
            {
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
              num=2;
              Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
              DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
```

```
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                DisplayPulldown(num);
                m=menu_num[num].menu;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
                mode=2;
              } break;
              case Alt_0: //reserve
              {
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=3;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                DisplayPulldown(num);
                m=menu_num[num].menu;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
                mode=2;
              }break;
              case Alt_R: //print
              {
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=4;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                DisplayPulldown(num);
                m=menu_num[num].menu;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
                mode=2;
              }break;
              case Alt_V://set serial port
              {
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=5;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                DisplayPulldown(num);
                m=menu_num[num].menu;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
                mode=2;
              }break;
              case Alt_X: return;
              case LEFT :
              {
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num--;
                if(num<0) num=MAX_MENU-1;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
              } break;
              case RIGHT :
              {
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num++;
                if(num>MAX_MENU-1) num=0;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
```

```
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
              } break;
              case DOWN :
              {
                DisplayPulldown(num);
                m=menu_num[num].menu;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,m[choice],
NORLMENU_R,HOTKEY_R);
                mode=2;
              } break;
            }
          }
        }
        else if(mode==2)
        {
          if(c.ch[0])
          {
            sel_choice=is_in(menu_num[num].keys,tolower(c.ch[0]));
            if(sel_choice)
            {
              Blockfill(menu_num[num].startx,choice+1,menu_num[num].
endx,choice+1,BKGPULLD );
              DisplayString(menu_num[num].startx,choice+1,
                m[choice],NORLMENU,HOTKEY);
              choice=sel_choice-1;
              Blockfill(menu_num[num].startx,choice+1,menu_num[num].
endx,choice+1,BKGPULLD_R );
              DisplayString(menu_num[num].startx,choice+1,
                m[choice],NORLMENU_R,HOTKEY_R);
              Blockfill(leftx,19,79,19,MAINBKG_1);
              DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R);
              putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                mainmenu_buf,COPY_PUT);
              user_select=TRUE;
            }
            if(!user_select)
            {
              switch(c.ch[0])
              {
                case ENTER :
                {
                  putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                    mainmenu_buf,COPY_PUT);
                  user_select=TRUE;
                }break;
                case ESC :
                {
                  putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                    mainmenu_buf,COPY_PUT);
                  choice=0;
                  mode=1;
                } break;
              }
            }
          }
          else
          {
            switch(c.ch[1])
            {
              case UP :
              {
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU,HOTKEY);
                choice--;
                if(choice<0) choice=menu_num[num].count-1;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case DOWN :
              {
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD);
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU,HOTKEY);
                choice++;
                if(choice>(menu_num[num].count-1))
                  choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R);
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case LEFT :
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num--;
                if(num<0) num=MAX_MENU-1;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R);
```

```
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case RIGHT :
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num++;
                if(num>MAX_MENU-1) num=0;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_A ://File
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=0;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_K ://Database
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=1;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_8 ://search+sort
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=2;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
```

```
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_0 ://reserve
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=3;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_R ://print
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=4;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_V ://set port
              {
                putimage(menu_num[num].startx*8-2,menu_num[num].
starty*24-2,
                  mainmenu_buf,COPY_PUT);
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_1);
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU,HOTKEY);
                num=5;
                Blockfill(main_menu_startx[num],0,main_menu_length
[num]+main_menu_startx[num],0,MAINBKG_R );
                DisplayString(main_menu_startx[num],0,main_menu
[num],NORLMENU_R,HOTKEY_R);
                Blockfill(leftx,19,79,19,MAINBKG_1);
                DisplayString(leftx,19,describ[num],DESATT,HOTKEY_R)
;
                m=menu_num[num].menu;
                DisplayPulldown(num);
                choice=0;
                Blockfill(menu_num[num].startx,choice+1,menu_num
[num].endx,choice+1,BKGPULLD_R );
                DisplayString(menu_num[num].startx,choice+1,
                  m[choice],NORLMENU_R,HOTKEY_R);
              } break;
              case Alt_X : return;
            }
          }
        }
        if(user_select)
        {
          user_select=FALSE;
          mode=1;
          switch(num)
          {
            case 0 :
            {
              switch(choice)
              { case 0 :OpenDatafile();break;//Creatfile("studfile");break;
                case 1 :DosShell();break;
                case 2 :return;
              }
              choice=0;
            } break;
            case 1 :
            {
              switch(choice)
              {
```

```
            case 0 :
            {
              append_buf=(char far *)farcalloc(72576L,sizeof(char));
              if(!append_buf)
              {
                closegraph();
                printf("Alloc error in Menumange function while call
AppendData");
                exit(0);
              }
              Getvideo(14,3,67,16,append_buf);
              do
              {
                InitialStudData();
              }while( AppendData()!=ESC );
              Putvideo(14,3,67,16,append_buf);
              farfree(append_buf);
            }break;
            case 1 :
            {
              if(GetmodifyRecord()==PASS)
                ModifyData();
            }break;
            case 2 : break;
          } choice=0;
        } break;
        case 2 :
        {
          switch(choice)
          {
            case 0 : SearchDialogue();break;
            case 1 : SortDialogue();break;
          }
          choice=0;
        } break;
        case 3 :
        {
          switch(choice)
          {
            case 0:
            {
              append_buf=(char far *)farcalloc(77760L,sizeof(char));
              if(!append_buf)
              {
                closegraph();
                printf("Alloc error in Menumanage while call
AppendReserveTime");
                exit(0);
              }
              Getvideo(14,2,67,16,append_buf);
              do
              {
                InitialReserveHeader();
              }while( AppendReserveTime()!=ESC );
              Putvideo(14,2,67,16,append_buf);
              farfree(append_buf);
            }break;
            case 1: break;
            case 2: break;
          }
          choice=0;
        } break;
        case 4 :
        {
          switch(choice)
          {
            case 0 : break;
          }
          choice=0;
        } break;
      }
    }
  }
}
void DosShell(void)
{
  int num;
  Blockfill(0,0,79,19,BLACK);
  system("command.com");
  Blockfill(0,0,79,1,MAINBKG_1);
  Blockfill(0,1,79,18,MAINBKG_2);
  Blockfill(0,19,79,19,MAINBKG_1);
  for(num=0;num<MAX_MENU;num++)
    DisplayString(main_menu_startx[num],0,main_menu[num],
NORLMENU,HOTKEY);
  DisplayString(2,19,"%P%1%0-Menu ! Arrow keys-Select. Enter-
Information.",DESATT,HOTKEY);
}
void main(void)
{
  Initialize();
  MenuManage();
  ExitProgram();
}
int Getdate(int xpos,int ypos,char inputdate[11])
{
  char active_att=MAGENTA,deactive_att=BLUE;
  union inkey{
    int i;
    char ch[2];
    } c;
  int count=0,curx=0;
  Blockfill(xpos,ypos,xpos+10,ypos,POPUPBKG);
  DisplayString(xpos,ypos,inputdate,active_att,active_att);
  if(inputdate[9]!=' ')
  {
```

```
    curx=10;count=10;
}
DisplayChar(xpos+curx,ypos,230,active_att);
for(;;)
{
   while(!bioskey(1));
   c.i=bioskey(0);
   if( c.ch[0] == ESC ) return ESC;
   if( c.ch[0] == TABKEY || c.ch[0] == ENTER )
   {
      if(inputdate[9]!=' ')
      {
         inputdate[10]='\x0';
         DisplayChar(xpos+curx,ypos,230,POPUPBKG);
         DisplayString(xpos,ypos,inputdate,deactive_att,deactive_att);
         return ENTER;
      }
      else
      {
         sound(2000);delay(200);nosound();
      }
   }
   if( c.ch[1] == SHFTAB)
   {
      if(inputdate[9]!=' ')
      {
         inputdate[10]='\x0';
         DisplayChar(xpos+curx,ypos,230,POPUPBKG);
         DisplayString(xpos,ypos,inputdate,deactive_att,deactive_att);
         return SHFTAB;
      }
      else
      {
         sound(2000);delay(200);nosound();
      }
   }
   switch(count)
   {
      case 0:
      {
         if( is_in("0123",c.ch[0]) )
         {
            inputdate[count]=c.ch[0];
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
            curx++;count++;
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
         }
         else
         {
            sound(2000);delay(200);nosound();
         }
      }break;
      case 1:
      {
         if( c.ch[0]==BKSP )
         {
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            curx--;count--;
            DisplayChar(xpos+curx,ypos,229,POPUPBKG);
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
         }
         switch(inputdate[count-1])
         {
            case '0':
            {
               if( is_in("123456789",c.ch[0]) )
               {
                  inputdate[count]=c.ch[0];
                  DisplayChar(xpos+curx,ypos,230,POPUPBKG);
                  DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
                  curx+=2;count+=2;
                  DisplayChar(xpos+curx,ypos,230,active_att);
                  break;
               }
               else
               {
                  sound(2000);delay(200);nosound();
               }
            }break;
            case '1':case '2':
            {
               if( is_in("0123456789",c.ch[0]) )
               {
                  inputdate[count]=c.ch[0];
                  DisplayChar(xpos+curx,ypos,230,POPUPBKG);
                  DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
                  curx+=2;count+=2;
                  DisplayChar(xpos+curx,ypos,230,active_att);
                  break;
               }
               else
               {
                  sound(2000);delay(200);nosound();
               }
            }break;
            case '3':
            {
               if( is_in("01",c.ch[0]) )
               {
                  inputdate[count]=c.ch[0];
                  DisplayChar(xpos+curx,ypos,230,POPUPBKG);
                  DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
                  curx+=2;count+=2;
```

```
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
          }
          else
          {
            sound(2000);delay(200);nosound();
          }
        }break;
      }
    }break;
    case 2:break;
    case 3:
    {
      if( c.ch[0]==BKSP )
      {
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        curx-=2;count-=2;
        DisplayChar(xpos+curx,ypos,229,POPUPBKG);
        DisplayChar(xpos+curx,ypos,230,active_att);
        break;
      }
      if( is_in("01",c.ch[0]) )
      {
        inputdate[ccount]=c.ch[0];
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
        curx++;count++;
        DisplayChar(xpos+curx,ypos,230,active_att);
        break;
      }
      else
      {
        sound(2000);delay(200);nosound();
      }
    }break;
    case 4:
    {
      if( c.ch[0]==BKSP )
      {
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        curx--;count--;
        DisplayChar(xpos+curx,ypos,229,POPUPBKG);
        DisplayChar(xpos+curx,ypos,230,active_att);
        break;
      }
      switch(inputdate[count-1])
      {
        case '0':
        {
          if( is_in("123456789",c.ch[0]) )
          {
            inputdate[count]=c.ch[0];
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
            curx+=2;count+=2;
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
          }
          else
          {
            sound(2000);delay(200);nosound();
          }
        }break;
        case '1':
        {
          if( is_in("012",c.ch[0]) )
          {
            inputdate[count]=c.ch[0];
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
            curx+=2;count+=2;
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
          }
          else
          {
            sound(2000);delay(200);nosound();
          }
        }break;
      }
    }break;
    case 5:break;
    case 6:
    {
      if( c.ch[0]==BKSP )
      {
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        curx-=2;count-=2;
        DisplayChar(xpos+curx,ypos,229,POPUPBKG);
        DisplayChar(xpos+curx,ypos,230,active_att);
        break;
      }
      if( is_in("2",c.ch[0]) )
      {
        inputdate[count]=c.ch[0];
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
        curx++;count++;
        DisplayChar(xpos+curx,ypos,230,active_att);
        break;
      }
      else
      {
        sound(2000);delay(200);nosound();
      }
    }break;
```

```
case 7:
{
   if( c.ch[0]==BKSP )
   {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   if( is_in("56789",c.ch[0]) )
   {
      inputdate[count]=c.ch[0];
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
      curx++;count++;
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   else
   {
      sound(2000);delay(200);nosound();
   }
}break;
case 8:
{
   if( c.ch[0]==BKSP )
   {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   switch(inputdate[count-1])
   {
      case '5':
      {
         if( is_in("3456789",c.ch[0]) )
         {
            inputdate[count]=c.ch[0];
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
            curx++;count++;
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
         }
         else
         {
            sound(2000);delay(200);nosound();
         }
      }break;
      case '6':case '7':case '8':case '9':
      {
         if( is_in("0123456789",c.ch[0]) )
         {
            inputdate[count]=c.ch[0];
            DisplayChar(xpos+curx,ypos,230,POPUPBKG);
            DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
            curx++;count++;
            DisplayChar(xpos+curx,ypos,230,active_att);
            break;
         }
         else
         {
            sound(2000);delay(200);nosound();
         }
      }break;
   }
}break;
case 9:
{
   if( c.ch[0]==BKSP )
   {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   if( is_in("0123456789",c.ch[0]) )
   {
      inputdate[count]=c.ch[0];
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
      curx++;count++;
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   else
   {
      sound(2000);delay(200);nosound();
   }
}break;
case 10:
{
   if( c.ch[0]==BKSP )
   {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
   }
   else
   {
```

```
                sound(2000);delay(200);nosound();
              }
            }break;//end case 8
          }//end switch main
        }
}
void DrawDbaseDialogue(void)
{
  char *message[]={
                    "ระเบียนนักศึกษาลำดับที่",
                    "ชื่อ",
                    "รหัสนักศึกษา",
                    "คณะ",
                    "ชั้นปี",
                    "วันออกบัตร",
                    "วันหมดอายุ",
                  };
  Blockfill(15,4,65,15,POPUPBKG);
  Dimension(15,4,65,15,4,RIMUPPER,RIMLOWER);
  Shadow(15,4,65,15,4,8,POPUPSHA);
  Blockfill(26,5,54,5,BLUE);
  Border(26,5,54,5,2,LIGHTBLUE);
  DisplayString(28,5,message[0],YELLOW,RED);
  DisplayString(25,7,message[1],BLACK,RED);
  DisplayString(18,8,message[2],BLACK,RED);
  DisplayString(24,9,message[3],BLACK,RED);
  DisplayString(24,10,message[4],BLACK,RED);
  DisplayString(19,11,message[5],BLACK,RED);
  DisplayString(19,12,message[6],BLACK,RED);
}
void DisplayRecordnumber(unsigned long int recordnum,int x,int y,int
bkg,int texte)
{
  char srecord[7]="";
  Blockfill(x,y,x+6,y,bkg);
  recordnum++;
  srecord[0]= (recordnum/100000)+0x30; recordnum%=100000;
  srecord[1]= (recordnum/10000)+0x30;  recordnum%=10000;
  srecord[2]= (recordnum/1000)+0x30;   recordnum%=1000;
  srecord[3]= (recordnum/100)+0x30;    recordnum%=100;
  srecord[4]= (recordnum/10)+0x30;     recordnum%=10;
  srecord[5]= recordnum+0x30;
  srecord[6]= '\x0';
  DisplayString(x,y,srecord,texte,texte);
}
void InitialStudData(void)
{
  strcpy(rdate," / /   ");
  strcpy(xdate," / /   ");
  StudentRecord.Name[0]='\x0';
  StudentRecord.StudentId[0]='\x0';
  StudentRecord.Faculty[0]='\x0';
  StudentRecord.Year='\x0';
  CurrentRecord=TotalRecord;
  DrawDbaseDialogue();
}
int AppendData(void)
{
  int x=28,y=7,choice=0;
  char active_att=MAGENTA,deactive_att=BLUE;
  DisplayRecordnumber(CurrentRecord,47,5,BLUE,YELLOW);
  Blockfill(28,14,38,14,LIGHTGREEN);
  Border(28,14,38,14,1,GREEN);
  Shadow(28,14,38,14,1,5,ICONSHA);
  PutCenterString(29,38,14,"ตกลง",BLUE);
  Blockfill(44,14,54,14,LIGHTGREEN);
  Border(44,14,54,14,1,GREEN);
  Shadow(44,14,54,14,1,5,ICONSHA);
  PutCenterString(44,54,14,"ยกเลิก",BLUE);
  for(;;)
  {
   switch(choice)
    {
      case 0:
      {
Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
       strcpy(InputString,StudentRecord.Name);
       DisplayString(x+1,y+choice,InputString,active_att,active_att);
       Language=THA;
       switch(
GetThaiString(x+1,y+choice,30,32,active_att,POPUPBKG) )
        {
         case ESC: return ESC;
         case ENTER:case TABKEY:
         {
          strcpy(StudentRecord.Name,InputString);
          Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
           choice=1;
         }break;
         case SHFTAB:
         {
          strcpy(StudentRecord.Name,InputString);
          Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
           choice=7;
         }break;
        }
      }break;
      case 1:
      {
Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
```

```
strcpy(InputString,StudentRecord.StudentId);
DisplayString(x+1,y+choice,InputString,active_att,active_att);
Language=ENG;
switch( GetThaiString(x+1,y+choice,9,32,active_att,POPUPBKG)
)
{
  case ESC: return ESC;
  case ENTER:case TABKEY:
  {
    strcpy(StudentRecord.StudentId,InputString);
    Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
    choice=2;
  }break;
  case SHFTAB:
  {
    strcpy(StudentRecord.StudentId,InputString);
    Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
    choice=0;
  }break;
 }
}break;
case 2:
{

Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
  strcpy(InputString,StudentRecord.Faculty);
  DisplayString(x+1,y+choice,InputString,active_att,active_att);
  Language=THA;
  switch(
GetThaiString(x+1,y+choice,30,32,active_att,POPUPBKG) )
  {
    case ESC: return ESC;
    case ENTER:case TABKEY:
    {
      strcpy(StudentRecord.Faculty,InputString);
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
      choice=3;
    }break;
    case SHFTAB:
    {
      strcpy(StudentRecord.Faculty,InputString);
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
      choice=1;
    }break;
  }
}break;
case 3:
{

Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
  InputString[0]=StudentRecord.Year;
  DisplayChar(x+1,y+choice,StudentRecord.Year,active_att);
  InputString[1]='\x0';
  Language=ENG;
  switch( GetThaiString(x+1,y+choice,2,32,active_att,POPUPBKG)
)
  {
    case ESC: return ESC;
    case ENTER:case TABKEY:
    {
      StudentRecord.Year=InputString[0];
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayChar(x+1,y+choice,StudentRecord.Year,deactive_att);
      choice=4;
    }break;
    case SHFTAB:
    {
      StudentRecord.Year=InputString[0];
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);

DisplayChar(x+1,y+choice,StudentRecord.Year,deactive_att);
      choice=2;
    }break;
  }
}break;
case 4:
{

Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
  switch(Getdate(x+1,y+choice,rdate))
  {
    case ESC: return ESC;
    case ENTER:
    {
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
      choice=5;
      StudentRecord.RegisDate.day= (((rdate[0]-0x30)*10)+rdate
[1]-0x30)
      StudentRecord.RegisDate.month= (((rdate[3]-0x30)*10)+
rdate[4]-0          x30);
      StudentRecord.RegisDate.year= (((rdate[6]-0x30)*1000)+
      ((rdate[7]-0x30)*100)+((rdate[8]-0x30)*10)+(rdate[9]-0
x30));
    }break;
    case SHFTAB:
    {
      Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
```

```
                choice=3;
                StudentRecord.RegisDate.day= (((rdate[0]-0x30)*10)+rdate
[1]-0x30)
                StudentRecord.RegisDate.month= (((rdate[3]-0x30)*10)+
rdate[4]-0          x30);
                StudentRecord.RegisDate.year= (((rdate[6]-0x30)*1000)+
                ((rdate[7]-0x30)*100)+((rdate[8]-0x30)*10)+(rdate[9]-0
x30));
              }break;
            }
          }break;
          case 5:
          {

   Dimension(x,y+choice,x+34,y+choice,2,RIMLOWER,RIMUPPER);
            switch(Getdate(x+1,y+choice,xdate))
            {
              case ESC: return ESC;
              case ENTER:
              {
                Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
                choice=6;
                StudentRecord.ExpirDate.day= (((xdate[0]-0x30)*10)+
xdate[1]-0          x30);
                StudentRecord.ExpirDate.month= (((xdate[3]-0x30)*10)+
xdate[4]-0          x30);
                StudentRecord.ExpirDate.year= (((xdate[6]-0x30)*1000)+
                ((xdate[7]-0x30)*100)+((xdate[8]-0x30)*10)+(xdate[9]-0
x30));
              }break;
              case SHFTAB:
              {
                Border(x,y+choice,x+34,y+choice,2,POPUPBKG);
                choice=4;
                StudentRecord.ExpirDate.day= (((xdate[0]-0x30)*10)+
xdate[1]-0          x30);
                StudentRecord.ExpirDate.month= (((xdate[3]-0x30)*10)+
xdate[4]-0          x30);
                StudentRecord.ExpirDate.year= (((xdate[6]-0x30)*1000)+
                ((xdate[7]-0x30)*100)+((xdate[8]-0x30)*10)+(xdate[9]-0
x30));
              }break;
            }
          }break;
          case 6:
          {
            Blockfill(28,14,38,14,YELLOW);
            Border(28,14,38,14,1,RED);
            PutCenterString(29,38,14,"ศกสง",BLUE);
            do
            {
              while(!kbhit());
              switch(getch())
              {
                case ESC: return ESC;
                case ENTER:
                {
                  fseek(StudentIdFile,CurrentRecord*sizeof
(StudentRecord)+13,          SEEK_SET);
                  fwrite(&StudentRecord,sizeof(StudentRecord),1,
StudentIdFile);
                  if(CurrentRecord==TotalRecord) TotalRecord++;
                  CurrentRecord++;
                  return ENTER;
                }
                case TABKEY:
                {
                  Blockfill(28,14,38,14,LIGHTGREEN);
                  Border(28,14,38,14,1,GREEN);
                  PutCenterString(29,38,14,"ศกสง",BLUE);
                  choice=7;
                }break;
                case SHFTAB:
                {
                  Blockfill(28,14,38,14,LIGHTGREEN);
                  Border(28,14,38,14,1,GREEN);
                  PutCenterString(29,38,14,"ศกสง",BLUE);
                  choice=5;
                }break;
              }
            } while( choice!=5&&choice!=7 );
          }break;
          case 7:
          {
            Blockfill(44,14,54,14,YELLOW);
            Border(44,14,54,14,1,RED);
            PutCenterString(44,54,14,"ยกเลิก",BLUE);
            do
            {
              while(!kbhit());
              switch(getch())
              {
                case ESC:
                case ENTER: return ESC;
                case TABKEY:
                {
                  Blockfill(44,14,54,14,LIGHTGREEN);
                  Border(44,14,54,14,1,GREEN);
                  PutCenterString(44,54,14,"ยกเลิก",BLUE);
                  choice=0;
                }break;
                case SHFTAB:
                {
                  Blockfill(44,14,54,14,LIGHTGREEN);
                  Border(44,14,54,14,1,GREEN);
                  PutCenterString(44,54,14,"ยกเลิก",BLUE);
```

```
                    choice=6;
                  }break;
               }
            } while( choice!=0&&choice!=6 );
         }break;
      }
   }
}
void DisplayStudentData(void)
{
   int x=28,y=7;
   char deactive_att=BLUE;
   char *recdate;

   fseek(StudentIdFile,CurrentRecord*sizeof(StudentRecord)+13,
SEEK_SET);
   fread(&StudentRecord,sizeof(StudentRecord),1,StudentIdFile);
   DisplayRecordnumber(CurrentRecord,47,5,BLUE,YELLOW);
   Blockfill(x,y,x+34,y,POPUPBKG);
   DisplayString(x+1,y,StudentRecord.Name,deactive_att,deactive_att);
   Blockfill(x,y+1,x+34,y+1,POPUPBKG);
   DisplayString(x+1,y+1,StudentRecord.StudentId,deactive_att,deactive_att
);
   Blockfill(x,y+2,x+34,y+2,POPUPBKG);
   DisplayString(x+1,y+2,StudentRecord.Faculty,deactive_att,deactive_att);
   Blockfill(x,y+3,x+34,y+3,POPUPBKG);
   DisplayChar(x+1,y+3,StudentRecord.Year,deactive_att);
   recdate=Convertdate(&StudentRecord.RegisDate);
   strcpy(rdate,recdate);
   Blockfill(x,y+4,x+34,y+4,POPUPBKG);
   DisplayString(x+1,y+4,recdate,deactive_att,deactive_att);
   recdate=Convertdate(&StudentRecord.ExpirDate);
   strcpy(xdate,recdate);
   Blockfill(x,y+5,x+34,y+5,POPUPBKG);
   DisplayString(x+1,y+5,recdate,deactive_att,deactive_att);
}
int ModifyData(void)
{
   char *iconmsg[4]={   "ระเบียนต่อไป",
                        "ระเบียนก่อน",
                        "แก้ไข",
                        "ยกเลิก",
                    };
   int startx[]={17,29,41,53};
   int choice=0;
   char far *modify_buf;
   char far *appd_buf;
   modify_buf=(char far *)farcalloc(72576L,sizeof(char));
   if(!modify_buf)
   {
      closegraph();
      printf("Alloc error in ModifyData function");
      exit(0);
   }
   Getvideo(14,3,67,16,modify_buf);
   DrawDbaseDialogue();
   DisplayStudentData();
   for(choice=1;choice<4;choice++)
   {
      Blockfill(startx[choice],14,startx[choice]+10,14,ICONORM);
      Shadow(startx[choice],14,startx[choice]+10,14,1,3,ICONSHA);
      Border(startx[choice],14,startx[choice]+10,14,1,ICONORIM);
      PutCenterString(startx[choice],startx[choice]+10,14,iconmsg[choice],
ICONTEXT);
   }
   choice=0;
   Blockfill(startx[choice],14,startx[choice]+10,14,ICONACTIVE);
   Shadow(startx[choice],14,startx[choice]+10,14,1,3,ICONSHA);
   Border(startx[choice],14,startx[choice]+10,14,1,ICONACRIM);
   PutCenterString(startx[choice],startx[choice]+10,14,iconmsg[choice],
ICONTEXT);
   for(;;)
   {
      while(!kbhit());
      switch(getch())
      {
         case ESC:
         {
            Putvideo(14,3,67,16,modify_buf);
            farfree(modify_buf);
            return ESC;
         }
         case ENTER:
         {
            switch(choice)
            {
               case 0:
               {
                  CurrentRecord++;
                  if( CurrentRecord>=TotalRecord )
                  {
                     if(TotalRecord)
                     {
                        CurrentRecord=TotalRecord-1;break;
                     }
                     else break;
                  }
                  else
                  {
                     DisplayStudentData();
                  }
               }break;
               case 1:
               {
                  if(CurrentRecord>0)
```

```
                CurrentRecord--;
                DisplayStudentData();
              }
            }break;
            case 2:
            {
              appd_buf=(char far *)farcalloc(72576L,sizeof(char));
              if(!appd_buf)
              {
                closegraph();
                printf("Alloc error in ModifyData function while call
                    AppendData");
                exit(0);
              }
              Getvideo(14,5,67,16,appd_buf);
              Blockfill(16,13,64,15,POPUPBKG);
              if( AppendData()!=ESC) CurrentRecord--;
              Putvideo(14,5,67,16,appd_buf);
              farfree(appd_buf);
              DisplayStudentData();
            }break;
            case 3:
            {
              Putvideo(14,3,67,16,modify_buf);
              farfree(modify_buf);
              return ESC;
            }
          }
        }break;
        case TABKEY:
        {
          Blockfill(startx[choice],14,startx[choice]+10,14,ICONORM);
          Shadow(startx[choice],14,startx[choice]+10,14,1,3,ICONSHA);
          Border(startx[choice],14,startx[choice]+10,14,1,ICONORIM);
          PutCenterString(startx[choice],startx[choice]+10,14,iconmsg
[choice],ICONTEXT);
          choice++;
          if(choice>3) choice=0;
          Blockfill(startx[choice],14,startx[choice]+10,14,ICONACTIVE);
          Shadow(startx[choice],14,startx[choice]+10,14,1,3,ICONSHA);
          Border(startx[choice],14,startx[choice]+10,14,1,ICONACRIM);
          PutCenterString(startx[choice],startx[choice]+10,14,iconmsg
[choice], ICONTEXT);
        }break;
        case 0x00:
        {
          switch(getch())
          {
            case SHFTAB: case LEFT:
            {
              Blockfill(startx[choice],14,startx[choice]+10,14,ICONORM)
;
              Shadow(startx[choice],14,startx[choice]+10,14,1,3,
ICONSHA);
              Border(startx[choice],14,startx[choice]+10,14,1,
ICONORIM);
              PutCenterString(startx[choice],startx[choice]+10,14,
iconmsg[choice],ICONTEXT);
              choice--;if(choice<0) choice=3;
              Blockfill(startx[choice],14,startx[choice]+10,14,
ICONACTIVE);
              Shadow(startx[choice],14,startx[choice]+10,14,1,3,
ICONSHA);
              Border(startx[choice],14,startx[choice]+10,14,1,
ICONACRIM);
              PutCenterString(startx[choice],startx[choice]+10,14,
iconmsg[choice],ICONTEXT);
            }break;
            case RIGHT:
            {
              Blockfill(startx[choice],14,startx[choice]+10,14,ICONORM)
;
              Shadow(startx[choice],14,startx[choice]+10,14,1,3,
ICONSHA);
              Border(startx[choice],14,startx[choice]+10,14,1,
ICONORIM);
              PutCenterString(startx[choice],startx[choice]+10,14,
iconmsg[choice],ICONTEXT);
              choice++;
              if(choice>3) choice=0;
              Blockfill(startx[choice],14,startx[choice]+10,14,
ICONACTIVE);
              Shadow(startx[choice],14,startx[choice]+10,14,1,3,
ICONSHA);
              Border(startx[choice],14,startx[choice]+10,14,1,
ICONACRIM);
              PutCenterString(startx[choice],startx[choice]+10,14,
iconmsg[choice],ICONTEXT);
            }break;
          }
        }break;
      }
    }
}
char *Convertdate(date_t *srcdate)
{
  char *strdate="00\\00\\0000";

  strdate[0]= ((srcdate->day)/10)+0x30;
  strdate[1]= ((srcdate->day)%10)+0x30;
  strdate[2]= '/';
  strdate[3]= ((srcdate->month)/10)+0x30;
  strdate[4]= ((srcdate->month)%10)+0x30;
  strdate[5]= '/';
  strdate[6]= ((srcdate->year)/1000)+0x30;srcdate->year%=1000;
```

```
    strdate[7]= ((srcdate->year)/100)+0x30;srcdate->year%=100;
    strdate[8]= ((srcdate->year)/10)+0x30;
    strdate[9]= ((srcdate->year)%10)+0x30;
    strdate[10]= '\x0';
    return strdate;
}
int Sort(FILE *fp,unsigned long int totalrecord)
{
    if( QuickDisk(fp,totalrecord)==USERCANCEL )
        return USERCANCEL;
    else return PASS;
}
int QuickDisk(FILE *fp,unsigned long int totalrecord)
{
    if( Qsdisk(fp,0L,totalrecord-1)==USERCANCEL )
    /* count-1 is a number of record begin at '0'*/
        return USERCANCEL;
    else return PASS;
}
int Qsdisk(FILE *fp,unsigned long int left,unsigned long int right)
{
    unsigned long int i,j;
    char Id[9];
    i=left;
    j=right;
    /*get the middle StudentId*/
    strcpy(Id,GetStudentId(fp,(long)(i+j)/2));
    do
    {
        while( strcmp(GetStudentId(fp,i),Id)<0 && i<right )
        {
            if(bioskey(1))
            {
                bioskey(0);
                return USERCANCEL;
            }
            i++;
        }
        while( strcmp(GetStudentId(fp,j),Id)>0 && j>left )
        {
            if(bioskey(1))
            {
                bioskey(0);
                return USERCANCEL;
            }
            j--;
        }
        if(i<=j)
        {
            Swapfield(fp,i,j);
            i++; if(j) j--;
        }
    } while(i<=j);
    if(left<j)
    {
        if( Qsdisk(fp,left,j)==USERCANCEL ) return USERCANCEL;
    }
    if(i<right)
    {
        if( Qsdisk(fp,i,right)==USERCANCEL ) return USERCANCEL;
    }
    return PASS;
}
char *GetStudentId(FILE *fp,unsigned long int record)
{
    StudentRecord_t *p;
    p=&StudentRecord;
    fseek(fp,13L+record*sizeof(StudentRecord),SEEK_SET);
    fread(p,sizeof(StudentRecord),1,fp);
    return StudentRecord.StudentId;
}
void Swapfield(FILE *fp,unsigned long int i,unsigned long int j)
{
    char a[sizeof(StudentRecord_t)],b[sizeof(StudentRecord_t)];
    /*first read in record i and j*/
    fseek(fp,13L+sizeof(StudentRecord_t)*i,SEEK_SET);
    fread((char *)a,sizeof(StudentRecord_t),1,fp);
    fseek(fp,13L+sizeof(StudentRecord_t)*j,SEEK_SET);
    fread((char *)b,sizeof(StudentRecord_t),1,fp);
    /*then write them back in opposite slots*/
    fseek(fp,13L+sizeof(StudentRecord_t)*i,SEEK_SET);
    fwrite((char *)b,sizeof(StudentRecord_t),1,fp);
    fseek(fp,13L+sizeof(StudentRecord_t)*j,SEEK_SET);
    fwrite((char *)a,sizeof(StudentRecord_t),1,fp);
}
/*************************************************************
**/
/***     a return value is a number of record in Database file      ***/
/***         the value starts at 0 upto TotalRecord or NOTFOUND
***/
/*************************************************************
**/
long int Search(FILE *fp,char *StudentId,unsigned long int totalrecord)
{
    unsigned long int low,high,middle;
    low=0;  high=totalrecord-1;
    while(low<=high)
    {
        if(kbhit()) return USERCANCEL;
        middle=(low+high)/2;
        if( strcmp(StudentId,GetStudentId(fp,middle)) < 0 ) high=middle-1;
        else if( strcmp(StudentId,GetStudentId(fp,middle)) > 0 ) low =
middle+1;
        else return middle; /*found*/
    }
    return NOTFOUND;
```

