ระบบส่งผ่านข้อมูลควบคุมเครือข่ายคอมพิวเตอร์

COMPUTER DATALINK CONTROL

โดย

นางสาวเตือนใจ ภัทรกุล

นายอรุณ สุวิทย์เรืองฤทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ ปีการศึกษา 2537

ภาควิชา คอมพิวเตอร์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบส่งผ่านข้อมูลควบคุมเครือข่ายคอมพิวเตอร์
COMPUTER DATALINK CONTROL

ผู้จัดทำ นางสาวเตือนใจ ภัทรกุล 35103224
นายอรุณ สุวิทย์เรืองฤทธิ์ 35103260

---------- (อาจารย์ที่ปรึกษา)

(อาจารย์สมศักดิ์ มิตะถา)

ระบบส่งผ่านข้อมูลควบคุมเครือข่ายคอมพิวเตอร์

นางสาวเตือนใจ ภัทรกุล 35103224
นายอรุณ สุวิทย์เรืองฤทธิ์ 35103260
อาจารย์ที่ปรึกษา
อาจารย์สมศักดิ์ มิตะถา
ปีการศึกษา 2537

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและใช้งานเครื่องคอมพิวเตอร์ เพื่อทำการควบคุมกรณีไมโครคอนโทรลเลอร์ผ่านพอร์ทสื่อสารอนุกรมโดยเป็นการติดต่อสื่อสารข้อมูลในรูปแบบบัสทำการจัดให้เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นสถานีแม่คอยตรวจเช็คสถานะของ ข้อมูลอินพุตและเอาท์พุตของสถานีลูกแล้วเอามาประมวลผล เพื่อควบคุมการใช้งาน ในแต่ละสถานีลูกที่ต่อในระบบ ในส่วนการแสดงผลของสถานีแม่จัดเป็น กราฟฟิกส์โหมด จะแสดงสถานะของทุกๆ สถานีลูกที่ต่ออยู่และที่สถานีลูกจะมีการต่ออุปกรณ์เซ็นเซอร์กับการต่อไปควบคุมอุปกรณ์ภายนอกการประยุกต์ใช้งานสามารถทำเป็นระบบรักษาความปลอดภัย ระบบควบคุมในโรงงานอุตสาหกรรม หรืออาจจะทำเป็นระบบบ้านอัตโนมัติก็ได้

COMPUTER DATALINK CONTROL

MISS. TOENJAI PHATARAKUL 35103224

MR.AROON SUWITRENGRIT 35103260

ADVISER

MR. SOMSAK MITATHA

1994

ABSTRACT

This thesis is about studying and how to used computer to control microcontroller device with pass by serial port. This communication is use bus topologhy. Computer is manage in server station. It will check status of input and output of node station. The server use then to process for control node station in system. In displayed form will use graphics mode to show status of every node station to connect in the system and node station toconnect the input senser device and control external output device. The application of this project is use in security system, the system control of industrial or the home automation .

สารบัญ

หน้า

บทที่ 1
บทนำ

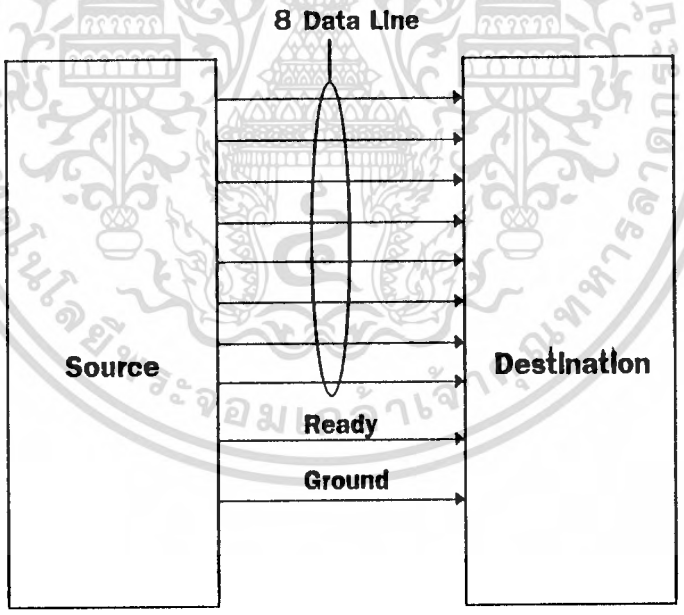
ปัจจุบันนี้ ความก้าวหน้า ทางเทคโนโลยีได้ เพิ่มมากขึ้นกว่า สมัยก่อนมากมีการนำอุปกรณ์เกี่ยวกับไมโครคอมพิวเตอร์ มาใช้ในชีวิตประจำวันกันมาก ไม่ว่าจะเป็นในลักษณะของรูปแบบเครื่องใช้ในสำนักงาน หรือเครื่องใช้ไฟฟ้าภายในบ้าน ล้วนแต่ประกอบด้วยอุปกรณ์ไฮเทคโนโลยีทั้งสิ้น การควบคุมอุปกรณ์ไฟฟ้าจำพวกนี้จึงมีความสะดวกสบายมากขึ้น ยกตัวอย่าง เช่น การใช้รีโมทคอนโทรล จึงเป็นเหตุให้เกิดโครงงานนี้ขึ้นมาสำหรับการควบคุมอุปกรณ์ไฟฟ้าได้อย่างอัตโนมัติตามแต่เราจะโปรแกรมค่าเอาไว้ซึ่ง ก็ต้องอาศัยคอมพิวเตอร์กับอุปกรณ์ไมโครคอมพิวเตอร์มา ต่อร่วมเพื่อให้ติดต่อสื่อสารส่งข้อมูลถึงกัน จึงทำให้เกิดการทำงานได้ตามต้องการ โดยรูปแบบของ การควบคุม เราสามารถจะประยุกต์ใช้งานได้อย่างกว้างขวางมากมาย ตัวอย่างเช่น ทำเป็นระบบบ้านอัตโนมัติ (Home Automation),ระบบรักษาความปลอดภัยต่าง ๆ (Security System),ระบบควบคุมภายในโรงงานอุตสาหกรรม (Industial Control System) เป็นต้น

บทที่ 2
ทฤษฎี

2.1 การสื่อสารข้อมูลคอมพิวเตอร์

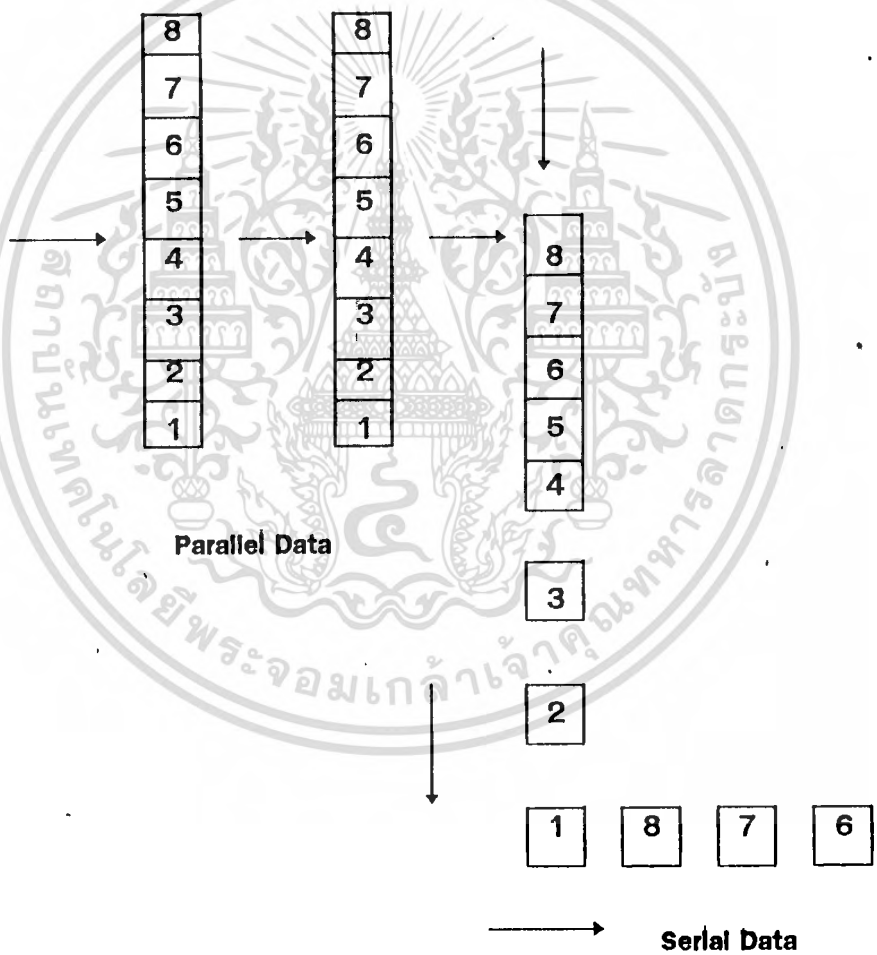
การสื่อสารข้อมูล ในระบบไมโครคอมพิวเตอร์ เป็นการส่งข้อมูล ติดต่อกันระหว่างตัวคอมพิวเตอร์และอุปกรณ์รอบข้างได้แก่ คีย์บอร์ด จอแสดงผล และอื่นๆ โดยสามารถแยกออกได้เป็น 2 ชนิด คือการส่งข้อมูลแบบขนาน และการส่งข้อมูลแบบอนุกรม

1. การส่งข้อมูลแบบขนาน ทำโดยการส่งข้อมูลออกมาทีละ 1 ไบต์ คือ 8 บิต จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ตัวกลางระหว่าง 2 เครื่อง จะต้องมีช่องทาง ให้ข้อมูลเดินทางอย่างน้อย 8 ช่อง โดยมากจะเป็นสายขนานให้กระแสไฟฟ้าวิ่ง เนื่องจากมีสัญญาณสูญเสียไปกับความต้านทานของสาย ระยะทางระหว่าง 2 เครื่อง จึงสั้นและไม่ควรเกิน 100 ฟุต และยังมีปัญหาของระดับกราวน์ทางไฟฟ้าที่จุดรับผิดไปจากจุดส่ง ทำให้เกิดการผิดพลาดของข้อมูล ซึ่งจะมีการแก้ไขโดยมีสายควบคุมเพิ่มไปด้วย เช่น สายควบคุมการโต้ตอบ (Ready)



รูปที่ 1 แสดงการส่งข้อมูลแบบขนาน

2. การส่งข้อมูลแบบอนุกรมข้อมูลจะถูกส่งออกมาทีละบิต ระหว่างจุดส่งและจุดรับ โดยการส่งข้อมูลแบบนี้จะช้ากว่าแบบขนาน แต่ที่ดีกว่าคือมีช่องทางการสื่อสารเพียงช่องเดียว และในกรณีการส่งระยะทางไกลๆ จะสามารถใช้ระบบการสื่อสารทางโทรศัพท์มาช่วยสนับสนุนได้ การสื่อสารข้อมูลแบบขนานจะถูกเปลี่ยนให้เป็นแบบอนุกรม และส่งผ่านตัวกลางจนถึงตัวรับ ตัวรับจะเปลี่ยนข้อมูลแบบอนุกรมไปเป็นแบบขนานอีกที



รูปที่ 2 แสดงการส่งข้อมูลแบบอนุกรม

รูปแบบการสื่อสารข้อมูล

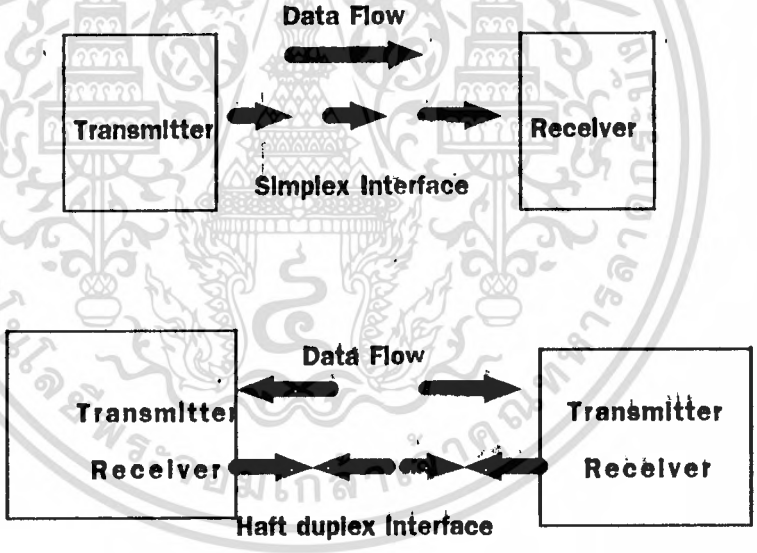
การติดต่อแบบอนุกรมสามารถแบ่งได้ 3 แบบคือ

1. แบบซิมเพล็กซ์ (Simplex) ข้อมูลส่งได้ในทิศทางเดียวเท่านั้น

2. แบบฮาล์ฟดูเพล็กซ์ (Half Duplex) ข้อมูลสามารถส่งได้ทั้งสองสถานี แต่จะผลัดกันรับส่ง จะส่งและรับพร้อมกันไม่ได้

3. แบบฟูลดูเพล็กซ์ (Fulln Duplex) ทั้งสองสถานีสามารถรับส่งได้ในเวลาเดียวกัน

ความเร็วของการถ่ายโอนข้อมูลแบบอนุกรม มีหน่วยเป็นบิตต่อวินาที (bps) และหน่วยที่แสดงถึงการเปลี่ยนแปลงข้อมูลของสัญญาณใน 1 วินาที เรียกว่า อัตราบอด (buad rate) สำหรับในโครงงานนี้อัตราบิตเท่ากับอัตราบอด



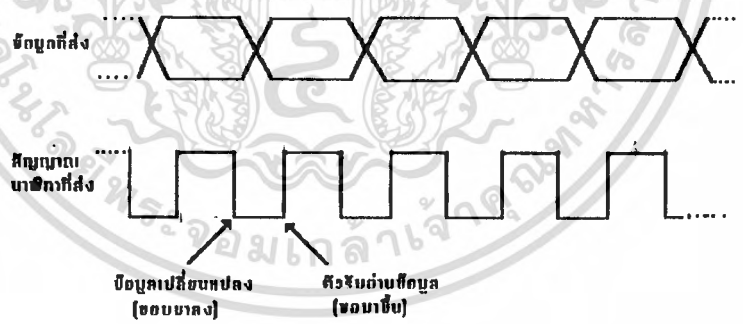
รูปที่ 3 แสดงรูปแบบการสื่อสารข้อมูล

2.2 ชนิดการสื่อสารข้อมูลแบบอนุกรม

การสื่อสารข้อมูลแบบอนุกรมสามารถแบ่งออกเป็น 2 อย่างคือ แบบซิงโครนัส และอะซิงโครนัส

2.2.1 การติดต่อสื่อสารแบบซิงโครนัส

ลักษณะการติดต่อแบบซิงโครนัสสัญญาณนาฬิกาจะเป็นตัวกำหนดตำแหน่งของแต่ละข้อมูลที่ส่งมาโดยตัวส่งจะเปลี่ยนแปลงข้อมูลที่ตำแหน่งขอบขาลง ของสัญญาณนาฬิกาซึ่งข้อมูลจะยังคงสภาวะอยู่ไม่เปลี่ยนแปลง ลักษณะการส่งรับจะเป็น ดังรูปที่1การติดต่อสื่อสารแบบซิงโครนัสจะได้เปรียบหรือ มีข้อดีก็ตรงที่ สามารถส่งข้อมูลได้เร็ว และ มีความเที่ยงตรงสูงกว่าแบบอะซิงโครนัสอย่างไรก็ตามการติดต่อสื่อสารแบบนี้ก็มีความยุ่งยากและละเอียดอ่อนกว่าแบบอะซิงโครนัส การติดต่อสื่อสารแบบซิงโครนัสนี้ส่วนใหญ่จะใช้สำหรับ มินิคอมพิวเตอร์และคอมพิวเตอร์ระบบเมนเฟรม เพราะระบบต่างๆ ต้องใช้ความเร็วในการติดต่อ สื่อสารและต้องการความถูกต้องสูง



รูปที่4 แสดงถึงข้อมูลที่ส่งและข้อมูลที่รับซึ่งสัมพันธ์กับสัญญาณนาฬิกา

2.2.2 การติดต่อสื่อสารแบบอะซิงโครนัส

การติดต่อสื่อสารแบบนี้เป็นการติดต่อที่ง่าย และมีความซับซ้อนน้อยนิยมใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป การติดต่อแบบอะซิงโครนัสไม่จำเป็นต้องใช้สัญญาณ นาฬิกาเข้ามาเกี่ยวข้องด้วยเลย

การติดต่อสื่อสารแบบอะซิงโครนัสนี้ตัวรับและตัวส่งจะต้องกำหนดความเร็วในการรับและส่ง หรือบ็อดเรด (baud rate = จำนวนบิตข้อมูลที่ส่งใน 1 วินาที) ให้เท่ากัน ข้อมูลจะถูกส่งทีละชุดคือ จะมีบิตเริ่ม (start bit) และบิตจบ (stop bit) เป็นตัว บ่งบอก ซึ่งบิตเริ่ม จะมีสภาวะลอจิกเป็น "ต่ำ" (low) ตามด้วยบิตข้อมูล อาจเป็น 7 บิต หรือ 9 บิต ก็แล้วแต่ ต่อจากนั้นก็จะตามด้วยพาริตี้บิท และปิดท้ายด้วยบิตจบ ซึ่งมีสภาวะ ลอจิกเป็น "สูง" (hight) มี ความกว้างเท่ากับ 1,1.5 หรือ 2 เท่าของบิตข้อมูล

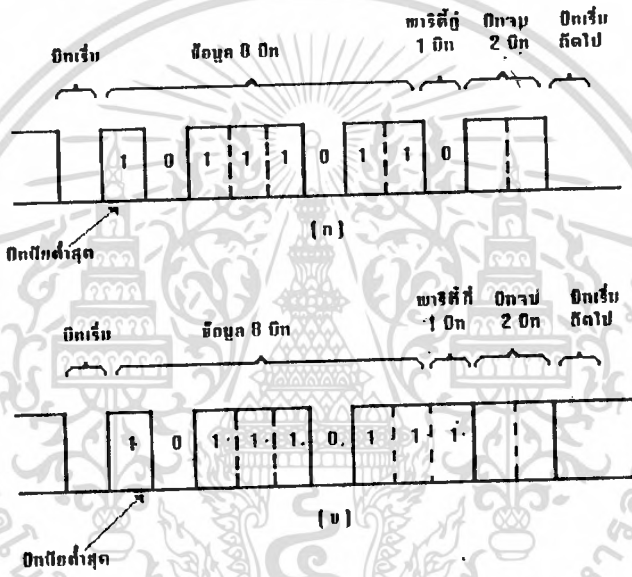
จุดมุ่งหมายของการมีบิตจบก็คือ

1.ใช้ตรวจสอบข้อมูลที่รับได้ว่าถูกต้องหรือไม่ ถ้าไม่มีการพบบิตจบหลังจากสิ้นสุดการรับ ข้อมูล แฟลกจะแสดงข้อผิดพลาดให้ทราบ

2. เพื่อให้ทราบได้ว่าข้อมูลชุดก่อนได้สิ้นสุด และพร้อมที่จะส่งข้อมูลชุดใหม่ได้

รูปที่ 5 ลักษณะสัญญาณของการรับส่งข้อมูลแบบอะซิงโครนัส 1 ชุดข้อมูล

จากรูปที่ 5 เป็นรูปแสดงการส่งข้อมูลแบบอะซิงโครนัส จะเห็นว่ามีบิตเริ่ม บิตข้อมูล พาริตี้บิต และบิตจบ ยกตัวอย่างการส่งข้อมูล 8 บิต มีข้อมูลเป็น 1011101" การส่งข้อมูลจะเป็นดังรูปที่ 6



รูปที่ 6 ตัวอย่างการส่งข้อมูล 8 บิต "11011101" แบบพาริตี้คู่ (ก) และพาริตี้คี่ (ข)

จากรูปที่ 6 จะเป็นการส่งข้อมูลแบบมีพาริตี้คู่ และแบบมีพาริตี้คี่ ซึ่งความสำคัญของการส่งข้อ มูลแบบมีพาริตี้คือเพื่อใช้ตรวจสอบว่าข้อมูลที่รับได้นั้นถูกต้องตามที่ส่งมาหรือไม่ การติดต่อสื่อสารแบบมี้มักใช้ชิพไอซีที่ทำหน้าที่เป็นตัวรับข้อมูลแบบอะซิงโครนัส โดยมีชื่อ ว่า UART(Universal Asynchronouse Reciver/Transmitter) ซึ่งใช้ ความถี่ของสัญญาณ นาฬิกา เป็นตัวกำหนดอัตราการรับข้อมูล โดยมีค่าเท่ากับ 16 * ค่าบ็อดเรตเช่น มีค่าเป็น 153,600 Hz สำหรับบ็อดเรต 9600 บิตต่อวินาที โดยปกติถ้ายัง ไม่มีการส่งข้อมูลใด ๆ ใน สายส่งสัญญาณจะเป็นลอจิก "สูง" อยู่ เมื่อมีการเริ่มส่งข้อมูลสภาวะของสายสัญญาณจะเปลี่ยนเป็น "ต่ำ" แสดงถึงบิตเริ่ม ตัวรับ UART จะหน่วงเวลาโดยการนับสัญญาณนาฬิกาไป 8 ลูกคลื่น เพื่อเช็คดูอีกครั้ง ให้แน่ใจ (ที่จุด นี้จะเป็นจุดกึ่งกลางของบิต เพราะ 1 บิต จะกว้าง 16 ลูกคลื่นของสัญญาณนาฬิกา) ถ้าพบว่า สภาวะลอจิกเป็น "สูง" แสดงว่า เมื่อสักครู่เป็นสัญญาณรบกวน ก็จะ ไม่สนใจ แต่ถ้าพบว่าลอจิกยัง คงเป็น "ต่ำ" มันก็จะเริ่มต้นรับข้อมูลโดยการนับสัญญาณนาฬิกาไปทีละ 16 ลูก แล้วอ่านข้อมูลเข้า ไปเก็บในชิฟต์รีจิสเตอร์ทีละบิต

หลังจากข้อมูลทั้งหมดและพาริตี้บิตถูกรับเข้ามาหมดแล้ว บิตต่อไปที่ตามมาก็ควรจะเป็น บิตจบ ถ้าเป็นลอจิก "ต่ำ" UART จะระบุว่าไม่สามารถรับข้อมูลได้อย่างถูกต้อง และแสดงข้อ ผิดพลาดขึ้นมา เพราะฉะนั้นบ็อดเรตที่เที่ยงตรงทั้งตัวรับและตัวส่งจึงจำ เป็นมาก เพราะถ้า ผิดพลาดแล้ว จะทำให้การส่งข้อมูลและการรับข้อมูลผิดพลาดทันที

