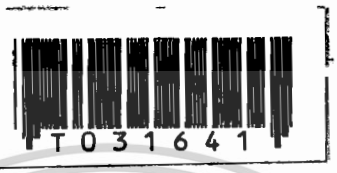
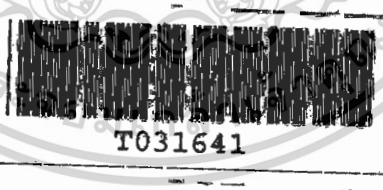
โมดูลส่งไฟล์เลขฐานสอง



นายภูเมศร์ จินดาจิธาวัฒน์



โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต
สาขาวิชาฟิสิกส์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2540

Binary File Transferring Module

Mr.Poomate Jindajitawat

A Special Project Submitted in Partial Fulfillment of the

Requirement for the Degree of Bachelor of Science

Department of Applied Physics

Faculty of Science

King Mongkut's Institute of Technology Ladkrabang

1997

หัวข้อโครงงานพิเศษ	โมดูลส่งไฟล์เลขฐานสอง
โดย	นายภูเมศร์ จินดาจิธาวัฒน์
ภาควิชา	ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์วิชิต ศิริโชติ

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

.....
(รองศาสตราจารย์สุรพล รักวิจัย) หัวหน้าภาควิชาฟิสิกส์ประยุกต์

คณะกรรมการโครงงานพิเศษ

.....
(ผู้ช่วยศาสตราจารย์วิชิต ศิริโชติ) ประธานกรรมการ

.....
(อาจารย์สุรวุฒิ กิจสัมพันธ์) กรรมการ

.....
(อาจารย์ทรงวิทย์ เจริญรวยวัฒนา) กรรมการ

.....
(อาจารย์บดินทร์ ดำรงศักดิ์) กรรมการ

หัวข้อโครงงานพิเศษ	โมดูลส่งไฟล์เลขฐานสอง
โดย	นายภูเมศร์ จินดาจิธาวัฒน์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์วิชิต ศิริโชติ
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	2540

บทคัดย่อ

โครงงานพิเศษนี้เป็นการพัฒนาอุปกรณ์ที่ทำหน้าที่ในการติดต่อรับและส่งไฟล์ไบนารีกับคอมพิวเตอร์ระยะไกล มีชื่อเรียกว่า โมดูลส่งไฟล์ไบนารี ซึ่งอาศัยเครือข่ายระบบโทรศัพท์สาธารณะเป็นตัวกลางในการสื่อสาร โดยโมดูลส่งไฟล์ไบนารีนี้ทำหน้าที่ในการเก็บข้อมูลไบนารีจากอุปกรณ์อื่น ๆ แล้วจะทำการบันทึกไว้ในหน่วยความจำแบบ nonvolatile ที่มีขนาด 32 กิโลไบต์ ซึ่งทำหน้าที่ในการพักข้อมูลก่อนที่จะทำการส่งข้อมูลไบนารีนั้นให้กับคอมพิวเตอร์ที่ติดต่อเข้ามา โดยวงจรหลักของโมดูลส่งไฟล์ไบนารีประกอบไปด้วย

1) วงจรไมโครคอนโทรลเลอร์ DS5000T
2) วงจรการสื่อสารอนุกรมมาตรฐาน RS232C
3) โมเด็ม

มีการใช้โปรโตคอล XMODEM มาตรฐานเป็นโปรโตคอลที่ใช้ในการรับและส่งไฟล์ โดยชุดของข้อมูลมีขนาด 128 ไบต์และมีการตรวจสอบความผิดพลาดด้วยค่า checksum ซึ่งสามารถทำการพัฒนาโดยการเพิ่มการรับส่งข้อมูลด้วยโปรโตคอล ZMODEM และ kermit เพื่อเพิ่มประสิทธิภาพในการรับส่งข้อมูล

Special Project Title	Binary File Transferring Module
Name	Mr.Poomate Jindajitawat
Special Project Advisor	Asst. Prof. Wichit Sirichote
Department	Applied Physics
Academic Year	1997

Abstract

A device used for transmitting and receiving binary file to and from remote host computer has been built. The device, namely Binary File Transferring Module, employs public telephone network as a communication media. The module functional is collecting the binary data form another devices saved to the nonvolatile 32kB memory as a data buffering before transmitting to the dial in computer. Main circuit of the module comprising 1) a soft microcontroller, DS5000T, 2) RS232C interfacing circuit and 3) MODEM. The uploading and downloading protocol is a standard XMODEM with a simple checksum for error detection. A data packet is 128 byte long. In Addition, ZMODEM and kermit protocol was also studied.

กิตติกรรมประกาศ

โครงงานพิเศษฉบับนี้ข้าพเจ้ามีความภูมิใจเป็นอย่างมากเนื่องจากข้าพเจ้าทุ่มเทให้กับการทำงานอย่างตั้งใจจริงเพื่อให้ผลงานออกมาด้วยดี นอกจากความสามารถของข้าพเจ้าแล้วยังได้รับความอนุเคราะห์จากบุคคลหลายฝ่าย ดังนี้

บิดา มารดา	บุคคลผู้ให้กำเนิดข้าพเจ้าให้มีอาการครบ 32 ประการ และส่งเสียทุนทรัพย์ให้ข้าพเจ้าได้รับการศึกษาจนมีความรู้พอเพียงที่จะทำโครงงานพิเศษฉบับนี้ได้
ผู้ช่วยศาสตราจารย์วิชิต ศิริโชติ	ผู้ให้วิชาความรู้ และคำแนะนำต่าง ๆ
อาจารย์ภาควิชาฟิสิกส์ประยุกต์	ที่ได้ให้ความรู้ต่าง ๆ ซึ่งสามารถนำมาใช้ในการทำโครงงานพิเศษฉบับนี้ได้
คุณพรทิพย์ เตรียมประทีป	ผู้ที่คอยให้คำปรึกษาการช่วยเหลือและให้กำลังใจ
คุณยงยุทธ อารีย์รัตน์	ผู้ที่คอยช่วยเหลือในด้านอุปกรณ์
คุณจิตตกานต์ ไชยปัญญา	ผู้ที่คอยช่วยเหลือในด้านการพิมพ์
เพื่อนๆ	ที่คอยให้ความช่วยเหลือและให้กำลังใจในการทำโครงงานพิเศษฉบับนี้ตลอดมา

ท้ายที่สุดนี้ข้าพเจ้าขออุทิศความดีทั้งหมดที่ทุก ๆ ท่านทั้งที่กล่าวนามและไม่ได้กล่าวนามในที่นี้ที่ได้ให้การช่วยเหลือข้าพเจ้าในการทำโครงงานพิเศษฉบับนี้ จงดลบันดาลให้ทุกท่านได้ประสบแต่ความสุขความเจริญยิ่งขึ้นไป

นายภูเมศร์ จินดาจิธาวัฒน์

สารบัญ

สารบัญภาพ

หน้า

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

บทที่ 3 การดำเนินการวิจัย

บทที่ 4 การทดลอง

สารบัญตาราง

หน้า

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

บทที่ 3 การดำเนินการวิจัย

บทที่ 1
บทนำ

ในปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทในงานทางด้านอิเล็กทรอนิกส์อย่างมาก เนื่องมาจากอุปกรณ์ทางอิเล็กทรอนิกส์บางชนิดทำหน้าที่ในการเก็บข้อมูลจากเหตุการณ์ต่าง ๆ เพื่อนำมาศึกษาและวิเคราะห์ด้วยโปรแกรมคอมพิวเตอร์เพื่อนำไปใช้ในงานพัฒนาด้านต่าง ๆ ดังนั้นจึงทำให้ต้องมีการติดต่อสื่อสารเพื่อทำการถ่ายโอนข้อมูลระหว่างอุปกรณ์ทางอิเล็กทรอนิกส์เหล่านั้นกับคอมพิวเตอร์อยู่เสมอ โดยทั่วไปข้อมูลทางด้านอิเล็กทรอนิกส์จะถูกเก็บมาจากหลายสถานที่และมักจะมีการบันทึกอยู่ในลักษณะของรหัสเลขฐานสองจึงทำให้ต้องมีการเดินทางในการเก็บข้อมูลอยู่บ่อย ๆ เพื่อเป็นการลดค่าใช้จ่ายและเวลาในการเดินทาง และเพื่อความสะดวกในการจัดเก็บข้อมูลจากสถานที่ต่าง ๆ จึงเป็นที่มาของแนวคิดของโครงงานพิเศษนี้ โครงงานนี้จะเป็นการสร้างเครื่องมือที่ทำหน้าที่ในการถ่ายโอนข้อมูลจากอุปกรณ์ทางอิเล็กทรอนิกส์มาเก็บไว้ในหน่วยความจำแบบ NVRAM เพื่อรอที่จะทำการส่งข้อมูลเหล่านั้นให้กับคอมพิวเตอร์ที่ทำการติดต่อเข้ามา ซึ่งในการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับเครื่องส่งไฟล์ไบนารีนี้จะอาศัยเครือข่ายของระบบโทรศัพท์เพื่อเป็นสื่อกลางในการส่งข้อมูล ทำให้ขอบข่ายในการทำงานของเครื่องส่งไฟล์กว้างไกลยิ่งขึ้น ในลักษณะการทำงานส่งไฟล์ไบนารีนั้นได้มีการออกแบบให้มีการส่งเป็นไปตามมาตรฐานของการสื่อสารข้อมูลอนุกรมโดยมีการใช้โปรโตคอลในการควบคุมการส่งถ่ายไฟล์ข้อมูล ซึ่งจะเป็นผลทำให้การส่งไฟล์มีประสิทธิภาพและความถูกต้องแม่นยำมากยิ่งขึ้น

1.1 วัตถุประสงค์

จากที่ได้กล่าวมาข้างต้นจึงได้มีการพัฒนาสร้างเครื่องส่งไฟล์ไบนารีระยะไกลนี้ขึ้น โดยโครงงานนี้จะมีวัตถุประสงค์ คือ

1.1.1 เพื่อเป็นตัวกลางในการทำหน้าที่รับ-ส่งไฟล์ระหว่างอุปกรณ์ทางอิเล็กทรอนิกส์กับคอมพิวเตอร์

1.1.2 เพื่อเพิ่มประสิทธิภาพในการรับส่งไฟล์ระหว่างอุปกรณ์ทางอิเล็กทรอนิกส์กับคอมพิวเตอร์

1.1.3 เพื่อลดค่าใช้จ่ายและเวลาที่ต้องใช้ในการเดินทางไปเก็บข้อมูลยังสถานที่ต่าง ๆ

1.2 วิธีการดำเนินงาน

ขั้นตอนในการดำเนินการสร้างเครื่องส่งไฟล์ไบนารีระยะไกลนั้น มีดังนี้

1.2.1 ทำการศึกษาและพิจารณาข้อมูลที่เกี่ยวข้องแล้วทำการเลือกแนวทางในการทำงานของเครื่อง

1.2.2 ทำการออกแบบและพิจารณาหน้าที่และส่วนประกอบต่าง ๆ ของระบบ

1.2.3 สร้างวงจรการทำงานของไมโครคอนโทรลเลอร์ DS5000

1.2.4 สร้างและติดตั้งส่วนที่ทำหน้าที่ในการติดต่อเชื่อมโยงเครือข่ายระบบโทรศัพท์

1.2.5 ออกแบบและพัฒนาโปรแกรมที่ใช้ในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ DS5000 ในการรับ-ส่งไฟล์ตามลักษณะของโปรโตคอล XMODEM

1.2.6 ทำการทดลองรับส่งไฟล์ผ่านเครือข่ายและตรวจสอบความผิดพลาด

1.2.7 ทำการแก้ไขและปรับปรุงส่วนที่ทำให้เกิดการผิดพลาดของการทำงานให้ถูกต้องแม่นยำ

1.3 ประโยชน์ที่ได้รับ

ประโยชน์ที่ได้รับจากเครื่องส่งไฟล์ไบนารีระยะไกล มีดังนี้

1.3.1 ทำให้สามารถทำการถ่ายโอนข้อมูลที่มีลักษณะเป็นไบนารีไฟล์ระหว่างอุปกรณ์ทางอิเล็กทรอนิกส์กับคอมพิวเตอร์ได้

1.3.2 เพิ่มประสิทธิภาพในการถ่ายโอนข้อมูลระหว่างอุปกรณ์ทางอิเล็กทรอนิกส์กับคอมพิวเตอร์

1.3.3 เพิ่มความสะดวกในการเก็บข้อมูลจากสถานที่ต่าง ๆ เพื่อที่จะนำมาใช้ในการวิเคราะห์และพัฒนาอุปกรณ์อื่น ๆ

1.3.4 ลดค่าใช้จ่ายและเวลาในการจัดเก็บข้อมูลต่าง ๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 เทคนิคการสื่อสารข้อมูลดิจิตอล

สำหรับอุปกรณ์คอมพิวเตอร์ 2 เครื่องที่เชื่อมโยงกันด้วยสายสื่อสารเพื่อแลกเปลี่ยนข้อมูลกัน โดยปกติแล้วข้อมูลจะถูกส่งผ่านทีละ 1 บิตต่อครั้งผ่านสายสื่อสาร แต่ละบิตของข้อมูลที่ถูกส่งผ่านไปอย่างต่อเนื่องกัน อาจจะส่งไปในลักษณะแบบ อนุกรม (Serial) หรือแบบ ขนาน (Parallel) เพื่อให้อัตราการส่งข้อมูลเพิ่มมากขึ้น เวลา (หมายถึงอัตราช่วงเวลา หรือช่องว่าง) ของบิตเหล่านี้จะต้องเท่ากันทั้งทางด้านเครื่องส่งและเครื่องรับ เทคนิคที่ทำให้เวลาที่ปลายทางทั้งสองด้านพร้อมกันมี 2 วิธีคือวิธีแบบ อะซิงโครนัส (Asynchronization) และวิธีแบบ ซิงโครนัส (Synchronization)

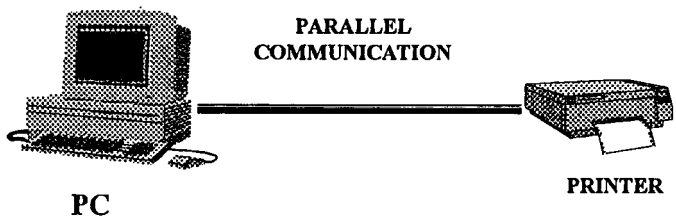
2.1.1 ความแตกต่างของการสื่อสารข้อมูลแบบขนานกับแบบอนุกรม