```
}
void SearchDialogue(void)
{
  char *message[]={
                   "ค้นหาข้อมูลตามรหัสนักศึกษา",
                   "รหัสนักศึกษาที่ต้องการค้นหา",
                   "ไม่พบระเบียนของนักศึกษารหัสดังกล่าว",
                   "ป้อนรหัสนักศึกษาไม่สมบูรณ์",
                   "กำลังค้นหาข้อมูลตามรหัส",
                  };
  long int recnum;
  char choice=0,active_att=MAGENTA,deactive_att=BLUE;
  char Idrequest[9]="";
  char far *srh_buf;
  srh_buf=(char far *)farcalloc(38016L,sizeof(char));
  if(!srh_buf)
  {
    closegraph();
    printf("Alloc error in SearchDialogue function");
    exit(0);
  }
  Getvideo(19,5,62,13,srh_buf);
  Blockfill(20,6,60,12,POPUPBKG);
  Dimension(20,6,60,12,3,RIMUPPER,RIMLOWER);
  Shadow(20,6,60,12,3,8,POPUPSHA);
  Blockfill(29,7,51,7,HEADBKG);
  Border(29,7,51,7,2,HEADRIM);
  PutCenterString(29,51,7,message[0],HEADTEXT);
  DisplayString(23,9,message[1],BLACK,BLACK);
  Dimension(45,9,56,9,2,RIMLOWER,RIMUPPER);
  Blockfill(28,11,38,11,ICONORM);
  Border(28,11,38,11,1,ICONORIM);
  Shadow(28,11,38,11,1,5,ICONSHA);
  PutCenterString(28,38,11,"ตกลง",ICONTEXT);
  Blockfill(43,11,53,11,ICONORM);
  Border(43,11,53,11,1,ICONORIM);
  Shadow(43,11,53,11,1,5,ICONSHA);
  PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
  for(;;)
  {
    switch(choice)
    {
      case 0:
      {
        Language=ENG;
        strcpy(InputString,Idrequest);
        switch(GetThaiString(46,9,9,10,active_att,POPUPBKG))
        {
          case ESC:
          {
            Putvideo(19,5,62,13,srh_buf);
            farfree(srh_buf);
            return;
          }
          case ENTER: case TABKEY:// call search
          {
            strcpy(Idrequest,InputString);
            DisplayString(46,9,Idrequest,deactive_att,RED);

            Blockfill(28,11,38,11,ICONACTIVE);
            Border(28,11,38,11,1,ICONACRIM);
            Shadow(28,11,38,11,1,5,ICONSHA);
            PutCenterString(28,38,11,"ตกลง",ICONTEXT);
            choice=1;
          }break;
          case SHFTAB:
          {
            strcpy(Idrequest,InputString);
            DisplayString(46,9,Idrequest,deactive_att,RED);

            Blockfill(43,11,53,11,ICONACTIVE);
            Border(43,11,53,11,1,ICONACRIM);
            Shadow(43,11,53,11,1,5,ICONSHA);
            PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
            choice=2;
          }
        }
      }break;
      case 1:
      {
        while(!kbhit());
        switch(getch())
        {
          case ESC:
          {
            Putvideo(19,5,62,13,srh_buf);
            farfree(srh_buf);
            return;
          }
          case ENTER:
          {
            if(strlen(Idrequest) < 8)
            {
              ErrorDialogue(message[3]);
              choice=0;
              Blockfill(28,11,38,11,ICONORM);
              Border(28,11,38,11,1,ICONORIM);
              Shadow(28,11,38,11,1,5,ICONSHA);
              PutCenterString(28,38,11,"ตกลง",ICONTEXT);
              DisplayString(46,9,Idrequest,active_att,RED);
              break;
            }
            Blockfill(20,8,60,12,POPUPBKG);
            DisplayString(27,9,message[4],BLACK,BLACK);
            DisplayString(47,9,Idrequest,BLACK,BLACK);
            Blockfill(35,11,44,11,ICONACTIVE);
```

```
Border(35,11,44,11,1,ICONACRIM);
Shadow(35,11,44,11,1,5,ICONSHA);
PutCenterString(35,44,11,"ยกเลิก",ICONTEXT);
recnum=Search(StudentIdFile,Idrequest,TotalRecord);

Blockfill(20,8,60,12,POPUPBKG);
DisplayString(23,9,message[1],BLACK,BLACK);
Dimension(45,9,56,9,2,RIMLOWER,RIMUPPER);
DisplayString(46,9,Idrequest,deactive_att,deactive_att);

Blockfill(28,11,38,11,ICONACTIVE);
Border(28,11,38,11,1,ICONACRIM);
Shadow(28,11,38,11,1,5,ICONSHA);
PutCenterString(28,38,11,"ตกลง",ICONTEXT);

Blockfill(43,11,53,11,ICONORM);
Border(43,11,53,11,1,ICONORIM);
Shadow(43,11,53,11,1,5,ICONSHA);
PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
switch(recnum)
{
  case NOTFOUND :
  {
    ErrorDialogue(message[2]);
    choice=0;
    Blockfill(28,11,38,11,ICONORM);
    Border(28,11,38,11,1,ICONORIM);
    Shadow(28,11,38,11,1,5,ICONSHA);
    PutCenterString(28,38,11,"ตกลง",ICONTEXT);
    DisplayString(46,9,Idrequest,active_att,RED);
  }break;
  default:
  {
    CurrentRecord=recnum;
    ModifyData();
  }
 }
}break;
case TABKEY:
{
  Blockfill(28,11,38,11,ICONORM);
  Border(28,11,38,11,1,ICONORIM);
  Shadow(28,11,38,11,1,5,ICONSHA);
  PutCenterString(28,38,11,"ตกลง",ICONTEXT);

  Blockfill(43,11,53,11,ICONACTIVE);
  Border(43,11,53,11,1,ICONACRIM);
  Shadow(43,11,53,11,1,5,ICONSHA);
  PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
  choice=2;
}break;
case 0x00:
{
  if( getch()==SHFTAB )
  {
    Blockfill(28,11,38,11,ICONORM);
    Border(28,11,38,11,1,ICONORIM);
    Shadow(28,11,38,11,1,5,ICONSHA);
    PutCenterString(28,38,11,"ตกลง",ICONTEXT);
    choice=0;
    DisplayString(46,9,Idrequest,active_att,RED);
  }
 }break;
}
}break;
case 2:
{
  while(!kbhit());
  switch(getch())
  {
    case ENTER: case ESC:
    {
      Putvideo(19,5,62,13,srh_buf);
      farfree(srh_buf);
      return;
    }
    case TABKEY:
    {
      Blockfill(43,11,53,11,ICONORM);
      Border(43,11,53,11,1,ICONORIM);
      Shadow(43,11,53,11,1,5,ICONSHA);
      PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
      choice=0;
      DisplayString(46,9,Idrequest,active_att,RED);
    }break;
    case 0x00:
    {
      if(getch()==SHFTAB)
      {
        Blockfill(43,11,53,11,ICONORM);
        Border(43,11,53,11,1,ICONORIM);
        Shadow(43,11,53,11,1,5,ICONSHA);
        PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);

        Blockfill(28,11,38,11,ICONACTIVE);
        Border(28,11,38,11,1,ICONACRIM);
        Shadow(28,11,38,11,1,5,ICONSHA);
        PutCenterString(28,38,11,"ตกลง",ICONTEXT);
        choice=1;
      }
    }break;
  }
 }break;
}
}
```

```
void ErrorDialogue(char *errmessage)
{
  int len,leftx,rightx;
  char *message="ตกลง";
  char far *errd_buf;

  len=Stringlength(errmessage);

  leftx=36-(len/2);
  rightx=44+(len/2);

  errd_buf=(char far *)farcalloc(53760L,sizeof(char));
  if(!errd_buf)
  {
    closegraph();
    printf("Alloc error in ErrorDialogue function");
    exit(0);
  }
  sound(2000);delay(200);nosound();
  Getvideo(0,6,79,12,errd_buf);
  Blockfill(leftx,7,rightx,11,RED);
  Border(leftx,7,rightx,11,2,WHITE);
  Shadow(leftx,7,rightx,11,2,8,BLACK);
  PutCenterString(leftx+1,rightx,8,errmessage,WHITE);
  Blockfill(36,10,44,10,YELLOW);
  Border(36,10,44,10,1,BLACK);
  Shadow(36,10,44,10,1,5,ICONSHA);
  PutCenterString(36,44,10,message,RED);
  getch();
  Putvideo(0,6,79,12,errd_buf);
  farfree(errd_buf);
}
void SortDialogue(void)
{
  char *message[]={ "เรียงข้อมูล",
                    "กำลังทำการเรียงข้อมูลในดิสก์",
                    "ยกเลิก",
                    "ยกเลิกการเรียงข้อมูล",
                    "ทำการเรียงข้อมูลเสร็จสิ้นแล้ว",
                    "ตกลง",
                  };
  char far *srt_buf;
  srt_buf=(char far *)farcalloc(29376L,sizeof(char));
  if(!srt_buf)
  {
    closegraph();
    printf("Alloc error in SortDialogue function");
    exit(0);
  }
  Getvideo(24,5,57,13,srt_buf);
  Blockfill(25,6,55,12,POPUPBKG);
  Dimension(25,6,55,12,3,RIMUPPER,RIMLOWER);
  Shadow(25,6,55,12,3,8,POPUPSHA);
  Blockfill(33,7,45,7,HEADBKG);
  Border(33,7,45,7,2,HEADRIM);
  PutCenterString(33,45,7,message[0],HEADTEXT);
  PutCenterString(25,55,9,message[1],BLACK);
  Blockfill(36,11,44,11,ICONACTIVE);
  Border(36,11,44,11,1,ICONACRIM);
  Shadow(36,11,44,11,1,5,ICONSHA);
  DisplayString(38,11,message[2],ICONTEXT,ICONTEXT);
  if( Sort(StudentIdFile,TotalRecord)==USERCANCEL )
  {
    ErrorDialogue(message[3]);
    Putvideo(24,5,57,13,srt_buf);
  }
  else
  {
    Blockfill(25,9,55,9,POPUPBKG);
    PutCenterString(25,55,9,message[4],BLACK);
    Blockfill(36,11,44,11,ICONACTIVE);
    Border(36,11,44,11,1,ICONACRIM);
    Shadow(36,11,44,11,1,5,ICONSHA);
    DisplayString(38,11,message[5],ICONTEXT,ICONTEXT);
    sound(1000);delay(200);nosound();
    sound(4000);delay(200);nosound();
    getch();
    Putvideo(24,5,57,13,srt_buf);
  }
}
int GetmodifyRecord(void)
{
  char *message[]={ "แก้ไขข้อมูล",
                    "กรุณาป้อนหมายเลขระเบียนหรือรหัสนักศึกษา",
                    "ไม่มีระเบียนข้อมูลนี้อยู่",
                  };
  char choice=0;
  int len,i;
  char modId[9]="";
  char active_att=MAGENTA,deactive_att=BLUE;
  long int recnum;
  char far *gmr_buf;
  gmr_buf=(char far *)farcalloc(46656L,sizeof(char));
  if(!gmr_buf)
  {
    closegraph();
    printf("Alloc error in GetmodifyRecord function");
    exit(0);
  }
  Getvideo(14,5,67,13,gmr_buf);
  Blockfill(15,6,65,12,POPUPBKG);
  Dimension(15,6,65,12,3,RIMUPPER,RIMLOWER);
  Shadow(15,6,65,12,3,8,POPUPSHA);
  Blockfill(32,7,47,7,HEADBKG);
  Border(32,7,47,7,2,HEADRIM);
  PutCenterString(32,48,7,message[0],HEADTEXT);
```

```
DisplayString(17,9,message[1],BLACK,BLACK);
Dimension(52,9,62,9,2,RIMLOWER,RIMUPPER);
Blockfill(26,11,36,11,ICONORM);
Border(26,11,36,11,1,ICONORIM);
Shadow(26,11,36,11,1,5,ICONSHA);
PutCenterString(26,36,11,"ตกลง",ICONTEXT);
Blockfill(44,11,54,11,ICONORM);
Border(44,11,54,11,1,ICONORIM);
Shadow(44,11,54,11,1,5,ICONSHA);
PutCenterString(44,54,11,"ยกเลิก",ICONTEXT);
for(;;)
{
  switch(choice)
  {
    case 0:
    {
      Language=ENG;
      strcpy(InputString,modId);
      switch(GetThaiString(53,9,9,10,active_att,POPUPBKG))
      {
        case ESC:
        {
          Putvideo(14,5,67,13,gmr_buf);
          farfree(gmr_buf);
          return FAIL;
        }
        case ENTER: case TABKEY:
        {
          strcpy(modId,InputString);
          DisplayString(53,9,modId,deactive_att,RED);
          Blockfill(26,11,36,11,ICONACTIVE);
          Border(26,11,36,11,1,ICONACRIM);
          Shadow(26,11,36,11,1,5,ICONSHA);
          PutCenterString(26,36,11,"ตกลง",ICONTEXT);
          choice=1;
        }break;
        case SHFTAB:
        {
          strcpy(modId,InputString);
          DisplayString(53,9,modId,deactive_att,RED);
          Blockfill(44,11,54,11,ICONACTIVE);
          Border(44,11,54,11,1,ICONACRIM);
          Shadow(44,11,54,11,1,5,ICONSHA);
          PutCenterString(44,54,11,"ยกเลิก",ICONTEXT);
          choice=2;
        }
      }
    }break;
    case 1:
    {
      while(!kbhit());
      switch(getch())
      {
case ESC:
{
  Putvideo(14,5,67,13,gmr_buf);
  farfree(gmr_buf);
  return FAIL;
}
case ENTER:
{
  if(strlen(modId) < 8)
  {
    recnum=0;
    len=strlen(modId)-1;
    for(i=len;i>=0;i--)
    {
      recnum+=(long)((modId[len-i]-0x30)*pow(10,i));
    }
    if( (recnum<TotalRecord)&&(recnum>0) )
    {
      CurrentRecord=recnum-1;
      Putvideo(14,5,67,13,gmr_buf);
      farfree(gmr_buf);
      return PASS;
    }
    else
    {
      ErrorDialogue(message[2]);
      choice=0;

      Blockfill(26,11,36,11,ICONORM);
      Border(26,11,36,11,1,ICONORIM);
      Shadow(26,11,36,11,1,5,ICONSHA);
      PutCenterString(26,36,11,"ตกลง",ICONTEXT);
      DisplayString(53,9,modId,active_att,RED);
      break;
    }
  }
  recnum=Search(StudentIdFile,modId,TotalRecord);
  switch(recnum)
  {
    case NOTFOUND :
    {
      ErrorDialogue(message[2]);
      choice=0;
      Blockfill(26,11,36,11,ICONORM);
      Border(26,11,36,11,1,ICONORIM);
      Shadow(26,11,36,11,1,5,ICONSHA);
      PutCenterString(26,36,11,"ตกลง",ICONTEXT);
      DisplayString(53,9,modId,active_att,RED);
    }break;
    default:
    {
      CurrentRecord=recnum;
      Putvideo(14,5,67,13,gmr_buf);
```

```
            farfree(gmr_buf);
            return PASS;
          }
        }
      }break;
      case TABKEY:
      {
        Blockfill(26,11,36,11,ICONORM);
        Border(26,11,36,11,1,ICONORIM);
        Shadow(26,11,36,11,1,5,ICONSHA);
        PutCenterString(26,36,11,"ตกลง",ICONTEXT);
        Blockfill(44,11,54,11,ICONACTIVE);
        Border(44,11,54,11,1,ICONACRIM);
        Shadow(44,11,54,11,1,5,ICONSHA);
        PutCenterString(44,54,11,"ยกเลิก",ICONTEXT);
        choice=2;
      }break;
      case 0x00:
      {
        if( getch()==SHFTAB )
        {
          Blockfill(26,11,36,11,ICONORM);
          Border(26,11,36,11,1,ICONORIM);
          Shadow(26,11,36,11,1,5,ICONSHA);
          PutCenterString(26,36,11,"ตกลง",ICONTEXT);
          choice=0;
          DisplayString(53,9,modId,active_att,RED);
        }
      }break;
    }
  }break;
  case 2:
  {
    while(!kbhit());
    switch(getch())
    {
      case ENTER: case ESC:
      {
        Putvideo(14,5,67,13,gmr_buf);
        farfree(gmr_buf);
        return FAIL;
      }
      case TABKEY:
      {
        Blockfill(44,11,54,11,ICONORM);
        Border(44,11,54,11,1,ICONORIM);
        Shadow(44,11,54,11,1,5,ICONSHA);
        PutCenterString(44,54,11,"ยกเลิก",ICONTEXT);
        choice=0;
        DisplayString(53,9,modId,active_att,RED);
      }break;
      case 0x00:
      {
        if(getch()==SHFTAB)
        {
          Blockfill(44,11,54,11,ICONORM);
          Border(44,11,54,11,1,ICONORIM);
          Shadow(44,11,54,11,1,5,ICONSHA);
          PutCenterString(44,54,11,"ยกเลิก",ICONTEXT);
          PutCenterString(43,53,11,"ยกเลิก",ICONTEXT);
          Blockfill(26,11,36,11,ICONACTIVE);
          Border(26,11,36,11,1,ICONACRIM);
          Shadow(26,11,36,11,1,5,ICONSHA);
          PutCenterString(26,36,11,"ตกลง",ICONTEXT);
          choice=1;
        }
      }break;
    }
  }
}
int Gettime(int xpos,int ypos,char inputtime[9])
{
  char active_att=MAGENTA,deactive_att=BLUE;
  union inkey{
    int i;
    char ch[2];
  } c;
  int count=0,curx=0;

  Blockfill(xpos,ypos,xpos+10,ypos,POPUPBKG);
  DisplayString(xpos,ypos,inputtime,active_att,active_att);
  if(inputtime[7]!=' ')
  {
    curx=8;count=8;
  }
  DisplayChar(xpos+curx,ypos,230,active_att);

  for(;;)
  {
    while(!bioskey(1));
    c.i=bioskey(0);

    if( c.ch[0] == ESC ) return ESC;
    if( c.ch[0] == TABKEY || c.ch[0] == ENTER )
    {
      if(inputtime[7]!=' ')
      {
        inputtime[8]='\x0';
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        DisplayString(xpos,ypos,inputtime,deactive_att,deactive_att);
        return ENTER;
      }
      else
```

```
        sound(2000);delay(200);nosound();
      }
    }
    if( c.ch[1] == SHFTAB)
    {
      if(inputtime[7]!=' ')
      {
        inputtime[8]='\x0';
        DisplayChar(xpos+curx,ypos,230,POPUPBKG);
        DisplayString(xpos,ypos,inputtime,deactive_att,deactive_att);
        return SHFTAB;
      }
      else
      {
        sound(2000);delay(200);nosound();
      }
    }
    switch(count)
    {
      case 0:
      {
        if( is_in("012",c.ch[0]) )
        {
          inputtime[count]=c.ch[0];
          DisplayChar(xpos+curx,ypos,230,POPUPBKG);
          DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
          curx++;count++;
          DisplayChar(xpos+curx,ypos,230,active_att);
          break;
        }
        else
        {
          sound(2000);delay(200);nosound();
        }
      }break;
      case 1:
      {
        if( c.ch[0]==BKSP )
        {
          DisplayChar(xpos+curx,ypos,230,POPUPBKG);
          curx--;count--;
          DisplayChar(xpos+curx,ypos,229,POPUPBKG);
          DisplayChar(xpos+curx,ypos,230,active_att);
          break;
        }
        switch(inputtime[count-1])
        {
          case '0':
          {
            if( is_in("0123456789",c.ch[0]) )
            {
              inputtime[count]=c.ch[0];
              DisplayChar(xpos+curx,ypos,230,POPUPBKG);
              DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
              curx+=2;count+=2;
              DisplayChar(xpos+curx,ypos,230,active_att);
              break;
            }
            else
            {
              sound(2000);delay(200);nosound();
            }
          }break;
          case '1':
          {
            if( is_in("0123456789",c.ch[0]) )
            {
              inputtime[count]=c.ch[0];
              DisplayChar(xpos+curx,ypos,230,POPUPBKG);
              DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
              curx+=2;count+=2;
              DisplayChar(xpos+curx,ypos,230,active_att);
              break;
            }
            else
            {
              sound(2000);delay(200);nosound();
            }
          }break;
          case '2':
          {
            if( is_in("0123",c.ch[0]) )
            {
              inputtime[count]=c.ch[0];
              DisplayChar(xpos+curx,ypos,230,POPUPBKG);
              DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
              curx+=2;count+=2;
              DisplayChar(xpos+curx,ypos,230,active_att);
              break;
            }
            else
            {
              sound(2000);delay(200);nosound();
            }
          }break;
        }
      }break;
      case 2:break;
      case 3:
      {
        if( c.ch[0]==BKSP )
        {
          DisplayChar(xpos+curx,ypos,230,POPUPBKG);
          curx-=2;count-=2;
          DisplayChar(xpos+curx,ypos,229,POPUPBKG);
          DisplayChar(xpos+curx,ypos,230,active_att);
```

```
      break;
    }
    if( is_in("012345",c.ch[0]) )
    {
      inputtime[count]=c.ch[0];
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
      curx++;count++;
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    else
    {
      sound(2000);delay(200);nosound();
    }
  }break;
  case 4:
  {
    if( c.ch[0]==BKSP )
    {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    switch(inputtime[count-1])
    {
      case '0':case '1':case '2':case '3':case '4':case '5':
      {
        if( is_in("0123456789",c.ch[0]) )
        {
          inputtime[count]=c.ch[0];
          DisplayChar(xpos+curx,ypos,230,POPUPBKG);
          DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
          curx+=2;count+=2;
          DisplayChar(xpos+curx,ypos,230,active_att);
          break;
        }
        else
        {
          sound(2000);delay(200);nosound();
        }
      }break;
    }
  }break;
  case 5:break;
  case 6:
  {
    if( c.ch[0]==BKSP )
    {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx-=2;count-=2;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    if( is_in("012345",c.ch[0]) )
    {
      inputtime[count]=c.ch[0];
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
      curx++;count++;
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    else
    {
      sound(2000);delay(200);nosound();
    }
  }break;
  case 7:
  {
    if( c.ch[0]==BKSP )
    {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    if( is_in("0123456789",c.ch[0]) )
    {
      inputtime[count]=c.ch[0];
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      DisplayChar(xpos+curx,ypos,c.ch[0],active_att);
      curx++;count++;
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    else
    {
      sound(2000);delay(200);nosound();
    }
  }break;
  case 8:
  {
    if( c.ch[0]==BKSP )
    {
      DisplayChar(xpos+curx,ypos,230,POPUPBKG);
      curx--;count--;
      DisplayChar(xpos+curx,ypos,229,POPUPBKG);
      DisplayChar(xpos+curx,ypos,230,active_att);
      break;
    }
    else
```

```
        {
          sound(2000);delay(200);nosound();
        }
      }break;//end case 8
    }//end switch main
  }
}
int AppendReserveTime(void)
{
  char *message[]={ "ตกลง",
                    "ยกเลิก",
                    "ป้อนวันที่เรียนผิดพลาด",
                    "ไม่สามารถจองในช่วงเวลานี้ได้",
                    };
  int iconx[]={27,42};
  int x=29,y=6,choice=0;
  char active_att=MAGENTA,deactive_att=BLUE;
  char inputdate[10];
  FILE *memberftemp;
  long netstudent,totalstudent,i;
  if( (memberftemp=fopen("mftemp.$$$","wb+"))==NULL )
  {
    closegraph();
    printf("Can't open temporary file in AppendReserveTime function\n");
    exit(1);
  }

  DisplayRecordnumber(CurrentReserve,44,4,BLUE,YELLOW);
  Blockfill(iconx[0],14,iconx[0]+10,14,ICONORM);
  Border(iconx[0],14,iconx[0]+10,14,1,ICONORIM);
  Shadow(iconx[0],14,iconx[0]+10,14,1,5,ICONSHA);
  PutCenterString(iconx[0],iconx[0]+10,14,message[0],ICONTEXT);
  Blockfill(iconx[1],14,iconx[1]+10,14,ICONORM);
  Border(iconx[1],14,iconx[1]+10,14,1,ICONORIM);
  Shadow(iconx[1],14,iconx[1]+10,14,1,5,ICONSHA);
  PutCenterString(iconx[1],iconx[1]+10,14,message[1],ICONTEXT);
  for(;;)
  {
    switch(choice)
    {
      case 0:
      {

  Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
        strcpy(InputString,ReserveHeader.SubjectName);
        DisplayString(x+1,y+choice,InputString,active_att,active_att);
        Language=ENG;
        switch(
  GetThaiString(x+1,y+choice,30,32,active_att,POPUPBKG) )
        {
          case ESC:
          {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
          }
          case ENTER:case TABKEY:
          {
            strcpy(ReserveHeader.SubjectName,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=1;
          }break;
          case SHFTAB:
          {
            strcpy(ReserveHeader.SubjectName,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=8;
          }break;
        }
      }break;
      case 1:
      {

  Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
        strcpy(InputString,ReserveHeader.SubjectId);
        DisplayString(x+1,y+choice,InputString,active_att,active_att);
        Language=ENG;
        switch( GetThaiString(x+1,y+choice,9,32,active_att,POPUPBKG)
        )
        {
          case ESC:
          {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
          }
          case ENTER:case TABKEY:
          {
            strcpy(ReserveHeader.SubjectId,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=2;
          }break;
          case SHFTAB:
          {
            strcpy(ReserveHeader.SubjectId,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=0;
          }break;
```

```
        }
      }break;
      case 2:
      {

  Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
        strcpy(InputString,ReserveHeader.TeacherName);
        DisplayString(x+1,y+choice,InputString,active_att,active_att);
        Language=THA;
        switch(
GetThaiString(x+1,y+choice,30,32,active_att,POPUPBKG) )
        {
          case ESC:
          {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
          }
          case ENTER:case TABKEY:
          {
            strcpy(ReserveHeader.TeacherName,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=3;
          }break;
          case SHFTAB:
          {
            strcpy(ReserveHeader.TeacherName,InputString);
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

  DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=1;
          }break;
        }
      }break;
      case 3:
      {

  Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
        intToDay(ReserveHeader.Day);
        DisplayString(x+1,y+choice,InputString,active_att,active_att);
        Language=THA;
        switch(
GetThaiString(x+1,y+choice,10,32,active_att,POPUPBKG) )
        {
          case ESC:
          {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
          }
          case ENTER:case TABKEY:
          {
            ReserveHeader.Day=Daytoint(InputString);
            if(!ReserveHeader.Day)
            {
              sound(2000);delay(200);nosound();
              Blockfill(x,y+choice,x+10,y+choice,POPUPBKG);
              ErrorDialogue(message[3]);
              choice=3;break;
            }
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=4;
          }break;
          case SHFTAB:
          {
            ReserveHeader.Day=Daytoint(InputString);
            if(!ReserveHeader.Day)
            {
              sound(2000);delay(200);nosound();
              Blockfill(x,y+choice,x+10,y+choice,POPUPBKG);
              ErrorDialogue(message[3]);
              choice=3;break;
            }
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);

DisplayString(x+1,y+choice,InputString,deactive_att,deactive_att);
            choice=2;
          }break;
        }
      }break;
      case 4:
      {

  Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
        switch(Gettime(x+1,y+choice,rtime))
        {
          case ESC:
          {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
          }
          case ENTER:
          {
            Border(x,y+choice,x+33,y+choice,2,POPUPBKG);
            choice=5;
            ReserveHeader.BeginTime.hour= (((rtime[0]-0x30)*10)+
rtime[1]-0x30);
            ReserveHeader.BeginTime.minutes= (((rtime[3]-0x30)*10)
+rtime[4]-0x30);
            ReserveHeader.BeginTime.secs= (((rtime[6]-0x30)*10)+
rtime[7]-0x30);
```

```
            }break;
            case SHFTAB:
            {
               Border(x,y+choice,x+33,y+choice,2,POPUPBKG);
               choice=3;
               ReserveHeader.BeginTime.hour= (((rtime[0]-0x30)*10)+
rtime[1]-0x30);
               ReserveHeader.BeginTime.minutes= (((rtime[3]-0x30)*10)
+rtime[4]-0x30);
               ReserveHeader.BeginTime.secs= (((rtime[6]-0x30)*10)+
rtime[7]-0x30);
            }break;
         }
      }break;
      case 5:
      {

   Dimension(x,y+choice,x+33,y+choice,2,RIMLOWER,RIMUPPER);
         switch(Gettime(x+1,y+choice,xtime))
         {
            case ESC:
            {
               fclose(memberftemp);
               remove("mftemp.$$$");
               return ESC;
            }
            case ENTER:
            {
               Border(x,y+choice,x+33,y+choice,2,POPUPBKG);
               ReserveHeader.EndTime.hour= (((xtime[0]-0x30)*10)+
xtime[1]-0x30);
               ReserveHeader.EndTime.minutes= (((xtime[3]-0x30)*10)+
xtime[4]-0x30);
               ReserveHeader.EndTime.secs= (((xtime[6]-0x30)*10)+
xtime[7]-0x30);

   if(ReserveHeader.EndTime.hour>ReserveHeader.BeginTime.hour)
               {
                  choice=6;break;
               }
               else
               {

   if(ReserveHeader.EndTime.minutes>ReserveHeader.BeginTime.minutes)
                  {
                     choice=6;break;
                  }
                  else
                  {
                     ErrorDialogue(message[4]);
                     choice=5;break;
                  }
               }
            }
         }
      }
      case SHFTAB:
      {
         Border(x,y+choice,x+33,y+choice,2,POPUPBKG);
         choice=4;
         ReserveHeader.EndTime.hour= (((xtime[0]-0x30)*10)+
xtime[1]-0x30);
         ReserveHeader.EndTime.minutes= (((xtime[3]-0x30)*10)+
xtime[4]-0x30);
         ReserveHeader.EndTime.secs= (((xtime[6]-0x30)*10)+
xtime[7]-0x30);

if(ReserveHeader.EndTime.hour>ReserveHeader.BeginTime.hour)
         {
            choice=4;break;
         }
         else
         {

if(ReserveHeader.EndTime.minutes>ReserveHeader.BeginTime.minutes)
            {
               choice=4;break;
            }
            else
            {
               ErrorDialogue(message[4]);
               choice=5;break;
            }
         }
      }
   }break;
   case 6:
   {
      if(!ReserveHeader.Totalmember)
      {
         InputString[0]='\x0';
         netstudent=0; totalstudent=0;
      }
      switch(
GetReserveMember(memberftemp,&netstudent,&totalstudent) )
      {
         case ESC:
         {
            fclose(memberftemp);
            remove("mftemp.$$$");
            return ESC;
         }
         case TABKEY:
         {
            ReserveHeader.Totalmember=netstudent;
            choice=7;
         }break;
```

```
            case SHFTAB:
            {
              ReserveHeader.Totalmember=netstudent;
              choice=5;
            }break;
          }
        }break;
        case 7:/
        {
          Blockfill(iconx[0],14,iconx[0]+10,14,ICONACTIVE);
          Border(iconx[0],14,iconx[0]+10,14,1,ICONACRIM);
          PutCenterString(iconx[0],iconx[0]+10,14,message[0],
ICONTEXT);
          do
          {
            while(!kbhit());
            switch(getch())
            {
              case ESC:
              {
                fclose(memberftemp);
                remove("mftemp.$$$");
                return ESC;
              }
              case ENTER:
              {
                if(!ReserveHeader.SubjectName[0] ||
!ReserveHeader.SubjectId[0]
                    || !ReserveHeader.TeacherName[0] || !
ReserveHeader.Totalmember)
                {
                  sound(2000);delay(200);nosound();
                  choice=0;break;
                }
                fseek(ReserveHeaderFile,CurrentReserve*sizeof
(ReserveHeader)+13L,SEEK_SET);
                fwrite(&ReserveHeader,sizeof(ReserveHeader),1,
ReserveHeaderFile);
                if(CurrentReserve==TotalReserve) TotalReserve++;
                CurrentReserve++;
                fseek(memberftemp,0L,SEEK_SET);
                fseek(ReserveIdFile,0L,SEEK_END);
                for(i=0;i<netstudent;i++)
                {
                  do{
                    fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
                  } while(ReserveMember.StudentId[0]==DELCODE);
                  fwrite(&ReserveMember,sizeof(ReserveMember),1,
ReserveIdFile);
                }
                fclose(memberftemp);
                remove("mftemp.$$$");
                return ENTER;
              }
              case TABKEY:
              {
                Blockfill(iconx[0],14,iconx[0]+10,14,ICONORM);
                Border(iconx[0],14,iconx[0]+10,14,1,ICONORIM);
                PutCenterString(iconx[0],iconx[0]+10,14,message[0],
ICONTEXT);
                choice=8;
              }break;
              case SHFTAB:
              {
                Blockfill(iconx[0],14,iconx[0]+10,14,ICONORM);
                Border(iconx[0],14,iconx[0]+10,14,1,ICONORIM);
                PutCenterString(iconx[0],iconx[0]+10,14,message[0],
ICONTEXT);
                choice=6;
              }break;
            }
          } while( choice!=6&&choice!=8&&choice!=0 );
        }break;
        case 8:
        {
          Blockfill(iconx[1],14,iconx[1]+10,14,ICONACTIVE);
          Border(iconx[1],14,iconx[1]+10,14,1,ICONACRIM);
          PutCenterString(iconx[1],iconx[1]+10,14,message[1],
ICONTEXT);
          do
          {
            while(!kbhit());
            switch(getch())
            {
              case ESC:
              case ENTER:
              {
                fclose(memberftemp);
                remove("mftemp.$$$");
                return ESC;
              }
              case TABKEY:
              {
                Blockfill(iconx[1],14,iconx[1]+10,14,ICONORM);
                Border(iconx[1],14,iconx[1]+10,14,1,ICONORIM);
                PutCenterString(iconx[1],iconx[1]+10,14,message[1],
ICONTEXT);
                choice=0;
              }break;
              case SHFTAB:
              {
                Blockfill(iconx[1],14,iconx[1]+10,14,ICONORM);
                Border(iconx[1],14,iconx[1]+10,14,1,ICONORIM);
                PutCenterString(iconx[1],iconx[1]+10,14,message[1],
ICONTEXT);
```

