โมเด็มไร้สาย

WIRELESS MODEM

โดย

นาย พิเชษฐ์ มัชฌิมาปรีชานนท์

นาย รัตนะ ถาจอหอ

นาย วรัญญู คมขำ

ปริญญานิพนธ์นี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาตร์บัณฑิต

สาขา เทคโนโลยีอิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

หัวข้อปริญญานิพนธ์ โมเด็มไร้สาย (WIRELESS MODEM)

โดย

นาย พิเชษฐ์ มัชฌิมาปรีชานนท์ เลขประจำตัว 35102017

นาย รัตนะ ถาจอหอ เลขประจำตัว 35102021

นาย วรัญญู คมขำ เลขประจำตัว 35102023

อาจารย์ที่ปรึกษา อ. ชวลิต เบญจางคประเสริฐ

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2536

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

.....กรรมการ

( )

โมเด็มไร้สาย

WIRELESS MODEM

โดย นาย พิเชษฐ์ มัชฌิมาปรีชานนท์
นาย รัตนะ ถาจอหอ
นาย วรัญญู คมขำ

อาจารย์ที่ปรึกษา อ. ชวลิต เบญจางคประเสริฐ

บทคัดย่อ

ในการแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์นั้น สิ่งหนึ่งที่จำเป็นต้องใช้คือ โมเด็ม ซึ่งทำหน้าที่แปลงสัญญาณดิจิตอลให้เป็นอะนาล็อก แล้วจึงส่งสัญญาณอะนาล็อกไปยังฝ่ายรับ ที่ฝ่ายรับก็มีโมเด็มทำการแปลงสัญญาณอะนาล็อกให้เป็นสัญญาณดิจิตอลเหมือนกับที่ฝ่ายส่งส่งมา แล้วคอมพิวเตอร์จะนำสัญญาณที่ได้ไปประมวลผลต่อไป อัตราเร็วในการส่งข้อมูลขึ้นอยู่กับเทคนิคการมอดดูเลทสัญญาณที่ใช้ในโมเด็ม สำหรับโครงงานนี้อัตราการส่งข้อมูลเป็น 300 บ็อด ใช้เทคนิคการมอดดูเลทสัญญาณแบบความถี่ ซึ่งใช้ในโมเด็มความเร็วต่ำ และสัญญาณที่ได้จะถูกมอดดูเลทอีกครั้งกับคลื่นพาห์ในเครื่องรับส่งวิทยุ ก่อนที่จะส่งออกอากาศไป และโมเด็มนี้เป็นการรับส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์

ABSTRACT

Communication between computer and others need exchange data. The equipment convert signal is modem. At the transmitter , a modem will modulate signal from digital to analog and demodulate signal from analog to digital at the receiver. This project , the construction of a modem system using frequency shift keying, FSK, for serial data transmission working 300 baud, half duplex. The baud rate refers to the number of bits per second which can be sent and received. Analog signal from modem is modulated with carrier signal in transmitter before broadcasting

กิตติกรรมประกาศ

คณะผู้จัดทำต้องขอบคุณ อาจารย์ ชวลิต เบญจางคประเสริฐ ที่ให้คำปรึกษาและคำแนะนำต่าง ๆ คุณ ธีรพล ทิพย์พรหม ที่จัดหาเครื่องมือวัดและอุปกรณ์ต่าง ๆ ไม่ว่าจะเป็นออสซิลโลสโคป ฟรีเควนซีเคาน์เตอร์และสเตบิล็อก คุณ ชาลี สุขเทศ ที่ช่วยแนะนำและแก้ไขโปรแกรมที่ใช้ควบคุมการรับส่งข้อมูล

คณะผู้จัดทำขอขอบคุณทุกท่านที่มีส่วนช่วยเหลือไว้ ณ ที่นี้ด้วย

คณะผู้จัดทำ

เมษายน 2537

สารบัญ

บทที่ 1
บทนำ

ในยุคปัจจุบันนี้ได้มีการนำเอาเทคโนโลยีของคอมพิวเตอร์ ไปประยุกต์ใช้ในงานต่าง ๆ จนแพร่หลายมากขึ้นทั้งในงานอุตสาหกรรม การแพทย์ วิศวกรรมและในระบบสำนักงาน หรือที่เรียกว่า ระบบสำนักงานอัตโนมัติ (Office Automation) เมื่อมีการใช้งานมากขึ้น จำนวนข้อมูลที่ใช้ก็เพิ่มมากขึ้นตามไปด้วย จึงมีความต้องการข้อมูลจากที่อื่น ๆ มาใช้อันเป็นที่มาของการสื่อสารข้อมูลระหว่างกัน

สมัยก่อนการติดต่อระหว่างกันมักใช้จดหมาย หรือ เอกสารส่งไปให้ปลายทาง ต่อมาพัฒนามาใช้โทรสารแทนซึ่งรวดเร็วขึ้นและสะดวกกว่าแต่ก่อน แต่ก็ยังไม่ทำให้คอมพิวเตอร์รับส่งข้อมูลกับที่ต่าง ๆ ได้โดยตรง ถ้าผู้รับต้องนำข้อมูลนี้ไปใช้งานในเครื่องคอมพิวเตอร์ ก็ต้องป้อนเข้าเครื่องอีกทีหนึ่ง นอกจากจะชักช้าและเสียเวลาแล้ว ยังอาจเกิดข้อผิดพลาดได้อีกด้วย ข้อจำกัดอันนี้ทำให้การสื่อสารข้อมูลมีความสำคัญขึ้นมาทันที

จะเห็นว่าคอมพิวเตอร์มีบทบาทอย่างมากคู่กับการสื่อสารข้อมูลแยกออกจากกันไม่ได้ ถ้าขาดคอมพิวเตอร์ไปการสื่อสารข้อมูลจะไม่เกิดขึ้น โดยปกติการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ วิธีที่ง่ายที่สุดคือ การรับส่งข้อมูลผ่านสายเคเบิลโดยตรง ถ้าระยะทางไม่ห่างกันมากนัก หรืออาจรับส่งข้อมูลผ่านระบบเครือข่ายของคอมพิวเตอร์เอง เช่น LAN (Local Area Network),WAN (Wide Area Network)หรือ PDN (Public Data Network) ซึ่งเป็นการรับส่งข้อมูลระหว่างกันโดยผ่านโมเด็ม (MODEM) ไปตามสายโทรศัพท์ สามารถส่งข้อมูลไปได้ไกลทั่วโลกเท่าที่ระบบโทรศัพท์เข้าไปถึง

แต่ในการรับส่งข้อมูลตามสายยังมีข้อจำกัด ในเรื่องการรับส่งข้อมูลที่เพิ่มมากขึ้น ทำให้การเดินสายยุ่งยากและการเคลื่อนย้ายทำได้ลำบาก นอกจากนี้ในบางพื้นที่ที่การวางโครงข่ายโทรศัพท์ยังเข้าไม่ถึงก็จะไม่สามารถสื่อสารข้อมูลได้ ดังนั้นเพื่อแก้ปัญหาดังกล่าว โครงงานนี้จึงใช้การสื่อสารข้อมูลผ่านคลื่นวิทยุ โดยเป็นการใช้งานในการรับส่งข้อมูลระหว่างคอมพิวเตอร์ที่อยู่ในพื้นที่เดียวกัน

บทที่ 2
ทฤษฎีและหลักการเบื้องต้น

2.1 การสื่อสารข้อมูล

การสื่อสารข้อมูลคือ ขบวนการในการแลกเปลี่ยนข้อมูลหรือข่าวสาร ซึ่งในการสื่อสารก้นนั้นต้องประกอบไปด้วย ผู้ส่ง (Sender) ผู้รับ (Receiver) และตัวกลางในการส่งข้อมูล (Medium) โดยที่ข้อมูลที่จะส่งนั้นเป็นสัญญาณดิจิตอล คืออยู่ในรูปเลขฐานสองข้อมูลข่าวสารที่จะส่งจะเป็นรหัสตัวอักษร ตัวเลข หรือเครื่องหมายที่อยู่ในรูปเลขฐานสอง เช่น รหัสแอสกี (ASCII) หรือ รหัส (EBCDIC)

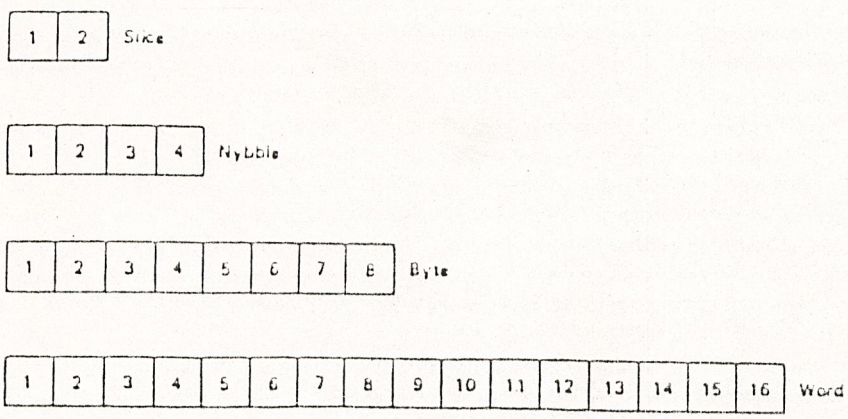
ในการสื่อสารข้อมูลจะเป็นการรับส่งข้อมูลระหว่าง 2 จุดหมาย เช่น การเชื่อมโยงระหว่างคอมพิวเตอร์ 2 เครื่องเข้าด้วยกันในระยะทางใกล้ ๆ ด้วยสายธรรมดา ข่าวสาร หรือ Encoded Information คือ ข้อมูลที่ถูกถ่ายโอนไปในรูปของสัญญานไฟฟ้าจะถูกส่งออกไปโดยผู้ส่ง เช่นเมื่อเรากดตัวอักษร D บนคีย์บอร์ดซึ่งมีรหัสในรูปเลขฐานสองคือ 01000100 จะถูกส่งผ่านสายนำสัญญาณไปยังผู้รับ โดยจะทำการแปลงรหัสเลขฐานสองนี้ให้กลับเป็นอักษร D ซึ่งข้อมูลที่ส่งนี้จะส่งในรูปของสัญญาณอะนาล็อก หรือ ดิจิตอลก็ได้

ถ้าเป็นการติดต่อแลกเปลี่ยนข่าวสารข้อมูลที่มากกว่า 2 จุดหมายขึ้นไปจะทำให้เกิดการสื่อสารข้อมูลในลักษณะของระบบเครือข่ายคอมพิวเตอร์ ถ้าเป็นเครือข่ายที่อยู่ในบริเวณเดียวกันเราเรียกว่า Local Area Network (LAN) มีรูปแบบของเครือข่ายหลายแบบตามความเหมาะสมของพื้นที่นั้น ๆ เช่น แบบสตาร์ หรือ แบบริง เป็นต้น โดยสามารถที่จะต่อสายเข้าด้วยกันระหว่างเทอร์มินอลได้ง่าย แต่ถ้าเป็นการสื่อสารข้อมูลในระยะทางไกล ๆ หรือที่เรียกว่า Wide Area Network (WAN) จะไม่นิยมเดินสายเชื่อมโยงระหว่างเทอร์มินอล เนื่องจากจะสิ้นเปลืองค่าใช้จ่ายในการวางสาย และปัญหาอันเนื่องมาจากสภาพความเป็นตัวเก็บประจุในสาย หรือ Stray Capacitance

เราสามารถแก้ไขได้โดยใช้เครือข่ายชนิดอื่น ๆ ที่มีอยู่แล้ว เช่น เครือข่ายโทรศัพท์ และเครือข่ายวิทยุมาทำการส่งข้อมูลได้ เช่น ในธนาคารพาณิชย์ จะมีการส่งข่าวสารข้อมูลระหว่างคอมพิวเตอร์เมนเฟรมที่สำนักงานใหญ่กระจายไปตามเทอร์มินอลที่สาขาต่าง ๆ โดยผ่านเครือข่ายโทรศัพท์ หรือการเชื่อมโยงด้วยคลื่นไมโครเวฟ และการสื่อสารผ่านดาวเทียม เป็นต้น

2.2 ประเภทของการส่งข้อมูล

โดยทั่ว ๆ ไปหลักใหญ่ของการส่งข้อมูลในคอมพิวเตอร์ หรือ ระหว่างคอมพิวเตอร์ด้วยกันจะมีลักษณะของการส่งข้อมูลอยู่ 2 แบบ คือ การส่งข้อมูลแบบขนาน และการส่งข้อมูลแบบอนุกรม คำสั่งหรือข้อมูลอยู่ในรูปของบิท คือหลาย ๆ บิทประกอบกันเป็นคำ ๆ หนึ่ง (word) หรือ คำสั่งหนึ่ง ๆ ดังในรูป 2.1 ได้แสดงถึงกลุ่มของบิทที่มีการใช้งานในไมโครคอมพิวเตอร์ จึงได้มีการกำหนดลักษณะมาตรฐานของข้อมูลคือ



รูปที่ 2.1 แสดงรูปแบบของข้อมูล

ถ้าข้อมูลหนึ่งตัวเมื่อแปลงให้อยู่ในรูปของเลขฐานสองแล้วประกอบด้วย 4 บิท เราเรียกว่า 4 บิทไมโคร หรือ 1 นิบเบิล (nybble) และถ้าข้อมูลประกอบไปด้วยกลุ่มของบิทที่มี 8 บิท เราเรียกว่าเป็น 1 ไบท์ (byte) แต่ในระบบอื่นอาจจะมี 16 บิท หรือ 32 บิท เป็น 1 ไบท์ ก็ได้ ดังนั้นเมื่อเรารู้ลักษณะของข้อมูลแล้วเราจะมาดูข้อแตกต่างของการส่งข้อมูลแบบขนานและแบบอนุกรม

1. การส่งข้อมูลแบบขนาน

ลักษณะของการส่งข้อมูลแบบขนาน หรือ เรียกอีกอย่างหนึ่งว่า Parallel Interface ทำได้โดยการส่งข้อมูลออกมาทีละไบท์ ซึ่งถ้าใน 1ไบท์มี 8 บิท ทั้ง 8 บิท จะถูกส่งจากอุปกรณ์ส่งไปอุปกรณ์รับพร้อมกัน และช่องสัญญาณในการส่งข้อมูลจะต้องมีอย่างน้อย 8 ช่องสัญญาณสำหรับสัญญาณแต่ละบิท พร้อมกับมีสัญญาณควบคุมอีกหลายเส้น ช่องสัญญาณในการส่งจะใช้สายเคเบิลแบบที่มีตัวนำหลายเส้น อาจจะเป็นสายเคเบิลชนิดแบนหรือกลมก็ได้ โดยที่ระยะทางระหว่างเครื่องทั้งสองไม่ควรมากเกินไป เนื่องจากสาเหตุต่าง ๆ หลายสาเหตุ เช่น การที่สัญญาณถูกลดทอนไปกับความต้านทานของสาย การผิดเพี้ยนของสัญญาณเนื่องจากสภาพความเป็นตัวเก็บประจุภายในสาย การเดินทางมาถึงอุปกรณ์รับไม่พร้อมกันของแต่ละบิท เพราะสภาพความไม่สมดุลย์ของตัวนำแต่ละเส้นภายในสายเคเบิล และการที่ระดับของกราวนด์ทางไฟฟ้าที่อุปกรณ์รับผิดไปจากอุปกรณ์ส่ง สาเหตุเหล่านี้อาจจะทำให้เกิดการผิดพลาดของข้อมูลขึ้นได้ นอกจากสายที่เป็นทางเดินของข้อมูลแล้ว อาจจะมีทางเดินของสัญญานควบคุมต่าง ๆ อีก เช่น สายที่ใช้ในการควบคุมในการโต้ตอบ (Handshake) ซึ่งการส่งข้อมูลแบบขนานจะใช้ในการส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์เป็นส่วนใหญ่ มีข้อดีคือสามารถส่งข้อมูลได้อย่างรวดเร็วและเป็นจำนวนมาก ข้อเสียคือ ไม่เหมาะที่จะใช้ส่งข้อมูลเป็นระยะทางไกล ๆ เนื่องจากค่าใช้จ่ายในเรื่องของสายสำหรับส่งข้อมูลมีราคาแพงเกินไป

2. การส่งข้อมูลแบบอนุกรม

มูลเข้ามาทีละบิทแล้วมารวมกันเป็นไบท์ ซึ่งทางด้านอุปกรณ์รับจะต้องคอยตรวจสอบว่า บิทใดเป็นบิทเริ่ม แรกของไบท์นั้น การตรวจสอบขึ้นอยู่กับรูปแบบของรหัสของบิทที่ใช้

การส่งข้อมูลแบบอนุกรมมี 2 แบบคือ

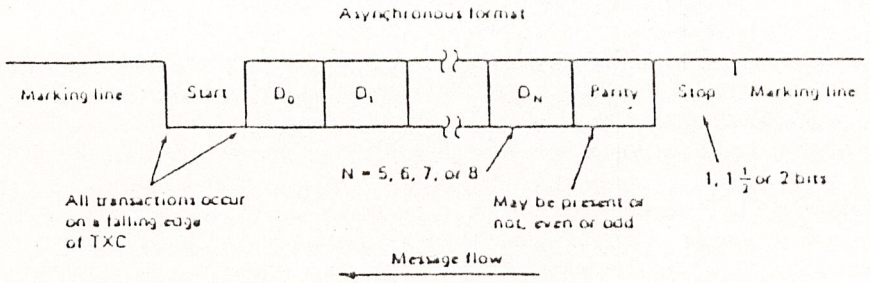
ก. การส่งข้อมูลแบบอะซิงโครนัส (Asynchronous Transmission)

ในการส่งแบบอะซิงโครนัส การส่งข้อมูลแต่ละตัวอักษรไม่มีกำหนดเวลาที่แน่นอน คือ แต่ละตัวอักษรอยู่ห่างกันเท่าไรก็ได้ หรือจะส่งติดกันไปตลอดก็ได้เช่นกัน ดังนั้นเพื่อให้ผู้รับแยกออกได้ว่าข้อมูลแต่ละตัวเริ่มต้นเมื่อใด ในการส่งข้อมูลแต่ละตัวหรือแต่ละไบท์นั้นจะมีสัญญาณสำหรับตรวจสอบบิทแรกภายในตัวของมันเอง โดยแต่ละไบท์จะถูกเพิ่มด้วยบิทเริ่มต้น (Start Bit) นำหน้าไบท์นั้น และบิทสิ้นสุด (Stop Bit) ตามหลังไบท์นั้น ซึ่งอาจจะมีการเพิ่มบิทพาริตี้ (Parity Bit) ก่อนบิทสิ้นสุดก็ได้ ดังนั้นระยะเวลาระหว่างข้อมูลแต่ละไบท์ไม่จำเป็นต้องแน่นอน เพราะอุปกรณ์รับจะตรวจสอบบิทแรกทีละไบท์เท่านั้น โดยขณะไม่มีการส่งข้อมูลสภาพลอจิกจะเป็น "1" อุปกรณ์รับจะคอยตรวจสอบการเปลี่ยนลอจิกจาก "1" เป็น "0" เมื่อกำหนดให้บิทเริ่มต้นมีลอจิกเป็น "0" ซึ่งหมายถึงบิทที่ตามมาหลังจากบิทเริ่มต้นคือบิทแรกของไบท์นั้นรูปแบบการจัดเรียงบิทในการส่งแบบอะซิงโครนัสแสดงดังรูป 2.4

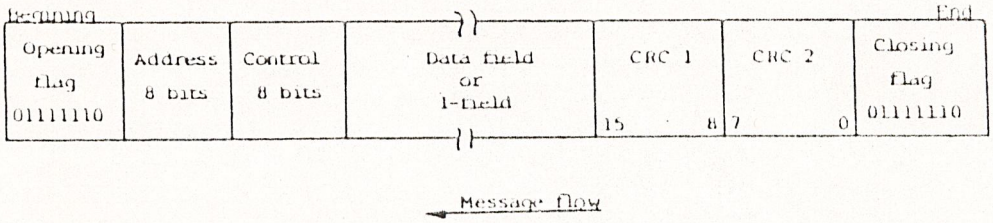
ข. การส่งข้อมูลแบบซิงโครนัส (Synchronous Transmission)

การส่งข้อมูลแบบซิงโครนัสหมายถึง การส่งข้อมูลแบบอนุกรมที่มีการกำหนดจำนวนอักขระที่จะส่งในแต่ละครั้งเป็นจำนวนที่แน่นอนเรียกว่า เฟรมข้อมูล การส่งข้อมูลแบบนี้จะต้องมีการส่งสัญญาณนาฬิกาไปพร้อม ๆ กับสัญญาณข้อมูล ในการส่งข้อมูลระยะสั้น ๆ สัญญาณนาฬิกาซึ่งใช้เป็นสัญญาณซิงค์อาจจะส่งแยกไปในสายส่งสัญญาณอีกเส้นหนึ่งไม่ส่งรวมไปในสายส่งข้อมูลก็ได้ แต่ถ้าเป็นการสื่อสารในระยะไกล ๆ แล้วสัญญาณนาฬิกาจะถูกเข้ารหัสส่งร่วมไปกับสัญญาณข้อมูลในสายส่งเส้นเดียวกัน การส่งแบบซิงโครนัส ข้อมูลจะเรียงติดกันไปโดยไม่มีบิทเริ่มต้นและบิทสุดท้ายของข้อมูลบล็อคหนึ่ง ๆ (บล็อคหนึ่ง ๆ ประกอบด้วยข้อมูลหลายชุด) จะแสดงถึงจุดเริ่มต้นและจุดสิ้นสุดของข้อมูลเท่านั้น เพราะ

ฉะนั้นถ้าเป็นการส่งอนุกรมแบบอะซิงโครนัสเราจะเพิ่ม Framing Bits รวมเข้าไปในแต่ละคาร์แรกเตอร์ และถ้าเป็นการส่งข้อมูลอนุกรมแบบซิงโครนัสเราจะเพิ่ม Framing Characters เข้าไปร่วมกับบล็อคของข้อมูลแต่ละบล็อค ซึ่งแสดงความแตกต่างได้ดังรูปที่ 2.4 และ 2.5



รูปที่ 2.4 การส่งข้อมูลอนุกรมแบบอะซิงโครนัส



รูปที่ 2.5 การส่งข้อมูลอนุกรมแบบซิงโครนัส

ข้อเปรียบเทียบระหว่างการส่งข้อมูลแบบขนานและแบบอนุกรม

	แบบขนาน	แบบอนุกรม
1.ระยะทาง	ปกติจะน้อยกว่า 100 ฟุต	ส่งได้ตั้งแต่ระยะทางสั้นๆจนถึงระยะทางเป็นไมล์
2.ความเร็ว	อัตราความเร็วสูงมากในระยะที่ไม่ไกลมากนักกำหนดได้เป็นจำนวนบิทต่อวินาที	อัตราความเร็วของข้อมูลที่ใช้กันอยู่ทั่วไปจะอยู่ในช่วง 0 ถึง 2 ล้านบิทต่อวินาที
3.ระดับของสัญญาณ	ในการอินเตอร์เฟสจะใช้ระดับของสัญญาณที่ใช้กับอุปกรณ์ TTL คือสัญญาณลอจิก 1และ 0 จะแทนด้วยระดับแรงดัน +5V และ 0V ตามลำดับ	ใช้มาตรฐานของ EIA-RS 232C คือมีระดับสัญญาณไฟฟ้าขนาด 12V หรืออาจจะใช้มาตรฐาน 20mA current loop หรืออาจจะใช้ระดับสัญญาณของ TTL ก็ได้ (ใช้กันน้อยมาก)
4.ความผิดพลาดของสัญญาณ	ถ้าส่งในระยะทางไกล ๆ ความผิดพลาดของข้อมูลจะเกิดขึ้นได้ง่าย	การผิดพลาดของสัญญาณจะมีน้อยลง
5.ค่าใช้จ่าย	ถ้าส่งในระทางไกลๆ จะสิ้นเปลืองค่าใช้จ่ายมากเพราะต้องใช้สายส่งสัญญาณหลายเส้น	สิ้นเปลืองน้อยกว่าหลายเท่า ถึงแม้ว่าจะใช้อุปกรณ์เปลี่ยนสัญญาณของข้อมูลจากแบบขนานไปเป็นแบบอนุกรมแล้วส่งผ่านสายส่งใช้อุปกรณ์ในการแปลงสัญญาณกลับมาเป็นแบบขนานอีกก็ยังลงทุนน้อยกว่า

2.3 ทิศทางของการส่งผ่านข้อมูล

ในการรับส่งข้อมูลระหว่างคอมพิวเตอร์นั้น อาจจะแบ่งตามลักษณะของการรับส่งได้เป็น 3 วิธีใหญ่ ๆ คือ

1) การรับหรือส่งทางเดียว (Simplex)

เป็นการส่งข้อมูลแบบไปในทางเดียว เช่นการส่งข้อมูลจากคอมพิวเตอร์ A ไปยังเทอร์มินอล B ในกรณีนี้คอมพิวเตอร์จะต้องเป็นเครื่องส่ง และ เทอร์มินอล B จะเป็นเครื่องรับเท่านั้น ตัวอย่างของการส่งข้อมูลแบบนี้คือ การส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ การส่งกระจายเสียงวิทยุหรือโทรทัศน์ การสื่อสารแบบ Simplex นี้เรามักจะไม่นำมาใช้ในการสื่อสารข้อมูล เนื่องจากว่าเราจำเป็นจะต้องตอบโต้กันระหว่างการรับส่งข้อมูล หรือบางทีก็เปลี่ยนจากผู้รับเป็นผู้ส่งซึ่งทำไม่ได้สำหรับการส่งแบบนี้

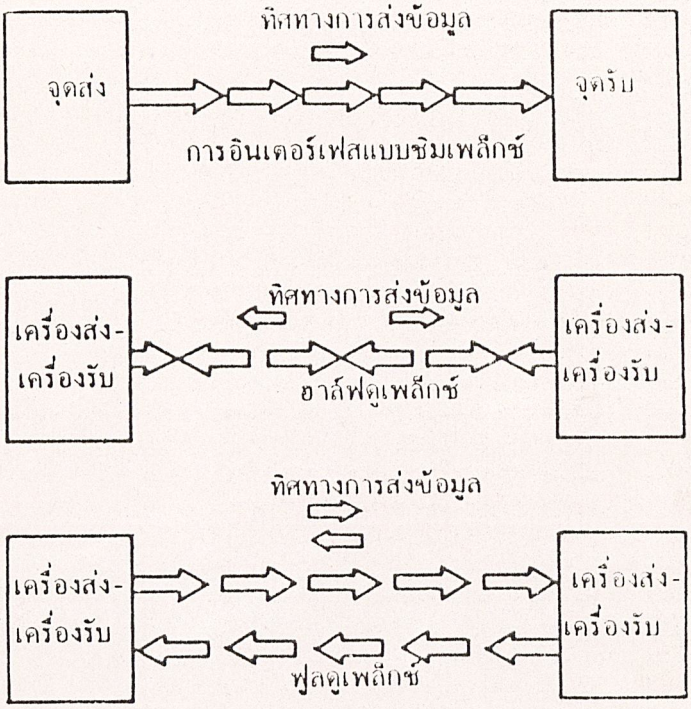
2) การรับส่งแบบผลัดกันส่ง (Half Duplex)

เป็นการรับส่งข้อมูลใน 2 ทิศทางสามารถทำหน้าที่เป็นเครื่องรับ และเครื่องส่งได้แต่ว่าต้องผลัดกันรับผลัดกันส่ง เมื่อฝ่ายใดฝ่ายหนึ่งส่งอีกฝ่ายหนึ่งก็จะทำหน้าที่รับ จนกระทั่งฝ่ายแรกส่งจบฝ่ายหลังจึงจะกลับมาเป็นผู้ส่งได้ และฝ่ายส่งในตอนแรกก็จะเป็นผู้รับสลับกันเช่นนี้เรื่อยไป ทั้งสองฝ่ายจะเป็นผู้ส่งพร้อมกันไม่ได้เพราะสัญญาณจะชนกัน อุปกรณ์ที่ใช้ในการติดต่อแบบ Half Duplex ได้แก่ ระบบวิทยุมือถือ (Walkie Talkie) ระบบ ATM และ Intercom เป็นต้น

3) การรับส่งสวนทางได้พร้อมกัน (Full Duplex)

เป็นการรับส่งข้อมูลใน 2 ทิศทางพร้อมกัน เช่น คอมพิวเตอร์ A ส่งข้อมูลไปให้คอมพิวเตอร์ B ซึ่งเป็นเวลาเดียวกันกับที่คอมพิวเตอร์ B ก็ส่งข้อมูลไปให้คอมพิวเตอร์ A คอมพิวเตอร์ A และ B จะมีการทำงานที่เป็นอิสระต่อกัน และสายส่งข้อมูลที่ใช้ในติดต่อระหว่างคอมพิวเตอร์ A และ B จะประกอบไปด้วยสายส่งทั้งหมด 3 เส้นคือ สายสัญญาณรับ/ส่ง และสายกราวนด์ โดยระบบของคอมพิวเตอร์ A และ B จะใช้สายกราวนด์ร่วมกันจะเรียกว่าเป็นระบบฟูลดูเพล็กซ์ 4 เส้น (Full duplex 4-wires)

ส่วนอีกระบบหนึ่งจะใช้สายสัญญาณส่งและรับร่วมกัน และในระบบจะมีสายส่งสัญญาณเพียง 2 เส้น ในลักษณะนี้เรียกว่าเป็นระบบฟูลดูเพล็กซ์ 2 เส้น (Full duplex 2 wires) แต่ในระบบฟูลดูเพล็กซ์ 2 เส้นนี้ จะต้องอาศัยเทคนิคการแบ่งความถี่เข้ามาช่วย คือ จะส่งในความถี่ช่วงหนึ่ง และจะรับความถี่อีกช่วงหนึ่ง



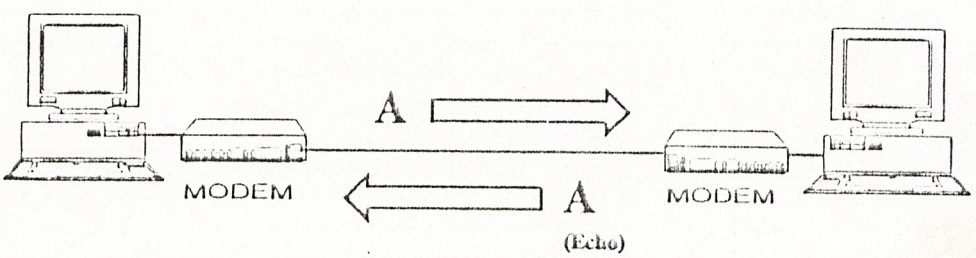
รูปที่ 2.6 การส่งข้อมูลในลักษณะต่าง ๆ

เมื่อคอมพิวเตอร์รับส่งข้อมูลในแบบ Half duplex หรือ Full duplex ก็ตามมันจะต้องใช้การรับส่งให้เหมือนกันทั้งสองด้าน การรับส่งข้อมูลแบบ Half duplex เครื่องคอมพิวเตอร์ผลัดกันส่งข้อมูลและมีหน้าที่พิมพ์ข้อความที่ตัวเองส่งออกไปขึ้น แสดงบนจอภาพด้วย หรือมีคุณสมบัติที่เราเรียกว่า "Echo On" นั่นเอง ข้อความไม่ว่าจะส่งออกไปโดยการพิมพ์จากแป้นพิมพ์หรืออ่านจากแผ่นดิสก์ตาม เครื่องคอมพิวเตอร์ที่ส่งข้อความจะต้องนำข้อความนั้นแสดงผลออกทางจอภาพด้วยตนเอง การรับส่งข้อมูลในแบบ Half

duplex จึงต้องกำหนดให้เครื่องคอมพิวเตอร์หรือเทอร์มินอลที่ใช้ทำงานในลักษณะ Echo On เสมอ มิฉะนั้นเราจะมองไม่เห็นข้อความที่เราส่งออกไป มองเห็นแต่เฉพาะข้อความที่อีกฝ่ายหนึ่งส่งมาเท่านั้น