เมื่อตัวรับอ่านข้อมูลได้ 8 หรือ 9 บิต (รวมกับพาริตี้บิต) ได้อย่างถูกต้องบิตต่อมาก็จะเป็นบิทจบ ซึ่งมีลอจิก "สูง" มีช่วงเวลาการคงสภาวะเป็น 1,1.5 หรือ 2 ช่วง เวลาแต่ละบิต หากถูกต้องหมดทุกอย่าง UART จะทำการตรวจสอบข้อมูลที่รับมาว่ามีพาริตี้ บิตถูกต้องหรือเปล่า โดยจะตรวจสอบว่าเป็นพาริตี้คู่หรือคี่ ถ้าพาริตี้ที่ตรวจพบไม่ถูกต้อง ตามที่กำหนด แฟลกจะบอกข้อ ผิดพลาดทันทีอีกเช่นกัน UART ส่วนมากสามารถรับข้อมูลเข้ามาทีละ 2 ชุด โดยชุดแรกเก็บไว้ในชิฟต์รีจิสเตอร์และที่เหลือเก็บไว้ในโฮลดิ้งรีจิสเตอร์ (holding reglster) ข้อมูลที่รับได้จะเลื่อนจากชิฟต์รีจิสเตอร์ไปยังโฮลดิ้งรีจิสเตอร์เพื่อรอให้โปรเซสเซอร์อ่านข้อมูลออกไป เมื่อ เลื่อนข้อมูลไป แล้วชิฟต์รีจิสเตอร์ก็จะรับข้อมูลใหม่เข้ามาเพื่อเก็บและเตรียมส่งให้กับโฮลดิ้งรีจิสเตอร์ต่อไป

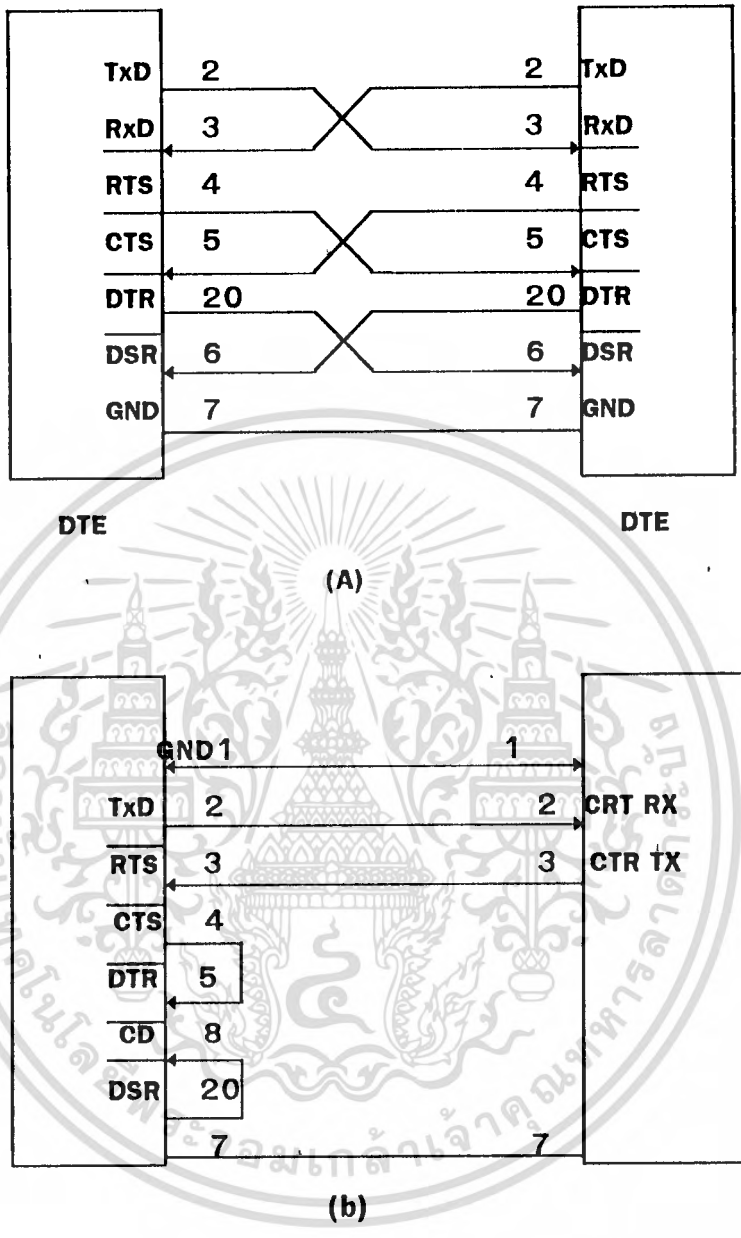
อย่างไรก็ตาม ถ้าโปรเซสเซอร์ไม่อ่านข้อมูลที่อยู่ในโฮลดิ้งรีจิสเตอร์ก่อนที่ข้อมูลตัวใหม่จะถูกส่งเข้ามา ข้อมูลเก่าก็จะถูกทับโดยข้อมูลตัวใหม่ ทำให้ข้อมูลไม่ครบถ้วน เกิดข้อผิดพลาด ในการรับส่งข้อมูล ในปัจจุบันได้มีการพัฒนา UART ที่ดีกว่าเดิม ซึ่งได้รับการออกแบบเพื่อลดปัญหาพวกนี้ ชิพไอซีรุ่นใหม่นี้จะประกอบด้วย การเก็บข้อมูลแบบ FIFO (เข้าก่อนออกก่อน) หมายความว่า จะ เก็บข้อมูลเป็นชุด ชุดละ 16 ตัวอักษร พอเต็มจึงจะมีการทับของข้อมูล ช่วยให้โปรเซสเซอร์มี เวลาในการอ่านข้อมูลได้นานขึ้น

สำหรับวิธีอื่นที่จะป้องกันการสูญหายของข้อมูลก็มีดังนี้

1.ควบคุมอัตราเร็วการส่งข้อมูล คือ ตัวรับข้อมูลสามารถที่จะสั่งให้ตัวส่งหยุด การส่งข้อมูลชั่วคราว เมื่อรับข้อมูลไม่ทัน

2.ควบคุมทางฮาร์ดแวร์ระบบนี้ควบคุมได้โดยใช้อุปกรณ์ทางอิเล็กทรอนิกส์โดยใช้สัญญาณที่มีอยู่ในพอร์ท RS-232 ซึ่งเป็นพอร์ทสื่อสาร เพื่อบอกความพร้อมของการรับข้อมูล ซึ่งมีชื่อ เรียกของสายสัญญาณว่าRTS (Request To Send) และ CTS (Clear To Send) ถ้าเป็น การติดต่อระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง RTS ของเครื่องที่ 1 จะต่อกับ CTS อีกเครื่อง หนึ่งดูได้ จากรูปที่ 7 ลักษณะการโต้ตอบกับแบบนี้เรียกว่า handshaking

เมื่อคอมพิวเตอร์ตัวรับพร้อมที่จะรับข้อมูลจะแจ้งสัญญาณไปที่สาย RTS คอมพิวเตอร์ตัวส่งจะตอบกลับที่สายสัญญาณ CTS นอกจากนี้ยังมีวิธี hand shaking โดยใช้ซอฟต์แวร์เรียกว่า XON/XOFF hand shaking ซึ่งมีวิธีการคล้ายๆ กัน



รูปที่ 7 ลักษณะการเชื่อมต่อสัญญาณโต้ตอบเพื่อป้องกันข้อผิดพลาดในการรับส่งข้อมูล

2.3 ระบบอินเตอร์เฟส RS-232 และ RS-422

2.3.1 ระบบอินเตอร์เฟส RS-232

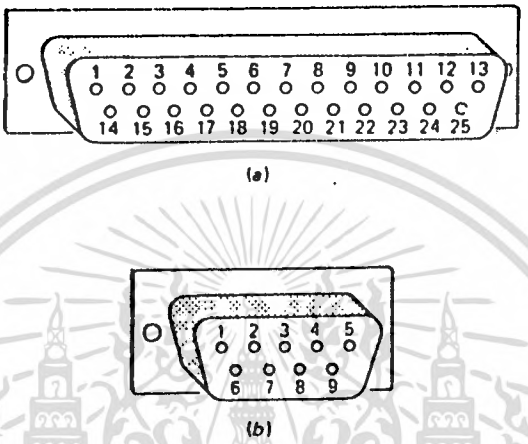
ผู้ที่กำหนดมาตรฐาน RS-232 ขึ้นเมื่อกว่า 20 ปีมาแล้วก็คือ สถาบันแห่งหนึ่งในสหรัฐอเมริกา ซึ่งเป็นที่ยอมรับกันมีชื่อว่า Electronics Industrials Association หรือที่ เรียกสั้นๆ ว่า EIA โดยได้กำหนดเป็นมาตรฐานสำหรับการเชื่อมต่อ คอมพิวเตอร์ (การอินเตอร์เฟส) ซึ่งมาตรฐานนี้เป็นการส่งข้อมูลแบบอะซิงโครนัส มาตรฐานนี้เป็นที่ยอมรับของบริษัทผู้ ผลิตคอมพิวเตอร์ส่วนใหญ่และมีความนิยมใช้กันอย่างแพร่ หลายในปัจจุบัน

ระบบ RS-232 ได้มีการปรับปรุงแก้ไขในปี 2529 เป็น EIA 232-D1986 (โดยทั่วไปเรียกว่า RS-232D) มาตรฐานนี้จะใช้ขั้วต่อขนาด 25ขาแบบ D(D subminiature connector) สำหรับการส่งข้อมูลระหว่างระบบ (Rs-232c ไม่ได้เจาะจงชนิดของคอนเน็กเตอร์ที่ใช้) ในระบบนี้สัญญาณจะตรงข้ามความเป็นจริงคือ ลอจิก "สูง" มีระดับแรงดัน -3 โวลต์ถึง -25 โวลต์ แต่ส่วนใหญ่จะใช้ -12 โวลต์ และลอจิก"ต่ำ" มีระดับแรงดันตั้งแต่ +3 ถึง +25 โวลต์ ส่วนใหญ่ใช้ +12 โวลตการที่ได้กำหนด RS232เป็นมาตรฐานในการสื่อสารข้อมูลระหว่าง คอมพิวเตอร์ทำให้มีการติดต่อสื่อสารข้อมูลกันอย่างแพร่หลายระบบ RS-232 นี้มีความเร็วในการ ส่งข้อมูล 20,000 บิตต่อวินาที (20kbps) และระยะส่งไม่เกิน 50 ฟุต ตามคุณสมบัติ อย่างไรก็ ตามสามารถยอมให้ส่งให้ไกลกว่านี้ได้ โดยจะต้องคำนึงถึงค่าอิมพีแดนซ์ในสาย (Xc) โดยให้มีค่า น้อยที่สุด

จะเห็นได้ว่าการส่งข้อมูลแบบ RS-232 ค่อนข้างจะมีข้อจำกัดคือส่งได้แค่ 20,000 บิตต่อวินาที ที่ 50 ฟุต ถ้าระยะส่งไกลกว่านี้ ความเร็วในการส่งจะลดลงและมีสัญญาณรบกวน มากขึ้นเนื่องจากสายส่งสัญญาณเป็นแบบอ้างอิงกับกราวด์ไม่ใช่ระบบสมดุล (unbalance)

2.3.1.1 การกำหนดจุดต่อของ RS-232C

ในทางฟิสิกส์แล้วมาตรฐานของ RS-232C กำหนดข้อต่อแบบ DB-25 แต่ละขาของข้อต่อกำหนดไว้ดังแสดงในรูปที่ 8 อย่างไรก็ตามผู้ผลิตไมโครคอมพิวเตอร์อาจจะ ใช้ข้อต่อชนิดอื่นที่นอกเหนือไปจาก DB-25 ยกตัวอย่างเช่น Fujitsu F-8, IBM AT, IBM JR เป็นต้น ตัวเมียของข้อต่อควรจะอยู่ที่ตัวโมเด็มขณะที่ตัวผู้ควรจะอยู่ที่AsynchronousCommunication adapter หรือที่ตัวไมโครคอมพิวเตอร์เอง อย่างไร ก็ตามผู้ผลิตหลายรายไม่ได้ทำตามกฎที่ว่านี้



รูปที่ 8 แสดงการกำหนดข้อต่อของ RS-232C

สัญญาณต่าง ๆ ถูกมอบหมายให้ทำหน้าที่ดังนี้

- **Trannsmit Data (TD ขาที่ 2)**

เป็นสัญญาณที่ส่งออกจาก DTE หรือตัวไมโครคอมพิวเตอร์ ไปยังโมเด็มหรือต่อเข้า โดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ " I " หรือเทียบเท่ากับสต็อบบิท

- **Recleve data (RD ขาที่ 3)**

เป็นการส่งของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพทางลอจิกเป็น "I"

- **Request to Send (RTS ขาที่ 4)**

ใช้สำหรับส่งสัญญาณไปยังโมเด็ม หรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมา
ทางขา 2 สัญญาณนี้ใช้คู่กับ CTS หรือ Clear To Send อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็ส่งสัญ ญาณออกไปที่สาย CTS

- **Clear to Send (CTS ขาที่ 5)**

ดังอธิบายไว้ใน RST เมื่อสัญญาณนี้อยู่ในสถานะออฟ (Negative Voltage หรือลอจิก " I ") หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมจะรับข้อมูลแล้ว

- **Data Set Ready (DSR ขาที่ 6)**

เมื่อสัญญาณสายนี้อยู่ในสถานะออน (หรือลอจิก 0) เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่าโมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่ง ได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่าต่อโทรศัพท์ได้สำเร็จแล้ว

- **Signal Ground (SG ขาที่ 7)**

SG ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุกๆ สายของสัญญาณ จะมีแรงดันเป็น "0 " เมื่อเทียบกับสัญญาณตัวอื่น

- **Carier Detect (CD ขาที่ 8)**

โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน (ลอจิก " 0 ") ไปบอกไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุด LED บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้ว ไฟ LED จะอยู่บนหน้าปัดของโมเด็มเอง

- **Data Terminal Ready (DTR ขาที่ 20)**

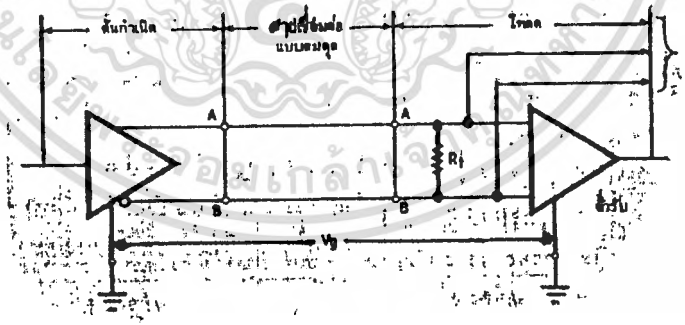
คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน (ลอจิก" 0 ") เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD, USR และ CTS) ให้ คอมพิว-เตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

- **Ring Indicator (RI ขาที่ 22)**

สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบรับอัตโนมัติ (Auto-answer) สัญญาณนี้จะ "ออน"เมื่อมีสัญญาณกระดิ่งมา และ "ออฟ" ระหว่างเสียงดังของกระดิ่ง ระบบอินเตอร์เฟส RS-422

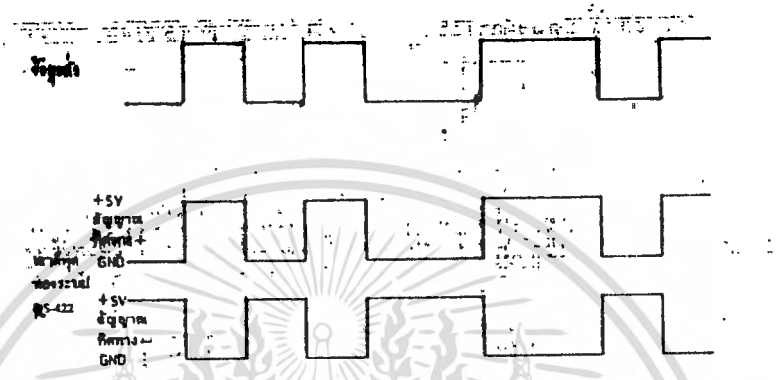
2.3.2 ระบบอินเตอร์เฟส RS-422

เมื่อไม่นานมานี้ EIA ก็ได้ออกแบบมาตรฐานชนิดใหม่ โดยพัฒนาจากของเดิม เพื่อมาใช้แทน RS-232 ในส่วนที่ขีดจำกัดของ RS-232 มี โดยได้ออกแบบให้ RS-422 มีความสามารถ ในการส่งข้อมูลได้ไกลๆ คือ สามารถส่งได้ไกลถึง 4,000 ฟุต และความเร็วในการส่งข้อมูลสูง สุดถึง 10 ล้านบิตต่อวินาที (10 Mbps) (แบบไม่เป็นทางการ) ความสามารถสูงเช่นนี้เพราะใช้ การแยกสายสัญญาณเป็น 2 เส้นต่อ 1 สัญญาณ โดยไม่ใช้กราวด์ร่วมเรียกว่า เป็นระบบสมดุล (balance) อีกทั้งยังใช้ระดับสัญญาณที่ +5 โวลต์ จึงไม่จำเป็นต้องใช้แหล่งจ่ายไฟเพิ่มเติมเหมือน RS-232อย่างไรก็ตาม ที่ความเร็ว90 kbps ยอมให้ส่งได้ไกลเต็มที่ 4,000 ฟุต ความเร็ว 2 Mbps ที่ระยะทาง 200 ฟุต และความเร็ว 10 Mbps ที่ 35 ฟุต ใน 1 เส้นสัญญาณ สายทั้ง 2 เส้นจะตีเกลียวไปด้วยกัน ทำให้มีอิมพีแดนซ์เท่ากันและ กระแสในสายจะเท่ากัน แต่ทิศทางเป็นตรงข้าม สัญญาณรบกวนต่างๆ จะหักล้างกันหมดสายกราวด์นั้น จะใช้เพียงการอ้างอิงระดับแรงดันเท่านั้นโดยไม่ได้เป็นทางผ่านของสัญญาณแต่อย่างใด ลักษณะการเชื่อมต่อแบบนี้แสดงในรูปที่ 9



รูปที่ 9 การเชื่อมต่อสัญญาณแบบสมดุลที่ใช้ในระบบ RS-422

ในการส่งสัญญาณแบบนี้นั้น ขณะที่เส้นหนึ่งมีแรงดัน +5 โวลต์ อีกเส้นที่คู่กันจะเป็น 0 โวลต์ และในทางตรงข้ามเมื่อสายหนึ่งเป็น 0 โวลต์ อีกสายจะเป็น +5 โวลต์ดังนั้นการสวิงแรงดันที่ปรากฏระหว่าง 2 สาย จะมีค่าเป็น 10 โวลต์ ดังแสดงในรูปที่ 10



รูปที่10 เปรียบเทียบข้อมูลที่ส่งกับสัญญาณเอาต์พุตของระบบ RS-422

สัญญาณ RS-422 จะแบ่งเป็น 1 คู่สาย ต่อ 1 สัญญาณ เพื่อลดค่าอิมพีแดนซ์ในสาย ทำให้ผลของความไว ต่อสัญญาณรบกวนมีน้อยมาก ถึงแม้ RS-422 จะมีคุณสมบัติที่ดีกว่า RS-232แต่ก็ ยังมีความนิยมน้อย ปัจจุบันกำลังเป็นที่นิยมใช้กันในโรงงานอุตสาหกรรมที่เป็นแบบอัตโนมัติ และ ต้องการความเชื่อถือได้สูง

2.3.2.1 การส่งข้อมูลแบบบาลานซ์ RS-422

การส่งข้อมูลแบบบาลานซ์ (balance) เป็นการรับส่งข้อมูลตามมาตรฐานของ EIA มีคุณสมบัติเหนือกว่าส่งข้อมูลแบบซิงเกิ้ลเอ็นด์ (single ended) เช่น RS-232 มีความสามารถในการรับส่งสัญญาณได้ในอัตราสูง 100 Kbps- 10 Mbps สามารถจำแนกข้อดีได้ดังนี้

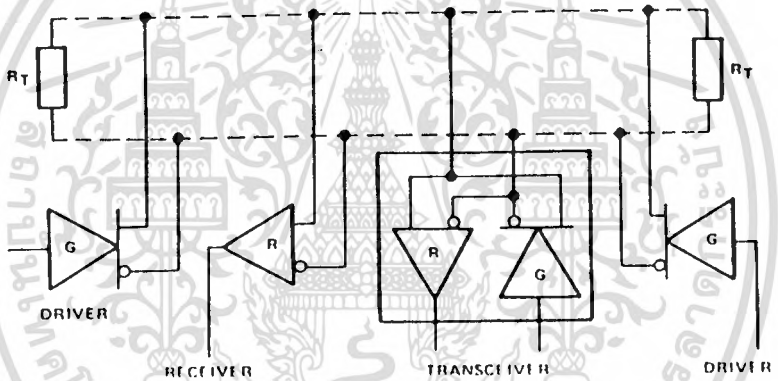
1. สามารถรับส่งสัญญาณได้ในระยะทางไกลถึง 4000 ft
2. ลดผลกระทบจากสัญญาณรบกวนได้สูงกว่าแบบซิงเกิ้ลเอ็นด์

ระบบของการส่งสัญญาณตามมาตรฐาน RS-422 นี้จะประกอบด้วย

1. ตัวขับสัญญาณ (line driver)

2. สายส่ง (transmission line) โดยทั่วไปคือ twisted pair

3. โหลดซึ่งประกอบด้วยตัวรับสัญญาณ (receiver) และตัวต้านทานปลายสาย (terminate resister)



รูปที่ 11 แสดงการส่งสัญญาณตามมาตรฐาน RS-422

PARAMETER	RS-232-C	RS-423A	RS-422-A	RS-485
Mode of operation	Single-ended	Single-ended	Differential	Differential
Number of drivers and Receivers allowed	1 Driver 1 Receiver	1 Driver 10 Receiver	1 Driver 10 Receiver	32 Driver 32 Receiver
Maximum cable length (ft)	50	4000	4000	4000
Maximum data rate bit per second	20K	100K	10M	10M
Maximum common mode voltage	±25 V	± 6 V	6 V 0.25 V	12 V 7 V
Driver output	± 5 V min ±15 V max	± 36 V min ± 6.0 V max	± 2 V min	± 15 V min
Driver load	3 KΩ to KΩ	450Ω min	100Ωmin	60Ωmin
Driver slew rate	30 V µs max	Externally controlled	NA	NA
Driver output short circuit current limit	500 mA to Vcc or GRD	150 mA to GRD	150 mA to GRD	150 mA to GRD 250 mA to 8V or 12 V
Driver output, poweron resistance (High Z statel),power on	NA 300Ω	NA 60KΩ	NA 60KΩ	120 KΩ 120 KΩ
Receiver Input resistance Ω	3k Ω to KΩ	4KΩ	4KΩ	12KΩ
Recever sensitivity	-3V	-200 V	-200V	-200V

ตารางที่ 1 แสดงคุณสมบัติการส่งสัญญาณตามมาตรฐานของ EIA

วงจรแปลงระบบอินเตอร์เฟสจาก RS-232 เป็น RS-422 มีประโยชน์มากในกรณีที่ต้องมีการติดต่อระหว่างอุปกรณ์ในงานอุตสาหกรรม แต่คอมพิวเตอร์และอุปกรณ์ที่ใช้มีแต่ระบบ RS- 232 ให้ซึ่งจะเห็นคุณประโยชน์ได้ชัดเจนหากต้องมีการเดินสายเชื่อมต่อไกลๆหรือมีเครื่องจักรที่ก่อ ให้เกิดสัญญาณรบกวนทางไฟฟ้าสูงมากๆ