```
                    choice=7;
                  }break;
               }
            } while( choice!=0&&choice!=7 );
         }break;
      }
   }
}
void DrawReserveDialogue(void)
{
   char *message[]={
                    "ของเวลาลำดับที่",
                    "วิชา",
                    "รหัสวิชา",
                    "อาจารย์ผู้สอน",
                    "วันที่เรียน",
                    "ตั้งแต่เวลา",
                    "จนถึงเวลา",
                    "รหัสนักศึกษา",
                   };
   Blockfill(15,3,65,15,POPUPBKG);
   Dimension(15,3,65,15,4,RIMUPPER,RIMLOWER);
   Shadow(15,3,65,15,4,8,POPUPSHA);
   Blockfill(28,4,52,4,BLUE);
   Border(28,4,52,4,2,LIGHTBLUE);
   DisplayString(31,4,message[0],YELLOW,RED);
   DisplayString(25,6,message[1],BLACK,RED);
   DisplayString(22,7,message[2],BLACK,RED);
   DisplayString(18,8,message[3],BLACK,RED);
   DisplayString(21,9,message[4],BLACK,RED);
   DisplayString(20,10,message[5],BLACK,RED);
   DisplayString(20,11,message[6],BLACK,RED);
   DisplayString(19,12,message[7],BLACK,RED);
}
void inttoDay(int d)
{
   switch(d)
   {
      case 0:InputString[0]=0;break;
      case 1:strcpy(InputString,"อาทิตย์");break;
      case 2:strcpy(InputString,"จันทร์");break;
      case 3:strcpy(InputString,"อังคาร");break;
      case 4:strcpy(InputString,"พุธ");break;
      case 5:strcpy(InputString,"พฤหัสบดี");break;
      case 6:strcpy(InputString,"ศุกร์");break;
      case 7:strcpy(InputString,"เสาร์");break;
   }
}
int Daytoint(char *d)
{
   if(strcmp(d,"อาทิตย์")==0) return 1;
   else if(strcmp(d,"จันทร์")==0) return 2;
   else if(strcmp(d,"อังคาร")==0) return 3;
   else if(strcmp(d,"พุธ")==0) return 4;
   else if(strcmp(d,"พฤหัสบดี")==0) return 5;
   else if(strcmp(d,"ศุกร์")==0) return 6;
   else if(strcmp(d,"เสาร์")==0) return 7;
   else return 0;
}
void InitialReserveHeader(void)
{
   strcpy(rtime," : : ");
   strcpy(xtime," : : ");
   ReserveHeader.SubjectName[0]='\x0';
   ReserveHeader.SubjectId[0]='\x0';
   ReserveHeader.TeacherName[0]='\x0';
   ReserveHeader.Day=0;
   ReserveHeader.Totalmember=0L;
   CurrentReserve=TotalReserve;
   DrawReserveDialogue();
}
int GetReserveMember(FILE *memberftemp,long *netstudent,long *
totalstudent)
{
char *message[]={ "ลำดับที่",
                  "รหัสนักศึกษา",
                  "ตกลง",
                  "ยกเลิก",
                  "รหัสนักศึกษาไม่สมบูรณ์",
                  };
long int currecnum=0,i;
int y=12;
union inkey{
   char ch[2];
   int i;
   }c;
char delfactor=0;
char active_att=MAGENTA,deactive_att=BLUE;

fseek(memberftemp,0L,SEEK_SET);
if(*netstudent)
{
   do{
      fread(&ReserveMember,sizeof(ReserveMember),1,memberftemp);
   } while( ReserveMember.StudentId[0]==DELCODE );
}
Dimension(39,y,58,y,2,RIMLOWER,RIMUPPER);
DisplayChar(60,y,231,BLUE);
DisplayChar(60,y,232,LIGHTBLUE);

strcpy(ReserveMember.SubjectId,ReserveHeader.SubjectId);
for(;;)
{
   strcpy(InputString,ReserveMember.StudentId);
   if( InputString[0]!=\x0' )
   {
```

```
        delfactor=0;
        for(;;)
        {
          if(currecnum)    DisplayChar(60,y,231,LIGHTBLUE);
          else  DisplayChar(60,y,231,BLUE);
          if(currecnum<*netstudent)  DisplayChar(60,y,232,LIGHTBLUE)
;
          else  DisplayChar(60,y,232,BLUE);
          DisplayRecordnumber(currecnum,30,y,POPUPBKG,BLUE);
          strcpy(InputString,ReserveMember.StudentId);
          Blockfill(40,y,47,y,LIGHTGREEN);
          DisplayString(40,y,InputString,active_att,active_att);

          while(!(c.i=bioskey(1)));
          if(c.ch[1]==DELETE)
          {
            c.i=bioskey(0);
            if(currecnum==*netstudent)
            { sound(2000);delay(200);nosound();}
            else
            {
              ReserveMember.StudentId[0]=DELCODE;
              *netstudent-=1;delfactor++;
              fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
              fwrite(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);

              i=currecnum+delfactor;
              fseek(memberftemp,0L,SEEK_CUR);
              while(ReserveMember.StudentId[0]==DELCODE && i<*
totalstudent)
              {
                i++;
                fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              }
              if(i>=*totalstudent)
              {
                ReserveMember.StudentId[0]='\x0';break;
              }
            }
          }
          else
          {
            Blockfill(40,y,47,y,POPUPBKG);
            DisplayString(40,y,InputString,active_att,active_att);
            break;
          }
        }
      }
      else
      {
        if(currecnum)    DisplayChar(60,y,231,LIGHTBLUE);
        else  DisplayChar(60,y,231,BLUE);
        if(currecnum<*netstudent)  DisplayChar(60,y,232,LIGHTBLUE);
        else  DisplayChar(60,y,232,BLUE);
        DisplayRecordnumber(currecnum,30,y,POPUPBKG,BLUE);
        Blockfill(40,y,50,y,POPUPBKG);
        DisplayString(40,y,InputString,active_att,active_att);
      }
      Language=ENG;
      switch( GetThaiString(40,y,9,12,active_att,POPUPBKG) )
      {
        case ESC: return ESC;
        case ENTER:
        {
          if(strlen(InputString)<8)
          {
            ErrorDialogue(message[4]);break;
          }
          else
          {
            strcpy(ReserveMember.StudentId,InputString);
            if(currecnum==*netstudent)
            {
              fseek(memberftemp,0L,SEEK_CUR);
              fwrite(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              ReserveMember.StudentId[0]='\x0';
              *netstudent+=1;*totalstudent+=1;
            }
            else
            {
              fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
              fwrite(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              fseek(memberftemp,0L,SEEK_CUR);
              i=currecnum+1;
              {
                i++;
                fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              } while(ReserveMember.StudentId[0]==DELCODE && i
<*totalstudent);
              if(i>=*totalstudent) ReserveMember.StudentId[0]='\x0';
            }
            currecnum++;
          }
        }break;
        case UP:
        {
          if(currecnum)
          {
            if(strlen(InputString)==8)
```

```
            {
              strcpy(ReserveMember.StudentId,InputString);
              if(currecnum==*netstudent)
              {
                fseek(memberftemp,0L,SEEK_CUR);
                *netstudent+=1;*totalstudent+=1;currecnum++;
              }
              else  fseek(memberftemp,(long)(-1)*sizeof
(ReserveMember),SEEK_CUR);
              fwrite(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
            }
            currecnum--;
            do
            {
              fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
              fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
              if( ReserveMember.StudentId[0] == DELCODE )
                fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
            } while(ReserveMember.StudentId[0]==DELCODE);
          }
          else
          {
            sound(2000);delay(200);nosound();break;
          }
        }break;
        case DOWN:
        {
          if(currecnum+1==*netstudent)
          {
            currecnum=*netstudent;
            ReserveMember.StudentId[0]='\x0';break;
          }
          else if(currecnum+1<*netstudent)
          {
            if(strlen(InputString)==8)
            {
              strcpy(ReserveMember.StudentId,InputString);
              fseek(memberftemp,(long)(-1)*sizeof(ReserveMember),
SEEK_CUR);
              fwrite(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
            }
            i=currecnum;
            currecnum++;
            fseek(memberftemp,0L,SEEK_CUR);
            do
            {
              i++;  if(i>=*totalstudent) break;
              fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
            } while(ReserveMember.StudentId[0]==DELCODE);
            if(i>=*totalstudent) ReserveMember.StudentId[0]='\x0';
          }
          else
          {
            sound(2000);delay(200);nosound();
            break;
          }
        }break;
        case TABKEY:
        {
          DisplayRecordnumber(0,30,y,POPUPBKG,BLUE);
          if(*netstudent)
          {
            fseek(memberftemp,0L,SEEK_SET);
            do
            {
              fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
            } while( ReserveMember.StudentId[0]==DELCODE );
            strcpy(InputString,ReserveMember.StudentId);
            Blockfill(40,y,50,y,POPUPBKG);
            DisplayString(40,y,InputString,deactive_att,deactive_att);
          }
          return TABKEY;
        }
        case SHFTAB:
        {
          DisplayRecordnumber(0,30,y,POPUPBKG,BLUE);
          if(*netstudent)
          {
            fseek(memberftemp,0L,SEEK_SET);
            do
            {
              fread(&ReserveMember,sizeof(ReserveMember),1,
memberftemp);
            } while( ReserveMember.StudentId[0]==DELCODE );
            strcpy(InputString,ReserveMember.StudentId);
            Blockfill(40,y,50,y,POPUPBKG);
            DisplayString(40,y,InputString,deactive_att,deactive_att);
          }
          return SHFTAB;
        }
      }
    }
  }
}
void getcurpath(void)
{
  strcpy(CurrentPath,"X:\");
  CurrentPath[0] = 'A' + getdisk();
```

```
    getcurdir(0, CurrentPath+3);
}
void OpenDatafile(void)
{
    char header[9];
    int option=0;
    getcurpath();
    strcpy(std_datfile,CurrentPath);
    strcat(std_datfile,"\\student.std");
    strcpy(header,"studfile");
    while( Openfile(1,&TotalRecord)==FAIL)
    {
        option=0;
        GetPathandFile(&option);
        strcpy(std_datfile,InputString);
        if(option==0);
        else if(option==1)
        {
            strcpy(std_datfile,CurrentPath);
            strcat(std_datfile,"\\student.std");
            if( Creatfile(&StudentIdFile,std_datfile,header)==PASS ) break;
            else
            {
                closegraph(); exit(0);
            }
        }
        else
        {
            closegraph(); exit(0);
        }
    }

    strcpy(header,"reserveH");
    strcpy(rsv_headfile,CurrentPath);
    strcat(rsv_headfile,"\\reserve.dat");
    while( Openfile(2,&TotalReserve)==FAIL)
    {
        option=1;
        GetPathandFile(&option);
        strcpy(rsv_headfile,InputString);
        if(option==0);
        else if(option==1)
        {
            strcpy(rsv_headfile,CurrentPath);
            strcat(rsv_headfile,"\\reserve.dat");
            if( Creatfile(&ReserveHeaderFile,rsv_headfile,header)==PASS )
break;
            else
            {
                closegraph(); exit(0);
            }
        }
        else
        {
            closegraph(); exit(0);
        }
    }

    strcpy(header,"reserveI");
    strcpy(rsv_stdfile,CurrentPath);
    strcat(rsv_stdfile,"\\reserve.std");
    while( Openfile(3,&TotalReserveStudent)==FAIL)
    {
        option=2;
        GetPathandFile(&option);
        strcpy(rsv_stdfile,InputString);
        if(option==0);
        else if(option==1)
        {
            strcpy(rsv_stdfile,CurrentPath);
            strcat(rsv_stdfile,"\\reserve.std");
            if( Creatfile(&ReserveIdFile,rsv_stdfile,header)==PASS ) break;
            else
            {
                closegraph(); exit(0);
            }
        }
        else
        {
            closegraph(); exit(0);
        }
    }
}
int Creatfile(FILE **fptr,char *fname,char *Textheader)
{
    unsigned long int number=0L;
    char *errmess="ไม่สามารถสร้างไฟล์ข้อมูลได้";
    if( (*fptr=fopen(fname,"wb+"))==NULL )
    {
        ErrorDialogue(errmess);
        return FAIL;
    }
    fwrite((char *)Textheader,sizeof(char),9,*fptr);
    fwrite((char *)&number,sizeof(number),1,*fptr);
    return PASS;
}
/*****************************************************/
/***                This procedure is                ***/
/***    Open file and Check whether there is a database file.    ***/
/*****************************************************/
int Openfile(int mode,unsigned long int *Nrecord)
{
    char Textheader[9];
    char header[9];
    char *errmsg="ไฟล์ดังกล่าวไม่ใช่ไฟล์ข้อมูล";
    switch(mode)
```

```
{
  case 1:
  {
    strcpy(header,"studfile");
    if( (StudentIdFile=fopen(std_datfile,"rb+"))==NULL )return FAIL;
    fread((char *)Textheader,sizeof(char),9,StudentIdFile);
    if( strcmp(Textheader,header)!=0 )
    {
      ErrorDialogue(errmsg);
      fclose(StudentIdFile);
      return FAIL;
    }
    fread((char *)Nrecord,sizeof(unsigned long int),1,StudentIdFile);
  }break;
  case 2:
  {
    strcpy(header,"reserveH");
    if( (ReserveHeaderFile=fopen(rsv_headfile,"rb+"))==NULL )
return FAIL;
    fread((char *)Textheader,sizeof(char),9,ReserveHeaderFile);
    if( strcmp(Textheader,header)!=0 )
    {
      ErrorDialogue(errmsg);
      fclose(ReserveHeaderFile);
      return FAIL;
    }
    fread((char *)Nrecord,sizeof(unsigned long
int),1,ReserveHeaderFile);
  }break;
  case 3:
  {
    strcpy(header,"reserveI");
    if( (ReserveIdFile=fopen(rsv_stdfile,"rb+"))==NULL )  return
FAIL;
    fread((char *)Textheader,sizeof(char),9,ReserveIdFile);
    if( strcmp(Textheader,header)!=0 )
    {
      ErrorDialogue(errmsg);
      fclose(ReserveIdFile);
      return FAIL;
    }
    fread((char *)Nrecord,sizeof(unsigned long int),1,ReserveIdFile);
  }break;
 }
 return PASS;
}
/*** mode use for determine below this      ***/
/*** mode=0 filename="student.std"      ***/
/*** mode=1 filename="reserve.dat"      ***/
/*** mode=2 filename="reserve.std"    ***/
/*** on return value mode use for determine */
/*** mode=0 creat new file          ***/
/*** mode=1 get new path success        ***/
/*** mode=2 cancel                  ***/
void GetPathandFile(int *mode)
{
 char *iconmsg[3]={   "ป้อนชื่อไฟล์ใหม่",
                      "สร้างใหม่",
                      "ยกเลิก",
                    };
 char *message[3]={   "กรุณาป้อนชื่อไฟล์ที่เก็บข้อมูลนักศึกษา",
                      "กรุณาป้อนชื่อไฟล์ที่เก็บข้อมูลการจองเวลา",
                      "กรุณาป้อนชื่อไฟล์ที่เก็บรายชื่อนักศึกษาในเวลาจอง",
                      };
 char errmsg[50];
 char *pname;
 int choice=0;
 int x[]={18,34,50};
 int att[]={LIGHTGREEN,GREEN,BLUE,YELLOW,RED,BLUE};
 char far *gpf_buf;
 gpf_buf=(char far *)farcalloc(36288L,sizeof(char));
 if(!gpf_buf)
 {
  closegraph();
  printf("Alloc error in GetPathandFile function");
  exit(0);
 }
 Getvideo(14,6,67,12,gpf_buf);
 Blockfill(15,7,65,11,POPUPBKG);
 Dimension(15,7,65,11,3,RIMUPPER,RIMLOWER);
 Shadow(15,7,65,11,3,8,POPUPSHA);
 switch(*mode)
 {
  case 0:strcpy(errmsg,"ไม่พบไฟล์ฐานข้อมูล student file");break;
  case 1:strcpy(errmsg,"ไม่พบไฟล์ฐานข้อมูล reserve header");break;
  case 2:strcpy(errmsg,"ไม่พบไฟล์ฐานข้อมูล reserve member");break;
 }
 PutCenterString(15,65,8,errmsg,RED);
 for(choice=0;choice<3;choice++)
 {
  Blockfill(x[choice],10,x[choice]+12,10,att[0]);
  Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
  Border(x[choice],10,x[choice]+12,10,1,att[1]);
  PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],att[2]);
 }
 choice=0;
 Blockfill(x[choice],10,x[choice]+12,10,att[3]);
 Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
 Border(x[choice],10,x[choice]+12,10,1,att[4]);
 PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],att[5]);
 for(;;)
 {
  while(!kbhit());
  switch(getch())
  {
   case TABKEY:
```

```
        {
          Blockfill(x[choice],10,x[choice]+12,10,att[0]);
          Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
          Border(x[choice],10,x[choice]+12,10,1,att[1]);
          PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],att
[2]);
          choice++;
          if(choice>2) choice=0;
          Blockfill(x[choice],10,x[choice]+12,10,att[3]);
          Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
          Border(x[choice],10,x[choice]+12,10,1,att[4]);
          PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],att
[5]);
        }break;
        case 0x00:
        {
          if(getch()==SHFTAB)
          {
            Blockfill(x[choice],10,x[choice]+12,10,att[0]);
            Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
            Border(x[choice],10,x[choice]+12,10,1,att[1]);
            PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],
att[2]);
            choice--;
            if(choice<0) choice=2;
            Blockfill(x[choice],10,x[choice]+12,10,att[3]);
            Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
            Border(x[choice],10,x[choice]+12,10,1,att[4]);
            PutCenterString(x[choice],x[choice]+12,10,iconmsg[choice],
att[5]);
          }
        }break;
        case ESC:
        {
          *mode=2;Putvideo(14,6,67,12,gpf_buf);
          farfree(gpf_buf);
          return;
        }
        case ENTER:
        {
          switch(choice)
          {
            case 0:
            {
              Blockfill(15,7,65,11,POPUPBKG);
              Blockfill(27,10,53,10,CYAN);
              Shadow(27,10,53,10,2,5,ICONSHA);
              Dimension(27,10,53,10,2,LIGHTCYAN,LIGHTBLUE);
              PutCenterString(27,53,10,"Enter คกคง-Esc ยกเลิก",
YELLOW);
              PutCenterString(15,65,7,message[*mode],RED);
              Blockfill(18,8,62,8,WHITE);
              DisplayString(18,8,"Path: ",GREEN,GREEN);
              GetEngString(24,8,127,38,MAGENTA,WHITE);
              if( InputString[0] != '\x0')
              {
                *mode=0;Putvideo(14,6,67,12,gpf_buf);
                farfree(gpf_buf);
                return;
              }
              else
              {
                Blockfill(15,7,65,11,POPUPBKG);
                Dimension(15,7,65,11,3,RIMUPPER,RIMLOWER);
                Shadow(15,7,65,11,3,8,POPUPSHA);
                PutCenterString(15,65,8,errmsg,RED);
                for(choice=0;choice<3;choice++)
                {
                  Blockfill(x[choice],10,x[choice]+12,10,att[0]);
                  Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA)
                  ;
                  Border(x[choice],10,x[choice]+12,10,1,att[1]);
                  PutCenterString(x[choice],x[choice]+12,10,iconmsg
[choice],
                    att[2]);
                }
                choice=0;
                Blockfill(x[choice],10,x[choice]+12,10,att[3]);
                Shadow(x[choice],10,x[choice]+12,10,1,5,ICONSHA);
                Border(x[choice],10,x[choice]+12,10,1,att[4]);
                PutCenterString(x[choice],x[choice]+12,10,iconmsg
[choice],att
                  [5]);
              }
            }break;
            case 1:
            {
              *mode=1;Putvideo(14,6,67,12,gpf_buf);
              farfree(gpf_buf);
              return;
            }
            case 2:
            {
              *mode=2;Putvideo(14,6,67,12,gpf_buf);
              farfree(gpf_buf);
              return;
            }
          }
        }break;
      }
    }
}
void interrupt com1()
{
  CommBuf[ CommDataPos++ ] = inportb( CommAddr );
  if ( CommDataPos == CommBufSize )
    CommDataPos = 0;
  outportb( 0x20 , EOI );
```

```
}
void interrupt com2()
{
 CommBuf[ CommDataPos++ ] = inportb( CommAddr );
 if ( CommDataPos == CommBufSize )
   CommDataPos = 0;
 outportb( 0x20 , EOI );
}
int InitSerialComm(int comno,int baudrate,int databit,int stopbit,int parity)
{
 char low_divisor,high_divisor,comm_spec;
 if ( installed )
   return( E_INSTALLED );
 if ( comno != COM1 && comno != COM2 )
   return( E_COMN );
 if ( baudrate < B1200 || baudrate > B19200 )
   return( E_BAUD );
 if ( databit != D7 && databit != D8 )
   return( E_DATABIT );
 if ( stopbit != S1 && stopbit != S2 )
   return( E_STOPBIT );
 if ( parity < PNONE || parity > PEVEN )
   return( E_PARITY );
 switch( comno )
 {
   case COM1 : CommAddr = COM1PORT; break;
   case COM2 : CommAddr = COM2PORT; break;
 }
 CommBuf = (char *) malloc(CommBufSize);
 if ( CommBuf == NULL )
   return( E_MEMORY );
 low_divisor = baud[baudrate];
 high_divisor = 0;
 comm_spec = Data[databit] | stop[stopbit] | pari[parity];
 outportb( CommAddr + LCR , comm_spec | 0x80 );
 outportb( CommAddr + DLL , low_divisor      );
 outportb( CommAddr + DLH , high_divisor     );
 outportb( CommAddr + LCR , comm_spec & 0x7F );
 outportb( CommAddr + IER , 1                ); /* enable int : Rx Ready */
 outportb( CommAddr + MCR , 0x0B             ); /* OUT2(3),RTS(1) */
 inportb( CommAddr + LSR );
 inportb( CommAddr + MSR );
 inportb( CommAddr + IIR );
 switch( comno )
 {
   case COM1 :   old1 = getvect( COM1INT );
                 setvect( COM1INT , com1 );
                 outportb( IMR , inportb(IMR) & (~IRQ4FLAG) );
                 break;
   case COM2 :   old2 = getvect( COM2INT );
                 setvect( COM2INT , com2 );
                 outportb( IMR , inportb(IMR) & (~IRQ3FLAG) );
                 break;
 }
 installed = comno;
 return( PASS );
}
void QuitSerialComm(void)
{
 if ( ! installed )
   return;
 switch( installed )
 {
   case COM1 :   setvect( COM1INT , old1 );
                 outportb( IMR , inportb(IMR) | IRQ4FLAG );
                 break;
   case COM2 :   setvect( COM1INT , old2 );
                 outportb( IMR , inportb(IMR) | IRQ3FLAG );
                 break;
 }
 free(CommBuf);
}
void ResetBuffer(void)
{
 CommDataPos=0;
 memset((char *)CommBuf,0,CommBufSize);
}
int xmit(ch)     /* transmit a character via serial port */
unsigned char ch;
{
 int i;
 for ( i=0; i<TXCOUNT; i++ )
   if ( inportb( CommAddr + LSR ) & 0x20 ) /* THR empty? */
   {
     outportb( CommAddr , ch );
     return( PASS );
   }
 return(FAIL);
}
void main(void)
{
  Initialize();
  MenuManage();
  ExitProgram();
}
```

```
;   ************************************
;   *        MAIN  PROGRAM             *
;   *        DEVELOP BY SUNG           *
;   *        START  30/1/94            *
;   *        END    31/1/94            *
;   ************************************
;   R2    : LOCAL COUNTER
;   R3    : GOBAL VAR
      CPU   "8048.TBL"
      HOF   "BIN8"
      ORG   00H
DBUF:  EQU   36          ;DISPLAY BUFFER
DADDR: EQU   32          ;ADDRESS OF
BUFFER OUTPUT DISPLAY
IDADDR: EQU  56          ;ADDRESS OF FIRST
BUFFER STUDENT ID
PDIP:  EQU   0FBH        ;PORT DIPSWITCH
PBAR:  EQU   0FDH        ;PORT BARCOAD
PISDN: EQU   0FEH
      SEL   RB0
      ORL   P2,#0FBH
      CALL  INITLCD      ;INITIAL LCD
      JMP   MAIN
      ORG   007H
      JMP   TIME
MAIN: MOV   R0,#32
      MOV   @R0,#00H
      CALL  DISPAGE
    ; CALL  BARC
      CALL  CHKON
      JB0   TEST1
      MOV   R0,#DADDR
      MOV   @R0,#0C8H
      CALL  DISPLY
      JMP   MAIN
TEST1: CALL  OPEN
      JMP   MAIN
;*************************************
;******      PAGE ISDN          ******
;*************************************
      ORG   200H
CHKON: MOV   R0,#DADDR
      MOV   @R0,#64H
      CALL  DISPAGE
      MOV   R1,#PDIP
      MOVX  A,@R1
      RET
CHKEND: MOV   R0,#PDIP
      MOVX  A,@R0
      RET
;*************************************
;******      PAGE TABLE 1       ******
;*************************************
      ORG   300H
      DFB   "                "     ;00
      DFB   "                "     ;14
      DFB   "                "     ;28
      DFB   "                "     ;3C
      DFB   " LOGOUT COMPLETE "    ;50
      DFB   "Computer Service 1.0"  ;64
      DFB   " You can turn PC on "  ;78
      DFB   " in 3 minutes left "   ;8C
      DFB   " Turn on PC again. "   ;A0
      DFB   "You must wait 15 sec"  ;B4
      DFB   " You can't use at "    ;C8
      DFB   "this time. GETOUT !"   ;DC
;*************************************
;******      PAGE TABLE 2       ******
;*************************************
      ORG   400H
      DFB   "First , Insert your "  ;00
      DFB   "card to read barcode"  ;14
      DFB   " CHECK ERROR FIND! "   ;28
      DFB   " Insert card again "   ;3C
      DFB   "Student Identity No."  ;50
      DFB   " YOUR REQUEST IS IN "  ;64
      DFB   " ACTIVE PLEASE WAIT "  ;78
      DFB   " Time is running out"  ;8C
      DFB   "in 10 minutes left. "  ;A0
      DFB   "Warning! please save"  ;B4
      DFB   "data in your diskete"  ;C8
      DFB   " FINAL COUNTDOWN "     ;DC
LINEP: MOV   R3,#20      ;SET COUNTER
FOR 20 CHARACTER DISPLAY
      MOV   R0,#DADDR
LINEP1: MOV   A,@R0
      MOVP  A,@A
      CALL  WRBYTE
      MOV   A,#30H
      CALL  DELAY
      INC   @R0
      DJNZ  R3,LINEP1
      RET
;*************************************
;**********      PAGE TIME      *******
;*************************************
      ORG   500H
;****TIMER INTERRUPT SERVICE ROUTINE
TIME:  SEL   RB1
      MOV   R6,A
      MOV   A,#06H
      MOV   T,A
      MOV   R0,#35
;1/100 SECOND ADDRESS
      MOV   R1,#34       ;SECOND ADDRESS
      MOV   A,@R0
      DEC   A
;DECREASE 1/100 SECOND
      MOV   @R0,A
      JNZ   TIME2
      MOV   @R0,#64H
;SET 1 SECOND AGAIN
      MOV   A,@R1
      DEC   A
;DECRESE 1 SECOND
      MOV   R7,A
      ORL   A,#0F0H
      XRL   A,#0FFH
      JNZ   TIME1
      MOV   A,R7
      ADD   A,#0FAH
      MOV   R7,A
```

```
TIME1: MOV   A,R7
       MOV   @R1,A
       XRL   A,#0F9H
       JNZ   TIME2
       MOV   @R1,#59H
       MOV   R0,#33
;MINUTE ADDRESS
       MOV   A,@R0
       DEC   A
;DECRESE 1 MINUTE
       MOV   @R0,A
       XRL   A,#0FFH
       JNZ   TIME2
       MOV   @R0,#0FFH
TIME2: MOV   A,R6
       SEL   RB0
       RETR
;*****PROGRAM INITIAL SET TIMER
;****INTERRUPT WAIT 1 SEC
WAITS: MOV   R0,#35          ;SET FOR 1
SECOND
       MOV   @R0,#64H
       MOV   A,#07
       MOV   T,A
       EN    TCNTI
       STRT  T
       RET
;***** DELAY SUBROUTINE *****
DELAY: MOV   R2,#00H
;A=3 EQU 2MS
DE2:   DJNZ  R2,DE2
       DEC   A
       JNZ   DELAY
       RET
; ****MAKE TIME TO DISPLAY BUFFER
OUTD:  MOV   R0,#38
;ADDRESS OF FIRST CHARACTER
       MOV   R1,#33
;ADDRESS OF MIN
       MOV   R2,#02H
OUTD1: MOV   A,@R1
       MOV   R7,A
       SWAP  A
       ANL   A,#0FH
       ADD   A,#30H
       MOV   @R0,A
       INC   R0
       MOV   A,R7
       ANL   A,#0FH
       ADD   A,#30H
       MOV   @R0,A
       INC   R0
       MOV   @R0,#3AH
;FOR ASCII " : "
       INC   R0
       INC   R1
       DJNZ  R2,OUTD1
       DEC   R0
       MOV   @R0,#20H
       RET
WAIT10M:MOV  R0,#33
;INITIAL TIMER FOR WAIT 10 MIN
       MOV   @R0,#09H
       INC   R0
       MOV   @R0,#59H
       CALL  WAITS
       MOV   R0,#44
;CHECK POWER OFF BEFORE TIME OUT
       MOV   @R0,#00
       JMP   WAIT
WAIT3M: MOV  R0,#33
;INITIAL TIMER FOR WAIT 3 MIN
       MOV   @R0,#03H
       INC   R0             ;SET FOR 60 SECOND
       MOV   @R0,#00H
       CALL  WAITS
       MOV   R0,#44
;CHECK POWER ON BEFORE TIME OUT
       MOV   @R0,#01
WAIT:  MOV   R0,#DADDR
       MOV   @R0,#00        ;WORD FROM TABLE
       CALL  CLRLCD
       CALL  BANK
       CALL  OUTD
       CALL  BUFDIS
WAIT1: MOV   A,#0CH
       CALL  EPULSE
       MOV   A,#06H
       CALL  EPULSE
       CALL  OUTD
       CALL  BUFD1
       MOV   R1,#44
;FOR CHECK FEEDBACK POWER
       MOV   A,@R1
       JB0   WAIT2
       MOV   A,#07BH
;CHECK POWER OFF
       OUTL  P2,A
       IN    A,P2
       ANL   A,#40H
       JNZ   WAIT3
       JMP   WAIT4
WAIT2: MOV   A,#07BH
;CHECK POWER ON
       OUTL  P2,A
       IN    A,P2
       JB6   WAIT4
WAIT3: MOV   R0,#33
       MOV   A,@R0
       XRL   A,#0FFH
       JNZ   WAIT1
       STOP  TCNT
       MOV   A,#00H
;TIME RUNNING OUT
       RET
WAIT4: STOP  TCNT
       MOV   A,#0FFH
;TURN OFF BEFORE END
       RET
;**** PROGRAM FOR WAIT 15 SECOND BEFORE
;****OPEN PC AGAIN
WAT15S: MOV  R0,#33
```