ในการสื่อสารข้อมูลโดยผ่านสายสื่อสารทำได้ 2 วิธีคือ การสื่อสารข้อมูลแบบอนุกรมหรือเรียงลำดับ (Serial) และการสื่อสารข้อมูลแบบขนาน (Parallel) การสื่อสารข้อมูลดิจิตอลทั้ง 2 แบบมีความแตกต่างกันอยู่หลายประการดังนี้

การส่งบิตต่างกัน

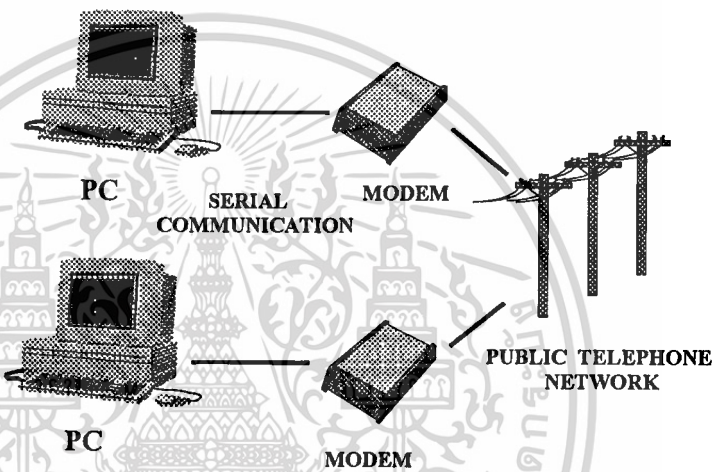
คำว่า "อนุกรม" หมายถึง หนึ่งต่อหนึ่งเรียงลำดับกันไป ดังนั้นการส่งข้อมูล (หรือบิต) แบบอนุกรมจึงเป็นการส่งข้อมูลทีละ 1 บิตต่อครั้งผ่านทางสายการสื่อสาร

แต่การส่งข้อมูลแบบขนานจะเป็นการส่งข้อมูลเป็นชุดของบิตเรียกว่า ไบต์ (Byte) จำนวนบิตในแต่ละไบต์ขึ้นอยู่กับจำนวนสายข้อมูล (Data Line) เช่น ถ้าสายสื่อสารมีสายข้อมูล 8 สาย ดังนั้นในการส่งข้อมูลทีละ 1 บิตต่อครั้งต่อสายสื่อสาร จะได้จำนวนข้อมูลทั้งหมดเท่ากับ 8 บิต หรือ 1 ไบต์ โดยมีการแปลงรหัส (Code) ของบิตแทนตัวอักขระ (Character) ก่อนทำการส่งออกไป



สายขนาน 8 สายข้อมูล (8 บิต)

ก. การสื่อสารข้อมูลแบบขนาน



ส่งทีละ 1 บิตเรียงตามกันในสายอนุกรม

ข. การสื่อสารข้อมูลแบบอนุกรม

รูปที่ 2.1 แสดงลักษณะการสื่อสารข้อมูลแบบอนุกรมและแบบขนาน

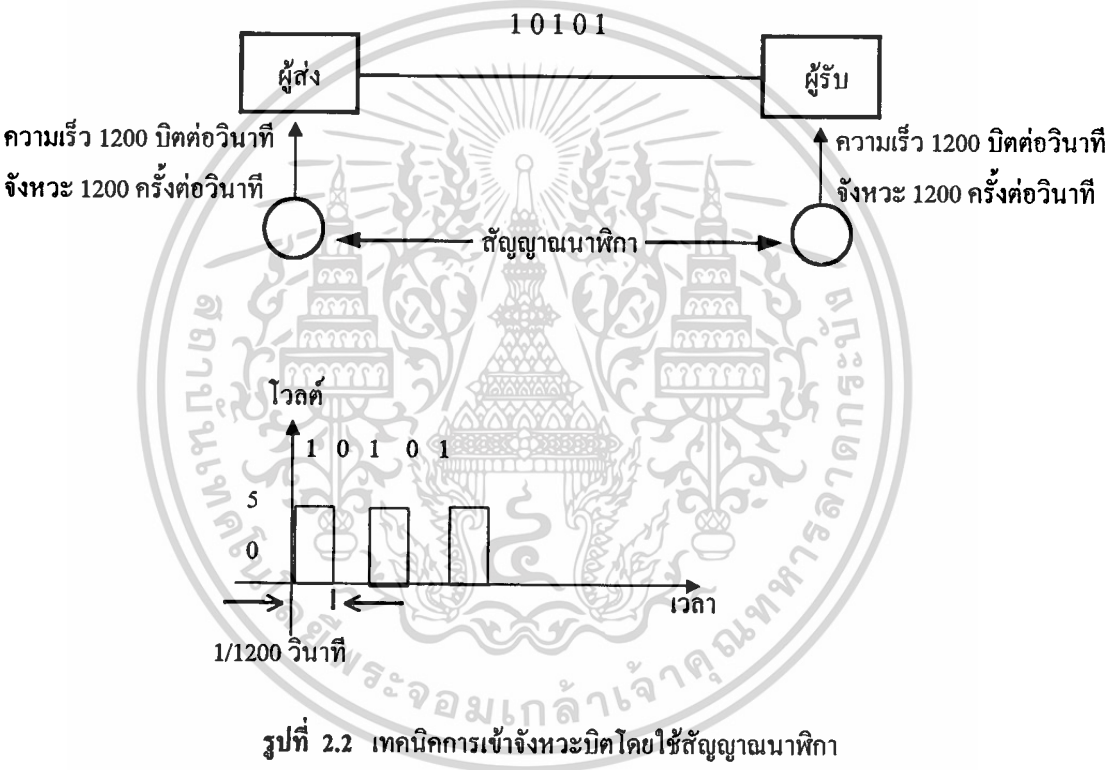
ระยะทางไกลต่างกัน

เพราะว่าการส่งข้อมูลแบบอนุกรมเป็นการส่งข้อมูลทีละ 1 บิต ความผิดพลาดจึงเป็นไปได้น้อยมาก จึงเหมาะสำหรับการส่งข้อมูลในระยะทางไกล ๆ เช่น ทำการส่งจากไมโครคอมพิวเตอร์ไปยังเทอร์มินัลที่อยู่คนละชั้น หรือคนละอาคาร หรือไกลกว่านั้น

ส่วนการส่งข้อมูลแบบขนานนั้น แม้จะสามารถส่งข้อมูลได้เป็นจำนวนมากแต่โอกาสผิดพลาดก็สามารถเกิดขึ้นได้มากด้วยเช่นกัน โดยเฉพาะในการส่งข้อมูลระยะทางไกล ๆ สัญญาณข้อมูลอาจจะจางหายหรือผิดเพี้ยนไปกับความต้านทานของสายส่งได้ ดังนั้นจึงเหมาะกับการส่งข้อมูลในระยะใกล้คือน้อยกว่า 100 ฟุต เช่น ระหว่างเครื่องคอมพิวเตอร์กับเครื่องพิมพ์ เป็นต้น

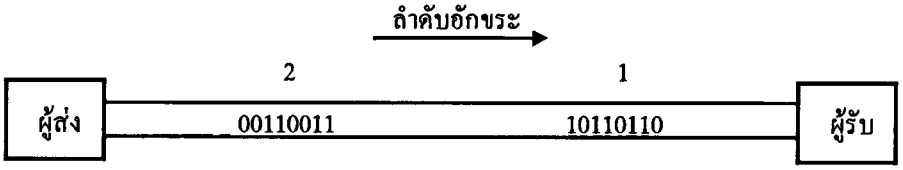
การเข้าจังหวะ (ซิงโครนัส) ต่างกัน

● การเข้าจังหวะบิต (Bit Synchronization) ในการส่งข้อมูลแบบอนุกรมข้อมูลจะถูกส่งทีละ 1 บิตเรียงลำดับกันไป ลำดับของการส่งและการรับข้อมูลจะต้องตรงกัน นั่นคือผู้ส่งและผู้รับจะต้องส่งและรับข้อมูลด้วยความถี่เดียวกัน และด้วยอัตราความเร็วเท่ากัน เราเรียกว่า การเข้าจังหวะบิต เทคนิคในการทำให้ลำดับของบิตทั้ง 2 ด้านตรงกันคือการใช้สัญญาณนาฬิกา (Clock) กำหนดจังหวะของเวลาบิตเริ่มต้นและบิตจบ หรือทั้งอักขระให้พร้อมกันทั้งทางผู้ส่งและผู้รับ

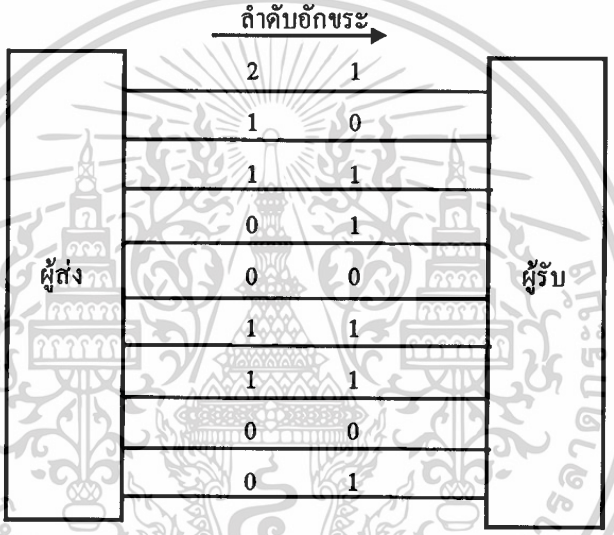


รูปที่ 2.2 เทคนิคการเข้าจังหวะบิตโดยใช้สัญญาณนาฬิกา

● การเข้าจังหวะอักขระ (Character Synchronization) ในการส่งข้อมูลแบบอนุกรมนั้นผู้รับจะต้องจัดลำดับของบิตที่รับมารวมเป็นตัวอักขระ ตำแหน่งของแต่ละบิตในตัวอักขระจะต้องถูกต้อง แต่สำหรับในการส่งข้อมูลแบบขนาน เนื่องจากข้อมูลจะถูกส่งมาทีละอักขระอยู่แล้ว ผู้รับเพียงแต่ตรวจสอบว่าบิตใดเป็นบิตเริ่มต้น และบิตใดเป็นบิตสุดท้ายของแต่ละตัวอักขระ วิธีการที่จะทำให้รู้ว่าบิตใดอยู่ตำแหน่งใดของตัวอักขระ ซึ่งก็ต้องอาศัยหลักการส่งข้อมูล แบบซิงโครนัส และแบบอะซิงโครนัส (Synchronous and Asynchronous Transmission) ซึ่งจะกล่าวถึงในหัวข้อต่อไป



ก. การส่งสัญญาณข้อมูลแบบอนุกรม (ทีละบิต)



ข. การส่งสัญญาณข้อมูลแบบขนาน (ทีละอักขระ)

รูปที่ 2.3 แสดงการส่งสัญญาณข้อมูลแบบอนุกรมและแบบขนาน

ค่าใช้จ่ายต่างกัน

ในการส่งข้อมูลแบบอนุกรมต้องการเพียง 1 ช่องทางสื่อสาร (Channel) ในสายส่งสัญญาณ และความเร็วในการส่งข้อมูลจัดอยู่ในขั้นความเร็วต่ำคือ 300-1,200 บิตต่อวินาที สายส่งสัญญาณที่ใช้จึงสามารถเลือกใช้แบบราคาถูกได้ เช่น สายเกลียวคู่ เป็นต้น

ส่วนในการส่งข้อมูลแบบขนานนั้นต้องการช่องทางสื่อสารอย่างน้อย 8 ช่องทาง (1ไบต์ ประกอบด้วย 8 บิต) ดังนั้นถ้าจะส่งข้อมูลด้วยความเร็วสูง (มากกว่า 9,600 บิตต่อวินาที) จะต้องใช้สายส่งคุณภาพสูง เช่น สายโคแอก ดังนั้นราคาของสายส่งสัญญาณจึงต้องสูงกว่าการส่งข้อมูลแบบอนุกรม

2.1.2 ลักษณะของการส่งข้อมูลแบบอะซิงโครนัสและแบบซิงโครนัส

ในการส่งและรับข่าวสาร สิ่งที่ทั้งเครื่องส่งและเครื่องรับจะต้องมีเหมือนกัน ได้แก่ อัตราเร็วของการส่ง ช่วงเวลาของสัญญาณ และช่องว่างระหว่างบิต

ก่อนจะทำการส่งสัญญาณข้อมูลดิจิตอล เราจะต้องทำการแปลงรหัสของตัวอักขระของข่าวสารที่จะส่งเป็นรหัสของบิตเสียก่อน แล้วจึงส่งแต่ละบิตออกไปยังผู้รับ จากนั้นผู้รับจะทำความรู้จักกับกลุ่มของบิตที่แทนตัวอักขระเหล่านั้นว่าบิตใดเป็นบิตแรก และบิตใดเป็นบิตสุดท้ายของอักขระ แต่ถ้าหากว่าจังหวะของเวลาในการอ่านและการส่งรับข้อมูลของเครื่องส่งและเครื่องรับต่างกัน ก็จะทำให้เกิดความผิดพลาดในการส่ง-รับข้อมูลได้การเข้าจังหวะหรือการซิงโครนัส (Synchronization) ของเวลาระหว่างผู้ส่ง-ผู้รับจำเป็นอย่างยิ่งที่จะต้องเหมือนกันและเท่ากัน

ในการเข้าจังหวะสามารถแบ่งได้เป็น 3 ระดับคือ

1. การเข้าจังหวะบิต หรือ การซิงโครนัสบิต (Bit Synchronization) เพื่อกำหนดจุดเริ่มต้นและจุดสิ้นสุดของการส่งข้อมูลของแต่ละบิต

2. การเข้าจังหวะอักขระ หรือ การซิงโครนัสอักขระ (Character Synchronization) เพื่อกำหนดจุดเริ่มต้น และจุดสิ้นสุดการส่งข้อมูลของแต่ละตัวอักขระ

3. การเข้าจังหวะบล็อก หรือ การซิงโครนัสบล็อก (Block Synchronization) เพื่อกำหนดตำแหน่งจุดเริ่มต้น และจุดสิ้นสุดของจำนวนข้อมูลขนาดใหญ่ หรือบล็อกข้อมูล