2.4 การใช้งานพอร์ทอนุกรมด้วยภาษาซีมาตรฐาน RS-232

พอร์ทแบบอนุกรมส่วนใหญ่ จะมีรูปร่างขึ้นอยู่กับมาตรฐานของ RS-232 คือ มีขา 25 ขาที่คอนเน็กเตอร์แต่ละปลายสายส่ง (แต่ใน IBM AT มีเพียง 9 ขา) แต่ถึงแม้ว่าจะมีขาจำนวนเท่ากัน แต่พอร์ทส่วนใหญ่จะมีสัญญาณที่ไม่เหมือนสัญญาณของ RS-232 ทั้งหมด เพราะว่าบางสัญญาณไม่จำเป็นต้องใช้ สัญญาณพื้นฐานของ RS-232 แสดงดังตารางที่ 3

สัญญาณ	ชื่อย่อ	หมายเลขขา
Request to send	RTS	4
Clear to send	CTS	5
Data set ready	DSR	6
Data terminal ready	DTR	20
Transmit data	TxD	2
Receive data	RxD	3
Ground	GRD	7

ตารางที่ 2 แสดงสัญญาณพื้นฐานของ RS-232

สัญญาณทั้งหมดมีมากกว่านี้ เพราะว่าแรกเริ่มนั้น พอร์ทอนุกรมถูกออกแบบมาเพื่อใช้ร่วมกับโมเด็ม ดังนั้นเมื่อนำไปใช้กับอุปกรณ์อื่นบางสัญญาณจึงไม่จำเป็น เพราะสัญญาณเหล่านี้ มีเพื่อใช้เป็นข้อตกลงระหว่างโมเด็มและคอมพิวเตอร์ว่า

1. คอมพิวเตอร์จะไม่ส่งข้อมูลให้แก่โมเด็ม ก่อนที่โมเด็มจะพร้อมส่งข้อมูล
2. คอมพิวเตอร์จะไม่อ่านข้อมูลจากโมเด็ม ก่อนที่โมเด็มจะพร้อม

ความผิดพลาดของกรอบข้อมูล

ความผิดพลาดของกรอบข้อมูล(framing error) คือ ความผิดพลาดของการส่งข้อมูล ที่เกิดจากสัญญาณนาฬิกา (clock) ที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านมีค่าไม่เท่ากัน เพราะว่าจากการทำงานของพอร์ทอนุกรม เมื่อพอร์ทได้รับบิตเริ่มต้นก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบ เพื่ออ่านบิตต่อไปซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบกำหนดได้จาก baud rate ถ้าหากว่าคอมพิวเตอร์ทั้ง 2 เครื่อง มีสัญญาณนาฬิกาไม่ตรงกัน คอมพิวเตอร์ด้านรับ ก็จะอ่านข้อมูลจากส่วนรับข้อมูลของตน ช้าเกินไปหรือเร็วเกินไปก่อนที่ข้อมูลจะถูกส่งมาจากคอมพิวเตอร์ด้านส่ง ทำให้เกิดเฟรมมิงเออร์เรอร์ขึ้น

ฮาร์แวร์แฮนด์เชคกิ้ง

ฮาร์แวร์แฮนด์เชคกิ้ง (hardwaer handshaking) คือ วิธีที่ใช้ในการรับส่งข้อมูลแบบอนุกรม โดยจะต้องตรวจดูสถานะของคอมพิวเตอร์ด้านของการรับข้อมูลว่าพร้อมจะรับข้อมูลหรือไม่ เมื่อคอมพิวเตอร์ด้านส่งพร้อมจะส่งข้อมูลให้ดังนั้นข้อมูลจะต้องไม่ถูกส่งออกไปจนกว่าสัญญาณพร้อมรับข้อมูลจะถูกส่งกลับมาจากคอมพิวเตอร์ด้านรับสัญญาณพร้อมรับข้อมูลของคอมพิวเตอร์ด้านรับคือ clear to send (CTS)

```
do {
        while (not CTS) wait;
        end (byte);
}       while (bytes to send);
```

หมายความว่า จะมีการตรวจสถานะ CTS ตลอดเวลาว่า พร้อมจะรับข้อมูลหรือไม่ ถ้าไม่พร้อม ก็จะรอไปเรื่อยๆ แต่ถ้าพร้อม ก็จะส่งข้อมูลไปให้ จะทำเช่นนี้ไปเรื่อยๆ ตลอดที่ยังมีข้อมูลที่จะต้องส่งอยู่

ปัญหาของการสื่อสาร

เพื่อที่จะให้การสื่อสจรผ่านโมเด็มเป็นไปอย่างถูกต้อง สัญญาณหลายๆ สัญญาณถูกใช้เพื่อตรวจดูว่า ข้อมูลจะพร้อมเมื่อใดหรือข้อมูลไบต์ต่อไปจะถูกส่งมาเมื่อใด แต่ต่อมาเมื่อมีการสื่อสารผ่านคอมพิวเตอร์ สัญญาณบางสัญญาณได้ถูกตัดทิ้งไป เพื่อว่าสายสัญญาณจะได้ลดน้อยลง และลดค่าใช้จ่ายเกี่ยวกับสายส่ง สัญญาณจึงเหลือเพียง GRD,TxD และ RxD ซึ่งในทางทฤษฎีแล้ว เมื่อคอมพิวเตอร์เครื่องหนึ่งพร้อมที่จะส่งข้อมูล คอมพิวเตอร์อีกเครื่องจะต้องพร้อมที่จะรับข้อมูล แต่เมื่อลดสายสัญญาณลงแล้ว จะทำให้เกิดปัญหาตามมามากมาย ที่ยุ่งยากที่สุดปัญหาหนึ่งก็คือ ปัญหาของข้อมูลถูกเขียนทับ (Overrun Error)

ปัญหาข้อมูลถูกเขียนทับ

เมื่อการติดต่อผ่านพอร์ทอนุกรมใช้สายส่งเพียง 2 สายนั้นจะต้องใช้กรรมวิธีพิเศษเล็กน้อยเพื่อให้พอร์ทตัวส่งเข้าใจว่าพอร์ทด้านรับพร้อมที่จะรับข้อมูลเสมอโดยการต่อขา6,8และขา 20 ของคอนเน็กเตอร์เข้าด้วยกันวิธีการเช่นนี้เป็นการตัดฮาร์แวร์แฮนด์เชคกิ้งไปนั่นเอง

แต่การทำเช่นนี้จะทำปให้เกิดปัญหาข้อมูลถูกเขียนทับได้ง่าย ซึ่งก็คือความผิดพลาดในการรับส่งข้อมูลอย่างหนึ่งที่เกิดขึ้นจากการที่คอมพิวเตอร์เครื่องส่งส่งข้อมูลใหม่มาให้คอมพิวเตอร์ด้านรับ ในขณะที่คอมพิวเตอร์ด้านรับยังไม่พร้อมจะรับข้อมูล เนื่องจากในขณะนั้นข้อมูลเดิมยังไม่ได้ถูกอ่านเข้าไปเก็บแต่ข้อมูลใหม่ก็ถูกส่งมาแล้วข้อมูลเดิมจึงถูกเขียนทับไป จึงเกิดเป็นความผิดพลาดของข้อมูลขึ้น

การใช้งานพอร์ทอนุกรมผ่าน BIOS

การใช้งานพอร์ทอนุกรมสามารถทำได้ 3 วิธีคือ ผ่านทางดอสผ่านทาง BIOS และสุดท้ายคือ การเขียนโปรแกรมควบคุมพอร์ทอนุกรมโดยตรง

การเรียกใช้พอร์ทอนุกรมผ่านทางดอสนั้น ไม่เหมาะสมเท่าใดนัก เพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะของการรับส่ง และ ตัวพอร์ทอนุกรมว่าการรับส่งข้อมูล ถูกต้องหรือไม่เพียงใด

การเขียนโปรแกรมควบคุมการทำงานของพอร์ทอนุกรมโดยตรงนั้นคงไม่จำเป็นเพราะมีวิธีที่ดีกว่าและง่ายกว่าคือ การเรียกใช้ผ่าน BIOS อินเตอร์รัพต์หมายเลข 14

การเตรียมสถานะเริ่มต้นของพอร์ท

ในการเตรียมสถานะของพอร์ทอนุกรมเราสามารถทำได้โดยผ่านอินเตอร์รัพต์หมายเลข 14 ฟังก์ชันหมายเลข 0 โดยมีรีจิสเตอร์ AH เป็นตัวผ่านค่าหมายเลขของฟังก์ชัน(ในที่นี้คือ 0) รีจิสเตอร์ AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ทอนุกรม โดยมีความหมายของแต่ละบิต

ตัวอย่างการใช้รหัสเพื่อเตรียมสถานะของพอร์ทอนุกรม เช่น ถ้าต้องการตั้งให้พอร์ทมีอัตรา 9600 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุด และใช้ 8บิตต่อ 1 ไบต์ข้อมูล จะได้รูปแบบของรหัสดังตารางที่ 3

หมายเลขประจำบิต								ความหมายของอัตรารับส่งข้อมูล
b7	b6	b5	b4	b3	b2	b1	b0	
0	0	0						110 baud
0	0	1						150 baud
0	1	0						300 baud
0	1	1						600 baud
1	0	0						1200 baud
1	0	1						2400 baud
1	1	0						4800 baud
1	1	1						9600 baud
			0	0				ไม่มีพาริตี้
			0	1				ไม่มีพาริตี้
			1	0				พาริตี้คี่
			1	1				พาริตี้คู่
					0			1 บิตสิ้นสุด
					1			2 บิตสิ้นสุด
						1	0	7 บิต
						1	1	8 บิต

บิตที่ 7 6 5 4 3 2 1 0

รหัส 1 1 1 1 1 0 1 1

ตารางที่ 3 รูปแบบรหัสเพื่อเตรียมสถานะของพอร์ทอนุกรม

เครื่องคอมพิวเตอร์โดยทั่วไปจะมีพอร์ทอนุกรมได้มากถึง7พอร์ทโดยหมายเลขที่จะใช้สามารถกำหนดผ่าน รีจิสเตอร์ DX พอร์ทแรกมีหมายเลข 0 พอร์ทต่อไปหมายเลข 1 เช่นนี้ต่อไปเรื่อยๆ ฟังก์ชันที่แสดงต่อไปนี้มีชื่อว่า init_port() มีหน้าที่เตรียมสถานะของพอร์ทในระบบ

```
/* ฟังก์ชันนี้ทำหน้าที่เตรียมสถานะเริ่มต้นของพอร์ท */
void port_init(port,code)
int port;
unsigned char code;
{
  union REGS r;
  r.x.dx = port;  /* หมายเลขพอร์ทอนุกรม*/
  r.h.ah = 0;     /* เรียกฟังก์ชันหมายเลข 0 ทำหน้าที่เตรียมสถานะ */
  r.h.al = code;  /* รหัสตั้งสถานะเริ่มต้น */
  int86(0x14,&r,&r);
```

การส่งข้อมูลผ่านพอร์ทอนุกรม 1 ไบต์

อินเตอร์รัพต์หมายเลข 14 ฟังก์ชันหมายเลข 1 ของ BIOS ทำหน้าที่ส่งข้อมูล 1 ไบต์ออกทางพอร์ทอนุกรม หมายเลขของพอร์ทอนุกรมที่จะทำการส่งข้อมูลสามารถกำหนดได้ผ่านรีจิสเตอร์ AL เมื่อทำการส่งเสร็จเรียบร้อยแล้ว สถานะของการส่งข้อมูลจะปรากฏในรีจิสเตอร์ AH เพื่อใช้สำหรับตรวจดูว่า การส่งข้อมูลถูกต้องหรือไม่

รายละเอียดของฟังก์ชัน sport() มีดังนี้

```
/* ฟังก์ชัน sport() ทำหน้าที่ส่งตัวอักษรออกทางพอร์ทอนุกรม */
void sport(port,c)
int port;
char c;
{
  union REGS r;
  r.x.dx = port;     /* กำหนดหมายเลขพอร์ท */
```

```
 r.h.al = c;        /* ตัวอักษรที่ต้องการส่ง */
 r.h.ah = 1;        /* ฟังก์ชันหมายเลข 1 ทำหน้าที่ส่งข้อมูล 1 ไบต์ */
 int86(0x14,&r,&r);
 if(r.h.ah & 128)  { /* ตรวจดูบิตที่ 7 */
   printf("send error detected in serial port");
   exit(1);
 }
}
```

ถ้าบิต 7 ของรีจิสเตอร์ AH มีค่า 1 เมื่อส่งข้อมูลเสร็จสิ้น แสดงว่าเกิดความผิดพลาดในการส่งข้อมูลขึ้น จะต้องตรวจดูสถานะของพอร์ทเพื่อดูว่าเกิดความผิดพลาดอะไร

การตรวจสอบสถานะของพอร์ทอนุกรม

ฟังก์ชันหมายเลข 3 ของอินเตอร์รัพต์หมายเลข 14 ของ BIOS ใช้ตรวจดูสถานะของพอร์ทอนุกรม รีจิสเตอร์ DX ใช้สำหรับหมายเลขพอร์ทที่จะตรวจสอบ สถานะของพอร์ทสามารถแปลความหมายได้จากรหัสที่ปรากฏในรีจิสเตอร์ AH และ AL

สถานะของสายส่ง (AH)		
ความหมายของแต่ละบิตเมื่อมีค่าเป็น	1	ตำแหน่งของบิต
Data Ready		0
Overrun Error		1
Parity Error		2
Framing Error		3
Break-Detect Error		4
Tranfer hold-register empty		5
Transfer shift-register empty		6
Time-out Error		7

สถานะของโมเด็ม (AL)

ความหมายของแต่ละบิตเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
Change in clear-to-send	0
Change in data-set-ready	1
Trailing-edge ring detector	2
Change in line signal	3
Clear-to-send	4
Data-set-ready	5
Ring indicator	6
Line signal detector	7

จะเห็นว่าสัญญาณสถานะต่างๆส่วนใหญ่จะใช้งานกับโมเด็มดังนั้นเมื่อนำมาใช้กับอุปกรณ์อื่นจึงลดความสำคัญลงแต่ก็ยังมีสัญญาณอีกสัญญาณหนึ่งที่มีความสำคัญมากก็คือ Data Ready ซึ่งจะเป็นตัวบอกว่า เมื่อไหร่ที่ข้อมูลถูกรับเข้ามา และพร้อมที่จะถูกอ่านเข้าไปเก็บ ฟังก์ชันในหัวข้อต่อไปมีชื่อว่า rport() มีหน้าที่อ่านข้อมูลจากพอร์ทโดยจะใช้สถานะ Data Ready นี้เป็นตัวบอกว่า ข้อมูลพร้อมที่จะถูกอ่านหรือยัง

การรับข้อมูลผ่านพอร์ทอนุกรม 1 ไบต์

การรับข้อมูลจากพอร์ทอนุกรมสามารถทำได้โดยเรียกผ่าน BIOSอินเตอร์รัพต์หมายเลข 14 ฟังก์ชันหมายเลข 2 รีจิสเตอร์ DX ใช้สำหรับกำหนดหมายเลขพอร์ท ข้อมูลที่อ่านได้จะเก็บในรีจิสเตอร์ AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ทจะสามารถตรวจดูได้ที่บิต 7 ของรีจิสเตอร์ AH ต่อไปนี้คือฟังก์ชัน rport() มีหน้าที่อ่านข้อมูลเข้า 1 ไบต์

```
/* ฟังก์ชันนี้ทำหน้าที่อ่านตัวอักษรจากพอร์ทอนุกรม */
rport(port)
int port;
{
  union REGS r;
/* รอจนกว่าจะได้รับตัวอักษร */
while(!(check_stat(PORT)&256))
  if(kbhit()) {  /* ยกเลิกเมื่อมีการกดคีย์ */
```

```
    getch();
    exit(1);
  }

r.x.dx = port;      */ กำหนดหมายเลขพอร์ท */
r.h.ah = 2;         */ ฟังก์ชันหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
int86(0x14,&r,&r);
if(r.h.ah & 128)
  printf("read error detected in serial port");
return r.h.al;
```

การทำงานของฟังก์ชันนี้คือจะรอจนกระทั่งข้อมูลถูกรับมาผ่านพอร์ทอนุกรม แล้วส่งค่าอักษรกลับมาแต่ว่าการทำงานเช่นนี้อาจทำให้โปรแกรมไม่หลุดจากลูปการทำงาน เช่น เมื่อสายส่งของพอร์ทเกิดเสีย ดังนั้นจึงต้องตรวจสถานะของพอร์ทอนุกรมเสียก่อน ฟังก์ชัน kbhit() ใช้ตรวจสอบการกดคีย์ ถ้าหากไม่มีข้อมูลผู้ใช้ก็สามารถกดคีย์ใดคีย์หนึ่งเพื่อออกจากลูปได้ แต่ถ้ามีข้อมูลรับเข้ามา ฟังก์ชันก็จะผ่านไปเรียกอินเตอร์รัพต์เพื่ออ่านข้อมูลเข้ามาและเช่นเดียวกับการส่งข้อมูลบิต 7 ของรีจิสเตอร์ AH ใช้บอกว่า การอ่านข้อมูลมีข้อผิดพลาดหรือไม่

ขนาดของข้อมูล 7 บิตกับ 8 บิต

หากว่าไฟล์ที่ต้องการโอนย้ายนั้นเป็นเพียงไฟล์ของตัวอักษร(text file) ไบต์ข้อมูลขนาด 7 บิตก็เพียงพอแล้ว เพราะว่าตัวอักษรและเครื่องหมายพิเศษอื่นๆ นั้น ใช้เพียง 7 บิตต่อ 1 ตัวอักษร แต่ถ้าหากเป็นไฟล์แบบอื่นที่ไม่ใช่ไฟล์ของตัวอักษร เช่น ไฟล์ที่ทำงานได้(executable file) ซึ่งจะใช้งานบิตที่ 8 ด้วย โปรแกรมการโอนย้ายไฟล์จะต้องส่งบิตที่ 8 ด้วย

ปัญหาที่ตามมาก็คือ รหัส EOF (end of file) ไม่สามารถนำมาเป็นตัวบอกจุดสิ้นสุดของไฟล์แบบไบนารีได้ ทางแก้ก็คือจะต้องหาขนาดของไฟล์ที่จะส่ง แล้วทำการส่งข้อมูลเป็นจำนวนตามที่คำนวณได้นั้น

2.5 พอร์ตสื่อสาร 8250

พอร์ตสื่อสารเป็นส่วนสำคัญส่วนหนึ่งที่มีอยู่บนไมโครคอมพิวเตอร์ 16 บิทพอร์ตสื่อสารนี้มีชื่อ เรียกอีกอย่าง หนึ่งว่า Com port (คอม-พอร์ต) การออกแบบพอร์ตสื่อสารต้องการให้เป็นไป ตามมาตรฐานการเชื่อมต่อแบบอนุกรมที่เรียกว่า RS-232C พอร์ตนี้เป็นทางออกของข้อมูลที่ผู้ใช้สามารถส่งหรือรับกับระบบอื่นได้พอร์ตสื่อสารจึงเป็นพอร์ตที่จำเป็นและมักจะกำหนดอยู่ในสเป็กด้วยเสมอ

พอร์ตสื่อสาร RS-232C บนเครื่องไมโครคอมพิวเตอร์ 16 บิท มีโครงสร้างที่สามารถโปรแกรมด้วยการ ส่งรหัสคำสั่งให้กับชิพหลักได้ อย่างไรก็ตามผู้ออกแบบได้ในทางเลือกในการสื่อสาร ด้วยกระแสวนรอบ (current loop) หรือแบบแรงดันคือ RS-232C โดยใช้จัมเปอร์เพื่อเลือกระบบสำหรับตัว RS-232C โดยใช้จัมเปอร์เพื่อเลือกระบบสำหรับตัว RS-232C นี้เป็นมาตรฐานแบบอะซิงโครนัสที่โปรแกรมสตาร์ตบิต สต็อปบิต และพาริตี้บิต อัตราการส่งก็สามารถกำหนดได้ตั้งแต่50บอดถึง 9600 บอด ลักษณะพิเศษของวงจร์คือสามารถส่งสัญญาณมาอินเตอร์รัพต์ซีพียูตามเงื่อนไขได้และยังมีโครงสร้างฮาร์ดแวร์ป้อนกลับเพื่อใช้ในการตรวจสอบระบบว่าทำงานปกติดีหรือไม่ได้อีกด้วยวงจรพอร์ตสื่อสารของไมโครคอมพิวเตอร์ 16 บิท นี้ใช้ไอซีหมายเลข 8250 เป็นตัวสำคัญของระบบ 8250 เป็นไอซีขนาด40ขามีขีดความสามารถพิเศษดังนี้

- มีบัฟเฟอร์ในตัวเพื่อทำให้ไม่จำเป็นต้องซิงโครไนซ์การรับส่ง
- ใช้สัญญาณนาฬิกาอิสระต่างหากไม่ขึ้นกับสัญญาณนาฬิกาของระบบ
- มีสัญญาณตอบโต้ควบคุมโมเด็มทั้ง CTS (clear to send) RTS (request to send) DSR (data set ready) DTR (data terminal ready) RC (ringIndicator) และสัญญาณดีเทคตัวพาหะ (carrier detect)
- ตรวจสอบสตาร์บิตที่ผิดพลาด
- ตรวจสอบและสร้างสัญญาณสายขาด (line break) เพื่อใช้ในการตรวจสอบการทำงานของระบบ

ชิพ 8250

มีโครงสร้างการทำงานคล้ายกับ8251ที่นักฮาร์ดแวร์ทั่วไปรู้จักดีการทำงานของ 8250 ต้องได้รับการโปรแกรมก่อน หลังจากนั้นจะทำงานตามรูปแบบที่ได้โปรแกรมไว้ จนกว่าจะมีการโปรแกรมค่าใหม่อย่างไรก็ตาม8250มีขีดความสามารถในการทำงานได้สูงกว่า 8251 หลายอย่างและด้วยเหตุนี้เอง 8250 จึงเป็นชิพประจำที่ใช้กับเครื่องไมโครคอมพิวเตอร์

โครงสร้างของพอร์ตสื่อสาร

พอร์ตสื่อสารของเครื่องไมโครคอมพิวเตอร์16 บิต ที่จะกล่าวถึงนี้เป็นระบบมาตรฐานตามแบบ เครื่องไอบีเอ็มพีซีเอ็กซ์ที โครงสร้างบล็อกไดอะแกรมแสดงดังรูปที่ 1

เห็นได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสของระบบ ซึ่งก็คือสล็อตขนาด 31*2 (62) ขาของระบบ นั่นเอง ซีพียูติดต่อกับ 8250ในลักษณะพคร์ตที่เป็นอินพุตเอาต์พุตการจัดพอร์ตนี้กำหนดหมายเลข พอร์ตอย่างเจาะจง

ระบบไมโครคอมพิวเตอร์ 16 บิท มีพอร์ตสื่อสารสองพอร์ตคือ คอม1(com1) และ คอม 2 (COM2) ทั้ง COM1 และ COM2 มีหมายเลขอินพุตเอาต์พุตพอร์ต ดังตารางที่ 4