```
;SET MINUTE = 0 MIN
    MOV   @R0,#00
    INC   R0
    MOV   @R0,#15H
;SET SECOND = 15 SEC
    CALL  WAITS
    CALL  BANK
    CALL  OUTD
    MOV   R0,#DADDR
    MOV   @R0,#30H
    CALL  CLRLCD
    CALL  BUFDIS
W15S1: MOV   A,#0CH
    CALL  EPULSE
    MOV   A,#06H
    CALL  EPULSE
    CALL  OUTD
    CALL  BUFD1
    MOV   R0,#34
    MOV   A,@R0
    XRL   A,#00H
    JNZ   W15S1
    STOP  TCNT
    RET
;*********************************************
;****        PAGE DISPLAY        ****
;*********************************************
    ORG   600H
; ***** INITIAL LCD DISPLAY *******
; P1.0-P1.3 : PIN D4-D7 (DATA READ/WRITE LCD)
; P1.4    : PIN RS    (REGISTER SELECTION)
; P1.5    : PIN R/W   (READ/WRITE)
; P1.6    : PIN E     (ENABLE SIGNAL PULSE)
;*****************EVENT****************
INITLCD:MOV   A,#0FBH
    OUTL  P2,A
    MOV   A,#02H
;SETS TO 4 BIT OPERATION
    CALL  EPULSE
;ENABLE SIGNAL PLUSE (WRITE)
    MOV   A,#06H
;WAIT MORE THAN 4.1 MS
    MOV   A,#02H      ;FUNCTION SET  28
    CALL  EPULSE      ;DL= 0 4 BIT, N=1 2
LINE ,F=0 5*7 DOT
    MOV   A,#08H
    CALL  EPULSE
    MOV   A,#00H
;DISPLAY ON-OFF CONTROL
    CALL  EPULSE      ;D=1 ON,C=0
CURSOR OFF,B=0 BLINK OFF
    MOV   A,#0CH
    CALL  EPULSE
    MOV   A,#00H      ;ENTRY MODE SET
    CALL  EPULSE      ;I/D=1
INCREMENT,S=0 RIGHT
    MOV   A,#06H
    CALL  EPULSE
    CALL  CLRLCD      ;CLEAR DISPLAY
    RET
; ***** CLEAR DISPLAY *****
CLRLCD: MOV   A,#00H
    CALL  EPULSE
    MOV   A,#01H
    CALL  EPULSE
    MOV   A,#03H
    CALL  DELAY
    RET
; ***** WRITE DATA BYTE TO LCD *****
WRBYTE: MOV   R7,A
    SWAP  A
    ANL   A,#0FH
    ORL   A,#50H
    CALL  EPULSE
    MOV   A,R7
    ANL   A,#0FH
    ORL   A,#50H
    CALL  EPULSE
    RET
; ***** GENERRATE ENABLE PLUSE & SENT
; ***** DATA OR INSTRUSTION
EPULSE: ORL   A,#40H
    OUTL  P1,A
    MOV   R2,#0FFH
EP1:  DJNZ  R2,EP1
    ANL   A,#0BFH
    OUTL  P1,A
    RET
;****FOR BANK DISPLAY RAM 1 LINE
BANK:  MOV   R2,#08H
    MOV   R0,#DBUF
BANK1: MOV   @R0,#20H
    INC   R0
    DJNZ  R2,BANK1
    RET
DISPLY: CALL  CLRLCD
    CALL  LINE
    MOV   A,#0CH
    CALL  EPULSE
    MOV   A,#00H
    CALL  EPULSE
    CALL  LINE
DIS1:  MOV   R3,#12
DIS2:  MOV   A,#00H
    CALL  DELAY
    DJNZ  R3,DIS2
    RET
LINE:  MOV   R3,#20      ;SET COUNTER
FOR 20 CHARACTER DISPLAY
    MOV   R0,#DADDR
LINE1: MOV   A,@R0
    MOVP3 A,@A
    CALL  WRBYTE
    MOV   A,#30H
    CALL  DELAY
    INC   @R0
    DJNZ  R3,LINE1
    RET
BUFDIS: MOV   R0,#DADDR
    MOV   @R0,#0DCH
    CALL  LINEP
BUFD1: MOV   A,#0CH
```

```
    CALL  EPULSE
    MOV   A,#06H
    CALL  EPULSE
BUFD2: MOV   R3,#08H
    MOV   R0,#DBUF
BUFD3: MOV   A,@R0
    CALL  WRBYTE
    INC   R0
    DJNZ  R3,BUFD3
    RET
DISPAGE:CALL  CLRLCD
    CALL  LINEP
    MOV   A,#0CH
    CALL  EPULSE
    MOV   A,#00H
    CALL  EPULSE
    CALL  LINEP
    JMP   DIS1
;*********************************************
;********       PAGE USER       *******
;*********************************************
    ORG   700H
;***** PROGRAM FOR USER INTERFACE *****
OPEN:  MOV   R0,#DADDR
    MOV   @R0,#78H
    CALL  DISPLY
    CALL  WAIT3M
    JZ    CLOSE
    CALL  CLRLCD
    MOV   A,#7BH
    OUTL  P2,A
    MOV   R0,#DADDR
    MOV   @R0,#64H
    CALL  LINE
OPEN1: CALL  CHKEND      ;CHECK TIME
OUT
    JZ    EXIT       ;ASK FOR TIME OUT
    MOV   A,#7BH     ;CHECK  YOU HAVE
TURN PC OFF
    OUTL  P2,A
    IN    A,P2
;IF CLOSE WILL WAIT 15 SEC BEFORE OPEN
    JB6   OPEN1      ;BIT FEEDBACK
    JMP   OPEN2
OPEN2: MOV   A,#0FBH
    OUTL  P2,A
    MOV   R0,#DADDR
    MOV   @R0,#0A0H
    CALL  DISPLY
    CALL  WAT15S
    JMP   OPEN
EXIT:  MOV   R0,#DADDR
    MOV   @R0,#08CH
    CALL  DISPAGE
    CALL  DISPAGE
    CALL  WAIT10M
CLOSE: MOV   R0,#DADDR
    MOV   @R0,#050H
    CALL  CLRLCD
    CALL  LINE
    MOV   A,#0FBH
    OUTL  P2,A
    JMP   DIS1
;***** PROGRAM FOR BARCODE INTERFACE
BARC:  MOV   A,#0FBH
    OUTL  P2,A
    MOV   R2,#08H     ;FOR CLEAR
STUDENT ID BUFFER = FFH
    MOV   R0,#IDADDR
BARC1: MOV   @R0,#0FFH
    INC   R0
    DJNZ  R2,BARC1
BARC2: ORL   P2,#06H     ;SET RDY &
STORBE
    IN    A,P2       ;CHECK STORBE = LOW
    JB1   BARC2
    ANL   P2,#0FBH   ;SET RDY=0
    ORL   P2,#02H    ;SET STORBE=1
    MOV   R1,#PBAR
    MOVX  A,@R1
;CHECK START BYTE = FFH
    XRL   A,#0FFH
    JNZ   BARC2
    MOV   R0,#IDADDR
BARC3: ORL   P2,#06H     ;SET RDY &
STORBE
    IN    A,P2       ;IS STORBE = LOW
    JB1   BARC3
    ANL   P2,#0FBH   ;SET RDY=0
    ORL   P2,#02H    ;SET STORBE=1
    MOV   R1,#PBAR
    MOVX  A,@R1
    MOV   @R0,A
    INC   R0
    XRL   A,#0F0H
    ;CHECK STOP BYTE = F0H
    JNZ   BARC3
; ** CHECK STUDENT ID NUMBER IS OVER 9
    MOV   R0,#IDADDR
    MOV   R2,#08H
BARC4: MOV   A,@R0
    ADD   A,#0F6H
    JNC   BARC5
; *** FOR DISPLAY ERROR MASSAGE ******
    MOV   R0,#DADDR
    MOV   @R0,#28H
    CALL  DISPAGE
    JMP   BARC
;****MAKE STUDENT ID TO ACSII NUMBER
BARC5: MOV   A,#30H
;MAKE ACSII NUMBER
    ADD   A,@R0
    MOV   @R0,A
    INC   R0
    DJNZ  R2,BARC4
;****SHOW STUDENT ID NUMBER *******
    CALL  CLRLCD
    MOV   R0,#DADDR
    MOV   @R0,#50H
    CALL  LINEP
    MOV   A,#0CH
    CALL  EPULSE
```

```
                MOV   A,#06H
                CALL  EPULSE
                MOV   R3,#08H
                MOV   R0,#IDADDR
                MOV   R1,#DBUF
BARC7:          MOV   A,@R0
                MOV   @R1,A
                INC   R0
                INC   R1
                DJNZ  R3,BARC7
                CALL  BUFD1
                MOV   R3,#08H
BARC9:          CLR   A
                CALL  DELAY
                DJNZ  R3,BARC9
                RET
;   ***************************************************
;   *   PROGRAM FOR LINK MC145425>PIC16C55<8749
;   *        ON SLAVE BOARD
;   *    DEVELOPE BY SAKSIT SUKAPIBAN
;   *    ELECTRONICS ENGINEER 4TH KMITL
;   *       V 1.0  COPYRIGHT 1995
;   ***************************************************
TX      EQU   0
CARRY   EQU   0
INT     EQU   0
W       EQU   0
SAME    EQU   1
BCLK    EQU   1
ENDR    EQU   1
VD      EQU   2
Z       EQU   2
EN1     EQU   3
STATUS  EQU   3
ENDS    EQU   3
PORTA   EQU   5
CRCV    EQU   5
PORTB   EQU   6
RELAY   EQU   6
PORTC   EQU   7
RX      EQU   7
ID      EQU   8
COUNT   EQU   8
CBIT    EQU   9
CBYT    EQU   .10
CSND    EQU   .13
COMBYT  EQU   .13
CKSUM   EQU   .16
FBDATAR EQU   .11
FBDATAS EQU   .24
CHECKBIT EQU  .31
;  * SETUP *
SETUP   CLRW
        MOVLW   2       ; SET A1 TO I/P PORT
        TRIS    PORTA   ;
        MOVLW   0FH     ; SET B0123 I/P B67 O/P
        TRIS    PORTB
        CLRF    ID
        MOVLW   1       ; SET ID NUMBER OF TERMINAL
        MOVWF   ID      ; SAVE IN F8
        CLRF    CBIT
        CLRF    CHECKBIT
        RETLW   0
CHKEN1  BTFSS   PORTB,EN1
        GOTO    CHKEN1
        BTFSS   PORTB,VD
        GOTO    CHKEN1
        RETLW   0
CHKENS  BTFSS   PORTB,EN1
        GOTO    CHKENS
        BTFSS   PORTB,BCLK
        GOTO    CHKENS
        RETLW   0
;CHECKID  IF CORRECT SET CHECKBIT,0
CHKID   BCF     CHECKBIT,0
        MOVF    .12,W
        XORWF   8,W
        BTFSC   STATUS,Z
        RETLW   0
        BSF     CHECKBIT,0
        RETLW   0
; CHECK  COMMAND=00H  IF YES DO NOTHING  & SET CHECKBIT,3
COM00   BCF     CHECKBIT,3
        MOVLW   00H
        XORWF   COMBYT,W
        BTFSS   STATUS,Z
        RETLW   0
        BSF     CHECKBIT,3
        RETLW   0
; RECIEVE DATA FROM 8749 11 BYTE
TENRCV  CLRW
        BSF     PORTA,ENDS
        BTFSS   PORTA,ENDR
        GOTO    TENRCV
        MOVF    PORTC,W
        MOVWF   0
        INCF    4,SAME
        DECFSZ  CBYT,SAME
        GOTO    TENRCV
        MOVLW   8       ; SET PORTA3 I/P A012 O/P
        TRIS    PORTA
        RETLW   0
; SEND DATA TO SLAVE   13 BYTE FF-ID-IDST-COMMAND-CHKS-F0  F11-F23
SNDSLAVE BSF    PORTB,RELAY
        MOVLW   .255
        MOVWF   .27
ONTIME  DECFSZ  .27,SAME
        GOTO    ONTIME
        MOVLW   CSND
        MOVWF   CBYT
        MOVLW   COUNT
        MOVWF   CBIT
        MOVLW   FBDATAR
        MOVWF   4
        MOVLW   0FFH
        MOVWF   .11     ; FFH START
        MOVF    8,W
        MOVWF   .12     ; ID MACHINE
        CALL    CHKENS
TBCLK   BTFSS   PORTB,BCLK
        GOTO    TBCLK
        BTFSS   0,0     ;CHK 1ST BYTE =0 OR 1
```

```
        GOTO    BITLOW
        BSF     PORTB,RX
LATER   RRF     0,SAME
        DECFSZ  CBIT,SAME
        CALL    CHKEN3
        GOTO    TBCLK
        INCF    4,SAME
        CALL    CHKEN1
        DECFSZ  CBYT,SAME
        GOTO    TBCLK
        BCF     PORTB,RELAY   ; OFF RELAY
        RETLW   0
BITLOW  BCF     PORTB,RX
        GOTO    LATER
;RECIEVE DATA 5 BYTE FROM SLAVE    F24-F28
RECIEVE MOVLW   COUNT
        MOVWF   CBIT
        MOVLW   CRCV      ; 5 BYTE
        MOVWF   CBYT
        MOVLW   FBDATAS   ; F24
        MOVWF   4
EN1     CALL    CHKEN1
TBCLK2  BTFSS   PORTB,BCLK
        GOTO    TBCLK2
        BTFSS   PORTB,TX
        GOTO    BITLO
        BSF     0,0
LATER2  RLF     0,SAME
        DECFSZ  CBIT,SAME
        GOTO    TBCLK2
        INCF    4,SAME
        DECFSZ  CBYT,SAME
        GOTO    EN1
        RETLW   0
BITLO   BCF     0,0
        GOTO    LATER2
;SEND DATA  TO 8749 3 BYTE FF-COMMAND-F0
TENRS
        BSF     PORTA,ENDR
        BTFSS   PORTA,ENDS
        GOTO    TENRS
        RETLW   0
SEND
        CLRW
        TRIS    PORTC
        MOVLW   0FFH
        CALL    TENRS
        MOVF    24,W      ; FF
        MOVWF   PORTC
        CALL    TENRS
        MOVF    26,W      ; COMMAND BYTE
        MOVWF   PORTC
        CALL    TENRS
        MOVLW   0F0H      ; F0
        MOVWF   PORTC
        MOVLW   2
        TRIS    PORTA     ; SET PORT A1 I/P 023 O/P
        RETLW   0
;  CHECK SUM BEFORE SEND TO MASTER
CHKSUM
        CLRW
        MOVLW   0F0H      ; SET STOP BYTE F23
        MOVWF   23
        CLRF    22
        MOVLW   .10       ; COUNT BYTE = 10
        MOVWF   29
        MOVLW   .13
        MOVWF   4
        MOVF    8,W
        ADDWF   .13,W
        MOVWF   22
        DECF    29,SAME
SUM
        INCF    4,SAME
        ADDWF   0,W
        MOVWF   22
        DECFSZ  29,SAME
        GOTO    SUM
        RETLW   0
;   CHECK SUM  FOR DATA FROM MASTER  & COMPARE  OLD CHKSUM
;   NEW CHKSUM  COMPARE OLD CHECKSUM IN F14
CPCHKSUM
        CLRW
        BCF     CHECKBIT,4
        MOVF    .12,W
        ADDWF   .13,W
        XORWF   .13,W
        BTFSC   STATUS,Z
        RETLW   0
        BSF     CHECKBIT,4
        RETLW   0
; DECODE INSTRUCTION TO ORDER RELAY
COMMAND
        BCF     CHECKBIT,1
        MOVLW   0FAH
        XORWF   .12,W     ; ID MACHINE
        BTFSS   STATUS,Z
        RETLW   0
        MOVLW   032H
        XORWF   COMBYT,W
        BTFSS   STATUS,Z
        RETLW   0
        BSF     CHECKBIT,1    ;SET BIT1 F26
        RETLW   0
; DETECT ERROR SIGNAL FROM MASTER
ERSUM
        BCF     CHECKBIT,2
        MOVLW   0EEH
        XORWF   COMBYT,W
        BTFSS   STATUS,Z
        RETLW   0
        BSF     CHECKBIT,2
        RETLW   0
;   SUBROUTINE DELAY  TIME
DELAY
        MOVLW   255
        MOVWF   27
  LOOP
        DECFSZ  27,F
        GOTO    LOOP2
        RETLW   0         ; RETURN TO PC
  LOOP2
```

```
        MOVLW   .255
        MOVWF   .28
D2      DECFSZ  .28,f
        GOTO    LOOP3
        GOTO    LOOP
LOOP3   MOVLW   .80
        MOVWF   .29
D3      DECFSZ  .29,f
        GOTO    D3
        GOTO    D2
; CHECK COMMAND BYTE = 050H , 051H  IF YES  CALL DELAY
DCOM    MOVLW   050H
        XORWF   COMBYT,W
        BTFSS   STATUS,Z
        GOTO    NEXT
        CALL    DELAY
        RETLW   0
NEXT    MOVLW   051H
        XORWF   COMBYT,W
        BTFSS   STATUS,Z
        RETLW   0
        CALL    DELAY
        RETLW   0
;
MAIN    CALL    SETUP
        MOVLW   0FFH        ;SET PORTC TO I/P PORT
        TRIS    PORTC
        MOVLW   .11         ; COUNT BYTE = 11
        MOVWF   CBYT
        MOVLW   .12         ; F12 FIRST BYTE
        MOVWF   4
        CALL    TENRCV      ;RECIEVE FROM 8749
        CALL    CHKSUM
WAIT    CALL    RECIEVE     ; RECIEVE POOL SIGNAL
CHKIDM  CALL    CHKID
        BTFSS   CHECKBIT,0
        GOTO    WAIT
AGAIN   CALL    CHKENS      ;CHECK ENABLE DATA TO SEND
        CALL    SNDSLAVE    ;SEND DATA TO SLAVE
WAIT2   CALL    RECIEVE     ;RCV FOR CHECK SUM
CHKIDM2 CALL    CHKID
        BTFSS   CHECKBIT,0
        GOTO    WAIT2
        CALL    ERSUM       ;CHECK IF CHKSUM ERROR
        BTFSC   CHECKBIT,2
        GOTO    AGAIN
ACCEPT  CALL    RECIEVE     ; RECIEVE DATA FROM MASTER
        CALL    COMMAND
        BTFSC   CHECKBIT,1
        GOTO    NEWEST
CHKIDM3 CALL    CHKID
        BTFSS   CHECKBIT,0
        GOTO    ACCEPT
NEWEST  CALL    CPCHKSUM
        BTFSC   CHECKBIT,4  ; CHK SUM ERROR(SET) OR NOT
        GOTO    SENDER      ; PREPARE SEND BACK TO MASTER
        CALL    DCOM
        CALL    COM00
        BTFSS   CHECKBIT,3
        GOTO    MAIN
        CALL    TENRS       ; CHK EN DATA
        CALL    SEND        ; SEND DATA TO 8749
        GOTO    MAIN
SENDER  MOVLW   0EEH
        MOVWF   .21
        CALL    CHKSUM
WAIT3   CALL    RECIEVE
CHKIDM4 CALL    CHKID
        BTFSS   CHECKBIT,0
        GOTO    WAIT3
        CALL    CHKENS
        CALL    SNDSLAVE
        GOTO    ACCEPT
        ORG     01FFH
        GOTO    MAIN
        END
;*****************************************
;* This subroutine for set initial of system
;*     set port A1,A2 to i/p  A3 to o/p
;*          set port B to o/p port
;*          set rtcc prescaler  = 64
;*****************************************
initial     clrw
        tris    6
        clrf    6
        clrf    5
        bsf     5,3
        movlw   6
        option      ;set prescaler = 128
        movlw   6
        tris    5       ; set port A1,A2 to i/p port A3 to o/p
        retlw   0
;   SUBROUTINE FOR CHECK FIRST BYTE (*) OF DATA
;    AND  SEARCH FOR WIDTH AND NARROW PULSE
beginsc
        clrf    1
        btfss   5,1
        goto    beginsc
sc10
        btfsc   5,1
        goto    over1
        goto    save1
over1
        movlw   0FA
        subwf   1,w
        btfss   3,0
        goto    sc10
        clrf    29
        bsf     29,0
        retlw   0
save1
        movf    1,w
        movwf   9
        movlw   9
        xorwf   30,w
        btfsc   3,2
        goto    first_bitby     ; = 1st bit
        goto    sevenb          ; != 1st bit
comp
        call    compare
        retlw   0
sevenb
```

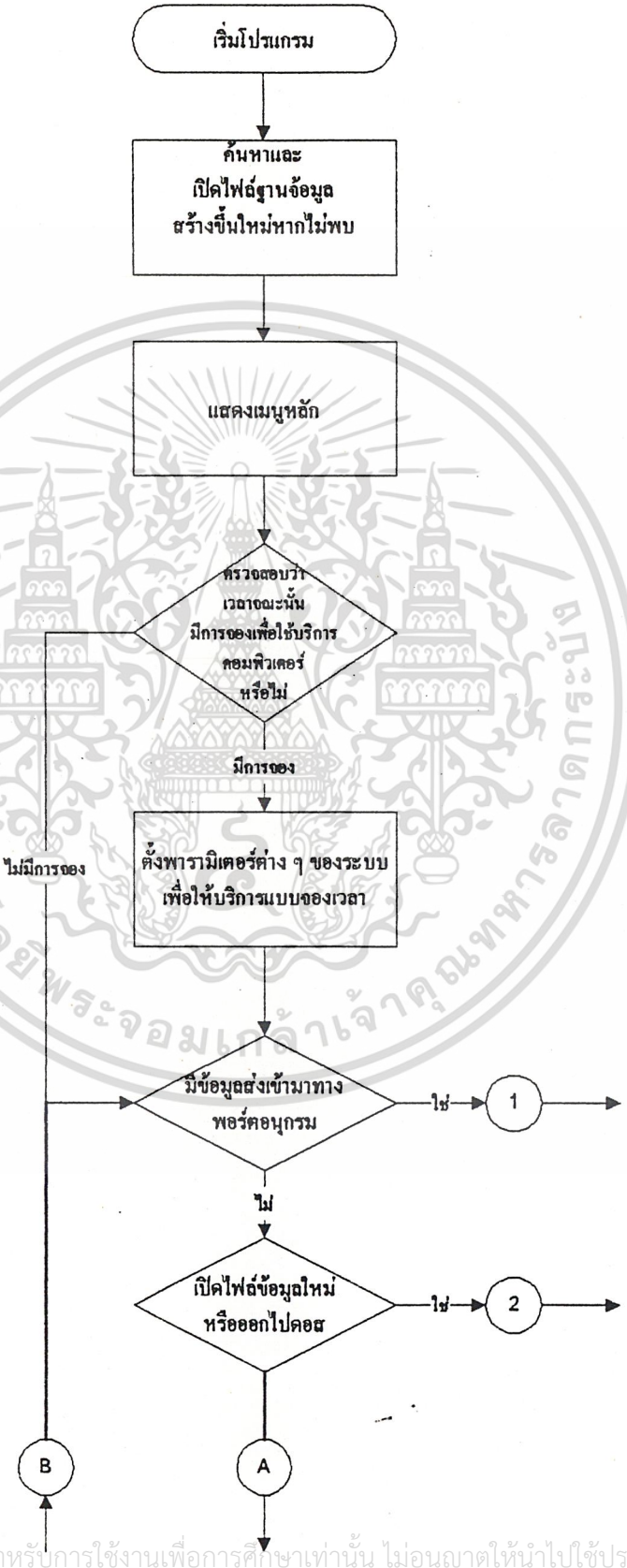
```
        movlw   3           ;
        xorwf   .30,w       ; [ chk = 7th bit or not]
        btfss   3,2         ;
        goto    comp
        movlw   .234        ; EA=234=f10
        xorwf   4,w
        btfss   3,2         ; chk = f10 (first byte) or not
        goto    comp        ; not ==> compare
        goto    newref
first_biby
        movlw   .234
        xorwf   4,w
        btfss   3,2
        retlw   0
        movf    9,w
        movwf   8
        movwf   .20
        retlw   0

newref
        movf    .20,w       ; process new reference save at f8
        subwf   9,w         ; f9-f20 = new width - old narrow
        movwf   7           ; store result at f7
        bcf     3,0         ; clear carry bit
        rrf     7,f         ; f7/2
        movf    7,w
        subwf   9,w         ; f9 - (f7/2)
        movwf   8           ; store new result at f8(new ref y)
        rlf     0,f
        bsf     0,0
        retlw   0
check01
        clrf    1
again
        btfss   5,1
        goto    over2
        goto    save2
 over2
        movlw   0FA
        subwf   1,w
        btfss   3,0
        goto    again
        clrf    .29
        bsf     .29,0
        retlw   0
 save2
        movf    1,w
        movwf   9
        movlw   .234        ;|
        xorwf   4,w         ;|[chk = first byte or not]
        btfss   3,2         ;|
        goto    comp2       ; if not compare bit
        movlw   8           ; yes chk = 2nd bit or not
        xorwf   .30,w
        btfsc   3,2
        goto    second_biby
comp2
        call    compare
        retlw   0
second_biby
        movf    8,w
        subwf   9,w         ; f9-f8
        movwf   7           ; store at f7
        bcf     3,0         ; clear carry bit
        rrf     7,f         ; f7/2
        movf    7,w
        subwf   9,w         ; f9-f7
        movwf   8           ;tore at f8
        clrf    0,f
        bsf     0,0
        retlw   0
compare
        clrw
        clrf    0,f
        movf    9,w
        xorwf   8,w         ; excusive or f8 with f9
        btfss   3,2
        goto    differ      ; f8 != f9 goto f8-f9
        bsf     0,0
        retlw   0
differ
        movf    9,w
        subwf   8,w         ; f8-f9
        btfsc   3,0
        goto    setn        ; f8>f9
        goto    setw        ; f8<f9
setn    bcf     0,0
        retlw   0
setw    bsf     0,0
        retlw   0
; subroutine for delay strobe ;
delay   bsf     5,3
        bcf     5,3
        movlw   3
        movwf   .20
lp      decfsz  .20,f       ; delay for bit 1
        goto    lp
        bsf     5,3
delay0
        movlw   5
        movwf   .21
lp2     decfsz  .21,f       ; delay for bit 0
        goto    lp2
        retlw   0
;    SUBROUTINE DELAY TIME
dlay    movlw   .255
        movwf   .26
loop    decfsz  .26,f
        goto    loop2
        retlw   0           ; return to pc
loop2   movlw   .255
        movwf   .27
d2      decfsz  .27,f
        goto    loop3
        goto    loop
loop3   movlw   5
        movwf   .28
d3      decfsz  .28,f
        goto    d3
        goto    d2
;*   SUBROUTINE FOR DECODE DATA BETWEEN F10 TO F19          *
;*          AND SAVE NEW DATA AT F10 TO F19                 *
```

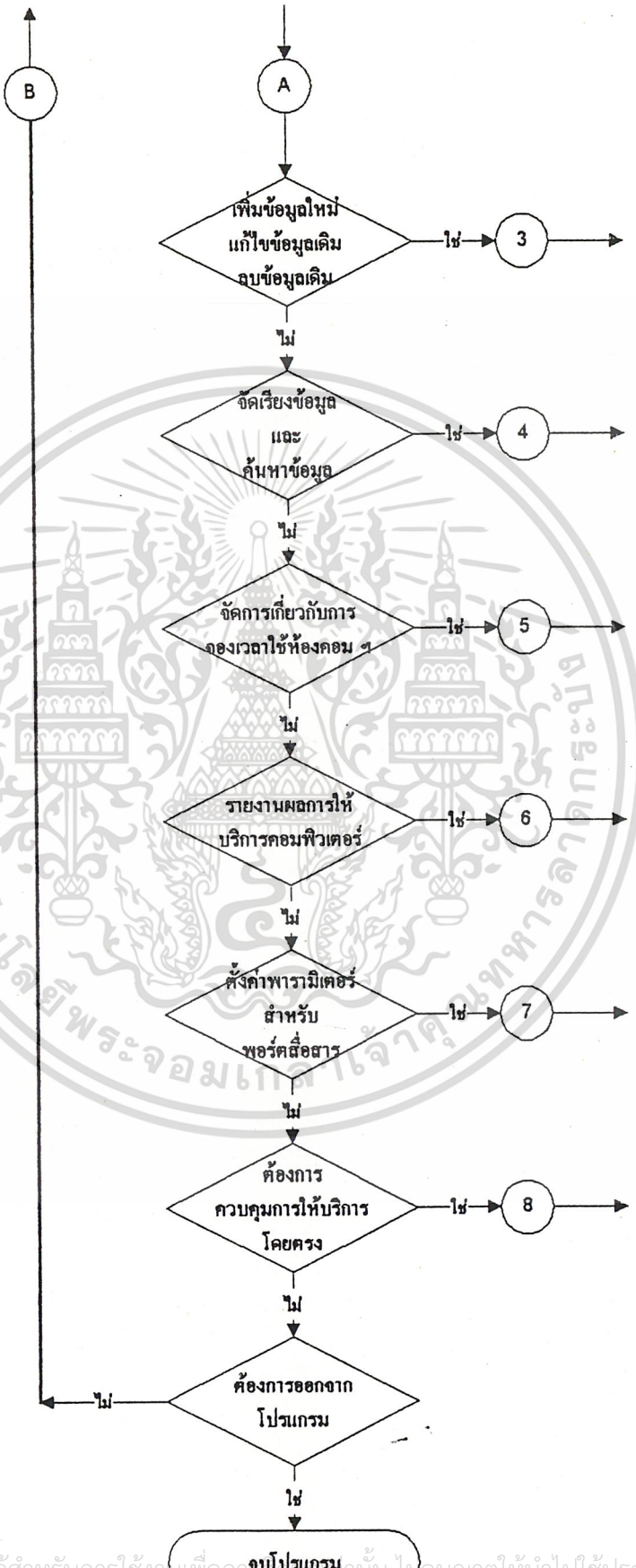
```
;*          F7 = 8 (count byte to decode)                    *
decode
        clrw
        clrf    4
        clrf    7
        movlw   9
        movwf   7
        movlw   .10
        movwf   4
;       * CHECK START CODE(*)  AND STOP CODE(*) *
        movlw   94
        xorwf   .10,w
        btfss   3,2
        goto    error
        movlw   .255            ; STORE FF AT F10
        movwf   .10
        movlw   94
        xorwf   .19,w
        btfss   3,2
        goto    error
        movlw   .240            ; STORE F0 AT F19
        movwf   .19
;       * DECODE DATA FROM PATTERN AND SAVE CODE(DECIMAL) *
;       *   AT F10 TO F19   F7 FOR COUNT BYTE(8 BYTE)   *
pattern
        decfsz  7,f
        goto    bgdecode
        goto    bgout
bgdecode
        incf    4,f             ; process  next byte
;COMPARE  PATTERN  OF  0 (34h)
        movlw   34
        xorwf   0,w
        btfsc   3,2
        goto    equal0
        movlw   34
        subwf   0,w             ; f? - 34
        btfss   3,0
        goto    much34          ; 34 > f?
;COMPARE PATTERN  OF 2 (61h)
        movlw   61
        xorwf   0,w
        btfsc   3,2
        goto    equal2
        movlw   61
        subwf   0,w             ; f? - 61
        btfss   3,0
        goto    much61          ; 61 > f?
;COMPARE PATTERN OF  9 (64h)
        movlw   64
        xorwf   0,w             ; f? - 64
        btfsc   3,2
        goto    equal9          ; f? = 9
        movlw   6               ; f? = 6
        movwf   0
        incf    4,f             ; process  next byte
        goto    pattern
;COMPARE PATTERN OF 7 (25h)
much34
        movlw   25
        xorwf   0,w
        btfsc   3,2
        goto    equal7
        movlw   25
        subwf   0,w             ; f? - 25
        btfss   3,0
        goto    much25          ; f? < 25
;COMPARE PATTERN OF 5  (30h)
        movlw   30
        xorwf   0,w
        btfsc   3,2
        goto    equal5
        movlw   4               ; f? = 4
        movwf   0
        goto    pattern
equal0
        clrw                    ; PATTERN F? = 0
        movwf   0               ; f?= 0
        goto    pattern
much61
        movlw   3               ; f? =3
        movwf   0
        goto    pattern
equal2
        movlw   2               ; f? = 2
        movwf   0
        goto    pattern
equal3
        movlw   3               ;f? = 3
        movwf   0
        goto    pattern