เมื่อคอมพิวเตอร์รับส่งข้อมูลในแบบ Full duplex มันก็สามารถรับและส่งข้อมูลสวนทางทางกันได้ในเวลาเดียวกัน เครื่องคอมพิวเตอร์ที่รับส่งแบบ Full duplex จะไม่พิมพ์ข้อความที่ตัวเองส่งออกไปขึ้นแสดงบนจอภาพแต่จะรอรับข้อความจากอีกฝ่ายหนึ่งส่งกลับมาให้เท่านั้น เราเรียกว่า "Echo Off" ข้อความจากแป้นพิมพ์และจากแผ่นดิสก์ที่ส่งออกไป ปกติเราจะมองไม่เห็น เครื่องคอมพิวเตอร์อีกด้านหนึ่งจะส่งข้อความนั้นกลับมาให้ปรากฏบนจอภาพของเราเองเมื่อมีความจำเป็น การรับส่งข้อมูลแบบ Full duplex จึงต้องกำหนดให้คอมพิวเตอร์ทำงานในลักษณะ Echo Off เสมอ ถ้ากำหนดผิดเราจะเห็นข้อความที่พิมพ์ออกไปกลายเป็นสองตัวซ้อนกันอย่างนี้บนจอภาพ "LLiikkee TThhiiss"

ในการเชื่อมต่อระหว่างคอมพิวเตอร์นั้น การรับส่งแบบ Half duplex จะมีประสิทธิภาพต่ำกว่าเนื่องจากต้องผลัดกันส่งข้อมูล แต่ก็มีข้อดีคือ ประหยัดสายส่งข้อมูล เพราะเราสามารถใช้สายส่งเพียงคู่เดียวในการรับส่งข้อมูลแบบนี้ การรับส่งแบบ Full duplex จะต้องใช้สายสองคู่ คือ สำหรับส่งข้อมูลหนึ่งคู่ และรับข้อมูลอีกหนึ่งคู่ แยกวงจรรับส่งให้เป็นอิสระออกจากกัน ประสิทธิภาพในการรับส่งข้อมูลจึงสูงกว่าในแบบ Half duplex ถึงสองเท่า



รูปที่ 2.7 การรับส่งแบบ Full duplex ผู้รับจะต้องส่งข้อมูลกลับ (Echo) ไปให้ผู้ส่งเสมอ

ในระบบการส่งผ่านข้อมูล เรามักจะได้ยินคำว่า Unidirectional และ Bidirectional อยู่บ่อย ๆ Unidirectional Bus หมายถึงระบบการส่งผ่านข้อมูลที่มีจุดส่งข้อมูลอยู่เพียงจุดเดียวแต่จุดรับข้อมูลปลายทางจะมีอยู่เพียงจุดเดียวหรือมากกว่าก็ได้ ส่วน Bidirectional Bus เป็นวิธีส่งผ่านข้อมูลแบบฮาล์ฟดูเพล็กซ์ซึ่งจุดส่งข้อมูลจะมีอยู่หลายจุดและจุดรับข้อมูลก็มีมากกว่าหนึ่งจุดเช่นกัน ซึ่งทำให้การส่งผ่านข้อมูลเป็นไปอย่างมีประสิทธิภาพมากขึ้น

2.4 อัตราบิทและอัตราบ็อด (Bit Rate Versus Baud Rate)

อัตราบิทเป็นการส่งข้อมูลแบบอนุกรมจากเครื่องคอมพิวเตอร์เครื่องหนึ่งไปยังอุปกรณ์ต่อพ่วง หรือไปยังคอมพิวเตอร์อีกเครื่องหนึ่ง วัดเป็นจำนวนบิทต่อวินาที อาจมีค่าไม่เท่ากับอัตราการเปลี่ยนแปลงในสายส่ง (Baud Rate) ก็ได้

อัตราบ็อด เป็นอัตราการเปลี่ยนแปลงของสัญญาณอะนาล็อกในสายส่ง ซึ่งอาจจะเป็นอัตราการเปลี่ยนความถี่ อัตราการเปลี่ยนขนาดของสัญญาณหรืออัตราการเปลี่ยนช่วงต่อเนื่องของมุม (Phase) ในหนึ่งวินาทีก็ได้ และ อัตราบ็อดไม่จำเป็นต้องมีค่าเท่ากับอัตราการส่งข้อมูล ซึ่งวัดเป็น Bit Per Second โดยอัตราบ็อดอาจจะมีค่าน้อยกว่าหรือมากกว่าอัตราการส่งข้อมูลก็ได้ ขึ้นอยู่กับเทคนิคการผสมสัญญาณที่ใช้

ในสมัยก่อนการรับส่งข้อมูลใช้เทคนิคการผสมสัญญาณแบบง่าย ๆ เช่น การเปลี่ยนแปลงความถี่ตามข้อมูล "0" และ "1"ที่ได้รับ อัตราการส่งข้อมูลและอัตราการเปลี่ยนแปลงของสัญญาณในสายส่งมีค่าเท่ากัน เราจึงถือว่าอัตราการเปลี่ยนแปลงของสัญญาณในสายส่ง (Baud Rate) คือ อัตราการส่งข้อมูลนั่นเอง ต่อมาเทคนิคการผสมสัญญาณซับซ้อนมากขึ้น ทำให้เราสามารถส่งข้อมูลด้วยความเร็วสูง โดยอัตราการเปลี่ยนแปลงในสายยังคงเท่าเดิม ดังนั้นเมื่อเราพูดว่าโมเด็มรับส่งข้อมูลที่ความเร็ว 1200 บ็อด เราจะไม่ทราบเลยว่า โมเด็มนั้นรับส่งข้อมูลได้กี่บิทต่อวินาที เนื่องจากว่าถ้าโมเด็มผสมสัญญาณ 1 บิทต่อหนึ่งลูกคลื่น โมเด็มนั้นจะรับส่งข้อมูลที่ความเร็ว 1200 บิทต่อวินาที หรือ

ถ้าโมเด็มผสมสัญญาณ 2 บิทต่อหนึ่งลูกคลื่นที่เปลี่ยนแปลงในสายส่ง โมเด็มจะรับส่งข้อมูลได้เร็วถึง 2400 บิทต่อวินาที โดยยังคงมีอัตราการเปลี่ยนแปลงในสายส่งเท่ากับ 1200 บ็อดเหมือนเดิม เพราะฉะนั้น เราจึงเลิกใช้คำว่า Baud Rate สำหรับบอกความเร็วการรับส่งข้อมูลของโมเด็มมาใช้คำว่า Bite Rateหรืออัตราการส่งข้อมูลเป็นบิทต่อวินาทีแทน ซึ่งสื่อความหมายได้เข้าใจตรงกันมากกว่า

2.5 โมเด็ม (Modem)

ในการสื่อสารข้อมูล ลักษณะของสัญญาณไฟฟ้าที่ใช้จะเป็นดิจิตอลที่ต้องการตัวนำสัญญาณที่ตอบสนองความถี่ได้ในช่วงกว้าง จากข้อจำกัดของสายส่งทำให้สามารถส่งสัญญาณดิจิตอลได้ในระยะทางใกล้ ๆ เท่านั้น เพราะที่ระยะทางไกล ๆ สภาพความเป็นตัวเก็บประจุในสาย (Stray Capacitance) จะทำให้ค่าเวลาหน่วง (Delay Time) ของแต่ละองค์ประกอบความถี่ที่รวมตัวเป็นสัญญาณดิจิตอลมีค่าไม่เท่ากัน ทำให้เมื่อส่งสัญญาณดิจิตอลที่ความเร็วสูงอาจจะเกิดการผิดพลาดของข้อมูลที่ปลายทางได้

ในการสื่อสารข้อมูลในระยะทางไกล ๆ จึงอาศัยโครงข่ายของช่องสัญญาณเสียงที่กระจายครอบคลุมโดยทั่วไปอยู่แล้ว เช่น ระบบโทรศัพท์ ระบบวิทยุ มาใช้ซึ่งช่องสัญญาณเสียงที่ใช้อยู่ในช่วงความถี่ 300 - 3400 เฮิร์ทซ์ เพราะฉะนั้นการที่จะสื่อสารข้อมูลด้วยระบบดิจิตอลเข้าไปในโครงข่ายดังกล่าว จึงต้องทำการมอดดูเลทสัญญาณดิจิตอลให้กลายเป็นสัญญาณเสียงอะนาล็อก แล้วจึงทำการส่งออกไปในโครงข่ายดังกล่าว ในขณะเดียวกัน เวลารับก็ต้องทำการดีมอดดูเลทเสียงให้กลับมาเป็นสัญญาณดิจิตอลอีกครั้งในภาครับ ซึ่งอุปกรณ์ที่ทำหน้าที่ในการมอดดูเลทและดีมอดดูเลทสัญญาณดังกล่าวนี้เรียกว่า โมเด็ม (Modem) มาจากคำว่า **MO**dulator และ **DEM**odulator ดังนั้นโมเด็มคืออุปกรณ์ที่ทำหน้าที่ได้ 2 หน้าที่ คือ กระบวนการที่นำสัญญาณที่จะส่งแฝงเข้าไปในสัญญาณพาห์ (Modulation) และกระบวนการถอดสัญญาณที่ส่งออกมาจากสัญญาณพาห์ (Demodulation) บางครั้งอาจพบว่า มีผู้เรียกโมเด็มว่า Data Set

เราสามารถแบ่งชนิดของโมเด็มตามอัตราการส่งข้อมูลได้เป็น

1. อัตราการส่งข้อมูลต่ำ (Low-speed)

   มีอัตราการส่งข้อมูลไม่เกิน 600 bps

2. อัตราการส่งข้อมูลปานกลาง (Medium-speed)

   มีอัตราการส่งข้อมูลไม่เกิน 1200 bps

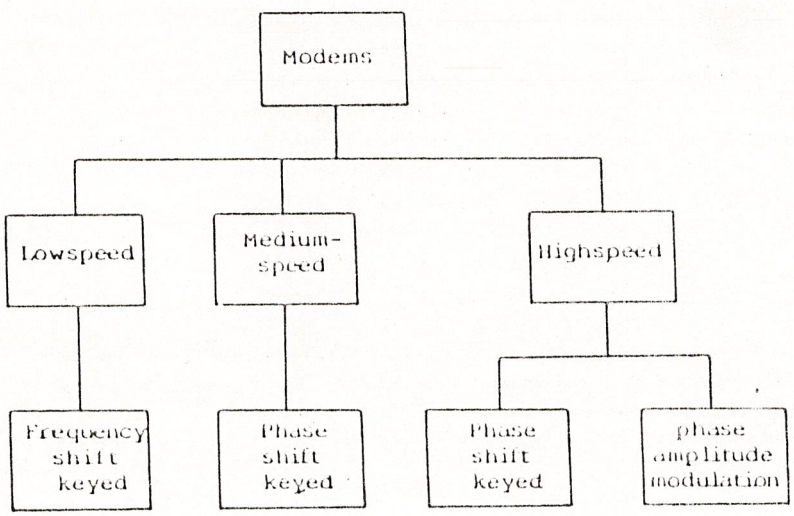
3. อัตราการส่งข้อมูลสูง (High-speed)

   มีอัตราการส่งข้อมูล 9600 bps

นอกจากนี้เรายังแบ่งชนิดของโมเด็มได้ตามเทคนิคการมอดดูเลทมีดังนี้

1. Frequency Shift Keying (FSK)
2. Phase Shift Keying (PSK)
3. Phase Amplitude Modulation (PAM)

สำหรับวิธีการมอดดูเลทยังมีเทคนิคอื่น ๆ อีกหลายแบบ แต่ค่อนข้างจะใช้น้อย ตามการแบ่งโดยลักษณะต่าง ๆ อาจจะสรุปเป็นบล็อกไดอะแกรมได้ดังรูป



รูปที่ 2.8 การแบ่งชนิดของโมเด็ม

สำหรับมาตรฐานของโมเด็มที่ใช้อยู่ทุกวันนี้เป็นไปตามที่ องค์การมาตรฐานสื่อสารสากลหรือ CCITT เป็นผู้กำหนดขึ้นโดยมีชื่อเรียกแต่ละมาตรฐานของโมเด็มขึ้นต้นด้วยอักษร V และตามด้วยตัวเลข จึงเรียกมาตรฐานเหล่านี้ว่า V-Series นอกจากมาตรฐานของโมเด็มแล้ว CCITT ยังเป็นผู้กำหนดมาตรฐานทางการสื่อสารอื่น ๆ อีก เช่น มาตรฐานของการสื่อสารผ่านดาวเทียม มาตรฐานของโทรสาร (Facsimile) มาตรฐานการสื่อสารข้อมูลต่าง ๆ ทั้งในแบบดิจิตอลและอะนาล็อกรวมถึงมาตรฐานที่เกี่ยวกับระบบโทรศัพท์อีกด้วย มาตรฐานที่ CCITT เป็นผู้กำหนดได้รับการยอมรับทั่วโลก การติดต่อสื่อสารระหว่างประเทศจึงดำเนินไปได้อย่างไม่มีปัญหา เนื่องจากทุกคนต่างก็ทำตามมาตรฐานเดียวกัน

มาตรฐาน V-Series นี้ไม่ใช่มาตรฐานของโมเด็มทั้งหมดบางมาตรฐานอาจหมายถึงการเชื่อมต่อแบบอื่น ๆเช่น V.24 เป็นมาตรฐานการรับส่งข้อมูลแบบอนุกรมเทียบได้กับ RS-232C และ V.35 หมายถึงการรับส่งข้อมูลแบบอนุกรมความเร็วสูงเป็นต้นในที่นี้จะกล่าวถึงมาตรฐานของโมเด็มแบบต่าง ๆ ที่ใช้กันมากตาม CCITT V-Series

- V.21เป็นมาตรฐานของโมเด็มความเร็ว 300 บิทต่อวินาที ใช้เทคนิคการผสมสัญญาณแบบ FSK รับส่งข้อมูลได้ในแบบ Full Duplex เป็นโมเด็มที่ใช้กับสายโทรศัพท์ ปัจจุบันนี้มีใช้กันน้อยมาก เนื่องจากความเร็วในการรับส่งข้อมูลต่ำ

- V.22 รับส่งข้อมูลด้วยความเร็ว 1200 บิทต่อวินาที หรือ ลดความเร็วลงมาที่ 600 บิทต่อวินาทีได้ ใช้เทคนิคการผสมสัญญาณแบบ PSK รับส่งข้อมูลในแบบ Full Duplex ใช้กับสายโทรศัพท์หรือสายตรงก็ได้ จัดเป็นโมเด็มความเร็วปานกลางที่ได้รับความนิยมอยู่ในปัจจุบัน

- V.22 bis รับส่งข้อมูลด้วยความเร็ว 2400 บิทต่อวินาทีหรือลดความเร็วลงมาที่ 1200 บิทต่อวินาที การผสมสัญญาณใช้เทคนิคของโมเด็มความเร็วสูงคือ QAM รับ

ส่งข้อมูลแบบ Full Duplex ใช้กับสายโทรศัพท์หรือสายตรงก็ได้จัดเป็นโมเด็มความเร็วปานกลางที่จะเข้ามาแทนที่ V.22 ซึ่งมาตรฐานนี้กำลังได้รับความนิยมมากเนื่องจากความเร็วสูงถึง 2400 บิทต่อวินาที และราคาของโมเด็มไม่แพงจนเกินไป

- V.23 คล้ายกับมาตรฐาน V.22 แต่รับส่งข้อมูลในแบบ Half Duplex คือมีความเร็ว 1200 บิทต่อวินาทีหรือลดความเร็วลงมาที่ 600 บิทต่อวินาทีใช้เทคนิคการผสมสัญญาณแบบ FSK ต่อใช้กับสายโทรศัพท์ได้ มาตรฐานนี้เราไม่ค่อยได้ใช้งานเท่าไรนักเพราะว่าประสิทธิภาพการรับส่งข้อมูลต่ำ และเป็นการติดต่อแบบ Half Duplex จึงสู้แบบ V.22 หรือ V.22 bis ไม่ได้

- V.26 เป็นมาตรฐานของโมเด็มสายตรง แบบใช้สาย 4 เส้น รับส่งข้อมูลในแบบ Full Duplex ใช้เทคนิคการผสมสัญญาณแบบ PSK มีความเร็วในการรับส่งข้อมูล 2400 บิทต่อวินาที จะนำมาใช้ต่อกับสายโทรศัพท์ไม่ได้ มาตรฐานนี้จึงไม่ค่อยได้พบเห็นมากนัก ปัจจุบันมีใช้น้อยเนื่องจากความเร็วต่ำเกินไปสำหรับสายตรงส่วนมากจะเลือกใช้มาตรฐานอื่นที่ความเร็วสูงกว่านี้

- V.26 bis เป็นมาตรฐานเหมือนกับ V.26 แต่ใช้กับสายโทรศัพท์แทน มีความเร็วในการรับส่งข้อมูลที่ 2400 บิทต่อวินาที หรือลดความเร็วลงมาที่ 1200 บิทต่อวินาทีได้ การรับส่งข้อมูลเป็นแบบ Half Duplex ใช้เทคนิคการผสมสัญญาณแบบ PSK มาตรฐานนี้จึงสู้ V.22 bis ไม่ได้ โดยทั่วไปจะใช้ V.22 bis ที่ความเร็ว 2400 บิทต่อวินาที มากกว่า

- V.27 เป็นมาตรฐานสำหรับโมเด็มความเร็ว 4800 บิทต่อวินาที ที่ใช้กับสายตรงเท่านั้น เทคนิคการผสมสัญญาณเป็นแบบ PSK รับส่งข้อมูลในแบบ Full Duplex ได้ ความเร็ว 4800 บิทต่อวินาทีนี้ถือว่าเป็นความเร็วการรับส่งข้อมูลสูงที่สุดของเทคนิค

การผสมสัญญาณแบบ PSK

- V.27 bis คล้ายกับมาตรฐานแบบ V.27 แต่ว่ารับส่งข้อมูลที่ 4800 บิทต่อวินาที หรือลดความเร็วลงมาที่ 2400 บิทต่อวินาทีได้ ใช้สำหรับสายตรงแบบ 4 เส้นเท่านั้น การผสมสัญญาณก็เป็นแบบ PSK สามารถรับส่งข้อมูลได้ทั้งในแบบ Full Duplex และ Half Duplex

- V.27 ter เป็นมาตรฐานโมเด็มความเร็ว 4800 บิทต่อวินาที หรือลดความเร็วลงมาที่ 2400 บิทต่อวินาทีได้ สำหรับใช้กับสายโทรศัพท์ การรับส่งข้อมูลเป็นแบบ Half Duplex เท่านั้นเทคนิคการผสมสัญญาณแบบ PSK มาตรฐานนี้คล้ายกับ V.27 bis เพียงแต่ใช้กับสายโทรศัพท์แทนที่จะเป็นสายตรง

- V.29 จัดเป็นมาตรฐานของโมเด็มความเร็วสูงใช้กับสายตรงแบบ 4 เส้นเท่านั้น การรับส่งข้อมูลใช้ได้ทั้ง Full Duplex และ Half Duplex สามารถรับส่งข้อมูลได้ตั้งแต่ 9600 บิทต่อวินาที ใช้เทคนิคการผสมสัญญาณแบบ QAM หรือลดความเร็วลงมาที่ 7200 บิทต่อวินาที และ 4800 บิทต่อวินาทีได้ ใช้การผสมสัญญาณแบบ PSK มาตรฐานนี้ใช้กันมากสำหรับการรับส่งข้อมูล ผ่านสายตรงระหว่างคอมพิวเตอร์กับคอมพิวเตอร์

- V.32 เป็นมาตรฐานโมเด็มความเร็วสูงสำหรับใช้กับสายโทรศัพท์ สามารถรับส่งข้อมูลได้ที่ความเร็ว 9600 บิทต่อวินาที ในแบบ Full Duplex หรือลดความเร็วลงมาที่ 4800 บิทต่อวินาทีได้ มาตรฐานนี้ยังใช้งานกับสายตรงแบบ 2-wires ได้อีกด้วย เทคนิคการผสมสัญญาณเป็นแบบ QAM ทั้งที่ความเร็ว 9600 และ 4800 บิทต่อวินาที การรับส่งข้อมูลความเร็วสูงผ่านสาย 2 เส้น ของ V.32 ใช้เทคนิค Echo Cancellation แทนที่จะใช้การแบ่งความถี่อย่างในโมเด็มความเร็วต่ำ

มาตรฐานของโมเด็ม V-Series ที่ได้กล่าวมานี้เป็นมาตรฐานที่พบเห็นได้ทั่ว

ไป ส่วนมาตรฐานของโมเด็มตามแบบสหรัฐอเมริกา หรือที่เรียกว่า Bell Standard ปัจจุบันค่อย ๆ ลดความนิยมลง เนื่องจากประเทศต่าง ๆ ใช้ตามมาตรฐานของ CCITT เป็นหลัก

ตารางแสดงมาตรฐานโมเด็ม V-Series ของ CCITT

มาตรฐานโมเด็มของ CCITT

Series Number	Line Speed	Channel Separation	FDX or HDX	Modulation Technique	Switch Lines	Leased Lines
V.21	300	FD	FDX	FSK	Yes	0
V.22	1200	FD	FDX	TSK	Yes	PP2W
V.22	600	FD	FDX	PSK	Yes	PP2W
V.22 bis	2400	FD	FDX	QAM	Yes	PP2W
V.22 bis	1200	FD	FDX	QAM	Yes	PP2W
V.23	600	NA	HDX	FMK	Yes	0
v.23	1200	NA	HDX	FMK	Yes	0
V.26	2400	4-Wire	FDX	PSK	No	PP MP4W
V.26 bis	2400	NA	HDX	PSK	Yes	No
V.26 bis	1200	NA	HDX	PSK	Yes	No
V.26 ter	2400	EC	Either	PSK	Yes	PP 2W
V.26 ter	1200	EC	Either	PSK	Yes	PP 2W
V.27	4800	ND	Either	PSK	No	Yes
V.27 bis	4800	4-Wire	Either	PSK	No	2W 4W
V.27 bis	2400	4-Wire	Either	PSK	No	2W 4W
V.27 ter	4800	None	HDX	PSK	Yes	No
V.27 ter	2400	None	HDX	PSK	Yes	No
V.29	9600	4-Wire	Either	QAM	No	PP 4W
V.29	7200	4-Wire	Either	PSK	No	PP 4W
V.29	4800	4-Wire	Either	PSK	No	PP 4W
V.32	9600	EC	FDX	QAM	Yes	PP 2W
V.32	9600	EC	FDX	TCM	Yes	PP 2W
V.32	4800	EC	FDX	QAM	Yes	PP 2W
V.32	14,400	4-Wire	FDX	TCM	FS	PP 4W

Also in preparation are:

V.34 a 14,400/19,200 half duplex asymmestrical duplex highspeed Modem for FAX applications. Unlikely to appear berore 1992.

V.32 bis a 14,400 full duplex PSTN modem; under study but not being formally discussed.

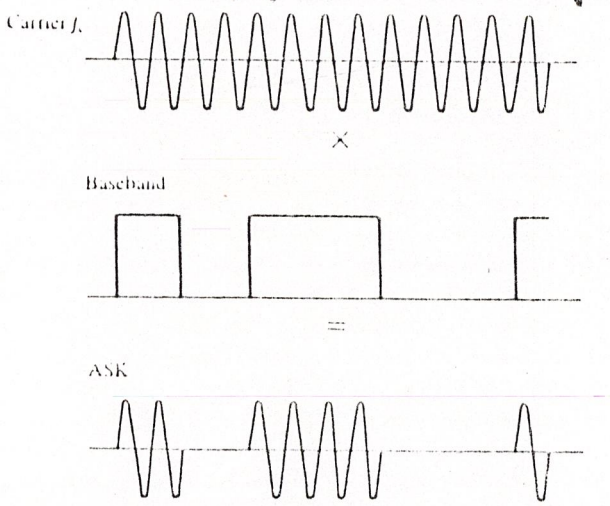
ND = not defined NA = not applicable EC = echo canceler
FD = frequency division FDX = full duplex HDX = half duplex
PP = point to point MP = multipoint FS = for further study

2.6 เทคนิคในการมอดดูเลทสัญญาณ (Digital Modulation Techniques)

การส่งผ่านข้อมูลจะต้องมีการมอดดูเลทสัญญาณข้อมูลแบ่งเป็นวิธีการต่างๆดังนี้

1) Amplitude Shift Keying (ASK)

เป็นการเปลี่ยนแปลงสัญญาณดิจิตอลให้อยู่ในรูปความถี่เดียว ที่มีความแตกต่างกันทางด้านระดับสัญญาณโดยให้ส่วนที่มีแอมปลิจูดสูงกว่าเป็นระดับลอจิก "1" และระดับแอมปลิจูดต่ำเป็นระดับลอจิก "0" โดยในด้านของวงจรด้านส่งจะต้องทำการแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอะนาล็อกก่อนโดยใช้ดีทูเอคอนเวอร์เตอร์ (D/A Converter) แล้วผ่านขบวนการมอดดูเลทส่งออกไป ส่วนทางด้านรับก็จะผ่านขบวนการดีมอดดูเลทเพื่อแยกเอาสัญญาณพาห์ออกแล้วจึงส่งผ่านวงจรเอทูดีคอนเวอร์เตอร์ (A/D Converter) เพื่อแปลงสัญญาณอะนาล็อกไปเป็นสัญญาณดิจิตอลแล้วส่งเข้าเทอร์มินอลเพื่อนำไปใช้งานต่อไป วิธีนี้จะมีความผิดพลาดในการส่งข้อมูลสูง ลักษณะของการมอดดูเลทเป็นรูปที่ 2.9

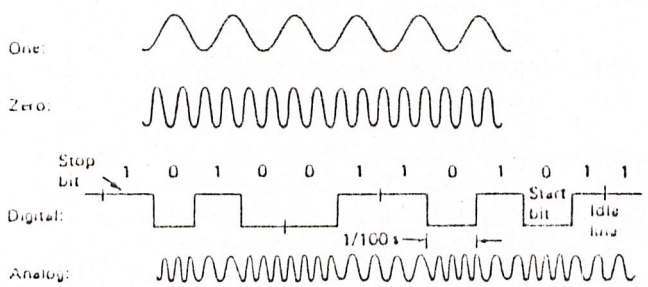


รูปที่ 2.9 การมอดดูเลทสัญญาณแบบขนาด

2) Frequency Shift Keying (FSK)

การมอดดูเลทสัญญาณด้วยวิธีนี้ เราทราบกันดีแล้วว่าใช้ในโมเด็มความเร็วต่ำโดยแทน "0" และ "1" ด้วยความถี่ต่างกัน ผู้ส่งจะใช้ความถี่สองความถี่ แทน "0" และ

"1" ของด้านส่ง ส่วนผู้รับก็ใช้ความถี่อีกสองความถี่แทน "0" และ "1" ของด้านรับเช่นกัน ทั้งหมดจึงใช้ความถี่รวม 4 ความถี่ การผสมสัญญาณแบบ FSK มักใช้กับโมเด็มความเร็วประมาณ 300 ถึง 600 บิตต่อวินาที ความเร็วสูงสุดของโมเด็มที่ใช้เทคนิคการผสมสัญญาณแบบนี้อยู่ที่ 1,200 บิตต่อวินาที แต่ตามปกติแล้วโมเด็มในท้องตลาดสำหรับเครื่องคอมพิวเตอร์ส่วนบุคคลที่ใช้การผสมสัญญาณแบบ FSK จะรับส่งข้อมูลด้วยความเร็ว 300 บิตต่อวินาที การผสมแบบ FSK นี้ อัตราการส่งข้อมูล (Bit Rate) จะเท่ากับ Baud Rate เสมอดังรูปที่ 2.10



รูปที่ 2.10 การมอดดูเลทสัญญาณแบบความถี่

โมเด็มแบบ FSK แบ่งออกได้ 2 แบบ คือ

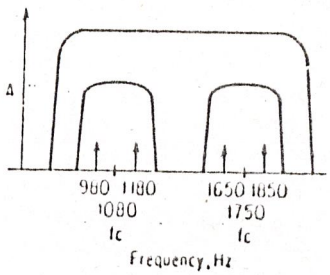
ก) ชนิดที่ใช้ในการส่งข้อมูลในระบบฟูลดูเพล็กซ์

โมเด็มชนิดนี้จะแบ่งแบนด์วิดธ์ของช่องสัญญาณเสียงออกเป็น 2 แบนด์เท่า ๆ กัน โดยแบนด์หนึ่งใช้ในการส่งข้อมูล ส่วนอีกแบนด์หนึ่งจะใช้ในการรับข้อมูล ทำให้สามารถส่งข้อมูลเข้าไปในสายได้พร้อม ๆ กัน ผลตอบสนองความถี่ (Frequency Response) ของโมเด็มแบบนี้เป็นดังรูปที่ 2.11

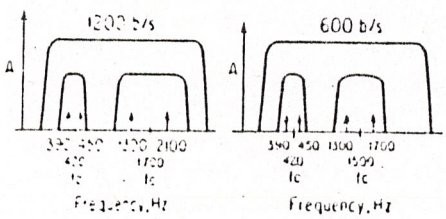
ข) ชนิดที่ใช้ในการส่งข้อมูลในระบบฮาล์ฟดูเพล็กซ์

โมเด็มในระบบนี้จะมีแบนด์ 2 แบนด์เหมือนกัน ส่งแบนด์วิดธ์ที่แคบจะใช้ในการส่งสัญญาณแนะนำควบคุม (Supervisory Signal) ของตัวรับไปยังตัวส่งเพื่อใช้ใน

การตรวจสอบสภาพการส่งข้อมูลว่าถูกต้องหรือไม่ เรียกแบนด์วิดธ์นี้ว่า ช่องสัญญาณย้อนกลับ (Reverse Channel) ผลตอบสนองความถี่ของโมเด็มแบบนี้เป็นดังรูปที่ 2.12



รูปที่ 2.11 ช่องสัญญาณในสายส่ง เมื่อใช้โมเด็ม FSK ฟูลดูเพล็กซ์

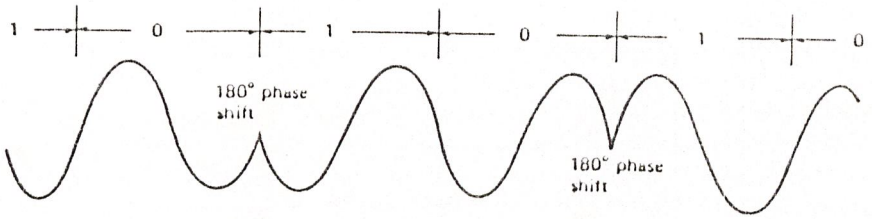


รูปที่ 2.12 ช่องสัญญาณในสายส่ง เมื่อใช้โมเด็ม FSK ฮาล์ฟดูเพล็กซ์

3) Phase Shift Keying (PSK)

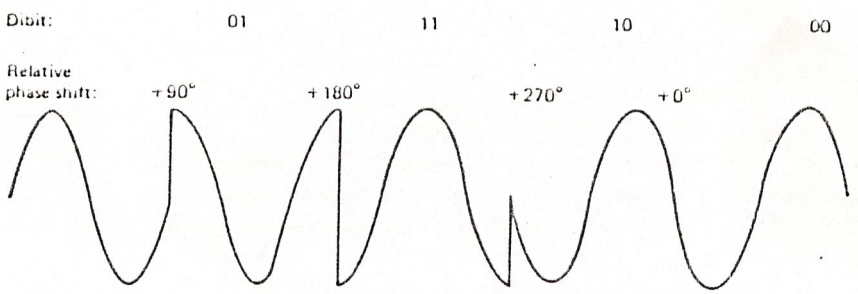
สำหรับการผสมสัญญาณแบบ Phase Shift Keying (PSK) นั้น ใช้หลัก

การที่ว่าแทนข้อมูล "0" และ "1" ด้วยการเปลี่ยนแปลงมุมของช่วงสัญญาณในสายส่ง (Phase) เช่นเราอาจกำหนดว่า "0" แทนด้วยมุมต่อเนื่องกันไป และ "1" แทนด้วยมุมเปลี่ยนไปจากเดิม 180 องศา ถ้าเราส่งข้อมูล 0101 รูปร่างคลื่นในสายส่งจะเป็นดังในรูปที่ 2.13



รูปที่ 2.13 การใช้ PSK ผสมสัญญาณแบบหนึ่งบิท

จากหลักการที่เราเปลี่ยน Phase ในสัญญาณส่งนี้ ด้วยมุมทั้ง 360 องศา ของคลื่นพาห์ทำให้เราสามารถแบ่งการเปลี่ยนแปลงของ Phase ให้เล็กลงไปได้อีก เช่น เปลี่ยน Phase ทีละ 90 องศา ซึ่งในหนึ่งลูกคลื่นจะเปลี่ยนได้ถึง 4 สภาวะและกำหนดให้ แต่ละสภาวะแทนด้วยข้อมูล 2 บิตได้ จากตารางเราสามารถกำหนด 00,01,10 และ 11 ให้มีการเปลี่ยน Phase ได้ครั้งละ 90 องศา เมื่อเราส่งข้อมูลเช่น 01 11 10 00 รูปร่างของคลื่นในสายส่งจะเป็นดังรูปที่ 2.14