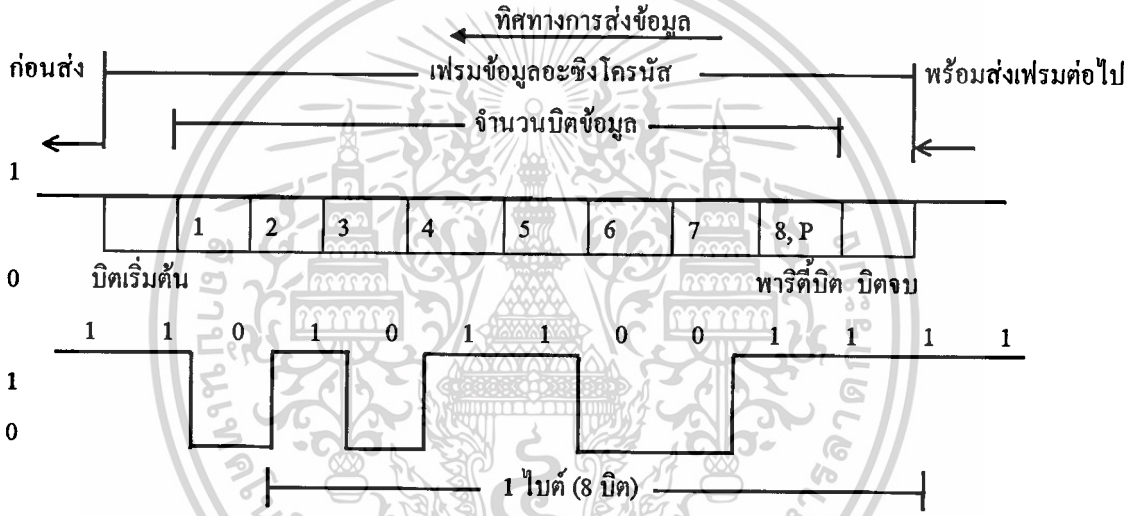
เทคนิคในการสื่อสารข้อมูลแบบอนุกรมที่ใช้ในการซิงโครนัสเพื่อเข้าจังหวะลำดับของข้อมูล และเพื่อควบคุมการสื่อสารข้อมูลระหว่างผู้ส่งและผู้รับข้อมูลคือวิธีอะซิงโครนัส (Asynchronous) บางครั้งเรียกว่า วิธีเริ่มและหยุด (Start/Stop) และอีกวิธีหนึ่งคือวิธีซิงโครนัส (Synchronous)

เทคนิคการส่งข้อมูลแบบอะซิงโครนัสและแบบซิงโครนัสนี้ จะเกี่ยวเนื่องกับเรื่องของการใช้งานโปรโตคอล (Protocol) ที่ใช้ควบคุมการสื่อสารข้อมูลระหว่างอุปกรณ์คอมพิวเตอร์ 2 เครื่อง หรือทั้งเครือข่าย โปรโตคอลที่ใช้ในการส่งข้อมูลแบบอะซิงโครนัสที่รู้จักกันดีคือโปรโตคอลเทเลไทป์ หรือ TTY (Teletype Protocol) ส่วนโปรโตคอลที่ใช้ในการส่งข้อมูลแบบซิงโครนัสที่รู้จักกันดีได้แก่ BSC หรือไบซิงก์ (Binary Synchronous Communications หรือ BISYNC) SDLC (Synchronous Data Link Control) และ HDLC (High-level Data Link Control) เป็นต้น

- การส่งข้อมูลแบบอะซิงโครนัส

ในการส่งข้อมูลดิจิตอลแบบอะซิงโครนัส กลุ่มของบิตจำนวน 5 บิต (รหัสโบดอต) หรือ 8 บิต (รหัสแอสกี) จะแทนตัวอักขระที่ถูกส่งออกไปเป็นเฟรม (Frame) บางครั้งเราเรียกว่าเป็นการส่งข้อมูลแบบ "Start/Stop"

การส่งข้อมูลจะส่งข้อมูลทีละอักขระโดยที่ช่วงเวลาระหว่างอักขระจะเป็นเท่าไรก็ได้ ดังนั้นตัวเครื่องรับจะต้องทำการตรวจสอบว่า บิตใดเป็นบิตเริ่มต้นของอักขระและบิตใดเป็นบิตสุดท้ายของอักขระ



รูปที่ 2.4 แสดงการส่งสัญญาณข้อมูลดิจิตอลเป็นเฟรมโดยวิธีอะซิงโครนัส

ในการส่งอักขระแต่ละตัวจะประกอบด้วยบิตเริ่มต้น (1 บิต) + ข้อมูล (8 หรือ 7 บิต) + 1 พาริตี้บิต (แต่อาจจะไม่ใช้ก็ได้) + บิตจบ (1 บิต) รวมเป็น 10 บิต คิดเป็น 1 เฟรม

ขั้นตอนการส่งข้อมูล

ขั้นตอนของการส่งข้อมูลดิจิตอลโดยวิธีแบบอะซิงโครนัสมี ดังนี้

1. ก่อนจะเริ่มทำการส่งข้อมูล สัญญาณจะมีค่าเป็น "1" ตลอดเวลา

2. เมื่อเริ่มส่งข้อมูลสัญญาณของบิตแรกจะเปลี่ยนเป็น "0" นั่นคือบิตเริ่มต้น เครื่องรับจะเริ่มสัญญาณนาฬิกาของตัวเอง เมื่อเวลาผ่านไป 1/2 บิต ถ้าสัญญาณยังคงเป็น "0" ต่อไป อีก 1/2 บิต ต่อมาก็จะเป็นการเริ่มของสัญญาณข้อมูล แต่ถ้าสัญญาณกลับไปเป็น "1" อีก ก็แสดงว่าเกิดความผิดพลาดอันเกิดจากสัญญาณรบกวนในสายส่งและยังไม่มีสัญญาณข้อมูลใด ๆ ส่งมายังปลายทาง

3. หลังจากได้เริ่มบิตเริ่มต้นแล้ว ผู้ส่งจะเริ่มส่งรหัสบิตของอักขระ อาจจะเป็น 5 บิต หรือ 8 บิต หรือ 7 บิต แล้วตามด้วยพาริตี้บิต (อาจจะใช้หรือไม่ใช้ก็ได้) ดังรูป 2.4 เป็นการส่งสัญญาณข้อมูลขนาด 8 บิต สำหรับ 1 อักขระ โดยเป็นสัญญาณข้อมูล 7 บิต บิตที่ 8 เป็นพาริตี้บิตคี่ (Odd) จากนั้นสัญญาณจะเป็น "1" ไปอีก 1 บิต ซึ่งถือว่าเป็นบิตจบ สัญญาณจะเป็น "1" ต่อไปเรื่อย ๆ จนกว่าจะเริ่มมีการส่งสัญญาณข้อมูลในเฟรมต่อไป

การส่งสัญญาณข้อมูลแบบอะซิงโครนัสมีใช้กันอย่างกว้างขวาง เพราะเทคนิคการส่งข้อมูลไม่ยาก รวมทั้งสายส่งสัญญาณก็มีราคาถูก ส่วนใหญ่ใช้ในการส่ง-รับข้อมูลกันระหว่างเครื่องคอมพิวเตอร์บุคคล (PC) กับศูนย์บริการข้อมูลที่อยู่ไกลออกไป เช่น โฮสต์คอมพิวเตอร์ของตลาดหลักทรัพย์ หรือ ระบบธนาคาร

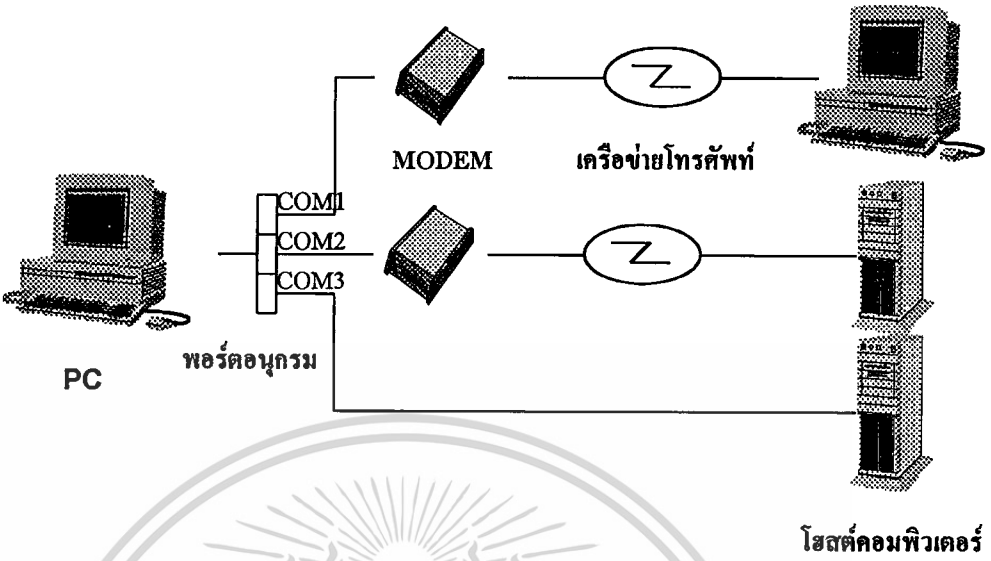
เนื่องจากการสื่อสารข้อมูลดิจิตอลแบบอะซิงโครนัสมีความเร็วในการส่งข้อมูลต่ำ จึงมักใช้กับเทอร์มินัลที่ไม่มีบัฟเฟอร์ นอกจากนั้นเวลาประมาณ 20 เปอร์เซ็นต์ของการส่งอักขระแต่ละตัวจะสูญเสียไปกับบิตเริ่มต้น บิตจบ และพาริตี้บิต

ในการส่งข้อมูลที่เป็นบล็อกโดยในแต่ละบล็อกมีอักขระมากกว่า 1 อักขระ มักจะส่งโดยวิธีซิงโครนัสซึ่งจะกล่าวถึงต่อไป

การตรวจสอบความผิดพลาด

การตรวจสอบความผิดพลาดของข้อมูลในการสื่อสารข้อมูลดิจิตอลแบบอะซิงโครนัส เราสามารถตรวจสอบได้จากพาริตี้บิต ซึ่งแบ่งเป็นพาริตี้คี่ (Odd) และพาริตี้คู่ (Even) เช่น ถ้าเราส่งข้อมูลเป็นพาริตี้คู่ นั่นคือเมื่อรวมบิตของอักขระทั้งหมดที่เป็น "1" กับพาริตี้บิต (ซึ่งอาจจะเป็น "0" หรือ "1") แล้วจะได้เป็นจำนวนคู่ ซึ่งเมื่อเครื่องรับได้รับเฟรมข้อมูลไปแล้ว ถ้ารวมบิตของ "1" ทั้งหมดได้เป็นจำนวนเลขคี่แล้วแสดงว่าข้อมูลที่รับมามีความผิดพลาดเกิดขึ้น แต่ในกรณีที่มีการผิดพลาดเกิดขึ้น แต่นับบิต "1" ทั้งหมดได้จำนวนคู่เช่นกัน เราก็ไม่สามารถตรวจความผิดพลาดได้

ตัวอย่างจากรูป 2.4 บิตข้อมูลมี 7 บิตคือ 1011001 และกำหนดพาริตี้บิตเป็นพาริตี้คู่ ดังนั้นค่าของพาริตี้บิตจึงเป็น "1" เพื่อให้จำนวนบิต "1" ทั้งหมดใน 8 บิตรวมกันเป็นเลขจำนวนคี่ วิธีตรวจสอบความผิดพลาดในการส่งข้อมูลแบบอะซิงโครนัสดังที่ได้กล่าวมาเราเรียกว่า การตรวจสอบพาริตี้ (Parity Check)



รูปที่ 2.5 แสดงตัวอย่างการสื่อสารข้อมูลดิจิตอลแบบอะซิงโครนัสจากพอร์ตอนุกรมของเครื่อง PC

- **การส่งข้อมูลแบบซิงโครนัส**

สำหรับเทคนิคการสื่อสารข้อมูลคอมพิวเตอร์ที่ให้ประสิทธิภาพดีกว่าการสื่อสารข้อมูลแบบ อะซิงโครนัส คือ การสื่อสารข้อมูลแบบซิงโครนัส ลักษณะของข้อมูลที่ถูกส่งผ่านสายสื่อสารจะถูกส่งไปเป็นบล็อกของอักขระหรือกลุ่มบิต (Block of Characters or Bits) โดยไม่จำเป็นต้องมีบิตเริ่มต้นและบิตจบเช่นเดียวกับการสื่อสารข้อมูลแบบอะซิงโครนัส

การพิจารณาเวลาเริ่มต้นและเวลาสิ้นสุดของบล็อกข้อมูลแต่ละบล็อก เราพิจารณาจากกลุ่มบิตส่วนหัว (Header) และกลุ่มบิตส่วนท้าย (Trailer) ของบล็อกข้อมูล กลุ่มบิตพิเศษทั้ง 2 กลุ่มนี้เป็นกลุ่มบิตแทนข่าวสารการควบคุมการส่งข้อมูลมากกว่าที่จะเป็นบิตข้อมูล ข้อมูลที่รวมเข้ากับข่าวสารการควบคุมการส่งข้อมูลนี้เราเรียกว่า เฟรม (Frame)

รูปแบบของเฟรมนั้นขึ้นอยู่กับว่าการส่งข้อมูลนั้นจะเป็นแบบซิงโครนัสอักขระ (Character Synchronization) หรือแบบซิงโครนัสบิต (Bit Synchronization)

การส่งข้อมูลแบบซิงโครนัสอักขระ

การส่งข้อมูลแบบซิงโครนัสอักขระ เฟรมที่ใช้ในการส่งข้อมูลจะมีรูปแบบเป็นเฟรมอักขระ (Character Oriented Frame) คืออักขระในบล็อกข้อมูลจะถูกเรียงลำดับกัน ส่วนบิตพิเศษแทนการควบคุมก็จะอยู่ในรูปของอักขระเช่นกัน เฟรมข้อมูลจะเริ่มต้นด้วยอักขระซิงโครนัส (Synchronization Character) หรือเรียกสั้น ๆ ว่า อักขระซิงก์ (SYN) ซึ่งอาจจะมีมากกว่า 1 อักขระ