equal9
        movlw   9               ; f? =9
        movwf   0
        goto    pattern
much25
        btfsc   3,2
        goto    equal7
        movlw   24
        subwf   0,w             ;f? - 24
        btfss   3,2
        goto    equal1          ; f? = 1
        movlw   8               ; f? = 8
        movwf   0
        goto    pattern
equal7
        movlw   7               ; f? = 7
        movwf   0
        goto    pattern
equal1
        movlw   1               ; f? = 1
        movwf   0
        goto    pattern
equal5
        movlw   5               ; f? = 5
        movwf   0
        goto    pattern
;*****************************************************
;       OUTPUT DATA TO 8749  THROUGH  PORT B            *
;*****************************************************
output
```

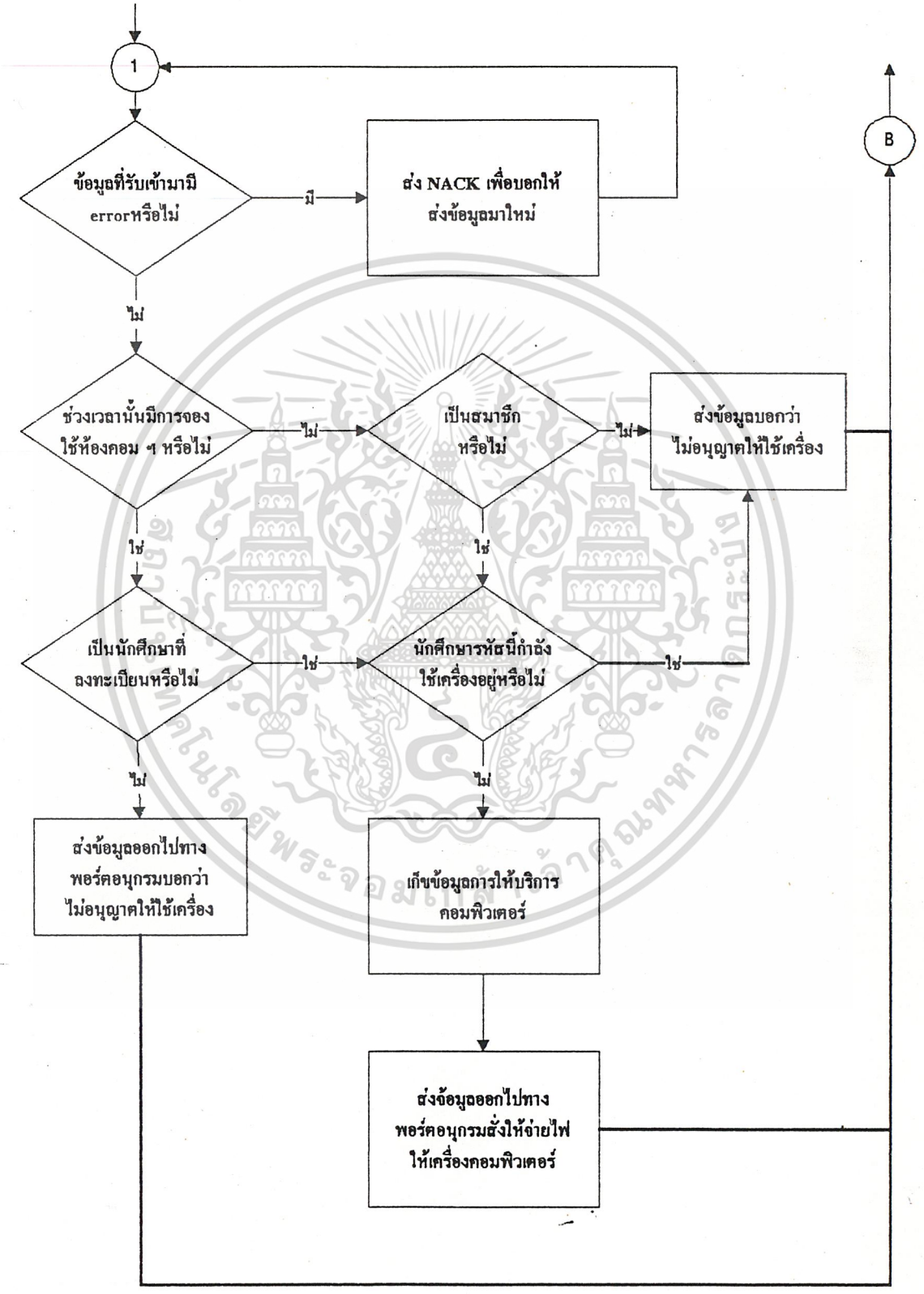
```
        clrw
        tris   6
        clrf   6
        bsf    5,3
        movlw  6
        tris   5
        movlw  .10
        movwf  4            ; (F10) 1st byte o/p data
        movwf  9            ; f9 store count byte(10 byte)
; main
loop1 btfss  5,2            ;chk rdy
        goto   loop1
        goto   lddata
lddata movf  0,w
        movwf  6            ; begin send data
        call   delay
        incf   4,f          ; next byte
        decfsz 9,f
        goto   loop1
        goto   finish
;     *** SUBROUTINE SENDERROR *******
;     **** WILL SEND FF AAAA AAAA F0  TO PORT B ****
error movlw  8              ; count byte= 8
        movwf  .28
        movlw  0A           ; f29 contain error value(A)
        movwf  .29
        movlw  0FF
ler1  btfss  5,2
        goto   ler1
        movwf  6            ; send FF
        call   delay
        movf   .29,w
repeat btfss  5,2
        goto   repeat
        movwf  6
        call   delay
        decfsz .28,f
        goto   repeat       ; send 5  8times
        movlw  0F0
ler2  btfss  5,2
        goto   ler2
        movwf  6            ; send F0
        call   delay
        call   dlay
        goto   start
;MAIN PROGRAM
start  clrw
        call   dlay
strt2  clrw
        movlw  .10
        movwf  .31          ; count byte = 10
        clrf   .29
        movlw  .10          ; first byte of data (*)
        movwf  4
        call   initial
cbit   movlw  9
        movwf  .30          ; count bit
begscan
     call   beginsc
        btfsc  29,0
        goto   strt2
        decfsz .30,f
        goto   scan01
        goto   nbdata
scan01
        call   check01
        btfsc  .29,0
        goto   strt2
        decfsz .30,f
        goto   begscan
        goto   nbdata
nbdata
        incf   4,f
        decfsz .31,f
        goto   cbit
        goto   decode
bgout
        goto   output
finish
        bsf    5,3
        call   dlay
wait
        btfss  5,2
        goto   wait
        goto   start
        org    01FF
        goto   start
        END
```

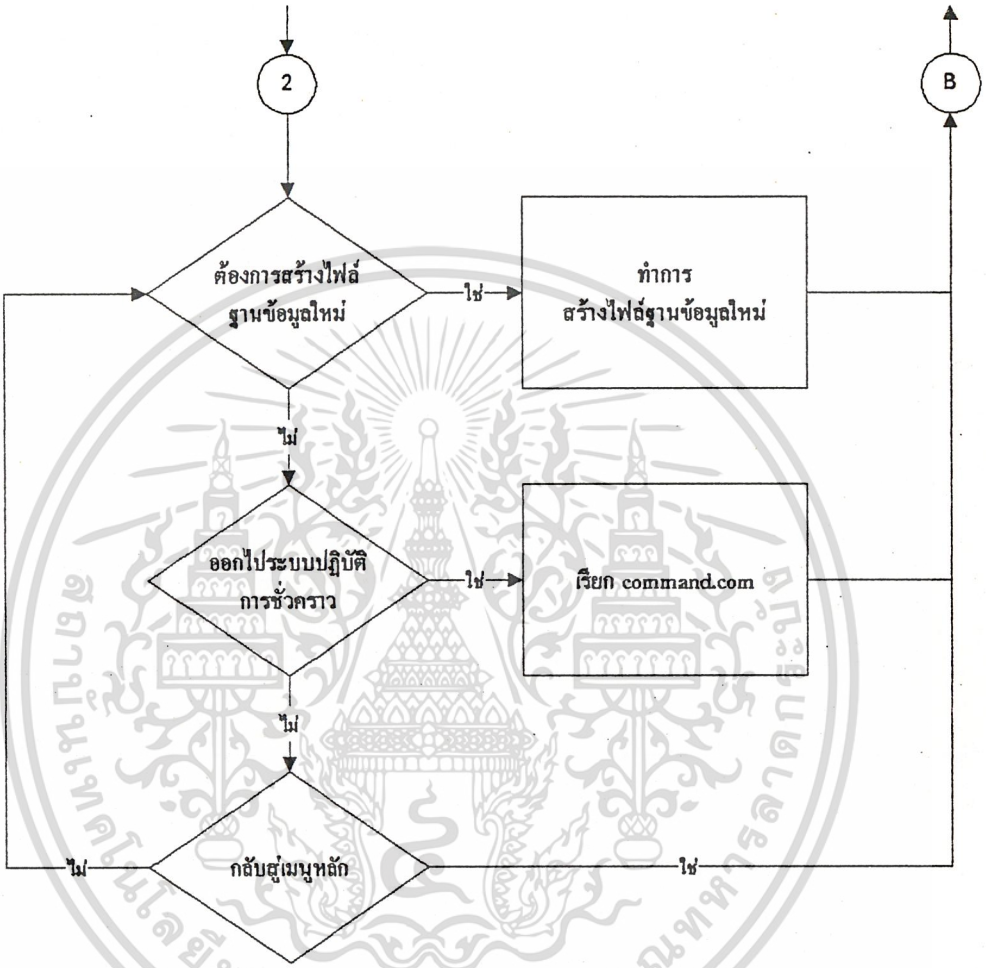
```
;Program for master board develop by sung
;PRODUCE IN 7/3/38
    CPU   "Z80.TBL"
    HOF   "INT8"
    ORG   0000H
SYSSTK: EQU   3FFFH
APDATA: EQU   20H          ;CHANNEL A PIO DATA
BPDATA: EQU   21H          ;CHANNEL B PIO DATA
APCONT: EQU   22H          ;CHANNEL A PIO CONTROL
BPCONT: EQU   23H          ;CHANNEL B PIO CONTROL
ASDATA: EQU   24H          ;CHANNEL A SIO DATA
BSDATA: EQU   25H          ;CHANNEL B SIO DATA
ASCONT: EQU   26H          ;CHANNEL A SIO CONTROL
BSCONT: EQU   27H          ;CHANNEL B SIO CONTROL
; ********** VARIABLE SET **********
POWER:  XOR   A            ;POWER UP DELAY
POWER1: DEC   A
    NOP
    JR    NZ,POWER1
    JP    RESET
; ********** SYSTEM RESET **********
RESET:  LD    HL,4000H     ;DELAY
RESET0: DEC   HL
    LD    A,H
    OR    L
    JR    NZ,RESET0
    LD    SP,SYSSTK    ;SYSTEM STACK
; ******** FOR INITIAL SIO ********
SIOINI: DI
    LD    HL,TABLE1
SINIT1: LD    A,(HL)
    CP    0FFH
    JR    Z,SINIT2
    OUT   (BSCONT),A
    INC   HL
    JR    SINIT1
;TABLE FOR INITIAL SIO INTERRUPT
TABLE1: DFB   18H          ;WR0 CHANNEL RESET
    DFB   02H          ;POINT TO WR2
    DFB   00H          ;WR2 INT VECT = 00H
    DFB   03H          ;POINT TO WR3
    DFB   0C1H         ;8 BIT REC ,RX ENABLE
    DFB   04H          ;POINT TO WR4
    DFB   4FH          ;WR4,X16 CLK ,2 STOP,EVEN PARITY
    DFB   05H          ;POINT TO WR5
    DFB   68H          ;8 BIT TRAN,TX ENABLE
    DFB   01H          ;POINT TO WR1
    DFB   14H          ;E NABLE TX  INTERRUPT
    DFB   0FFH
SINIT2: NOP
;INITIAL PIO
;SET PORT A (PIO) TO BE OUT PUT PORT (MODE 0) WITH INTERRUPT
        LD        A,0FH        ;PORT A IS OUTPUT
    OUT   (APCONT),A   ;OUTPUT CONTROLWORD
        LD        A,87H        ;ISABLE INTERRUPT ON PORT A OF
PIO
    OUT   (APCONT),A   ;OUTPUT CONTROLWORD
    LD    A,02H                ;INTERRUPT VECTOR
    OUT   (APCONT),A
;SET PORT B (PIO) TO BE IN PUT PORT (MODE 1 ) WITH INTERRUPT
    LD    A,4FH        ;PORT B IS INPUT
    OUT   (BPCONT),A   ;OUTPUT CONTROLWORD
    LD    A,87H        ;ENABLE INTERRUPT ON PORT B OF PIO
        OUT       (BPCONT),A   ;OUTPUT CONTROLWORD
    LD    A,00H        ;INTERRUPT VECTOR
    OUT   (BPCONT),A
        IM    2
    LD    A,11H
    LD    IA
    EI
    XOR   A
    OUT   (APDATA),A   ;FOR FIRST RESET PIO DATA
    LD    A,64
    LD    (TXCH),A
;MAIN PROGRAM
MAIN:   IN    A,(BSCONT)
    BIT   2,A
    JR    Z,MAIN
    LD    A,(TXCH)
    OUT   (BSDATA),A
        JR    MAIN
;TABLE FOR INTERRUPT VECTOR
    ORG   1100H
    DWL   RXPINT           ;RECEIVED DATA FROM CHANNEL B PIO
    DWL   TXPINT           ;TRANSMITED DATA TO CHANNEL A PIO
    DWL   RXSINT       ;RECEIVED CHARACTER CHANNEL B SIO
    DWL   RXSINT1      ;SPECTIAL RECEIVE CONDITION CHANNEL B SIO
    ORG   1200H
RXPINT: PUSH           AF
        IN             A,(BPDATA)
    LD    (TXCH),A
    POP   AF
    RETI
TXPINT: PUSH           AF
        LD             A,(RXCH)
        OUT            (APDATA),A
    POP   AF
    RETI
;RECEIVED DATA FROM RS232
RXSINT: PUSH   AF
    IN    A,(BSDATA)
    LD    (RXCH),A
    POP   AF
    EI
    RETI
;PARITY ERROR POUND
RXSINT1:PUSH   AF
    LD    A,30H
    OUT   (BSCONT),A   ;WR0, ERROR RESET COMMAND
    POP   AF
    EI
    RETI
```

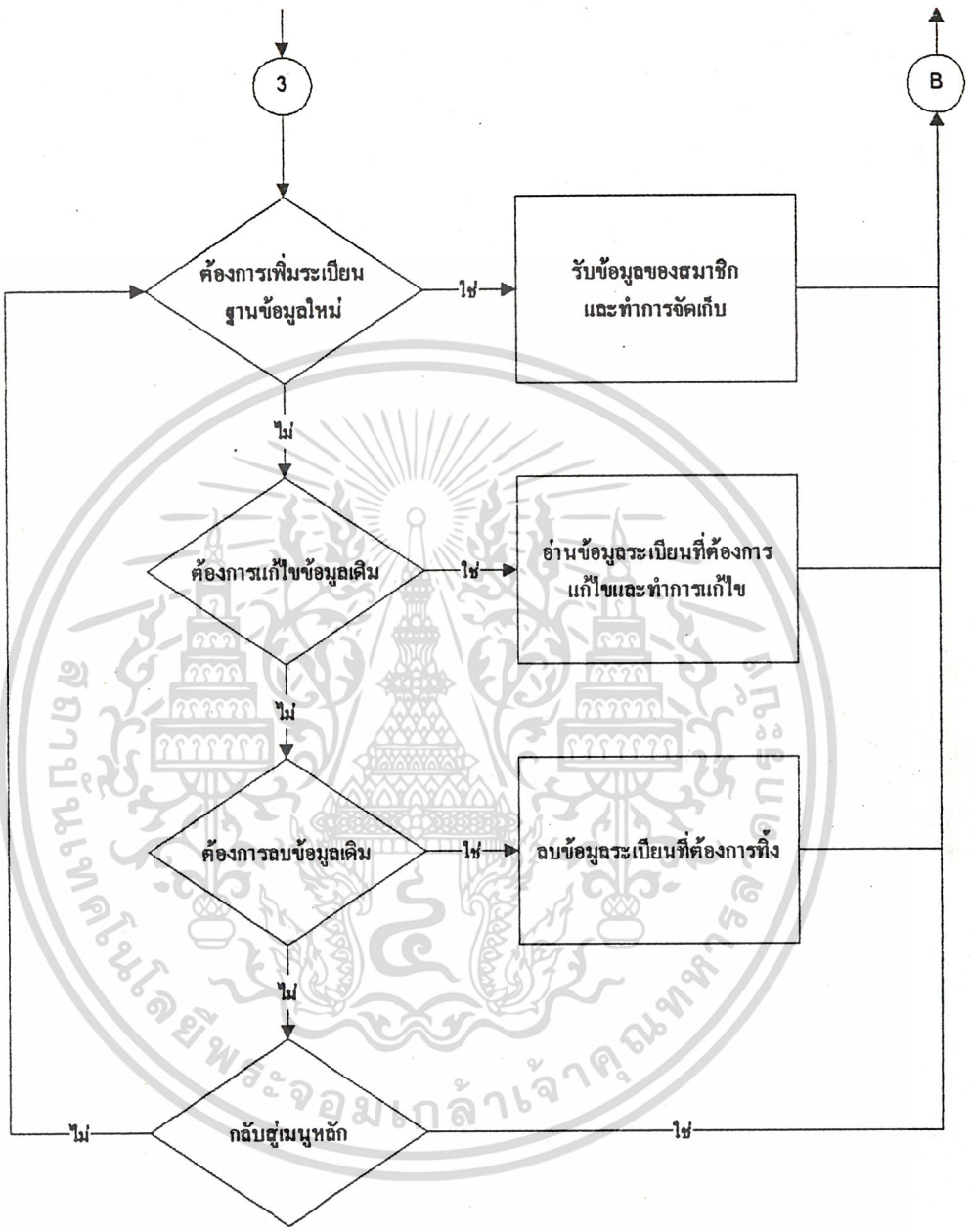
```
        ORG     2400H
TXCH:   DFB     00H
RXCH:   DFB     00H
        END
```

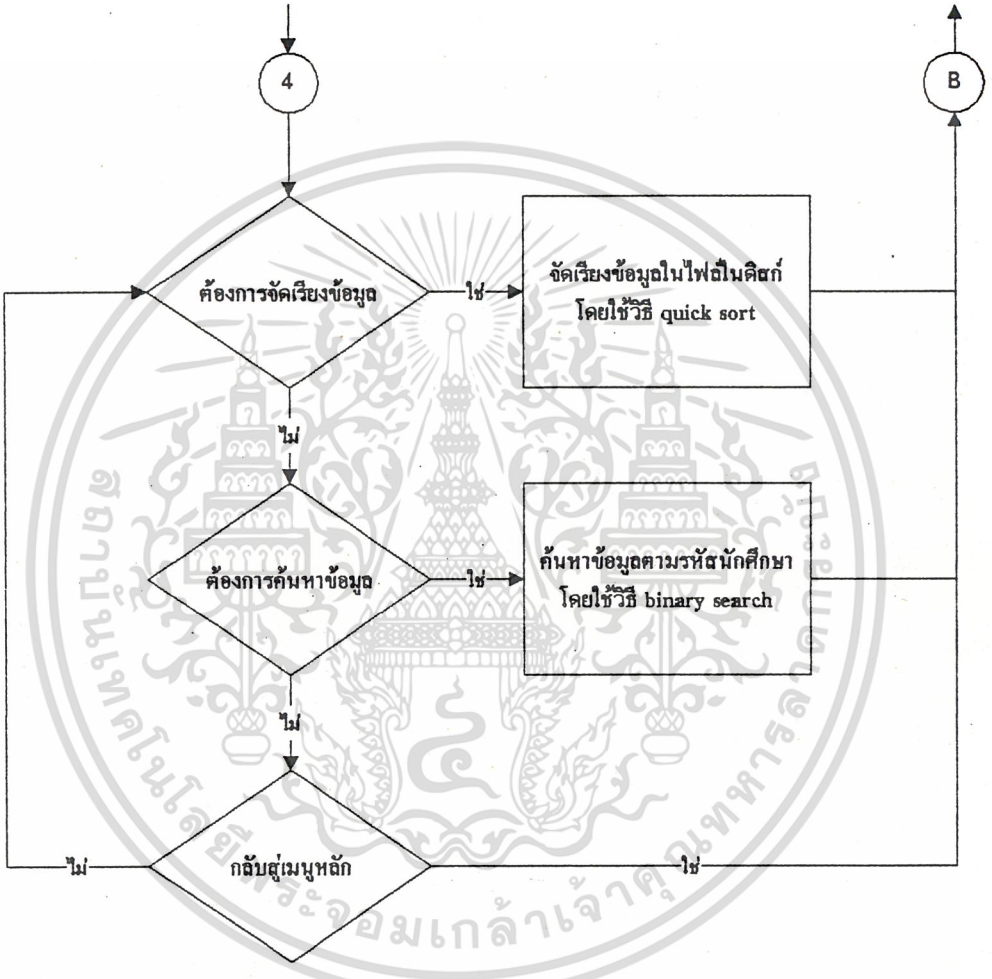


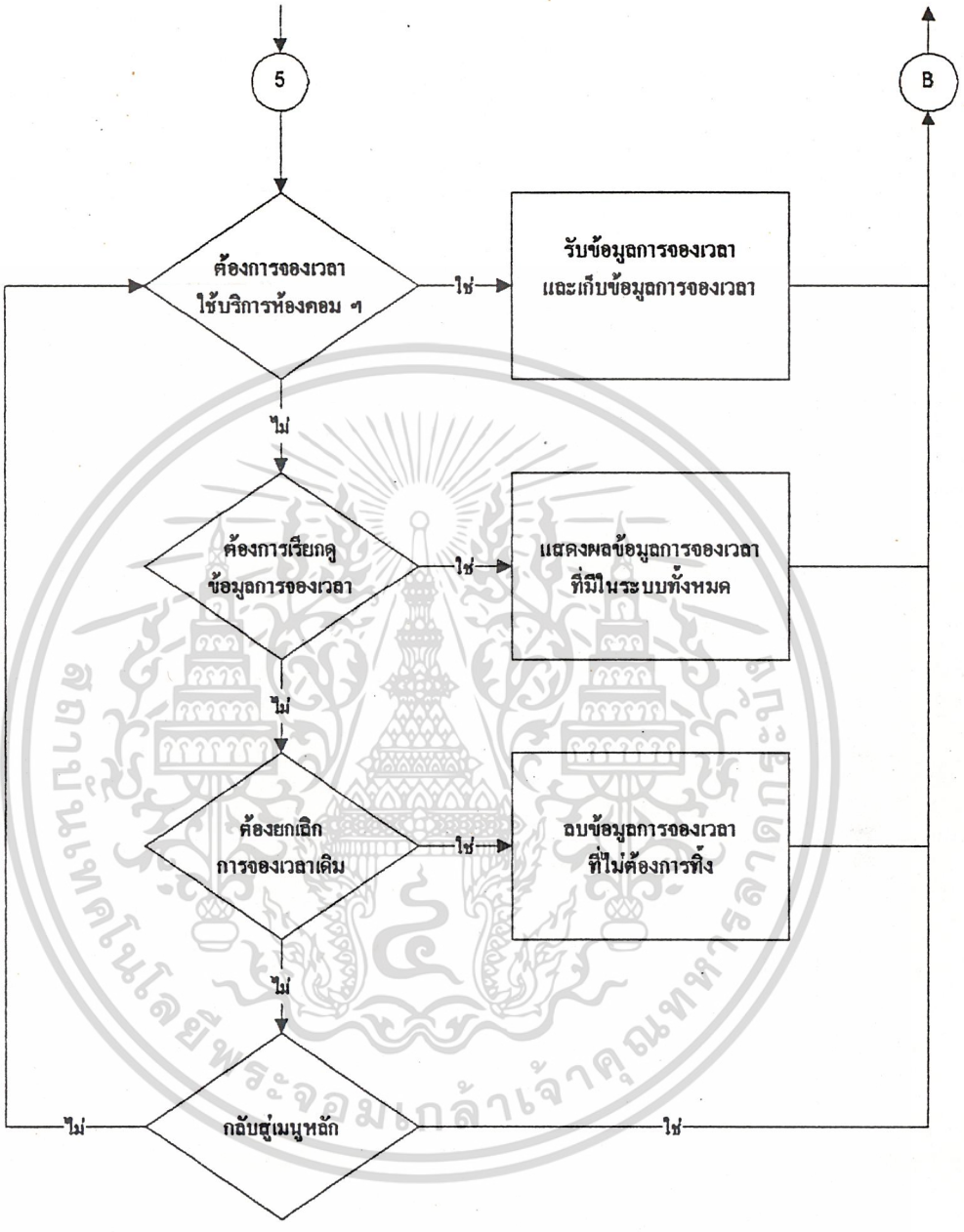


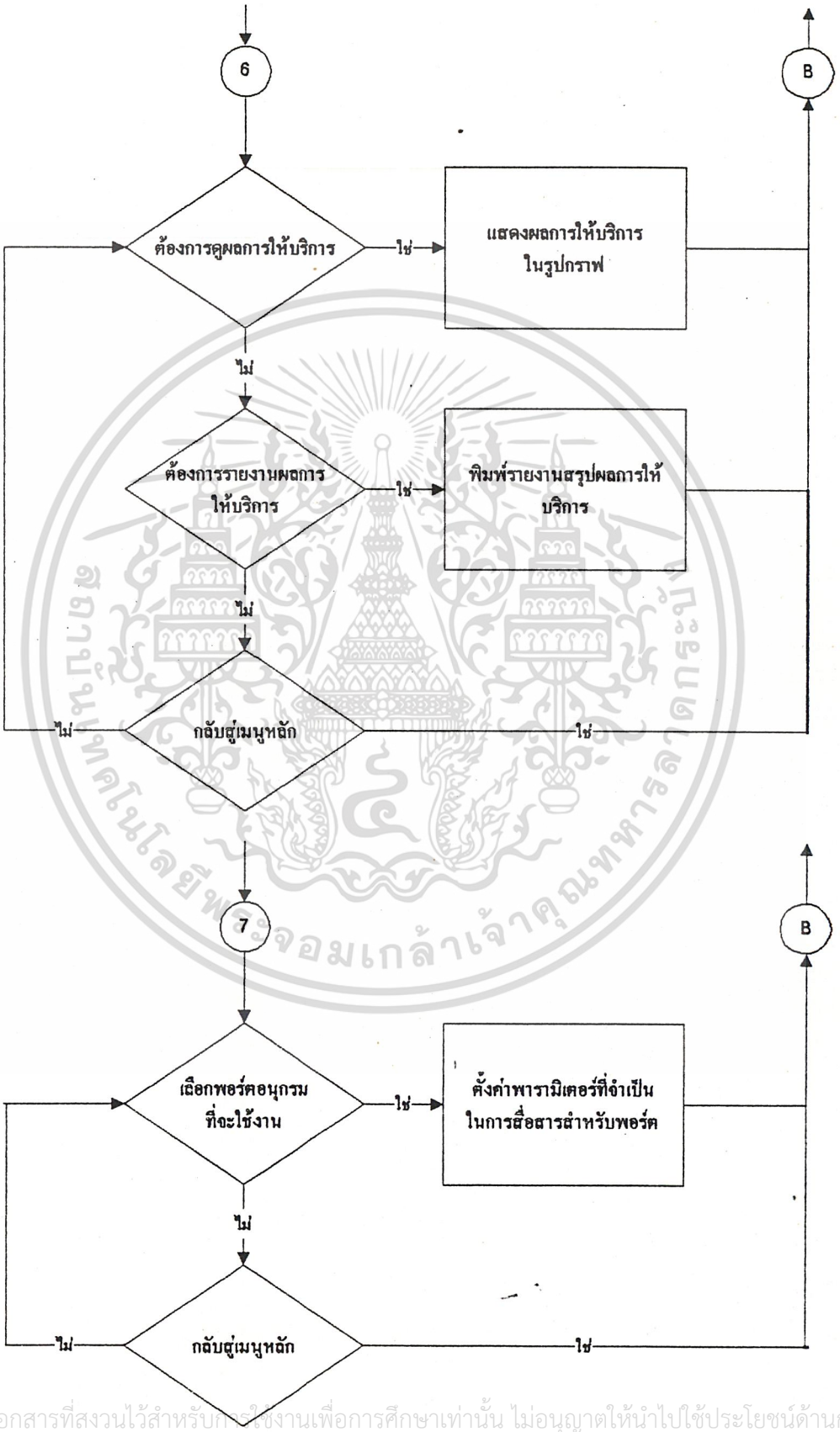


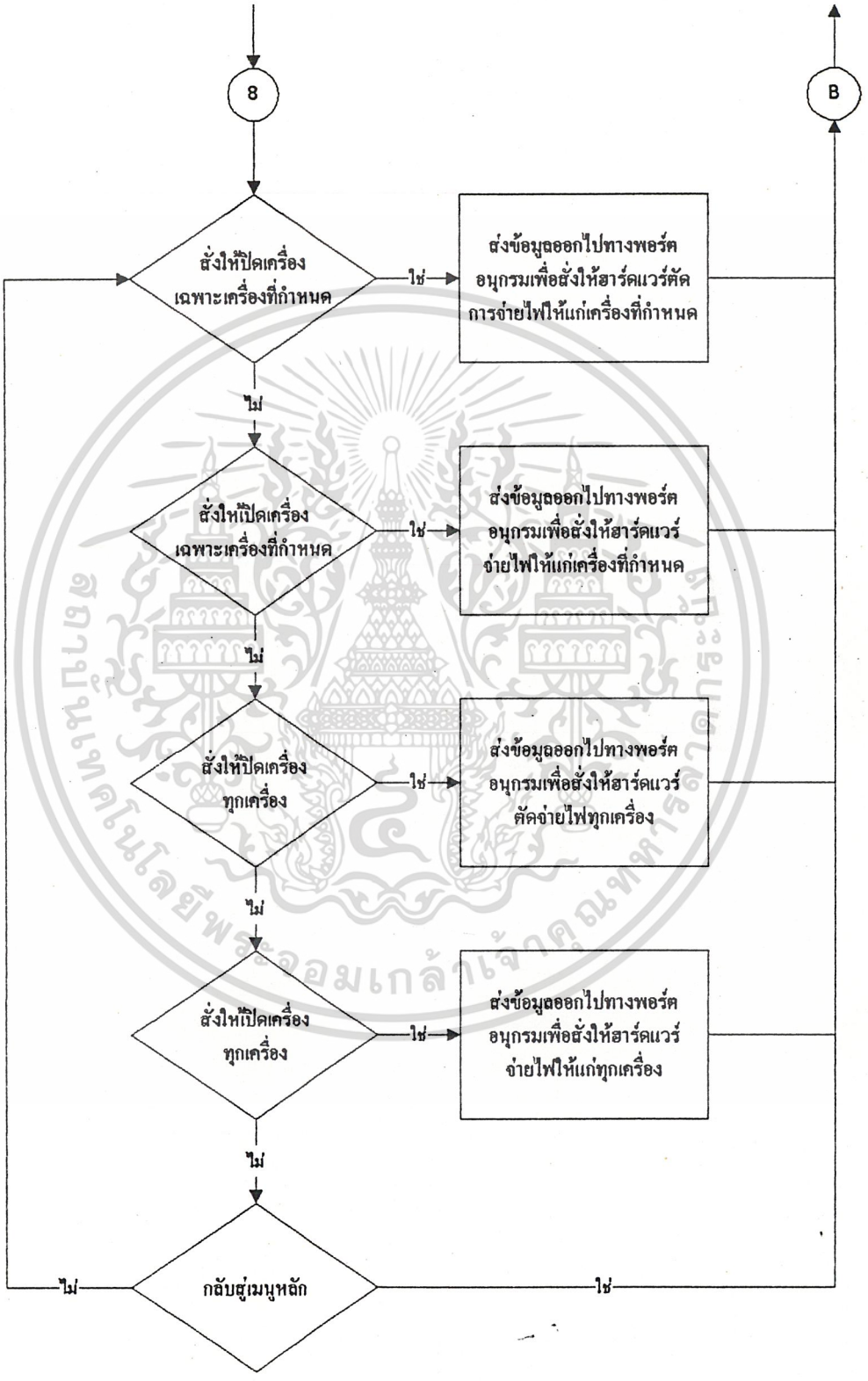


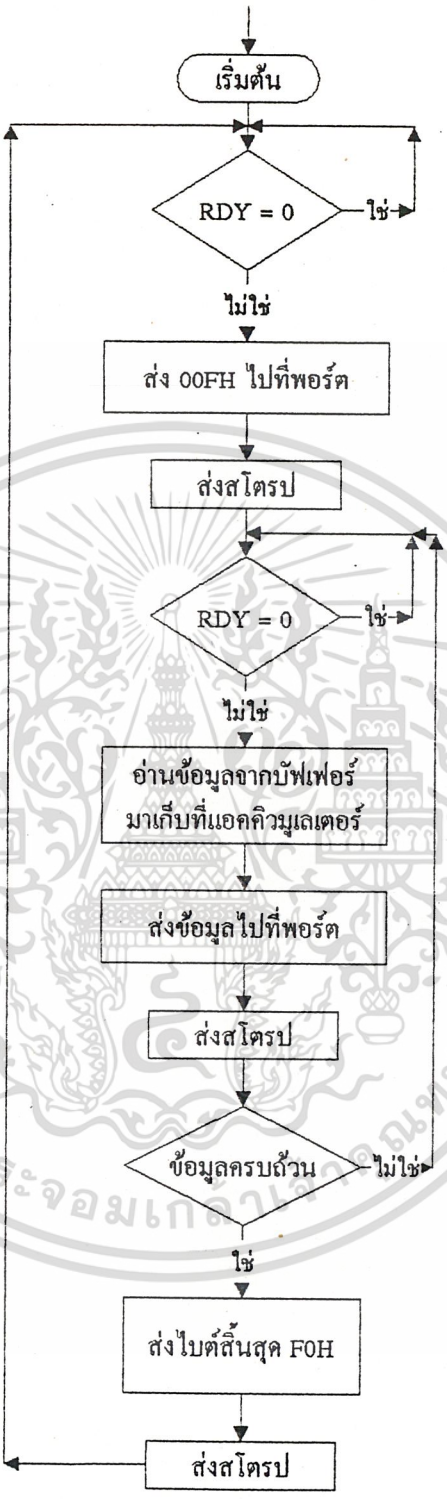


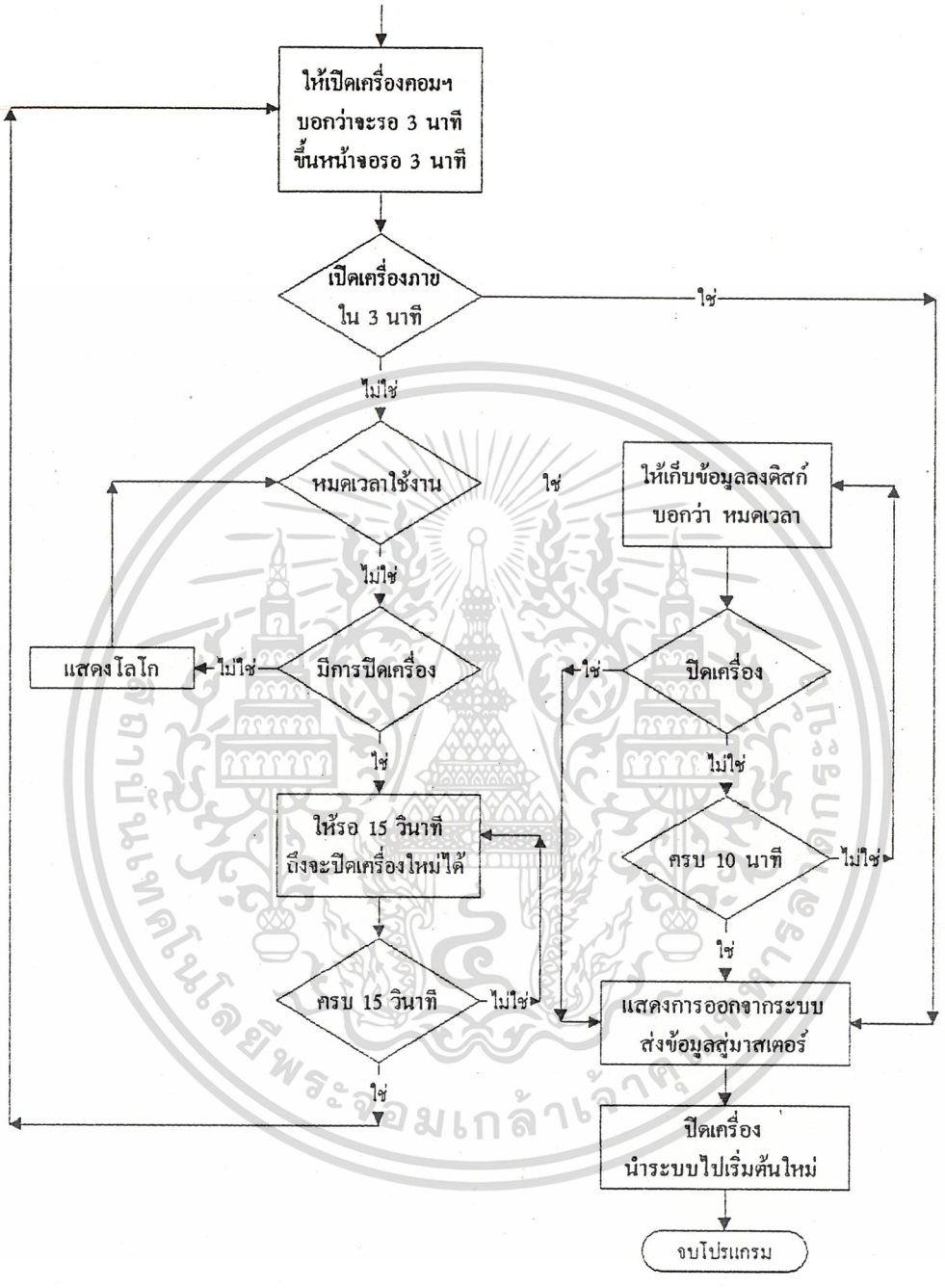


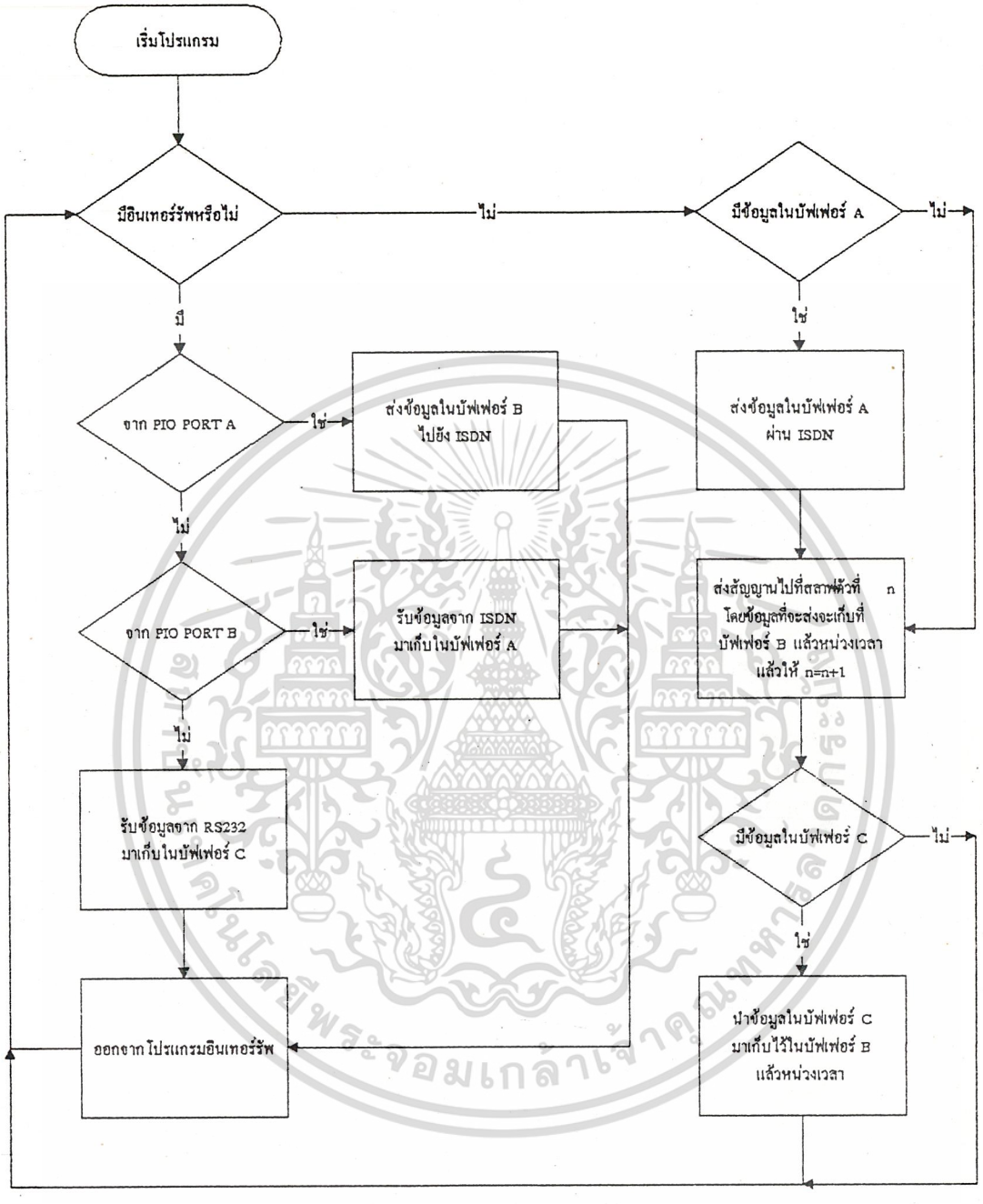


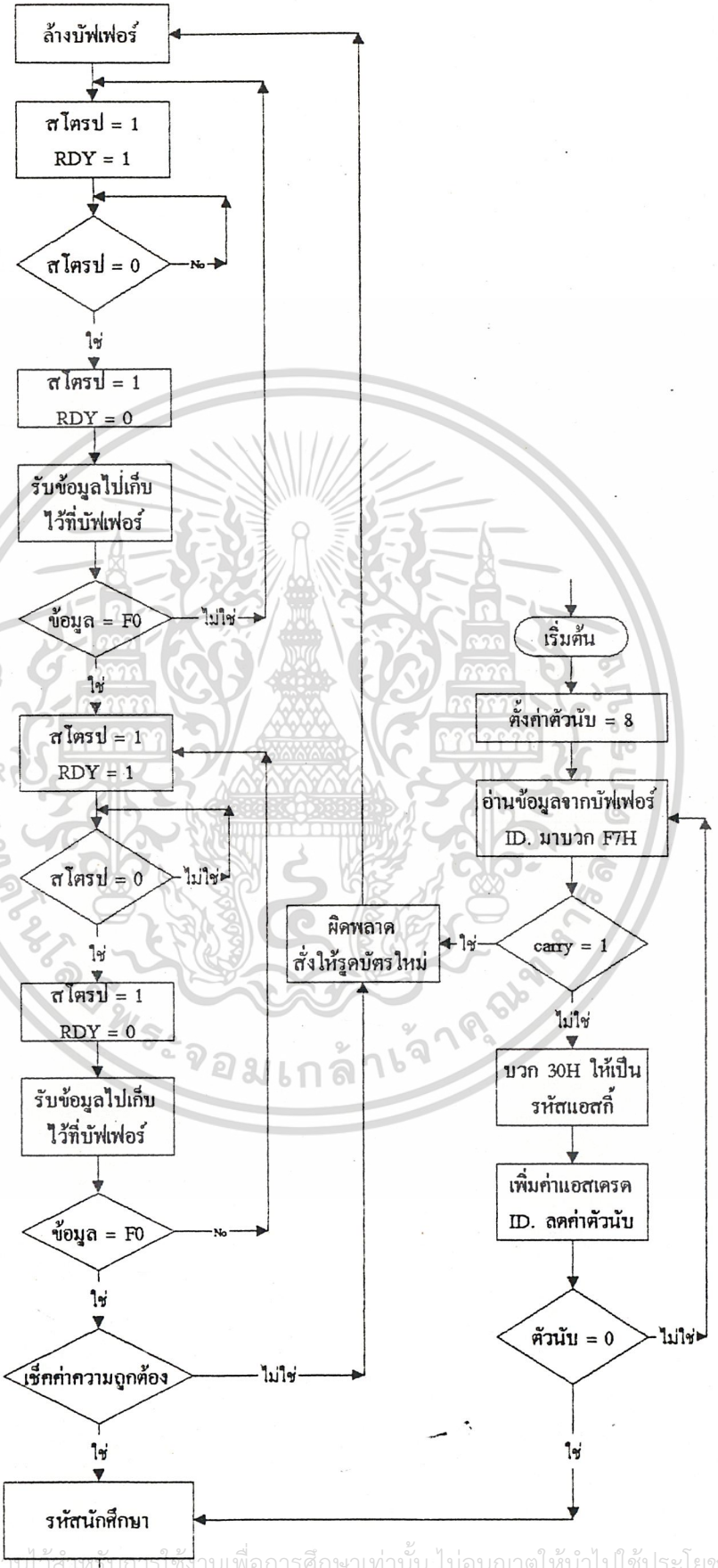












PRODUCT SPECIFICATION

Z8440/1/2/4, Z84C40/1/2/3/4

SERIAL INPUT/OUTPUT CONTROLLER

FEATURES

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rate in the x1 clock mode of 0 to 2.0M bits/second with a 10 MHz clock.
- NMOS version for cost sensitive performance solutions, CMOS version for the designs requiring low power consumption
- NMOS Z0844x04 - 4 MHz Z0844x06 - 6.17 MHz (Where x is the designator for the bonding option; 0, 1, 2 or 4)
- CMOS Z84C4x06 - DC to 6.7 MHz, Z84C4x08 - DC to 8 MHz, Z84C4x10 - DC to 10 MHz (Where 'x is the designator for the bonding option; 0, 1, 2 , 3or 4)
- 6 MHz version supports 6.144 MHz CPU clock operation.
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7, or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7, or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection, and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

GENERAL DESCRIPTION

The Z80 SIO (here in after referred to as the Z80 SIO or, SIO), Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs, and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast, or slow, peripheral devices. While designed primarily as a member of the Z80 family, its versatility makes it well suited to many other CPUs.

The Z80 SIO uses a single +5V power supply and the standard Z80 family single-phase clock. The SIO/0, SIO/1, and SIO/2 are packaged in a 40-pin DIP, the SIO/4 is packaged in a 44-pin PCC and the SIO/3 is packaged in a 44-pin QFP. Note that SIO/3 is only available in CMOS and in QFP package.

PIN DESCRIPTION

Figures 1 through 6 illustrate the three 40-pin configurations (bonding options) available in the Z80C SIO (hereafter referred to as SIO or Z80 SIO). The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together:

- Z80 SIO/2 lacks $\overline{SYNCB}$
- Z80 SIO/1 lacks $\overline{DTRB}$
- Z80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The 44-pin package, the Z80 SIO/4 for PLCC package; and Z80 SIO/3 for QFP, has all options (Figure 7a and 7b).

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data

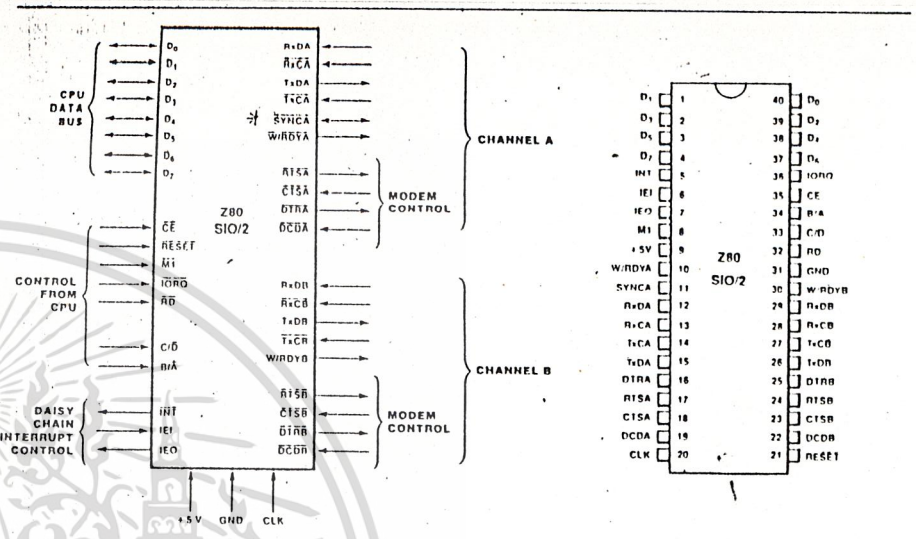


Figure 1. Pin Functions

Figure 2. 40-pin Dual-In-Line Package (DIP), Pin Assignments

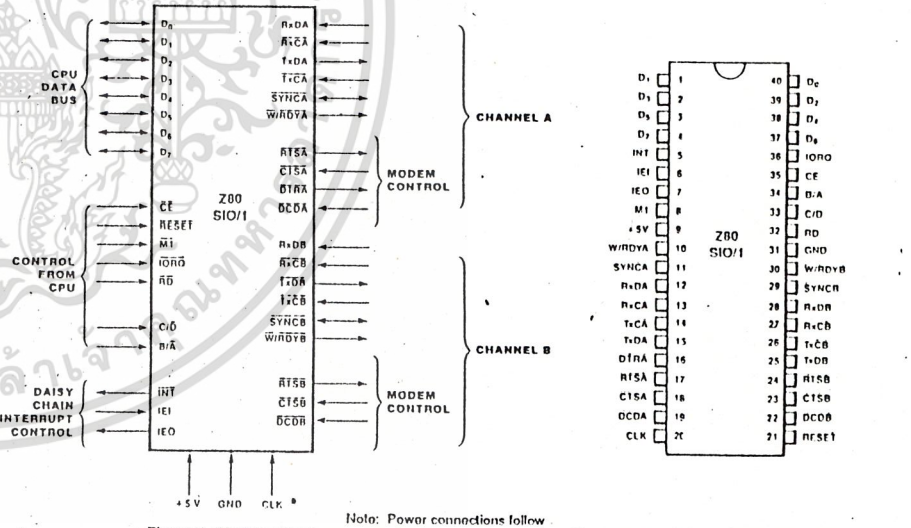


Figure 3. Pin Functions

Note: Power connections follow conventional descriptions below:

Connection	Circuit	Device
Power	V_{cc}	V_{DD}
Ground	GND	V_{SS}

Figure 4. 40-pin Dual-In-Line Package (DIP), Pin Assignments

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffering does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the Z80 SIO. They can also be programmed as general-purpose outputs.

In the Z80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. As mentioned previously, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle One* (input from Z80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32, or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered; no noise level margin is specified.

In the Z80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High, and disables all interrupts. The control registers must be rewritten after the SIO is reset and before data is transmitted or received.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (bidirectional, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{SYNC}$ must be driven Low on the second rising edge of $\overline{RxC}$ after that rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{SYNC}$ input. Once $\overline{SYNC}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{RxC}$ that immediately precedes the falling edge of $\overline{SYNC}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{RxC}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z80 SIO/2 bonding option, $\overline{SYNCB}$ is omitted.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32, or 64 times the data rate; however, the clock multiplier must be the same for the transmitter and the receiver. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements; no noise level margin is specified. Transmitter Clocks may be driven by the Z80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z80 SIO/0 bonding option, $\overline{TxCB}$ is bonded together with $\overline{RxCB}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{TxC}$.

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function; driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

FUNCTIONAL DESCRIPTION

The functional capabilities of the Z80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z80 family peripheral, it interacts with the Z80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling, and simple handshake capability. Figure 8 is a block diagram.

Figure 9 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

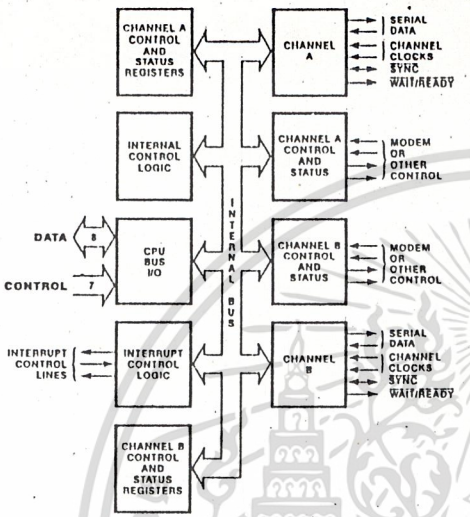


Figure 8. Block Diagram

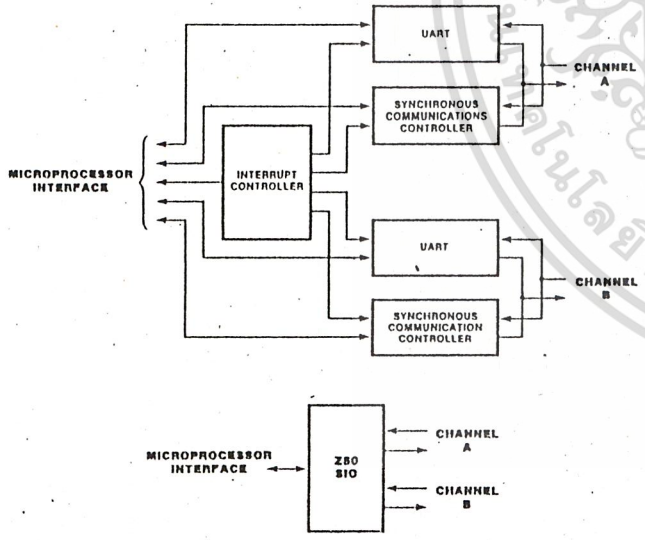


Figure 9. Conventional Devices Replaced by the Z80 SIO

DATA COMMUNICATION CAPABILITIES

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous, or synchronous data-communication protocol. Figure 10a illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z80 SIO Technical Manual* (03-3033-01).

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist, as in the case of a transient, the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit; a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals, a feature that allows it to be used with a Z80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16 bit sync pattern (Bisync), or with an external sync signal. Leading sync characters can be removed without interrupting the CPU.

Five-, six-, or seven bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple incoming sync characters, as shown in Figure 10b.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Figure 10a. Some Z80 SIO Protocols

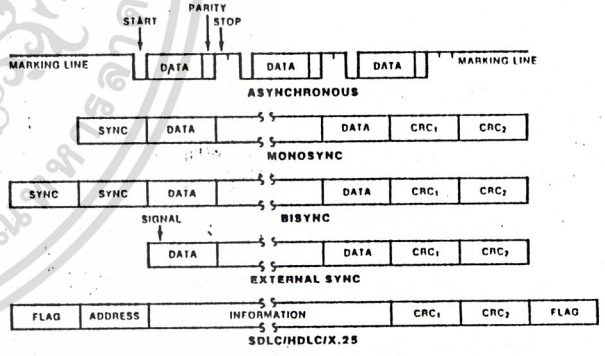


Figure 10b. Six-Bit Sync Character Recognition

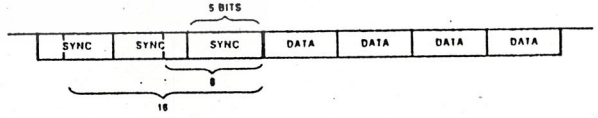


Figure 10. Data Communication

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non SDLC modes, the CRC generator is initialized to 0s; in SDLC modes, it is initialized to 1s. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disks, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{SYNC}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

I/O INTERFACE CAPABILITIES

The SIO offers the choice of polling, vectored or non-vectored interrupts and block-transfer modes to transfer data, status, and control information to, and from, the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts, and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit, and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$), and Synchronization ($\overline{SYNC}$) pins (Figures 1 through 7). In addition, an external/status interrupt is also caused by a CRC-sending condition, or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

In a Z80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{WAIT}$ line in the CPU block-transfer mode or as a $\overline{READY}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{READY}$ output indicates that the SIO is ready to transfer data to, or from, memory. To the CPU, the $\overline{WAIT}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

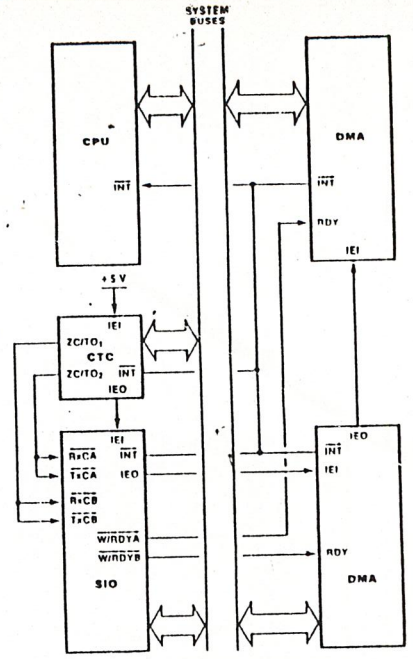


Figure 11. Typical Z80 Environment

INTERNAL STRUCTURE

The internal structure of the device includes a Z80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are

Table 1. Register Functions

	Read Register Functions
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)
	Write Register Functions
WR0	Register pointers; CRC initialize, and initialization commands for the various modes
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus; and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

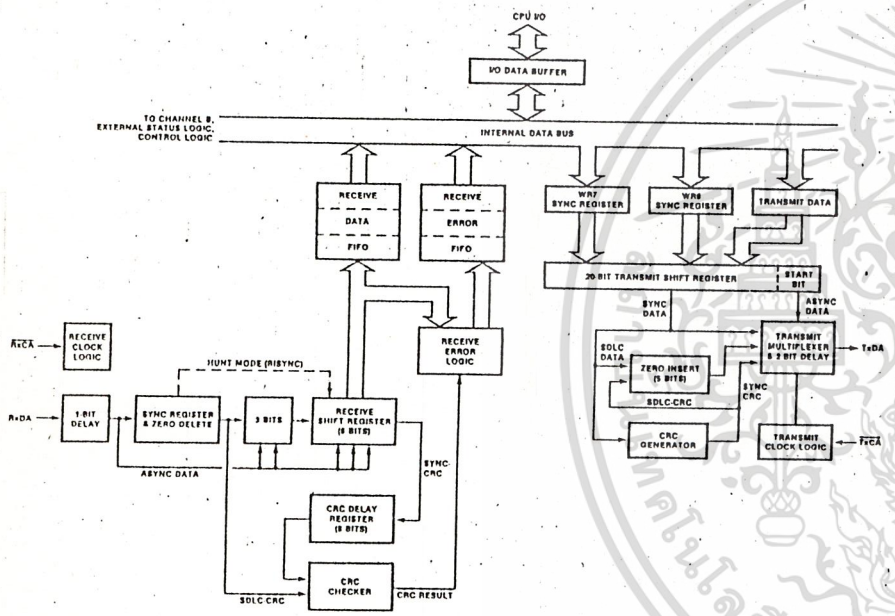


Figure 12. Transmit and Receive Data Path (Channel A)

PROGRAMMING

The system program first issues a series of commands that initialize the basic mode of operation and then issues other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information: RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector, and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

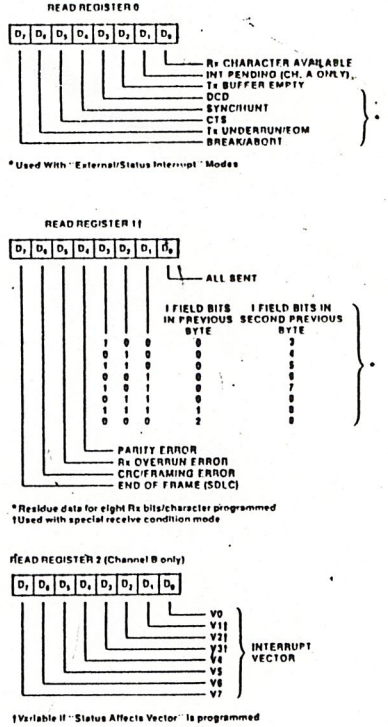


Figure 13. Read Register Bit Functions

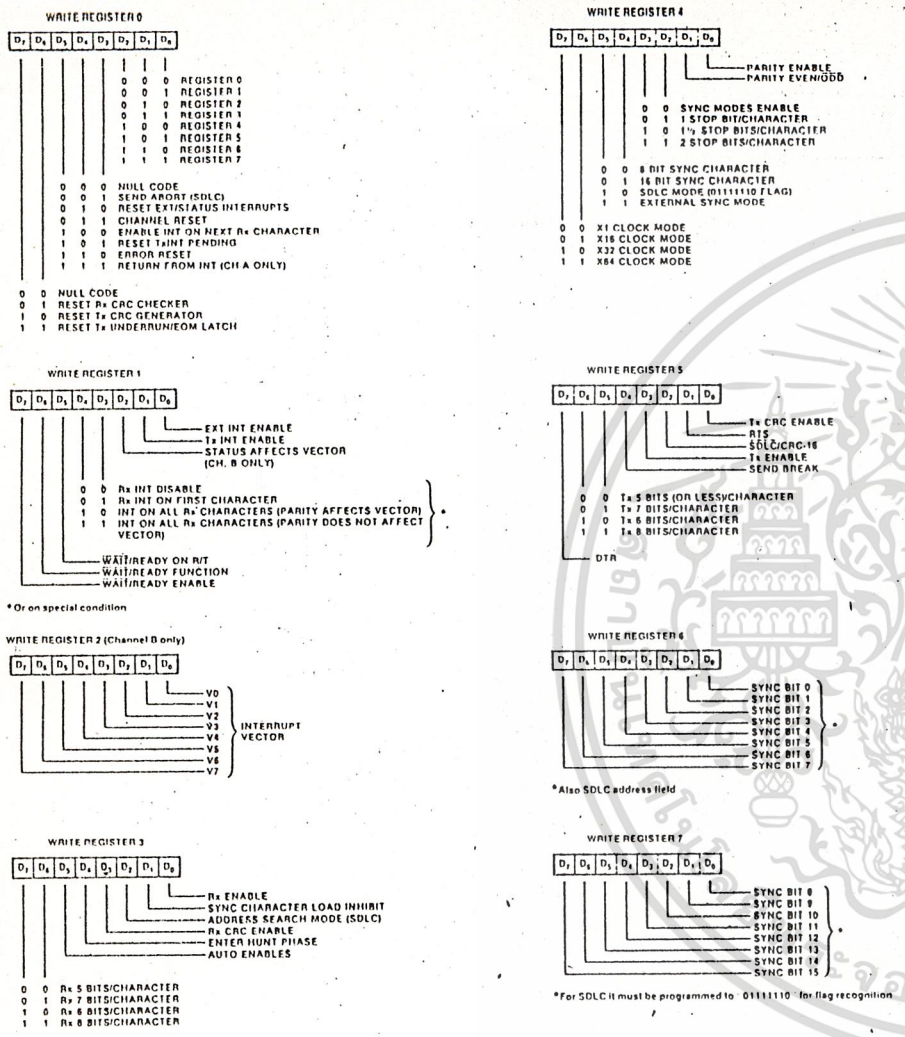


Figure 14. Write Register Bit Functions

TIMING

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z80 CPU sends an interrupt-acknowledge sequence, $\overline{M1}$ Low and $\overline{IORQ}$ Low, a few cycles later (Figure 17).

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z80 CPU issues a Return From Interrupt (RETI) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever ED is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is 4D, the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z8400 Z80 CPU Product Specification* (00-2001-04).

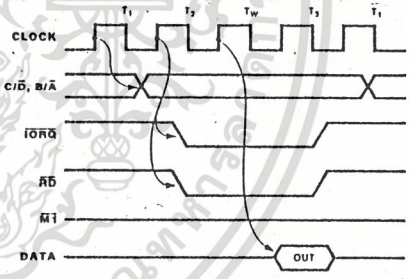


Figure 15. Read Cycle

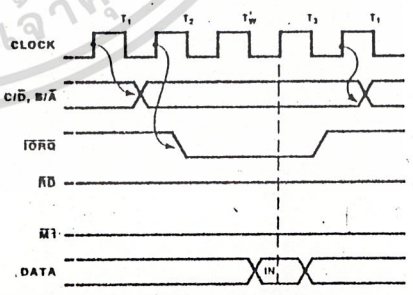


Figure 16. Write Cycle

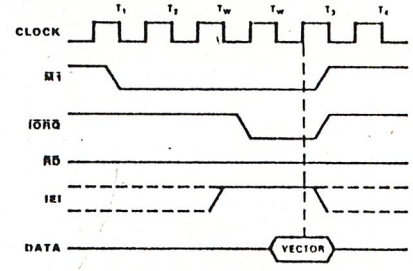


Figure 17. Interrupt Acknowledge Cycle

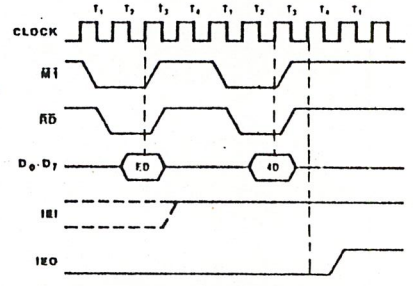


Figure 18. Return from Interrupt Cycle

AC CHARACTERISTICS (Z84C10 / CMOS Z80 DMA)
(Inactive State)

Number	Symbol	Parameter	Z84C1006 Min	Z84C1006 Max	Z84C1008 Min	Z84C1008 Max	Unit
1	TcC	Clock Cycle Time	162	DC	125	DC	
2	TwCh	Clock Width (High)	65	DC	55	DC	
3	TwCl	Clock Width (Low)	65	DC	55	DC	
4	TfC	Clock Rise Time		20		10	
5	TfC	Clock Fall Time		20		10	
6	Th	Hold Time for Any Specified Setup Time	0		0		ns
7	TsC(Cr)	$\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$ ↓ to Clock ↑ Setup	60		45		ns
8	TdDO(RDf)	$\overline{RD}$ ↓ to Data Output Delay		300		220	ns
9	TsDI(Cr)	Data In to Clock ↑ Setup ($\overline{WR}$ or $\overline{M1}$)	30		20		ns
10	TdDO(IOf)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		110		85	ns
11	TdRDr(Dz)	$\overline{RD}$ ↑ to Data Float Delay (output buffer disable)		70		50	ns
12	TsIEI(IORQf)	IEI to $\overline{IORQ}$ ↓ Setup (INTA Cycle)	100		80		ns
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		100		70	ns
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		100		70	ns