รูปที่ 2.14 การใช้ PSK ผสมสัญญาณแบบสองบิท

ถ้าคลื่นพาห์มีความถี่ 600 Hz ข้อมูลที่ส่งออกไปจะมีค่าเท่ากับ 1,200 บิตต่อวินาที เนื่องจากการเปลี่ยนแปลง Phase หนึ่งครั้งเท่ากับข้อมูลสองบิต

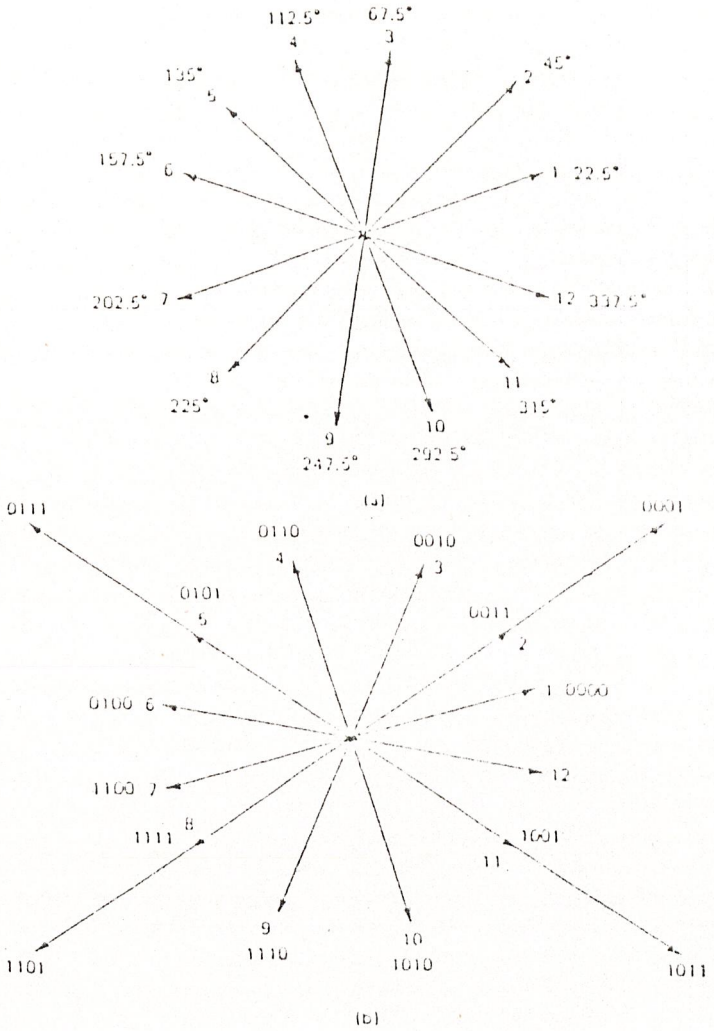
เราจะเห็นว่าการผสมสัญญาณแบบ PSK นี้ อัตราการส่งข้อมูล (Bit Rate) จะมีค่าสูงกว่าอัตราการเปลี่ยนแปลงของสัญญาณในสายส่ง (Baud Rate)ได้ การผสมสัญญาณที่เราใช้กันมาก คือ ใช้ข้อมูล 2 บิต 3 บิต และ 4 บิต ในการเปลี่ยน Phase โดยแบ่งมุมของคลื่นพาห์ ให้มีการเปลี่ยนแปลง เท่ากับ 90 องศา 45 องศา และ 22.5 องศา การเปลี่ยนแปลงมุม 22.5 องศานี้ นับว่าน้อยมาก ทำให้ข้อมูลมักจะผิดพลาดอยู่เสมอ ส่วนมากจึงใช้งาน PSK ที่ความเร็ว 4,800 บิต ต่อวินาที และที่ความเร็วสูงกว่านี้เราจะใช้เทคนิคการผสมสัญญาณที่ซับซ้อนขึ้นไปอีก

4) Phase Amplitude Modulation (PAM)

เป็นเทคนิคที่เกิดจากการรวมเอาสัญญาณมอดดูเลทที่เกิดจากเทคนิค PSK และ AM เข้าด้วยกัน ทำให้ได้อัตราการส่งสูงขึ้นคือสามารถส่งบิทข้อมูลได้ถึง 16 แบบต่าง ๆ กันดังรูปที่ 2.15

ในรูปที่ 2.15 จะเห็นว่ามีการแบ่งมุมเฟสออกเป็น 12 มุม สำหรับใช้ในการมอดดูเลทแบบ PAM และถ้าเราเพิ่มขนาดของแอมปลิจูดให้แก่มุมเฟส 4 มุมจากทั้งหมดทำให้มุม เฟส 4 มุมที่เราเพิ่มแอมปลิจูดเข้าไปนั้นมีแอมปลิจูดถึง 2 ค่า ดังนั้นเกิดมุมเฟสขึ้นอีก 4 ค่ารวมเป็น 16 ค่า ดังรูป ซึ่งมุมเฟสแต่ละค่าก็ถูกใช้ในการแทนข้อมูลกลุ่มหนึ่งเพราะ ฉะนั้นจึงสามารถแทนกลุ่มข้อมูลได้ทั้งหมด 16 แบบต่าง ๆ กัน ดังนั้นเทคนิควิธีนี้จึงทำให้อัตราการส่งข้อมูลสูงขึ้นถึง 9600 บิทต่อวินาที ส่วนใหญ่โมเด็มประเภทนี้จะใช้การรับส่งข้อ มูลแบบซิงโครนัส

นอกจากการมอดดูเลทดังกล่าวมาข้างต้นแล้ว ยังมีการมอดดูเลทอีกวิธีหนึ่งเรียกว่า Quadrature Amplitude Modulation (QAM) เป็นการผสมสัญญาณที่ใช้ทั้งการเปลี่ยน Phase และขนาดของสัญญาณควบคู่กันไป สำหรับใช้กับโมเด็มความเร็วสูง ซึ่งถ้าใช้การเปลี่ยน Phase เพียงอย่างเดียวมุมที่เปลี่ยนแปลงจะมีค่าน้อยเกินไปทำให้เกิดข้อผิดพลาดได้ง่ายถ้าเราใช้การเปลี่ยน Phase และขนาดของสัญญาณ ค่าของมุมก็จะอยู่



รูปที่ 2.15 PAM a) มุมเฟสต่าง ๆ กัน 12 มุม b) การเข้ารหัสขนาด 4 บิท

ห่างกันมากขึ้น ปกติที่มีใช้กันอยู่จะมี Phase ต่างกัน 8 Phase และขนาดของสัญญาณต่างกัน 4 ระดับ ใช้แทนข้อมูล 16 สถานะ ซึ่งในหนึ่งลูกคลื่นจะสามารถส่งข้อมูลได้คราวละ 4 บิต การผสมสัญญาณของ QAM บางแบบจะใช้ Phase ต่างไปจากนี้ เช่น ใช้ Phase ต่างกัน 12 Phase และขนาดของสัญญาณ 3 ระดับ หรืออาจใช้ Phase ต่างกัน 8 Phase และขนาดของสัญญาณต่างกัน 2 ระดับก็ได้ ขึ้นอยู่กับการออกแบบ แต่ว่าในมาตร

ฐานเดียวกัน โมเด็มจะต้องใช้การแบ่ง Phase และระดับสัญญาณเท่ากันเสมอ ความเร็วในการรับส่งข้อมูลของ QAM อยู่ที่ 9,600 บิตต่อวินาที โดยใช้ความถี่พาหะ 2,400 Hz และในหนึ่งลูกคลื่นแทนข้อมูลได้คราวละ 4 บิต เทคนิคการผสมสัญญาณแบบ QAM นี้อาจถูกนำไปใช้กับโมเด็มความเร็วปานกลางก็ได้ เช่น ที่ความเร็ว 2,400 บิตต่อวินาที เป็นต้น

นอกจากนี้ยังมีการผสมสัญญาณสำหรับโมเด็มความเร็วสูงอีกหลายแบบ แต่ส่วนมากยังไม่ได้รับการยอมรับให้เป็นมาตรฐานในปัจจุบัน การผสมสัญญาณแบบ QAM มีความซับซ้อนมาก ดังนั้นวงจรทางด้านฮาร์ดแวร์ไม่ว่าจะเป็นวงจรด้านส่ง วงจรด้านรับและการการแยกสัญญาณต่าง ๆ ยุ่งยากกว่าที่ใช้กับวงจรเปลี่ยน Phase อย่างเดียว ราคาของโมเด็มที่ใช้เทคนิคแบบ QAM จึงสูงกว่าโมเด็มที่ใช้เทคนิค PSK อยู่มาก

ในการแบ่งชนิดของโมเด็ม นอกจากที่ได้กล่าวมาแล้วยังมีการแบ่งโมเด็มออกเป็นแบบ 2-wires และ 4-wires การที่จะเลือกว่าควรจะใช้แบบใด ขึ้นกับว่าในการปฏิบัติงานต้องการจะใช้กับการส่งแบบฟูลดูเพล็กซ์ หรือ ฮาล์ฟดูเพล็กซ์ ในโมเด็มความเร็วต่ำแถบความถี่สัญญาณอยู่ในช่วงแคบ ๆ ไม่เกิน 3000 Hz สามารถใช้ได้กับเครือข่ายโทรศัพท์ แต่ในโมเด็มความเร็วปานกลางและความเร็วสูง ต้องการแถบความถี่ 3000 Hz เต็มที่เพื่อที่จะทำการส่งผ่านข้อมูลได้เร็วขึ้น ดังนั้นในการใช้งานถ้าต้องการความรวดเร็วและเป็นการส่งผ่านข้อมูลแบบฟูลดูเพล็กซ์แล้ว มักจะใช้โมเด็มความเร็วสูงแบบ 4-wires

การทำงานของโมเด็มชนิด 2-wires โครงสร้างของการส่งผ่านข้อมูลออกเป็น 2 แชนแนล ในแชนแนลที่ 1 จะถูกใช้ในการส่งผ่านข้อมูลในทิศทางใดทิศทางหนึ่งในขณะหนึ่ง ส่วนในแชนแนลที่ 2 จะถูกใช้สำหรับส่งสัญญาณควบคุมหรือสัญญาณแฮนด์เชค (Handshaking) ซึ่งจะส่งสัญญาณมาให้ทางด้านส่งทราบว่าทางด้านรับได้รับข้อมูลที่ส่งมาเรียบร้อยแล้ว ส่วนใหญ่โมเด็มชนิด 2-wires ใช้ในการรับส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ แต่ถ้าต้องการใช้งานแบบฟูลดูเพล็กซ์แล้วควรใช้โมเด็มชนิด 4-wires จึงจะเกิดประสิทธิภาพสูงสุด

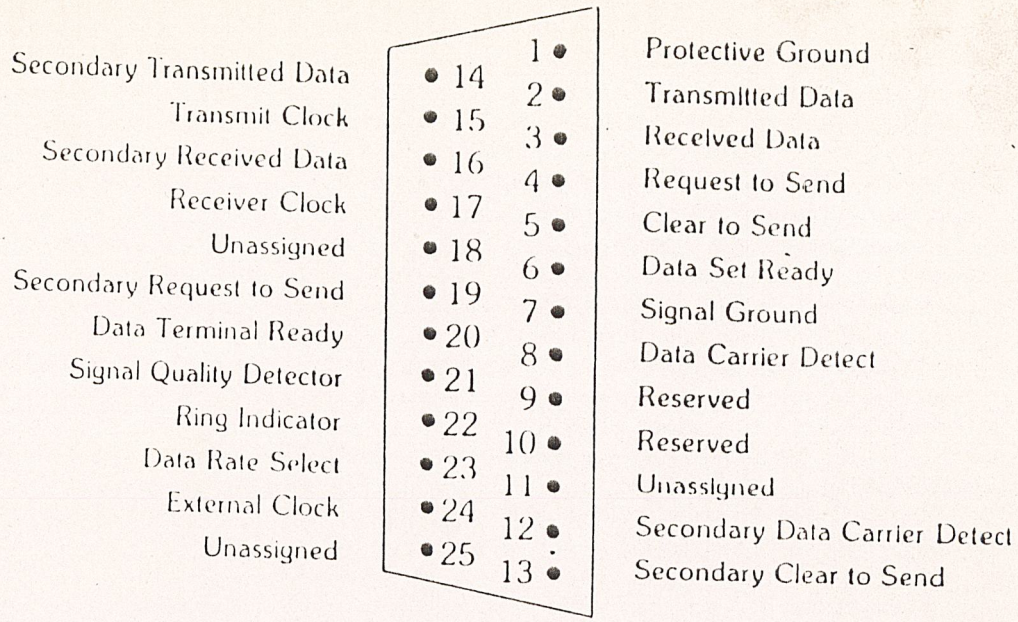
2.7 อินเตอร์เฟส EIA RS-232C

โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ทที่เป็นแบบอนุกรมเรียกกันว่า RS-232C อยู่ในตัวเองอยู่แล้วหลายเครื่องไม่มีมากับเครื่อง เช่น IBM PC จึงจำเป็นจะต้องมีการ์ดที่เรียกว่า Asynchronous Communication Adapter มาเสียบใส่

พอร์ท RS-232C นี้ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรมเ เรียกว่า Universal Asynchronous Aadapter เหตุที่มีชื่อเรียกว่า RS-232C ก็เนื่องจากสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิคส์ของอเมริกา หรือ EIA (Electronics Industries Association) ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมเอาไว้ภายใต้ชื่อว่า RS-232C ความจริงมาตรฐานของการส่งข้อมูลแบบอนุกรมมีหลายมาตรฐาน แต่ที่นิยมกันมากที่สุดสำหรับไมโครคอมพิวเตอร์ก็คือ RS-232C

มาตรฐาน RS-232C ได้จัดพิมพ์ขึ้นเมื่อปี ค.ศ. 1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิคส์แห่งสหรัฐอเมริกา RS ย่อมาจาก Recommended Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็หมายถึงตัวไมโครคอมพิวเตอร์ และ DCE ก็หมายถึงโมเด็มหรืออุปกรณ์อื่น ๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรม อาจจะเป็นได้ทั้ง DTE และ DCE ขึ้นอยู่กับผู้ผลิต ความจริงอีกประการหนึ่งของ RS-232C ก็คือ ความเร็วและระยะทางของการเชื่อมต่อ RS-232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0 ถึง 20,000 บิทต่อวินาที ซึ่งเพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบ็อด 110 ถึง 9,600 บ็อด ความยาวของสายเชื่อมต่อสัญญาณตามมาตรฐานของ RS-232C จำกัดอยู่แค่ 50 ฟุตซึ่งเพียงพอสำหรับการสื่อสารระหว่างไมโครคอมพิวเตอร์กับอุปกรณ์รอบนอก

อินเตอร์เฟส RS-232C จะใช้ต่อแบบ DB-25 (D-type 25 pin connector) เป็นมาตรฐานโดยข้อต่อบน DTE จะเป็นตัวผู้ ส่วนข้อต่อที่ DCE จะเป็นตัว



PIN NUMBER	COMMON NAME	RS-232-C NAME	DESCRIPTION	SIGNAL DIRECTION ON DCE
1		AA	PROTECTIVE GROUND	–
2	TXD	BA	TRANSMITTED DATA	IN
3	RXD	BB	RECEIVED DATA	OUT
4	$\overline{RTS}$	CA	REQUEST TO SEND	IN
5	$\overline{CTS}$	CB	CLEAR TO SEND	OUT
6	DSR	CC	DATA SET READY	OUT
7	GND	AB	SIGNAL GROUND (COMMON RETURN)	–
8	CD	CF	RECEIVED LINE SIGNAL DETECTOR	OUT
9		–	(RESERVED FOR DATA SET TESTING)	
10		–	(RESERVED FOR DATA SET TESTING)	–
11			UNASSIGNED	–
12		SCF	SECONDARY REC'D LINE SIG DETECTOR	OUT
13		SCB	SECONDARY CLEAR TO SEND	OUT
14		SBA	SECONDARY TRANSMITTED DATA	IN
15		DB	TRANSMISSION SIGNAL ELEMENT TIMING (DCE SOURCE)	OUT
16		SBB	SECONDARY RECEIVED DATA	OUT
17		DD	RECEIVER SIGNAL ELEMENT TIMING (DCE SOURCE)	OUT
18			UNASSIGNED	–
19		SCA	SECONDARY REQUEST TO SEND	IN
20	$\overline{DTR}$	CD	DATA TERMINAL READY	IN
21		CG	SIGNAL QUALITY DETECTOR	OUT
22		CE	RING INDICATOR	OUT
23		CH/CI	DATA SIGNAL RATE SELECTOR (DTE/DCE SOURCE)	IN/OUT
24		DA	TRANSMIT SIGNAL ELEMENT TIMING (DTE SOURCE)	IN
25			UNASSIGNED	–

รูปที่ 2.16 แสดงข้อต่อของ RS-232C

เมียโดยลักษณะของข้อต่อพร้อมทั้งชื่อขา แสดงได้ดังรูปที่ 2.16

ระดับสัญญาณที่ใช้ใน RS-232C จะเป็นสัญญาณโวลเตจสองระดับ (Bipolar Voltage Signal) ระดับสัญญาณต่าง ๆ เมื่อเทียบกับกราวด์จะต้องไม่เกิน ± 25 โวลท์ และเมื่อเกิดการลัดวงจรระหว่างขาต่าง ๆ จะต้องไม่เกิดการเสียหายขึ้นกับอุปกรณ์ใด ๆ ที่ต่ออยู่กับ RS-232C รวมทั้งตัวมันเองด้วย สัญญาณลอจิกเอาท์พุท "0" เมื่อผ่าน RS-232C จะมีระดับโวลเตจอยู่ในช่วง +5 ถึง +15 โวลท์ และลอจิก "1" จะมีระดับโวลเตจอยู่ในช่วง -5 ถึง -15 โวลท์ ส่วนที่ตัวรับสัญญาณอินพุทลอจิก "0" จะมีค่าในช่วง +3 ถึง +15 โวลท์ และสัญญาณอินพุทลอจิก "1" จะมีค่าในช่วง -3 ถึง -15 โวลท์ สำหรับระดับโวลเตจในช่วง ± 3 โวลท์ จะไม่มีความหมายใด ๆ

สัญญาณต่าง ๆ ที่ใช้ในบน RS-232C ที่สำคัญ แสดงได้ดังนี้

Transmit Data (TD ขาที่ 2)

เป็นสัญญาณที่ส่งออกจาก DTE (หรือตัวไมโครคอมพิวเตอร์) ไปยังโมเด็ม หรือต่อเข้าโดยตรงกับคอมพิวเตอร์ตัวอื่นหรือตัวเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออก สถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ "1" หรือเท่ากับบิทสิ้นสุด

Receive Data (RD ขาที่ 3)

เป็นทางของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณรับเข้ามา ขานี้จะมีสถานภาพทางลอจิกเป็น "1"

Request To Send (RTS ขาที่ 4)

ใช้สำหรับส่งสัญญาณไปยังโมเด็ม หรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้จะใช้คู่กับ CTS อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณหรือยัง หากพร้อมที่จะรับก็ส่งสัญญาณออกไปที่สาย CTS

Clear TO Send (CTS ขาที่ 5)

ดังอธิบายไว้ใน RTS เมื่อสัญญาณนี้อยู่ในสถานภาพออฟ (negative voltage หรือ ลอจิก "1") หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

Data Set Ready (DSR ขาที่ 6)

เมื่อสัญญาณสายนี้อยู่ในสถานะออน (หรือลอจิก "0") เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้ว และพร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนเวียนเลขอัตโนมัติ จะส่งสัญญาณสายนี้ไปบอกให้คอมพิวเตอร์รู้ว่า ต่อโทรศัพท์ได้สำเร็จแล้ว

Signal Ground (SG ขาที่ 7)

SG ทำหน้าที่เป็นระดับแรงดันหรือแรงดันอ้างอิงสำหรับทุก ๆ สายของสัญญาณจะมีแรงดันเป็น "0" เมื่อเทียบกับสัญญาณตัวอื่น

Carrier Detect (CD ขาที่ 8)

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

Data Terminal Ready (DTR ขาที่ 20)

คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน (ลอจิก "0") เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD, DSR, CTS) ให้คอมพิวเตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

Ring Indicator (RI ขาที่ 22)

สัญญาณที่ใช้ในโมเด็มที่เป็นระบบตอบโต้อัตโนมัติ (Auto-answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และออฟระหว่างเสียงดังของกระดิ่ง

นอกจากมาตรฐานการอินเตอร์เฟสของ EIA แล้วยังมีองค์กรต่าง ๆได้สร้างมาตรฐานของตนเองขึ้นมาซึ่งมีอยู่มากมาย จึงจำเป็นต้องทราบถึงมาตรฐานต่าง ๆ ที่มีใช้กันอยู่ให้ครบถ้วน

CCITT : The Consultative Committee in International Telegraphy and Telephony

เป็นองค์กรสากลที่กำหนดมาตรฐานเกี่ยวกับการควบคุมระบบภาษี แนะนำมาตรฐานหรือกำหนดมาตรฐานของระบบสื่อสารระหว่างประเทศ ทั้งโทรเลขและโทรศัพท์ CCITT เป็นหน่วยงานหนึ่งของ Internatonal Telecommunication Union มีหน้าที่กำหนดมาตรฐานทางด้านการสื่อสารข้อมูล

สำหรับมาตรฐานของ CCITT ที่ใช้กันแพร่หลายทางด้านการสื่อสารข้อมูล เช่น V.28 ใช้แทน RS-232C ได้ V.10 ใช้แทน EIA RS-423A ได้ V.11 EIA RS -422A ได้ และ X.21 ใช้แทน EIA RS-423A ได้ เป็นต้น

ในการกำหนดมาตรฐานของ CCITT จะกำหนดออกมาเป็นฉบับต่าง ๆ คือ Series A,B,C,D,...,Z ทุก ๆปีจะมีการร่วมประชุมปรึกษาหารือระหว่างประเทศสมาชิกในข้อกำหนดใหม่ ๆ หรือปรับปรุงมาตรฐานเก่าให้ทันสมัยและทุก 2-4 ปีจะออกหนังสือ Recommendation Series ต่าง ๆให้ทันสมัยขึ้น การอ้างถึง Recommendation ของ CCITT ต้องระบุว่าเป็นปีใด สีอะไร เช่น CCITT Recommendation Yellow book Series... Year... หรือ Orange book Series... Year... เป็นต้น ในที่นี้จะกล่าวถึงเฉพาะ Orange book

CCITT Series	Orange book
A	กล่าวถึงโครงสร้างขององค์กร CCITT และการปฏิบัติงาน
B	เกี่ยวกับความหมายของคำที่ใช้หรือชื่อ
C	เกี่ยวกับการสื่อสารทั่วไป
D	คำแนะนำเกี่ยวกับภาษีทั่วไป เช่นการเช่าวงจร Private line หรือการให้การบริการ
E	คุณภาพของการบริการโทรศัพท์และภาษีรวมทั้งค่าบริการระหว่างประเทศ
F	การบริการโทรเลขและภาษีรวมทั้งค่าบริการระหว่างประเทศ
G,H,J	เกี่ยวกับสายสื่อสาร
M,N	เกี่ยวกับการวัด ตรวจสอบและการบำรุงรักษาสายสื่อสาร
O	คุณลักษณะเฉพาะของเครื่องทดสอบและเครื่องมือวัด
P	คุณภาพของสายโทรศัพท์และเครื่องโทรศัพท์
Q	เรื่องทั่วไปเกี่ยวกับชุมสายโทรศัพท์และระบบสัญญาณที่ใช้
R1/R2	เกี่ยวกับระบบสัญญาณต่าง ๆ
R,S,T,U	เกี่ยวกับเทคนิคทางด้านโทรเลข
V	การสื่อสารข้อมูลผ่านสายโทรศัพท์
K/L	การป้องกันต่าง ๆ
X	โครงข่ายของการสื่อสารข้อมูลสาธารณะ
Z	การใช้ภาษาสำหรับโปรแกรมที่ใช้สำหรับการสื่อความหมาย

ISO : The International Standard Organization

เป็นองค์กรที่ตั้งอยู่ในเมืองเจนีวา สวิตเซอร์แลนด์ ทำหน้าที่กำหนดมาตรฐานทางกายภาพของอุปกรณ์ต่าง ๆ ที่ใช้ในการสื่อสารโทรคมนาคม โดยเฉพาะที่เกี่ยวกับคอมพิวเตอร์ และ อินฟอร์เมชั่นโพรเซสชิ่ง องค์กรนี้จะประสานงานกับ CCITT อย่างใกล้ชิด

ANSI : The American National Standard Institute

มาตรฐาน ANSI ส่วนใหญ่จะเกี่ยวกับ

1.กำหนดลักษณะของระบบผลิตสัญญาณและรับสัญญาณที่ใช้ในระบบสื่อสารทั่วไป

2.กำหนดคุณภาพและคุณลักษณะของข้อมูลขณะที่กำลังส่งออกไป

3.ให้บริการเกี่ยวกับมาตรฐานสากลและมาตรฐานภายในประเทศสหรัฐอเมริกา

สำหรับหน่วยงานของ ISO จะแบ่งออกเป็น 2 ชุด คือ

Technical committee 97 จะศึกษาเกี่ยวกับคอมพิวเตอร์และ อินฟอร์เมชั่นโพรเซสซิ่ง โดยที่คณะกรรมการชุดแรกคือ ISO/TC97/SC6จะรับผิดชอบเกี่ยวกับการพัฒนามาตรฐานทางด้านการสื่อสารข้อมูล และคณะกรรมการชุดที่สอง คือ ISO/TC97/SC 16 จะพัฒนาทางด้าน Open System Interconnection (OSI)

มาาตรฐานของ OSI สามารถใช้แทนมาตรฐานประเภทเดียวกันของ CCITT และ EIA ได้เช่น ISO 2110 สามารถใช้แทน EIA RS-232C และ RS-366A ได้ นอกจากนี้ ISO 4902 ยังใช้แทน EIA RS-499 ได้เป็นต้น

สำหรับคณะกรรมการของ ANSI ที่ค้นคว้าทางด้านการสื่อสารข้อมูลคือ Committee on Computers and Information Processing X3 มาตรฐานของ ANSI ที่รู้จักกันแพร่หลายคือ รหัสแอสกี (ASCII : American Standard Code for Information Interchange) ซึ่งเป็นรหัสมาตรฐานที่ใช้กันทั่วไปในระบบไมโครคอมพิวเตอร์

Federal Goverment Standard

เป็นหน่วยงานของรัฐบาลสหรัฐอเมริกา ทำหน้าที่กำหนดมาตรฐานทางด้านการสื่อสารในส่วนที่เกี่ยวกับการสื่อสารข้อมูล เมื่อถูกกำหนดใช้เป็นมาตรฐานโดย National Bureau of Standards (NBS) แล้วเราจะรู้จักกันในชื่อของ Federal Information Processing Standards (FIPS) ซึ่งมาตรฐานส่วนใหญ่จะเหมือนกับมาตรฐานของ EIA

Millitary Standard-188

เป็นมาตรฐานของทหารซึ่งถูกกำหนดเป็นมาตรฐานสำหรับเทคนิคการสื่อสารโทรคมนาคมของทหาร ตัวอย่างของมาตรฐาน คือ MIL-STD-185, MIL-STD-188A และ MIL-STD-1888 และที่ใช้กันแพร่หลาย คือ MIL-STD-188C มาตรฐานเหล่านี้เรียกว่า MIL-STD-188 Series ซึ่งใน Series นี้ อาจจะแยกออกเป็นหลายหมวด เช่น Common Long Haul/Tractical ใช้ MIL-STD-188-100 เป็นต้น

Belll System

Bell เป็นมาตรฐานที่กำหนดโดยองค์กรทางโทรศัพท์ของบริษัท Bell Laboratory มาตรฐานของBell ถูกกำหนดขึ้นเพื่อใช้ควบคุมโรงงานผู้ผลิตสินค้าที่ต้องการใช้งานร่วมกับระบบของ Bell เนื่องจาก Bell เป็นหน่วยงานที่ใหญ่มากในสหรัฐอเมริกา และมีส่วนแบ่งการขายในตลาดสูง ทำให้อุปกรณ์ที่จะใช้ร่วมกับระบบโทรศัพท์ของ Bell จะต้องเป็นไปตามมาตรฐานที่ Bell กำหนด แต่ระยะหลัง ๆ Bell เองก็เริ่มผ่อนปรน หรือแก้ข้อกำหนดของตนเองให้เข้ากับ CCITT ได้ เฉพาะบางส่วนเท่านั้น

มาตรฐานหรือข้อกำหนดของแต่ละองค์กรต่าง ๆ จะยกมาพอสังเขปดังนี้

X3.1 เป็นรายละเอียดคุณลักษณะของ Signaling rate สำหรับการสื่อสารข้อมูลระบบ Synchronous ทั้งแบบ Serial or ParallelI อัตราความเร็วที่ว่านี้ apply ที่ interface ระหว่าง DTE กับ DCE ซึ่งใช้งานบนช่องสัญญาณเสียงปกติคือ 4KHz (FIPS 22)

X3.4 Code ของ Character ทั่วไปที่ใช้งานสื่อสารระหว่างเครื่องมือชนิดต่าง ๆ (FIPS 1)

x3.24 มาตรฐานนี้ใช้สำหรับบอกให้ทราบถึงสัญญาณข้อมูลและสัญญาณเวลาระหว่างการอินเตอร์เฟส DTE กับ DCE

X3.15 กำหนดมาตรฐานของบิทใน ASCII Code แบบ Serial bit สำหรับการสื่อสารข้อมูล (FIPS 16)

X3.16 กำหนดโครงสร้างของ Character และ การตรวจจับ Character parity แบบ Serial bit ที่ใช้ ASCII ในการสื่อสารข้อมูล (FIPS 17) ทั้งระบบ Synchronous และ Asynchronous

X3.25 กำหนดโครงสร้างของ Character และการตรวจรับ Character Parity แบบ Parallel bit ใน ASCII Code ที่การสื่อสารแบบเรียงตัวอักษร (FIPS 18)

X3.28 ขั้นตอนในการใช้ Control Character ในการติดต่อควบคุมการสื่อสารข้อมูล

X3.36 กำหนดคุณลักษณะของการสื่อสารข้อมูลความเร็วสูงระบบ Synchronous และกล่าวถึงอัตราการสื่อสารความเร็วสูงระหว่าง DTE กับ DCE ซึ่งสื่อสารบนช่องสัญญาณความเร็วสูง

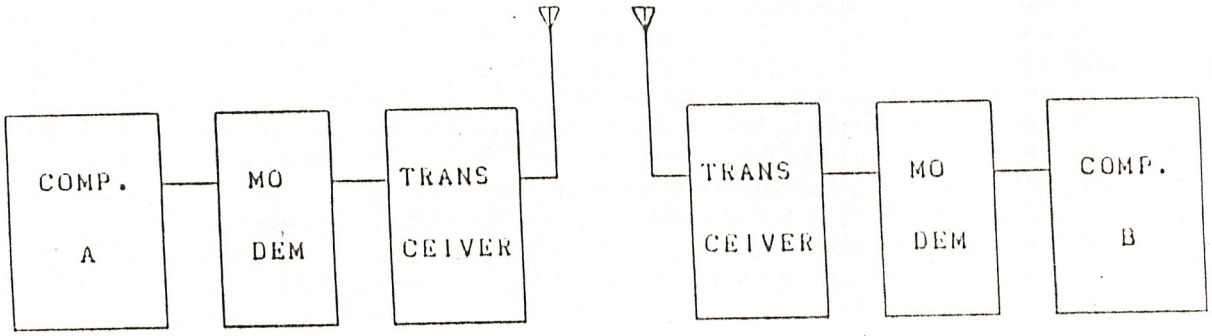
X3.44 กำหนดมาตรฐานและการทำงานของอุปกรณ์ที่การสื่อสารข้อมูลใช้เป็นตัวกลาง

2.8 หลักการทำงานของโครงงานเบื้องต้น

การส่งข่าวสารระหว่างอุปกรณ์ปลายทางข้อมูลซึ่งก็คือ คอมพิวเตอร์ส่วนบุคคล สำหรับโครงงานนี้ เป็นการสื่อสารข้อมูลโดยใช้คลื่นวิทยุนำข่าวสารข้อมูลจากต้นทางไปยังปลายทางได้โดยผ่านอากาศ ซึ่งเป็นตัวกลางนั่นเอง