อินพุตเอาต์พุทพอร์ต		เลือกรีจิสเตอร์	สถานะ DLAB
พอร์ต COM1	พอร์ต COM 2		
3F8	2F8	บัฟเฟอร์ IX	DLB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLB = 0 (อ่าน)
3F8	2F8	แลทช์ตัวหาร LSB	DLBA = 1
3F9	2F9	แลทช์ตัวหาร MSB	DLAT = 1
3F9	2F9	รีจิสเตอร์อินเตอร์รัพต์อีนาเบิ้ล	
3FA	2FA	รีจิสเตอร์เลือกอินเตอร์รัพท์	
3FB	2FB	รีจิสเตอร์ควบคุมสายสื่อสาร	
3FC	2FC	รีจิสเตอร์ควบคุมโมเด็ม	
3FD	2FD	รีจิสเตอร์แสดงสถานะสายสื่อสาร	
3FE	2FE	รีจิสเตอร์แสดงสถานะโมเด็ม	

ตารางที่ 4 หมายเลขอินพุตเอาต์พุตพอร์ทของ COM1 และ COM2

การเลือกหมายเลขอินพุตเอาต์พุตพอร์ตแยกเป็นสองกลุ่ม กลุ่มหนึ่งคือ COM1 จะกำหนดหมายเลขพอร์ตจาก 3F8 ถึง 3FF อีกกลุ่มหนึ่งถ้ากำหนดหมายเลขพอร์ตเป็น2F8-FE ในการเลือกหมายเลข รีจิสเตอร์ภายในกำหนดด้วยแอดเดรส A0,A1 และ A2 สำหรับการเลือก COM1 และ COM2 เราใช้แอดเดรส A8 เป็นตัวเลือกการเลือกนี้กำหนดเป็นตารางได้ดังตารางที่5

แอดเดรส 2F8-3FF และ 2F8 ถึง 2FF											
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	DLAB	รีจิสเตอร์
1	1/0	1	1	1	1	1	X	X	X		
						0	0	0		0	บัฟเฟอร์รีจิสเตอร์ตัวรับข้อมูล(อ่าน) รีจิสเตอร์โฮลดิ้ง (เขียน)
						0	0	1		0	อินเตอร์รัพต์อีนาเบิ้ล
						0	1	0		X	กำหนดอินเตอร์รัพต์
						0	1	1		X	ควบคุมสายสื่อสาร
						1	0	0		X	ควบคุมโมเด็ม
						1	0	1		X	สถานะสายสื่อสาร
						1	1	0		X	สถานะโมเด็ม
						1	1	1		X	ไม่ใช้
						0	0	0		X	แลทช์ตัวหาร (LSB)
						0	0	1		X	แลทช์ตัวหาร(MSB)

ตารางที่ 5 การกำหนดแอดเดรสสำหรับหมายเลขพอร์ทต่าง ๆ

การอินเตอร์รัพต์

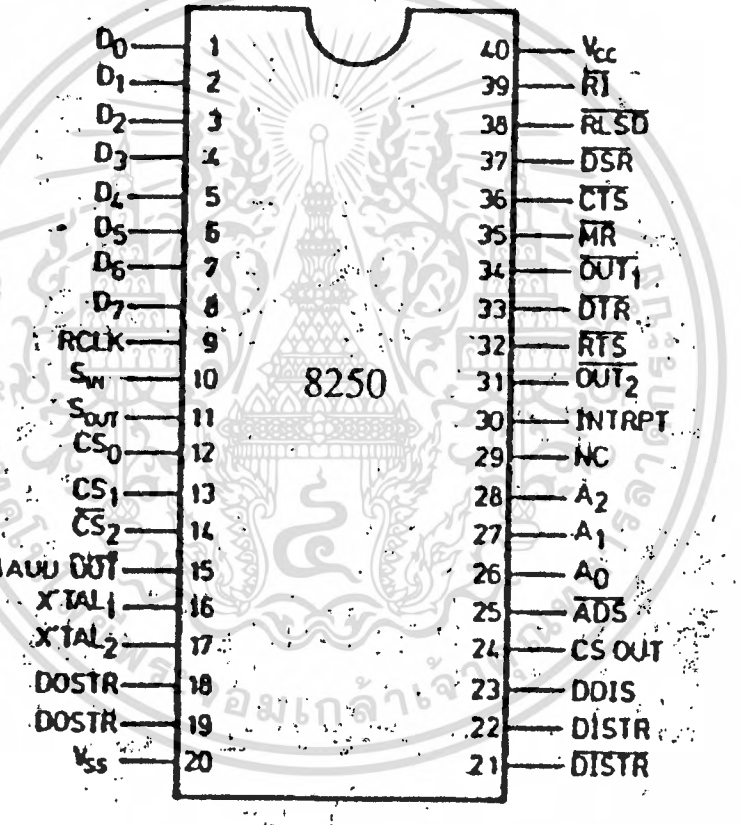
จากที่เคยกล่าวถึงโครงสร้างการควบคุมอินเตอร์รัพต์ของชิพ 8259 มาแล้ว 8259 ได้จัดลำดับการ อินเตอร์รัพต์ไว้ 8 ระดับ สัญญาณรับอินเตอร์รัพต์ที่เข้าทางชิพ 8259 มี 8 เส้น คือ IRQ0-IRQ1 สำหรับกรณีของระบบสื่อสารอนุกรมได้กำหนดสัญญาณการอินเตอร์รัพต์ไว้แล้วคือให้ IRQ4 เป็นสัญญาณ อินเตอร์รัพต์ของระบบCOM1 และ COM2 ในการที่จะส่งสัญญาณอินเตอร์รัพต์

รูปแบบข้อมูลที่รับหรือส่ง

การสื่อสารข้อมูลของระบบนี้เป็นการสื่อสารแบบซิงโครนัสรูปแบบของข้อมูลจะมีสตาร์ตบิตบิตตรวจสอบพาริตี้ และสต็อปบิต ข้อมูลบิตแรกของการส่งเป็นสตาร์ตบิตระบบจะส่งสตาร์ดบิตก่อนหลังจากนั้นจะตามด้วยข้อมูล และแทรกด้วยบิตตรวจสอบพาริตี้ตามด้วยสต็อปบิตขนาดของข้อมูลมีค่าได้ตั้งแต่ 5-8 บิต แต่ทั่วๆไปใช้ 8 บิต สต็อปบิตมีได้ 1,1.5หรือ 2 บิต ค่าเหล่านี้สามารถกำหนดลงไปใน รีจิสเตอร์ควบคุมสายสื่อสาร

ชิพ 8250

8250 เป็นไอซีขนาด 40 ขา มีการทำงานเพื่อควบคุมสิ่งต่าง ๆ ที่เกี่ยวกับการสื่อสารแบบอนุกรมได้หมดโครงสร้างทางฮาร์ดแวร์ของอะแดปเตอร์การ์ดนี้จึงไม่ยุ่งยากมากนัก การจัดวางขาของไอซี 8250 เป็นดังรูปที่ 12



รูปที่ 12 การจัดวางขาของไอซี 8250

ขาต่าง ๆ แต่ละขามีรายละเอียดดังต่อไปนี้คือ

สัญญาณอินพุต

ขาเลือกชิพ CS0,CS1,CS2(chip select) ขา 12-14 เป็นสัญญาณเลือกชิพโดยที่เมื่อต้องการเลือก ชิพจะให้ CS0,CS1 เป็น "1" และ CS2 เป็น "0" สัญญาณเลือกชิพนี้จะได้รับการเลือกแลตช์ไว้ไน ขณะที่สัญญาณ ADS มีค่าเป็น "0"การเลือกชิพนี้จะมีไว้เพื่อให้ซีพียูติดต่อกับ 8250

สโตรบข้อมูล

DISTR, DISTR data input strobe ขา 22,21 เมื่อสัญญาณที่ DISTR เป็น "1" และ DISTR เป็น "0" ในขณะที่มีการเลือกชิพ เป็นขณะที่ซีพียูจะอ้างข้อมูลจากรีจิสเตอร์ภายในที่ได้รับการกำหนดไว้แล้วมายังซีพียู สัญญาณนี้จึงเป็นสัญญาณอ่านข้อมูลหรือ read นั่นเอง

ขาสโตรบข้อมูลเอาท์พุต

DOSTR, DOSTR ขา 19 และขา 18 เป็นสัญญาณที่แอคตีฟขึ้นเพื่อให้ซีพียูเขียนข้อมูลลงมายังรีจิสเตอร์ ของ 8250

ขาสโตรบแอดเดรส

ADS ขา 25 เมื่อมีค่าเป็น "0" จะแอคทีฟเพื่อแลตช์ค่า A0-A2 เลือกรีจิสเตอร์ภายใน การเลือกชิพแอคทีฟอยู่

ขาเลือกรีจิสเตอร์

A0,A1,A2, ขา 26-28 เป็นตัวกำหนดแอดเดรสของรีจิสเตอร์ภายในเพื่อให้ซีพียูทำการติดต่อ กับ 8250 ตามค่ารีจิสเตอร์ที่กำหนดเพื่อการเขียนหรืออ่าน อย่างไรก็ตามการทำงานของ A0-At นี้ขึ้น กับสัญญาณเลือกตัวหาร DLAB divisor latch access bit ซึ่งกำหนดค่ารีจิสเตอร์ได้ดังนี้

รีเซต MR-master reset

ขา 35 เมื่อมีค่าเป็น "1" จะรีเซตการทำงานของชิพ 8250 โดยทำให้ค่าต่าง ๆ ในรีจิสเตอร์ ถูกเคลียร์หมด (ยกเว้นบัฟเฟอร์ของตัวรับตัวส่งและตัวหารขณะทำการรีเซตจะมีผลต่อสัญญาณเอาต์พุตด้วย) ผลของการรีเซตแสดงไว้ในตารางที่ 3

สัญญาณนาฬิกาตัวรับ

RCLK -receiver clock ขา 9 เป็นขาที่ตัวรับสัญญาณนาฬิกาเพื่อกำหนดอัตราบอดสัญญาณนาฬิกา นี้จะมีค่าเป็น 16 เท่าของที่นำมาใช้

ขาอินพุตข้อมูลอนุกรม

SIN- serial input ขา 10 เป็นขารับข้ออนุกรมจากสายส่งในการเชื่อมโยงการติดต่อสื่อสาร

ขาเคลียทูเซนด์

CTS-clear to send ขา 36 เป็นสัญญาณที่ใช้ในการติดต่อกับโมเด็ม เงื่อนไขของสัญญาณนี้ สามารถเก็บไว้ภายในชิพ 8250 ที่จะให้ซีพียูอ่านไปตรวจสอบได้ โดยเก็บไว้ที่บิต 4 ของรีจิสเตอร์แสดง สถานะจะเป็นตัวบอกว่า CTS ได้ เปลี่ยนสถานะไปหลังจากการอ่านครั้งก่อนแล้วหรือไม่

ขาดาต้าเซตรีดี้

DSR - data set ready ขา 37 เมื่อ เป็น "0" จะแสดงว่าโมเด็มหรือข้อมูลได้รับการเซตเตรียม พร้อมแล้วสำหรับการเชื่อมต่อกับสายสื่อสาร และส่งข้อมูลระหว่าง 8250 กับโมเด็ม สัญญาณ DSR เป็นสัญญาณอินพุตของ 8250 ที่ซีพียูสามารถอ่านไปดูได้ทางบิตที่ 5 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 1 ของรีจิสเตอร์แสดงสถานะจะเป็นตัวบอกว่าสัญญาณ DSR ได้เปลี่ยนสถานะไปหลังจากที่อ่านครั้งก่อนแล้วหรือไม่

หมายเหตุ

ทั้ง CTS และ DSR เมื่อมีการเปลี่ยนสถานะและถ้าได้รับการอีนาเบิ้ลที่ modem status interrupt จะส่งผลในการสร้างสัญญาณอินเตอร์รัพต์

ตรวจสอบสายสื่อสาร

RLSD received line signal detect ขา 38 ถ้าเป็น "0" หมายถึงแอคทีฟคือ 8250 รับสัญญาณตรวจสอบสัญญาณพาหะจากโมเด็ม ว่า โมเด็มตรวจสอบได้แล้ว หรือข้อมูลได้รับการเซตแล้วซีพียูสามารถตรวจสอบสัญญาณนี้ทางบิต 7 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 3 จะเป็นบิตที่แสดงสถานะ ว่าสัญญาณนี้ได้รับการเปลี่ยนแปลงหลังจากอ่านไปแล้วหรือยัง

ขาแสดงวงจรเรียก RI ring indicator

ขา 39 สัญญาณนี้แอคทีฟด้วย ลอจิก "0" เป็นสัญญาณที่ส่งมาจากโมเด็ม โมเด็มตรวจสอบสัญญาณการเรียก (ringing) สัญญาณนี้ตรวจสอบได้ทางบิต 6 และดูสถานะการเปลี่ยนหลัง จากอ่านแล้วจากบิต 2

ขาไฟเลี้ยง Vcc ขา 40 Vss ขา 20

เป็นสัญญาณจากแหล่งจ่ายไฟเลี้ยง 5 โวลต์และกราวด์

Request to send RTS

ขา 32 เริ่มขานี้มีลอจิกเป็น "0" หมายความว่า8250 พร้อมที่จะส่งข้อมูลแล้ว สัญญาณขา นี้จะได้รับการเซตให้ แอคทีฟด้วยการโปรแกรมค่าลงไปในรีจิสเตอร์ควบคุมที่บิต 1

เอาต์พุต 1

OUT 1 ขา 34 เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอคทีฟเป็น "0" ด้วยการโปรแกรมลงไปในบิต 2 ของรีจิสเตอร์ควบคุมโมเด็ม

เอาต์พุต 2

OUT 2 ขา 31เป็นขาที่ผู้ใช้สามรถโปรแกรมให้แอคทีฟเป็น"0"ด้วยการโปรแกรมค่าลงในรีจิสเตอร์ควบคุมโมเด็มทางบิต 3

เลือกชิพเอาต์

CSOUT chip select out ขา 34 เมื่อมีค่าเป็น "1" จะบอกว่าชิพนี้ได้รับการเลือกชิพโดยซีพียูทางขา CS0,CS1 และ CS2

ไดร์ฟเวอร์ดีสเอเบิ้ล

DDIS driver disable ขา 23 เป็นลอจิก "0"เมื่อซีพียูกำลังอ่านข้อมูลจาก 8250 สัญญาณ DDISเป็น "1" มีไว้สำหรับการดิสเอเบิ้ลการรับส่งภายนอกในกรณีที่ใช้ 8250กับซีพียูผ่านทางบิต D0-D7 เพื่อบอกเวลาว่าซีพียูเป็น 8250 ติดต่อกันอย่างไร

สัญญาณบอดเอาต์

BAUDOUT ขา 15 เป็นสัญญาณนาฬิกาที่มีความถี่เป็น 16 เท่าของสัญญาณนาฬิกาแล้วหารด้วยค่าที่โปรแกรมกำหนดในตัวหาร

อินเตอร์รัพท์

INTRPT-ขา 30 เมื่อเป็น "1" เป็นการส่งสัญญาณอินเตอร์รัพต์ออกไปจาก 8250

ข้อมูลเอาต์พุต

SOUT ขา 11 เป็นขาที่ใช้ส่งข้อมูลอนุกรมออกไปยังสายสื่อสาร

สัญญาณอินพุตเอาต์พุต

- ข้อมูล D6-D7 เป็นสัญญาณต่อเชื่อมกับบัสข้อมูลของระบบ
- ขาสัญญาณXTAL1,XTAL2 ขา 16-17 เป็นขาต่อกับคริสตอลเพื่อสร้างสัญญาณนาฬิกา

สถานะของ 8250 เมื่อเริ่มต้น

ก่อนการทำงานหรือโปรแกรมเราควรจะทราบว่าสถานะต่างๆ ของ 8250 เป็นอย่างไร โดยเฉพาะอย่างยิ่งเมื่อเริ่มเปิดเครื่อง จะมีสัญญาณรีเซตซึ่งเป็นสัญญาณเดียวกับซีพียูมาทำการรีเซต8250 8250 ขณะนั้นจะมีสถานะเป็นอย่างไร เอาต์พุตของวงจรจะให้สัญญาณอะไรตารางที่ 6 นี้เป็นตารางที่จะให้รายละเอียด 8250 ขณะเริ่มต้น

รีจิสเตอร์/สถานะ	การควบคุม	สถานะเมื่อรีเซต
อินเตอร์รัพต์รีจิสเตอร์ภายใน	มาสเตอร์รีเซต	ทุกบิตเป็น "0" (0-3 ถูกกำหนด 4-7 จะเป็นถาวร)
รีจิสเตอร์ Interrupt Identification	มาสเตอร์รีเซต	บิต 0 เป็น "1" บิต 1-2 เป็น "0" บิต 3-7 เป็น "0" ถาวร
รีจิสเตอร์ควบคุมสาย	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
รีจิสเตอร์ควบคุมโมเด็ม	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
รีจิสเตอร์แสดงสถานะสาย	มาสเตอร์รีเซต	ยกเว้นบิต5 และ 6 เป็น"1"
รีจิสเตอร์สถานะโมเด็ม	มาสเตอร์รีเซต	บิต 0-3 เป็น "0" บิต 4-7 เป็นสัญญาณอินพุท
SOUT	มาสเตอร์รีเซต	เป็น "1"
INTRPT (RCVR Error)	Readd LSR /มาสเตอร์รีเซต	เป็น "0"
INTRP (RCVR Data Ready)	Read RBR/มาสเตอร์รีเซต	เป็น "0"
INTRP (RCVR Data Ready)	Read IIR/Write THR/มาสเตอร์รีเซต	เป็น "0"
INTRP (เปลี่ยนสถานะโมเด็ม)	Read MSR/มาสเตอร์รีเซต	เป็น "0"
OUT 2	มาสเตอร์รีเซต	เป็น "1"
RTS	มาสเตอร์รีเซต	เป็น "1"
DTR	มาสเตอร์รีเซต	เป็น "1"
OUT 1	มาสเตอร์รีเซต	เป็น "1"

ตารางที่ 6 ค่าเริ่มต้นและเอาต์พุตของ 8250

การต่อวงจรเข้ากับระบบ

บอร์ดอะแดปเตอร์สื่อสารนี้มีโครงสร้างการเชื่อมต่อตามพอร์ด 3F8-3FE และ 2F8-2FE ดังนั้นจะใช้บิต A8 เป็นตัวเลือกบนบอร์ดจะมีจัมเปอร์เพื่อจะบอกว่าเป็นพอร์ตสื่อสารแบบ COM1 หรือ COM2

การเลือกแอดเดรสใช้ 74LS30 ซึ่งเป็น NAND เกตแบบ 8 อินพุตมาเป็นตัวเลือกโดยมี U15 ในรูปเป็นตัวเลือก Aa ดังได้กล่าวแล้ว

ส่วนของบัสข้อมูล D0-D7 จะผ่านบัฟเฟอร์คือ74LS245ก่อนเข้าสู่8250ส่วนสัญญาณ 8250 ทั้ง 10 มีดังนี้

- ขามาสเตอร์รีเซต MR จะต่อโดยตรงกับสัญญาณรีเซต(Reset) ของระบบ

ขา ADS,DISTR,DOSTR ต่อลงกราวด์ ทั้งนี้เป็นสัญญาณแอดเดรสที่ต่อมาที่ชิพ 8250 นี้เป็นสัญญาณแอดเดรสล้วนแล้วไม่ต้องสโตรบอีก

- ขา DISTR ต่อกับ IOR ของระบบโดยผ่านอินเวอร์เตอร์ 2 ตัว
- ขา DOSTR ต่อกับ LOW ของระบบ
- ขา CS1 CS2 ต่อขึ้นเป็นลอจิก "1"
- ขา CS มาจากการถอดรหัสให้เป็นพอร์ตของ COM1 หรือ COM2 ตามต้องการ
- ขา XTAL2 ไม่ใช้ ส่วน XTAL1 ต่อมาจากออสซิสเลเตอร์ 18.432 MHz ของระบบวงจรของบอร์ดอะแดปเตอร์สื่อสาร
- ส่วนของสัญญาณเอาต์พุตประกอบด้วยสัญญาณควบคุมโมเด็ม RLSD,DSR,CTS,RI ต่อออกไปยังหัวต่อตามมาตรฐาน EIA RS232
- ขา SIN,SOUT ต่อออกไปขาเอาต์พุตเช่นกันแต่มีการปรับให้เป็นสัญญาณแรงดันตามมาตรฐาน EIA หรือเลือกส่งเป็นกระแสวนรอบก็ได้การเลือกใช้จัมเปอร์ I1-I5 เป็นตัวเลือกการเลือกจัมเปอร์บนบอร์ดอะแดปเตอร์นั้นผู้ออกแบบบอร์ดทำให้ง่ายต่อการใช้ด้วยการให้จัมเปอร์มา เพียงผู้ใช้เลือกทิศทางก็จะได้พอร์ต COM1 หรือ COM2 หรือ ถ้าเลือกจัมเปอร์อีกตัวก็เลือกการส่งแบบกระแสหรือแรงดันได้

การใช้งานรีจิสเตอร์ต่าง ๆ บน 8250

ผู้ใช้งาน 8250 จำเป็นต้องเข้าใจว่ารีจิสเตอร์ของ 8250 มีความหมายดังนี้

- รีจิสเตอร์ควบคุมสายสื่อสาร (line control register)

ในการควบคุมรูปแบบของข้อมูลแบบอะซิงโครนัสนั้น ผู้โปรแกรมจะต้องกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร รีจิสเตอร์ตัวนี้มี 8 บิต โดยแต่ละบิตมีความหมายดังนี้

- บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง
- บิต 2 เป็นบิตที่ใช้ในการกำหนดจำนวนสต๊อปบิตถ้าเป็น "0" หมายถึงใช้สต๊อปบิต 1 บิต แต่ถ้าบิต 2 เป็น "1" ในกรณีส่งแบบ 5 บิตจะมีความยาวของสต๊อปบิตเป็น 1.5 บิต แต่ถ้าส่งแบบ 6,7 หรือ 8 บิตความยาวของสต๊อปบิตจะเป็น 2
- บิต 3 บิตนี้เป็นบิตแสดงการอีนาเบิ้ลให้มีการตรวจสอบพาริตี้ โดยถ้าบิตนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้
- บิต 4 มีค่าเป็น "0" และบิต 3 มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะมีการกำหนดเป็นพาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะเป็นฟาริตี้คู่
- บิต 5 เมื่อบิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" และบิต 4 มีค่าเป็น "1"จะมีการแทรกหรือตรวจสอบพาริตี้ด้วยเงื่อนไขกำหนดให้เป็น "0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น "1"และบิต 5 มีค่าเป็น "1" จะมีการกำหนดบิตพาริตี้เป็น "1"
- บิต 6 เป็นบิตที่ควบคุมการเบรคเมื่อบิต 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด
- บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิตที่จะมีผลต่อการแลตช์ตัวหารดังที่กล่าวมาแล้วในตารางที่ 1 และ 2

การโปรแกรมอัตราบอด (Boud Rate Generator)

อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz การกำหนดอัตราบอดด้วยการกำหนดหารนี้ตัวหารจึงเป็นค่าที่กำหนดในรีจิสเตอร์ 2 ตัวตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ DLAB = 1 แล้วให้ลดลงมาในรีจิสเตอร์ 3F ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F เมื่อ DLAB=1 จะเป็นค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตารางที่ 4

รีจิสเตอร์แสดงสถานะสายสื่อสาร (Line Status Register)

รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบิตต่างๆ ในรีจิสเตอร์นี้เป็นดังนี้