15	TdM1f(IEOf)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{M1}$ ↓)		100		80	ns
16	TsM1f(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup	70		45		ns
17	TsM1r(Cf)	$\overline{M1}$ ↑ to Clock Setup	-15		-15		ns
18	TsRDf(Cr)	$\overline{RD}$ ↓ to Clock ↑ Setup ($\overline{M1}$ Cycle)	60		45		ns
19	TdI(INTf)	Interrupt Cause to $\overline{INT}$ ↓ Delay ($\overline{INT}$ generated only when DMA is inactive)		450		400	ns
20	TdBAIr(BAOr)	$\overline{BAI}$ ↑ to $\overline{BAO}$ ↑ Delay		100		70	ns
21	TdBAIf(BAOf)	$\overline{BAI}$ ↓ to $\overline{BAO}$ ↓ Delay		100		70	ns
22	TsRDY(Cr)	RDY Active to Clock ↑ Setup	50		50		ns

NOTE: Negative minimum setup values mean that the first-mentioned event can come after the second mentioned event.

* M1 must be active for a minimum of two clock cycles to reset the DMA (This feature is only with C-MOS Z80 DMA).

Z8420/Z84C20 NMOS/CMOS Z80® PIO Parallel Input/Output

FEATURES

- Provides a direct interface between Z80 microcomputer systems and peripheral devices.
- Two ports with interrupt-driven handshake for fast response.
- Four programmable operating modes: Output, Input, Bidirectional (Port A only), and Bit Control
- Programmable interrupts on peripheral status conditions. (1.5 mV @ 1.5V)
- NMOS version for cost sensitive performance solutions.
- CMOS version for the designs requiring high speed and low power consumption
- NMOS Z0842004 - 4 MHz, Z0842006 - 6.17 MHz.
- CMOS Z84C2006 - DC to 6.17 MHz, Z84C2008 - DC to 8 MHz
- Standard Z80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5V).
- 6 MHz version supports 6.144 MHz CPU clock operation.

GENERAL DESCRIPTION

The Z80 PIO Parallel I/O Circuit (hereinafter referred to as the Z80 PIO or PIO) is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z80 CPU (Figures 1 and 2). Note the QFP package is only available in CMOS version. The CPU configures the Z80 PIO to interface with a wide range of peripheral devices that are compatible with the Z80 PIO include most keyboards, paper tape readers and punches, printers, and PROM programmers.

One characteristic of the Z80 peripheral controllers that separates them from other interface controllers is that all

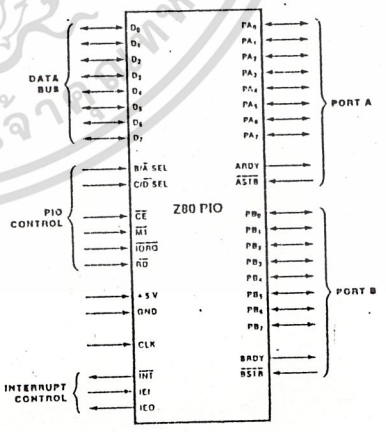


Figure 1. Pin Functions

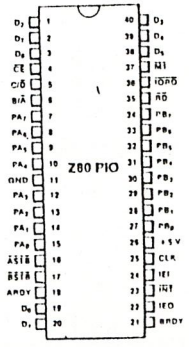


Figure 2a. 40-pin Dual-In-Line Package (DIP), Pin Assignments

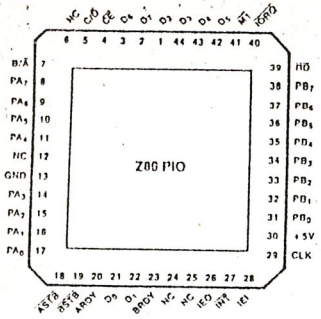


Figure 2b. 44-pin Chip Carrier, Pin Assignments

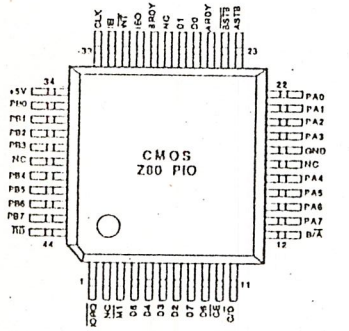


Figure 2c. 44-pin Quad Flat Pack Pin Assignments.

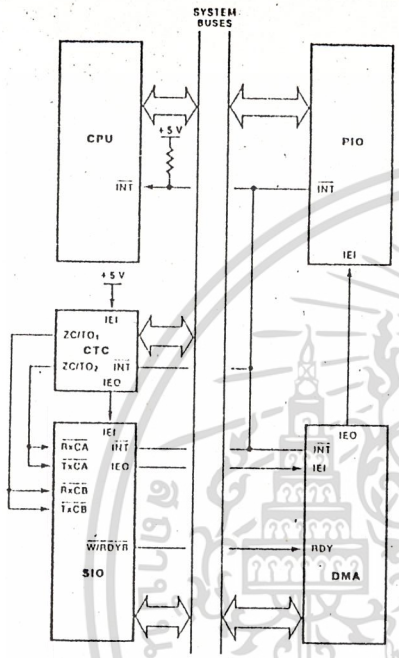


Figure 3. PIO in a Typical Z80 Family Environment

data transfer between the peripheral device and the CPU is accomplished under interrupt control. Thus, the interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO (Figure 3).

Another feature of the PIO is the ability to interrupt the CPU upon occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

The Z80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates when a data transfer has occurred.

Operating Modes. The Z80 PIO ports can be programmed to operate in four modes: Output (Mode 0), Input (Mode 1), Bidirectional (Mode 2) and Bit Control (Mode 3).

Either Port A or Port B can be programmed to output data in Mode 0. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

Either Port A or Port B can be programmed to input data in Mode 1. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is detected by the external device. The external device then places data on the I/O lines and strobes the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses only Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3 and masked off from generating interrupts. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except that data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. For input, operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt, if enabled.

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR) or all inputs active (AND). For example, if the port is programmed for active Low inputs and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

Data outputs are controlled by the CPU and can be written or changed at any time.

- Individual bits can be masked off.
- The handshake signals are not used in Mode 3; Ready is held Low, and Strobe is disabled.
- When using the Z80 PIO interrupts, the Z80 CPU interrupt mode must be set to Mode 2.

INTERNAL STRUCTURE

The internal structure of the Z80 PIO consists of a Z80 CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic (Figure 4). The CPU bus interface logic allows the Z80 PIO to interface directly to the Z80 CPU with no other external logic. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

Port Logic. Each port contains separate input and output registers, handshake control logic, and the control registers shown in Figure 5. All data transfers between the peripheral unit and the CPU use the data input and output registers. The handshake logic associated with each port controls the data transfers through the input and the output registers. The mode control register (two bits) selects one of the four programmable operating modes.

The Bit Control mode (Mode 3) uses the remaining registers. The input/output control register specifies which of the eight data bits in the port are to be outputs and enables these bits; the remaining bits are inputs. The mask register and the mask control register govern Mode 3 interrupt conditions. The mask register specifies which of the bits in the port are active and which are masked or inactive.

The mask control register specifies two conditions: first, whether the active state of the input bits is High or Low, and second, whether an interrupt is generated when any one unmasked input bit is active (OR condition) or if the interrupt is generated when *all* unmasked input bits are active (AND condition).

Interrupt Control Logic. The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. Any device's physical location in a

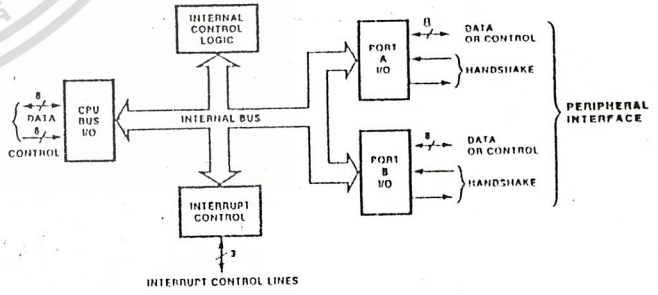


Figure 4. Block Diagram

daisy-chain configuration determines its priority. Two lines (IEI and IEO) are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output, or bidirectional modes, an interrupt can be generated whenever the peripheral requests a new byte transfer. In the bit control mode, an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routines completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

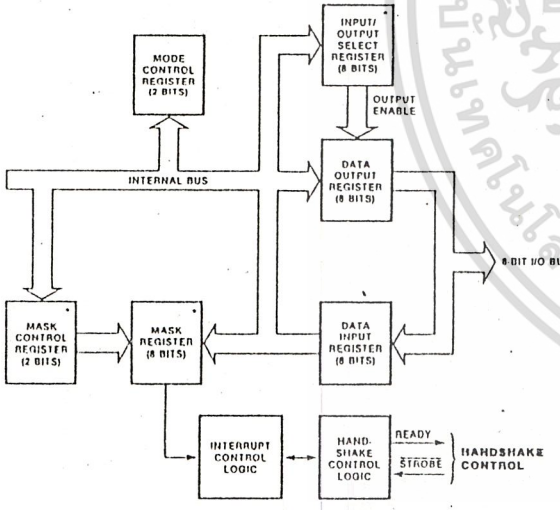
If the CPU (in interrupt Mode 2) accepts an interrupt, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector forms a pointer to a location in memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant eight bits of the indirect pointer while the I Register in the CPU provides the most significant eight bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to 0 within the PIO because the pointer must point to two adjacent memory locations for a complete 16-bit address.

Unlike the other Z80 peripherals, the PIO does not enable interrupts immediately after programming. It waits until $\overline{M1}$ goes Low (e.g., during an opcode fetch). This condition is unimportant in the Z80 environment but might not be if another type of CPU is used.

The PIO decodes the RETI (Return From Interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine. No other communication with the CPU is required.

CPU Bus I/O Logic. The CPU bus interface logic interfaces the Z80 PIO directly to the Z80 CPU, so no external logic is necessary. For large systems, however, address decoders and/or buffers may be necessary.

Internal Control Logic. This logic receives the control words for each port during programming and, in turn, controls the operating functions of the Z80 PIO. The control logic synchronizes the port operations, controls the port mode, port addressing, selects the read/write function, and issues appropriate commands to the ports and the interrupt logic. The Z80 PIO does not receive a write input from the CPU; instead, the $\overline{RD}$, $\overline{CE}$, C/$\overline{D}$ and $\overline{IORQ}$ signals internally generate the write input.



*Used in the bit mode only to allow generation of an interrupt if the peripheral I/O pins go to the specified state.

Figure 5. Typical Port I/O Block Diagram

PROGRAMMING

Mode 0, 1, or 2. (Input, Output, or Bidirectional). Programming a port for Mode 0, 1, or 2 requires at least one, and up to three, control words per port. These words are:

Mode Control Word (Figure 6). Selects the port operating mode. This word is required and may be written at any time.

Interrupt Vector Word (Figure 7). The Z80 PIO is designed for use with the Z80 CPU in interrupt Mode 2. This word must be programmed if interrupts are to be used.

Interrupt Control Word (Figure 9) or ***Interrupt Disable Word*** (Figure 11). Controls the enable or disable of the PIO interrupt function.

Mode 3 (Bit Control). Programming a port for Mode 3 requires at least two, and up to four, control words.

Mode Control Word (Figure 6). Selects the port operating mode. This word is required and may be written at any time.

I/O Register Control Word (Figure 8). When Mode 3 is selected, the Mode Control Word must be followed by the I/O Control Word. This word configures the I/O control register, which defines which port lines are inputs or outputs. This word is required.

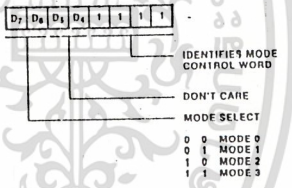


Figure 6. Mode Control Word

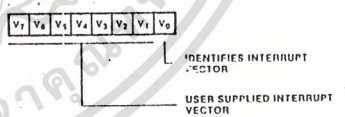


Figure 7. Interrupt Vector Word

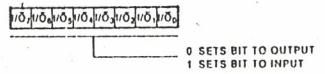


Figure 8. I/O Register Control Word

Interrupt Vector Word (Figure 7). The Z80 PIO is designed for use with the Z80 CPU in interrupt Mode 2. This word must be programmed if interrupts are to be used.

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_5 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable Word. This control word can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

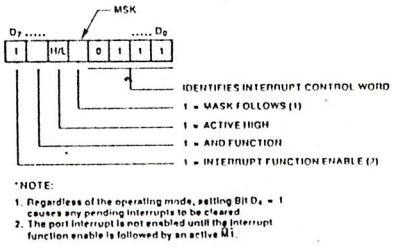


Figure 9. Interrupt Control Word

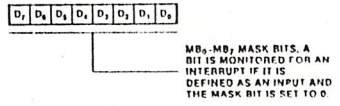


Figure 10. Mask Control Word

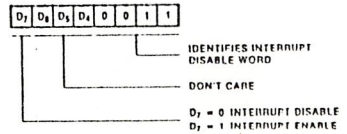


Figure 11. Interrupt Disable Word

PIN DESCRIPTION

PA_0-PA_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. PA_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{ASTB}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{ASTB}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

PB_0-PB_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5V to drive Darlington transistors. PB_0 is the least significant bit of the bus.

B/$\overline{A}$. *Port B or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{BSTB}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{ASTB}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{D}$. *Control or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{A}$ Select line. A Low on this pin means that the Z80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z80 PIO uses the standard single-phase Z80 system clock.

D_0-D_7. *Z80 CPU Data Bus* (bidirectional, 3 state). This bus is used to transfer all data and commands between the Z80 CPU and the Z80 PIO. D_0 is the least significant bit.

IEI. *Interrupt Enable In* (input, active High). This signal is used to form a priority-interrupt daisy chain when more than one interrupt driven device is being used. A High level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When $\overline{INT}$ is active the Z80 PIO is requesting an interrupt from the Z80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from Z80 CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{RD}$ to transfer commands and data between the Z80 CPU and the Z80 PIO. When $\overline{CE}$, $\overline{RD}$, and $\overline{IORQ}$ are active, the port addressed by B/$\overline{A}$ transfers data to the CPU (a read operation). Conversely, when $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is not, the port addressed by B/$\overline{A}$ is written into from the CPU with either data or control information, as specified by C/$\overline{D}$. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{M1}$ and $\overline{RD}$ signals are active, the Z80 CPU is fetching an instruction from memory. Conversely, when both $\overline{M1}$ and $\overline{IORQ}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{M1}$ has two other functions within the Z80 PIO: it synchronizes the PIO interrupt logic; when $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal, the PIO is reset.

$\overline{RD}$. *Read Cycle Status* (input from Z80 CPU, active Low). If $\overline{RD}$ is active, or an I/O operation is in progress, $\overline{RD}$ is used with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the Z80 PIO to the Z80 CPU.

TIMING

The following timing diagrams show typical timing in a Z80 CPU environment. For more precise specifications refer to the composite ac timing diagram.

Write Cycle. Figure 12 illustrates the timing for programming the Z80 PIO or for writing data to one of its ports. The PIO does not receive a specific write signal; it internally generates its own from the lack of an active $\overline{RD}$ signal.

Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z80 PIO ports.

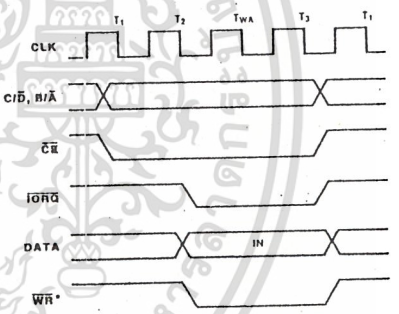


Figure 12. Write Cycle Timing

Output Mode (Mode 0). An output cycle (Figure 14) is always started by the execution of an output instruction by the CPU. The $\overline{WR}$* pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The $\overline{WR}$* pulse sets the Ready flag after a Low-going edge of CLK, indicating data is available. Ready stays active until the positive edge of the strobe line is received, indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{INT}$ if the interrupt enable flip-flop has been set and if this device has the highest priority.

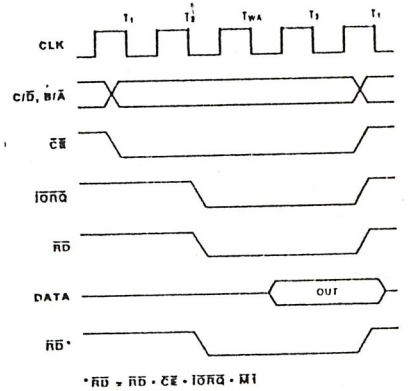


Figure 13. Read Cycle Timing

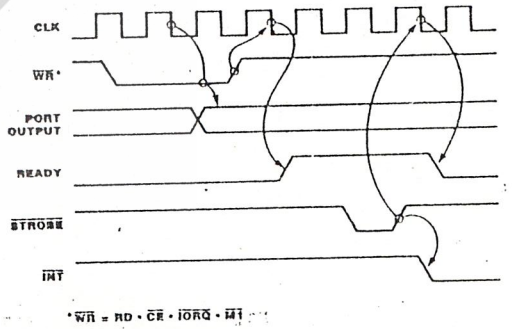


Figure 14. Mode 0 Output Timing

Input Mode (Mode 1). When $\overline{STROBE}$ goes from Low to High, data is latched into the selected port input register (Figure 15). While $\overline{STROBE}$ is Low, the input data latches are transparent. The next rising edge of $\overline{STROBE}$ activates $\overline{INT}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets Ready to an inactive state, indicating that the input register is full and cannot accept any more data until the CPU completes a read. When a read is complete, the positive edge of $\overline{RD}$ sets Ready at the next Low-going transition of CLK. At this time new data can be loaded into the PIO.

Bidirectional Mode (Mode 2). This is a combination of Modes 0 and 1 using all four handshake lines and the eight Port A I/O lines (Figure 16). Port B must be set to the bit mode and its inputs must be masked. The Port A handshake lines are used for output control and the Port B lines are used for input control. If interrupts occur, Port A's vector will be used during port output and Port B's will be used during port input. Data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. The rising edge of this strobe can be used to latch the data into the peripheral.

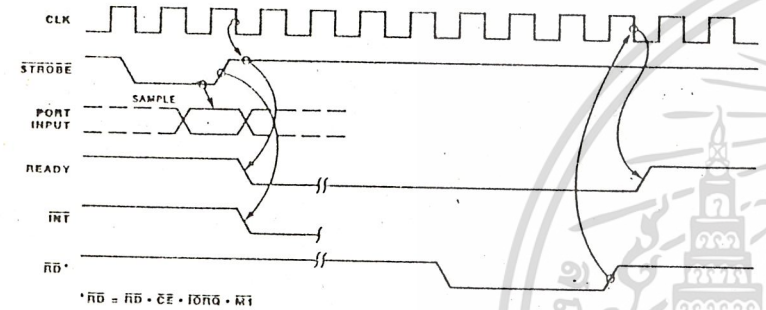


Figure 15. Mode 1 Input Timing

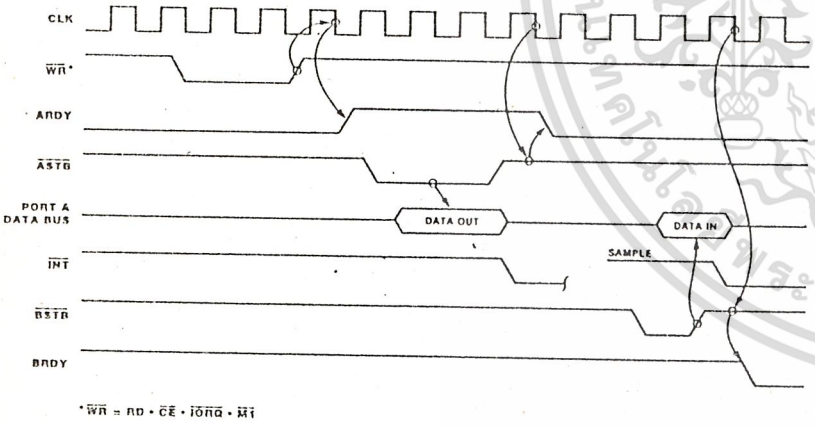


Figure 16. Mode 2 Bidirectional Timing

Bit Control Mode (Mode 3). The bit mode does not utilize the handshake signals, and a normal port write or port read can be executed at any time. When writing, the data is latched into the output registers with the same timing as the output mode.

When reading (Figure 17) the PIO, the data returned to the CPU is composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register contains data that was present immediately prior to the falling edge of $\overline{RD}$. An interrupt is generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers. However, if Port A is programmed in bidirectional mode, Port B does not issue an interrupt in bit mode and must therefore be polled.

Interrupt Acknowledge Timing. During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the Interrupt Enable signal to ripple through the daisy chain. The peripheral with IEI High and IEO Low during INTACK places a preprogrammed 8-bit interrupt vector on the data bus at this time (Figure 18). IEO is held Low until a Return From Interrupt (RETI) instruction is executed by the CPU while IEI is High. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.