การทำงานของระบบสามารถแสดงได้ดังรูปที่ 2.17

จากรูป อุปกรณ์ DTE (Data Terminal Equipment) หมายถึง อุปกรณ์ต้นทางและปลายทางที่รับส่งข้อมูล ในที่นี้คือตัวเครื่องคอมพิวเตอร์จะทำการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ โดยในขณะที่คอมพิวเตอร์ด้านหนึ่งทำการส่งข้อมูลออกไปแบบอนุกรมผ่านตัวอินเตอร์เฟส RS-232C ไปยังโมเด็ม ซึ่งโมเด็มจะทำการแปลงข้อมูลแบบดิจิตอลให้เป็นสัญญาณอะนาล็อกในย่านความถี่เสียง ด้วยวิธีการเข้ารหัสแบบความถี่ (FSK) นั่นคือ ใช้ค่าความถี่หนึ่งแทนบิท "0" และใช้ค่าความถี่อีกค่าหนึ่งแทนบิท "1"



รูปที่ 2.17 บล็อคไดอะแกรมของโครงงาน

จากนั้นสัญญาณอะนาล็อกจะถูกมอดดูเลทกับสัญญาณพาหะในส่วนของเครื่องส่ง แล้วส่งออกอากาศไป

คลื่นวิทยุที่ถูกมอดดูเลทแล้ว จะถูกส่งไปยังส่วนของเครื่องรับของคอมพิวเตอร์อีกด้านหนึ่ง โดยเครื่องรับจะทำการดีมอดดูเลทสัญญาณพาหะออกจากสัญญาณอะนาล็อก แล้วนำสัญญาณที่ได้นี้ผ่านโมเด็ม ซึ่งจะทำการแปลงสัญญาณอะนาล็อกให้กลับเป็นสัญญาณดิจิตอลแล้วส่งผ่านอินเตอร์เฟส RS-232C ไปให้คอมพิวเตอร์ทางด้านรับทำการประมวลผลต่อไป

ในส่วนของโมเด็มและเครื่องรับส่ง (TRANSCEIVER) เราเรียกว่า DCE (Data Communications Equipment) ซึ่งเป็นอุปกรณ์ที่ทำหน้าที่รับส่งข้อมูล และใช้ตัวอินเตอร์เฟส RS-232C เชื่อมต่อระหว่าง DTE กับ DCE

สำหรับโครงงานนี้ได้เลือกใช้ความถี่ย่าน FM ในการออกอากาศ คือ ความถี่ 27 MHz และความถี่ 107 MHz โดยมีลักษณะการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์

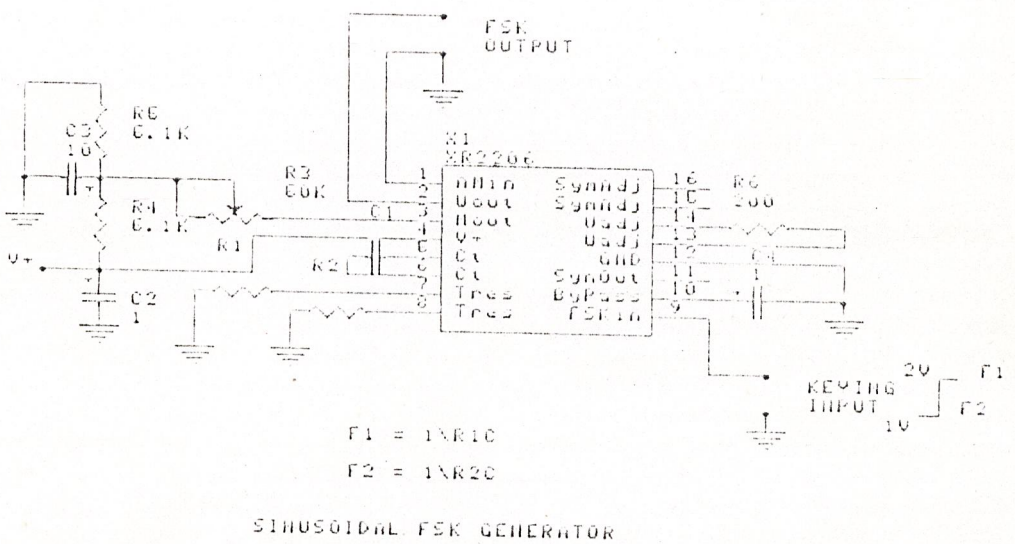
บทที่ 3

การออกแบบและการสร้าง

สำหรับการออกแบบและการสร้างวงจรในโครงงานนี้ จะประกอบไปด้วย 3 ส่วนคือ โมเด็ม เครื่องส่งและเครื่องรับวิทยุ และในส่วนของการอินเตอร์เฟสกับคอมพิวเตอร์ซึ่งรวมทั้งโปรแกรมที่ใช้ในการสื่อสารด้วย

3.1โมเด็ม (Modem)

3.1.1 ภาค Modulator



รูปที่ 3.1 วงจรภาค Modulator ใช้ไอซีเบอร์ XR-2206

จากรูปเป็นการต่อวงจร Modulator แบบ Sinusoidal FSK ใช้ไอซีเบอร์ XR 2206 เป็น Modulator และ XR 1489 เป็น Line Receiver ซึ่งสัญญาณดิจิตอลจากขา 3 ของ Line Receiver จะป้อนให้กับอินพุทของ Modulator โดยได้สัญญาณ

FSK output ที่ขา 2 ของ XR 2206 ซึ่งเป็นความถี่ 2 ระดับที่แทนสัญญาณจากอินพุท คือ High Frequency แทนสภาวะ "0" และ Low Frequency แทนสภาวะ "1" เราสามารถปรับค่า R2 เพื่อให้ได้ความถี่ 1270 Hz (Originate Modem) และ 2225 Hz (Answer Modem) ซึ่งแทนสภาวะ "0" และปรับ R1 เพื่อให้ได้ความถี่ 1070 Hz (Origanate Modem) และ 2025 Hz (Answer Modem) ซึ่งแทนสภาวะ "1" ส่วน R3 ใช้ปรับค่า Out Level ของสัญญาณ FSK

จาก f1 = 1/R1C และ f2 = 1/R2C เมื่อกำหนดให้ C = 0.1uF

Originate Modem

ที่ความถี่ 1070 Hz R1 = 1/f1C = 1/(1070×0.1×10-6) = 9.345 kohms

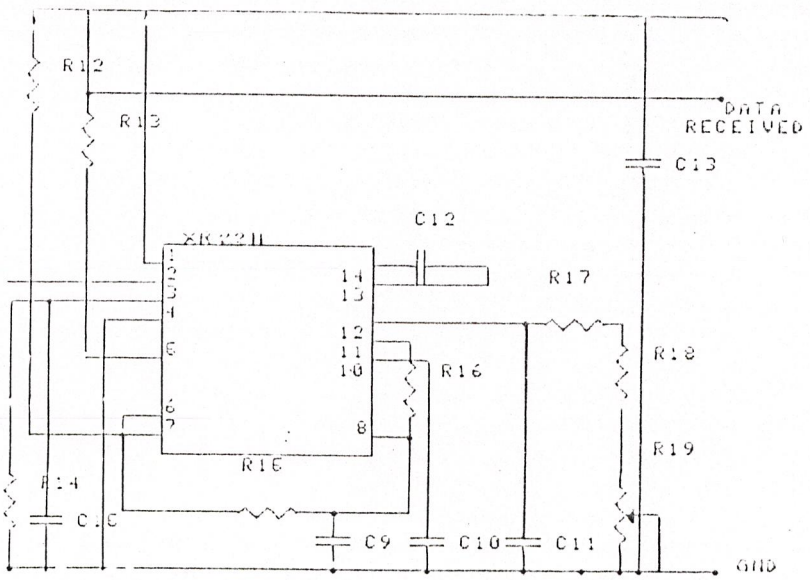
ที่ความถี่ 1270 Hz R2 = 1/f2C = 1/(1270×0.1×10-6) = 7.874 kohms

Answer Modem

ที่ความถี่ 2025 Hz R1 = 1/f1C = 1/(2025×0.1×10-6) = 4.938 kohms

ที่ความถี่ 2225 Hz R2 = 1/f2C = 1/(2225×0.1×10-6) = 4.494 kohms

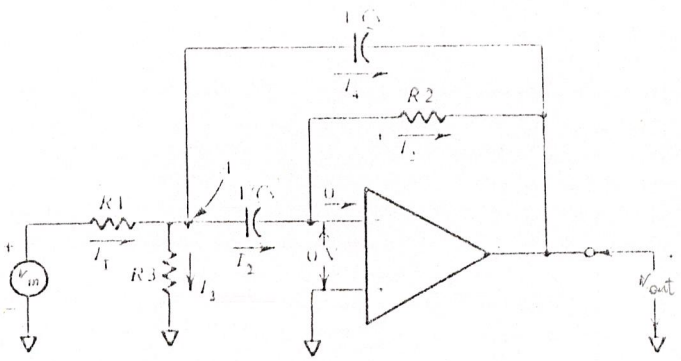
3.1.2 ภาค Demodulator



รูปที่ 3.2 วงจรภาค Demodulator ใช้ไอซีเบอร์ XR-2211

วงจร Demodulator ใช้ไอซีเบอร์ XR 2211 ขาที่ 3 เป็น FSK input สัญญาณเอาท์พุทได้ที่ขา 6,7 ซึ่งเป็นสัญญาณดิจิตอล ในการดีโค้ดสัญญาณ FSK เมื่ออินพุทเป็นความถี่ต่ำ Data Output จะมีสภาวะ High เมื่ออินพุทเป็นความถี่สูง Data Outputจะมีสภาวะ Low โดยเอาท์พุทที่ได้นี้จะป้อนให้กับขาที่ 2 ของ XR 1488 ซึ่งเป็น Line Driver เอาท์พุทจาก Line Driver จะป้อนไปยังขาที่ 3 ของ RS 232C (Rx) ขาที่ 5 ของ XR 2211 เป็นสัญญาณ Carrier Detect ป้อนให้กับขาที่ 4,5 ของ Line Driver จะได้เอาท์พุทป้อนไปยังขาที่ 8 ของ RS 232C ซึ่งเป็นขา Data Carrier Detect และเมื่อไม่มี Carrier ป้อนเข้ามา Data Output จะเป็น Low

3.1.3 วงจร Bandpass Filter



รูปที่ 3.3 วงจร Bandpass Filter

จากรูปเป็น Single Op Amp Bandpass Filter สามารถคำนวณหาทรานส์เฟอร์ฟังชั่นได้ดังนี้

จาก

$$I_2 = -\frac{V_{out}}{R2}$$

ที่ node A เมื่อเทียบกับกราวนด์ แรงดันที่ตกคร่อมคาปาซิเตอร์คือ

$$v_A = \frac{I_2}{Cs}$$

$$v_A = -\frac{v_{out}}{R2Cs}$$

กระแสที่ไหลผ่าน R3 คือ I3

$$I_3 = \frac{v_A}{R3}$$

$$I_3 = -\frac{v_{out}}{R2R3Cs}$$

กระแสที่ไหลผ่านคาปาซิเตอร์ฟีดแบ็คคือ I4

$$I_4 = -\frac{v_A - v_{out}}{\frac{1}{Cs}} = -(v_A - v_{out})(Cs)$$

$$= -\frac{v_{out}(R2Cs + 1)}{R2}$$

กระแสอินพุทที่ไหลผ่าน R1

$$I_1 = \frac{v_{in} - v_A}{R1}$$

$$I_1 = \frac{v_{in}}{R1} + \frac{v_{out}}{R1R2Cs}$$

ที่ node A กระแสที่ไหลเข้าเท่ากับกระแสที่ไหลออก

$$I_1 = I_2 + I_3 + I_4$$

แทนค่า

$$\frac{v_{in}}{R1} + \frac{v_{out}}{R1R2Cs} = -\frac{v_{out}}{R2} - \frac{v_{out}}{R2R3Cs} - \frac{v_{out}(R2Cs + 1)}{R2}$$

สามารถจัดรูปแบบได้

$$\frac{v_{out}}{v_{in}} = \frac{\left(\frac{1}{-R1C}\right)s}{s^2 + \frac{2}{R2C}s + \frac{R1 + R3}{R1R2R3C^2}}$$

เมื่อเทียบกับทรานส์เฟอร์ฟังชั่นของ Second Order Bandpass Filter

$$\frac{V_o}{V_{in}} = \frac{A_o \alpha \omega_o s}{s^2 + \alpha \omega_o s + \omega_o^2}$$

เมื่อ

α = damping coefficient
ω_o = center frequency = critical frequency
A_o = gain at the center

จะได้

$$\omega_o^2 = \frac{R1 + R3}{R1R2R3C^2}$$

$$\omega_o = \sqrt{\frac{R1 + R3}{R1R2R3C^2}}$$

$$f_c = \frac{1}{2\pi}\sqrt{\frac{R1 + R3}{R1R2R3C^2}}$$

$$\alpha\omega_o = \frac{2}{R2C}$$

$$A_o\alpha\omega_o = -\frac{1}{R1C}$$

เพราะฉะนั้นจะหา Aoได้

$$A_o = -\frac{R2}{2R1}$$

เกนที่ Center Frequency นี้จะขึ้นอยู่กับอัตราส่วนของ Feedback Resistor กับ Input Resistor ซึ่งเหมือนกับวงจร Inverting Amplifier

จาก $$\alpha\omega_o = \frac{2}{R2C}$$

$$\alpha = \frac{2}{R2\omega_o C}$$

ค่า Selectivity กำหนดได้จากอัตราส่วนของ Center frequency ต่อ Bandwidth

$$Q = \frac{f_o}{\Delta f}$$

ซึ่งก็คือส่วนกลับของ Damping coefficient

$$Q = \frac{1}{\alpha}$$

ดังนั้น

$$Q = \frac{R2\omega_o C}{2}$$

เพราะฉะนั้น Bandwidth

$$Q = \frac{\omega_o}{B} = \frac{R2\omega_o C}{2}$$

$$B = \frac{2}{R2C}$$

$$\Delta f = \frac{2}{2\pi R2C}$$

ตัวอย่าง จงออกแบบวงจร Bandpass filter เมื่อกำหนดให้

Q = 24

fl = 2025 Hz , fh = 2225 Hz

Ao = -3.75

C1 = C2 = C = 0.01uf

หาความถี่ Center freqquency จากสูตร

$$f_c = \sqrt{f_h * f_l} = \sqrt{2225 * 2025} = 2122.65 \text{ Hz}$$

จาก $Q = 1/\zeta$

$$\zeta = 0.042$$

จาก $\zeta\omega_o = 2/(R3C)$

$$R2 = 2/(\zeta\omega_o C) = 2/(0.042 * 6.28 * 2122.65 * 0.01\text{uF})$$

$$= 357.225\text{K} = 357\text{K}$$

จาก $A_o\zeta\omega_o = -1/(R1C)$

$$R1 = -1/(A_o\zeta\omega_o C)$$

$$= -1/(-3.75 * 0.042 * 6.28 * 2122.65 * 0.01\text{uF})$$

$$= 47.63\text{K} = 47.5\text{K}$$

จาก $\omega_{o2} = (R1+R3)/(R1R2R3C2)$

$$(R1+R3)/R3 = R1R2C2\omega_{o2}$$

$$(R1/R3)+1 = R1R2C2\omega_{o2}$$

$$R1/R3 = R1R2C2\omega_{o2}-1$$

$$R3 = R1/(R1R2C2\omega_{o2})$$

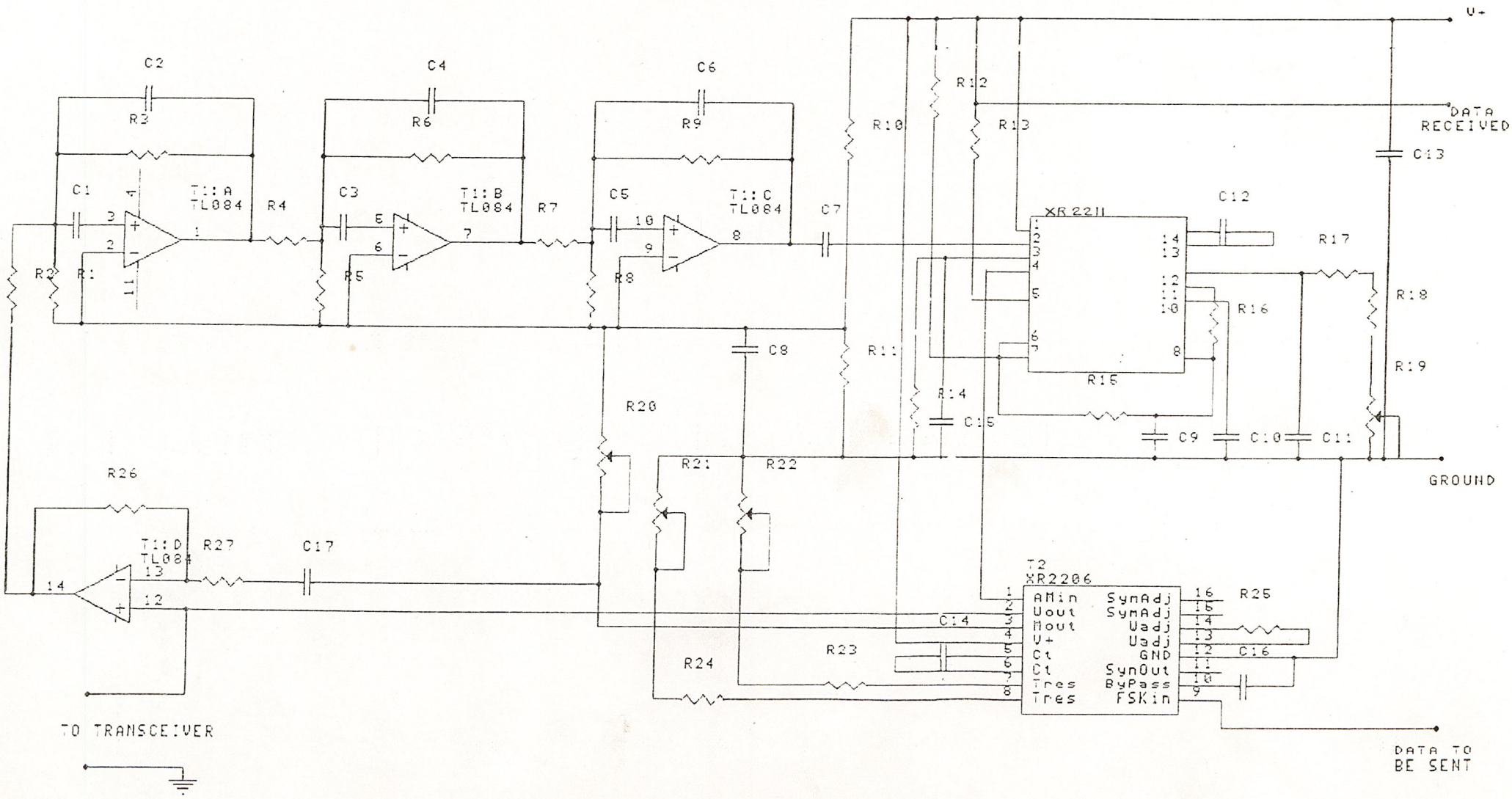
$$= 47.63\text{K}/(47.63\text{K} * 357.225\text{K} * (0.01\text{uF})2$$

$$* (6.28 * 2122.65)2 - 1)$$

$$= 157.64 = 191$$

เนื่องจากวงจร Bandpass filter 1 stage จะได้อัตรา rolloff ที่เอาท์พุทเป็น 20 dB/decade เพราะฉะนั้นเพื่อให้ได้ curve ที่เอาท์พุทมีความชันมากขึ้น คือให้ได้เฉพาะความถี่ที่ต้องการเท่านั้น จึงใช้วงจร Bandpass filter ต่อ cascade กัน 3 stage ซึ่งจะได้อัตรา rolloff เท่ากับ 60 dB/decade

FSK MODEM USING XR-2206 AND XR-2211



ตารางแสดงค่าอุปกรณ์ต่าง ๆ ของโมเด็ม

IC1A-D	XR-084
IC2	XR-2211
IC3	XR-2206

	ANSWER	ORIGINATE
R1*	40.2K	47.5K
R2*	499	191
R3*	270K	357K
R4*	60.4K	39.4K
R5*	680	160
R6*	383K	270K
R7*	24.9K	20K
R8*	1.21K	360
R9*	160K	160K
R10	1K	1K
R11	1K	1K
R12	5.1K	5.1K
R13	5.1K	5.1K
R14	510K	510K
R15	510K	510K
R16	100K	100K
R17	47K	100K
R18	7.5K	9.1K

R19	2K	2K
R20	50K	50K
R21	2K	2K
R22	2K	2K
R23	3.9K	8.2K
R24	3.6K	6.8K
R25	200	200
R26	1M	1M
R27	1M	1M
C1-C6*	0.01uF	0.01uF
C7	0.1uF	0.1uF
C8	22uF	22uF
C9	0.01uF	0.01uF
C10	0.1uF	0.1uF
C11	0.022uF	0.01uF
C12	0.1uF	0.047uF
C13	1uF	1uF
C14	0.1uF	0.1uF
C15	0.1uF	0.1uF
C16	1uF	1uF
C17	1uF	1uF

ค่าความต้านทานทุกตัวใช้ 1/4 W ค่าผิดพลาด 5% ยกเว้นที่มีเครื่องหมาย (*) ค่าผิดพลาด 1% หน่วยเป็นโอห์ม ส่วนคาปาซิเตอร์ค่าผิดพลาด 5% ยกเว้นที่มีเครื่องหมาย (*) ค่าผิดพลาด 1%

3.2 เครื่องรับส่งวิทยุ (Transceiver)

3.2.1 ภาครับ (Receiver)

การทำงานเริ่มจากความถี่อาร์เอฟ 27.125 MHz ที่ส่งมาจากเครื่องส่งอีกชุดหนึ่งจะเข้ามายังสารอากาศผ่านวงจรกรองความถี่ต่ำผ่าน ซึ่งประกอบด้วย L10,L11, L12,C58,C59 และ C60 เพื่อทำหน้าที่กรองเอาเฉพาะความถี่ 27.125 MHz เท่านั้น แต่อาจจะมีความถี่อื่นแทรกผ่านไปได้บ้างแต่ปริมาณที่ไม่มาก

จากวงจรกรองความถี่ต่ำผ่านจะถูกส่งมายังชุดกรองความถี่อาร์เอฟ และ ภาคขยายอาร์เอฟ มี C1 เป็นตัวคัปปลิ้งสัญญาณ D1และD2 ทำหน้าที่เป็นวงจรคลิปเปอร์และป้องกันสัญญาณที่แรงเกินไปซึ่งอาจจะเป็นอันตรายต่อ L1 และวงจรต่อไปได้ L1,C2,C5, L2 ทำหน้าที่จูนรับเอาความถี่ 27.125 MHz เท่านั้น ส่วนความถี่อื่นจะถูกบาย พาสลงกราวนด์ไป มี C4 คัปปลิ้งสัญญาณระหว่างคอยล์ทั้งสอง C6 คัปปลิ้งสัญญาณที่ผ่านการจูนเอาเฉพาะความถี่เข้าสู่ขาเบสของ Q1 ซึ่งเป็นภาคขยายความถี่อาร์เอฟ

ในบางครั้งสัญญาณที่รับเข้าได้ จะมีความแรงของสัญญาณที่ต่ำมาก (ยิ่งเป็นการส่งระยะทางไกล ๆ จะยิ่งมีสัญญาณที่เบามาก) จึงต้องทำการขยายสัญญาณนั้นให้มีความแรงขึ้นมาอยู่ในระดับหนึ่งก่อน โดยมี R2 ทำหน้าที่ไบแอสกระแสให้ขาเบส มี L3 ปรับอัตราการขยายความถี่อาร์เอฟของ Q1 และปรับระดับสัญญาณ เพื่อการผสมสัญญาณในภาคมิเซอร์ด้วย

ทรานซิสเตอร์ Q3 ทำหน้าที่เป็นชุดกำเนิดความถี่มูลฐานทำงานร่วมกับคริสตอล X'TAL1a และ X'TAL1b มี R3 และ R4 ต่อเป็นวงจรขยายความถี่คอมมอนคอลเลคเตอร์ C9 และ C10 ทำหน้าที่รักษาเสถียรภาพการกำเนิดความถี่ของคริสตอล

ตัวเก็บประจุ C12 จะคัปปลิ้งความถี่ออสซิลเลเตอร์มาเข้าสู่ขาเกต G1 ของ Q2 เพื่อการผสมความถี่กับความถี่อาร์เอฟที่คัปปลิ้งผ่าน L3 มาเข้าขาเกต G2 ของ Q2 ซึ่ง Q2 นี้เป็นทราซิสเตอร์มอสเฟตแบบเกตคู่ (Dual gate mosfet) สัญญาณที่ออกไปทางขาซอส จะเป็นความถี่ที่ถูกลดลงมาเป็นความถี่ไอเอฟ ค่าความถี่เท่ากับ 1.7 MHz L4 ทำหน้าที่ปรับอัตราการขยายและเรโซแนนท์ความถี่ของ Q2 ความถี่ไอเอฟนี้จะผ่าน CFU2

เพื่อกรองเอาเฉพาะความถี่ 10.7 MHz เข้าสู่ขา 16 ของ IC1 ซึ่งภายในจะประกอบไปด้วยวงจรมิกเซอร์ชุดที่สองและออสซิลเลเตอร์ชุดที่สอง จึงทำให้การออกแบบวงจรในส่วนหลังนี้ค่อนข้างง่าย

ดังนั้นภาคมิกเซอร์และภาคออสซิลเลเตอร์ภายในก็จะทำการผสมสัญญาณความถี่ไอเอฟ 10.7 MHz เข้ากับความถี่ออสซิลเลเตอร์มูลฐานที่ได้จากคริสตอล X'TAL2 ความถี่ 10.24 MHz ที่ต่ออยู่ที่ขา 1 และ 2 ของ IC1 มี C17 เป็นตัวรักษาความถี่ออสซิลเลเตอร์ จากนั้นจะเหลือความถี่ไอเอฟ 455 KHz จะออกมาทางขา 3 ของ IC1 เข้าสู่ชุดรกรองความถี่ CFU1

ที่ชุดกรองความถี่ CFU1 นี้ จะกรองเอาเฉพาะความถี่ 455 KHz ผ่านเท่านั้นเข้าสู่ขาของ IC1 โดยมี C20,C21 ทำหน้าที่เป็นวงจรดีคลิปเปอร์ ขดลวด L5, R7 ที่ต่ออยู่กับขา8ของ IC1 ทำหน้าที่เป็นควอดราเจอร์ดีเท็กเตอร์และทำการดีเท็กสัญญาณเอฟเอ็มออกมาทางขา 9 ผ่านวงจรกรองสัญญาณ R11,R12,C26,C27 และ C28 จะคัปปลิ้งผ่าน C28 เข้าสู่ภาคขยายเสียงโดยมีโวลุ่ม VR2 ทำหน้าที่เร่งลดระดับแอมปลิจูด

ที่ขา 2 ของ IC2 จะเป็นขาอินพุทของภาคขยายเสียง มี C29 ต่อระหว่างขา 1 กับขา 8 เพื่อเป็นวงจรป็อนกลับสัญญาณ R16,C30 ทำหน้าที่ป้องกันการออสซิลเลตที่ความถี่สูง C31 คัปปลิ้งสัญญาณทางเอาท์พุทออกสู่โมเด็มต่อไป

3.2.2 ภาคส่ง (Transmitter)

ในส่วนของภาคส่งจะเริ่มจากข้อมูลที่ต้องการจะส่งหลังผ่านการมอดดูเลทแล้วนั้นโดยที่มี R18,C34 ทกหน้าที่กรองเอาเฉพาะความถี่เสียงเท่านั้นผ่าน C35 คัปปลิ้งสัญญาณไปเข้าขาเบสของ Q6 มี R20,R21,C36 ไบแอสให้ และ R19 ทำหน้าที่ป้อนกลับสัญญาณเพื่อกำหนดอัตราการขยายของ Q6

สัญญาณเสียงจาก Q6 จะถูกส่งมาเข้าขาเบสของ Q7 ซึ่งทำหน้าที่เป็นวงจรออสซิลเลเตอร์ผลิตความถี่ 9.0416 MHz โดยต่ออนุกรมกับวงจรจูน L6,VC1 ซึ่งจะเปลี่ยนไปตามค่าของแรงดันสัญญาณเสียงที่เข้ามา กล่าวคือเมื่อไม่มีสัญญาณมาที่อินพุท

แรงดันที่ขาแคโถดของ VC1 จะวัดได้ประมาณ 2.0 ถึง 2.5 โวลท์ แต่พอมีสัญญาณเข้ามาแรงดันจะเปลี่ยนแปลงขึ้นลงตามสัญญาณอินพุท

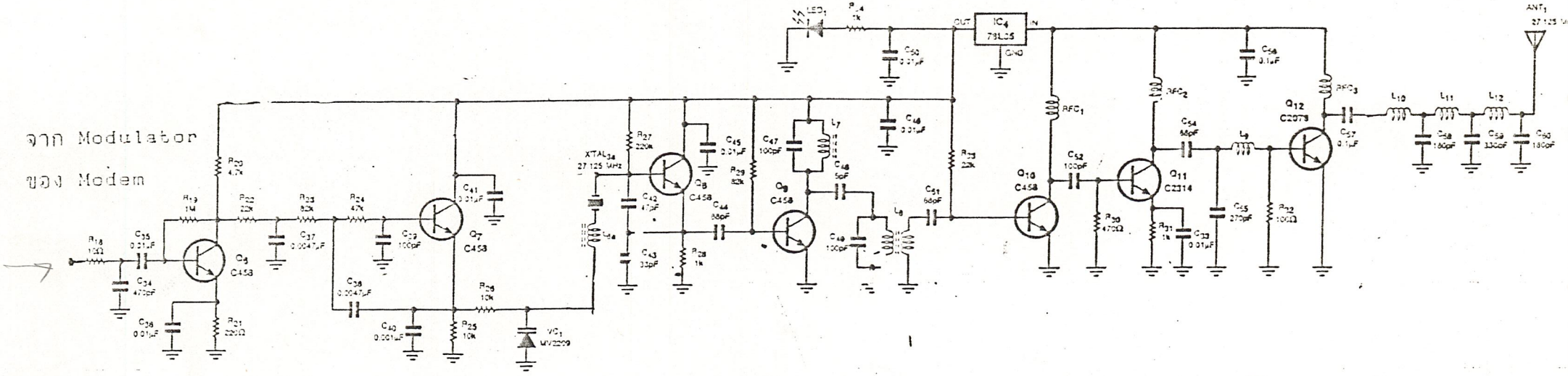
เนื่องจากความถี่ออสซิลเลเตอร์ที่กำหนดขึ้นมาจากความถี่มูลฐานในออร์เดอร์ที่หนึ่งของคริสตอล X'TAL3 มีค่าความถี่เท่ากับ 9.0416 MHz เท่านั้น จึงต้องอาศัย Q8 ทำหน้าที่เป็นวงจรทวีคูณความถี่ฮาร์มอนิกที่สองของ X'TAL3 ส่งความถี่ที่ได้ผ่าน C44 มาเข้าขาเบสของ Q9 ซึ่งทำหน้าที่เป็นวงจรทวีคูณความถี่ฮาร์มอนิกที่สามของ X'TAL3 โดยมีวงจรแทงก์ L7,C47 ทำหน้าที่จูนให้ได้ความถี่เรโซแนนท์ที่ 27.125 MHz แต่เอาท์พุทของ Q8 ที่ขาคอลเลคเตอร์จะมีความถี่ฮาร์มอนิกที่หนึ่ง 9.0416 MHz และฮาร์มอนิกที่สอง 18.0832 MHz ปะปนออกมาด้วย

จากความถี่ที่ปะปนมากับความถี่ 27.125 MHz ซึ่งเราต้องการเพียงความถี่หลังเท่านั้น จึงต้องทำการกำจัดความถี่ที่ปะปนออกมาด้วยออกไปโดยใช้วงจรแทงก์ L8,C49 ต่ออยู่ในลักษณะดับเบิลจูนก็จะได้เฉพาะความถี่ 27.125 MHz คัปปลิ้งผ่าน C51 ไปเข้าภาคปรีไดรเวอร์ต่อไป