- บิต0 บิตนี้เป็นบิตที่บอกสถานะการรับข้อมูล ถ้าบิตนี้เป็น "1" แสดงว่าการรับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว หรือจะให้ซีพียูเขียนข้อมูลกลับมายังรีจิสเตอร์นี้ก็ได้

- บิต 1 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด Overrun Error กล่าวคือขณะที่มีข้อมูลที่บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไปปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับบนบัฟเฟอร์นี้ บิตนี้จะรีเซตโดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว
- บิต 2 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิด Parity Error กล่าวคือถ้ามีการตรวจสอบบิตพาริตี้แล้ว ไม่เป็นไปตามที่กำหนดไว้ บิตนี้จะได้รับการรีเซต โดยซีพียูเมื่อ ซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว
- บิต 3 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่า เฟรมของข้อมูล ไม่เป็นไปตามที่กำหนด เช่น ตรวจสอบจำนวนบิตโดยดูที่พาริตี้และสต๊อปบิตไม่เป็นไปตามที่กำหนด
- บิต 4 บิตนี้เรียกว่า Break interrupt (BI) บิตนี้จะได้รับการเซตให้มีค่าเป็น "1"ถ้าหากว่ารับข้อมูลอินพุทเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร
- บิต 5 บิตนี้เป็นบิตที่บอกว่า 8250 พร้อมที่จะรับข้อมูล จากสื่อสารบิตนี้ จะได้รับการเซตให้มีค่าเป็น "1" บิตนี้ยังคงสร้างสัญญาณอินเตอร์รัพต์ เพื่อส่งไปบอกซีพียูด้วย บิตนี้จะมีสถานะเซตเมื่อมีการส่งถ่ายข้อมูลจากโฮลด์รีจิสเตอร์ไปยังชิพรีจิสเตอร์เพื่อพร้อมที่จะส่ง
- บิต 6 เป็นบิตที่จะบอกว่า ชิพรีจิสเตอร์ว่างเปล่า บิตนี้จะได้รับการเซตให้มีค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว
- บิต 7 จะเป็น "0" ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพต์ (IIR Interrupt Identification Register)

ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพต์ภายในชิพ เพื่อว่าให้การทำงานระหว่าง 8250 กับซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และเพื่อให้ผู้เขียนซอฟต์แวร์เขียนซอฟต์แวร์ได้ง่ายและสั้นลงได้มาก 8250 กำหนดความสำคัญของอินเตอร์รัพต์ไว้ 4 ระดับ คือ

- ระดับแรก สถานะการรับข้อมูลจากสายสื่อสาร
- ระดับสอง การพร้อมรับข้อมูล
- ระดับสาม ขณะรีจิสเตอร์โฮลดิ้งสำหรับส่งข่าว
- ระดับสี่ สัญญาณสถานะโมเด็ม

ในขณะที่มีความต้องการอินเตอร์รัพต์หลายระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่า รอไว้ก่อนโดยเก็บสถานะ การอินเตอร์รัพต์นี้ไว้ใน Interrupt Identification Register

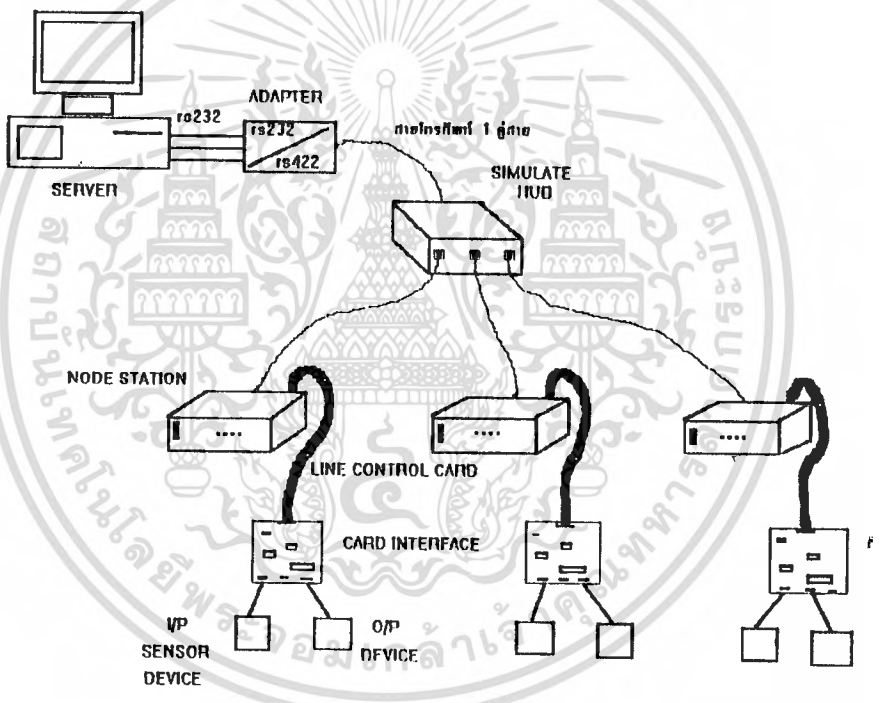
ความหมายของรีจิสเตอร์นี้มีดังนี้

- บิต 0 เป็นบิตที่ใช้แสดงว่า มีอินเตอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถให้ ซีพียูตรวจสอบ ดูด้วยวิธีการ Polling ได้ถ้าบิตนี้เป็น "1" หมายถึง ไม่มีอินเตอร์รัพต์เกิดขึ้น
- บิต 1-2 เป็นบิตที่แสดงความหมายบอกว่าการอินเตอร์รัพต์ที่เกิดขึ้นนั้นมาจากการอินเตอร์รัพต์ตามฟังก์ชันใด

บทที่ 3
หลักการออกแบบ

3.1 การออกแบบการต่อใช้งานระบบ

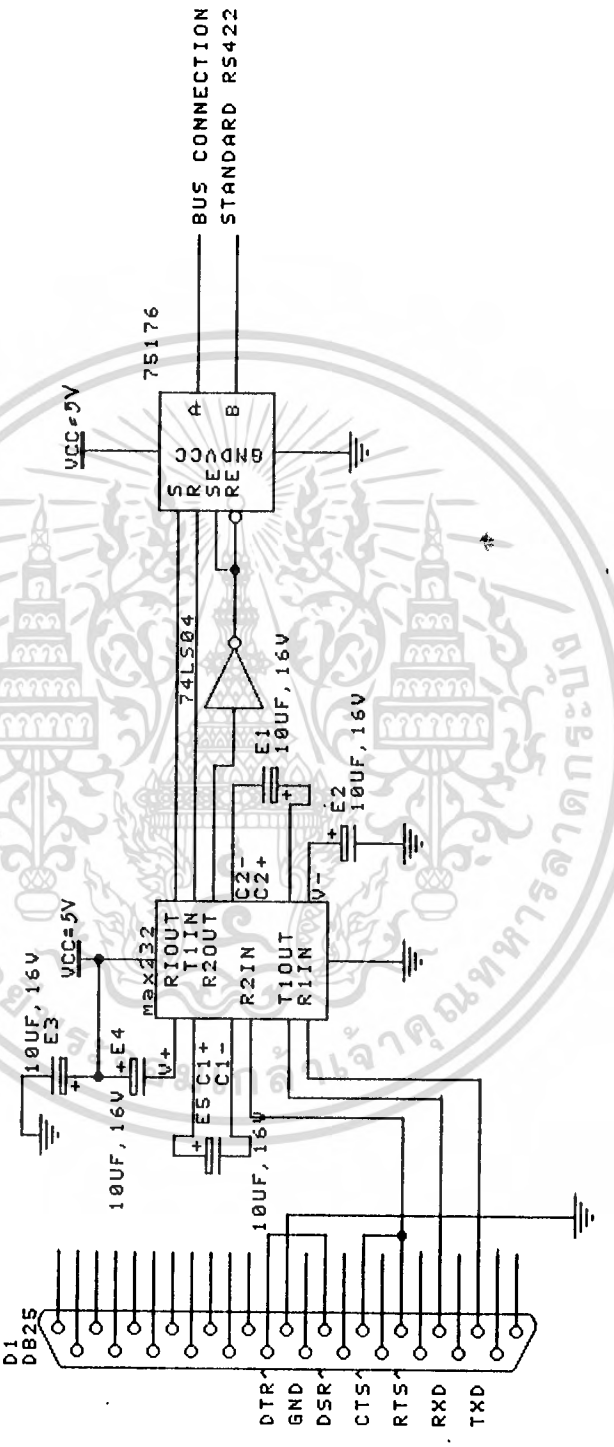
ในการออกแบบโครงงานนี้ได้ใช้รูปแบบการติดต่อระหว่างสถานีแม่และสถานีลูก เป็นแบบบัส โดยอาศัยแอดเดรสของแต่ละสถานีเป็นตัวแบ่งแยกข้อมูลที่ใช้ในการติดต่อ เพื่อให้รู้ว่าข้อมูลที่ส่งอยู่นั้นเป็นของสถานีไหน ดังรูปแสดงการต่อระบบ



รูปที่ 13 แสดงการต่อระบบ

จากรูปที่ 13 จะเห็นว่าที่บัสข้อมูล ที่ติดต่อระหว่างสถานีแม่ (Server Station) ซึ่งใช้คอม พิวเตอร์มาทำกับสถานีลูก (Node Station) ที่เป็นบอร์ดคอนโทรลเลอร์ จะผ่านการแปลงจากมาตรฐาน RS-232 ไปเป็น RS-422 ในขณะที่สถานีแม่ส่งข้อมูลออกไปที่สถานีลูกและมีการแปลงจากมาตรฐาน RS-422 ไปเป็น RS-232 ในขณะที่สถานีแม่เป็นตัวรับข้อมูลส่วนการรับหรือส่งที่สถานีลูกจะเป็นแบบมาตรฐาน RS-422 อย่างเดียว

3.2 วงจรที่ใช้แปลงจาก RS-232 เป็น RS-422



รูปที่ 14 แสดงวงจรการแปลงลักษณะการส่งข้อมูลจากมาตรฐาน RS-232 เป็น RS-422 และการแปลงกลับด้วย

อธิบายการทำงานดังนี้

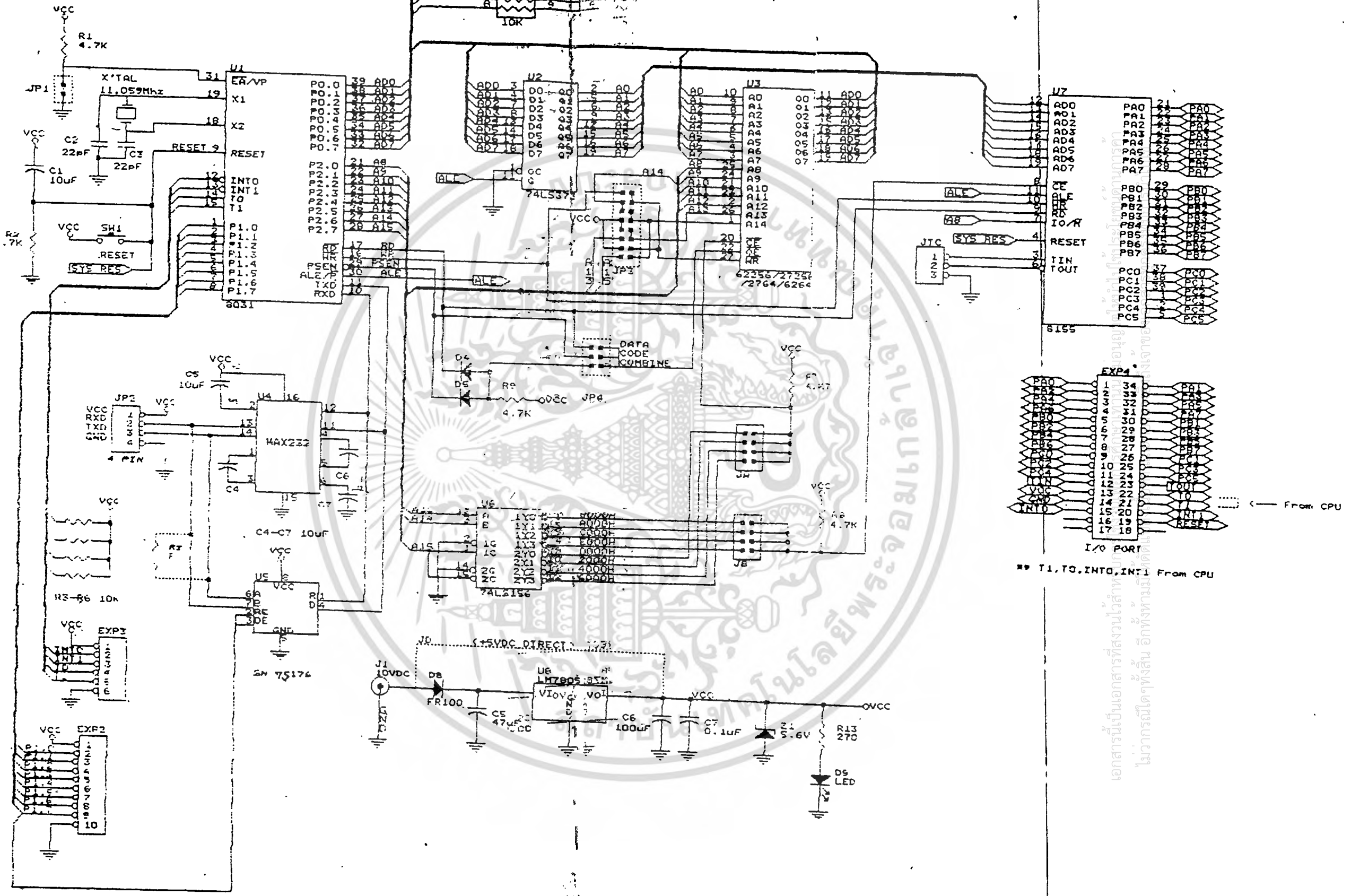
IC MAX232

- ขา R1IN จะรับสัญญาณส่งจากคอมพิวเตอร์ ที่เป็นระดับสัญญาณ (+12v,-12v) RS-232 ทำการแปลงเป็นระดับสัญญาณ (+5v,-5v) RS422 ออกมาที่ขา R1OUT
- ขา T1IN จะรับสัญญาาณที่ระดับ (+5v, 5v) RS-422 เข้ามาแล้วแปลง เป็นระดับสัญญาณ (+12v,-12v) RS-232 ออกมาที่ขา T1OUT ส่งเข้าคอมพิวเตอร์
- ขา R2IN รับสัญญาณ RTS จากคอมพิวเตอร์โดยทำแฮนเช็คกิ้งกับขา CTS เมื่อต้องการที่จะส่งข้อมูลต้องให้ขา RTS แอคทีฟไฮ จะได้สัญญาณโลจิก "0" ที่ขา R2OUT ไปควบคุมทิศทางการติดต่อข้อมูลที่ IC 75176 เมื่อต้องการจะรับข้อมูลต้องให้ขา RTS แอคทีฟโลว์

IC 75176

- ขา A,B เป็นบัสข้อมูลแบบ RS-422 บาลานซ์ ขา RE แอคทีฟโลว์ ควบคุมการรับข้อมูลจากบัส RS-422
- ขา SE แอคทีฟไฮ ควบคุมการส่งข้อมูลไปที่บัส RS-422
- ขา S รับข้อมูลจาก MAX232 ส่งออกบัส RS-422
- ขา R รับข้อมูลจากบัส RS-422 ส่งให้ MAX232

3.3 วงจรการใช้งานของสถานีลูก



รูปที่ 15 แสดงวงจรที่ใช้งานของสถานีลูก (Node Station)

ส่วนประกอบของวงจรมีดังนี้

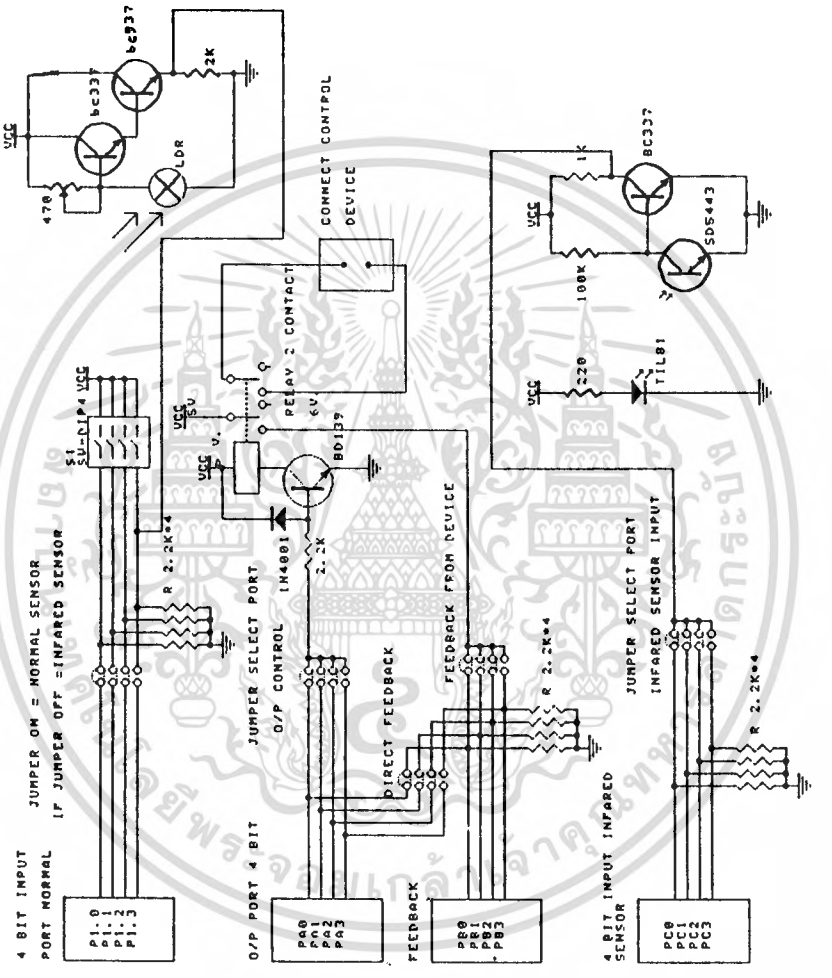
- U1 เป็น IC 8031 ทำหน้าที่ประมวลผล เป็น CPU ที่ Node Station
- U2 เป็น IC 74LS373 ทำหน้าที่เป็น Buffer ในการอ่านข้อมูลจะ EPROM
- U3 ใช้ IC เบอร์ 2764 หรือ 27256 ทำหน้าที่เป็น Program Monitor ของ Node Station
- U4 IC เบอร์ MAX232 ไว้ติดต่อสื่อสารในมาตรฐาน RS-232
- U5 IC เบอร์ 75176 ใช้ติดต่อสื่อสารในมาตรฐาน RS-422
- U6 IC เบอร์ 74LS154 เป็นตัว Decode Address ให้แก่ Port ใช้งานและ Memory
- U7เป็น IC เบอร์ 8155 ทำหน้าที่ขยาย Port ใช้งานให้แก่ Node Stationประกอบด้วย
- PORT A = Bidirectional 8 Bit
- PORT B = Bidirectional 8 Bit
- PORT C = Bidirectional 8 Bit

การทำงานของวงจรมีดังนี้

U1 (8031) จะเป็นตัวประมวลผลการทำงานตามโปรแกรม Monitor ใน U3 (2764) ซึ่งมีการทำงานปกติคืออ่านค่าสถานะของPort ทั้ง 3 Port ของ U7 (8155) เข้ามาเก็บไว้ใน Buffer อยู่ตลอดเวลา

เมื่อเกิดการร้องขอข้อมูลจาก Server Station จะมีสัญญาณเข้ามาทาง U5 (75176) มา Interrupt CPU ทางขา RxD ทำให้ CPU รู้ว่าขณะนี้มีการร้องขอข้อมูลก็จะทำการส่งข้อมูลที่อ่านได้จาก Port ออกมาทางขา TxD มาผ่าน U5 (75176) แล้วจึงส่งออกไปที่ Data Bus เพื่อไปเข้าที่ Server Station

3.4 วงจรใช้งานของการ์ดต่ออินเตอร์เฟส



พอร์ทที่ใช้งานของสถานีลูกมีดังนี้

- พอร์ท A ใช้ 4 บิทล่างเป็นเอาท์พุทพอร์ทคอนโทรล
- พอร์ท B ใช้ 4 บิทล่างเป็นอินพุทพอร์ทเพื่อรับ Feedback จากพอร์ท A
- พอร์ท C ใช้ 4 บิทล่างเป็นอินพุทพอร์ทที่มีสถานะคงที่ เช่น จากดอนแทคสวิทช์
- พอร์ท 1 ใช้ 4 บิทล่างเป็นอินพุทพอร์ทที่มีสถานะไม่คงที่ เช่น จากอินฟาเรดเซ็นเซอร์

** ข้อควรระวัง **

การต่ออินพุทเซ็นเซอร์ที่พอร์ทCกับพอร์ท1ถ้าเราเลือกต่อใช้ของตำแหน่งบิทนั้นๆ ในพอร์ทใดพอร์ทหนึ่งแล้ว ในตำแหน่งบิทนั้นของอีกพอร์ทต้องห้ามต่อ เช่น

- บิท 1 พอร์ท C ต่อไมโครสวิทช์
- บิท 1 พอร์ท 1 ต้องว่างไว้ห้ามต่อ

ซึ่งในการ์ดอินเตอร์เฟสจะมีจั้มเปอร์ให้เซทได้

3.5 ฟังก์ชั่นภาษาซีในการควบคุมพอร์ทอนุกรม

แสดงฟังก์ชั่นของภาษาซี ที่มีการควบคุมขาสัญญาณ RTS ในการกำหนดทิศทางว่าจะรับ หรือส่งข้อมูล