Return From Interrupt Cycle. If a Z80 peripheral has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always Low, inhibiting lower priority devices from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO is Low unless an "ED" is decoded as the first byte of a 2-byte opcode (Figure 19). In this case, IEO goes High until the next opcode byte is decoded, whereupon it goes Low again. If the second byte of the opcode was a "4D," then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service has its IEI High and its IEO Low. This device is the highest-priority device in the daisy chain that has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D," this peripheral device resets its "interrupt under service" condition.

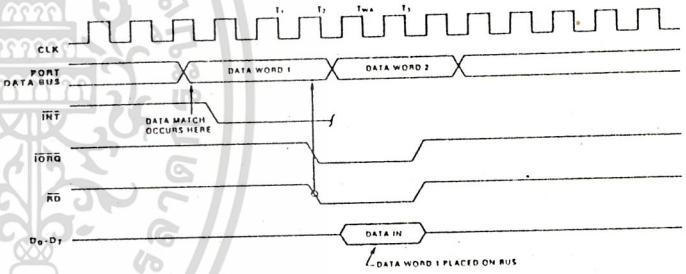


Figure 17. Mode 3 Bit Control Mode Timing, Bit Mode Read

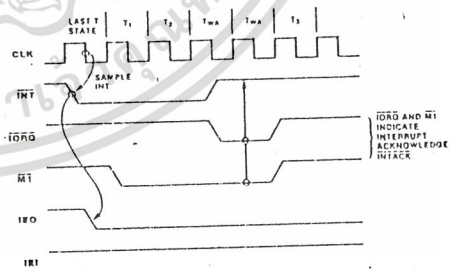


Figure 18. Interrupt Acknowledge Timing

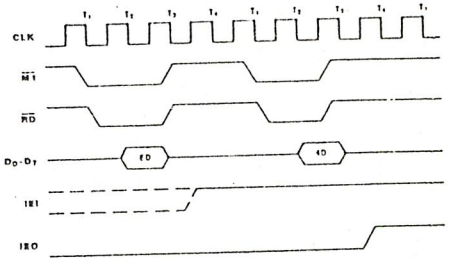


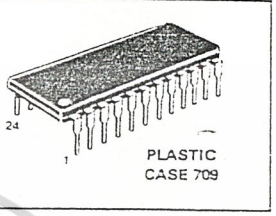
Figure 19. Return From Interrupt

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC145421
MC145425

Advance Information

ISDN Universal Digital Loop Transceivers II (UDLT II)



2

The MC145421 and MC145425 UDLTs are high-speed data transceivers capable of providing 160 kbps full duplex data communication over 26 awg and larger twisted-pair cable up to 1 km in length. These devices are primarily used in digital subscriber voice and data telephone systems. In addition, the devices meet and exceed the CCITTs recommendations for data transfer rates of ISDNs on a single twisted pair. The devices utilize a 512 kilobaud MDPSK burst modulation technique to supply the 160 kbps full duplex data transfer rates. The 160 kbps rate is provided through four channels. There are two B channels, which are 64 kbps each. In addition, there are two D channels which are 16 kbps each.

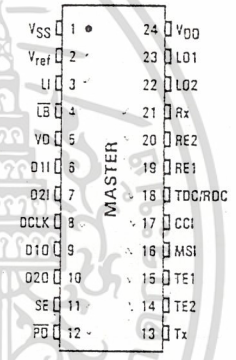
The MC145421 and MC145425 UDLTs are designed for upward compatibility with the existing MC145422 and MC145426 80 kbps UDLTs, as well as compatibility with existing and evolving telephone switching hardware and software architectures.

The MC145421 (MASTER) UDLT is designed for use at the telephone switch line card while the MC145425 (SLAVE) UDLT is designed for use at the remote digital telset or data terminal.

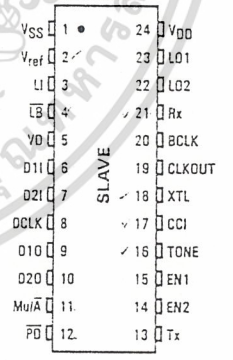
- Employs CMOS Technology, in Order to Take Advantage of Its Proven Capability for Complex Analog and Digital LSI Functions.
- Provides Synchronous Full Duplex 160 kbps Voice and Data Communication in a 2B + 2D Format For ISDN Compatibility.
- Provides the CCITT's Basic Access Data Transfer Rate (2B + D) for ISDNs on a Single Twisted Pair up to 1 km.
- Compatible with Existing and Evolving Telephone Switch Architectures and Call Signalling Schemes.
- Protocol Independent
- Single +5 V Power Supply

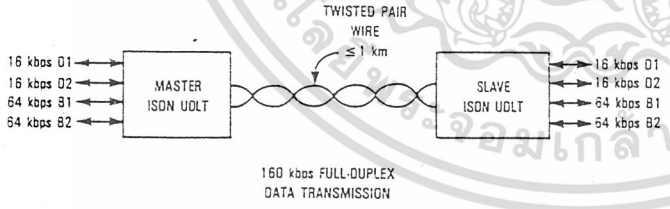
PIN ASSIGNMENTS

MC145421



MC145425

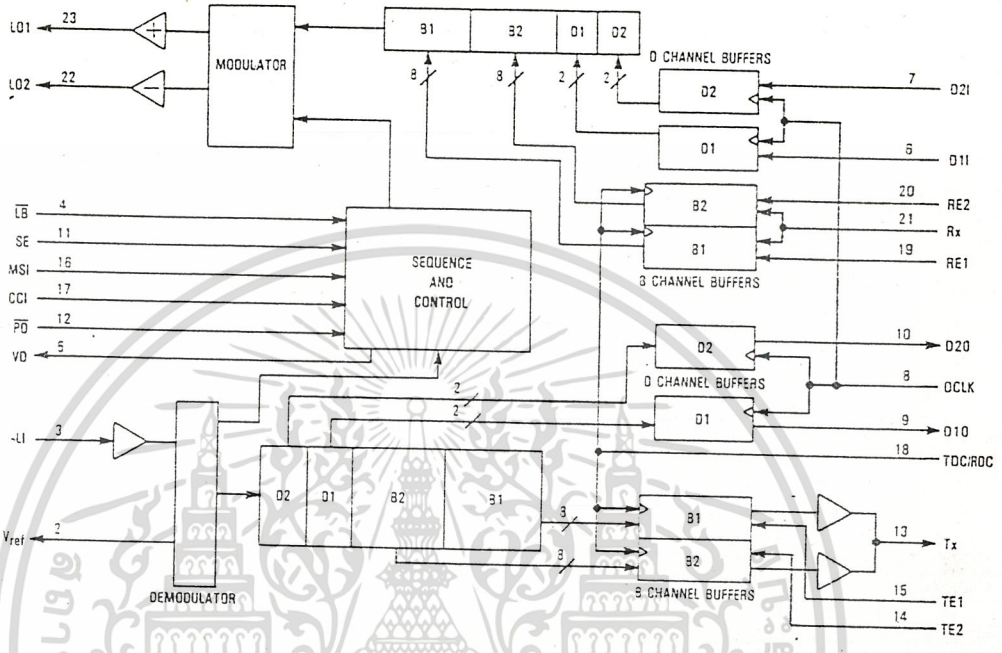




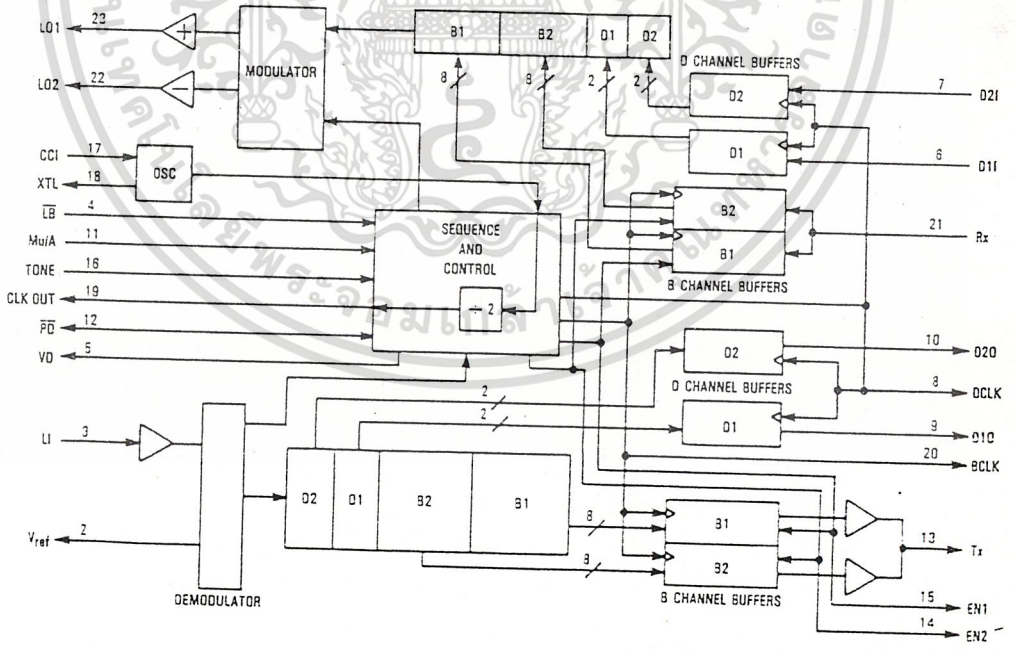
This document contains information on a new product. Specifications and information herein are subject to change without notice.

2

MC145421 MASTER ISDN BLOCK DIAGRAM



MC145425 SLAVE ISDN BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS (Voltage Referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}-V_{SS}	−0.5 to 6.5	V
Voltage Any Pin to V_{SS}	V_{in}	−0.5 to V_{DD} + 0.5	V
DC Current, Any Pin (Excluding V_{DD}, V_{SS})	I	± 10	mA
Operating Temperature	T_A	−40 to +85	°C
Storage Temperature	T_{stg}	−65 to +150	°C

This device contains circuitry to protect the inouts against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid applications of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

RECOMMENDED OPERATING CONDITIONS (T_A = −40 to +85°C)

Parameter	Pins	Min	Typ	Max	Unit
DC Supply Voltage	V_{DD}	4.5	5.0	5.5	V
Frame Rate MC145421 (See Note 1)	MSI	—	8.0	—	kHz
MC145421/25 Frame Slip Rate (See Note 1)	—	—	—	0.25	%
CCI Clock Frequency	—	—	8.192	8.29	MHz
TDC/RDC Data Clocks (for Master)	—	0.128	—	4.1	MHz
DCLK	—	0.016	—	4.1	MHz
Modulation Baud Rate (CCI/16)	LO1, LO2	—	512	—	kHz

NOTE:
1. The slave's crystal frequency divided by 1024 must equal the master's MSI frequency ± 0.25% for optimum operation. Also, the 8.192 MHz input at the master divided by 1024 must be within 0.048% of the master's 8 kHz MSI clock frequency.

DIGITAL CHARACTERISTICS (V_{DD} = 5 V, T_A = −40 to +85°C)

Parameter		Min	Max	Unit
Input High Level		3.5	—	V
Input Low Level		—	1.5	V
Input Current, V_{DD}		—	15	mA
Input Current (Digital Pins)		—	5	µA
Input Capacitance		—	10	pF
Output High Current (Except Tx on Master and Slave, and $\overline{PD}$ on the Slave)	V_{OH} = 2.5 V_{OH} = 4.6	−1.7 −0.36	— —	mA
Tx Output High Current	V_{OH} = 2.5 V_{OH} = 4.6	−3.4 −0.7	— —	mA
$\overline{PD}$ (Slave Output High Current (See Note 2)	V_{OH} = 2.5	—	−90	µA
Output Low Current (Except Tx on Master and Slave, and $\overline{PD}$ on Slave)	V_{OL} = 0.4 V_{OL} = 0.8	0.36 0.8	— —	mA
Tx Output Low Current	V_{OL} = 0.4 V_{OL} = 0.8	1.7 3.5	— —	mA
$\overline{PD}$ (Slave) Output Low Current (See Note 2)	V_{OL} = 0.4	30	60	µA
Tx Three-State Impedance		100	—	kΩ
XTL Output High Current	V_{OH} = 4.6	—	−450	µA
XTL Output Low Current	V_{OL} = 0.4	450	—	µA

NOTE:
2. To overdrive $\overline{PD}$ from a low level to 3.5 V, or a high level to 1.5 V requires a minimum of ± 800 µA drive capability.

ANALOG CHARACTERISTICS (V_{DD} = 5 V, T_A = 0 to 70°C)

Parameter	Min	Max	Unit
Modulation Differential Amplitude RL = 880 Ω (LO1-LO2)	4.6	—	Vpeak
Modulation Differential DC Offset	—	40	mV
V_{ref} Voltage (Typically 9/20•(V_{DD} − V_{SS}))	2.0	2.5	V
PCM Tone Level	−22	−18	dBm
Demodulator Input Amplitude	50	—	mVpeak
Demodulator Input Impedance (LI to V_{ref})	75	300	kΩ

2

MC145421 MASTER PIN DESCRIPTIONS

V_{DD}—POSITIVE SUPPLY (PIN 24)

The most positive power supply pin, normally +5 volts with respect to V_{SS}.

V_{SS}—NEGATIVE SUPPLY (PIN 1)

The most negative supply pin and logic ground, normally 0 volts.

V_{ref}—REFERENCE OUTPUT (ANALOG GROUND) (PIN 2)

This pin is the output of the internal reference supply and should be bypassed to V_{DD} and V_{SS} with 0.1 μF capacitors. This pin usually serves as an analog ground reference for transformer coupling of the device's incoming bursts from the line. No external dc load should be placed on this pin.

LI—LINE INPUT (PIN 3)

This pin is an input to the demodulator for the incoming bursts. The input has an internal 240 kΩ resistor tied to the V_{ref} pin, so an external capacitor or line transformer may be used to couple the input signal to the device with no dc offset.

LO1, LO2—LINE DRIVER OUTPUTS (PINS 23, 22)

These push-pull outputs drive the twisted pair transmission line with a 512 kHz modified DPSK (MDPSK) burst each 125 μs, in other words at an 8 kHz rate. When not modulating the line, these pins are driven to the active high state—being the same potential, they create an ac short. When used in conjunction with feed resistors, proper line termination is maintained.

SE—SIGNAL ENABLE INPUT (PIN 11)

At the time of a negative transition on this pin, an internal latch stores the states of $\overline{LB}$ and $\overline{PD}$ for as long as SE is held low. During this time, the VD, DO1, and DO2 outputs are driven to the high impedance state. When SE is high, all pins function normally.

$\overline{LB}$—LOOP BACK CONTROL (PIN 4)

A low level on this pin ties the internal modulator output to the internal demodulator input which loops the entire burst for testing purposes. During the loopback operation, the LI input is ignored and the LO1 and LO2 drivers are driven to the active high level. The state of this pin is internally latched if the SE pin is held low. This feature is only active when the $\overline{PD}$ input is high.

$\overline{PD}$—POWER DOWN INPUT (PIN 12)

When held low the ISDN UDLT powers down, except the circuitry that is necessary to demodulate an incoming burst and to output VD, B channel and D channel data bits. When $\overline{PD}$ is brought high, the ISDN UDLT powers up. Then, it begins transmitting every MSI period to the slave device, shortly after the rising edge of MSI. The state of this pin is latched if the SE pin is held low.

VD—VALID DATA OUTPUT (PIN 5)

A high level on this pin indicates that a valid line transmission has been demodulated. A valid transmission burst is determined by proper synchronization and the absence of detected bit errors. VD changes state on the rising edge of MSI when $\overline{PD}$ is high. When $\overline{PD}$ is low, VD changes state at the end of demodulation of a transmission burst and does not change again until three MSI rising edges have occurred, at which time it goes low, or until the next demodulation of a burst. VD is a standard B-series CMOS output and is high-impedance when SE is low.

MSI—MASTER SYNC INPUT (PIN 16)

This pin is the master, 8 kHz, frame reference input. The rising edge of MSI loads B and D channel data which had been input during the previous frame into the modulator section of the device and initiates the out-bound burst onto the twisted-pair cable. The rising edge of MSI also initiates the buffering of the B and D channel data demodulated during the previous frame. MSI should be approximately leading edge aligned with the TDC/RDC data clock input pin.

CCI—HIGH-SPEED CLOCK INPUT (PIN 17)

An 8.192 MHz clock should be supplied to this input. The 8.192 MHz input should be 50% duty cycle, however it may free-run with respect to all other clocks without performance degradation.

D1I, D2I—D CHANNEL SIGNALING BIT INPUTS (PINS 6, 7)

These inputs are 16 kbps serial data inputs. Two bits should be clocked into each of these inputs between the rising edges of the MSI frame reference clock. The first bit of each D channel is clocked into an intermediate buffer on the first falling edge of the DCLK following the rising edge of MSI. The second bit of each D channel is clocked in on the next negative transition of the DCLK. If further DCLK negative edges occur, new information is serially clocked into the buffer replacing the previous data one bit at a time. Buffered D channel data bits are burst to the slave device on the next rising edge of the MSI frame reference clock.

D1O, D2O—D CHANNEL SIGNAL OUTPUTS (PINS 9, 10)

These serial outputs provide the 16 kbps D channel signaling information from the incoming burst. Two data bits should be clocked out of each of these outputs between the rising edges of the MSI frame reference clock. The rising edge of MSI produces the first bit of each D channel on its respective pin. Circuitry then searches for a negative D clock edge. This tells the D channel data shift register to produce the second D channel bit on the next rising edge of the DCLK. Further positive edges of the DCLK recirculate the D channel output buffer information.

DCLK—D CHANNEL CLOCK INPUT (PIN 8)

This input is the transmit and receive data clock for both D channels. D channel input and output operation is described in the D1O, D2O pin description.

Tx—TRANSMIT DATA OUTPUT (PIN 13)

This pin is high impedance when both TE1 and TE2 are low. This pin serves as an output for B channel information received from the slave device. The B channel data is under the control of TE1, TE2, and TDC/RDC. (See TE1, TE2 description.)

Rx—RECEIVE DATA INPUT (PIN 21)

B channel data is input on this pin and is controlled by the RE1, RE2, and TDC/RDC pins. (See RE1, RE2 description.)

TE1, TE2—TRANSMIT DATA ENABLE INPUT (PINS 14, 15)

These two pins control the output of data for their respective B channel on the Tx output pin. When both TE1 and TE2 are low, the Tx pin is high impedance. The rising edge of the respective enable produces the first bit of the selected B channel data on the Tx pin. Internal circuitry then scans for the next negative transition of the TDC/RDC clock. Following this event the next seven bits of the selected B channel data are output on the next seven rising edges of the TDC/RDC data clock. When TE1 and TE2 are high simultaneously, data on the Tx pin is undefined. TE1 and TE2 should be approximately leading-edge aligned with the TDC/RDC data clock signal. In order to keep the Tx pin out of the high-impedance state, these enable lines should be high while the respective B channel data is being output.

RE1, RE2—RECEIVE DATA ENABLE INPUTS (PINS 19, 20)

These inputs control the input of B channel data on the Rx pin of the device. The rising edge of the respective enable signal causes the device to load the selected receive data buffer with data from the Rx pin on the next eight falling edges of the TDC/RDC clock input. The RE1 and RE2 enables should be roughly leading-edge aligned with the TDC/RDC data clock input. These enables are rising edge sensitive and need not be high for the entire B-channel input period.

TDC/RDC—TRANSMIT/RECEIVE DATA CLOCK INPUT (PIN 18)

This input is the transmit and receive data clock for the B channel data. As described in the TE1/TE2 and the RE1/RE2 sections, output data changes state on the rising edge of this signal, and input data is read on the falling edges of this signal. TDC/RDC should be roughly leading-edge aligned with the TE1, TE2, RE1, and RE2 enables, as well as the MSI frame reference signal.

MC145425 SLAVE PIN DESCRIPTIONS

V_{DD}—POSITIVE SUPPLY (PIN 24)

The most positive power supply pin, normally +5 volts with respect to V_{SS}.

V_{SS}—NEGATIVE SUPPLY (PIN 1)

The most negative supply pin and logic ground, normally 0 volts.

V_{ref}—REFERENCE OUTPUT (ANALOG GROUND) (PIN 2)

This pin is the output of the internal reference supply and should be bypassed to V_{DD} and V_{SS} with 0.1 μF capacitors. This pin usually serves as an analog ground reference for transformer coupling of the device's incoming bursts from the line. No external dc load should be placed on this pin.

LI—LINE INPUT (PIN 3)

This pin is an input to the demodulator for the incoming bursts. The input has an internal 240 kΩ resistor tied to the V_{ref} pin, so an external capacitor or line transformer may be used to couple the input signal to the device with no dc offset.

LO1, LO2—LINE DRIVER OUTPUTS (PINS 23, 22)

These push-pull outputs drive the twisted pair transmission line with a 512 kHz modified DPSK (MDPSK) burst each 125 μs; in other words at an 8 kHz frame rate. When not modulating the line, these pins are driven to the active high state—being the same potential, they create an ac short. So when used in conjunction with feed resistors, proper line termination is maintained.

CLK OUT—CLOCK OUTPUT (PIN 19)

This pin serves as a buffered output of the crystal frequency divided by two. This clock is provided for systems using the MC145428 Data Set Interface asynchronous/synchronous terminal adaptor device.

$\overline{LB}$—LOOPBACK CONTROL INPUT (PIN 4)

When this pin is low, the incoming B channels from the master are burst back to the master—instead of the Rx B channel input data. The B channel data from the master continues to be output at the slave's Tx pin during loopback. If the TONE and the loopback function are active simultaneously, the loopback function overrides the TONE function. D channel data is not affected by $\overline{LB}$.

VD—VALID DATA OUTPUT (PIN 5)

A high on this pin indicates that a valid transmission burst has been demodulated. A valid burst is determined by proper synchronization and the absence of detected bit errors. If no transmissions from the master have been received in the last 250 μs, as determined by an internal oscillator, VD will go low.

Mu/$\overline{A}$—TONE FORMAT INPUT (PIN 11)

This pin determines the PCM code for the 500 Hz square wave tone generated when the TONE input is high—Mu law (Mu/$\overline{A}$ = 1) or CCITT A law (Mu/$\overline{A}$ = 0) format.

TONE—TONE ENABLE INPUT (PIN 16)

A high on this pin causes a 500 Hz square wave PCM tone to be inserted in place of the demodulated B channel data on B channel 1. This feature allows the designer to provide audio feedback for telset keyboard operations.

$\overline{PD}$—POWER DOWN INPUT/OUTPUT (PIN 12)

This is a bidirectional pin with a weak output driver so that it can be externally overdriven. When held low, the ISDN UDLT is powered down, and the only active circuitry is that which is necessary for demodulation, generation of EN1, EN2, BCLK, and DCLK, and outputting of the data bits and VD. When held high, the ISDN UDLT is powered up and transmits normally in response to received bursts from the master. If the ISDN UDLT is powered up for 250 μs—which is derived from an internal oscillator and no bursts from the master have occurred, the ISDN slave UDLT generates a free-running set of

EN1, EN2, BCLK, and DCLK signals and sends a burst to the master device every other 125 μs frame. This is a wake-up signal to the master.