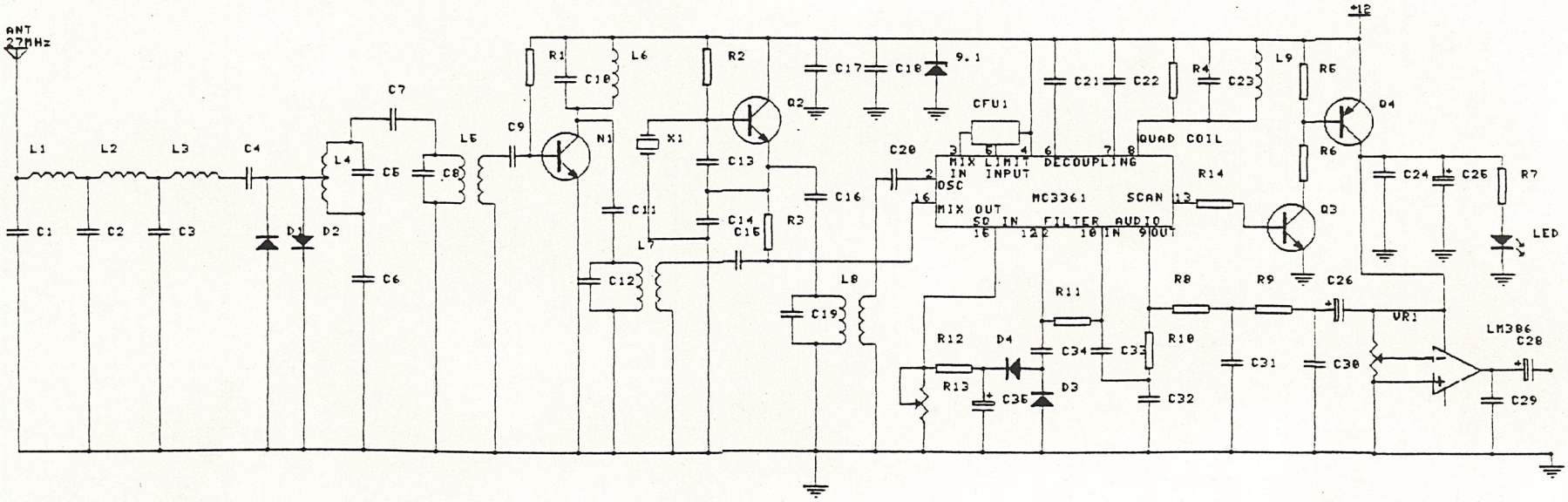
ทรานซิสเตอร์ Q10 ทำหน้าที่ขยายความถี่อาร์เอฟให้มีความแรงอยู่ในระดับหนึ่งก่อนโดยมี R33 ไบแอสกระแสให้เอาท์พุทที่ขาคอลเลคเตอร์จะถูกคัปปลิ้งผ่าน C52 ไปเข้าขาเบสของ Q11 จะต่อ R31และ C33 ลงกราวนด์อยู่เพื่อเป็นการจำกัดอัตราขยายของภาคเพาเวอร์ไม่ให้เกิน 1 วัตต์ Q12 เป็นเพาเวอร์เอาท์พุททำหน้าที่ขยายสัญญาณความถี่อาร์เอฟออกสู่อากาศโดยคัปปลิ้งผ่าน C57และวงจรกรองความถี่ต่ำผ่าน L10-L12

วงจรเครื่องส่งวิทยุ

จาก Modulator
ของ Modem



วงจรเครื่องรับวิทยุ 27 MHz



3.3 โปรแกรมรับส่งข้อมูลผ่านพอร์ตอนุกรม

โปรแกรมที่ใช้ในการรับส่งข้อมูลโดยผ่านพอร์ตอนุกรม (RS232C) โดยเป็นแบบอะซิงโครนัส (Asynchronous) เป็นการสื่อสารผ่านพอร์ตแบบอนุกรมครั้งละ 1 บิท โดยระยะเวลาของการส่งข้อมูลอะซิงโครนัสในแต่ละไบต์ไม่จำเป็นต้องเท่ากัน

ในการส่งข้อมูลผ่านพอร์ตอนุกรมนั้น ข้อมูลแต่ละไบต์จะประกอบด้วย

1. บิตเริ่มต้น (start bit) 1 บิต
2. บิตข้อมูล (data bit) 7 หรือ 8 บิต
3. พาริตี้บิต (parity bit) (จะมีหรือไม่ก็ได้)
4. บิตสิ้นสุด (stop bit) 1 หรือ 2 บิต

สถานะของสายส่งในขณะไม่มีข้อมูลจะมีสถานะเป็นสูง (สถานะทางดิจิตอลมี 2 สถานะคือ สูง(high) และต่ำ(low) ข้อมูลบิตใดมีค่า 0 จะทำให้สายส่งมีสถานะต่ำ ข้อมูลบิตใดมีค่า 1 ก็จะทำให้สายส่งมีสถานะสูงอยู่เช่นเดิม บิตเริ่มต้นใช้สำหรับบอกจุดเริ่มต้นของไบต์ของข้อมูล โดยการทำให้สถานะของสายส่งมีค่าต่ำ เป็นเวลา 1 รอบ(cycle) จากนั้นจะเป็นบิตของข้อมูล ตามด้วยพาริตี้ ซึ่งจะมีหรือไม่ก็ได้ สุดท้ายคือ บิตสิ้นสุด ซึ่งจะมี 1 หรือ 2 บิตก็ได้ ขึ้นอยู่ว่าจะใช้เท่าใด

พาริตี้บิต ถ้าหากมีในไบต์ข้อมูลก็จะทำหน้าที่ตรวจเช็คความผิดพลาดของข้อมูล พาริตี้มี 2 อย่างคือ เป็น คู่ หรือ คี่ (even or odd) ถ้าเป็นคู่หมายความว่า เมื่อรวมพาริตี้บิตแล้วจำนวนของบิตข้อมูลที่มีค่าเป็น 1 จะเป็นจำนวนคู่และถ้าพาริตี้บิตเป็นคี่ หมายความว่าเมื่อรวมพาริตี้บิตแล้วจำนวนของบิตที่มีค่าเป็น 1 จะเป็นจำนวนคี่

การเตรียมสถานะเริ่มต้นของพอร์ต

ฟังก์ชัน initial() ซึ่งอยู่ใน postman.c จะทำการจัดจอภาพ และทำการอ่านข้อมูลของpostman.ini(ซึ่งถูกสร้างโดย SETUP.C) มารอขั้นตอนขอการกด function key (F1,F2,F3,F4,F5,F6)เมื่อมีการกดก็จะกระโดดไปทำที่ฟังก์ชั่นที่เราต้องการ ดังการทำงานของโปรแกรมทั้ง 4 แบบดังนี้คือ

1. Transmit files เป็นการส่งข้อมูลที่เป็นไฟล์ (text fille หรือ binary file ก็ได้)โดยจะเรียกใช้ไฟล์ send_file ในtrans.c มารับชื่อไฟล์มาทำการเปิดไฟล์ที่รับเข้ามา หลังจากนั้นก็ทำการ ส่งชื่อ ส่งขนาดของไฟล์ ส่งข้อมูลของไฟล์ เมื่อสิ้นสุดของการส่งไฟล์ก็ทำการปิดไฟล์ แล้วกลับเข้าสู่โปรแกรมหลัก

2. Transmit screen เป็นการส่งข้อความ มีหลักการทำงานคล้ายกันโดยจะเรียกไฟล์ edit() อยู่ใน editor.c มาจัดจอใหม่สำหรับส่งข้อความและจะเก็บลงไปใน buf(เป็นตัวเก็บข้อมูลที่จะส่ง) แล้วทำการส่งข้อมูลโดยใช้หลักการใช้ software ในการตรวจสอบความพร้อมของข้อมูล

3. Received files เป็นการรับข้อมูลที่เป็น ไฟล์โดยการเรียกใช้ฟังก์ชั่น rec_file() ใน trans.c จะทำการรับชื่อไฟล์ ขนาดของไฟล์และทำการเขียนข้อมูลลงในไฟล์ที่เปิดขึ้นใหม่เพื่อการเก็บข้อมูล

4. Received screen เป็นการรับข้อมูลที่เป็น ข้อความที่แสดงที่จอภาพทางด้านส่งโดยรับข้อมูลมาจากด้านส่ง rec() ในtrans.c ซึ่งจะทำการตรวจสอบข้อมูลเหมือนกันแล้วแสดงข้อมูลทางจอภาพ

การใช้โปรแกรม Postman

การใช้งานของโปรแกรม Postman นี้ได้ออกแบบให้สะดวกต่อการใช้งานมากและได้มีการยืดหยุ่นการใช้งานของโปรแกรมนี้เพื่อให้สามารถที่จะนำไปใช้โอนย้ายข้อมูลระหว่างเครื่องคอมพิวเตอร์ได้อีกด้วย ทั้งยังสามารถที่จะเปลี่ยนแปลงสถานะของพอร์ตได้ตลอดได้อีกด้วย

หมายเลขประจำบิต	ความหมาย
7,6,5	อัตรารับส่งข้อมูล 000 = 110 baud 001 = 150 baud 010 = 300 baud 011 = 600 baud 100 = 1200 baud 101 = 2400 baud 110 = 4800 baud 111 = 9600 baud
3	พาริตี้ 00 หรือ 10 = ไม่มีพาริตี้ 01 = พาริตี้คี่ 11 = พาริตี้คู่

2	จำนวนของบิตสิ้นสุด 0 = 1 บิตสิ้นสุด 1 = 2 บิตสิ้นสุด
1,0	จำนวนบิตในข้อมูล 1 ไบต์ 10 = 7 บิต 11 = 8 บิต

การเตรียมสถานะของพอร์ตอนุกรมในโปรแกรม Postman นั้นสามารถที่จะเลือกพอร์ตได้และทั้งยังสามารถเลือก ความเร็วในการส่ง ,Start bit ,Stop bit Data bit parity bit โดยค่าที่จะใส่นั้นจะต้องเป็นเลขฐานสิบก่อน โดยดูจากในหน้าที่แล้ว

ตัวอย่างการใช้ตารางในการกำหนดสถานะของพอร์ตอนุกรมในโปรแกรมโดยสมมุติ ความเร็วของการส่งเป็น 300 baud มีพาริตี้คู่ มี 1 บิตสิ้นสุด และใช้ 8 บิตต่อ 1ไบท์ ข้อมูลจะได้ รูปแบบของรหัสดังตารางข้างล่าง

บิตที่	7	6	5	4	3	2	1	0
รหัส	0	1	0	1	1	0	1	1

ที่ด้านล่างของจอจะแสดง Function Key ซึ่งมีทั้งหมดอยู่ 6 Key ด้วยกันคือ

F1 การส่งข้อมูลที่เป็นไฟล์

F2 การส่งข้อมูลที่เป็นข้อความบนจอภาพ

F3 การรับข้อมูลที่เป็นไฟล์

F4 การรับข้อมูลที่เป็นข้อความแสดงบนจอภาพ

F5 แสดงข้อมูลของโปรแกรม

F6 ต้องการออกสู่ main menu หรือ ออกจากโปรแกรม

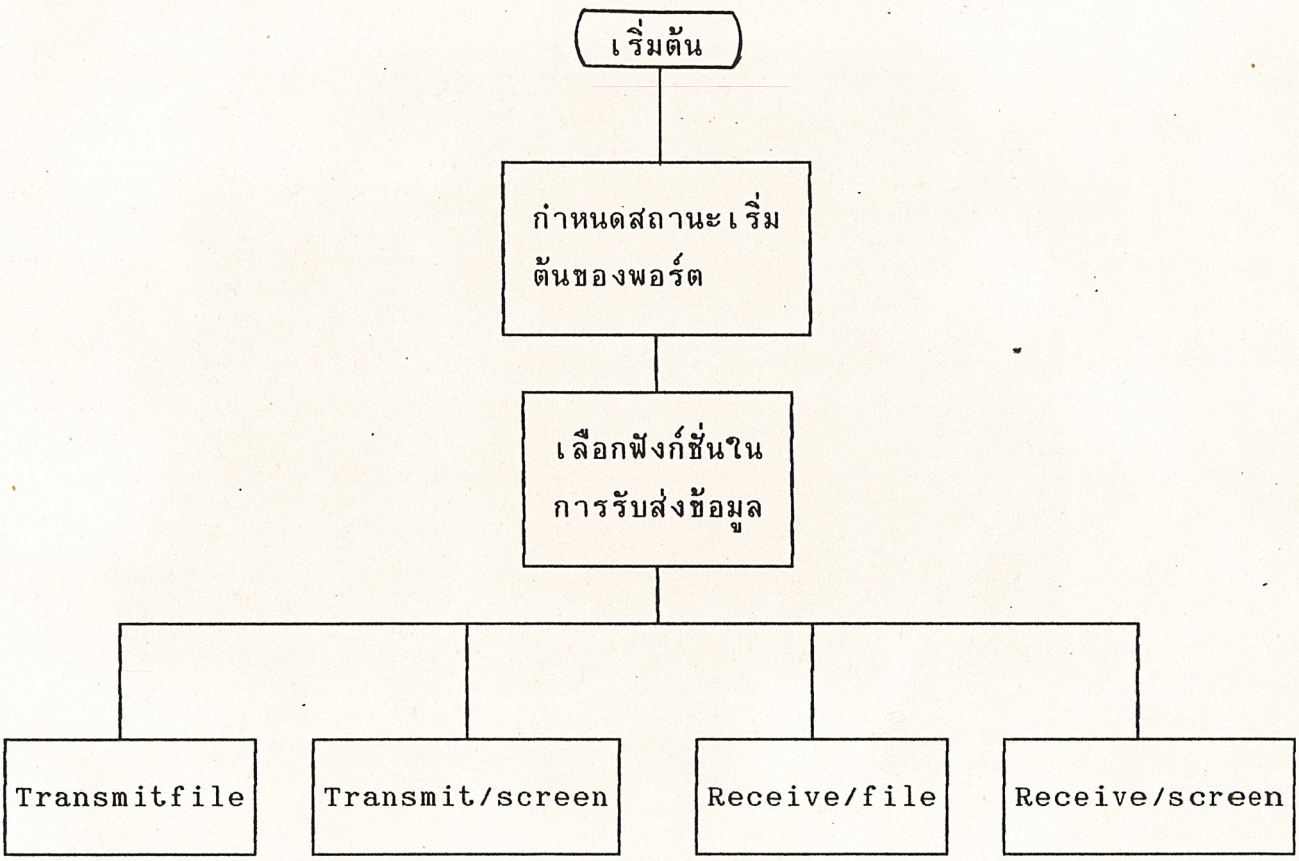
ที่ด้านล่างขวาของจอภาพจะแสดง สภาวะการทำงาน (STATUS) ของโปรแกรม ว่าอยู่ กำลังทำงานอยู่ใน Mode ใด

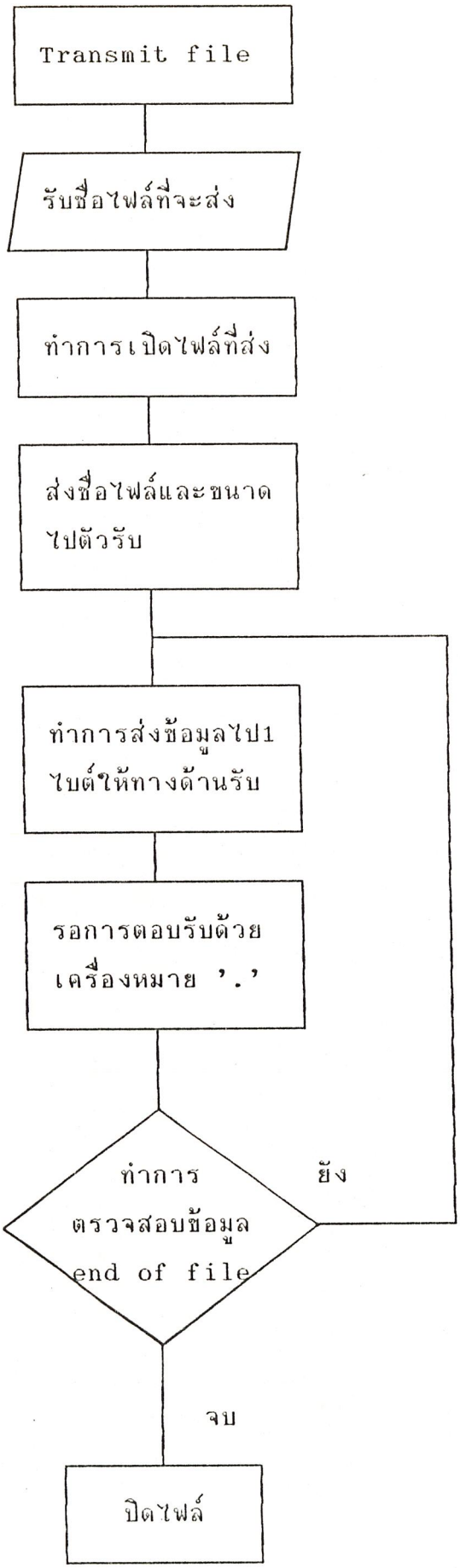
การรับส่งข้อมูลที่เป็นไฟล์

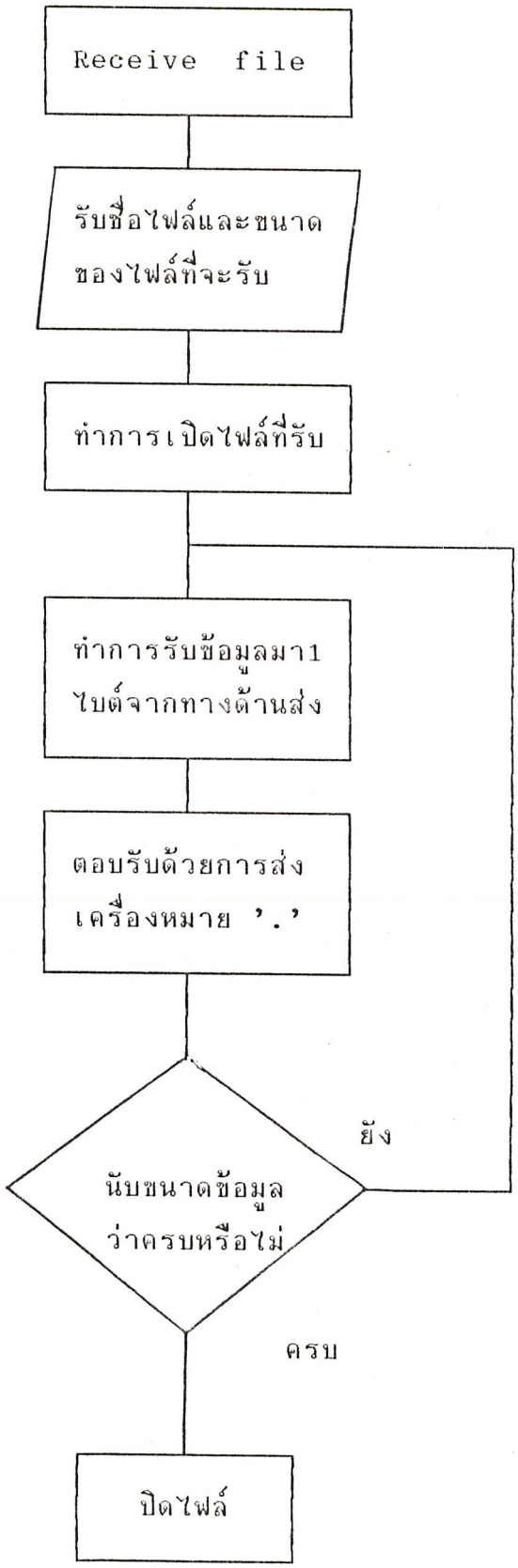
การรับส่งข้อมูลที่เป็นไฟล์ ให้ทางด้านเครื่องรับเตรียมพร้อมด้วยการกด F3 ที่ด้านรับ (เมื่อกดF3แล้วหน้าจอจะแสดงดังรูป) แล้วทำการไปเตรียมด้านส่งโดยการกด F1 ที่หน้าจอภาพถามชื่อไฟล์ที่จะส่งดังรูป เมื่อป้อนชื่อเสร็จกด Enter เครื่องทำการส่งข้อมูล โดยจะแสดงชื่อไฟล์ก่อน เมื่อทำการส่งเรียบร้อยแล้ว ที่ด้านล่างจะขึ้น Success แสดงว่า การส่งข้อมูลถูกต้อง

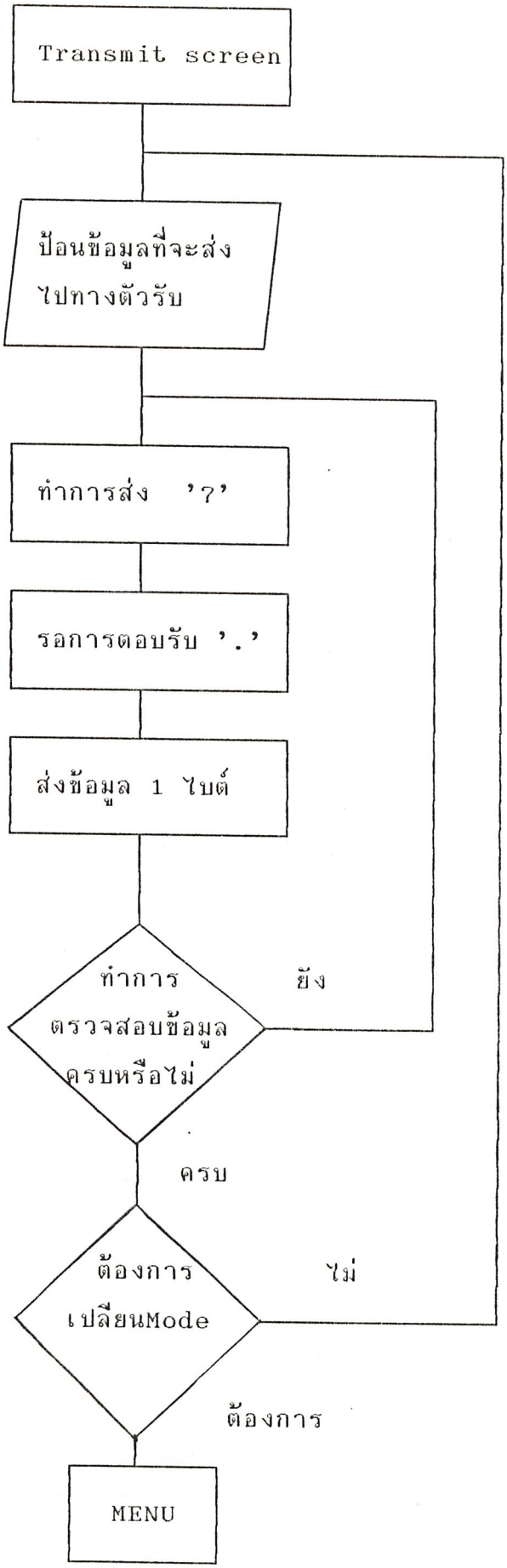
การรับส่งข้อมูลเป็นข้อความที่จอภาพ

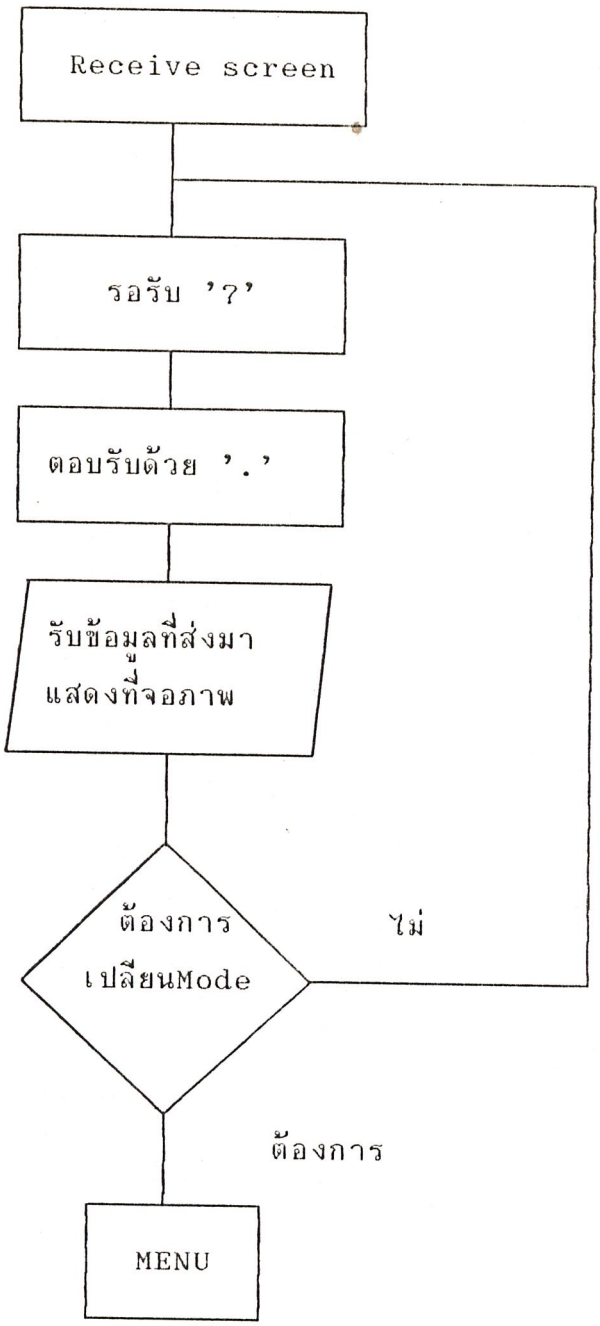
การรับส่งข้อมูลเป็นข้อความที่จอภาพ เริ่มเตรียมเครื่องรับโดยการกด F4 ที่หน้าจอจะแสดงดังรูปพร้อมรับข้อความ ส่วนทางด้านเครื่องส่งเริ่มกด F2 หลังจากนั้นหน้าจอภาพจะเปลี่ยนไปพร้อมที่จะส่งข้อความ เมื่อพิมพ์ข้อความเสร็จ กด Enter ข้อความก็จะไปแสดงทางด้านเครื่องรับ




Transmit file
รับชื่อไฟล์ที่จะส่ง
ทำการเปิดไฟล์ที่ส่ง
ส่งชื่อไฟล์และขนาด
ไปตัวรับ
ทำการส่งข้อมูลไป1
ไบต์ให้ทางด้านรับ
รอการตอบรับด้วย
เครื่องหมาย '.'
ทำการ
ตรวจสอบข้อมูล
end of file
ยัง
จบ
ปิดไฟล์


Receive file
รับชื่อไฟล์และขนาด
ของไฟล์ที่จะรับ
ทำการเปิดไฟล์ที่รับ
ทำการรับข้อมูลมา1
ไบต์จากทางด้านส่ง
ตอบรับด้วยการส่ง
เครื่องหมาย '.'
นับขนาดข้อมูล
ว่าครบหรือไม่
ยัง
ครบ
ปิดไฟล์


Transmit screen
ป้อนข้อมูลที่จะส่ง
ไปทางตัวรับ
ทำการส่ง '?'
รอการตอบรับ '.'
ส่งข้อมูล 1 ไบต์
ทำการ
ตรวจสอบข้อมูล
ครบหรือไม่
ยัง
ครบ
ต้องการ
เปลี่ยนMode
ไม่
ต้องการ
MENU


Receive screen
รอรับ '?'
ตอบรับด้วย '.'
รับข้อมูลที่ส่งมา
แสดงที่จอภาพ
ต้องการ
เปลี่ยนMode
ไม่
ต้องการ
MENU

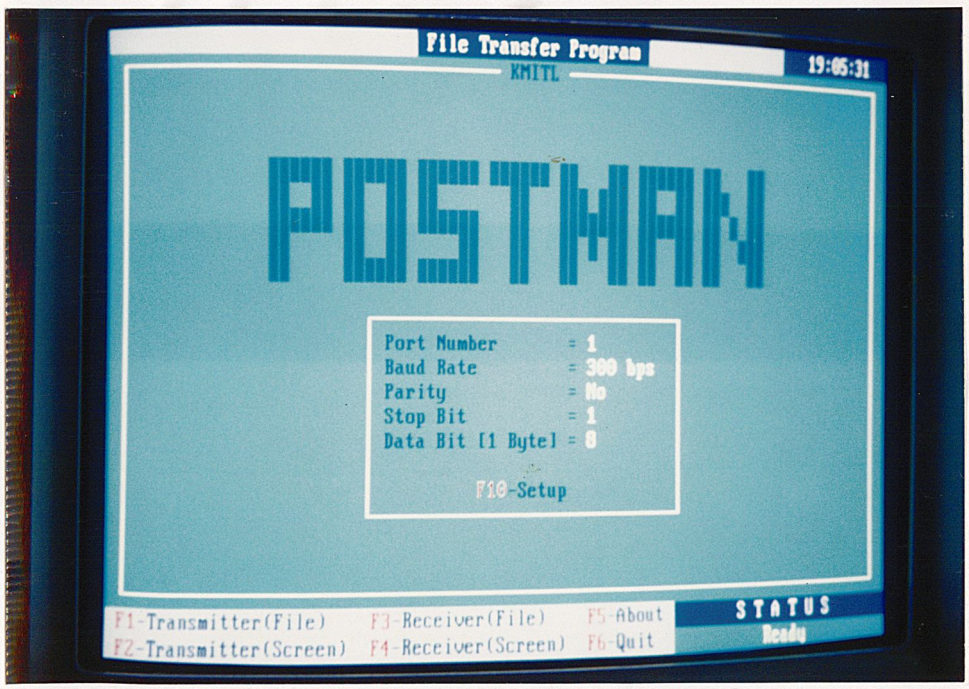
บทที่ 4

ผลการทดลอง

จากการทดลองในการรับส่งข้อมูล โปรแกรมที่ใช้ควบคุมการสื่อสารข้อมูลคือ postman ฟังก์ชันคีย์ F10 เป็นการ setup สร้างไฟล์ postman.iniโดยเป็นการ set environment ได้แก่ port number , baud rate , parity , stop bit และ data bit (1 byte)โดย port number สามารถเลือกได้จาก 0 ถึง 6 และเลือก modem setup code ซึ่งเป็นเลขdecimal จาก 2 ถึง 255 ในการทดลองเลือก port number 1 และ modem setup code , decimal number 67 , baud rate เป็น 300 bps , no parity bit , 1 stop bit และ data bit 1 byte แล้วทำการ save setup data ลงในไฟล์ postman.ini

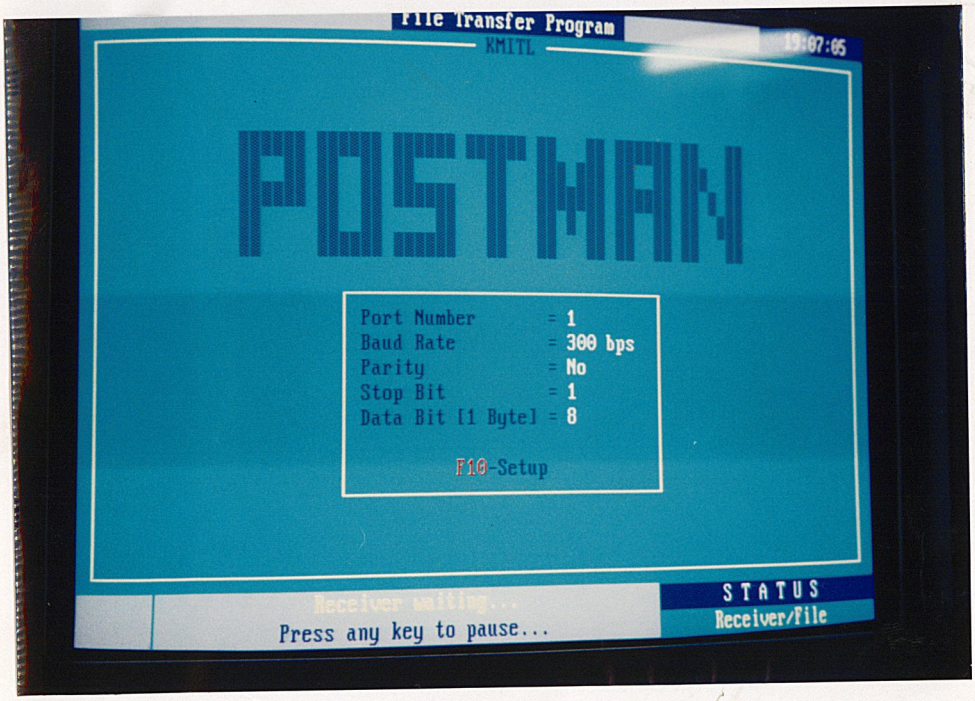
ฟังก์ชันคีย์ F1,F2,F3 และ F4 ใช้ในการรับส่งข้อมูล