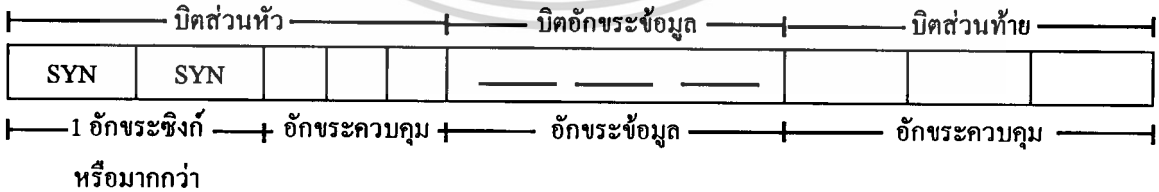
(ดูรูปที่ 2.6 ประกอบ) อักขระซิงก์จะมีรูปแบบของบิตที่แน่นอน เพื่อให้ทางผู้รับสามารถรู้ได้ทันทีว่าเมื่อมีอักขระซิงก์เข้ามาที่เครื่องรับแสดงว่าจะเป็นการเริ่มต้นบล็อกข้อมูลแล้ว ทางผู้รับก็จะเตรียมพร้อมรับบล็อกข้อมูลที่จะตามมา ต่อจากอักขระซิงก์ก็จะเป็นอักขระควบคุมซึ่งจะเป็นข่าวสาร เช่น บอกจุดเริ่มต้นของบล็อกข้อมูล จำนวนอักขระในบล็อกข้อมูล ความยาวของบล็อกข้อมูล หรือตำแหน่งปลายทางของข้อมูล เป็นต้น จากนั้นจึงตามด้วยบล็อกข้อมูล

จำนวนสูงสุดของบิตแทนอักขระใน 1 เฟรม จะเป็นจำนวนทวีคูณกับจำนวนบิตข้อมูลใน 1 อักขระคือถ้าใน 1 อักขระมี 8 บิต (7 บิตข้อมูล + 1 พาริตี้บิต ดังตัวอย่างในรูปที่ 2.6 ข.) ดังนั้น จำนวนของบิตของเฟรมจะเท่ากับ 2^7 หรือ 128 บิต

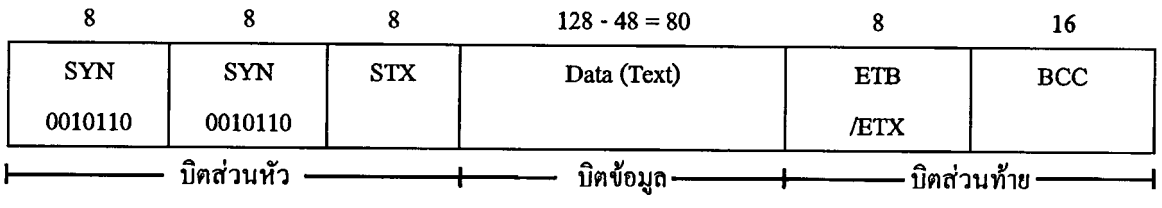
เราจะเห็นว่าบิตส่วนหัวซึ่งประกอบด้วยอักขระซิงก์ (SYN) และอักขระควบคุม จะมีไว้สำหรับบอกแก่เครื่องรับว่าบัดนี้ข้อมูลกำลังจะมาถึง เครื่องรับจะเริ่มรับ (อ่าน) ข้อมูลเมื่ออักขระเริ่มต้นข้อมูลผ่านไป (STX) ในขณะเดียวกันเครื่องรับก็จะหารหัสที่ระบุจุดสิ้นสุดของข้อมูล (ETX หรือ ETB) หรือบิตส่วนท้ายนั่นเอง

ส่วนการใส่อักขระ SYN ที่ส่วนหัว 2 อักขระหรือมากกว่า ก็เพื่อป้องกันการผิดพลาดเมื่อผู้รับตรวจพบการเปลี่ยนแปลงในสัญญาณที่ผ่านเข้ามากับสายสื่อสารซึ่งมีลักษณะเหมือนกับอักขระ SYN ดังนั้นถ้าเครื่องรับตรวจพบอักขระ SYN ติดกัน 2 อักขระ แสดงว่าต่อไปจะเป็นการเริ่มต้นส่งข้อมูลที่แท้จริงแล้ว

ในการส่งข่าวสารแต่ละบล็อก จำนวนของบิตในแต่ละบล็อกไม่จำเป็นต้องเป็นจำนวนทวีคูณของจำนวนบิตของอักขระ อาจจะมีแค่อักขระส่วนหัวและอักขระส่วนท้ายโดยไม่มีอักขระข้อมูลเลยก็ได้



ก. รูปแบบของเฟรมอักขระ



ข. เฟรมอักขระ 8 บิต

SYN = Synchronization Character, อักขระซิงก์

STX = Start of Text, เริ่มต้นส่งข้อความ (ข้อมูล)

ETX = End of Text, สิ้นสุดข้อความ (ข้อมูล)

ETB = End of Transmission Block, สิ้นสุดบล็อกข้อมูล

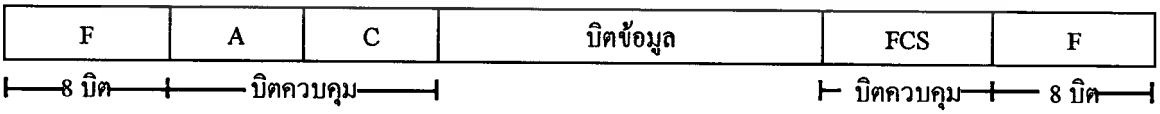
BCC = Block Check Character, สำหรับตรวจสอบความผิดพลาดในบล็อก หรือเฟรมข้อมูล

รูปที่ 2.6 แสดงบล็อกหรือเฟรมอักขระของการส่งข้อมูลแบบซิงโครนัสอักขระ

การส่งข้อมูลแบบซิงโครนัสบิต

แทนที่ข้อมูลจะถูกส่งไปทีละอักขระเรียงลำดับกัน เช่น ในการส่งข้อมูลแบบซิงโครนัสอักขระ ในวิธีการส่งข้อมูลแบบซิงโครนัสบิต ข้อมูลจะถูกส่งออกไปทีละบิตเรียงลำดับกันไป ทั้งบิตข้อมูลและบิตการควบคุมโดยไม่จำเป็นต้องแปลงมาอยู่ในรูปของอักขระ การเริ่มต้นบล็อก หรือเฟรมข้อมูลจะเริ่มต้นด้วยกลุ่มบิตพิเศษขนาด 8 บิตซึ่งเรียกว่า แฟลก (Flag) และจะมีแฟลกรูปแบบเหมือนเดิมกับแฟลกเริ่มต้นเพิ่มในส่วนท้ายของบล็อก หรือเฟรมข้อมูลเป็นแฟลกจบ (ดูรูปที่ 2.7 ประกอบ)

เครื่องรับของผู้รับจะรอคอยสัญญาณดิจิตอล หรือกลุ่มบิตที่ผ่านเข้ามาที่มีรูปแบบเดียวกันกับแฟลกเพื่อเป็นสัญญาณให้เริ่มต้นเฟรมข้อมูล หลักจากแฟลกเริ่มต้นผ่านไปแล้วจะเป็นกลุ่มของบิตควบคุม (Control Field) ทำหน้าที่บอกความยาวของบล็อกข้อมูล ตำแหน่งปลายทางของผู้รับ วิธีการอ่านบิตข้อมูล และข่าวสารอื่น ๆ จากนั้นจึงจะเป็นบิตข้อมูลจริง ๆ ที่ต้องการส่งไปให้ผู้รับ จบบิตข้อมูลจะเป็นบิตควบคุมตรวจสอบความผิดพลาดของข้อมูล (Frame Check Sequence) และจบเฟรมด้วยแฟลกจบ



F = Flag, แฟลก

A = Address, ตำแหน่งปลายทางข้อมูล

C = Control, ส่วนควบคุมการส่งข้อมูล

FCS = Frame Check Sequence, ตรวจสอบความผิดพลาดของเฟรมข้อมูล

รูปที่ 2.7 แสดงเฟรมข้อมูลของการส่งข้อมูลแบบซิงโครนัสบิต

เปรียบเทียบประสิทธิภาพระหว่างการส่งข้อมูลแบบอะซิงโครนัสและแบบซิงโครนัส

ถ้ากล่าวถึงประสิทธิภาพในการส่งข้อมูลในจำนวนบิตที่เท่ากัน ใน 1 เฟรมข้อมูล การส่งข้อมูลแบบซิงโครนัสสามารถส่งบิตข้อมูลได้จำนวนมากกว่า เพราะไม่ได้ถูกจำกัดขนาดของบล็อกข้อมูลดังเช่นในการส่งข้อมูลแบบอะซิงโครนัส นอกจากนั้นในการส่งข้อมูลแบบอะซิงโครนัสจะใช้จำนวนบิตสำหรับบิตส่วนหัวและส่วนท้ายประมาณ 20 % ของบล็อกข้อมูล ในขณะที่การส่งข้อมูลแบบซิงโครนัสใช้บิตทั้งหมดสำหรับแฟลกและส่วนควบคุมโดยทั่วไปน้อยกว่า 100 บิต ดังนั้นถ้าในการส่งบล็อกข้อมูล 1,000 บิต โดยวิธีซิงโครนัสจะใช้จำนวนบิตทั้งหมดสำหรับแฟลกและส่วนควบคุมเพียง (48/1048) x 100 % = 4.6 % ของจำนวนบิตทั้งหมด อย่างไรก็ตามถ้าเกิดมีความผิดพลาดในการส่งข้อมูลก็ต้องส่งข้อมูลกันใหม่ทั้งบล็อก ซึ่งวิธีการแบบซิงโครนัสย่อมจะใช้เวลามากกว่า เพราะบล็อกข้อมูลมีขนาดใหญ่กว่า

สิ่งที่จำเป็นอย่างหนึ่งของการส่งข้อมูลแบบซิงโครนัส คือ หน่วยความจำชั่วคราว หรือบัฟเฟอร์ ทั้งในเครื่องรับและเครื่องส่ง เมื่อหน่วยความจำของบัฟเฟอร์เก็บข้อมูลที่ต้องการจะส่งจนครบแล้ว หรือบัฟเฟอร์เต็มแล้ว เครื่องส่งก็จะทำการส่งข้อมูลออกไปทั้งหมดด้วยความเร็วของสายสื่อสาร ทำให้การส่งข้อมูลแบบซิงโครนัสสามารถใช้สายสื่อสารได้อย่างเต็มประสิทธิภาพ

นอกจากนั้นการส่งข้อมูลแบบซิงโครนัสยังให้ประสิทธิภาพในการส่งและรับได้อย่างถูกต้อง ถูกตำแหน่งและรวดเร็วกว่าการแบบอะซิงโครนัส ส่วนการตรวจสอบความผิดพลาดก็สามารถตรวจสอบได้จากพาริตี้บิตเช่นเดียวกับแบบอะซิงโครนัส ขึ้นอยู่กับว่าได้กำหนดวิธีการอ่านข้อมูลไว้อย่างไรในส่วนของบิตควบคุม

2.1.3 ความผิดพลาดในการส่ง-รับข้อมูล (Data Transmission Error)

เมื่อเฟรมของข้อมูลได้ส่งออกไปจากเครื่องส่งแล้ว โอกาสที่เป็นไปได้ที่เครื่องรับจะได้รับเฟรมข้อมูลนั้นมีอยู่ 3 กรณีคือ

1. ได้รับข้อมูลถูกต้องทั้งเฟรม
2. ได้รับข้อมูลที่มีบางบิตของสัญญาณข้อมูลผิดพลาดและตรวจจับไม่ได้
3. ได้รับข้อมูลที่มีบางบิตของสัญญาณข้อมูลผิดพลาดและตรวจจับได้

สาเหตุของการเกิดความผิดพลาด ได้แก่ สัญญาณรบกวน (Noise) และลักษณะของช่องทาง หรือสายสื่อสาร สัญญาณรบกวนที่เกิดขึ้นที่สำคัญ ได้แก่ สัญญาณรบกวนแบบพัลส์ (Noise

Pulse) และสัญญาณรบกวนแบบเบิรสต์ (Noise Burst) หรือกลุ่มสัญญาณรบกวน ซึ่งมักจะเกิดในช่วงเวลาประมาณ 1/100 วินาที

สัญญาณรบกวนที่เกิดขึ้นอาจจะเกิดขึ้นจากการเปิด-ปิดสวิตช์ไฟฟ้า ฝนตกฟ้าคะนอง ฯลฯ เหตุการณ์เหล่านี้เป็นลักษณะของสัญญาณรบกวนแบบเบิรสต์ ส่วนการผิดเพี้ยนของสัญญาณซึ่งอาจจะเกิดจากการเปลี่ยนแปลงรูปแบบการส่งของสัญญาณ เช่น จากอนาล็อกเป็นดิจิตอล หรือจากดิจิตอลเป็นอนาล็อก และการสะท้อนกลับของสัญญาณ (Echo) เป็นสัญญาณรบกวนแบบพัลส์ก็เช่นกัน

โดยมากแล้วสัญญาณรบกวนที่เกิดขึ้นระหว่างการส่งและรับข้อมูลมักจะเป็นสัญญาณรบกวนแบบเป็นกลุ่มหรือเบิรสต์มากกว่าแบบพัลส์

อัตราความผิดพลาด (Error Rate)

การวัดอัตราความผิดพลาดของข้อมูล เราจะนับจากจำนวนบิตที่เกิดความผิดพลาดไม่เป็นไปตามข้อมูลจริงต่อจำนวนบิตทั้งหมดในช่วงเวลา 1/100 วินาที ดังนั้นยิ่งมีการส่งข้อมูลด้วยอัตราเร็วมากเท่าใด ถ้ามีความผิดพลาดเกิดขึ้นอัตราความผิดพลาดก็จะยิ่งมีมากขึ้นเท่านั้น โดยทั่วไปอัตราความผิดพลาดที่ยอมรับได้จะอยู่ประมาณ 1 บิตต่อ 100,000 บิต จะสังเกตได้ว่าค่าอัตราความผิดพลาดเป็นค่าทางสถิติเท่านั้น