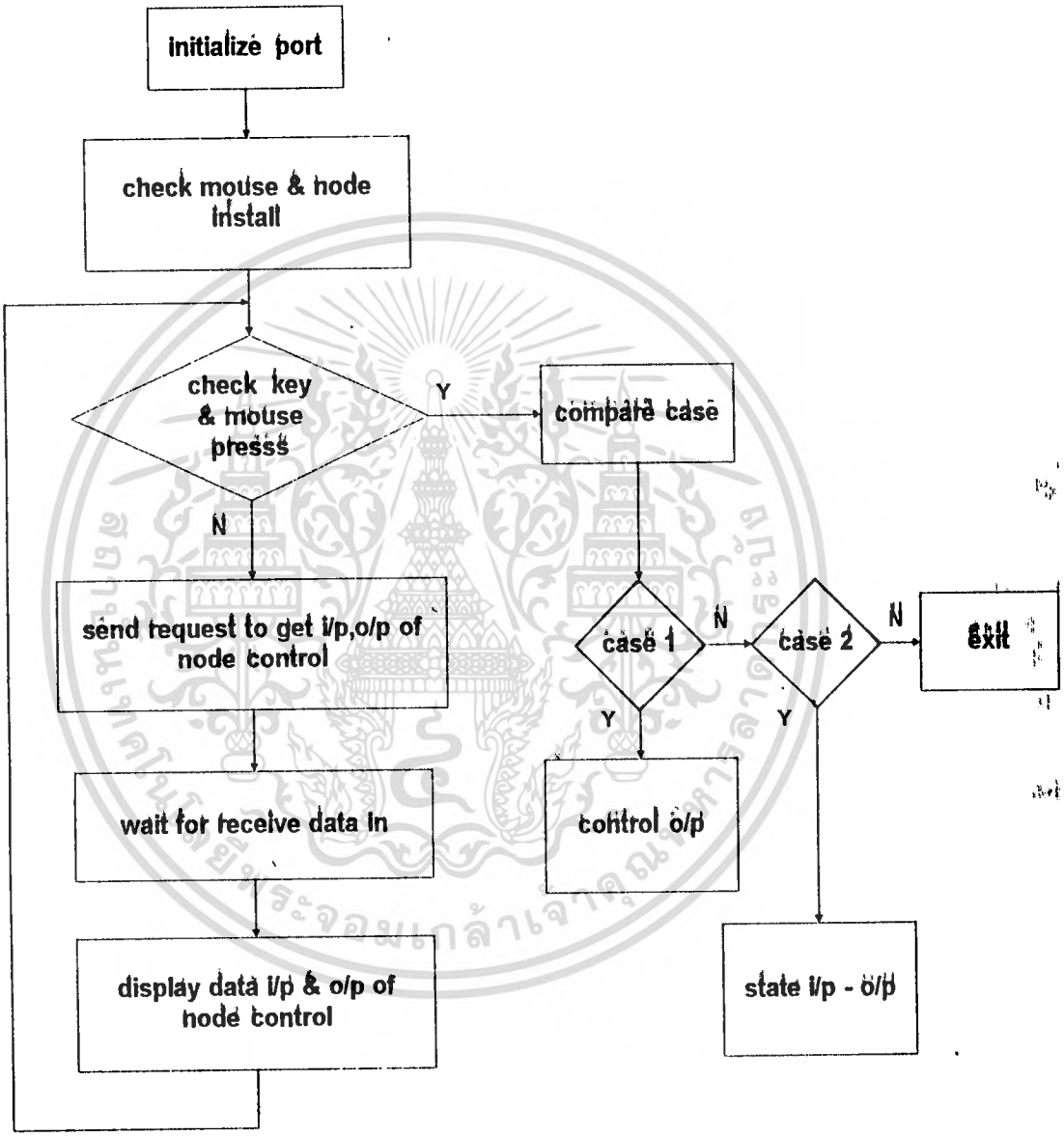
/* ROUTINE FOR SEND DATA TO SERIAL PORT */

```
void send_port(char dat_send)
{
unsigned far *ser_base1 = (unsigned far *)c_port1;
unsigned far *ser_base2 = (unsigned far *)c_port2;
if(Port==0)
{
outp((*ser_base1)+4,inp((*ser_base1)+4) | 0x02); /* logic '1' to RTS */
De_lay(Ray);
outp(*ser_base1,dat_send);
De_lay(Ray);
outp((*ser_base1)+4,inp((*ser_base1)+4) & ~0x02); /* logic '0' to RTS */
}
else
{
outp((*ser_base2)+4,inp((*ser_base2)+4) | 0x02);
        /*  set RTS to active to 75176 = active for send */
De_lay(Ray);
outp(*ser_base2,dat_send);
De_lay(Ray);
outp((*ser_base2)+4,inp((*ser_base2)+4) & ~0x02);
}
}
```

/* ROUTINE FOR RECEIVE DATA FROM SERIAL PORT */

```
r_port(int port)
{
unsigned far *ser_base1 = (unsigned far *)c_port1;
unsigned far *ser_base2 = (unsigned far *)c_port2;
char ina_code;
if(port==0)        /* NORMAL FOR RECEIVE */
 {
outp((*ser_base1)+4,inp((*ser_base1)+4) & ~0x02);
De_lay(Ray);
ina_code = inp(*ser_base1);
 }
else      /* RTS active to 75176 unactive for receive */
{
outp((*ser_base2)+4,inp((*ser_base2)+4) & ~0x02);
De_lay(Ray);
ina_code = inp(*ser_base2);
  }
return ina_code;
}
```

3.6 โฟล์วชาร์จ (Algorithm of Software)



รูปที่ 16 โฟลวชาร์จแสดงการทำงานของสถานีแม่ (Server Station)

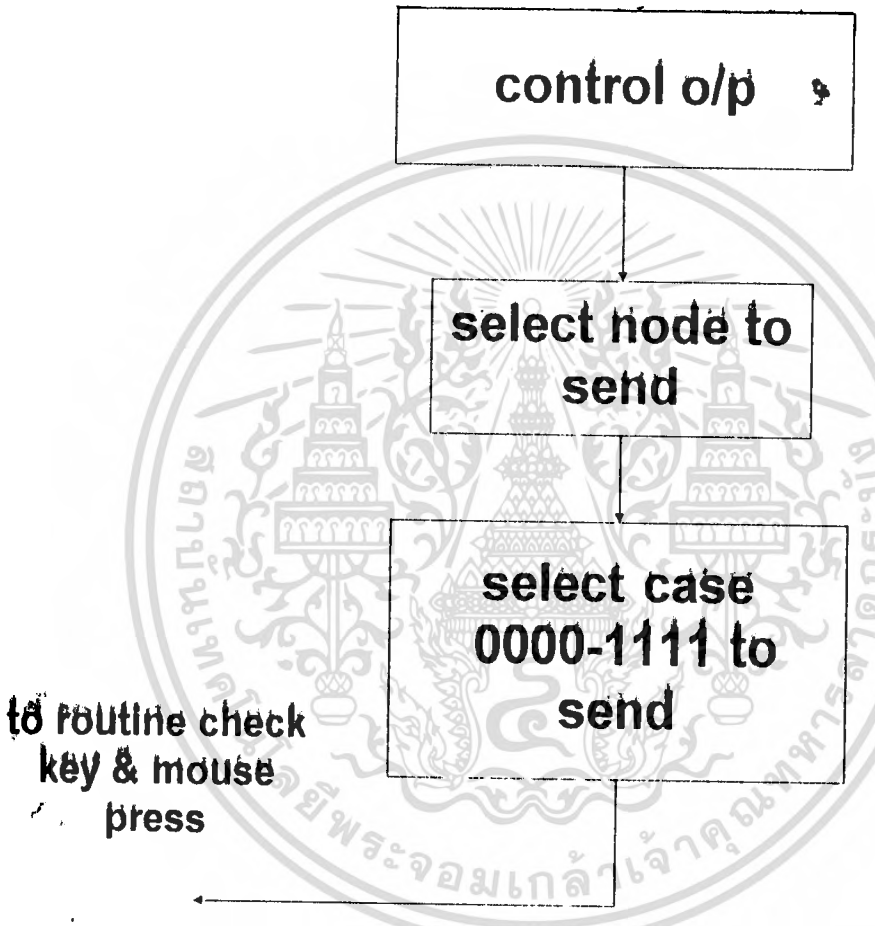
อธิบายการทำงานของ Flowchart ของสถานีแม่

เริ่มแรก Initial Port คือที่ Comport ของเครื่องคอมพิวเตอร์ต้องกำหนดรูปแบบการติดต่อว่าใช้ Baud Rate เท่าไร ข้อมูลที่ส่งเป็นกี่บิทจากนั้นจึงเช็คดูว่าต่อ Mouse อยู่ที่ Port หรือไม่ ถ้ามีการต่อจะใช้ได้แต่ถ้าหากไม่มีการต่อ Mouse จะมีการบอกให้รู้ว่าไม่ได้ต่อMouse อยู่ จากนั้นมาเช็คดูว่า Node Station ต่ออยู่ในระบบที่ Node ใด โดยทำการสแกนค้นหาไปตามลำดับ จากนั้นจะมี Dialog Box ขึ้นแสดงว่าในขณะนี้ต่ออยู่กี่โหนด

ใน Loop ปกติ ถ้าไม่มีการกด Keyboard หรือ Mouse จะอยู่ที่การส่งสัญญาณไปยังสถานะของอินพุตและเอาท์พุตพอร์ทของ Node Station ที่ต่อในระบบเข้ามาที่ Server Station จากนั้นก็เอาข้อมูลที่ได้มาแสดงที่หน้าจอแล้วจะวน Loop ตลอด

ถ้ามีการกด Key ใดๆ เกิดขึ้น ถ้าอยู่ใน Case ที่ต้องการคือ 1 หรือ 2 หรือ 3 ก็จะไปตาม Case นั้นๆ ดังนี้

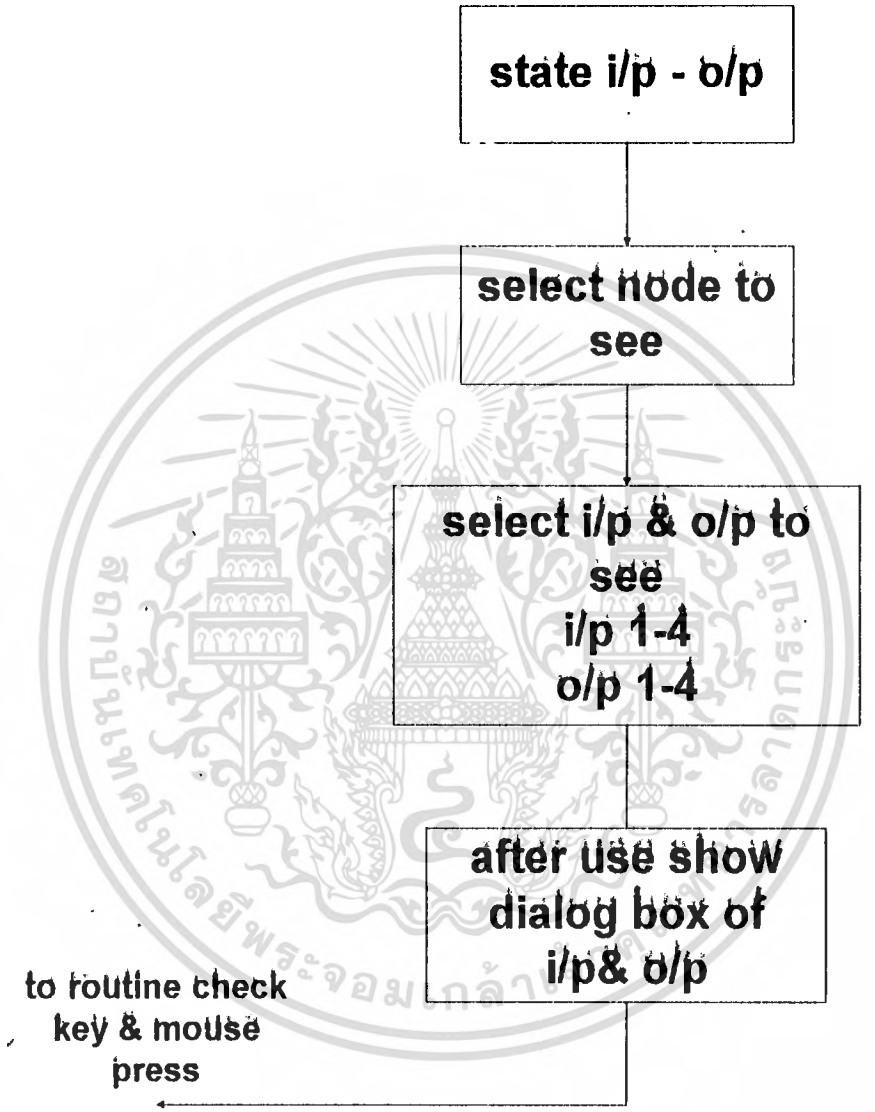
- Case 1 จะไปทำงานที่ Rountine Control O/P
- Case 2 จะไปทำงานที่ Rountine state I/P - O/P
- Case 3 จะออกจากโปรแกรม



รูปที่ 17 โฟลวชาร์จของ ROUNTINE CONTROL O/P

อธิบายการทำงานของโฟล์วชาร์จของ Routine Control O/P

ที่ Rountine Control O/P จะทำหน้าที่ควบคุมสถานะของเอาท์พุตพอร์ทของ Node Station ตามการเลือกของ User และสถานะที่จะให้เป็นก็ขึ้นอยู่กับการเลือกใช้ของ User เช่นกันโดยส่วนนี้จะเป็น Slide Menu Bar ต้องการตรงไหนก็ใช้การเลื่อนแถบสว่างสีไปแล้วกด Enter ก็จะทำงานตามต้องการจากนั้นจะกลับไปวน Loop ใน Main Function ใหม่



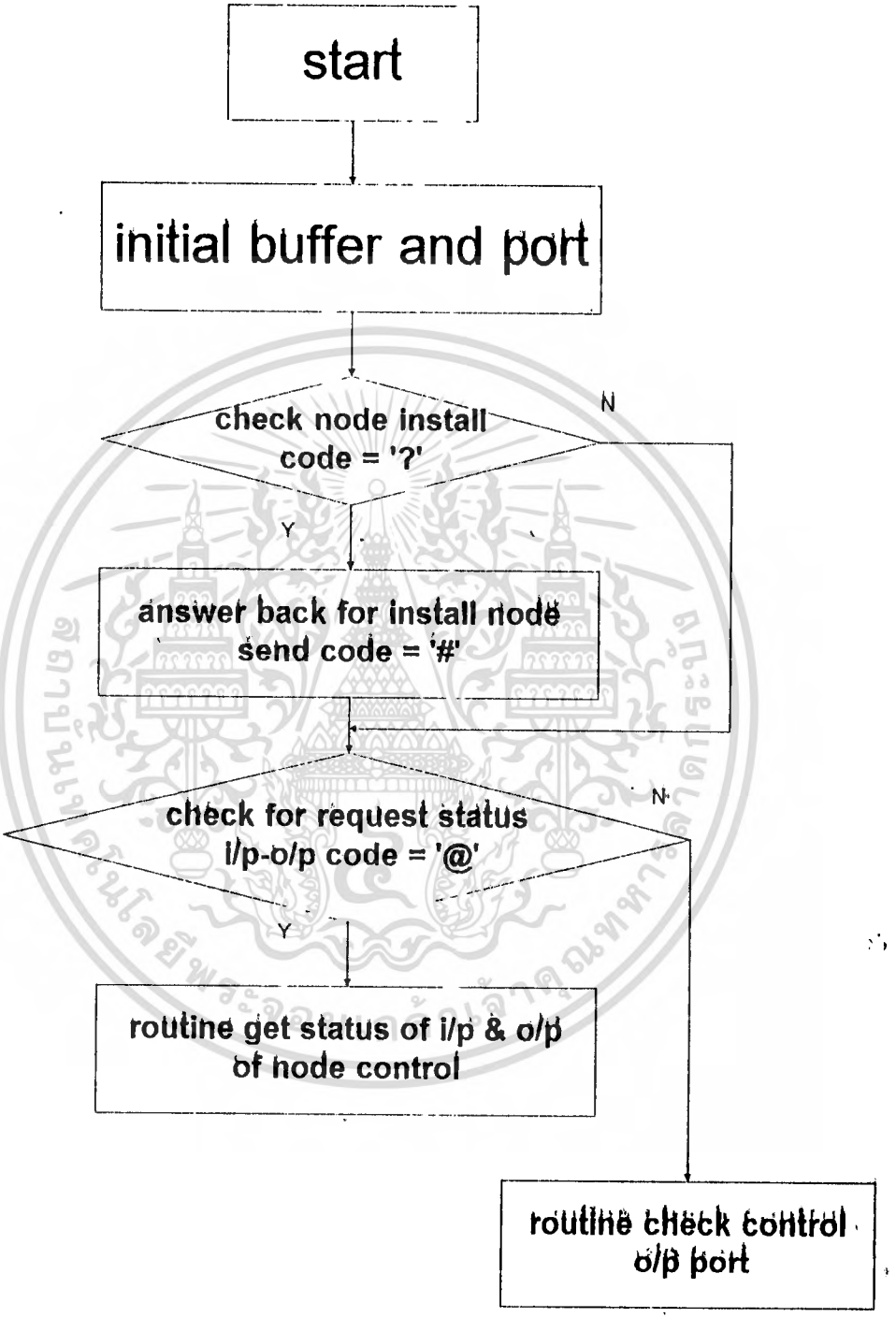
รูปที่ 18 โฟลวชาร์จของ ROUNTINE STATE I/P-O/P

อธิบายการทำงานโฟล์วชาร์จ Rountine State I/O

Rountine State I/P- O/P นี้เป็นการดูสถานะของ อินพุตหรือเอาท์พุตของ Node Station โดย

- อินพุตมีอยู่ 4 Bit
- เอาท์พุตมีอยู่ 4 Bit

ในขั้นแรกต้องเลือกเบอร์ Node ที่จะดูก่อน จากนั้นก็เลือกเบอร์ อินพุต หรือเอาท์พุตที่ต้องการจะดูเมื่อเลือกแล้วก็จะมี Dialog Box ขึ้นมาแสดงว่าตอนนี้ พอร์ทนั้นๆ ต่ออยู่ที่บริเวณใด และต่ออุปกรณ์อะไรอยู่บ้าง จากนั้นก็จะกลับสู่ Loop ใน Main Function

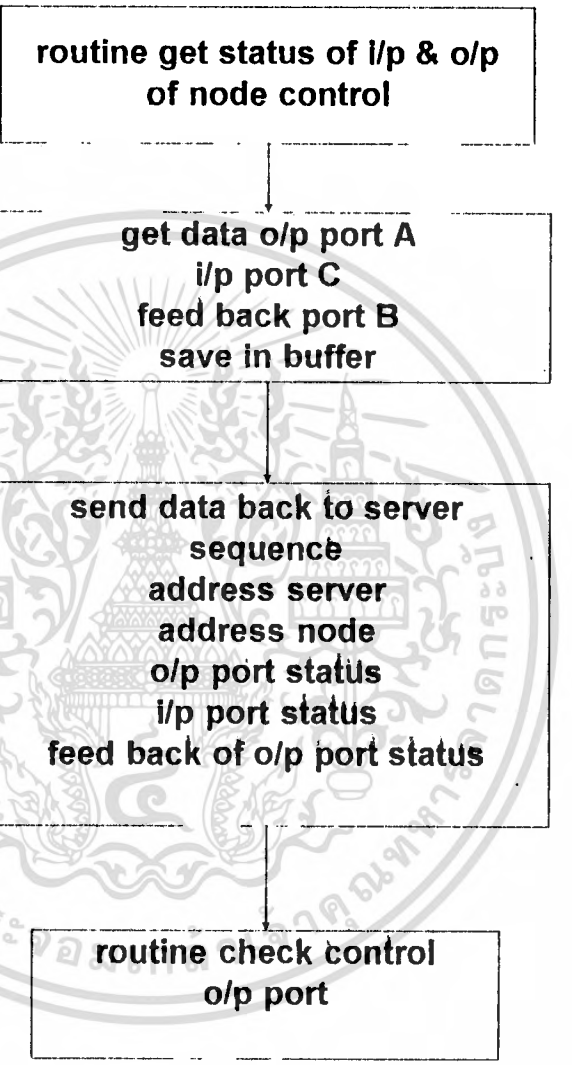


รูปที่ 19 โฟลวชาร์จแสดงการทำงานของสถานีลูก (Node Station)

การทำงานโฟลว์ชาร์จของสถานีลูก (Node Station)

เริ่มแรก Initial Buffer ที่จะเอาไว้เก็บสถานะพอร์ทก่อนจากนั้นจึงคอยเช็คดูว่ามีการส่งสัญญาณจาก Server มาถาม Node Install หรือไม่ ถ้ามีก็ตอบกลับไปด้วยสัญญาณตามกำหนด ถ้าหากไม่มีการถามก็ให้ข้ามขั้นตอนการตอบกลับไป

จากนั้นจึงทำการเช็คดูสัญญาณให้อ่านสถานะของอินพุต-เอาท์พุตพอร์ทหรือไม่ถ้าหากว่ามีก็ไปเข้า Rountine การอ่านสถานะพอร์ทเข้ามา แต่ถ้าไม่มีก็ไป Rountine ถามดูว่ามีการสั่งควบคุมสถานะของเอาท์พุตของ Node Station หรือไม่



รูปที่ 20 โฟลว์ชาร์จของ ROUNTINE การอ่านสถานะของ PORT

การทำงานของโฟลว์ชาร์จของ ROUNTIN การอ่านสถานะ PORT

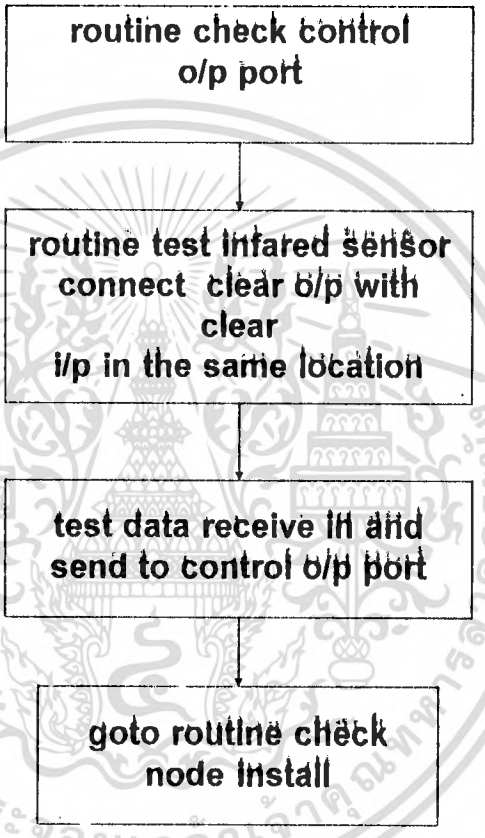
จะอ่านสถานะของ

- A PORT ที่เป็น O/P
- PORT C ที่เป็น I/P
- PORT B ที่เป็น Feedback PORT A

เข้ามาเก็บใน Buffer จากนั้นจะส่งข้อมูลนี้กลับไปให้ Server Station โดยเรียงรูปแบบของข้อมูลดังนี้

- ส่ง Address ปลายทาง
- ส่ง Address ต้นทาง
- ส่งสถานะO/P Port
- ส่งสถานะ I/P Port
- ส่งสถานะ Feedback Port

จากนั้นจะเข้าสู่ Rountine เช็คการควบคุม Port O/P



รูปที่ 21 โฟลว์ชาร์จการควบคุม PORT O/P

อธิบายการทำงานของโฟลว์ชาร์จการควบคุม Port O/P

เช็คดูว่าข้อมูลที่ส่งมาควบคุม O/P Port ของ Node นั้นๆ ไปตรงกับการกำหนด I/P Port แบบ Interface หรือไม่ถ้าตรงในกรณีที่เป็นการ Clear ให้ Clear I/P Port Interface นั้นด้วยและดูว่าที่ Output Port นั้นๆ ก็ให้มีสถานะเปลี่ยนแปลงตามการควบคุมด้วยจากนั้นกลับสู่การ Check Node Install ใหม่วน Loop อยู่เรื่อยไป

3.7 โปรโตคอลที่ใช้ในการติดต่อสื่อสาร

การส่งข้อมูลในรูปแบบ Asynchronous เราจะให้ค่าของความหมายในแต่ละ Character เพื่อกำหนดว่าค่าที่ส่งนั้นให้ทำอะไร ซึ่งแบ่งแยกได้ดังนี้