ฟังก์ชันคีย์ F6 ใช้ในการออกจากโปรแกรม



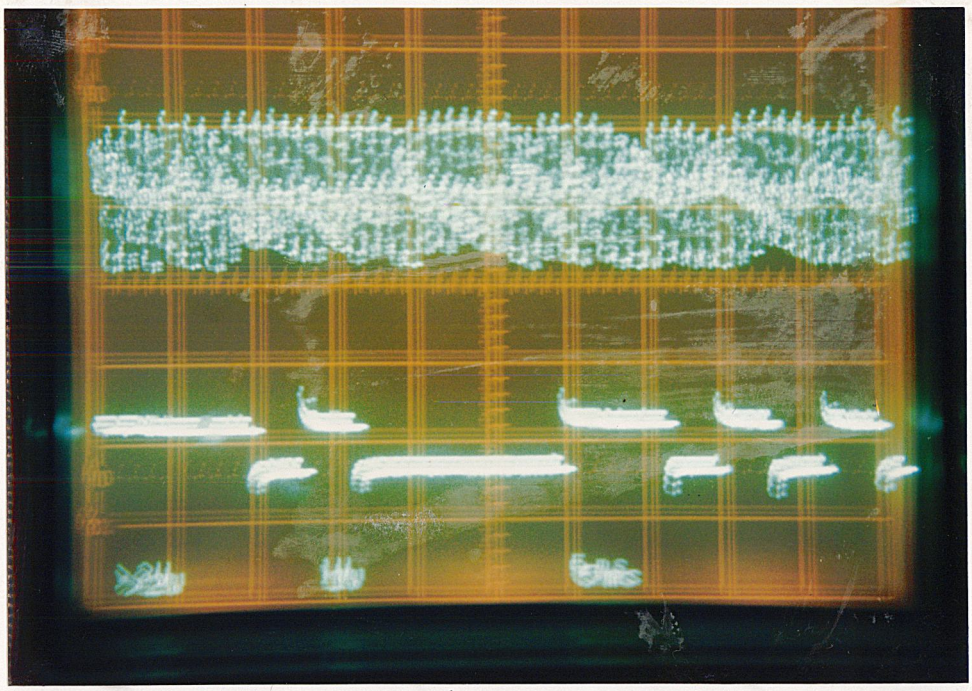
รูปที่ 4.1 โปรแกรมที่ใช้ในการรับส่งข้อมูล

รูปที่ 4.2 การส่งข้อมูลไฟล์ postman.ini



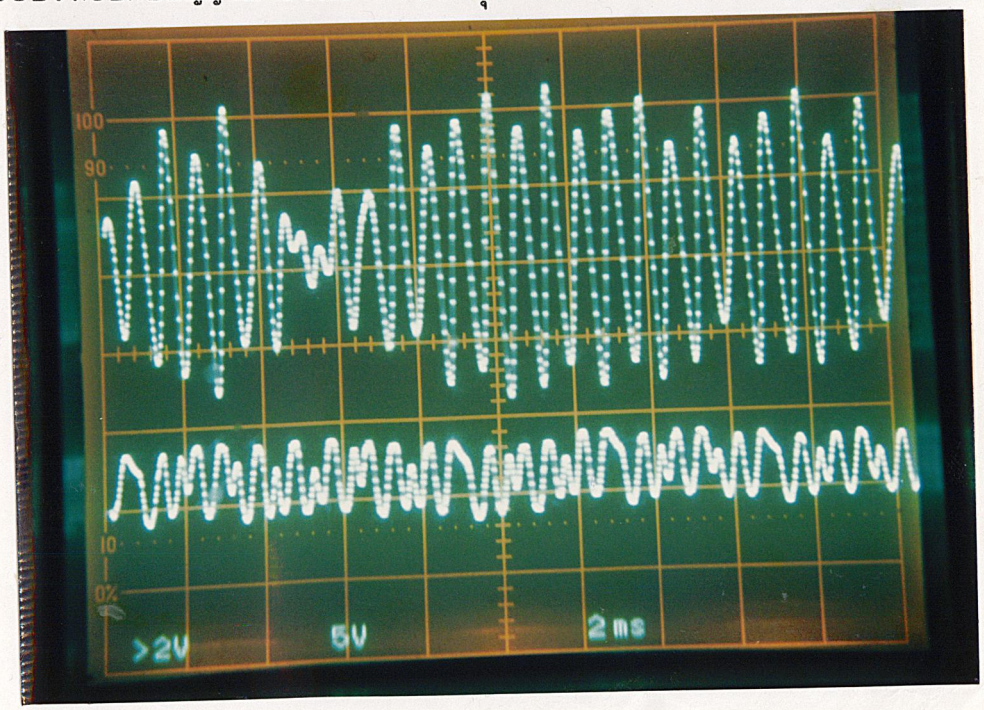
รูปที่ 4.3 แสดงการรับข้อมูล

เมื่อทำการส่งข้อมูลแล้วเราสามารถวัดสัญญาณ ตามจุดต่าง ๆได้ดังนี้ คือ



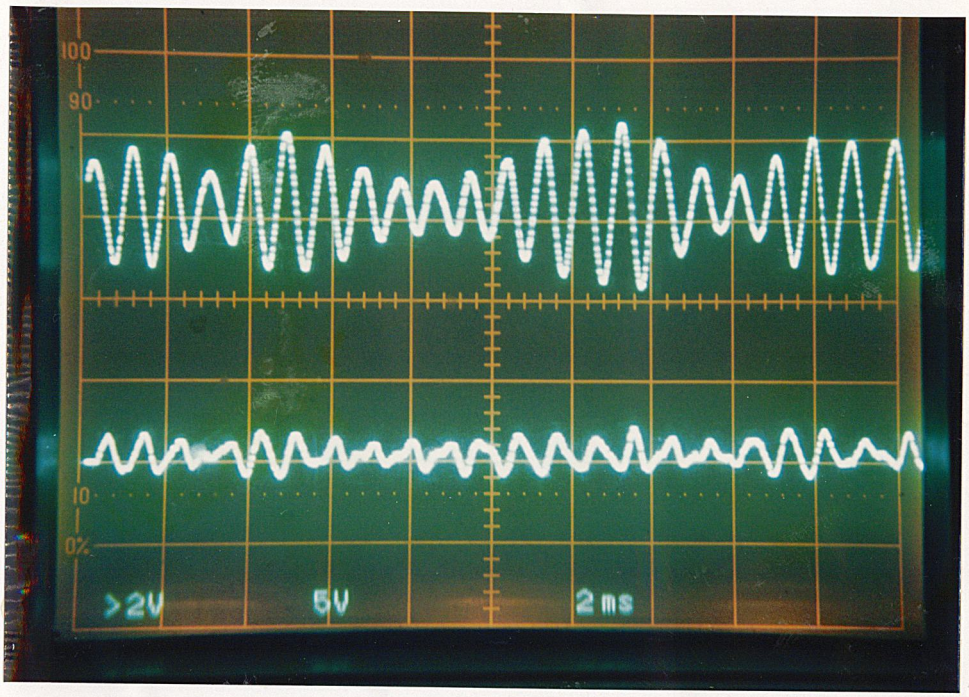
รูปที่ 4.4

จากรูปที่ 4.4 เป็นการวัดสัญญาณอินพุท square wave ที่ขา 9 ของ XR 2206 เปรียบเทียบกับสัญญาณ sine ที่เอาท์พุท หลังจากผ่านการ modulate แล้ว

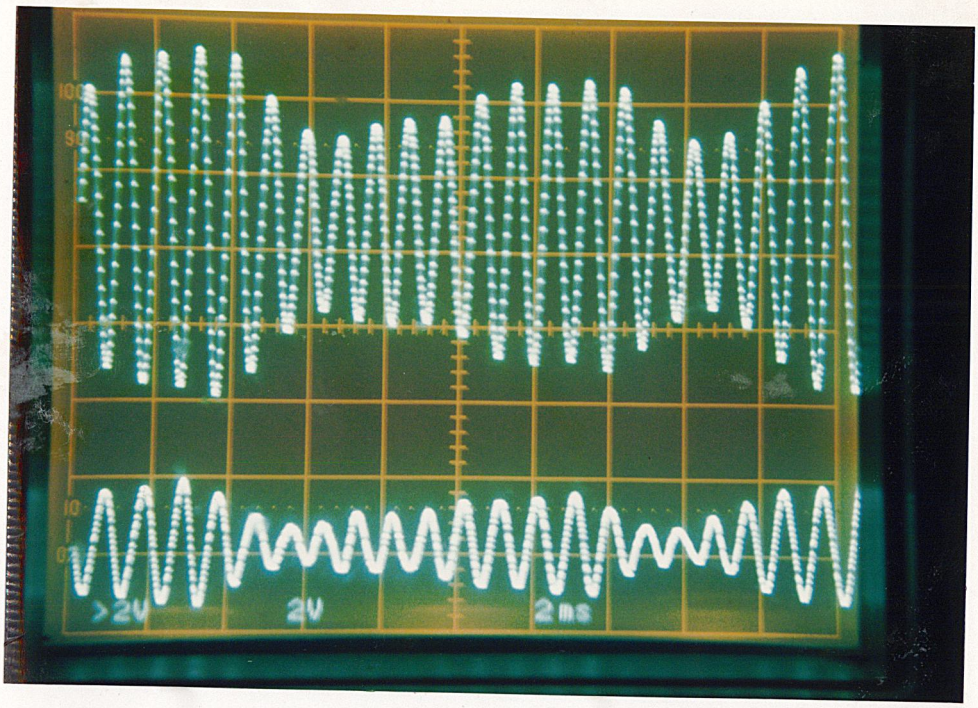


รูปที่ 4.5

รูปที่ 4.5 สัญญาณอินพุท (รูปล่าง) เมื่อผ่านวงจร band pass filter stage ที่ 1 จะได้สัญญาณไซน์ (ดังรูปคลื่นบน) โดยวัดที่ขา7 ของ XR-084 รูปที่ 4.6 และ รูปที่ 4.7 แสดงถึงสัญญาณไซน์ที่ band pass filter stage ที่ 2 และ ที่ 3 ตามลำดับ

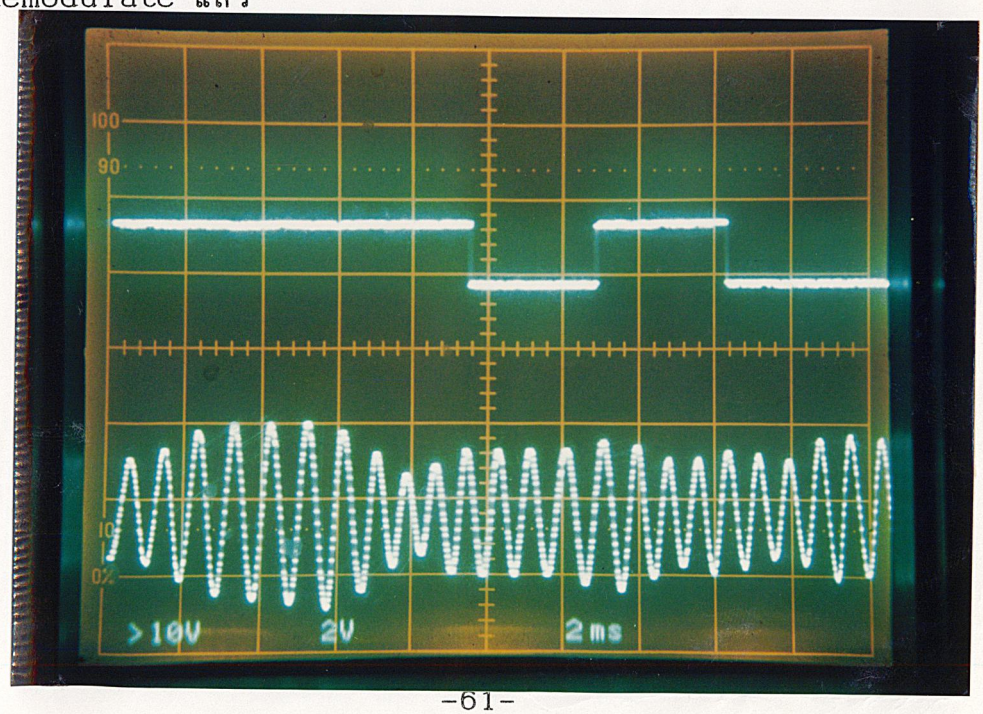


รูปที่ 4.6



รูปที่ 4.7

รูปที่ 4.8 แสดงการเปรียบเทียบสัญญาณไซน์ก่อน demodulate โดยวัดที่ขา 2 ของ XR-2211 และสัญญาณที่ขา 6 ของXR-2211 ซึ่งเป็นสัญญาณ square หลังจากถูก demodulate แล้ว



บทที่ 5

สรุปและวิจารณ์

ในการรับส่งข้อมูลของ WIRELESS MODEM นี้ ในส่วนของตัวโมเด็มเองไม่มีปัญหา แต่ในส่วนของวิทยุรับส่งซึ่งใช้ในการรับส่งข้อมูลมี 2 ความถี่ คือ 27 MHz และ 107 MHz โดยใช้เสาอากาศร่วมกัน ระยะทางที่ใช้ในการรับส่งข้อมูลนั้น โมเด็มที่ส่งข้อมูลความถี่ 27 MHz ทดลองส่งได้ในระยะทางประมาณ 5 เมตร ส่วนการส่งข้อมูลที่ความถี่ 107 MHz เนื่องจากเป็นความถี่ในย่าน FM จะมีปัญหาในการปรับแต่งเนื่องจากถูกคลื่นจากสถานีวิทยุรบกวน เพราะเครื่องส่งมีกำลังส่งต่ำ ดังนั้นระยะทางที่ทำการส่งข้อมูลจึงได้ไม่ไกลมาก คือ ไม่เกิน 1 ฟุต ถ้าระยะทางในการส่งมากกว่านี้ ตัวรับจะรับข้อมูลผิดพลาดหรือไม่สามารถทำการรับส่งข้อมูลได้เลย ส่วนขนาดของไฟล์ข้อมูลที่ใช้ในการส่งจะมีขนาดเท่าไรก็ได้ แต่เนื่องจากโมเด็มนี้อัตราการส่งข้อมูลเป็น 300 บ็อด ถ้าไฟล์ข้อมูลมีขนาดใหญ่มาก จะใช้เวลาในการรับส่งข้อมูลเป็นเวลานาน