ถ้าความผิดพลาดเกิดขึ้นเฉพาะทีละบิต (สัญญาณรบกวนแบบพัลส์) และข้อมูลที่ถูกส่งออกไปส่งเป็นบล็อกข้อมูลจะส่งด้วยอัตราเร็วสูง อัตราการเกิดความผิดพลาดย่อมจะสูง และความผิดพลาดสามารถเกิดขึ้นได้ทุกบล็อกข้อมูล

แต่ถ้าความผิดพลาดที่เกิดขึ้นนั้นเกิดขึ้นกับกลุ่มข้อมูล (สัญญาณรบกวนแบบเบิรสต์) ข้อมูลบางบล็อกอาจจะไม่มีความผิดพลาดเลยก็ได้ ดังนั้นถ้าในการส่งและรับข้อมูลที่มีอัตราความผิดพลาดสูง การส่งและรับข้อมูลเป็นบล็อกเล็ก ๆ จะทำให้ถูกรบกวนได้น้อยกว่า แต่ความเร็วหรือประสิทธิภาพในการส่งและรับข้อมูลก็จะลดลงด้วยเช่นกัน

การแก้ไขสายสื่อสาร

สาเหตุหนึ่งที่ทำให้เกิดมีความผิดพลาดขึ้นกับข้อมูลคือปัญหาเรื่องสายสื่อสาร ในการแก้ไขเราสามารถทำได้ดังนี้

1. *การปรับแต่งสายสื่อสาร* ในกรณีที่สายสื่อสารที่ใช้ส่ง-รับข้อมูลมีคุณภาพไม่ค่อยดี เราสามารถติดตั้งอุปกรณ์อิเล็กทรอนิกส์เข้ากับสายสื่อสาร เพื่อลดการรบกวนของสัญญาณ หรือลด

การสะท้อนของสัญญาณ การปรับแต่งสายสื่อสารทำได้เฉพาะกรณีของการเชื่อมโยงแบบจุดต่อจุดเท่านั้น

2. *การใช้เครื่องปรับเท่า* หรือ Equalizer เป็นการปรับแต่งคุณสมบัติของช่องทางการสื่อสาร หรือสายสื่อสารเช่นกัน โดยมากจะมีอยู่ในโมเด็ม โดยโมเด็มจะเป็นตัวรับสัญญาณข้อมูลเข้ามา จากนั้นเครื่องปรับเท่าจะทำงานโดยอัตโนมัติในการปรับแต่งสัญญาณให้ได้รูปร่างดีที่สุด

การแก้ความผิดพลาดของบิต (Bit Error Correction)

ในกรณีที่ความผิดพลาดเกิดขึ้นกับบิตข้อมูลโดยตรง การแก้ไขสามารถทำได้โดยอาศัยเทคนิคการส่งข้อมูลเพิ่มเป็นข้อมูลซ้ำซ้อน (Redundant) เข้าไปในข้อมูลด้วย เพื่อให้เครื่องรับสามารถตรวจจับ และแก้ไขความผิดพลาดที่เกิดขึ้นกับบิตข้อมูล โดยมีวัตถุประสงค์ 2 ประการคือ

1. เพื่อให้ผู้รับสามารถรู้และแก้ไขข้อผิดพลาดได้ถ้าหากมีความผิดพลาดเกิดขึ้น หรือ

2. เพื่อให้ผู้รับสามารถรู้ได้เฉพาะความผิดพลาดที่เกิดขึ้นเท่านั้น แต่ไม่สามารถแก้ไขความผิดพลาดได้ ผู้รับต้องร้องขอให้ผู้ส่งทำการส่งข้อมูลมาใหม่

จำนวนบิตของข้อมูลซ้ำซ้อน หรือเรียกว่า บิตตรวจสอบ (Check Bits) นั้นคำนวณได้จากสูตร

$$M+R+1 < 2^R$$

โดยที่ M = จำนวนบิตของข้อมูล

R = จำนวนบิตตรวจสอบที่น้อยที่สุด

เช่น สมมติว่าจะส่งข้อมูลเป็น ASCII Code ขนาด 7 บิต ถ้าจะให้มีการตรวจสอบ และแก้ไขความผิดพลาดจะต้องมีบิตตรวจสอบเท่ากับ 4 บิต นั่นคือจะต้องส่งข้อมูลเท่ากับ 12 บิตต่อ 1 อักขระ ($7+4+1 < 2^4$) รหัสที่ใช้ในการกำหนดบิตตรวจสอบโดยทั่วไป ได้แก่ Hamming Code ซึ่งจะไม่ขอกล่าวถึงวิธีการกำหนดในที่นี้

การตรวจสอบความผิดพลาด (Error Detection)

ลักษณะของการตรวจสอบความผิดพลาดเป็นการส่งข้อมูลซ้ำซ้อนไปพร้อมกับข้อมูลจริงด้วยจำนวนแค่เพียงพอสำหรับการตรวจสอบได้ว่ามีความผิดพลาดเกิดขึ้นหรือไม่เท่านั้น ซึ่งถ้ามีก็จะทำการร้องขอกลับไปยังผู้ส่งให้ส่งข้อมูลมาใหม่

การทำงานแบบนี้ให้ประสิทธิภาพมากกว่าการแก้ไขความผิดพลาดเพราะจำนวนของข้อมูลซ้ำซ้อนจะน้อยกว่า และโดยทั่วไปถ้าอัตราการผิดพลาดไม่มากนัก การส่งข้อมูลให้ใหม่ก็จะไม่มากเช่นกัน

การตรวจสอบความผิดพลาดสามารถทำได้โดยการส่ง-รับข้อมูล 2 แบบคือ

1. การส่ง-รับข้อมูลทีละอักขระ หรือแบบ Manual มักใช้กับระบบ On-line

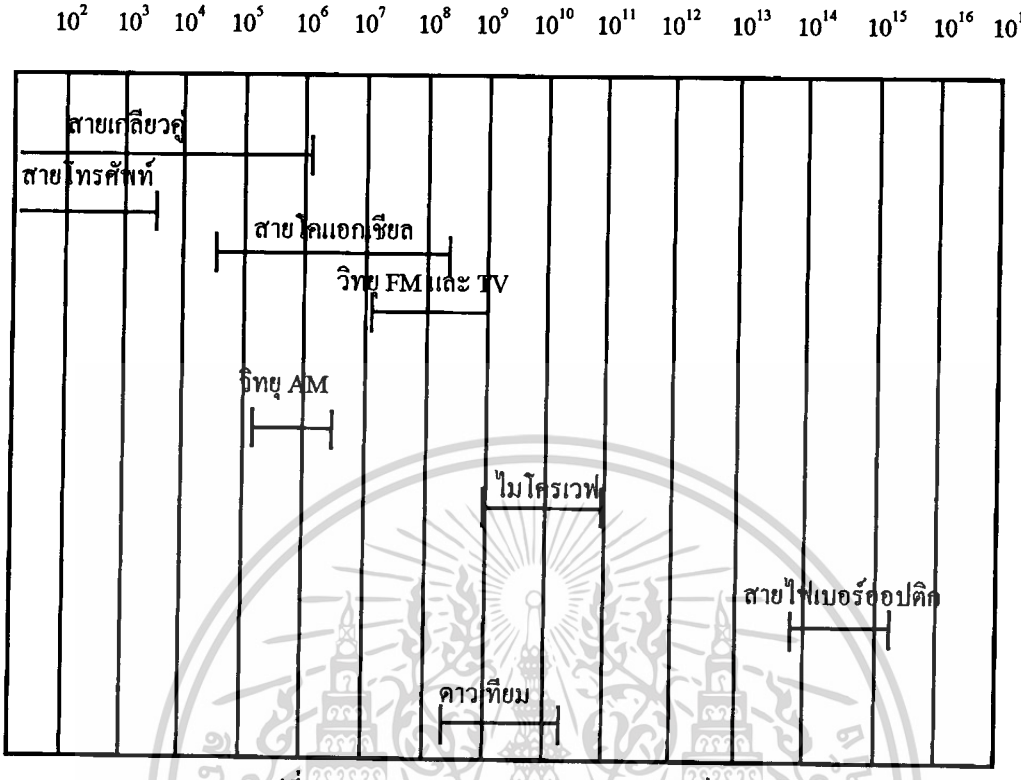
2. การส่ง-รับข้อมูลทีละบล็อก หรือแบบ Automatic โดยจะมีการสร้าง FCS (Frame Check Sequence) หรือ BCC (Block Check Character)

วิธีการตรวจสอบความผิดพลาดของข้อมูลที่นิยมใช้มี 3 แบบคือ Parity Bit Check ซึ่งได้กล่าวถึงไปแล้วในเรื่องการส่งข้อมูลแบบอะซิงโครนัส อีก 2 วิธี ได้แก่ Two-coordinate Parity Check และ Cyclic Redundancy Check

2.2 ชนิดของสื่อกลางในการสื่อสารข้อมูล

สื่อกลางแบ่งออกเป็น 2 จำพวกใหญ่ ๆ คือ สื่อกลางจำพวกกำหนดเส้นทางได้ เช่น สายเกลียวคู่ สายโคแอกเซียล และสายไฟเบอร์ออปติก และสื่อกลางจำพวกกำหนดเส้นทางไม่ได้ เช่น คลื่นไมโครเวฟ คลื่นวิทยุ และคลื่นดาวเทียม

การเลือกใช้ตัวกลางหรือสื่อกลางชนิดใดนั้น ขึ้นอยู่กับตัวแปรหลายอย่าง เช่น ราคาของตัวกลางหรือสื่อ ค่าบริการ อัตราเร็วในการส่งผ่านข้อมูล สถานที่ หรือภูมิประเทศ การบริการ และการควบคุม รวมทั้งเทคโนโลยีด้วย



รูปที่ 2.8 แสดงย่านสเปกตรัมของสัญญาณคลื่นแม่เหล็กไฟฟ้า

ตาราง 2.1 คุณลักษณะของย่านความถี่ของการสื่อสารประเภทไม่กำหนดเส้นทาง

ช่วงย่านความถี่	ชื่อ	ข้อมูลอนาล็อก		ข้อมูลดิจิตอล		การใช้งานทั่วไป
		การมอดูเลต	แบนด์วิดท์	การมอดูเลต	อัตราข้อมูล	
30-300 KHz	LF (Low Frequency)	-	-	ASK, FSK, MSK	0.1-100 bps	สื่อสารการบิน
300-3,000 KHz	MF (Medium Frequency)	AM	to 4 KHz	ASK, FSK, MSK	10-1,000 bps	วิทยุ AM
3-30 MHz	HF (High Frequency)	AM, SSB	to 4 KHz	ASK, FSK, MSK	10-3,000 bps	วิทยุคลื่นสั้น
30-300 MHz	VHF (Very High Frequency)	AM, SSB, FM	5 KHz-5 MHz	FSK, PSK	to 100 Kbps	วิทยุ FM ทีวี VHF
300-3,000MHz	UHF (Ultra High Frequency)	FM, SSB	to 20 MHz	PSK	to 10 Mbps	ทีวี UHF ไมโครเวฟ
3-30 GHz	SHF (Super High Frequency)	FM	to 500 MHz	PSK	to 100 Mbps	ไมโครเวฟ ดาวเทียม
30-300 GHz	EHF(Extremly High Frequency)	FM	to 1 GHz	PSK	to 750 Mbps	การเชื่อมโยงระยะสั้นๆ

2.2.1 สายเกลียวคู่ (Twisted Pair Cable)

สายเกลียวคู่เป็นสายที่มีราคาถูกที่สุด ประกอบด้วยสายทองแดง 2 เส้น แต่ละเส้นมีฉนวนหุ้มพันกันเป็นเกลียว สามารถลดการรบกวนจากสนามแม่เหล็กไฟฟ้าได้ แต่ไม่สามารถป้องกันการสูญเสียพลังงานจากการแผ่รังสีความร้อนในขณะที่มีสัญญาณส่งผ่านสาย สายเกลียวคู่ 1 คู่ จะแทนการสื่อสารได้ 1 ช่องทางสื่อสาร (Channel) ในการใช้งานจริง เช่น สายโทรศัพท์จะเป็นสายรวมที่ประกอบด้วยสายเกลียวคู่อยู่ภายในเป็นร้อยๆคู่ สายเกลียวคู่ 1 คู่จะมีขนาดประมาณ 0.016-0.036 นิ้ว

สายเกลียวคู่สามารถใช้ได้ทั้งการส่งสัญญาณข้อมูลแบบอนาล็อกและแบบดิจิตอล เนื่องจากสายเกลียวคู่จะมีการสูญเสียสัญญาณขณะส่งสัญญาณ จึงจำเป็นต้องมี "เครื่องขยาย" (Amplifier) สัญญาณ สำหรับการส่งสัญญาณข้อมูลแบบอนาล็อกในระยะทางไกล ๆ หรือทุก 5-6 กม. ส่วนการส่งสัญญาณข้อมูลแบบดิจิตอลต้องมี "เครื่องทบทวน" (Repeater) สัญญาณ ทุกๆระยะ 2-3 กม.