- "0" = แอดเดรสของสถานีลูกเบอร์ 1
- "1" = แอดเดรสของสถานีลูกเบอร์ 2
- "2" = แอดเดรสของสถานีลูกเบอร์ 3
- "3" = แอดเดรสของสถานีลูกเบอร์ 4
- "4" = แอดเดรสของสถานีลูกเบอร์ 5
- "T" = แอดเดรสของสถานีแม่
- "?" = โค้ดที่ใช้ถามถึง Node Install
- "@" = โค้ดที่ใช้อ่านสถานะของพอร์ทอินพุทและเอาท์พุทของสถานีลูก
- "v" = โค้ดจบหลังจากส่งคอนโทรลเอาท์พุทไปที่โหนด 5
- "w" = โค้ดจบหลังจากส่งคอนโทรลเอาท์พุทไปที่โหนด 4
- "x" = โค้ดจบหลังจากส่งคอนโทรลเอาท์พุทไปที่โหนด 3
- "y" = โค้ดจบหลังจากส่งคอนโทรลเอาท์พุทไปที่โหนด 2
- "z" = โค้ดจบหลังจากส่งคอนโทรลเอาท์พุทไปที่โหนด 1
- "A" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1111'
- "B" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1110'
- "C" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1101'
- "D" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1100'
- "E" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1011'
- "F" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1010'
- "G" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1001'
- "H" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '1000'
- "I" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0111'
- "J" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0110'
- "K" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0101'
- "L" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0100'
- "M" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0011'

- "L" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0100'
- "M" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0011'
- "N" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0010'
- "O" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0001'
- "P" = โค้ดควบคุมเอาท์พุทให้มีสถานะ '0000'
- "a" = โค้ดแสดงสถานะของอินพุทพอร์ท '1111'
- "b" = โค้ดแสดงสถานะของอินพุทพอร์ท '1110'
- "c" = โค้ดแสดงสถานะของอินพุทพอร์ท '1101'
- "d" = โค้ดแสดงสถานะของอินพุทพอร์ท '1100'
- "e" = โค้ดแสดงสถานะของอินพุทพอร์ท '1011'
- "f" = โค้ดแสดงสถานะของอินพุทพอร์ท '1010'
- "g" = โค้ดแสดงสถานะของอินพุทพอร์ท '1001'
- "h" = โค้ดแสดงสถานะของอินพุทพอร์ท '1000'
- "i" = โค้ดแสดงสถานะของอินพุทพอร์ท '0111'
- "j" = โค้ดแสดงสถานะของอินพุทพอร์ท '0110'
- "k" = โค้ดแสดงสถานะของอินพุทพอร์ท '0101'
- "l" = โค้ดแสดงสถานะของอินพุทพอร์ท '0100'
- "m" = โค้ดแสดงสถานะของอินพุทพอร์ท '0011'
- "n" = โค้ดแสดงสถานะของอินพุทพอร์ท '0010'
- "o" = โค้ดแสดงสถานะของอินพุทพอร์ท '0001'
- "p" = โค้ดแสดงสถานะของอินพุทพอร์ท '0000'
- "Z" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0000'
- "Y" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0001'
- "X" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0010'
- "W" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0011'
- "V" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0100'
- "U" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0101'
- "S" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0110'
- "R" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '0111'

- "Q" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1000'
- "/" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1001'
- "." = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1010'
- "," = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1011'
- ";" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1100'
- "[" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1101'
- "" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1110'
- ":" = โค้ดแสดงสถานะการย้อนกลับของเอาท์พุทพอร์ท A '1111'

3.7.1 รูปแบบการติดต่อโดยใช้รหัสที่กำหนดไว้มีดังนี้

3.7.1.1 การตรวจเช็ค การติดตั้งโหนด (Node Install)

เป็นการให้สถานีแม่ส่งรหัสไปถามสถานีลูกเช่นถามว่าNode1Install?จะมีการส่งข้อมูลดังนี้

- Send_port('0'); คือ หมายเลขเบอร์โหนด
- Send_port('?'); คือ รหัสถาม Install

ถ้าหากว่า Node 1 Install อยู่ในระบบ โหนด 1 จะตอบกลับมาที่สถานีแม่ดังนี้

- Send_port('T'); คือ แอดเดรสของสถานีแม่
- Send_port('#'); คือ รหัสตอบกลับว่าติดตั้งอยู่

3.7.1.2 การขอข้อมูลโดยสถานีแม่เรียกขอไปที่สถานีลูก

จะมีการส่งข้อมูลดังนี้

- Send_port('0'); คือ หมายเลขเบอร์โหนด
- Send_port('@'); คือ รหัสขอสถานะข้อมูลอินพุทและเอาท์พุท

ในกรณีที่โหนดนั้นๆต่ออยู่ในระบบก็จะส่งข้อมูลตอบกลับมาดังนี้

- ส่งรหัส 'T' เพื่อระบุตำแหน่งปลายทางคือสถานีแม่
- ส่งรหัสหมายเลขของตัวเอง เช่น ถ้าเป็น Node 1 จะส่งรหัส '0'
- ส่งรหัสที่เป็นสถานะของเอาท์พุทพอร์ท มี 'a','b',...,'p'

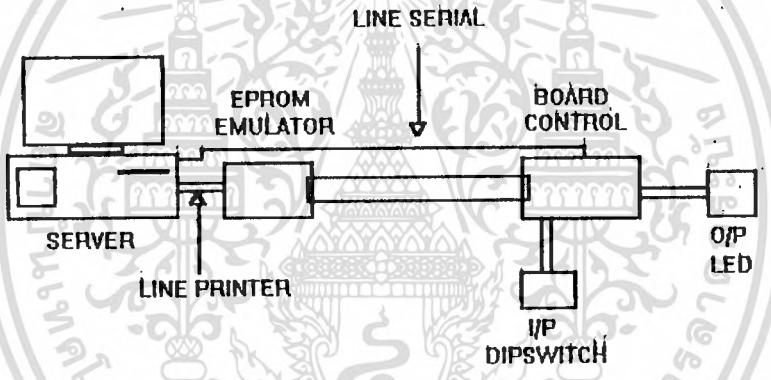
- ส่งรหัสที่เป็นสถานะของอินพุทพอร์ท มี 'a','b'...,'p'
- ส่งรหัสที่เป็นสถานะของฟีดแบคพอร์ท ตามรหัสที่กล่าวไว้ตอนต้นแล้ว

3.7.1.3 การส่งข้อมูลจากสถานีแม่ไปสถานีลูกเพื่อควบคุมเอาท์พุทพอร์ทมีดังนี้

- Send_port('0'); คือ หมายเลขเบอร์โหนด
- Send_port('A'); คือ รหัสควบคุมเอาท์พุทพอร์ทมี 'A','B'...,'P'
- Send_port('z'); คือ รหัสสิ้นสุดการควบคุมในแต่ละเบอร์โหนดนั้นๆ

บทที่ 4
การทดลอง

ในการทดลองและพัฒนาโครงงานต้องมีการติดต่อระหว่างเครื่องคอมพิวเตอร์กับบอร์ดควบคุมที่เป็นอุปกรณ์ไมโครคอลโทรลเลอร์8031อยู่ตลอดเวลาซึ่งที่บอร์ดต้องมีโปรแกรมภาษาแอสเซมบลีควบคุมอยู่ตลอดเวลาจึงพัฒนาผ่านเครื่องอีพรอมอีมูเลเตอร์ในการโหลดโปรแกรมโดยอุปกรณ์เซนเซอร์ใช้ดิฟสวิทช์ กับเอาท์พุทใช้ LED ดังแสดงในรูปที่ 22



รูปที่ 22 แสดง Diagram การต่อทดลองโครงงาน

เมื่อพัฒนาแล้วทำงานได้ตามต้องการก็จัดการอัดโปรแกรมมอนิเตอร์ของบอร์ดคอนโทรลลงอีพรอมจากนั้นพัฒนาบอร์ดอินเตอร์เฟสขึ้นมาเพื่อใช้ต่อกับอุปกรณ์เซนเซอร์ที่มีอินฟาเรดเซนเซอร์,อุปกรณ์ตรวจจับแสง (LDR) และไมโครสวิทช์ ส่วนเอาท์พุทคอนโทรลนั้นเราใช้ไซเรน

ซึ่งผลการทดลองก็ถูกต้องตามความต้องการทั้งหมดไม่ว่าจะเป็นในส่วนของการตั้งค่าคอมมานไลน์คือการตั้งค่าให้ทำงานแบบอัตโนมัติว่าสถานะของอินพุทเซนเซอร์เกิดเปลี่ยนแปลงไป จะมีผลต่อสถานะของเอาท์พุทพอร์ทอย่างไรในส่วนนี้ก็ทำงานได้ตามต้องการ

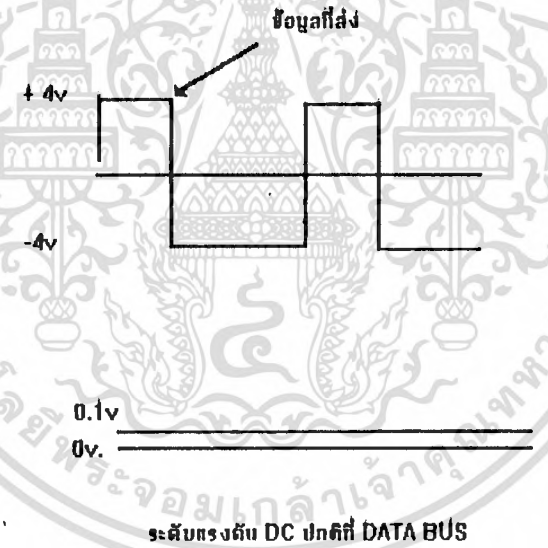
ตามทฤษฎีของมาตรฐาน RS-422 ว่ากำหนดให้ต่อโหนดได้ไม่เกิน 10 โหนด ระยะที่หวังผลสายสัญญาณที่ 1,200 เมตร

กำหนดบอร์ดเรทที่ 4,800 baud 3 โหนด

โดยการทดสอบผลการทดลองที่ได้มีดังนี้คือ

- ผลการทดลองที่ระยะ 10 เมตร ผลการติดต่อสื่อสารข้อมูลถูกต้องทุกประการ
- ผลการทดลองที่ระยะ 50 เมตร ผลการใช้งานก็ยังถูกต้องอยู่
- ผลการทดลองที่ระยะ 100 เมตร ผลการใช้งานก็ยังถูกต้องอยู่

ในขณะที่ทำการทดสอบใช้สโคปจับรูปสัญญาณที่สายบัสข้อมูลที่ระดับ(+4V,-4V) ลักษณะรูปของสัญญาณแสดงได้ดังต่อไปนี้



ระดับแรงดัน DC ปกติที่ DATA BUS

การต่อโหนดเข้ามาในระบบสภาวะที่บัสไอเดิลคือไม่มีการติดต่อส่งผ่านข้อมูลจะมีแรงดันประมาณ0.1VDCเมื่อเราต่อโหนดเพิ่มเข้ามา1ตัวจะทำให้แรงดันในสายข้อมูลลดลงประมาณ 0.01 V คือสถานะการทำงานถ้าระดับแรงดันใช้ Bus มากกว่าหรือเท่ากับ 0.01 VDC ระบบจึงสามารถที่จะ Detect สัญญาณนั้นได้หมายถึงถ้าหากต่อ Node ไม่เกิน Limit ของระบบถ้าหากมากเกินไป ระดับแรงดันลดต่ำลงกว่า 0.01 V จะ Detect สัญญาณในสายบัสไม่ได้

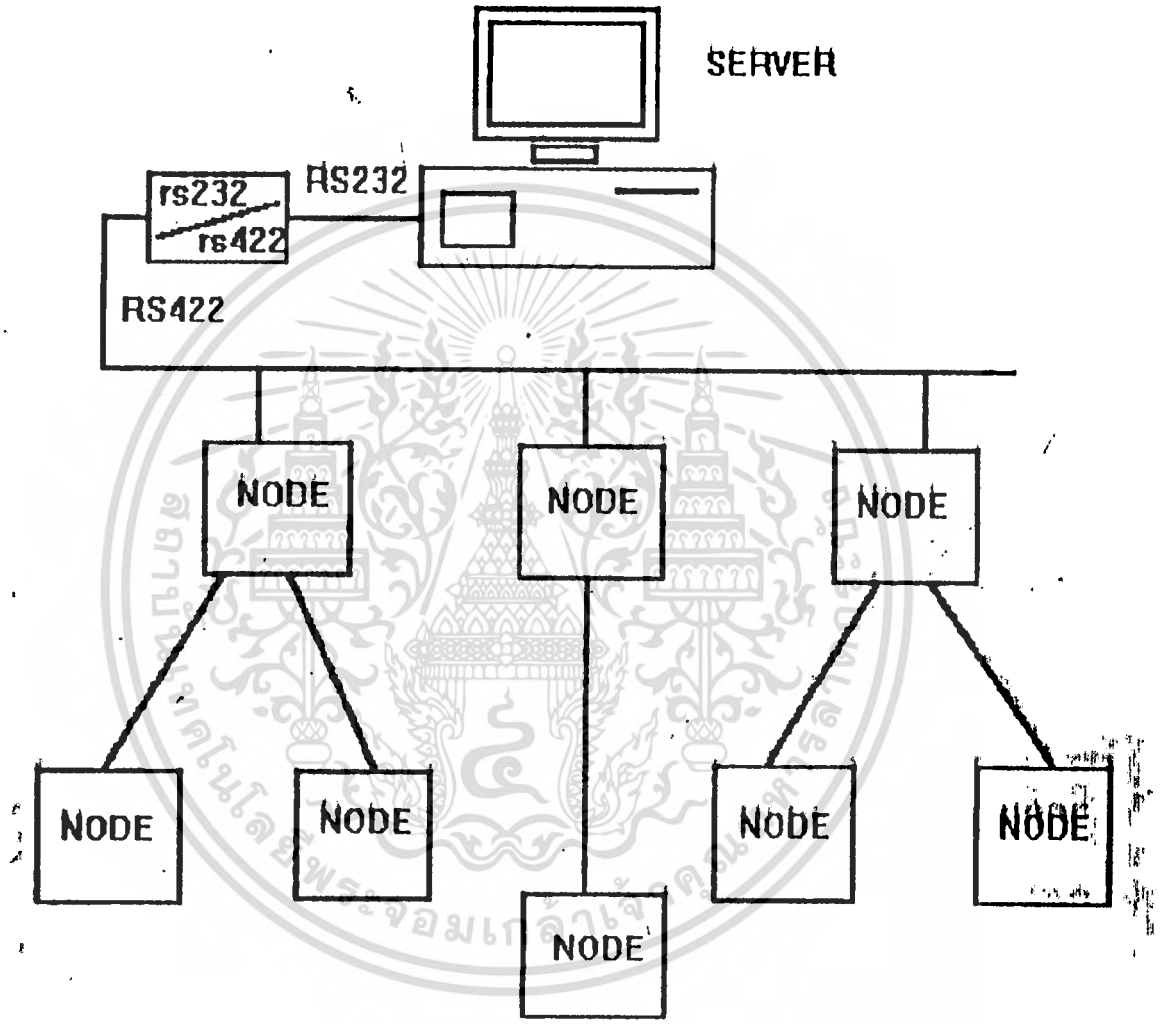
การป้องกันในระบบมีดังนี้

- สายบัสสื่อสารถ้าถูกตัดจะมีการแสดง Error ที่สถานีแม่พร้อมทั้งส่งเสียงเตือน
- อินพุทเซนเซอร์ที่ต่อถ้าเกิดตรวจจับได้ จะมีสถานะแสดงที่หน้าจอสถานีแม่พร้อมเสียงเตือน
- การปิดโหนด (Turn Off) ตัวสถานีแม่ก็จะแสดง Error ด้วย
- โหนดที่ต่อในระบบ (Install) ตามจำนวนในการเริ่มต้นตอนแรก สถานีแม่ก็จะรับรู้เพียงแค่นั้น ถ้าต่อเพิ่มในภายหลังก็จะมองไม่เห็น

บทที่ 5
สรุปผลและวิจารณ์

โครงงานนี้มีการใช้งานจริงซึ่งขึ้นอยู่กับการนำไปประยุกต์ใช้งาน ตัวอย่างเช่น ถ้าเราทำในลักษณะ Home Automation อุปกรณ์ Sensor I/F อาจจะเป็น Infared,Microswitch, LDR หรือ Sensorอุปกรณ์เอาท์พุทอาจเป็น Contact Relay On-Offจำพวก Motor, Air,Switch ไฟฟ้าต่างๆ

ในการใช้งานที่ 3 Node ที่มีความยาวสายรวมประมาณ 50 เมตร การใช้งานไม่เกิดข้อผิดพลาดเลย ไม่ทำให้เกิดการรบกวนของกระแสไฟฟ้าขณะที่กำลังใช้งานหากต้องการเปิดหรือปิดอุปกรณ์ไฟฟ้าอื่นๆ ก็ไม่มีผลต่อระบบ

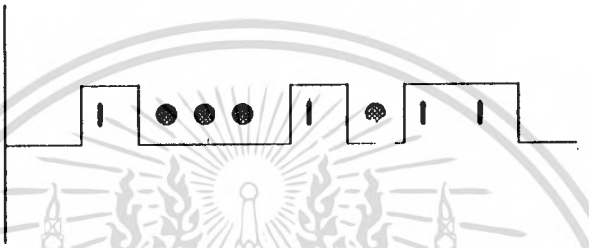


รูปที่ 23 แสดงการต่อขยายระบบเพื่อต่อเพิ่ม NODE

แนวทางในการพัฒนาระบบ

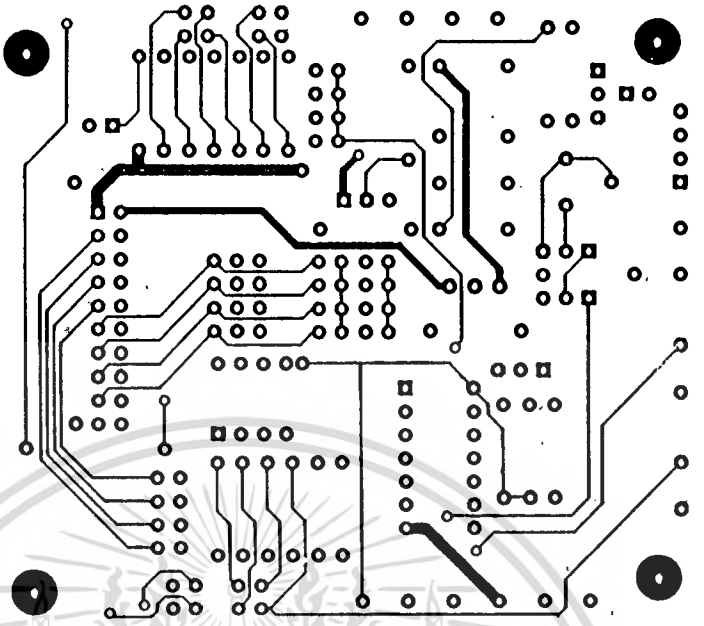
เราสามารถจะต่อขยายระบบได้ โดยต้องเพิ่มในส่วนของ Software และ Hardware ในส่วนของ Hareware ที่จะต้องพัฒนาต่อคือ

ต้องมีตัวทำหน้าที่ Repeater คล้ายตัว Line Drive เพื่อมาเพิ่มระดับแรงดันใน Bus ให้เพิ่มมากขึ้น กรณีการต่อ Node เพิ่มขึ้น และการต่อ Node ได้มากนั้นต้องมีการอ้าง Address มากขึ้น ปกติในการส่ง 1 ชุด จะใช้ข้อมูลขนาด 8 Bit ยกตัวอย่างเช่น

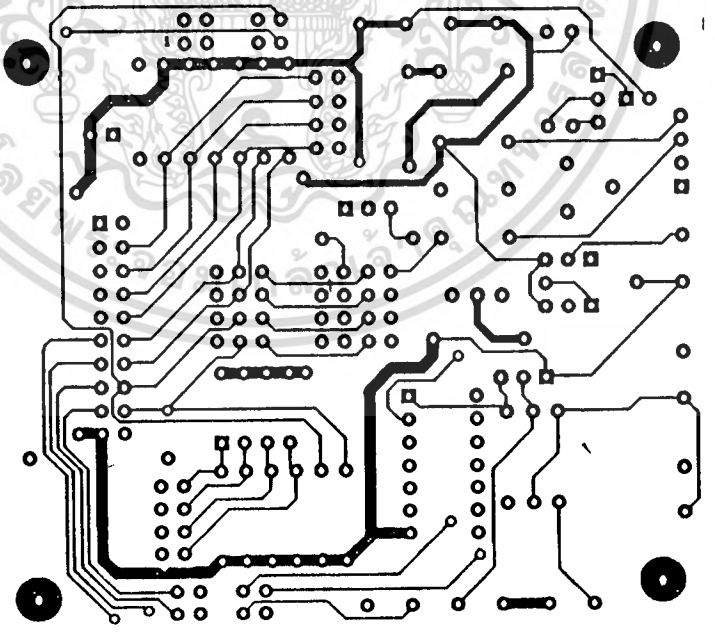


ถ้าเราต้องการเพิ่มระบบก็สามารถทำได้โดยจัดให้ Address ของ Node มีขนาดเป็น 16 บิตคือส่งข้อมูล 2 ชุดต่อการอ้าง Address 1 ครั้ง จะทำให้ต่อเพิ่มระบบได้เรื่อยๆ ถ้ามากขึ้นอีกก็อาจใช้ 24 Bit ก็ได้ ขึ้นอยู่กับว่าจะนำไปใช้งานในลักษณะใด