When $\overline{PD}$ is floating and a burst from the master is demodulated, the weak output drivers will try to force $\overline{PD}$ high. It will try to force $\overline{PD}$ low if 250 μs have elapsed without a burst from the master being successfully demodulated. This allows the slave device to self power up and down in demand-powered loop systems.

CCI—CRYSTAL INPUT (PIN 17)

Normally, an 8.192 MHz crystal is tied between this pin and the XTL pin. A 10 MΩ resistor between CCI and XTL and 25 pF capacitors from CCI and XTL to V_{SS} are required to ensure stability and start-up. CCI may also be driven with an external 8.192 MHz signal if a crystal is not desired.

XTL—CRYSTAL OUTPUT (PIN 18)

This pin is capable of driving one external CMOS input and 15 pF of additional load capacitance.

D1I, D2I—D CHANNEL INPUTS (PINS 6, 7)

These two pins are inputs for the 16 kbps D data channels. The D channel data bits are clocked in serially on the negative edge of the 16 kbps DCLK output pin.

D1O, D2O—D CHANNEL OUTPUTS (PINS 9, 10)

These two pins are outputs for the 16 kbps D data channels. These pins are updated on the rising edges of the slave DCLK output pin.

Tx—TRANSMIT DATA OUTPUT (PIN 13)

This line is an output for the B channel data received from the master. B channel 1 data is output on the first eight cycles of the BCLK output when EN1 is high. B channel 2 data is output on the next eight cycles of the BCLK, when EN2 is high. B channel data bits are clocked out on the rising edge of the BCLK output pin.

DCLK—D CHANNEL CLOCK OUTPUT (PIN 8)

This output is the transmit and receive data clock for both D channels. It starts upon demodulation of a burst from the master device. This signal is rising edge aligned with the EN1 and BCLK signals. After the demodulation of a burst, the DCLK line completes two cycles and then remains low until another burst from the master is demodulated. In this manner synchronization with the master is established and any clock slip between master and slave is absorbed each frame.

Rx—RECEIVE DATA INPUT (PIN 21)

This pin is an input for the B channel data. B channel 1 data is clocked in on the first eight falling edges of the BCLK output following the rising edge of the EN1 output. B channel 2 data is clocked in on the next eight falling edges of the BCLK following the rising edge of the EN2 output.

EN1—B CHANNEL 1 ENABLE OUTPUT (PIN 15)

This line is an 8 kHz enable signal for the input and output of the B channel 1 data. While EN1 is high, B channel 1 data is clocked out on the Tx pin on the first eight rising edges of the BCLK. During this same time B channel 1 input data is clocked in on the Rx pin on the first eight falling edges of the BCLK. The VD pin is also updated on the rising edge of the EN1 signal. EN1 serves as the slave device's 8 kHz frame reference signal.

EN2—B CHANNEL 2 ENABLE OUTPUT (PIN 14)

This pin is the logical inverse of the EN1 output and is used to signal the time slot for the input and output of data for the B channel 2 data.

BCLK—B CHANNEL DATA CLOCK OUTPUT (PIN 20)

This is a standard B series output which provides the data clock for the B channel data. This clock signal is 128 kHz and begins operating upon the successful demodulation of a burst from the master. At this time, EN1 goes high and BCLK starts toggling. BCLK remains active for 16 periods, at the end of which time it remains low until another burst is received from the master. In this manner synchronization between the master and slave is established and any clock slippage is absorbed each frame.

BACKGROUND

The MC145421 and the MC145425 ISDN UDLTs provide an economical means of sending and receiving two B channels (64 kbps each) of voice/data and two D channels (16 kbps each) of signal data in a two wire configuration at distances up to one kilometer. There are two ISDN UDLTs, master and slave. The master UDLT is compatible with existing and evolving PABX architectures. This device transmits 2B + 2D channels of data to the remote slave. At the remote end, the slave device presents a replica of the PBX backplane to the terminal devices.

These devices permit existing digital PBX architectures to remain unchanged and provide enhanced voice/data communication services throughout the PBX service area by simply replacing a subscriber's line card and telset.

All operations occur within the boundaries of an 8 kHz frame (125 μs). In the master, the frame sequence begins on the rising edge of MSI. In the slave, the frame begins after the demodulation of a burst from the master. The slave initializes its timing controls at this point to stay synchronized with the master.

During one 125 μs frame four main activities are performed:

1. Previously buffered 2B + 2D channel data is burst to the other end.
2. New 2B + 2D channel data is accepted for the next frame's transmission.
3. An incoming burst is demodulated and stored.
4. 2B + 2D channel data from the previous demodulated frame is output.

The bursts are 20 bits long, composed of two 8-bit B channels and two 2-bit D channels. Bursts are encoded using a modified DPSK method at 512 kHz. Since a single wire pair is used, half duplex operation is used. A 512 kHz burst is sent from end to end in a ping-pong fashion. This method provides apparent full duplex 160 kbps transmission of data at distances up to one kilometer.

GENERAL

The ISDN UDLT consists of a modulator, a demodulator, intermediate data registers, receive and transmit data registers, and sequencing and control logic. The Rx and Tx buffers interface digitally to the line card backplane signals, while the modulator and demodulator interface to the twisted pair transmission media. Intermediate data registers buffer data between these main components. The ISDN UDLT is intended to operate with a 5 volt power supply and can be driven by CMOS or TTL logic.

MASTER OPERATION

In the master, the rising edge of MSI initiates the 125 μs frame. B channel data is clocked into the Rx registers under control of TDC/RDC, RE1, and RE2. This data is combined with the D channel data clocked in on pins D1I and D2I by the DCLK. The resulting 20 bit packet is stored for the next frame transmission to the slave UDLT.

The burst output to the slave consists of the 2B + 2D data loaded during the previous frame. The burst received from the slave is demodulated and stored for outputting in the following frame.

B channel bits demodulated in the previous frame are output on the Tx pin under control of TDC/RDC, TE1, and TE2. Demodulated D channel bits are output on the D1O and D2O output pins. The indication of a valid burst demodulation is the VD output, which is updated at the start of every frame.

SLAVE OPERATION

In normal slave operation, the main synchronizing event is completion of demodulating a burst from the master UDLT. This action initializes the 125 μs frame boundary of the slave. During the slave frame, B channel data is loaded and stored under control of the BCLK, EN1, and EN2 outputs. D channel data is loaded at D1I and D2I under control of the DCLK output.

The demodulated burst from the master is separated into its D channel and B channel components and output on the D1O, D2O, and Tx pins. The return burst to the master consisting of previously loaded 2B + 2D data is transmitted eight bauds after the completion of demodulation of the master's burst. This provides a period for line transients to diminish.

The start of the slave frame initiates two cycles of the 16 kHz DCLK, and one cycle each of the 8 kHz EN1 and EN2 enables. After completing their cycles, these outputs remain low until another demodulation signals the start of a new slave frame. In this manner, clock slip between the master and slave UDLTs is absorbed each frame.

POWER-DOWN OPERATION

When $\overline{PD}$ is low in the master, the ISDN UDLT is powered down and only that circuitry necessary to demodulate incoming bursts is active. No transmissions to the slave occur during power down. If the master is receiving bursts from the slave, the VD pin will change state upon completion of the demodulation.

When the $\overline{PD}$ input pin is driven high, the master ISDN UDLT is powered up. In this mode, the master bursts to the slave every frame. B and D channel data can be loaded and unloaded and VD is updated on the MSI rising edge.

If no bursts are received by the master, whether powered up or not, the B channel data is unknown and the D channel bits will remain at their last known values.

The $\overline{PD}$ pin on the slave UDLT is bidirectional with a weak output driver that can be overdriven externally. When low — either externally or internally derived, the slave is powered down. No bursts to the master can be transmitted. EN1, EN2, BCLK, and DCLK outputs are inactive during power down except when TONE is high or a burst has been received from the master. B and D channel data can be loaded and unloaded, and VD is updated upon completion of demodulation of an incoming burst from the master. Input B and D channel data is not transmitted until the slave is powered up, in which case the first burst contains the most recently loaded data.

When the $\overline{PD}$ pin is high, the slave is powered up and transmits every frame. The data enables and clocks are output and data can be loaded and unloaded.

TIMEOUT OPERATION

Timeout is an operating state in both the UDLT master and slave devices. This state indicates that no incoming bursts have been demodulated, forcing the VD pin low. An internal counter is incremented for each frame that does not contain an incoming burst. The counter is reset upon demodulating a burst from the far end. Timeout can occur whether the device is powered up or down.

In the master, timeout begins on the rising edge of the third MSI following the last received burst. This is equivalent to two MSI frames. The VD output is forced low during timeout. The B channel output data will be unknown, but the D channel bits will remain at their last values. Successful demodulation of a burst from the slave will result in leaving the timeout state on the next rising MSI edge.

Timeout in the slave begins during the third frame without an incoming burst. The VD pin is forced low and the last D channel bits are saved. Normally, the slave timing is synchronized to the incoming master bursts, but in timeout, the slave operates from a free-running internal frame clock accompanied by BCLK, EN1, and EN2. These clocks are not generated during the two frames prior to entering timeout. If powered up during timeout, the slave will burst to the master on every other frame. This mode allows the terminal equipment to transmit its status to the master even though it is not receiving data. Demodulation of a burst from the master will cause the slave to exit the timeout mode.

When the $\overline{PD}$ pin is used as an output on the slave UDLT, timeout controls the pin. Timeout forces the $\overline{PD}$ output low to indicate that the device has powered itself down. In this case, the slave will not transmit to the master. However, when a valid burst is received, timeout ends and the $\overline{PD}$ pin is driven high to indicate power up. This feature allows the slave UDLT to self-power-up and down in demand-powered loop systems.

NOTE

The slave uses a free running clock during timeout. After a long period without a burst from the master, the timing between master and slave could be such that more than one burst will be needed to resync the two devices.

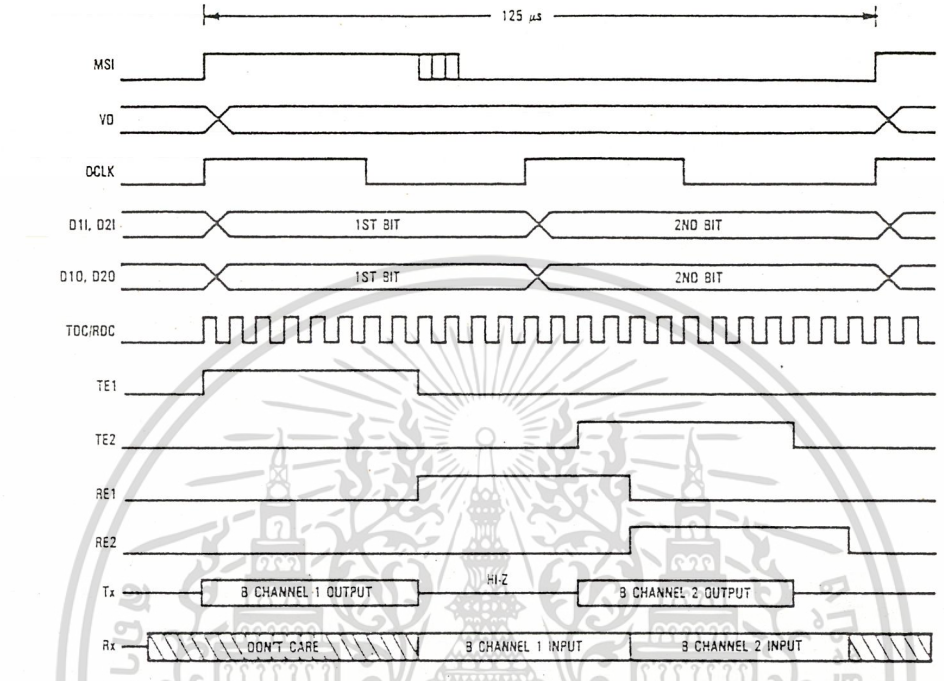


Figure 1. Typical MC145421 Master ISDN UDLT Timing

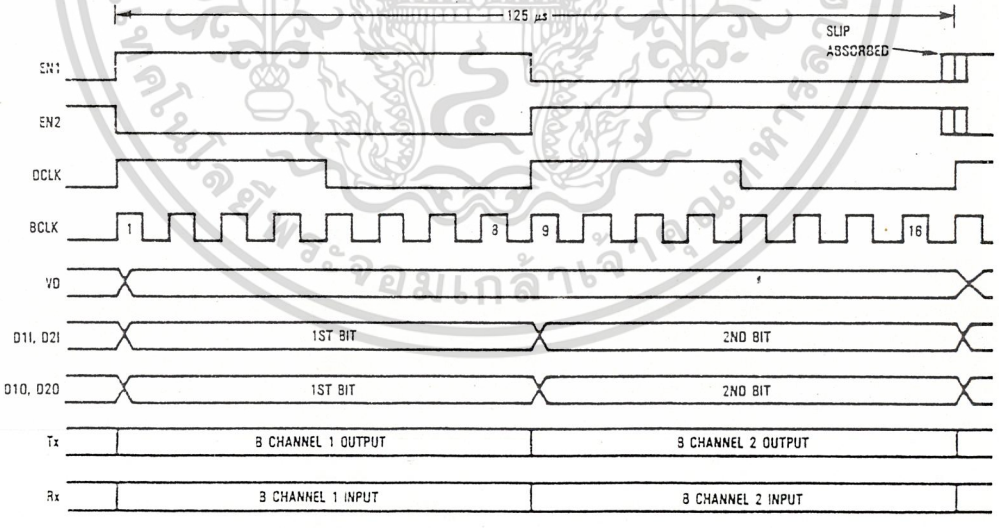
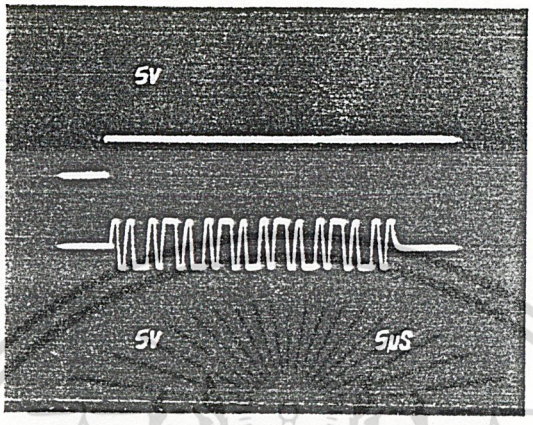
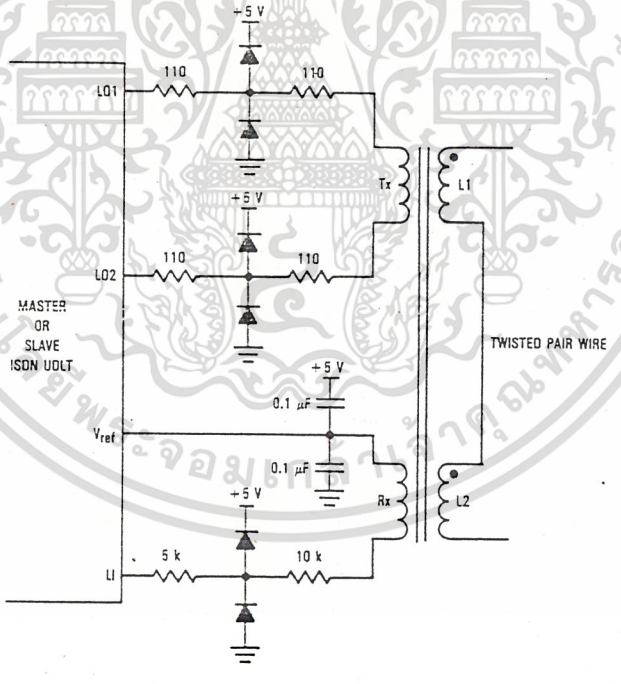


Figure 2. MC145425 Slave ISDN UDLT Timing



Top Trace: MSI
Bottom Trace: Outgoing burst measured at LI (with respect to V_{ref})

Figure 3. Master Burst



TRANSFORMER PARAMETERS
INDUCTANCE OF Tx WINDING: 1.75 mH
TURNS RATIO: Tx: L1 + L2 2:1
TURNS RATIO: Rx: L1 + L2 4:1

DIODES: 1N4148 OR EQUIVALENT

Figure 4. Interface to Twisted Pair Wire

2

SWITCHING CHARACTERISTICS (V_{DD} = 5 V, T_A = 0 to 70°C)

No.*	Parameter	Min	Max	Unit
Master Timing				
1	TDC/RDC Pulse Width High	110		ns
2	TDC/RDC Pulse Width Low	110		ns
3	MSI Rising Edge to TDC/RDC Falling Edge	90		ns
4	MSI Pulse Width	200		ns
5	MSI Rising Edge to First DCLK Falling Edge	90		ns
6	MSI Rising Edge to First D1O, D2O Bit Valid		–	ns
7	TE1, TE2 Rising Edge to TDC/RDC Falling Edge	110		ns
8	TDC/RDC Falling Edge to TE1, TE2 Rising Edge	20		ns
9	TE1, TE2 Rising Edge to First Tx Data Bit Valid		50	ns
10	TDC/RDC Rising Edge to Tx Data Bits 2 Through 8 Valid		50	ns
11	TE1, TE2 Falling Edge to Tx High-Impedance		70	ns
12	RE1, RE2 Rising Edge to TDC/RDC Falling Edge	110		ns
13	TDC/RDC Falling Edge to RE1, RE2 Rising Edge	20		ns
14	Rx Data Setup (Data Valid Before TDC/RDC Falling Edge)	50		ns
15	Rx Data Hold (Data Valid After TDC/RDC Falling Edge)	20		ns
16	RE1, RE2 Pulse Width	220		ns
17	DCLK Rising Edge to D1O, D2O Bit Valid		–	ns
18	D1I,D2I Data Setup (Data Valid Before DCLK Falling Edge)	50		ns
19	D1I, D2I Data Hold (Data Valid After DCLK Falling Edge)	20		ns
20	DCLK Pulse Width Low	110		ns
21	DCLK Pulse Width High	110		ns
22	MSI Rising Edge to VD Valid		–	ns
23	$\overline{PD}$, $\overline{LB}$ Setup ($\overline{PD}$, $\overline{LB}$ Valid Before MSI Rising Edge)	50		ns
24	$\overline{PD}$, $\overline{LB}$ Hold ($\overline{PD}$, $\overline{LB}$ Valid After MSI Rising Edge)	20		ns
Slave Timing				
25	BCLK Pulse Width High (CCI = 8.192 MHz)	3.66	4.15	µs
26	BCLK Pulse Width Low (CCI = 8.192 MHz)	3.66	4.15	µs
27	EN1 or EN2 Rising Edge to BCLK Rising Edge	75	175	ns
28	EN1 or EN2 Rising Edge to DCLK Rising Edge		± 50	ns
29	EN1 or EN2 Rising Edge to First Tx Data Bit Valid		50	ns
30	BCLK Rising Edge to Tx Data Bits 2 Through 8 Valid		– 75	ns
31	DCLK Pulse Width High (CCI = 8.192 MHz)	31.0	31.5	µs
32	DCLK Pulse Width Low (CCI = 8.192 MHz)	31.0	31.5	µs
33	DCLK Rising Edge to D1O, D2O Bits Valid		50	ns
34	Rx Setup (Rx Data Valid Before BCLK Falling Edge)	175		ns
35	Rx Hold (Rx Data Valid After BCLK Falling Edge)	20		ns
36	D1I, D2I Setup (D1I, D2I Valid Before DCLK Falling Edge)	50		ns
37	D1I, D2I Hold (D1I, D2I Valid After DCLK Falling Edge)	20		ns
38	EN1 Rising Edge to VD Valid		50	ns
SE Pin Timing				
39	$\overline{LB}$, $\overline{PD}$ Hold ($\overline{LB}$, $\overline{PD}$ Valid After SE Falling Edge)	20		ns
40	D1O, D2O, VD High-Impedance After SE Falling Edge		70	ns
41	D1O, D2O, VD Valid After SE Rising Edge)	60		ns
42	$\overline{LB}$, $\overline{PD}$ Setup ($\overline{LB}$, $\overline{PD}$ Valid Before SE Rising Edge)	50		ns

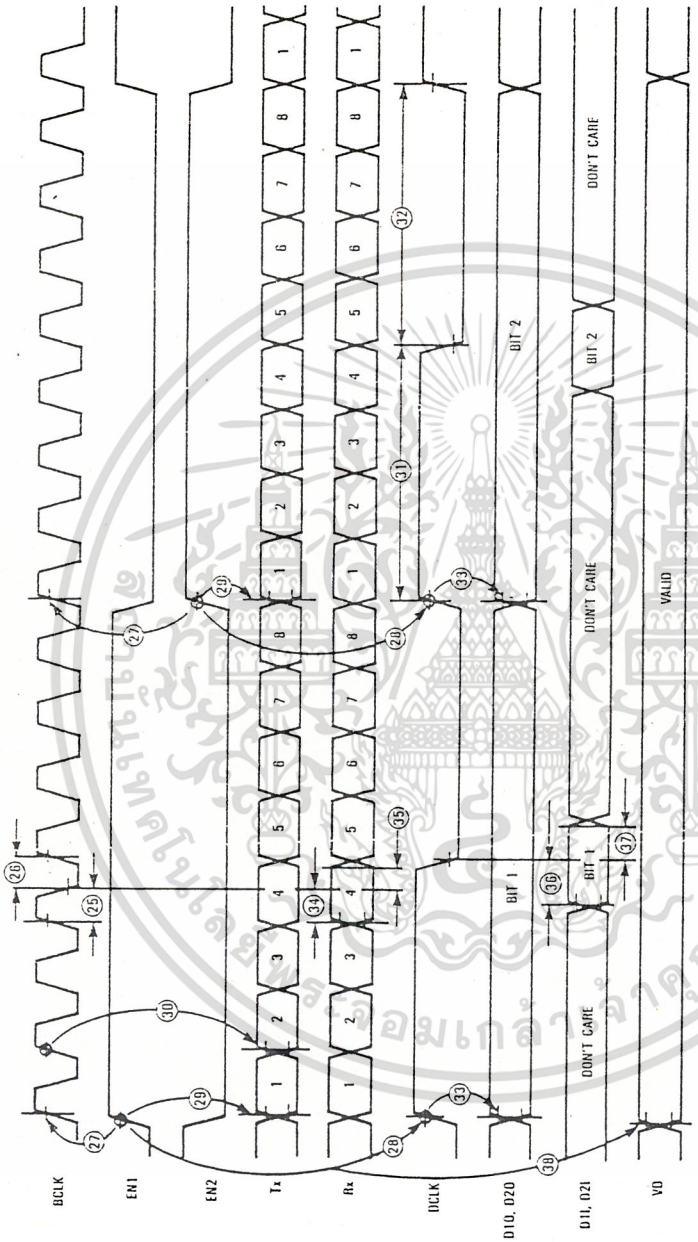
*See Switching Characteristics waveforms

MASTER SWITCHING CHARACTERISTICS

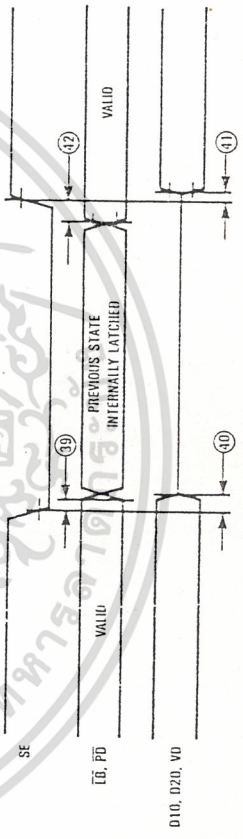
NOTE: All measurement thresholds are 30% or 70% of V_{DD}.

2

SLAVE SWITCHING CHARACTERISTICS



MASTER SE PIN TIMING



NOTE: All measurement thresholds are 30% or 70% of V_{DD}.

หนังสืออ้างอิง

1.ยืน ภู่วรวรรณ,รศ. .เทคโนโลยีฮาร์ดแวร์ IBM PC".ซีเอ็ดยูเคชั่น,กรุงเทพฯ,2531.

2.ไพศาล สงวนหมู่,ยืน ภู่วรวรรณ."การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เน็ตเวอร์ก", ซีเอ็ดยูเคชั่น, กรุงเทพฯ,2532.

3.วิสันต์ อาชาเดโชพล."การสื่อสารข้อมูลดิจิตอล",ฟิสิกส์เซ็นเตอร์,กรุงเทพฯ,2537.

4.พงษ์ระพี เตชพาหพงษ์."แอดวานซ์เอ็มเอสดอส", ซีเอ็ดยูเคชั่น,กรุงเทพฯ,2533.

5.ประกาศิต ชาติบุรุษ,อาทิตย์ จิตต์จุพานนท์. "โครงสร้างข้อมูลและอัลกอริทึม",ซีเอ็ดยูเคชั่น, กรุงเทพฯ,2533.

6.จรณิต แก้วกังวาล."การออกแบบและจัดการฐานข้อมูล",ซีเอ็ดยูเคชั่น,กรุงเทพฯ,2536.

7.Roger.C Palmer. "The barcode book",Helmer publishing.Inc, Englewood,USA,1989.

8. Rodnay Zaks. "The Z80 programming",Sybex.Inc,USA,1979.

9.Joe Compbell."C Programmer's guide to serial communication",Howard W. Sams&Company.Inc,11711 North College,Carmel,Indiana,1987.

10.Herbert Schildt,"C Power user's guide",Osborne Mc.Graw-Hill Berkery, California, USA,1989.

11.Herbert Schildt,"C The complete reference",Osborne Mc.Graw-Hill Berkery, California,USA,1989.

12.Ben Ezzell."Graphics programming in turbo C",Prentice-Hall.Inc,USA,1988.

13.Robert L.Kruse,Bruce P.Leung,Clovis L.Tondo."Data structures and program design in C" Prentice-Hall.Inc,USA,1991.

14.Ted.J Biggerstaff. "System software tools",Prentice-Hall.Inc, Newhamshire, USA,1986.

กิตติกรรมประกาศ

ขอกราบขอบพระคุณอาจารย์พรชัยยศ ศรีปัญญาพงศ์ และอาจารย์ขนิษฐา แซ่ตั้ง ที่ได้ให้ความรู้เพิ่มเติมและให้คำแนะนำอย่างมาก ช่วยให้สามารถแก้ปัญหาต่าง ๆ ที่เกิดขึ้นให้สำเร็จลุล่วงไปได้ ทำให้ปริญญานิพนธ์นี้เสร็จสมบูรณ์

ขอกราบขอบพระคุณอาจารย์ภาควิชาอิเลคทรอนิคส์ ทุกท่านที่ได้ประสิทธิ์ประสาทความรู้ให้แก่ผู้จัดทำ นำมาใช้ในการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณ พี่ ๆ ห้อง CT-LAB สำนักวิจัยและบริการคอมพิวเตอร์ที่เอื้อเฟื้อให้ยืมเครื่องพิมพ์ ขอบคุณแพร และเพื่อน ๆ ทุกคนที่ช่วยเหลือให้คำแนะนำที่ดี ๆ