ภาคผนวก

```
/***********************************************
 *  1) cls();                                  *
 *  2) draw_border();                          *
 *  3) get_key(); !                            *
 *  4) goto_xy();                              *
 *  5) paint_window();                         *
 *  6) print_title();                          *
 *  7) restore_video();                        *
 *  8) save_video();                           *
 *  9) video_mode(); !                         *
 * 10) write_char();                           *
 * 11) write_string();                         *
 *                                             *
 *  ! = Return value function, default (int)   *
 ***********************************************/

#include "conio.h"
#include "dos.h"
#include "stdio.h"
#include "string.h"

/*********************************************
 * D E C L A R A T I O N   F U N C T I O N S *
 *********************************************/

void cls(void);
void draw_border(int startx, int starty, int endx, int endy, int border_type,
                 int attrib);
int get_key(void);
void goto_xy(int x, int y);
void paint_window(int startx, int starty, int endx, int endy,
                  int ascii_code, int attrib);
void print_title(int startx, int starty, int endx, char *title,
                 int textb, int textc);
void read_cursor(void);
void restore_video(int startx, int starty, int endx, int endy);
void save_video(int startx, int starty, int endx, int endy);
int video_mode(void);
void write_char(int x, int y, char ch, int attrib);
void write_string(int x, int y, char *p, int attrib);

void exit(int status);     /* C function */
void *malloc(size_t size); /* C function */

/*********************************************
 * D E C L A R A T I O N   V A R I A B L E S *
 *********************************************/

int sing_border[] = {196, 179, 218, 192, 191, 217};
int doub_border[] = {205, 186, 201, 200, 187, 188};

int cursor_posit[2];            /* cursor_posit[0] = Column */
                                /* cursor_posit[1] = Row    */
unsigned char *p;

extern char far *vid_mem;

void cls(void)
{
  union REGS r;

  r.h.ah =  6;
  r.h.al =  0;
  r.h.ch =  0;
  r.h.cl =  0;
  r.h.dh = 24;
```

```
    r.h.dl = 79;
    r.h.bh =  7;
    int86(0x10, &r, &r);
}

void draw_border(int startx, int starty, int endx, int endy, int border_type,
                 int attrib)
{
    register int i;
             int *border_patt;
    char far *v, far *t;

    if (border_type == 1)
        border_patt = sing_border;
    else
        border_patt = doub_border;

    v = vid_mem;
    t = v;
    for (i = startx + 1; i < endx; i++) {
         v += (starty * 160) + (i * 2);
         *v++ = border_patt[0];
         *v = attrib;
         v = t;
         v += (endy * 160) + (i * 2);
         *v++ = border_patt[0];
         *v = attrib;
         v = t;
    }
    for (i = starty + 1; i < endy; i++) {
         v += (i * 160) + (startx * 2);
         *v++ = border_patt[1];
         *v = attrib;
         v = t;
         v += (i * 160) + (endx * 2);
         *v++ = border_patt[1];
         *v = attrib;
         v = t;
    }
    write_char(startx, starty, border_patt[2], attrib);
    write_char(startx, endy, border_patt[3], attrib);
    write_char(endx, starty, border_patt[4], attrib);
    write_char(endx, endy, border_patt[5], attrib);
}

int get_key(void)
{
    union REGS r;

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

void goto_xy(int x, int y)
{
    union REGS r;

    r.h.ah = 2;
    r.h.dl = x;
    r.h.dh = y;
    r.h.bh = 0;
    int86(0x10, &r, &r);
}

void paint_window(int startx, int starty, int endx, int endy,
                  int ascii_code, int attrib)
```

```
{
   register int i, j;

   for (i = starty; i <= endy; i++)
        for (j = startx; j <= endx; j++)
             write_char(j, i, ascii_code, attrib);
}

void print_title(int startx, int starty, int endx, char *title,
                 int textb, int textc)
{
   int middle, half;

   middle = startx + ((endx - startx + 1) / 2);
   half = strlen(title) / 2;
   write_char(middle - half - 1, starty, 219, textb);
   write_char(middle + half + 1, starty, 219, textb);
   textbackground(textb);
   textcolor(textc);
   goto_xy((middle - half), starty);
   cprintf("%s", title);
}

void restore_video(int startx, int starty, int endx, int endy)
{
   register int i, j;
   char far *v, far *t;
   char *buf_ptr;

   buf_ptr = p;
   v = vid_mem;
   t = v;
   for (j = startx; j <= (endx - 2); j++) {
        v = t;
        v += (starty * 160) + (j * 2);
        *v++ = *buf_ptr++;
        *v = *buf_ptr++;
   }
   for (i = (starty + 1); i <= (endy - 1); i++)
        for (j = startx; j <= endx; j++) {
             v = t;
             v += (i * 160) + (j * 2);
             *v++ = *buf_ptr++;
             *v = *buf_ptr++;
        }
   for (j = (startx + 2); j <= endx; j++) {
        v = t;
        v += (endy * 160) + (j * 2);
        *v++ = *buf_ptr++;
        *v = *buf_ptr++;
   }
   goto_xy(79, 24);
}

void save_video(int startx, int starty, int endx, int endy)
{
   register int i, j;
   char far *v, far *t;
   char *buf_ptr;

   if ((startx < 0) || (endx > 79) || (starty < 0) || (endy > 24)) {
      cprintf("Range Error, Window won't fit\n");
      cprintf("\007");
      exit(1);
   }
   p = (unsigned char *) malloc (2 * (endx - startx + 1) * (endy - starty + 1));
```

```
    if (!p) {
        cprintf("Out of memory\n");
        cprintf("\007");
        exit(1);
    }

    buf_ptr = p;
    v = vid_mem;
    for (j = startx; j <= (endx - 2); j++) {
        t = v + (starty * 160) + (j * 2);
        *buf_ptr++ = *t++;
        *buf_ptr++ = *t;
    }
    for (i = (starty + 1); i <= (endy - 1); i++)
        for (j = startx; j <= endx; j++) {
            t = v + (i * 160) + (j * 2);    /* Calculate address */
            *buf_ptr++ = *t++;              /* Read character    */
            *buf_ptr++ = *t;                /* Read attribute    */
        }
    for (j = (startx + 2); j <= endx; j++) {
        t = v + (endy * 160) + (j * 2);
        *buf_ptr++ = *t++;
        *buf_ptr++ = *t;
    }
}

int video_mode(void)
{
    union REGS r;

    r.h.ah = 15;
    return(int86(0x10, &r, &r) & 255);
}

void write_char(int x, int y, char ch, int attrib)
{
    char far *v;

    v = vid_mem;
    v += (y * 160) + (x * 2);       /* Calculate address */
    *v++ = ch;                      /* Write character   */
    *v = attrib;                    /* Write attribute   */
}

void write_string(int x, int y, char *p, int attrib)
{
    register int i;
    char far *v;

    v = vid_mem;
    v += (y * 160) + (x * 2);
    for (i = y; *p; i++) {
        *v++ = *p++;                /* Write character */
        *v++ = attrib;              /* Write attribute */
    }
}
```

```
 #include "conio.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

/**********************************
 * D E F I N E   C O N S T A N T S *
 **********************************/

#define BIT0_1   3
#define BIT2     4
#define BIT3_4  24
#define BIT5_7 224

#define SINGLE_BORDER 1
#define DOUBLE_BORDER 2

#define ATTRIB               48 /* Background = CYAN, Foreground = BLACK      */
#define CYAN_RED             52 /* Background = CYAN, Foreground = RED        */
#define CYAN_BLACK           48 /* Background = CYAN, Foreground = BLACK      */
#define CYAN_BLACK_BLINK    176 /* Background = CYAN, Foreground = BLACK      */
 /* And Blink                   */
#define CYAN_BLUE            49 /* Background = CYAN, Foreground = BLUE       */
#define CYAN_WHITE           63 /* Background = CYAN, Foreground = WHITE      */
#define CYAN_YELLOW          62 /* Background = CYAN, Foreground = YELLOW     */
#define LIGHTGRAY_BLACK     112 /* Background = LIGHTGRAY, Foreground = BLACK */
#define LIGHTGRAY_BLUE      113 /* Background = LIGHTGRAY, Foreground = BLUE  */
#define LIGHTGRAY_BLUE_BLINK 241 /* Background = LIGHTGRAY, Foreground = BLUE  */
 /* And Blink                   */
#define LIGHTGRAY_CYAN      115 /* Background = LIGHTGRAY, Foreground = CYAN  */
#define LIGHTGRAY_RED_BLINK 244 /* Background = LIGHTGRAY, Foreground = RED   */
 /* And Blink                   */
#define LIGHTGRAY_RED       116 /* Background = LIGHTGRAY, Foreground = RED   */
#define LIGHTGRAY_WHITE     127 /* Background = LIGHTGRAY, Foreground = WHITE */
#define LIGHTGRAY_YELLOW    126 /* Background = LIGHTGRAY, Foreground = YELLOW */

/*********************************************
 * D E C L A R A T I O N   V A R I A B L E S *
 *********************************************/

char setup_code[35] = "x-Port number, xxx-Modem setup code";
char port_number;
int port_;
char modem_setup[3] = "   ";
char *ptr;
int modem_code;
int data_code;
char ans;

int data_index[] = {
    2,                              /* Dummy value */
    2, 3,                           /* bit 0, 1    */
    0, 4,                           /* bit 2       */
    0, 16, 8, 24,                   /* bit 3, 4    */
    0, 32, 64, 96, 128, 160, 192, 224 /* bit 5, 6, 7 */
};

char *data_list[] = {
     "Error",
     "7", "8",                       /* bit 0, 1    */
     "1", "2",                       /* bit 2       */
     "No", "No", "Odd", "Even",      /* bit 3, 4    */
     "110 bps", "150 bps", "300 bps", "600 bps", "1200 bps", "2400 bps",
     "4800 bps", "9600 bps"          /* bit 5, 6, 7 */
};
```

```
/*******************************************
 *DECLARATION  FUNCTIONS*
 *******************************************/

/*************************************
 *INTERNAL  FUNCTIONS*
 *************************************/

void read_config(void);
void set_config(void);
void show_config(void);

/*************************************
 *EXTERNAL  FUNCTIONS*
 *************************************/

void cls(void);
void draw_border(int startx, int starty, int endx, int endy, int border_type,
                 int attrib);
void goodbye(void);
void goto_xy(int x, int y);
void print_title(int startx, int starty, int endx, char *title,
                 int textb, int textc);
void paint_window(int startx, int starty, int endx, int endy,
                  int ascii_code, int attrib);
void snd5(void);
void write_char(int x, int y, char ch, int attrib);
void write_string(int x, int y, char *p, int attrib);

int tolower(int ch); /* C function */

void read_config(void)
{
   FILE *fp;
   register int i;

   while (!(fp = fopen("postman.ini", "rt"))) {
         goodbye();
         printf("Cannot open postman.ini\n");
         printf("Please run setup.exe\n");
         printf("\007"); /* Beep */
         exit(1);
   }
   i = 0;
   do {
      setup_code[i] = getc(fp);
      if (ferror(fp)) {
   goodbye();
          printf("Error reading postman.ini\n");
          printf("Please run setup.exe\n");
          printf("\007"); /* Beep */
   fclose(fp);
          exit(1);
      }
      i++;
   } while(!feof(fp));
   fclose(fp);

   port_number = setup_code[0]; /* Read port number */
   port_ = port_number - 48;

   for (i = 0; i <= 2; i++) /* Read modem setup code */
       modem_setup[i] = setup_code[i + 15];
   modem_code = atoi(modem_setup);

   paint_window(24, 11, 55, 19, 219, CYAN); /* Clear window */
```

```
    draw_border(24, 11, 55, 19, SINGLE_BORDER, CYAN_YELLOW);
    write_string(26, 12, "Port Number       =", CYAN_BLUE);
    write_char(46, 12, port_number, CYAN_WHITE);

    data_code = modem_code & BIT5_7;
    for (i = 9; i <= 16; i++)
         if (data_code == data_index[i])
             break;
    if (i > 16)
        i = 0;
    write_string(26, 13, "Baud Rate         =", CYAN_BLUE);
    write_string(46, 13, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT3_4;
    for (i = 5; i <= 8; i++)
         if (data_code == data_index[i])
             break;
    if (i > 8)
        i = 0;
    write_string(26, 14, "Parity           =", CYAN_BLUE);
    write_string(46, 14, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT2;
    for (i = 3; i <= 4; i++)
         if (data_code == data_index[i])
             break;
    if (i > 4)
        i = 0;
    write_string(26, 15, "Stop Bit          =", CYAN_BLUE);
    write_string(46, 15, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT0_1;
    for (i = 1; i <= 2; i++)
         if (data_code == data_index[i])
             break;
    if (i > 2)
        i = 0;
    write_string(26, 16, "Data Bit [1 Byte] =", CYAN_BLUE);
    write_string(46, 16, data_list[i], CYAN_WHITE);

    write_string(35, 18, "F10", CYAN_RED);
    write_string(38, 18, "-Setup", CYAN_BLUE);
}

void show_config(void)
{
    register int i;

    paint_window(24, 11, 55, 19, 219, CYAN); /* Clear window */
    draw_border(24, 11, 55, 19, SINGLE_BORDER, CYAN_YELLOW);
    write_string(26, 12, "Port Number       =", CYAN_BLUE);
    write_char(46, 12, port_number, CYAN_WHITE);

    data_code = modem_code & BIT5_7;
    for (i = 9; i <= 16; i++)
         if (data_code == data_index[i])
             break;
    if (i > 16)
        i = 0;
    write_string(26, 13, "Baud Rate         =", CYAN_BLUE);
    write_string(46, 13, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT3_4;
    for (i = 5; i <= 8; i++)
         if (data_code == data_index[i])
```

```
            break;
    if (i > 8)
        i = 0;
    write_string(26, 14, "Parity           =", CYAN_BLUE);
    write_string(46, 14, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT2;
    for (i = 3; i <= 4; i++)
         if (data_code == data_index[i])
            break;
    if (i > 4)
        i = 0;
    write_string(26, 15, "Stop Bit         =", CYAN_BLUE);
    write_string(46, 15, data_list[i], CYAN_WHITE);

    data_code = modem_code & BIT0_1;
    for (i = 1; i <= 2; i++)
         if (data_code == data_index[i])
            break;
    if (i > 2)
        i = 0;
    write_string(26, 16, "Data Bit [1 Byte] =", CYAN_BLUE);
    write_string(46, 16, data_list[i], CYAN_WHITE);

    write_string(35, 18, "F10", CYAN_RED);
    write_string(38, 18, "-Setup", CYAN_BLUE);
}

void set_config(void)
{
    FILE *fp;
    register int i;

    for (;;) {
        print_title(24, 11, 55, "Set Environment", CYAN, BLACK);
        paint_window(46, 12, 54, 16, 219, CYAN); /* Clear data */
        paint_window(25, 18, 54, 18, 219, CYAN); /* Clear function key F10 */
        paint_window(2, 20, 77, 21, 219, CYAN); /* Clear message bar */
        textcolor(WHITE);
        do {
            write_string(26, 20, "Enter port number [0 - 6] :",
                         CYAN_BLACK);
     goto_xy(54, 20);
            port_number = getche();
        } while((port_number < '0') || (port_number > '6'));
        port_ = port_number - 48;

        do {
            write_string(10, 21, "Enter modem setup code, Decimal number only [2 - 255] :",
                         CYAN_BLACK);
            goto_xy(66, 21);
            modem_setup[0] = 4;
            ptr = cgets(modem_setup);
            modem_code = atoi(ptr);
        } while((modem_code < 2) || (modem_code > 255));

        write_char(46, 12, port_number, CYAN_WHITE);

        data_code = modem_code & BIT5_7;
        for (i = 9; i <= 16; i++)
     if (data_code == data_index[i])
                break;
        if (i > 16)
            i = 0;
write_string(46, 13, data_list[i], CYAN_WHITE);
```

```
        data_code = modem_code & BIT3_4;
        for (i = 5; i <= 8; i++)
             if (data_code == data_index[i])
                 break;
        if (i > 8)
            i = 0;
        write_string(46, 14, data_list[i], CYAN_WHITE);

        data_code = modem_code & BIT2;
        for (i = 3; i <= 4; i++)
             if (data_code == data_index[i])
                 break;
if (i > 4)
            i = 0;
        write_string(46, 15, data_list[i], CYAN_WHITE);

data_index[1] = 2;
        data_code = modem_code & BIT0_1;
        for (i = 1; i <= 2; i++)
             if (data_code == data_index[i])
                 break;
        if (i > 2)
            i = 0;
        write_string(46, 16, data_list[i], CYAN_WHITE);

        write_string(27, 18, "Is this all right (y/n)?", CYAN_BLACK);
        do {
            goto_xy(52, 18);
            ans = tolower(getche());
        } while((ans != 'y') && (ans != 'n'));
snd5(); /* Beep */
        if (ans == 'y')
            break;
    }
    paint_window(2, 20, 77, 21, 219, CYAN); /* Clear message bar */
    print_title(2, 20, 77, "Save setup data to postman.ini", CYAN, BLACK);
    print_title(2, 21, 77, "Please wait...", CYAN, BLACK);

    setup_code[0] = port_number; /* Put port number */
    for (i = 0; i <= 2; i++)     /* Put modem setup code */
         setup_code[i + 15] = *ptr++;

    remove("postman.ini");
    if (!(fp = fopen("postman.ini", "wt"))) {
        goodbye();
        printf("Cannot open postman.ini\n");
        printf("Please run setup.exe\n");
        printf("\007"); /* Beep */
        exit(1);
    }
    for (i = 0; i <= 34; i++) {
         putc(setup_code[i], fp);
if (ferror(fp)) {
     goodbye();
     printf("Error writing postman.ini\n");
     printf("Please run setup.exe\n");
     printf("\007"); /* Beep */
     exit(1);
        }
    }
    fclose(fp);
    paint_window(2, 21, 77, 21, 219, CYAN); /* Clear message bar */
    write_string(35, 21, "-Success-", CYAN_BLACK_BLINK);
    getch();
    snd5(); /* Beep */
```

```
    paint_window(2, 20, 77, 21, 219, CYAN); /* Clear message bar */
    draw_border(24, 11, 55, 19, SINGLE_BORDER, CYAN_YELLOW);
    paint_window(25, 18, 54, 18, 219, CYAN); /* Clear message bar */
    write_string(35, 18, "F10", CYAN_RED);
    write_string(38, 18, "-Setup", CYAN_BLUE);
}
```

```
#include "dos.h"
#include "stdio.h"

/*********************************************
 * D E C L A R A T I O N   F U N C T I O N S *
 *********************************************/

void snd1(void), snd2(void), snd3(void);
void snd4(void), snd5(void), snd6(void);
void sound_(int freq);

/*********************************************
 * D E C L A R A T I O N   V A R I A B L E S *
 *********************************************/

int rand(void);  /* C function */
int kbhit(void); /* C function */

void snd1(void)
{
   register int i;

   for (i = 10; i <= 60; i++) {
        sound(i * 50); delay(8);
   }
   nosound();
}

void snd2(void)
{
   register int i, j;

   for (i = 1; i <= 2; i++) {
        for (j = 2; j <= 30; j++) {
             sound(j * 100); delay(5);
        }
        for (j = 75; j >= 25; j--) {
             sound(j * 40); delay(5);
        }
   }
   nosound();
}

void snd3(void)
{
   register int i;

   for (i = 200; i <= 1200; i++) {
        sound(i * 3); delay(1);
   }
   for (i = 1200; i >= 100; i--) {
        sound(i * 3); delay(1);
   }
   nosound();
}

void snd4(void)
{
   register int i;

   for (i = 0; i <= 1500; i++) {
        sound(i * 10); delay(1);
        sound(i * 3); delay(1);
   }
   nosound();
}
```

```
void snd5(void)
{
   register int i;

   for (i = 10000; i >= 1000; i--)
        sound(i);
   nosound();
}

void snd6(void)
{
   int freq;

   do {
       do {
           freq = rand();
       } while(freq > 5000);
       sound_(freq);
   } while(!kbhit());
}

void sound_(int freq)
{
   unsigned i;
   union {
       long divisor;
       unsigned char c[2];
   } count;

   unsigned char p;

   count.divisor = 1193280 / freq;
   outportb(67, 182);
   outportb(66, count.c[0]);
   outportb(66, count.c[1]);

   p = inportb(97);
   outportb(97, p | 3);

   for (i = 0; i < 64000; i++);

   outportb(97, p);
}
```

```
#include "conio.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

#define BIT0_1   3
#define BIT2     4
#define BIT3_4  24
#define BIT5_7 224

char setup_code[35] = "x-Port number, xxx-Modem setup code";
char port_number;
int port_;
char modem_setup[3] = "   ";
char *ptr;
int modem_code;
int data_code;

int data_index[] = {
    0,                              /* Dummy value */
    2, 3,                           /* bit 0, 1    */
    0, 4,                           /* bit 2       */
    0, 16, 8, 24,                   /* bit 3, 4    */
    0, 32, 64, 96, 128, 160, 192, 224 /* bit 5, 6, 7 */
};

char *data_list[] = {
     "Error",
     "7", "8",                      /* bit 0, 1    */
     "1", "2",                      /* bit 2       */
     "No", "No", "Odd", "Even",     /* bit 3, 4    */
     "110 bps", "150 bps", "300 bps", "600 bps", "1200 bps", "2400 bps",
     "4800 bps", "9600 bps"         /* bit 5, 6, 7 */
};

void main(void)
{
   FILE *fp;
   register int i;

   clrscr();

   if (!(fp = fopen("postman.ini", "rt"))) {
       printf("Cannot open postman.ini\n");
       printf("Please run setup.exe\n");
       printf("\007");
       exit(1);
   }
   i = 0;
   do {
       setup_code[i] = getc(fp);
       if (ferror(fp)) {
           printf("Error reading postman.ini\n");
           printf("Please run setup.exe\n");
           printf("\007");
           exit(1);
       }
       i++;
   } while(!feof(fp));
   fclose(fp);

   port_number = setup_code[0]; /* Read port number */
   port_ = port_number - 48;

   for (i = 0; i <= 2; i++)
       modem_setup[i] = setup_code[i + 15];
```

```
    modem_code = atoi(modem_setup);

    clrscr();
    gotoxy(32, 1); printf("-Show Setup Data-");
    gotoxy(25, 3); printf("Port Number       = %d", port_);

    data_code = modem_code & BIT5_7;
    for (i = 9; i <= 16; i++)
         if (data_code == data_index[i])
             break;
    if (i > 16)
        i = 0;
    gotoxy(25, 4); printf("Baud Rate         = %s", data_list[i]);

    data_code = modem_code & BIT3_4;
    for (i = 5; i <= 8; i++)
         if (data_code == data_index[i])
             break;
    if (i > 8)
        i = 0;
    gotoxy(25, 5); printf("Parity            = %s", data_list[i]);

    data_code = modem_code & BIT2;
    for (i = 3; i <= 4; i++)
         if (data_code == data_index[i])
             break;
    if (i > 4)
        i = 0;
    gotoxy(25, 6); printf("Stop Bit          = %s", data_list[i]);

    data_code = modem_code & BIT0_1;
    for (i = 1; i <= 2; i++)
         if (data_code == data_index[i])
             break;
    if (i > 2)
        i = 0;
    gotoxy(25, 7); printf("Data Bit [1 Byte] = %s", data_list[i]);

    gotoxy(17, 10);
    printf("-If setup data incorrect, Please run setup.exe-");
}
```

```
/*********************************************************************
 * POSTMAN.C is file transfer program. it use software handshaking. *
 *              -> boudrate = 9,600 bps                              *
 *              -> no parity bit                                     *
 *              -> data size = 8 bit = 1 byte                        *
 *              -> 2 stop bit                                        *
 *********************************************************************/

#include "conio.h"
#include "dos.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

/***********************************
 * D E F I N E   C O N S T A N T S *
 ***********************************/

#define STEP             24
#define SINGLE_BORDER     1
#define DOUBLE_BORDER     2
#define NULL_CHAR        32 /* NULL_CHAR = 32 = 0x20 = ' ' */

/************************
 * C O L O R   C O D E *
 ************************/

#define BLUE_RED_BLINK      148 /* Background = BLUE, Foreground = RED        */
                                /* And Blink                                  */
#define CYAN_RED             52 /* Background = CYAN, Foreground = RED        */
#define CYAN_YELLOW          62 /* Background = CYAN, Foreground = YELLOW     */
#define LIGHTGRAY_BLACK     112 /* Background = LIGHTGRAY, Foreground = BLACK */
#define LIGHTGRAY_BLUE      113 /* Background = LIGHTGRAY, Foreground = BLUE  */
#define LIGHTGRAY_CYAN      115 /* Background = LIGHTGRAY, Foreground = CYAN  */
#define LIGHTGRAY_RED       116 /* Background = LIGHTGRAY, Foreground = RED   */
#define LIGHTGRAY_RED_BLINK 244 /* Background = LIGHTGRAY, Foreground = RED   */
                                /* And Blink                                  */
#define LIGHTGRAY_YELLOW    126 /* Background = LIGHTGRAY, Foreground = YELLOW */

/**********************
 * S C A N   C O D E *
 **********************/

#define F1  59
#define F2  60
#define F3  61
#define F4  62#define F5  63
#define F6  64
#define F10 68
#define ESC 27

/*********************************************
 * D E C L A R A T I O N   F U N C T I O N S *
 *********************************************/

void about(void);
void cls(void); /* External function */
void draw_border(int startx, int starty, int endx, int endy, int border_type,
                 int attrib); /* External function */
void edit(void); /* External function */
void get_filename(void);
int get_key(void); /* External function */
void goto_xy(int x, int y); /* External function */
void initial(void);
void monitors(void);
void paint_window(int startx, int starty, int endx, int endy,
```

```
                int ascii_code, int attrib); /* External function */
void print_logo(void); /* External function */
void print_title(int startx, int starty, int endx, char *title,
                int textb, int textc); /* External function */
void quit(void);
void RecFile(void);
void rec_file(void); /* External function */
void rec(void); /* External function */
void read_config(void); /* External function */
void restore_menu(void);
void restore_screen(void);
void restore_video(int startx, int starty, int endx, int endy);
    /* External function */
void rotate_message(void); /* External function */
void snd4(void), snd5(void), snd6(void); /* External function */
void save_video(int startx, int starty, int endx, int endy);
    /* External function */
void SendFile(void);
void send_file(char *fname); /* External function */
void set_config(void); /* External function */
void show_config(void); /* External function */
void show_status(char *status);
int video_mode(void); /* External function */
void windwheel(int col); /* External function */
void write_char(int x, int y, char ch, int attrib);   /* External function */
void write_string(int x, int y, char *p, int attrib); /* External function */

int bioskey(int cmd); /* C function */
int tolower(int ch);  /* C function */
char *strset(char *s, int ch); /* C function */

extern void SETNEW(void); /* External function, TIME.ASM */
extern void SETOLD(void); /* External function, TIME.ASM */

/*********************************************
 * D E C L A R A T I O N   V A R I A B L E S *
 *********************************************/

char *helpline_row23 =
" F1-Transmitter(File)    F3-Receiver(File)    F6-About ";

char *helpline_row24 =
" F2-Transmitter(Screen)  F4-Receiver(Screen)  F7-Quit  ";

char *func_keys[] = {
    "F1", "F3", "F5",
    "F2", "F4", "F6"
};

char *ready =
"Ready";

char *trans_f =
"Transmitter/File";

char *trans_s =
"Transmitter/Screen";

char *recei_f =
"Receiver/File";

char *recei_s =
"Receiver/Screen";

char *about_ =
"About";
```

```
char *quit_ =
"Exit to DOS";

char *setup_ =
"Set Environment";

union inkey {
     char ch[2];
     int  i;
} key;

char far *vid_mem;

unsigned char HOUR, MIN, SEC;

extern unsigned int prev_row;

/***************************
 *M A I N   P R O G R A M*
 ***************************/

main(void)
{
   int vmode;
   char ans;

   vmode = video_mode();
   if ((vmode != 2) && (vmode != 3) && (vmode != 7)) {
      cprintf("Video must be in 80 column text mode\n");
      cprintf("\007");
      exit(1);
   }
   /* Set start address of VIDEO RAM */

   if (vmode == 7) vid_mem = (char far *) 0xB0000000;
   else vid_mem = (char far *) 0xB8000000;

   initial();
   read_config();

   for (;;) {
        show_status(ready);
        key.i = bioskey(0);
        if (!key.ch[0]) {              /* If keypressed is not character */
           switch(key.ch[1]) {
                 case F1  : show_status(trans_f); /* F1 were pressed     */
                            snd5();               /* Transmitter(File)   */
                            SendFile();
                            break;

                 case F2  : show_status(trans_s); /* F2 were pressed     */
                            snd5();               /* Transmitter(Screen) */
                            edit();               /* EDITOR.C            */
                            break;

                 case F3  : show_status(recei_f); /* F3 were pressed     */
                            snd5();               /* Receiver(File)      */
                            RecFile();
                            break;

                 case F4  : show_status(recei_s); /* F4 were pressed     */
                            snd5();               /* Receiver(Screen)    */
                            monitors();
                            break;
```

```
                case F5  : show_status(about_);  /* F5 were pressed     */
                           about();              /* About               */
                           break;

                case F6  : show_status(quit_);   /* F6 were pressed     */
                           quit();               /* Quit                */
                           do {
                              ans = tolower(getch());
                           } while((ans != 'y') && (ans != 'n'));
                           snd5(); /* Beep */
                           if (ans == 'y') {
                              SETOLD();
                              cls();
                              snd4();
                              exit(0);
                           }
                           /* Do not exit */
                           /* Restore Screen */
                           restore_video(29, 9, 51, 13);
                           break;

                case F10 : show_status(setup_);  /* F10 were pressed    */
                           snd5();               /* Setup Config        */
                           set_config();
                           break;
            }
        }
    }
}

void initial(void)
{
    register int r, p;
             int col, row, index;

    cls();
    paint_window(0, 0, 79, 24, 219, CYAN);
    paint_window(0, 0, 79, 0, 219, LIGHTGRAY);
    paint_window(67, 0, 79, 0, 219, BLUE);
    paint_window(0, 23, 54, 24, 219, LIGHTGRAY);
    paint_window(55, 23, 79, 23, 219, BLUE);
    print_title(55, 23, 79, "S T A T U S", BLUE, WHITE);
    draw_border(1, 1, 78, 22, SINGLE_BORDER, CYAN_YELLOW);
    print_title(0, 0, 79, "File Transfer Program", BLUE, YELLOW);
    print_title(1, 1, 78, "KMITL", CYAN, BLUE);

    write_string(0, 23, helpline_row23, LIGHTGRAY_BLUE);
    write_string(0, 24, helpline_row24, LIGHTGRAY_BLUE);

    col = 1; row = 23;
    index = 0;
    for (r = 1; r <= 2; r++) {
        for (p = 1; p <= 3; p++, col = col + STEP, index++) {
            if (p == 3)
               col = col - 3;
            write_string(col, row, func_keys[index], LIGHTGRAY_RED);
        }
        col = 1;
        row++;
    }
    print_logo();
    SETNEW();
}

void show_status(char *status)
{
```

```
    int col, len;

    paint_window(55, 24, 79, 24, 219, CYAN);      /* Clear status */

    len = strlen(status);
    col = 67 - (len / 2);
    write_string(col, 24, status, CYAN_YELLOW);
}

void about(void)
{
    save_video(22, 8, 56, 15);

    paint_window(24, 9, 56, 15, 219, BLACK);      /* Shadow */
    paint_window(22, 8, 54, 14, 219, LIGHTGRAY);
    draw_border(22, 8, 54, 14, SINGLE_BORDER, LIGHTGRAY_YELLOW);
    print_title(22, 8, 54, "About", LIGHTGRAY, RED);

    write_string(35, 9, "-KMITL-", LIGHTGRAY_BLUE);
    write_string(35, 10, "POSTMAN", LIGHTGRAY_BLUE);
    write_string(33, 11, "Version 1.1", LIGHTGRAY_BLUE);
/* write_string(24, 12, "Programmer : Charlie Sookthet", LIGHTGRAY_BLUE); */
    write_string(35, 13, "  OK  ", BLUE_RED_BLINK);
    snd5();
/* snd6(); */
    rotate_message();
/* getch(); */

    restore_video(22, 8, 56, 15);
}

void get_filename(void)
{
    save_video(21, 14, 57, 17);

    paint_window(23, 15, 57, 17, 219, BLACK);      /* Shadow */
    paint_window(21, 14, 55, 16, 219, LIGHTGRAY);
    draw_border(21, 14, 55, 16, SINGLE_BORDER, LIGHTGRAY_YELLOW);
    print_title(21, 14, 55, "Enter Flie Name", LIGHTGRAY, BLUE);

    window(23, 16, 55, 16);
}

void quit(void)
{
    save_video(29, 9, 51, 13);

    paint_window(31, 10, 51, 13, 219, BLACK);     /* Shadow */
    paint_window(29, 9, 49, 12, 219, LIGHTGRAY);
    draw_border(29, 9, 49, 12, SINGLE_BORDER, LIGHTGRAY_YELLOW);
    print_title(29, 9, 49, "Exit POSTMAN?", LIGHTGRAY, RED);

    write_char(31, 10, 'Y', LIGHTGRAY_RED);
    write_char(31, 11, 'N', LIGHTGRAY_RED);

    write_string(32, 10, "es (exit to DOS)", LIGHTGRAY_BLUE);
    write_string(32, 11, "o  (do not exit)", LIGHTGRAY_BLUE);
    snd5();
}

void restore_screen(void)
{
    register int r, p;
             int col, row, index;

  paint_window(0, 1, 79, 22, 219, CYAN);
```

```
    draw_border(1, 1, 78, 22, SINGLE_BORDER, CYAN_YELLOW);
    print_title(1, 1, 78, "KMITL", CYAN, BLUE);
    print_logo();
    paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */

    write_string(0, 23, helpline_row23, LIGHTGRAY_BLUE);
    write_string(0, 24, helpline_row24, LIGHTGRAY_BLUE);

    col = 1; row = 23;
    index = 0;
    for (r = 1; r <= 2; r++) {
        for (p = 1; p <= 3; p++, col = col + STEP, index++) {
            if (p == 3)
               col = col - 3;
            write_string(col, row, func_keys[index], LIGHTGRAY_RED);
        }
        col = 1;
        row++;
    }
    window(1, 1, 80, 25);     /* Set window to the normal size */
    show_config();
}

void SendFile(void)
{
    char fname[15]; /* fname[0] = Maximum length of the string */
    char *p;        /* fname[1] = The number of characters actually read */
                    /* fname[2] = The characters read start */
    char ans;

    for (;;) {
        strset(fname, NULL_CHAR); /* Clear array */

        paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */
        write_char(3, 23, 179, LIGHTGRAY_CYAN);
        write_char(3, 24, 179, LIGHTGRAY_CYAN);
        get_filename();

        fname[0] = 13; /* 12 Characters */
        /* Set fname[0] to the maximum length of the string to be read. */
        p = cgets(fname);
        restore_video(21, 14, 57, 17);
        window(1, 1, 80, 25); /* Set window to the old size */

        send_file(p); /* Call send_file() function */

        paint_window(4, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
        print_title(4, 23, 54, "Do you want to continue?",
                    LIGHTGRAY, YELLOW);
        print_title(4, 24, 54, " es (do not exit),  o (exit to main menu)",
                    LIGHTGRAY, BLUE);
        write_char(9, 24, 'Y', LIGHTGRAY_RED);
        write_char(28, 24, 'N', LIGHTGRAY_RED);
        do {
           ans = tolower(getch());
        } while((ans != 'y') && (ans != 'n'));
        snd5(); /* Beep */
        if (ans == 'n') {
           restore_menu();
           break; /* Return to main menu */
        }
    }
}

void RecFile(void)
{
```

```
    paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */
    write_char(7, 23, 179, LIGHTGRAY_CYAN);
    write_char(7, 24, 179, LIGHTGRAY_CYAN);
    window(1, 1, 80, 25); /* Set window to the normal size */

    do {
        paint_window(0, 23, 6, 24, 219, LIGHTGRAY);  /* Clear "PAUSE" */
        paint_window(8, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
        print_title(8, 24, 54, "Press any key to pause...",
                    LIGHTGRAY, BLUE);
        rec_file(); /* Call rec_file() function */

        write_string(1, 24, "PAUSE", LIGHTGRAY_RED_BLINK);
        paint_window(8, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
        print_title(8, 23, 54, "Press any key to continue",
                    LIGHTGRAY, BLUE);
        write_string(28, 24, "F6", LIGHTGRAY_RED);
        write_string(30, 24, "-Quit", LIGHTGRAY_BLUE);

        getch(); /* Wait for key */
        key.i = get_key(); /* Wait for key */
        snd5(); /* Beep */
    } while(!(key.ch[0] == 0 && key.ch[1] == F6));
      /* Loop until were pressed F6 */
    restore_menu(); /* Return to main menu */
}

void monitors(void)
{
    union inkey {
          char ch[2];
          int  i;
    } key;

    window(1, 1, 80, 25); /* Set window to the normal size */

    paint_window(0, 1, 79, 22, 219, CYAN);
    paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */
    write_char(14, 23, 179, LIGHTGRAY_CYAN);
    write_char(14, 24, 179, LIGHTGRAY_CYAN);

    do {
        paint_window(0, 23, 13, 24, 219, LIGHTGRAY);
        paint_window(15, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
        print_title(15, 24, 54, "Press any key to pause...",
                    LIGHTGRAY, BLUE);
        rec(); /* Call rec(); function */

        write_string(5, 24, "PAUSE", LIGHTGRAY_RED_BLINK);
        paint_window(15, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
        print_title(15, 23, 54, "Press any key to continue",
                    LIGHTGRAY, BLUE);
        write_string(32, 24, "F6", LIGHTGRAY_RED);
        write_string(34, 24, "-Quit", LIGHTGRAY_BLUE);

        getch(); /* Wait for key */
        key.i = get_key(); /* Wait for key */
        snd5(); /* Beep */
    } while(!(key.ch[0] == 0 && key.ch[1] == F6));
      /* Loop until were pressed F6 */
    prev_row = 1;
    restore_screen(); /* Return to main menu */
}

void restore_menu(void)
{
```

```
    register int r, p;
            int col, row, index;

    paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */

    write_string(0, 23, helpline_row23, LIGHTGRAY_BLUE);
    write_string(0, 24, helpline_row24, LIGHTGRAY_BLUE);

    col = 1; row = 23;
    index = 0;
    for (r = 1; r <= 2; r++) {
        for (p = 1; p <= 3; p++, col = col + STEP, index++) {
            if (p == 3)
                col = col - 3;
            write_string(col, row, func_keys[index], LIGHTGRAY_RED);
        }
        col = 1;
        row++;
    }
    window(1, 1, 80, 25); /* Set window to the normal size */
}
```

```
/*****************************************************************
 * TRANS.C is file transfer program. it use software handshaking. *
 *             -> boudrate = 9,600 bps                            *
 *             -> no parity bit                                   *
 *             -> data size = 8 bit = 1 byte                      *
 *             -> 2 stop bit                                      *
 *             -> initial code = 0x00E7                           *
 *                                                                *
 *                        (Functions)                             *
 *                                                                *
 *             - void send_file(char *fname);                     *
 *             - void rec_file(void);                             *
 *             - void send(char *buff, char y);                   *
 *             - void rec(void);                                  *
 *             - unsigned int filesize(FILE *fp);                 *
 *             - void send_file_name(char *f);                    *
 *             - void get_file_name(char *f);                     *
 *             - void wait(int port);                             *
 *             - void wait_s(int port);                           *
 *             - void sport(int port, char c);                    *
 *             - rport(int port);                                 *
 *             - check_stat(int port);                            *
 *             - void port_init(int port, unsigned char code);    *
 *             - void appear(char count, unsigned int col,        *
 *                           unsigned int row);                   *
 *             - void init_array(void);                           *
 *             - void goodbye(void);                              *
 *****************************************************************/

#include "alloc.h"
#include "bios.h"
#include "conio.h"
#include "ctype.h"
#include "dos.h"
#include "stdio.h"
#include "string.h"

/*********************************