เพราะว่าแต่ละคู่ของสายเกลียวคู่จะแทนการทำงาน 1 ช่องทาง และสามารถมีแบนด์วิดท์ได้กว้างถึง 250 กิโลเฮิรตซ์ ดังนั้นในการส่งข้อมูลไปพร้อมกันหลาย ๆ ช่องทางจำเป็นต้องอาศัยหลักการมัลติเพล็กซ์สัญญาณเพื่อให้สัญญาณทั้งหมดสามารถส่งผ่านสายสื่อสารไปได้พร้อม ๆ กัน ในการมัลติเพล็กซ์แบบ FDM จะสามารถส่งสัญญาณข้อมูลได้ถึง 24 ช่องทาง ๆ ละ 74 กิโลเฮิรตซ์ ส่วนอัตราเร็วสูงสุดในการส่งข้อมูลดิจิตอลผ่านของสายเกลียวคู่สามารถมีได้ถึง 4 เมกะบิตต่อวินาที แต่ถ้าเป็นการส่งข้อมูลผ่านโมเด็ม จะส่งได้ด้วยอัตราเร็วสูงสุด 9,600 บิตต่อวินาที

2.2.2สายโคแอกเชียล (Coaxial Cable)

สายเคเบิลแบบโคแอกเชียลหรือเรียกสั้นๆว่า "สายโคแอก" จะเป็นสายสื่อสารที่มีคุณภาพดีกว่าและราคาแพงกว่าสายเกลียวคู่ ส่วนของสายส่งข้อมูลจะอยู่ตรงกลางเป็นลวดทองแดงมีชั้นของตัวเหนี่ยวนำหุ้มอยู่ 2 ชั้น ชั้นในเป็นฟ่นเกลียวหรือชั้นแข็ง ชั้นนอกเป็นฟ่นเกลียว และคั่นระหว่างชั้นด้วยฉนวนหนา เปลือกชั้นนอกสุดเป็นฉนวน สายโคแอกสามารถม้วนโค้งงอได้ง่าย มี 2 แบบ คือ 75 โอห์มและ 50 โอห์ม ขนาดของสายมีตั้งแต่ 0.4-1.0 นิ้ว

ชั้นตัวเหนี่ยวนำทำหน้าที่ป้องกันการสูญเสียพลังงานจากการแผ่รังสี เปลือกฉนวนหนาทำให้สายโคแอกมีความคงทนสามารถฝังเดินสายใต้พื้นดินได้ นอกจากนั้นสายโคแอกยังช่วยป้องกัน "การสะท้อนกลับ" (Echo) ของเสียงได้อีกด้วย และลดการรบกวนจากภายนอกได้ดีเช่นกัน

สายโคแอกสามารถส่งสัญญาณข้อมูลได้ทั้งในช่องทางแบบ เบสแบนด์ และแบบ บรอด-แบนด์ การส่งสัญญาณในเบสแบนด์สามารถทำได้เพียง 1 ช่องทางและเป็นแบบครึ่งดูเพล็กซ์ แต่ในส่วนของการส่งสัญญาณในบรอดแบนด์จะเป็นเช่นเดียวกับสายเคเบิลทีวี คือ สามารถส่งได้พร้อมกันหลายช่องทางทั้งข้อมูลแบบดิจิตอลและแบบอนาล็อก สายโคแอกของเบสแบนด์สามารถส่งสัญญาณได้ไกลถึง 2 กม. ในขณะที่บรอดแบนด์ส่งได้ไกลกว่าถึง 6 เท่าโดยไม่ต้องใช้เครื่องทวนสัญญาณ หรือเครื่องขยายสัญญาณเลย ถ้าอาศัยหลักการมัลติเพล็กซ์สัญญาณแบบ FDM สายโคแอกสามารถมีช่องทาง (เสียง) ได้ถึง 10,000 ช่องทางในเวลาเดียวกัน อัตราเร็วในการส่งข้อมูลมีได้สูงถึง 50 เมกะบิตต่อวินาที หรือ 800 เมกะบิตต่อวินาที ถ้าใช้เครื่องทวนสัญญาณทุก ๆ 1.6 กม.

ตัวอย่างการใช้สายโคแอกในการส่งสัญญาณข้อมูลที่ใช้กันมากในปัจจุบัน คือ สายเคเบิลทีวี และสายโทรศัพท์ทางไกล (อนาล็อก) สายส่งข้อมูลในระบบเครือข่ายท้องถิ่น หรือ LAN (ดิจิตอล) หรือใช้ในการเชื่อมโยงสั้น ๆ ระหว่างอุปกรณ์อิเล็กทรอนิกส์

2.2.3 สายไฟเบอร์ออปติก (Fiber Optic Cable)

หลักการทั่วไปของการสื่อสารในสายไฟเบอร์ออปติก คือ การเปลี่ยนสัญญาณ (ข้อมูล) ไฟฟ้าให้เป็นคลื่นแสงก่อน จากนั้นจึงส่งออกไปเป็นพัลส์ของแสงผ่านสายไฟเบอร์ออปติก ทำจากแก้วหรือพลาสติกสามารถส่งลำแสงผ่านสายได้ทีละหลาย ๆ ลำแสงด้วยมุมที่ต่างกัน ลำแสงที่ส่งออกไปเป็นพัลส์นั้นจะสะท้อนกลับไปมาที่ผิวของสายชั้นในจนถึงปลายทาง (ดูในรูป 2.9)

จากสัญญาณข้อมูลซึ่งอาจจะเป็นสัญญาณอนาล็อกหรือดิจิตอล จะผ่านอุปกรณ์ที่ทำหน้าที่มอดูเลตสัญญาณเสียก่อน จากนั้นจะส่งสัญญาณมอดูเลตแล้วผ่านไดโอดซึ่งมี 2 ชนิดคือ LED ไดโอด (Light Emitting Diode) และเลเซอร์ไดโอด หรือ ILD ไดโอด (Injection Leser Diode) โดยไดโอดจะมีหน้าที่เปลี่ยนสัญญาณมอดูเลตให้เป็นลำแสงเลเซอร์ซึ่งเป็นคลื่นแสงในย่านที่มองเห็นได้ หรือเป็นลำแสงในย่านอินฟราเรดซึ่งไม่สามารถมองเห็นได้ ความถี่ย่านอินฟราเรดที่ใช้จะอยู่ในช่วง 1014-1015 เฮิรตซ์ ลำแสงจะถูกส่งออกไปตามสายไฟเบอร์ออปติก เมื่อถึงปลายทางก็จะมีตัวโฟโต้ไดโอด (Photo Diode) ที่ทำหน้าที่รับลำแสงที่ถูกส่งมาเพื่อเปลี่ยนสัญญาณแสงให้กลับไปเป็นสัญญาณมอดูเลตตามเดิม จากนั้นก็จะส่งสัญญาณผ่านเข้าอุปกรณ์ดีมอดูเลต เพื่อทำการดีมอดูเลตสัญญาณมอดูเลตให้เหลือแต่สัญญาณข้อมูลที่ต้องการ

สายไฟเบอร์ออปติกสามารถมีแบนด์วิดท์ (BW) ได้กว้างถึง 3 จิกะเฮิรตซ์ (1จิกะ = 10^9) และมีอัตราเร็วในการส่งข้อมูลได้ถึง 1 จิกะบิตต่อวินาที ภายในระยะทาง 100 กม. โดยไม่ต้องการเครื่องทบทวนสัญญาณเลย สายไฟเบอร์ออปติกสามารถมีช่องทางสื่อสารได้มากถึง 20,000-60,000

ช่องทาง สำหรับการส่งข้อมูลในระยะทางไกล ๆ แต่ถ้าเป็นการส่งข้อมูลในระยะทางใกล้ๆไม่เกิน 10 กม. จะสามารถมีช่องทางได้มากถึง 100,000 ช่องทางทีเดียว

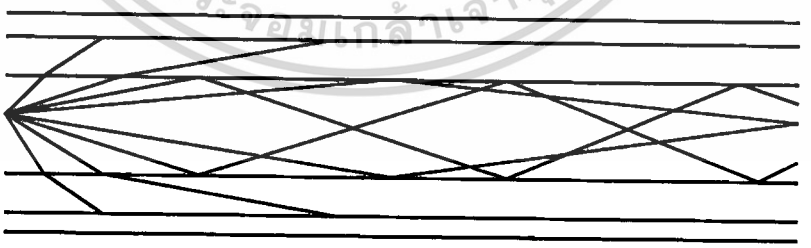
ความผิดพลาดในการส่งข้อมูลผ่านสายไฟเบอร์ออปติกนั้นมีน้อยมาก คือประมาณ 1 ใน 10 ล้านบิตต่อการส่ง 1,000 ครั้ง เท่านั้น ทั้งยังป้องกันการรบกวนจากสัญญาณภายนอกได้โดยสิ้นเชิง แม้ว่าการส่งข้อมูลผ่านทางสายไฟเบอร์ออปติกจะทำได้อย่างมีประสิทธิภาพยอดเยี่ยม และมีจำนวนมหาศาลดังกล่าวมาแล้วก็ตาม แต่เราต้องคำนึงถึงปัญหาและความเหมาะสมบางประการอีกด้วย

อย่างที่ 1 ก็คือราคา ทั้งสายไฟเบอร์ออปติกและอุปกรณ์ประกอบการทั้งหลายมีราคาสูงกว่าการส่งสัญญาณผ่านสายเคเบิลธรรมดามาก

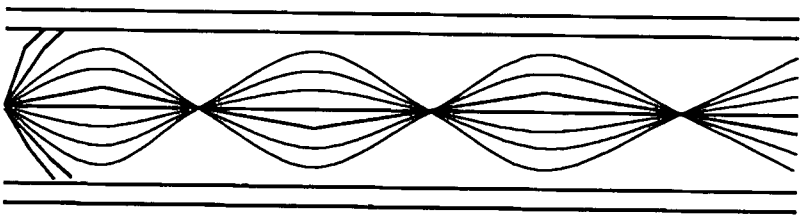
อย่างที่ 2 คือต้องใช้อุปกรณ์พิเศษสำหรับการเปลี่ยนสัญญาณไฟฟ้าให้เป็นคลื่นแสง และจากคลื่นแสงกลับมาเป็นสัญญาณไฟฟ้า และยังมีเครื่องทบทวนสัญญาณอีก อุปกรณ์ดังกล่าวเป็นเทคโนโลยีสมัยใหม่ซึ่งมีความซับซ้อน และราคาแพงมาก

อย่างที่ 3 คือเทคนิคในการติดตั้งระบบ เนื่องจากสายไฟเบอร์ออปติกมีความแข็งแต่เปราะ จึงยากต่อการเดินสายไปตามที่ต่าง ๆ ได้ตามที่ต้องการ อีกทั้งการเชื่อมต่อระหว่างสายก็ทำได้ยากมาก เพราะต้องระวังไม่ให้เกิดการหักเหของลำแสงที่ผิดออกไปซึ่งจะทำให้สัญญาณข้อมูลผิดเพี้ยนได้ รวมถึงการซ่อมแซม หรือการติดตั้งเพิ่มเติมก็ทำได้ยากเช่นกัน

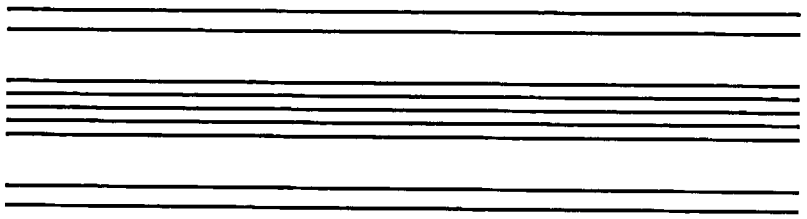
ดังนั้นราคาเทคโนโลยี เทคนิคและประโยชน์ที่จะได้รับจากส่งข้อมูลผ่านสายไฟเบอร์-ออปติก จึงเป็นสิ่งที่ต้องนำมาพิจารณาร่วมกันเสมอถึงความคุ้มค่าในการลงทุน



ก. การส่งสัญญาณแบบ Multimode



ข. การส่งสัญญาณแบบ Multimode Grade Index



ค. การส่งสัญญาณแบบ Singlemode

รูปที่ 2.9 การส่งข้อมูลผ่านสายไฟเบอร์ออปติก

ตารางที่ 2.2 แสดงการเปรียบเทียบคุณลักษณะของสายไฟเบอร์ออปติก 3 ชนิด

ชนิดของไดโอด	LED หรือ ILD	LED หรือ ILD	ILD
แบนด์วิดท์	20 KHz/Km	> 1 GHz/Km	ถึง 1,000 GHz/Km
เส้นผ่าศูนย์กลางเฉพาะของสายใน (10^6 ม.)	> 80 μm	50-60 μm	1.5-5 μm
เส้นผ่าศูนย์กลางรวมเปลือกของสาย (10^6 ม.)	> 160 μm	100-120 μm	15-50 μm
การเบาบางของสัญญาณ (dB/Km)	0.5-2.0	0.5-2.0	0.15
การเชื่อมต่อสาย	ยากมาก	ยากมาก	ยากมาก
ประเภทของงาน	เชื่อมต่อข้อมูลดิจิตอล, เครือข่ายท้องถิ่น (LAN)	สายโทรศัพท์	สายการสื่อสารระยะไกลมากๆ
ราคา	แพงน้อยที่สุด	แพงปานกลาง	แพงที่สุด

2.2.4 ไมโครเวฟ (Microwave)

การส่งสัญญาณข้อมูลไปกับคลื่นไมโครเวฟเป็นการส่งสัญญาณข้อมูลแบบรับช่วงต่อ ๆ กันจากหอ (สถานี) ส่ง-รับสัญญาณหนึ่งไปยังอีกหอหนึ่ง แต่ละหอจะครอบคลุมพื้นที่รับสัญญาณประมาณ 30-50 กม. ระยะห่างของแต่ละหอคำนวณง่ายๆได้จากสูตร

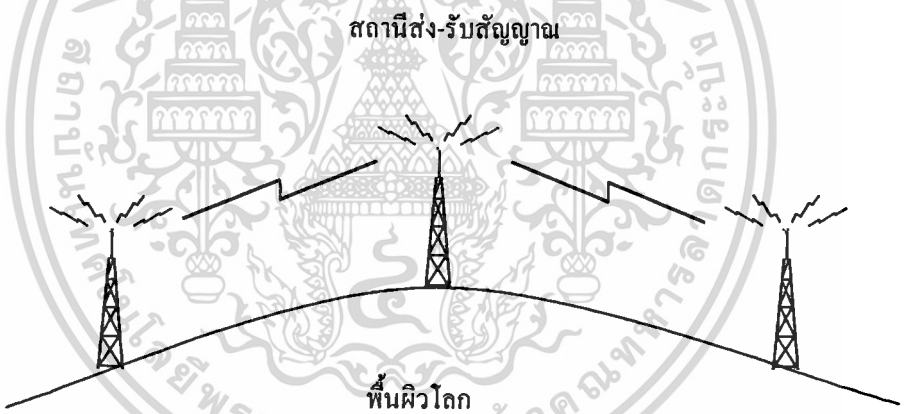
$$d = 7.14(1.33h)^{1/2} \quad \text{กม.}$$

เมื่อ d = ระยะห่างระหว่างหอ

h = ความสูงของหอ

การส่งสัญญาณข้อมูลไมโครเวฟมักใช้กันในกรณีที่การติดตั้งสายเคเบิลทำได้ไม่สะดวก เช่น ในเขตเมืองใหญ่ ๆ หรือในเขตที่ป่าเขา แต่ละสถานีไมโครเวฟจะติดตั้งจานส่ง-รับสัญญาณข้อมูลซึ่งมีเส้นผ่าศูนย์กลางประมาณ 10 ฟุต สัญญาณไมโครเวฟเป็นคลื่นย่านความถี่สูง (2-10 จิกะเฮิรตซ์) เพื่อป้องกันการแทรกหรือรบกวนจากสัญญาณอื่น ๆ แต่สัญญาณอาจจะอ่อนลง หรือหักเหได้ในที่ที่มีอากาศร้อนจัด พายุหรือฝน ดังนั้นการติดตั้งจานส่ง-รับสัญญาณจึงต้องให้หันหน้าของจานตรงกัน และหอยิ่งสูงยิ่งส่งสัญญาณได้ไกล