ภาคผนวก

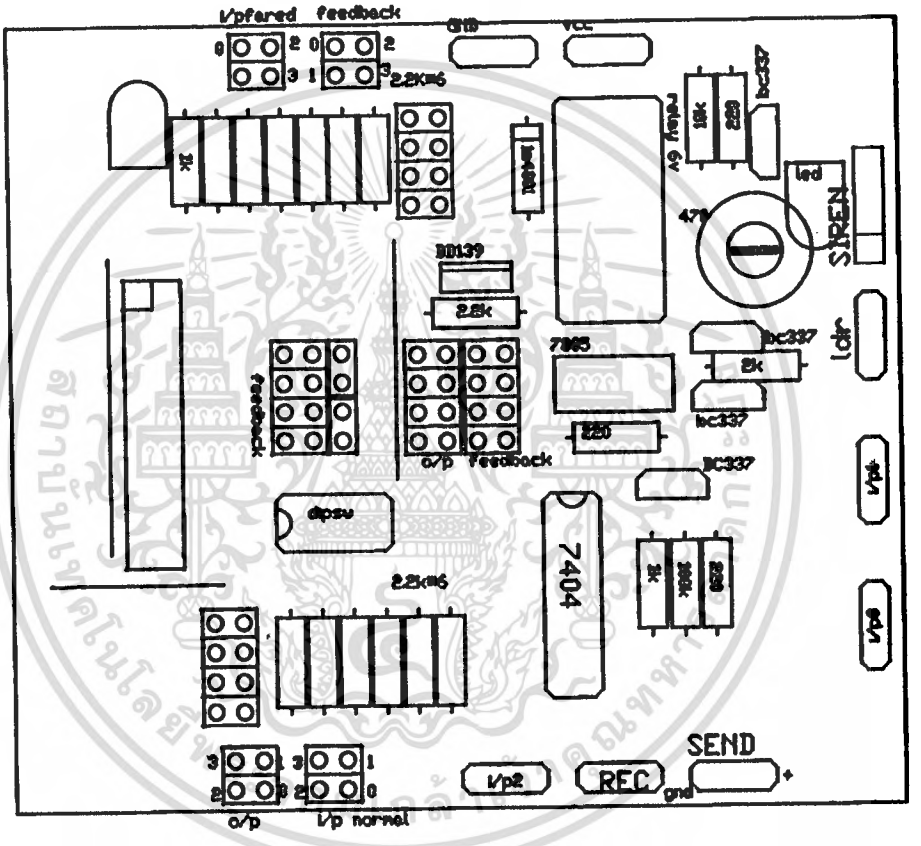


BOARD.PCB Top Layer

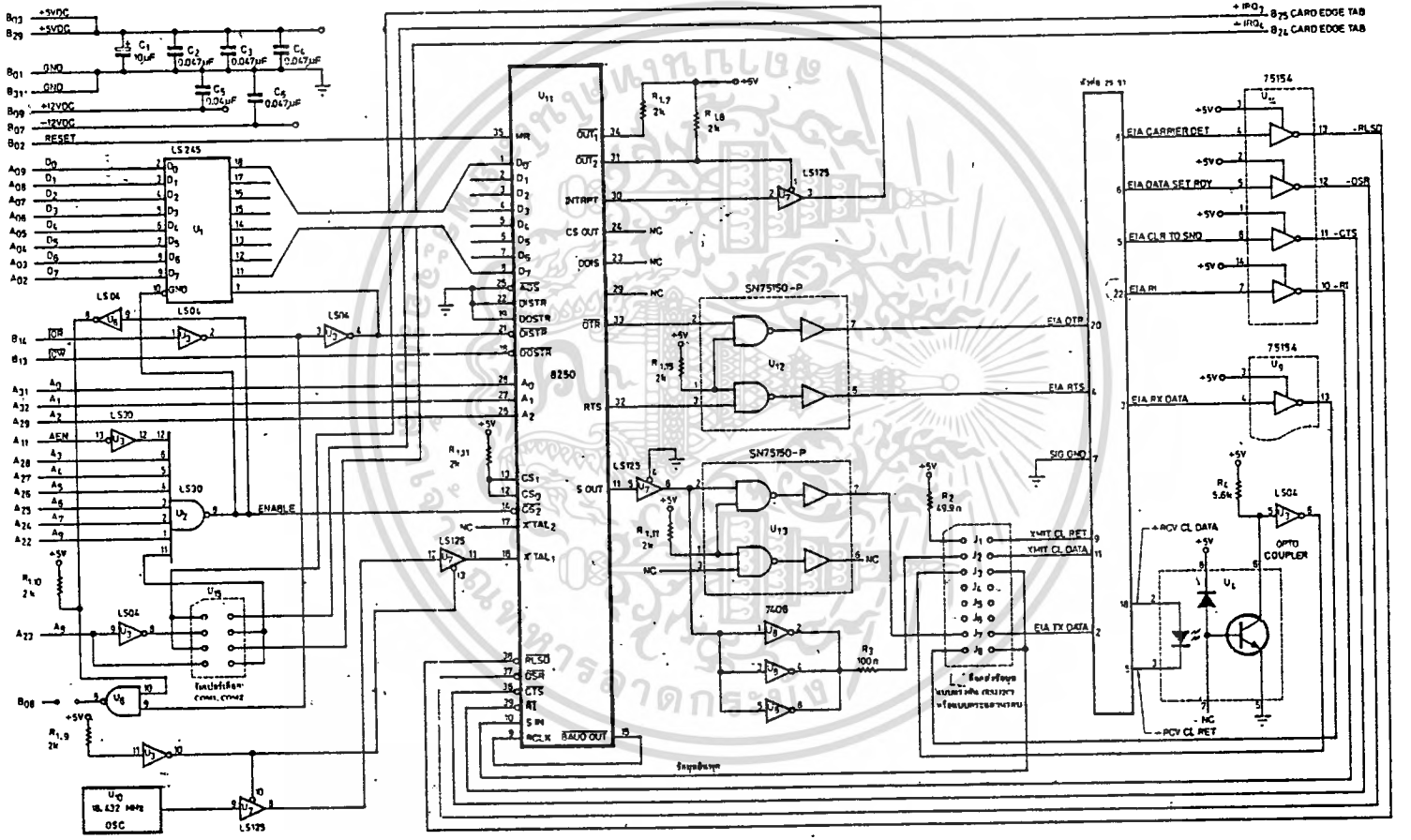


BOARD.PCB Bottom Layer

รูปแสดงลายปริ้นส์และการวางอุปกรณ์ของการ์ดอินเตอร์เฟส



BOARD Top Overlay



วงจรของบอร์ดอะแดปเตอร์สื่อสาร

intel

8155H/8156H/8155H-2/8156H-2
2048-BIT STATIC HMOS RAM WITH I/O PORTS AND TIMER

- Single +5V Power Supply with 10% Voltage Margins
- 30% Lower Power Consumption than the 8155 and 8156
- 100% Compatible with 8155 and 8156
- 256 Word x 8 Bits
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8-Bit I/O Ports
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/ Timer
- Compatible with 8085AH, 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8155H and 8156H are RAM and I/O chips implemented in N-Channel, depletion load, silicon gate technology (HMOS), to be used in the 8085AH and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085AH CPU.The 8155H-2 and 8156H-2 have maximum access times of 330 ns for use with the 8085AH-2 and the 5 MHz 8088 CPU.

The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

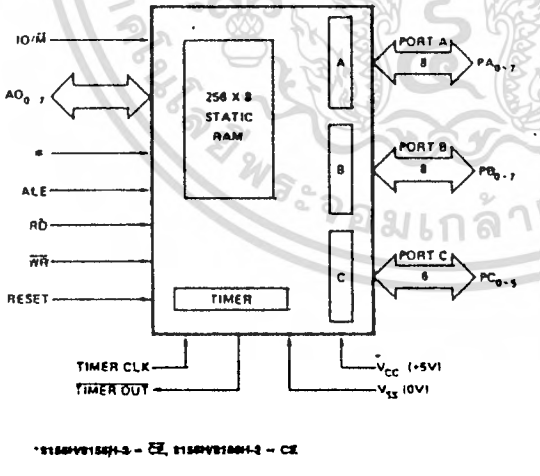


Figure 1. Block Diagram

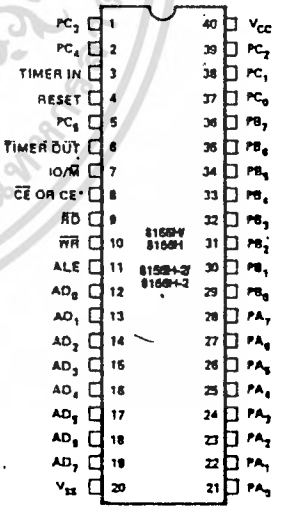


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Type	Name and Function
RESET	I	**Reset:** Pulse provided by the 8085AH to initialize the system (connect to 8085AH RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulse should typically be two 8085AH clock cycle times.
AD_{0-7}	I/O	**Address/Data:** 3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8155H/56H on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/$\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal.
CE or $\overline{CE}$	I	**Chip Enable:** On the 8155H, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156H, this pin is CE and is ACTIVE HIGH.
$\overline{RD}$	I	**Read Control:** Input low on this line with the Chip Enable active enables and AD_{0-7} buffers. If IO/$\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus.
$\overline{WR}$	I	**Write Control:** Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register, depending on IO/$\overline{M}$.
ALE	I	**Address Latch Enable:** This control signal latches both the address on the AD_{0-7} lines and the state of the Chip Enable and IO/$\overline{M}$ into the chip at the falling edge of ALE.
IO/$\overline{M}$	I	**I/O Memory:** Selects memory if low and I/O and command/status registers if high.
PA_{0-7}(8)	I/O	**Port A:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PB_{0-7}(8)	I/O	**Port B:** These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
PC_{0-5}(6)	I/O	**Port C:** These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When PC_{0-5} are used as control signals, they will provide the following: PC_0 — A INTR (Port A Interrupt) PC_1 — ABF (Port A Buffer Full) PC_2 — A $\overline{STB}$ (Port A Strobe) PC_3 — B INTR (Port B Interrupt) PC_4 — B $\overline{BF}$ (Port B Buffer Full) PC_5 — B $\overline{STB}$ (Port B Strobe)
TIMER IN	I	**Timer Input:** Input to the counter-timer.
$\overline{TIMER\ OUT}$	O	**Timer Output:** This output can be either a square wave or a pulse, depending on the timer mode.
V_{CC}		**Voltage:** +5 volt supply.
V_{SS}		**Ground:** Ground reference.

FUNCTIONAL DESCRIPTION

The 8155H/8156H contains the following:

- 2k Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The IO/$\overline{M}$ (IO/Memory Select) pin selects either the five registers Command, Status, PA_{0-7}, PB_{0-7}, PC_{0-5}, or the memory (RAM) portion.

The 8-bit address on the Address/Data lines, Chip Enable input CE or $\overline{CE}$, and IO/$\overline{M}$ are all latched on-chip at the falling edge of ALE.

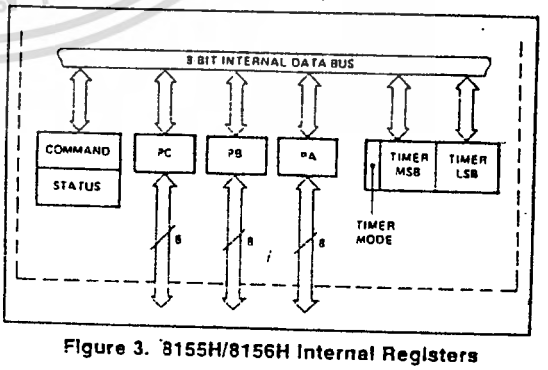


Figure 3. 8155H/8156H Internal Registers

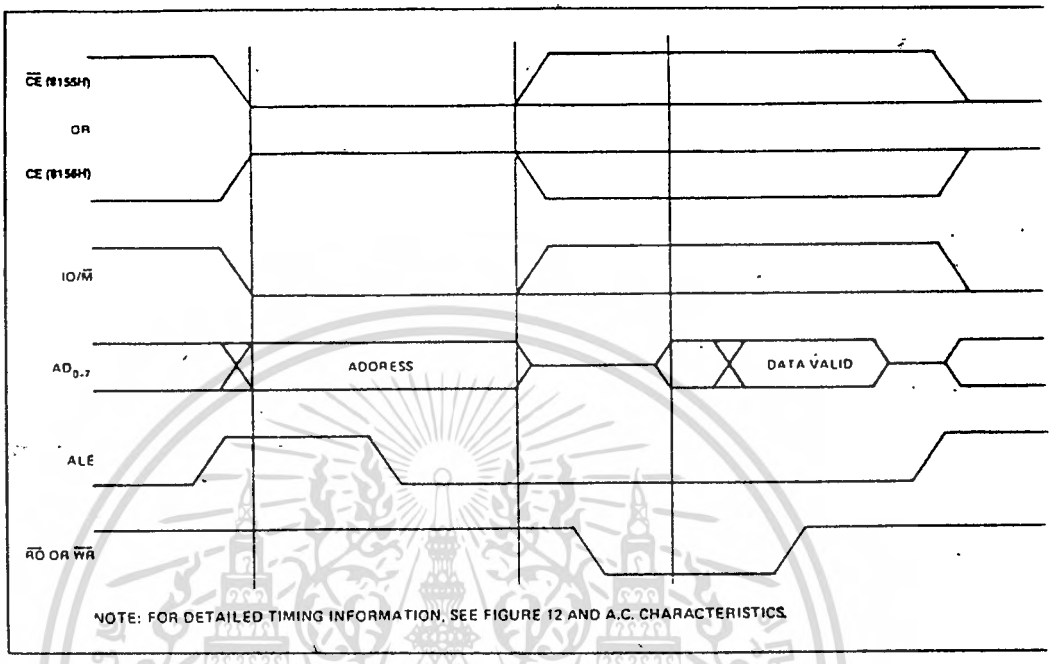


Figure 4. 8155H/8156H On-Board Memory Read/Write Cycle

PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits 0-3 define the mode of the ports, two bits 4-5 enable or disable the interrupt from port C when it acts as control port, and the last two bits 6-7 are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and IO/$\overline{M}$ = 1. The meaning of each bit of the command byte is defined in Figure 5. The contents of the command register may never be read.

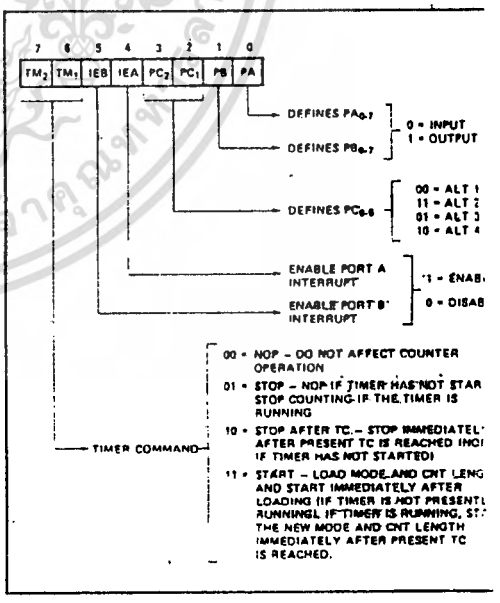


Figure 5. Command Register Bit Assignmen

READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit; six 0-5 for the status of the ports and one 6 for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 6. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

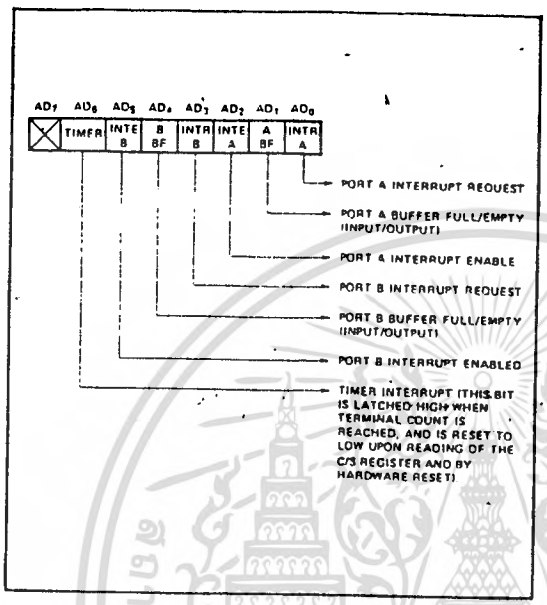


Figure 6. Status Register Bit Assignment

INPUT/OUTPUT SECTION

The I/O section of the 8155H/8156H consists of five registers: (See Figure 7.)

- **Command/Status Register (C/S)** — Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.

  When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins:

  When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the AD_{0-7} lines.

- **PA Register** — This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode (See timing diagram). The I/O pins assigned in relation to this register are PA_{0-7}. The address of this register is XXXXX001.

- **PB Register** — This register functions the same as PA Register. The I/O pins assigned are PB_{0-7}. The address of this register is XXXXX010.

- **PC Register** — This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports, output ports or as control signals for PA and PB by properly programming the AD_2 and AD_3 bits of the C/S register.

  When PC_{0-5} is used as a control port, 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155H sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. (See Table 2.)

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

CONTROL	INPUT MODE	OUTPUT MODE
BF	Low	Low
INTR	Low	High
STB	Input Control	Input Control

I/O ADDRESS†								SELECTION
A7	A6	A5	A4	A3	A2	A1	A0	
X	X	X	X	X	0	0	0	Interval Command Status Register
X	X	X	X	X	0	0	1	General Purpose I O Port A
X	X	X	X	X	0	1	0	General Purpose I O Port B
X	X	X	X	X	0	1	1	Port C — General Purpose I O or Control
X	X	X	X	X	1	0	0	Low-Order 8 bits of Timer Count
X	X	X	X	X	1	0	1	High 6 bits of Timer Count and 2 bits of Timer Mode

X Don't Care

†: I/O Address must be qualified by CE = 1 (8156H) or $\overline{CE}$ = 0 (8155H) and IO/$\overline{M}$ = 1 in order to select the appropriate register.

Figure 7. I/O Port and Timer Addressing Scheme

Figure 8 shows how I/O PORTS A and B are structured within the 8155H and 8156H:

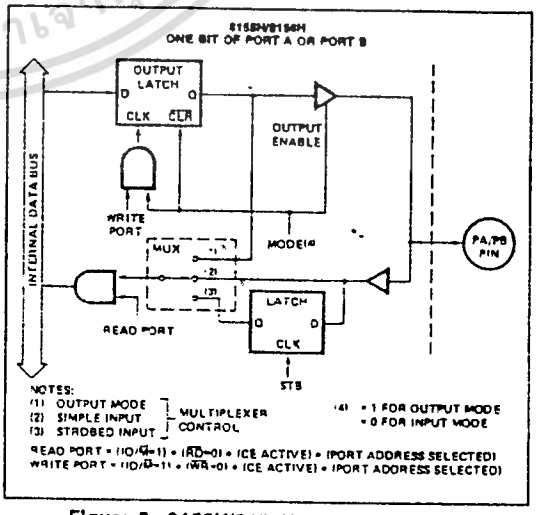


Figure 8. 8155H/8156H Port Functions

Table 2. Port Control Assignment

Pin	ALT 1	ALT 2	ALT 3	ALT 4
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)
PC2	Input Port	Output Port	A $\overline{STB}$ (Port A Strobe)	A $\overline{STB}$ (Port A Strobe)
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)
PC5	Input Port	Output Port	Output Port	B $\overline{STB}$ (Port B Strobe)

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155H/8156H are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pins will go low. When the 8155H/56H is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 9 shows how the 8155H/8156H I/O ports might be configured in a typical MCS-85 system.

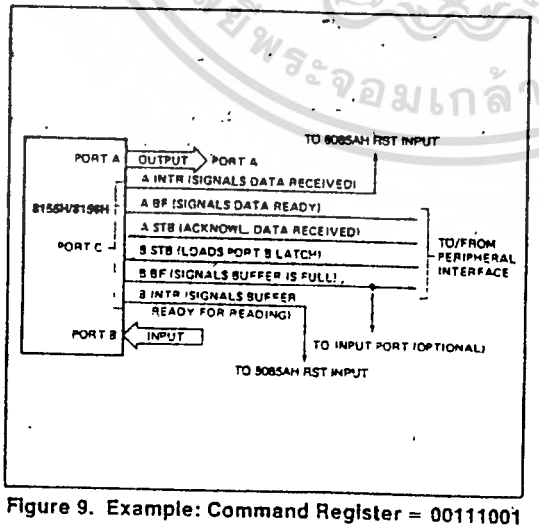


Figure 9. Example: Command Register = 00111001

TIMER SECTION

The timer is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 7.)

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0-13 of the high order count register will specify the length of the next count and bits 14-15 of the high order register will specify the timer output mode (see Figure 10). The value loaded into the count length register can have any value from 2H through 3FFFH in Bits 0-13.

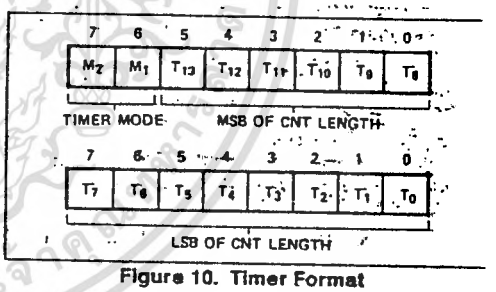


Figure 10. Timer Format

There are four modes to choose from: M2 and M1 determine the timer mode, as shown in Figure 11.

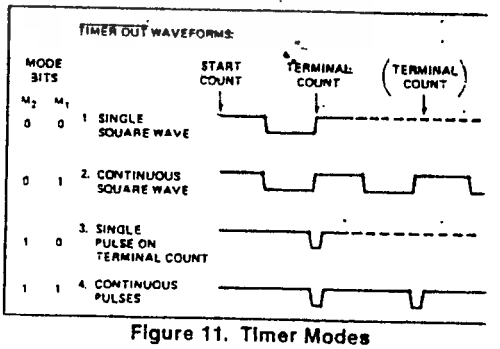


Figure 11. Timer Modes

Bits 6-7 (TM_2 and TM_1) of command register contents are used to start and stop the counter. There are four commands to choose from:

TM_2	TM_1	
0	0	NOP — Do not affect counter operation.
0	1	STOP — NOP if timer has not started; stop counting if the timer is running.
1	0	STOP AFTER TC — Stop immediately after present TC is reached (NOP if timer has not started)
1	1	START — Load mode and CNT length and start immediately after loading (if timer is not presently running). If timer is running, start the new mode and CNT length immediately after present TC is reached.

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you must issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second (low) half-cycle, as shown in Figure 12.

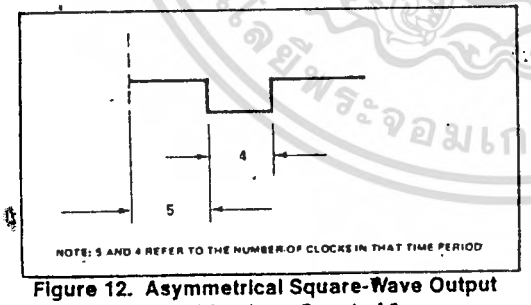


Figure 12. Asymmetrical Square-Wave Output Resulting from Count of 9

The counter in the 8155H is not initialized to any particular mode or count when hardware RESET occurs, but RESET does *stop* the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155H/8156H chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by twos twice in completing one cycle. Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 (binary) or 2 (decimal). (For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085AH be used.) After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count
2. Read in the 16-bit value from the count length registers
3. Reset the upper two mode bits
4. Reset the carry and rotate right one position all 16 bits through carry
5. If carry is set, add 1/2 of the full original count (1/2 full count — 1 if full count is odd).

Note: If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155H/56H always counts out the right number of pulses in generating the TIMER OUT waveforms.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงได้ด้วยดี ต้องขอขอบคุณท่านอาจารย์สมศักดิ์ มิตะถา เป็นอย่างยิ่งที่ได้ให้คำแนะนำต่าง ๆ ในการแก้ปัญหาตลอดการทำงาน และเพื่อน ๆทุกคน ที่คอยเป็นกำลังใจให้

หนังสืออ้างอิง

ยืน ภู่วรวรรณ, "เทคโนโลยีฮาร์แวร์ IBM PC", 2533, กรุงเทพมหานคร , บริษัทซีเอ็ดยูเคชั่น จำกัด.

ศิริชัย ดิลกรัตนพิจิตร, ณรงค์ศักดิ์ ตั้งกาญจนศรี, " เจาะหัวใจ MCS-51", ฉบับที่ 14, 2533, หน้า 104-114.

ศิริวัฒน์ ศิวะบวร, พรชัย จักรธำรงค์, จิรศักดิ์ ชัยวิริยะกุล, "การประยุกต์ใช้งานภาษาซี", สำนักพิมพ์ซีเอ็ด, 2535.

อาจหาญ สัตยารักษ์, "สร้างภาพด้วยกราฟฟิกส์ด้วยเมาส์", วารสารเซมิคอนดักเตอร์, ฉบับที่ 80, 2535, หน้า 289-296.

เอกชัย สันกำแพง, "วงจรแปลง RS-232 เป็น RS-422",วารสารเซมิคอนดักเตอร์, ฉบับที่ 115, 2535, หน้า 132-136.

DORGLAS V.HALL, "MICRO PROCESSOR AND INTERFACING PROGRAMMING AND HARDWARE", 1992, MC GRAW HILL.