 * D E F I N E   C O N S T A N T S *
 *********************************/
/*
#define PORT          1
#define INIT_CODE 0x00E7
*/
#define BUF_SIZE  79
#define LINE_LEN  79
#define MAX_LINES 22
#define NULL_CHAR 32 /* NULL_CHAR = 32 = 0x20 = ' ' */
#define F6        64

#define ATTRIB               48 /* Background = CYAN, Foreground = BLACK      */
#define CYAN_RED             52 /* Background = CYAN, Foreground = RED        */
#define CYAN_BLUE            49 /* Background = CYAN, Foreground = BLUE       */
#define LIGHTGRAY_BLACK     112 /* Background = LIGHTGRAY, Foreground = BLACK */
#define LIGHTGRAY_BLUE      113 /* Background = LIGHTGRAY, Foreground = BLUE  */
#define LIGHTGRAY_BLUE_BLINK 241 /* Background = LIGHTGRAY, Foreground = BLUE  */
                                 /* And Blink                                  */
#define LIGHTGRAY_CYAN      115 /* Background = LIGHTGRAY, Foreground = CYAN  */
#define LIGHTGRAY_RED_BLINK  244 /* Background = LIGHTGRAY, Foreground = RED   */
                                 /* And Blink                                  */
#define LIGHTGRAY_RED       116 /* Background = LIGHTGRAY, Foreground = RED   */
#define LIGHTGRAY_WHITE     127 /* Background = LIGHTGRAY, Foreground = WHITE */
#define LIGHTGRAY_YELLOW    126 /* Background = LIGHTGRAY, Foreground = YELLOW */

#define DELAY_TIME 2000
```

```
/*********************************************
 *D E C L A R A T I O N   F U N C T I O N S*
 *********************************************/

/*************************************
 *I N T E R N A L   F U N C T I O N S*
 *************************************/

void send_file(char *fname);
void rec_file(void);
void send(char *buff, char y);
void rec(void);
unsigned int filesize(FILE *fp);
void send_file_name(char *f);
void get_file_name(char *f);
void wait(int port);
void wait_s(int port);
void sport(int port, char c);
int rport(int port);
int check_stat(int port);
void port_init(int port, unsigned char code);
void appear(char count, unsigned int col, unsigned int row);
void init_array(void);
void goodbye(void);

/*************************************
 *E X T E R N A L   F U N C T I O N S*
 *************************************/

void paint_window(int startx, int starty, int endx, int endy,
                  int ascii_code, int attrib); /* External function */
void print_title(int startx, int starty, int endx, char *title,
                 int textb, int textc); /* External function */
void write_string(int x, int y, char *p, int attrib); /* External function */
void write_char(int x, int y, char ch, int attrib); /* External function */
void restore_screen(void); /* External function */
int get_key(void); /* External function */
void cls(void);    /* External function */
void SETOLD(void); /* External function */
void snd5(void); /* External function */
void windwheel(int col); /* External function */

void exit(int status); /* C function */
int tolower(int ch);   /* C function */
int kbhit(void);       /* C function */
int getch(void);       /* C function */

/*********************************************
 *D E C L A R A T I O N   V A R I A B L E S*
 *********************************************/

char buffers[BUF_SIZE]; /* User : rec();, appear(); */

int port, init_code;

unsigned int prev_row = 1; /* User : appear(); */

int w_flag = 0; /* Set initial value */

extern char far *vid_mem;

extern int port_, modem_code;

void send_file(char *fname)
{
   FILE *fp;
```

```
    char ch;
    union {
        char c[2];
        unsigned int count;
    } cnt;

    port = port_;
    init_code = modem_code;
    port_init(port, init_code); /* Initialization */

    if (!(fp = fopen(fname, "rb"))) {
        printf("\007"); /* Beep */
        paint_window(4, 23, 54, 24, 219, LIGHTGRAY);
        /* Clear message bar, 23 - 24 */
        print_title(4, 23, 54, "Cannot open input file",
                    LIGHTGRAY, RED);
        print_title(4, 24, 54, "Please wait...", LIGHTGRAY, BLUE);
        delay(DELAY_TIME);
        goto Quit_snd_fle; /* Return to SendFile(); function */
    }

    send_file_name(fname);
    if (kbhit()) /* User interrupt */
        goto Quit_snd_fle; /* Return to SendFile(); function */

        wait(port);
    /* Ready */
    cnt.count = filesize(fp); /* Calculate file size */
    sport(port, cnt.c[0]); /* Send file size, Low byte */
        wait(port);
    sport(port, cnt.c[1]); /* Send file size, High byte */

    do {
        ch = getc(fp);

        if (ferror(fp)) {
            printf("\007"); /* Beep */
            paint_window(4, 23, 54, 24, 219, LIGHTGRAY);
            /* Clear message bar, 23 - 24 */
            print_title(4, 23, 54, "Error reading input file",
                        LIGHTGRAY, RED);
            print_title(4, 24, 54, "Please wait...", LIGHTGRAY, BLUE);
            delay(DELAY_TIME);
            break;
        }

        if (!feof(fp)) {
            wait(port);
/*          windwheel(1); */
            sport(port, ch);
        }
    } while(!feof(fp));
    wait(port);
    fclose(fp); /* Close file */
    paint_window(4, 24, 54, 24, 219, LIGHTGRAY); /* Clear message bar, 24 */
    write_string(25, 24, "-Success-", LIGHTGRAY_BLUE_BLINK);
    getch();
Quit_snd_fle:; /* Quit_snd_fle Label */
}

void rec_file(void)
{
    FILE *fp;
    char ch;
    char fname[14];
    union {
```

```
        char c[2];
        unsigned int count;
    } cnt;

    port = port_;
    init_code = modem_code;
    port_init(port, init_code); /* Initialization */

    get_file_name(fname);
    if (kbhit()) /* User interrupt */
        goto Quit_rec_fle; /* Return to RecFile(); function */

    paint_window(8, 23, 54, 24, 219, LIGHTGRAY);
    /* Clear message bar, 23 - 24 */
    write_string(18, 23, "Receiving file", LIGHTGRAY_YELLOW);
    write_string(33, 23, fname, LIGHTGRAY_BLACK);
    print_title(8, 24, 54, "Please wait...", LIGHTGRAY, BLUE);

    remove(fname);
    if (!(fp = fopen(fname, "wb"))) {
        printf("\007"); /* Beep */
        paint_window(8, 23, 54, 24, 219, LIGHTGRAY);
        /* Clear message bar, 23 - 24 */
        print_title(8, 23, 54, "Cannot open output file",
                    LIGHTGRAY, RED);
        print_title(8, 24, 54, "Please wait...", LIGHTGRAY, BLUE);
        delay(DELAY_TIME);
        goto Quit_rec_fle; /* Return to RecFile(); function */
    }

        sport(port, '.'); /* get_file_name() function */
    cnt.c[0] = rport(port);
        sport(port, '.');
    cnt.c[1] = rport(port);
        sport(port, '.');

    for (; cnt.count; cnt.count--) {
         ch = rport(port);
         putc(ch, fp);
         if (ferror(fp)) {
             printf("\007"); /* Beep */
             paint_window(8, 23, 54, 24, 219, LIGHTGRAY);
             /* Clear message bar, 23 - 24 */
             print_title(8, 23, 54, "Error writing file",
                         LIGHTGRAY, RED);
             print_title(8, 24, 54, "Please wait...", LIGHTGRAY, BLUE);
             delay(DELAY_TIME);
             break; /* Return to RecFile(); function */
         }
         sport(port, '.');
/*       windwheel(3); */
    }
    fclose(fp); /* Close file */
    paint_window(8, 24, 54, 24, 219, LIGHTGRAY); /* Clear message bar, 24 */
    write_string(25, 24, "-Success-", LIGHTGRAY_BLUE_BLINK);
    getch(); /* Wait */
Quit_rec_fle:; /* Quit_rec_fle Label */
}

void send(char *buff, char y)
{
    register int i;
    union REGS r;
    char data_size, line;

   port = port_;
```

```
    init_code = modem_code;
    port_init(port, init_code); /* Initialization */

    window(1, 1, 80, 25); /* Set window to the normal size */

    paint_window(15, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
    print_title(15, 23, 54, "Transmitter waiting...", LIGHTGRAY, YELLOW);

    wait_s(port);
    if (w_flag) { /* User interrupt */
        w_flag = 0;
        goto Quit_snd; /* Return to enter(); function, EDITOR.C */
    }
    /* Ready */
    paint_window(15, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
    print_title(15, 23, 54, "Sending data...", LIGHTGRAY, YELLOW);

    data_size = strlen(buff); /* Calculate data size */
    sport(port, data_size); /* Send data size */
        wait(port);

    line = y; /* Put position of row to line variable */
    sport(port, line); /* Send position of row */
        wait(port);

    for (i = 0; buff[i] != '\0'; i++) {
         sport(port, buff[i]);
         wait(port);
/*       windwheel(16); */
    }
    paint_window(15, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
Quit_snd:; /* Quit_snd Label */
    window(2, 2, 80, 23); /* Set window for editing */
}

void rec(void)
{
    register int i;
    unsigned int col;
    char data_size, counter, line;

    port = port_;
    init_code = modem_code;
    port_init(port, init_code); /* Initialization */

    for (;;) {
         init_array(); /* Clear array */
         col = line = 1; /* Set initial value */
         data_size = counter = 0; /* Set initial value */

         paint_window(15, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
         print_title(15, 23, 54, "Receiver waiting...", LIGHTGRAY, YELLOW);

         if ((rport(port) == '?'))
             sport(port, '.');
         else {
              while (rport(port) != '?' && !kbhit());
              if (kbhit()) { /* User interrupt */
                  snd5(); /* Beep */
                  break;  /* Return to monitors(); function */
              }
              sport(port, '.');
         }

         paint_window(15, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
         print_title(15, 23, 54, "Receiving data...", LIGHTGRAY, YELLOW);
```

```
        data_size = rport(port); /* Get data size */
            sport(port, '.');
            counter = data_size; /* copy to dummy variable */

        line = rport(port); /* Get position of row */
            sport(port, '.');

        for (i = 0; data_size; data_size--, i++) {
             buffers[i] = rport(port);
             sport(port, '.');
/*           windwheel(16); */
        }
        appear(counter, col, line);
        /* Display data to screen */
        paint_window(15, 23, 54, 23, 219, LIGHTGRAY);
        /* Clear message bar */
    }
}

unsigned int filesize(FILE *fp)
{
    unsigned long int i;

    i = 0;
    do {
        getc(fp);
        i++;
    } while(!feof(fp));
    rewind(fp);
    return(i - 1);
}

void send_file_name(char *f)
{
    union REGS r;

    paint_window(4, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
    print_title(4, 23, 54, "Transmitter waiting...",
                LIGHTGRAY, YELLOW);
    print_title(4, 24, 54, "Press any key to cancel",
                LIGHTGRAY, BLUE);

    r.x.ax = (r.x.ax - r.x.ax); /* Clear ax register */
    do {
        sport(port, '?');
    } while(!kbhit() && !(check_stat(port) & 256));

    if (kbhit()) {
        snd5(); /* Beep */
        goto Quit_snd_fname; /* Return to send_file(); function */
    }

        wait(port);

    paint_window(4, 23, 54, 24, 219, LIGHTGRAY); /* Clear message bar */
    write_string(17, 23, "Sending file", LIGHTGRAY_YELLOW);
    write_string(30, 23, f, LIGHTGRAY_BLACK);
    print_title(4, 24, 54, "Please wait...", LIGHTGRAY, BLUE);

    while (*f) {
                sport(port, *f++);
                wait(port);
    }
    sport(port, '\0');
Quit_snd_fname:; /* Quit_snd_fname Label */
```

```
}

void get_file_name(char *f)
{
   paint_window(8, 23, 54, 23, 219, LIGHTGRAY); /* Clear message bar */
   print_title(8, 23, 54, "Receiver waiting...",
               LIGHTGRAY, YELLOW);

   while (rport(port) != '?' && !kbhit());
   if (kbhit()) {
      snd5();
      goto Quit_get_fname; /* Return to rec_file(); function */
   }
      sport(port, '.');

   while ((*f = rport(port))) {
         if (*f != '?') {
            f++;
            sport(port, '.');
         }
   }
Quit_get_fname:; /* Quit_get_fname Label */
}

void wait(int port) /* Change */
{
   if (rport(port) != '.') {
/*    printf("\007");  Beep */
/*    paint_window(15, 23, 54, 24, 219, LIGHTGRAY);  Clear message bar */
/*    print_title(15, 23, 54, "Communication error.", LIGHTGRAY, RED); */
/*    print_title(15, 24, 54, "Please wait...", LIGHTGRAY, BLUE); */
/*    write_string(32, 24, "F6", LIGHTGRAY_RED);     */
/*    write_string(34, 24, "-Quit", LIGHTGRAY_BLUE);  */
   }
}

void wait_s(int port) /* New */
{
   do {
      sport(port, '?');
   } while((rport(port) != '.') && !kbhit());

   if (kbhit())
      w_flag = 1;
}

void sport(int port, char c)
{
   union REGS r;

   r.x.dx = port; /* Number of port */
   r.h.al = c; /* Character */
   r.h.ah = 1; /* Number of function */
   int86(0x14, &r, &r);

   if (r.h.ah & 128) {
      goodbye(); /* AH = 1xxx xxxx = Time-out Error */
      printf("Send error detected in serial port.\n");
      printf("\007"); /* Beep */
      exit(1); /* Return to DOS */
   }
}

rport(int port)
{
   union REGS r;
```

```
    r.x.ax = (r.x.ax - r.x.ax); /* Clear ax register */
    while (!(check_stat(port) & 256))
          if (kbhit()) /* Cancel if key pressed */
             goto Quit_rport;
    /* if AH = xxxx xxx1 xxxx xxxx = Data Ready */

    r.x.dx = port; /* Number of port */
    r.h.ah = 2; /* Number of function */
    int86(0x14, &r, &r); /* Call interrupt BIOS */

    if (r.h.ah & 128) {
        goodbye(); /* AH = 1xxx xxxx = Time-out Error */
        printf("Read error detected in serial port.\n");
        printf("\007"); /* Beep */
        exit(1); /* Return to DOS */
    }
Quit_rport:; /* Quit_rport Label */
    return r.h.al; /* Return input data */
}

check_stat(int port)
{
    union REGS r;

    r.x.ax = (r.x.ax - r.x.ax); /* Clear ax register */
    r.x.dx = port; /* Number of port */
    r.h.ah = 3; /* Number of function */
    int86(0x14, &r, &r); /* Call interrupt BIOS */

    return r.x.ax; /* Return flag */
}

void port_init(int port, unsigned char code)
{
    union REGS r;

    r.x.dx = port;
    r.h.ah = 0;
    r.h.al = code; /* Setup data */
    int86(0x14, &r, &r); /* Call interrupt BIOS */
}

void appear(char count, unsigned int col, unsigned int row) /* New */
{
    register int i;
    int next_row;
    char far *v;

    if (count > (LINE_LEN - 1)) /* Out of range */
        count = LINE_LEN - 1; /* count = Maximum character per line */

    if (row > MAX_LINES) /* Out of range */
        row = MAX_LINES; /* row = Maximum line per page */

    if (row <= prev_row) {
        paint_window(0, 1, 79, 22, 219, CYAN);
        col = row = prev_row = 1;
        write_char(0, row, '#', CYAN_RED); /* Display prompt sign '#' */
    }

    if ((row - prev_row) > 1) {
        for (i = 1; i < row; i++) {
            write_char(0, i, '#', CYAN_RED); /* Display prompt sign '#' */
            write_string(1, i, "-NULL-", CYAN_BLUE);
        }
```

```
        write_char(0, i, '#', CYAN_RED); /* Display prompt sign '#' */
    }

    write_char(0, row, '#', CYAN_RED); /* Display prompt sign '#' */

    for (i = 0, col = 1; count; count--, i++, col++) {
        /* Put data until end of line */
        v = vid_mem + (row * 160) + (col * 2);
        /* Calculate address of video RAM */
        if (buffers[i] == '\0')
            continue; /* Not display '\0' to screen */
        /* Set address of video RAM */
        *v++ = buffers[i]; /* Put normal character */
        *v = ATTRIB; /* Put attribute */
    }

    col = 1; /* col = Start column */
    prev_row = row;

    if (row == MAX_LINES)
        next_row = 1; /* Next page */
    else
        next_row = row + 1; /* Next line */

    write_char(0, next_row, '#', CYAN_RED); /* Display prompt sign '#' */
}

void init_array(void)
{
    strset(buffers, NULL_CHAR); /* Clear array */
}

void goodbye(void)
{
    SETOLD(); /* Return interrupt vector of timer */
    window(1, 1, 80, 25); /* Set window to the normal size */
    cls(); /* Clear screen */
}
```

```
#include "conio.h"
#include "dos.h"
#include "stdio.h"

/**********************************
 * D E F I N E   C O N S T A N T S *
 **********************************/

#define I        178
#define O        255
#define X         15
#define Y          5
#define CYAN_BLUE 49

#define ROW            24
#define DELAY_TIME      1
#define LIGHTGRAY_BLUE 113 /* Background = LIGHTGRAY, Foreground = BLUE */

/*********************************************
 * D E C L A R A T I O N   F U N C T I O N S *
 *********************************************/

void print_logo(void);
void windwheel(int col);
void write_char(int x, int y, char ch, int attrib); /* External function */

int fonts[5][50] = {
    I, I, I, I, I, I, O, I, I, I, I, I, I, O, I, I, I, I, I, I, O, I, I, I, I, I, I, O, I, I, O, I, I, I, I, I, O, I, I, O, I, I, I, I, I, O, I, I, I, O,
    I, I, O, O, I, I, O, I, I, O, O, I, I, O, I, I, O, O, O, O, O, O, O, I, I, O, O, O, I, I, I, O, I, I, O, I, I, O, O, I, I, O, I, I, I, O, I, I, O, I,
    I, I, I, I, I, I, O, I, I, O, O, I, I, O, I, I, I, I, I, I, I, O, O, O, I, I, O, O, O, I, I, I, I, I, I, I, O, I, I, I, I, I, I, O, I, I, O, I, I, O,
    I, I, O, O, O, O, O, I, I, O, O, I, I, O, O, O, O, O, I, I, O, O, O, I, I, O, O, O, I, I, O, I, O, I, I, O, I, I, O, O, I, I, O, I, I, O, I, I, O, O,
    I, I, O, O, O, O, O, I, I, I, I, I, I, O, I, I, I, I, I, I, O, O, O, I, I, O, O, O, I, I, O, O, O, I, I, O, I, I, O, O, I, I, O, I, I, O, I, I, O, O,
};

int pattern[9] = {196, 92, 124, 47, 196, 92, 124, 47, 196};
             /* -- , \ , ¦ , / , -- , \ , ¦ , / , -- */

void print_logo(void)
{
   register int i, j;

   for (i = 0; i < 5; i++)
      for (j = 0; j < 50; j++)
           write_char((X + j), (Y + i), fonts[i][j], CYAN_BLUE);
}

void windwheel(int col)
{
   register int i;

   for (i = 0; i < 9; i++) {
        write_char(col, ROW, pattern[i], LIGHTGRAY_BLUE);
        delay(DELAY_TIME);
        write_char(col, ROW, ' ', LIGHTGRAY_BLUE); /* Clear */
   }
   write_char(col, ROW, pattern[0], LIGHTGRAY_BLUE);
}
```

```
#include "conio.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

#define BIT0_1   3
#define BIT2     4
#define BIT3_4  24
#define BIT5_7 224

char setup_code[35] = "x-Port number, xxx-Modem setup code";
char port_number;
int port;
char modem_setup[5] = "     ";
char *ptr;
int modem_code;
int data_code;
char ans;

int data_index[] = {
    0,                              /* Dummy value */
    2, 3,                           /* bit 0, 1    */
    0, 4,                           /* bit 2       */
    0, 16, 8, 24,                   /* bit 3, 4    */
    0, 32, 64, 96, 128, 160, 192, 224 /* bit 5, 6, 7 */
};

char *data_list[] = {
     "Error",
     "7", "8",                       /* bit 0, 1    */
     "1", "2",                       /* bit 2       */
     "No", "No", "Odd", "Even",      /* bit 3, 4    */
     "110 bps", "150 bps", "300 bps", "600 bps", "1200 bps", "2400 bps",
     "4800 bps", "9600 bps"          /* bit 5, 6, 7 */
};

int tolower(int ch); /* C function */

void main(void)
{
   FILE *fp;
   register int i;

   for (;;) {
        clrscr(); /* Clear screen */
        gotoxy(26, 1); printf("*****************************");
        gotoxy(26, 2); printf("*        -POSTMAN-          *");
        gotoxy(26, 3); printf("* S E T U P   P R O G R A M *");
        gotoxy(26, 4); printf("*       version 1.0         *");
        gotoxy(26, 5); printf("*****************************");

        do {
            gotoxy(10, 7);
            printf("Enter port number [0 - 6] : ");
            port_number = getche();
        } while((port_number < '0') || (port_number > '6'));
        port = port_number - 48;

        do {
            gotoxy(10, 8);
            printf("Enter modem setup code, Decimal number only [2 - 255] : ");
            modem_setup[0] = 4;
            ptr = cgets(modem_setup);
            modem_code = atoi(ptr);
        } while((modem_code < 2) || (modem_code > 255));
```

```
        gotoxy(32, 10); printf("-Show Setup Data-");
        gotoxy(25, 12); printf("Port Number       = %d", port);

        data_code = modem_code & BIT5_7;
        for (i = 9; i <= 16; i++)
            if (data_code == data_index[i])
                break;
        if (i > 16)
            i = 0;
        gotoxy(25, 13); printf("Baud Rate         = %s", data_list[i]);

        data_code = modem_code & BIT3_4;
        for (i = 5; i <= 8; i++)
            if (data_code == data_index[i])
                break;
        if (i > 8)
            i = 0;
        gotoxy(25, 14); printf("Parity            = %s", data_list[i]);

        data_code = modem_code & BIT2;
        for (i = 3; i <= 4; i++)
            if (data_code == data_index[i])
                break;
        if (i > 4)
            i = 0;
        gotoxy(25, 15); printf("Stop Bit          = %s", data_list[i]);

        data_code = modem_code & BIT0_1;
        for (i = 1; i <= 2; i++) {
            if (data_code == data_index[i])
                break;
        }
        if (i > 2)
            i = 0;
        gotoxy(25, 16); printf("Data Bit [1 Byte] = %s", data_list[i]);

        gotoxy(28, 18); printf("Is this all right (y/n)?");
        do {
            gotoxy(53, 18);
            ans = tolower(getche());
        } while((ans != 'y') && (ans != 'n'));
        if (ans == 'y')
            break;
    }
    gotoxy(25, 20); printf("Save setup data to postman.ini");
    gotoxy(33, 21); printf("Please wait...");

    setup_code[0] = port_number; /* Put port number */
    for (i = 0; i <= 2; i++)     /* Put modem setup code */
        setup_code[i + 15] = *ptr++;

    remove("postman.ini");
    if (!(fp = fopen("postman.ini", "wt"))) {
        clrscr();
        printf("Cannot open postman.ini\n");
        printf("\007");
        exit(1);
    }
    for (i = 0; i <= 34; i++) {
        putc(setup_code[i], fp);
        if (ferror(fp)) {
            clrscr();
            printf("Error writing postman.ini\n");
            printf("\007");
            exit(1);
```

```
        }
    }
    fclose(fp);
    gotoxy(35, 23); printf("-Success-");
}
```

```
#include "stdio.h"
#include "conio.h"
#include "dos.h"
#include "string.h"
#include "math.h"

/***********************************
 * D E F I N E   C O N S T A N T S *
 ***********************************/

#define ROW            12
#define MIDDLE         38
#define SHOW_TIME    2000
#define DELAY_TIME      7
#define LIGHTGRAY_BLUE 113 /* Background = LIGHTGRAY, Foreground = BLUE */

/*********************************************
 * D E C L A R A T I O N   F U N C T I O N S *
 *********************************************/

void rotate_message(void);
void suck(char *message);
void jet(char *message);

void goto_xy(int x, int y); /* External function */
void snd5(void); /* External function */
void write_char(int x, int y, char ch, int attrib); /* External function */

/*********************************************
 * D E C L A R A T I O N   V A R I A B L E S *
 *********************************************/

char *message1 =
"Programmer";

char *message2 =
"Waranyoo Khomkham";

char *message3 =
"Charlie Sookthet";

char *message4 =
"Charlie Sookthet";

void rotate_message(void)
{
   do {
       jet(message1); delay(SHOW_TIME); suck(message1);
       jet(message2); delay(SHOW_TIME); suck(message2);
/*     jet(message3); delay(SHOW_TIME); suck(message3); */
/*     jet(message4); delay(SHOW_TIME); suck(message4); */
   } while(!kbhit());
   snd5();
}

void suck(char *message)
{
   register int i;
            int count, first_ch, last_ch;

   count = (strlen(message) / 2);
   first_ch = MIDDLE - count;
   last_ch = MIDDLE + count;

   for (i = 0; i <= (count - 1); i++) {
       write_char(first_ch + i, ROW, ' ', LIGHTGRAY_BLUE);
```

```
        delay(DELAY_TIME);
        write_char(last_ch - i, ROW, ' ', LIGHTGRAY_BLUE);
        delay(DELAY_TIME);
    }
    write_char(MIDDLE, ROW, ' ', LIGHTGRAY_BLUE);
    delay(DELAY_TIME);
}

void jet(char *message)
{
    register int i;
             int count;

    count = ceil((strlen(message) / 2));

    write_char(MIDDLE, ROW, message[count], LIGHTGRAY_BLUE);
    delay(DELAY_TIME);
    for (i = 1; i <= (count); i++) {
        write_char(MIDDLE - i, ROW, message[count - i], LIGHTGRAY_BLUE);
        delay(DELAY_TIME);
        write_char(MIDDLE + i, ROW, message[count + i], LIGHTGRAY_BLUE);
        delay(DELAY_TIME);
    }
}
```

```
DGROUP GROUP _DATA

_DATA SEGMENT WORD PUBLIC 'DATA'

_DATA ENDS

_TEXT SEGMENT WORD PUBLIC 'CODE'
     ASSUME CS:_TEXT,DS:DGROUP

     DOS_FUN  EQU 21H ;Interrupt of DOS that service the user
     GETVEC   EQU 35H ;Service 35h of DOS to get address of interrupt
     SETVEC   EQU 25H ;Service 25h of DOS to set address of interrupt
     TIME     EQU 1CH ;BIOS software interrupt 1Ch for timer.
                      ;We will change this interrupt.
     GET_TIME EQU 2CH ;Service 2Ch of DOS to get the current time.

     _OLDSEG   DW ?   ;Variable to save the segment of interrupt
                      ;service routine 1Ch
     _OLDOFF   DW ?   ;Variable to save the offset of interrupt
                      ;service routine 1Ch
     SIXTY     DB 60
     TW4       DB 24
     COUNT     DB 0
     MODE      DB ?
     VIDEO_RAM DW ?

     EXTRN _HOUR : BYTE
     EXTRN _MIN  : BYTE
     EXTRN _SEC  : BYTE
     EXTRN _FLAG : BYTE   ;Hour,Min,Sec,Flag are the variables that
                          ;must be defined in the C main program

NEWINT PROC FAR

      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH ES
      PUSH SI
      PUSH DI
      PUSH BP

      MOV BH,1EH          ;Attribute Reverse Video
      MOV BL,':'
      MOV ES,VIDEO_RAM    ;Get segment of video RAM
      MOV DI,70*2         ;Calculate the offset in video RAM
      MOV DL,COUNT
      CMP DL,18           ;We will change the time every count = 18
      JNZ INCRE

      XOR AH,AH
      MOV AL,_SEC
      INC AL
      MOV _SEC,AL
      DIV SIXTY
      MOV _SEC,AH         ;Adjust the variable SEC of C main program
      ADD _MIN,AL
      XOR AH,AH
      MOV AL,_MIN
      DIV SIXTY
      MOV _MIN,AH         ;Adjust the variable MIN of C main program
      ADD _HOUR,AL
      XOR AH,AH
      MOV AL,_HOUR
```

```
        DIV TW4
        MOV _HOUR,AH        ;Adjust the variable HOUR of C main program
        MOV AL,AH
        CALL HEX2BCD
        MOV ES:[DI],BX
        INC DI
        INC DI
        MOV AL,_MIN
        CALL HEX2BCD
        MOV ES:[DI],BX
        INC DI
        INC DI
        INC DH
        MOV AL,_SEC
        CALL HEX2BCD
        XOR DL,DL
        JMP NOT_INC

INCRE:  INC DL

NOT_INC: MOV COUNT,DL

          POP BP
          POP DI
          POP SI
          POP ES
          POP DX
          POP CX
          POP BX
          POP AX
          STI
          IRET

NEWINT ENDP

HEX2BCD PROC NEAR           ;Input in AL = HEX, Output in AX = BCD

         XOR AH,AH

BEG:     CMP AL,0AH
         JB EXT
         SUB AL,0AH
         INC AH
         JA BEG

EXT:     ADD AH,30H         ;Display Upper BCD
         MOV ES:[DI],AH
         INC DI
         MOV ES:[DI],BH
         INC DI
         ADD AL,30H
         MOV ES:[DI],AL     ;Display Lower BCD
         INC DI
         MOV ES:[DI],BH
         INC DI
         RET

HEX2BCD ENDP

        PUBLIC _SHOW

_SHOW PROC NEAR             ;Show line and column of the cursor position

        PUSH BP             ;Save BP
        MOV BP,SP           ;BP = SP
        MOV ES,VIDEO_RAM    ;Load segment of video RAM
```

```
        MOV DI,0E6AH        ;Load offset of video RAM
        MOV BH,71H          ;Attribute reverse video
        MOV AL,[BP+6]       ;Load the second parameter (Y)
        CALL HEX2BCD
        MOV DI,0E78H
        MOV AL,[BP+4]       ;Load the first parameter (X)
        CALL HEX2BCD
        POP BP
        RET

_SHOW ENDP

        PUBLIC _SETNEW

_SETNEW PROC NEAR

        MOV AX,0040H
        MOV ES,AX
        MOV DI,0049H
        MOV AH,ES:[DI]
        MOV MODE,AH         ;Check mode by reading from the BIOS data area
                            ;and save in variable MODE
        CMP MODE,07H        ;If MODE = 07h, it means that it is monochrome.
        JZ MONO

        MOV VIDEO_RAM,0B800H
        JMP CONTI

MONO:   MOV VIDEO_RAM,0B000H

CONTI:  MOV AH,GET_TIME     ;Get the current time
        INT DOS_FUN
        MOV _HOUR,CH        ;Save the hour in variable HOUR
        MOV _MIN,CL         ;Save the minute in variable MIN
        MOV _SEC,DH         ;Save the second in variable SEC
        MOV AL,TIME
        MOV AH,GETVEC
        INT DOS_FUN         ;Get the interrupt vector
        MOV _OLDSEG,ES      ;Save segment in variable OLDSEG
        MOV _OLDOFF,BX      ;Save offset in variable OLDOFF
        PUSH DS
        PUSH CS
        POP DS
        MOV DX,OFFSET NEWINT
        MOV AL,TIME
        MOV AH,SETVEC
        INT DOS_FUN         ;Set the new interrupt vector
        POP DS
        RET

_SETNEW ENDP

        PUBLIC _SETOLD

_SETOLD PROC NEAR

        MOV DX,_OLDOFF
        MOV AX,_OLDSEG
        PUSH DS
        PUSH AX
        POP DS
        MOV AL,TIME
        MOV AH,SETVEC
        INT DOS_FUN         ;Return the old interrupt vector
        POP DS
        RET
```

```
_SETOLD ENDP

        PUBLIC _TOGGLECUR

_TOGGLECUR PROC NEAR      ;CH = start line of cursor, CL = end line of cursor
                          ;For monochrome if FLAG = 1 then CH = 10, CL = 12
                          ;else CH = 1, CL = 12
                          ;For another display if FLAG = 1 then CH = 6,
                          ;CL = 7 else CH = 1, CL = 7
        CMP MODE,07H
        JZ CHROM

        MOV CL,7
        MOV AL,0
        CMP _FLAG,AL
        JZ ZERO1
        MOV CH,6
        JMP INTER

ZERO1:  MOV CH,1
        JMP INTER

CHROM:  MOV CL,12
        MOV AL,0
        CMP _FLAG,AL
        JZ ZERO

        MOV CH,10
        JMP INTER

ZERO:   MOV CH,1

INTER:  MOV AH,1
        INT 10H
        RET

_TOGGLECUR ENDP

      PUBLIC _INKEY

_INKEY PROC NEAR

      XOR AH,AH
      INT 16H             ;Get scan code of keyboard and return in AX
      RET

_INKEY ENDP

_TEXT ENDS
     END
```

```
#include "alloc.h"
#include "bios.h"
#include "conio.h"
#include "ctype.h"
#include "dos.h"
#include "stdio.h"
#include "string.h"

/***********************************
* D E F I N E   C O N S T A N T S *
***********************************/

#define BUF_SIZE  79
#define LINE_LEN  79
#define MAX_LINES 22
#define NULL_CHAR 32 /* NULL_CHAR = 32 = 0x20 = ' ' */

#define LEFT   75
#define RIGHT  77
#define HOME   71
#define END    79
#define INSERT 82
#define DELETE 83
#define ALT_X  45
#define F1     59
#define F2     60
#define F6     64
#define ESC    27

/************************
* C O L O R   C O D E *
************************/

#define ATTRIB               48 /* Background = CYAN, Foreground = BLACK      */
#define CYAN_RED             52 /* Background = CYAN, Foreground = RED        */
#define LIGHTGRAY_BLACK     112 /* Background = LIGHTGRAY, Foreground = BLACK  */
#define LIGHTGRAY_BLUE      113 /* Background = LIGHTGRAY, Foreground = BLUE   */
#define LIGHTGRAY_BLUE_BLINK 241 /* Background = LIGHTGRAY, Foreground = BLUE   */
                                 /* And Blink                                  */
#define LIGHTGRAY_CYAN      115 /* Background = LIGHTGRAY, Foreground = CYAN   */
#define LIGHTGRAY_RED       116 /* Background = LIGHTGRAY, Foreground = RED    */
#define LIGHTGRAY_WHITE     127 /* Background = LIGHTGRAY, Foreground = WHITE  */
#define LIGHTGRAY_YELLOW    126 /* Background = LIGHTGRAY, Foreground = YELLOW */

/*********************************************
* D E C L A R A T I O N   V A R I A B L E S *
*********************************************/

char buf[BUF_SIZE];

char *ins = "Insert";

char *ovw = "OvrWrt";

unsigned int curloc, endloc;

int x, y;
int max_y = 1;
int FLAG  = 1; /* Insert mode */

extern char far *vid_mem;

extern union inkey {
       char ch[2];
       int  i;
```

```
} key;

/*********************************************
*D E C L A R A T I O N   F U N C T I O N S*
*********************************************/

void edit(void);
void backspace(void);
void clear_screen(void);
void delete(void);
void display(int line);
void dis_ins(void);
void end(void);
void enter(void);
void erase(void);
void home(void);
void init_vars(void);
void left(void);
void right(void);
void set_screen(void);

void SHOW(int x, int y); /* External function, TIME.ASM */
void TOGGLECUR(void);    /* External function, TIME.ASM */
void write_char(int x, int y, char ch, int attrib);   /* External function */
void write_string(int x, int y, char *p, int attrib); /* External function */
int get_key(void); /* External function */
void snd5(void);   /* External function */
void print_title(int startx, int starty, int endx, char *title,
                 int textb, int textc);       /* External function */
void paint_window(int startx, int starty, int endx, int endy,
                 int ascii_code, int attrib); /* External function */
void restore_screen(void); /* External function */
void send(char *buff, char y); /* External function */
void windwheel(int col); /* External function */

char *strset(char *s, int ch); /* C function */

/* Edit uses normal principle as general editors. */

void edit(void)
{
set_screen();
window(2, 2, 80, 23);
dis_ins();
TOGGLECUR();

init_vars();
x = y = 1;
write_char(0, y, '>', CYAN_RED); /* Display prompt sign '>' */
gotoxy(1, 1);
SHOW(x, y);

do {
key.i = get_key(); /* Wait for key */
if (!key.ch[0]) {  /* If keypressed is not character */
       switch(key.ch[1]) {
              case F1     : erase();        /* Delete line */
                     break;

         case F2     : clear_screen(); /* Clear screen */
                        break;

              case LEFT   : left();         /* Move left */
                            break;

              case RIGHT  : right();        /* Move right */
```

```
                        break;

                case INSERT : FLAG++;           /* Insert & Overwrite mode */
                        if (FLAG > 1 || FLAG < 0)
                                FLAG = 0;
                        TOGGLECUR();
                        dis_ins();
                        break;

                case DELETE : delete();
                        break;          /* Delete */

                case HOME   : home();           /* Home */
                        break;

                  case END    : end();          /* End */
                        break;
        }
        gotoxy(x, y);
        SHOW(x, y);
}
else {
        switch (key.ch[0]) {
                case '\r' : enter();            /* Enter */
                        break;

                case '\b' : backspace();        /* Backspace */
                        break;

                default   :                     /* Key pressed */
        /* OvrWrt mode */ if ((FLAG == 0) && (curloc == BUF_SIZE - 1))
                        break;          /* Buffer full */
        /* Insert mode */ if ((FLAG == 1) && (endloc == BUF_SIZE - 1))
                        break;          /* Buffer full */
                    if (x == LINE_LEN)
                       break;          /* End column? */
                    if (FLAG == 1)
                       memmove(buf + curloc + 1, buf + curloc,
                                    endloc - curloc + 1);
                     buf[curloc] = key.ch[0];
                     /* Save key to buffer */
                     x++;
                     if ((FLAG == 0) && (curloc < endloc)) {
                            curloc++;
                    }
                     else {
                               curloc++;
                            endloc++;
                     }
        }
        gotoxy(x, y);
        SHOW(x, y);
}
display(y);       /* display data in buffer to screen */
} while(!(key.ch[0] == 0 && key.ch[1] == F6));
/* Loop until F6 were pressed */
max_y = 1;
FLAG = 1;
TOGGLECUR();
snd5();
restore_screen();
}

/* Delete the previous character */

void backspace(void)
```

```
{
/* Check backspace non character */
if (curloc != 0) {
x--; /* Delete normal character */
memmove(buf + curloc - 1, buf + curloc, endloc - curloc + 1);
curloc--;
endloc--;
}
}

/* Clear screen */

void clear_screen(void)
{
paint_window(0, 1, 79, 22, 219, CYAN);
init_vars();
x = y = 1;
max_y = 1;
   write_char(0, y, '>', CYAN_RED); /* Display prompt sign '>' */
}

/* Delete character */

void delete(void)
{
/* Delete only character in line */
if (curloc != endloc) {
memmove(buf + curloc, buf + curloc + 1, endloc - curloc);
/* Update character */
endloc--;
/* Update EOF */
}
}

/* To display from text buffers */

void display(int line)
{
char far *v;
int col, row;
int n, l;

col = 1;
row = line;
       n = 0;

   /* Put data until end of line */
while (col <= 79) {
  v = vid_mem + (row * 160) + (col * 2);
       /* Set address of video RAM */
       *v++ = buf[n];              /* Put normal character */
  *v = ATTRIB;                /* Put attribute        */
  n++;
  col++;
}
}

/* Display "Insert" or "OvrWrt" */

void dis_ins(void)
{
window(1, 1, 80, 25);         /* Set window to the old size */
if (FLAG == 0)
print_title(0, 24, 13, ovw, LIGHTGRAY, RED);
/* If FLAG = 0 display "OvrWrt"
       FLAG = 1 display "Insert" */
```

```
        else
        print_title(0, 24, 13, ins, LIGHTGRAY, RED);
        window(2, 2, 80, 23);          /* Set window for editing */
        }

        /* Move cursor to end column of message in line */

        void end(void)
        {
        if (buf[endloc - 1] == ' ')
        do {
               endloc--;
        } while(buf[endloc - 1] == ' ');
               x = endloc + 1;
        curloc = x - 1;
        }

        /* When press 'Enter' */

        void enter(void)
        {
        if (curloc != endloc)

endloc - 1] == ' ')
          do {
             endloc--; /* Delete null character */
          } while(buf[endloc - 1] == ' ');
        curloc = endloc;

        if (max_y <= MAX_LINES) {
        memmove(buf + curloc + 1, buf + curloc, endloc - curloc + 1);
        buf[curloc] = '\0'; /* Fill 1 EOF */

        send(buf, y); /* Send data */

              max_y++;
        init_vars();
        x = 1;
        y++;

        if (y > MAX_LINES) {
               paint_window(0, 1, 79, 22, 219, CYAN);
               init_vars();
               x = y = 1;
               max_y = 1;
        }
        }
        write_char(0, y, '>', CYAN_RED); /* Display prompt sign '>' */
        }

        /* Delete line */

        void erase(void)
        {
        if (endloc != 0) {
        init_vars();
        x = 1;
        }
        }

        /* Move cursor to start column (x = 1) in line */

        void home(void)
        {
        if (x > 1) {
```

```
x = 1;
curloc = 0;
}
}

/* Clear array and set curloc = endloc = 0 */

void init_vars(void)
{
strset(buf, NULL_CHAR);
curloc = endloc = 0;
}

/* Move cursor to left 1 position */

void left(void)
{
if (x > 1) {
x--;
curloc--;
}
}

/* Move cursor to right 1 position */

void right(void)
{
if (curloc < endloc)
if (x < LINE_LEN) {
          x++;
        curloc++;
}
}

void set_screen(void)
{
paint_window(0, 1, 79, 22, 219, CYAN);
paint_window(0, 23, 54, 24, 219, LIGHTGRAY); /* Clear line 23, 24 */
write_string(0, 23, " Col   :Row    ", LIGHTGRAY_BLUE);
write_char(14, 23, 179, LIGHTGRAY_CYAN);
write_char(14, 24, 179, LIGHTGRAY_CYAN);
write_string(32, 24, "F6", LIGHTGRAY_RED);
write_string(34, 24, "-Quit", LIGHTGRAY_BLUE);
}
```

บรรณานุกรม

1. สุพจน์ ปุณณชัยยะ,"โมเด็ม",บริษัทอินฟอร์เมติก บิซิเนส พับลิเคชั่น จำกัด,2534
2. ชูชัย ธนสารตั้งเจริญ,ทินกร ดุ๊ก,"การสื่อสารข้อมูล",สำนักพิมพ์ ฟิสิกส์เซนเตอร์
3. คิวัฒน์ ศิวะบวร,พรชัย จักรธำรงค์,จิรศักดิ์ ชัยวิริะกุล,"การประยุกต์ใช้งานภาษาซี", บริษัทซีเอ็ดยูเคชั่น
4. บรรเจิด ตันติกัลยาภรณ์,"เครื่องรับส่ง"
5. J.Michael Jacob,"Application and Design with Analog Integrated Circuits,Second edition",Printice-Hall,International,Inc.,1993
6. Staff,"XR-2206 Monolitic Function,XR-2211 FSK Demodulator/Tone Decodes,XR-1488/1489A Quad Line Driver/Receiver,XR-084 Quad Bipolar JFET Operational Amplifier",Exar Data Book,Exar Corp., 1986