ปัจจุบันมีการใช้การส่งสัญญาณข้อมูลทางไมโครเวฟกันอย่างแพร่หลาย สำหรับการสื่อสารข้อมูลในระยะทางไกล ๆ หรือระหว่างอาคาร โดยเฉพาะในกรณีที่ไม่สะดวกที่จะใช้สายไฟเบอร์ออปติก หรือการสื่อสารดาวเทียม อีกทั้งไมโครเวฟยังมีราคาถูกกว่า และติดตั้งได้ง่ายกว่า และสามารถส่งข้อมูลได้คราวละมาก ๆ ด้วย (ดูตารางที่ 2.3) อย่างไรก็ตามปัจจัยสำคัญที่ทำให้สื่อกลางไมโครเวฟเป็นที่นิยม คือราคาที่ถูกกว่า



รูปที่ 2.10 การส่งสัญญาณข้อมูลผ่านสื่อไมโครเวฟ

ตารางที่ 2.3 ชนิดของสื่อกลางไมโครเวฟของการส่งข้อมูลดิจิตอล

ย่านความถี่ (GHz)	แบนด์วิดท์ (MHz)	อัตราเร็วข้อมูล (Mbps)
2	7	12
6	30	90
11	40	90
18	220	274

2.2.5 ดาวเทียม (Satellite)

ที่จริงดาวเทียมก็คือสถานีไมโครเวฟลอยฟ้านั่นเอง ซึ่งทำหน้าที่ขยายและทบทวนสัญญาณข้อมูล รับและส่งสัญญาณข้อมูลกับสถานีดาวเทียมที่อยู่บนพื้นโลก สถานีดาวเทียมภาคพื้นจะทำการส่งสัญญาณข้อมูลไปยังดาวเทียมซึ่งจะหมุนไปตามการหมุนของโลกซึ่งมีตำแหน่งคงที่เมื่อเทียบกับตำแหน่งบนพื้นโลก ดาวเทียมจะถูกส่งขึ้นไปให้ลอยอยู่สูงจากพื้นโลกประมาณ 36,000 กม. เครื่องทบทวนสัญญาณของดาวเทียม (Transponder) จะรับสัญญาณข้อมูลจากสถานีภาคพื้นซึ่งมีกำลังอ่อนลงมากแล้วมาขยาย จากนั้นจะทำการทบทวนสัญญาณและตรวจสอบตำแหน่งของสถานีปลายทาง แล้วจึงส่งสัญญาณข้อมูลไปด้วยความถี่ในอีกความถี่หนึ่งลงไปยังสถานีปลายทาง การส่งสัญญาณข้อมูลขึ้นไปหาดาวเทียมเรียกว่า สัญญาณอัปลิงก์ (Up-link) และการส่งสัญญาณข้อมูลกลับมายังพื้นโลกเรียกว่า สัญญาณดาวน์-ลิงก์ (Down-link)

ลักษณะของการรับส่งสัญญาณข้อมูลอาจจะเป็นแบบจุดต่อจุด (Point-to-Point) หรือแบบแพร่สัญญาณ (Broadcast) ดังรูปที่ 2.11 ก. และ ข. สถานีดาวเทียม 1 ดวง สามารถมีเครื่องทบทวนสัญญาณดาวเทียมได้ถึง 25 เครื่อง และสามารถครอบคลุมพื้นที่การส่งสัญญาณได้ถึง 1 ใน 3 ของพื้นผิวโลก ดังนั้นถ้าจะส่งสัญญาณข้อมูลให้ได้รอบโลกสามารถทำได้โดยการส่งสัญญาณผ่านสถานีดาวเทียมเพียง 3 ดวงเท่านั้น

ระหว่างสถานีดาวเทียม 2 ดวง ที่ใช้ความถี่ของสัญญาณเท่ากัน ถ้าอยู่ใกล้กันเกินไปอาจจะทำให้เกิดการรบกวนสัญญาณซึ่งกันและกันได้ เพื่อหลีกเลี่ยงการรบกวน หรือชนกันของสัญญาณดาวเทียม จึงได้มีการกำหนดมาตรฐานระยะห่างของสถานีดาวเทียม และย่านความถี่ของสัญญาณดังนี้

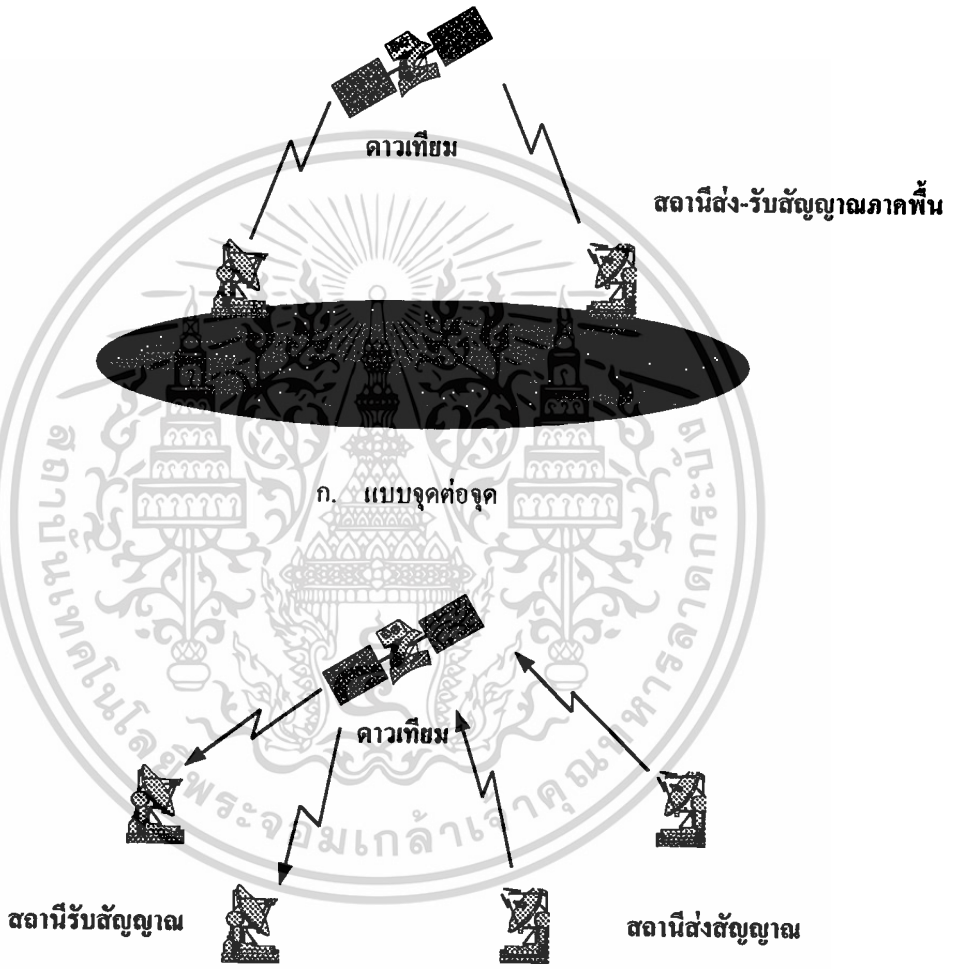
1. ระยะห่างกัน 4 องศา (วัดมุมเทียบกับจุดศูนย์กลางของโลก) ให้ใช้ย่านความถี่ของสัญญาณ 4/6 จิกะเฮิรตซ์ หรือย่าน C แบนด์โดยมีแบนด์วิดท์ของสัญญาณอัป-ลิงก์เท่ากับ 5.925-6.425 จิกะเฮิรตซ์ และมีแบนด์วิดท์ของสัญญาณดาวน์-ลิงก์เท่ากับ 3.7-4.2 จิกะเฮิรตซ์

2. ระยะห่างกัน 3 องศา ให้ใช้ย่านความถี่ของสัญญาณ 12/14 จิกะเฮิรตซ์ หรือย่าน KU แบนด์ โดยมีแบนด์วิดท์ของสัญญาณอัป-ลิงก์เท่ากับ 14.0-14.5 จิกะเฮิรตซ์ และมีแบนด์วิดท์ของสัญญาณดาวน์-ลิงก์เท่ากับ 11.7-12.2 จิกะเฮิรตซ์

นอกจากนี้สภาพอากาศ เช่น ฝนหรือพายุ ก็สามารถทำให้สัญญาณผิดเพี้ยนไปได้เช่นกัน

สำหรับการส่งสัญญาณข้อมูลนั้นในแต่ละเครื่องทบทวนสัญญาณจะมีแบนด์วิดท์เท่ากับ 36 เมกะเฮิรตซ์ และมีอัตราเร็วการส่งข้อมูลสูงสุดเท่ากับ 50 เมกะบิตต่อวินาที

ข้อเสียของการส่งสัญญาณข้อมูลทางดาวเทียมคือ สัญญาณข้อมูลสามารถถูกรบกวนจากสัญญาณภาคพื้นอื่น ๆ ได้ อีกทั้งยังมีเวลาประวิง (Delay Time) ในการส่งสัญญาณเนื่องจากระยะทางขึ้น-ลงของสัญญาณ และที่สำคัญคือมีราคาสูงที่สุดในการลงทุนทำให้ค่าบริการสูงตามขึ้นมาเช่นกัน



รูปที่ 2.11 แสดงการสื่อสารข้อมูลผ่านดาวเทียม

ในการสื่อสารข้อมูลผ่านดาวเทียม ผู้ใช้เพียงทำการติดตั้งจานส่ง-รับสัญญาณข้อมูลกับดาวเทียมซึ่งมีขนาดเล็กประมาณ 0.76-2.40 ม. แต่สามารถส่งสัญญาณข้อมูลได้ด้วยอัตราเร็วถึง 50 กิโลบิตต่อวินาที และมีอัตราเร็วในการรับสัญญาณข้อมูลสูงถึง 256 กิโลบิตต่อวินาที ราคาขึ้นอยู่กับขนาดของจานดาวเทียม ส่วนค่าบริการในการเช่าเครือข่ายวีแซทจะคิดเป็นรายเดือนต่อ 1 สถานี และขึ้นอยู่กับระยะทางด้วย โดยค่าบริการจะประมาณ 3,000-10,000 บาทต่อเดือนต่อสถานี

2.3 อุปกรณ์และการเชื่อมต่อการสื่อสารอนุกรม

2.3.1 โมเด็ม (MODEM)

MODEM ย่อมาจากคำว่า MOdulation-DEModulation เป็นอุปกรณ์สำหรับเปลี่ยนข้อมูลอนุกรมที่ได้รับจากคอมพิวเตอร์ให้อยู่ในรูปแบบที่เหมาะสมกับการส่งผ่านระบบโทรศัพท์และเปลี่ยนข้อมูลที่ได้รับกลับมา โมเด็มถูกใช้เพื่อการส่งข้อมูลระหว่างคอมพิวเตอร์สองเครื่องที่อยู่ห่างไกลกัน หรือใช้เพื่อเชื่อมต่อเทอร์มินัลเข้ากับคอมพิวเตอร์ ในการแทรกโมเด็มสองตัวและสายโทรศัพท์เพิ่มเข้ามาไม่มีทำให้ผลต่ออัตราบอด จำนวนบิตข้อมูล บิตพาริตี้ บิตลบ หรือซอฟท์แวร์แฮนด์เช็คกิ้ง โมเด็มเพียงแต่ทำหน้าที่แปลงแรงดันไฟฟ้าบวกและลบที่แทนแต่ละบิตของตัวอักษรให้เป็นสัญญาณที่เหมาะสมกับการสื่อสารทางโทรศัพท์ โมเด็มในปัจจุบันส่วนใหญ่จะมีชิพประมวลผล (Processor) และหน่วยความจำ (ROM) อยู่ในตัวเครื่อง

วิธีการส่งข้อมูล

สำหรับวิธีการส่งข้อมูลของโมเด็มจะอาศัยเทคนิคการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อกซึ่งลักษณะที่นิยมใช้ คือ

1. วิธีแบบ ASK (Amplitude Shift Keying) ในการมอดูเลตแบบ AM ความถี่ของคลื่นพาห์ (Carrir Wave) ทำหน้าที่นำสัญญาณผ่านตัวกลาง ลักษณะของสัญญาณมอดูเลต เมื่อค่าบิตของสัญญาณข้อมูลดิจิตอลมีค่าเป็น "1" ขนาดของคลื่นพาห์จะสูงขึ้นกว่าปกติ และเมื่อบิตมีค่าเป็น "0" ขนาดของคลื่นพาห์จะตกลงกว่าปกติ

2. วิธีแบบ FSK (Frequency Shift Keying) ในการมอดูเลตแบบ FM ขนาดของคลื่นพาห์จะไม่เปลี่ยนแปลง ที่เปลี่ยนแปลงคือค่าความถี่ของคลื่นพาห์ นั่นคือเมื่อบิตมีค่าเป็น "1" ความถี่ของคลื่นพาห์จะสูงกว่าปกติ และเมื่อบิตมีค่าเป็น "0" ความถี่ของคลื่นพาห์จะต่ำกว่าปกติ

3. วิธีแบบ PSK (Phase Shift Keying) ในการมอดูเลตแบบ PM ค่าของขนาดและความถี่ของคลื่นพาห์จะไม่มีการเปลี่ยนแปลงแต่ที่เปลี่ยนคือเฟสของสัญญาณ กล่าวคือเมื่อมีการเปลี่ยนแปลงสภาวะของบิตจาก "1" ไปเป็น "0" หรือเปลี่ยนจาก "0" ไปเป็น "1" เฟสของคลื่นจะเปลี่ยนไปเป็น 180 องศาด้วย ซึ่งหลักการของ PSK สามารถทำได้ทั้งแบบ 2 เฟส (0, 90, 180 และ 270 องศา) และแบบ 8 เฟส (0, 45, 90, 135, 180, 225, 270 และ 315 องศา)

4. วิธีการมอดูเลตแบบ QAM (Quadrature Amplitude Modulation) ซึ่งเป็นการมอดูเลตสัญญาณข้อมูล โดยอาศัยหลักการของ AM และ PM รวมกันกล่าวคือสัญญาณข้อมูลจะรวมเข้ากับสัญญาณคลื่นพาห์เมื่อแอมปลิจูดและเฟสของสัญญาณเปลี่ยนแปลง

วิธีแบบ ASK จะใช้กับการส่งข้อมูลอัตราเร็วต่ำ วิธีแบบ FSK จะใช้ในการส่งข้อมูลด้วยอัตราเร็วปานกลาง ส่วนวิธีแบบ PSK และ QAM เราใช้ในการส่งข้อมูลอัตราเร็วสูง

ชนิดของโมเด็ม

ในปัจจุบันโมเด็มมีการพัฒนาออกมาใช้งานอย่างมากมาย ดังแสดงตัวอย่างในตารางที่ 2.4

ตารางที่ 2.4 ตารางเปรียบเทียบชนิดของโมเด็ม

มาตรฐาน	อัตราเร็ว (bps)	การมอดูเลต	สายสื่อสาร	การเชื่อมโยง	การส่งข้อมูล
Bell 103	300	FSK	dial-up	ดูเพล็กซ์เต็ม	อะซิงโครนัส
Bell 202	1,200	PSK	dial-up	ครึ่งดูเพล็กซ์	อะซิงโครนัส
Bell 202	1,200	PSK	leased	ดูเพล็กซ์เต็ม	อะซิงโครนัส
Bell 212A	1,200	PSK	dial-up	ดูเพล็กซ์เต็ม	อะซิงโครนัส
CCITT V.22	1,200	PSK	dial-up	ดูเพล็กซ์เต็ม	อะซิงโครนัส
CCITT V.22 bis	2,400	QAM	dial-up	ดูเพล็กซ์เต็ม	อะซิง/ซิงโครนัส
CCITT V.26 bis	2,400	PSK	dial-up	ครึ่งดูเพล็กซ์	ซิงโครนัส
CCITT V.26 ter	2,400	PSK	dial-up	ดูเพล็กซ์เต็ม	ซิงโครนัส
Bell 208	4,800	QAM	leased	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.27 ter	4,800	PSK	dial-up	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.29	9,600	QAM	dial-up	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.32	9,600	QAM	dial-up	ดูเพล็กซ์เต็ม	อะซิง/ซิงโครนัส
CCITT V.32 bis	14,400	QAM	dial-up	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.33	14,400	QAM	leased	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.42	19,200-56,000	QAM	dial-up, leased	ดูเพล็กซ์เต็ม	ซิงโครนัส
CCITT V.42 bis	19,200-56,000	QAM	dial-up, leased	ดูเพล็กซ์เต็ม	ซิงโครนัส

รูปแบบข้อมูล

ส่วนรูปแบบของข้อมูลที่โมเด็มส่งผ่านไปสู่สายสื่อสารสามารถแยกได้ตามประเภทของโมเด็มคือโมเด็มแบบอะซิงโครนัสและโมเด็มแบบซิงโครนัส

อัตราเร็วข้อมูล

อัตราเร็วในการส่งข้อมูลแบ่งออกเป็น 3 ระดับคือ

1. อัตราเร็วต่ำ ประมาณ 300 บิตต่อวินาที (ส่วนใหญ่เป็นโมเด็มแบบอะซิงโครนัส และใช้กับซีพียูขนาด 8 บิต)

2. อัตราเร็วปานกลาง อยู่ในช่วง 1,200-9,600 บิตต่อวินาที (มีทั้งโมเด็มแบบอะซิงโครนัสและแบบซิงโครนัส และใช้กับซีพียูขนาด 16 บิต)

3. อัตราเร็วสูง มากกว่า 9,600 บิตต่อวินาที (ส่วนใหญ่เป็นโมเด็มแบบซิงโครนัส และใช้กับซีพียูขนาด 32 หรือ 64 บิต)

ภาวะคำสั่งและภาวะออนไลน์

ในการทำงานโมเด็มจะอยู่ในภาวะใดภาวะหนึ่งคือ ภาวะคำสั่ง (Command mode) หรือไม่ก็ภาวะออนไลน์ (On-line mode) ขณะที่โมเด็มอยู่ในภาวะคำสั่งสามารถสั่งงานมันได้จากคอมพิวเตอร์ ตัวอย่างเช่น คำสั่งที่สั่งให้โมเด็มทำการหมุนหมายเลขโทรศัพท์ หรือตอบรับเมื่อกริ่งโทรศัพท์ดังโดยอัตโนมัติ คำสั่งจะถูกส่งให้กับโมเด็มไม่ใช่ถูกส่งออกไป

เมื่อมีการเชื่อมต่อเกิดขึ้นกับโมเด็มระยะไกล โมเด็มท้องถิ่นจะเข้าสู่ภาวะออนไลน์และจะไม่แปลงความหมายข้อมูลที่ถูกส่งให้กับมัน แต่จะส่งผ่านข้อมูลออกไปแทน ถ้าสัญญาณพาหะหายไป เช่น โมเด็มระยะไกลวางหูโทรศัพท์โมเด็มจะกลับสู่ภาวะคำสั่ง การออกจากภาวะออนไลน์โดยไม่ต้องตัดการเชื่อมต่อ ทำได้โดยรอคอยเป็นช่วงเวลา Guard time (คำปริยายคือ 1 วินาที) พิมพ์ Escape command (คำปริยายคือ +++) , และรอสัก 1 วินาที ก่อนที่ข้อมูลจะถูกแปลงเป็นคำสั่ง

บรรทัดคำสั่ง

บรรทัดคำสั่งทั้งหมดเริ่มต้นด้วย AT หรือ at (ไม่ใช่ At หรือ aT) ถ้าไม่มีการระบุเป็นอย่างอื่น โมเด็มสามารถตรวจสอบอัตราบอด ความยาวเวิร์ดและพาริตี้จากอักษรสองตัวนี้ แน่นอนว่าจะต้องตั้งค่าพารามิเตอร์เหล่านี้บนคอมพิวเตอร์ด้วย หลายๆคำสั่งสามารถใช้ได้พร้อมกันในหนึ่งบรรทัดคำสั่ง

รหัสผลลัพธ์

เมื่อโมเด็มได้รับคำสั่ง มันจะส่งรหัสผลลัพธ์ (Result code) กลับมา รหัสนี้อาจอยู่ในรูปของข่าวสารข้อความหรือรหัสตัวเลข ถ้าคุณกำลังควบคุมโมเด็มผ่านทางซอฟต์แวร์ การใช้รหัสตัวเลขจะเหมาะสมกว่า ถ้าคุณกำลังควบคุมโมเด็มจากแป้นพิมพ์ ข่าวสารข้อความก็น่าจะดี ชนิดของรหัสผลลัพธ์สามารถเลือกได้โดยการใช้คำสั่งหรือการตั้งค่าสวิตช์ ตารางที่ 2.5 แสดงรหัสคำสั่งต่างๆ

ตารางที่ 2.5 แสดงรหัสคำสั่งของโมเด็ม Hayes

รหัสตัวเลข	รหัสข้อความ	ความหมาย
0	OK	คำสั่งถูกนำไปปฏิบัติ
1	CONNECT	เชื่อมต่อที่ 0-300 bps*
2	RING	ตรวจพบสัญญาณกริ่ง
3	NO CARRIER	ไม่พบโมเด็มระยะไกล
4	ERROR	ความผิดพลาดในคำสั่ง
5	CONNECT 1200	เชื่อมต่อที่ 1200 bps*
6	NO DIALTONE	ไม่พบสัญญาณให้หมุนบนสายโทรศัพท์ท้องถิ่น
7	BUSY	สายไม่ว่าง
8	NO ANSWER	ไม่มีการตอบสนองจากโมเด็มระยะไกล
9	CONNECT 2400	เชื่อมต่อที่ 2400 bps

*ในโมเด็ม 1200 bps ถ้า X0 (ค่าโดยปริยาย) ถูกเซต รหัสผลลัพธ์ 1 ถูกใช้สำหรับการเชื่อมต่อ 0-300-bps และ 1200-bps เพื่อประกันความเข้ากันได้กับซอฟต์แวร์ที่เขียนมาสำหรับโมเด็ม 300-bps

ภาวะการตอบรับโทรศัพท์

โมเด็มสามารถถูกตั้งให้ตอบรับโทรศัพท์อัตโนมัติ โดยสามารถกำหนดจำนวนครั้งของเสียงกริ่งก่อนที่โมเด็มจะตอบรับโดยการตั้งค่ารีจิสเตอร์ S0 โมเด็มตอบรับโทรศัพท์ มันจะส่งสัญญาณพาหะและรอคอยการตอบสนอง ถ้าไม่มีสัญญาณพาหะส่งกลับมาภายในช่วงเวลาที่ตั้งโดยรีจิสเตอร์ S7 มันจะวางสาย

การตรวจสอบอัตราบอดที่เข้ามาจะเป็นไปโดยอัตโนมัติ และโมเด็มจะส่งรหัสผลลัพธ์เพื่อบอกอัตราบอด คอมพิวเตอร์ของคุณต้องรู้จักรหัสผลลัพธ์นี้และปรับอัตราบอดของมันเพื่อที่จะตอบสนองอัตราบอดที่เข้ามาแตกต่างกันโดยอัตโนมัติ

คำสั่งหมุนหมายเลขโทรศัพท์

ชุดคำสั่งที่ใช้ในการหมุนหมายเลขโทรศัพท์แสดงดังตารางที่ 2.6

ตารางที่ 2.6 คำสั่งหมุนหมายเลขโทรศัพท์ของโมเด็ม Hayes

คำสั่ง	ความหมาย
ATDT	หมุนหมายเลขโทรศัพท์แบบ Touch-Tone
ATDP	หมุนหมายเลขโทรศัพท์แบบ Pulse หยุดชั่วขณะระหว่างหมายเลขแต่ละข้างของคอมมา
ATT	ใช้ Touch-tone โดยปริยาย
ATP	ใช้ Pulse โดยปริยาย
!	โอนสาย
W	รอจนได้รับสัญญาณให้หมุนครั้งที่สอง
@	รอเสียงกริ่งหนึ่งครั้งหรือมากกว่าที่ตามด้วยการไม่มีเสียง 5 วินาที
O	(ที่จุดสิ้นสุดบรรทัดคำสั่ง) กลับสู่ภาวะออนไลน์
;	(ที่จุดสิ้นสุดบรรทัดคำสั่ง) พักอยู่ในภาวะคำสั่งหลังจากทำตามคำสั่ง
R	เรียกโมเด็มที่เป็นผู้เรียกเพียงอย่างเดียว
/	รอ 1/8 วินาที

คำสั่ง ATX

เป็นชุดคำสั่งที่เพิ่มเข้ามาในโมเด็มรุ่นใหม่ ซึ่งมีรายละเอียดดังแสดงในตารางที่ 2.7

ตารางที่ 2.7 รหัส ATX ของโมเด็ม Hayes

หมายเลข ATX	รหัสสำหรับ 1200 bps	รอสัญญาณให้หมุน	แจ้งข้อผิดพลาดไม่มีสัญญาณให้หมุน	แจ้งข้อผิดถ้าสายไม่ว่าง
0	1	NO	NO	NO
1	5	NO	NO	NO
2	5	YES	YES	NO
3	5	NO	NO	YES
4	5	YES	YES	YES

คำสั่งอื่นๆ

ยังมีคำสั่งอีกมากดังรายการในตารางที่ 2.8 ชุดคำสั่ง ATS0 จนถึง ATS16 ใช้เพื่อตั้งค่ารีจิสเตอร์ของโมเด็ม รีจิสเตอร์เหล่านี้จะทำการบันทึกค่าพารามิเตอร์ต่าง ๆ เช่น การจัดเวลา การตั้งค่ารีจิสเตอร์ทำได้โดยส่งคำสั่ง ATS ตามด้วยหมายเลขรีจิสเตอร์และค่านั้น ค่าในรีจิสเตอร์สามารถ

ตรวจสอบโดยใช้ ATS ตามด้วยหมายเลขรีจิสเตอร์และเครื่องหมายคำถาม โดยในตารางที่ 2.9 จะแสดงรีจิสเตอร์ที่มีการใช้บ่อย

ตารางที่ 2.8 คำสั่งอื่นของโมเด็ม Hayes

คำสั่ง	ความหมาย
ATH	วางสาย
ATZ	วางสายและรีเซตกลับสู่ค่าโดยปริยาย
A/	ทำคำสั่งล่าสุดซ้ำ
ATB0	ใช้โปรโตคอลนานาชาติ
ATB1	กลับคืนสู่ Bell mode
ATC0	ปิดสัญญาณพาหะ
ATC1	เปิดสัญญาณพาหะ
ATE0	ไม่ให้มีเอคโคไปยังจอภาพ
ATE1	ให้มีเอคโคไปยังจอภาพ
ATF0	ไม่ใช้การส่งแบบฮาล์ฟดูเพล็กซ์
ATF1	ใช้การส่งแบบฮาล์ฟดูเพล็กซ์
ATL1-3	ตั้งความดังของลำโพง
ATM0	ปิดลำโพง
ATM1	เปิดลำโพงจนกระทั่งถูกเชื่อมต่อ
ATM2	เปิดลำโพงทิ้งไว้
ATQ0	ใช้รหัสผลลัพธ์ (ค่าโดยปริยาย)
ATQ1	ไม่ใช้รหัสผลลัพธ์
ATV0	แสดงรหัสผลลัพธ์เป็นตัวเลข
ATV1	แสดงรหัสผลลัพธ์เป็นคำ
ATY1	ส่งสัญญาณเบรก 4 วินาที ก่อนยกเลิกการเชื่อมต่อม, ยกเลิกการเชื่อมต่อถ้าได้รับสัญญาณเบรก 1.6 วินาที
ATY0	ไม่ส่งและไม่ตอบสนองต่อสัญญาณเบรก (ค่าปริยาย)
ATH1	ใช้งานรีเลย์สายโทรศัพท์ (line relay) และรีเลย์เสริม (auxiliary relay)
ATH2	ใช้งานรีเลย์สายโทรศัพท์
ATS0-16	ตั้งค่ารีจิสเตอร์ของโมเด็ม

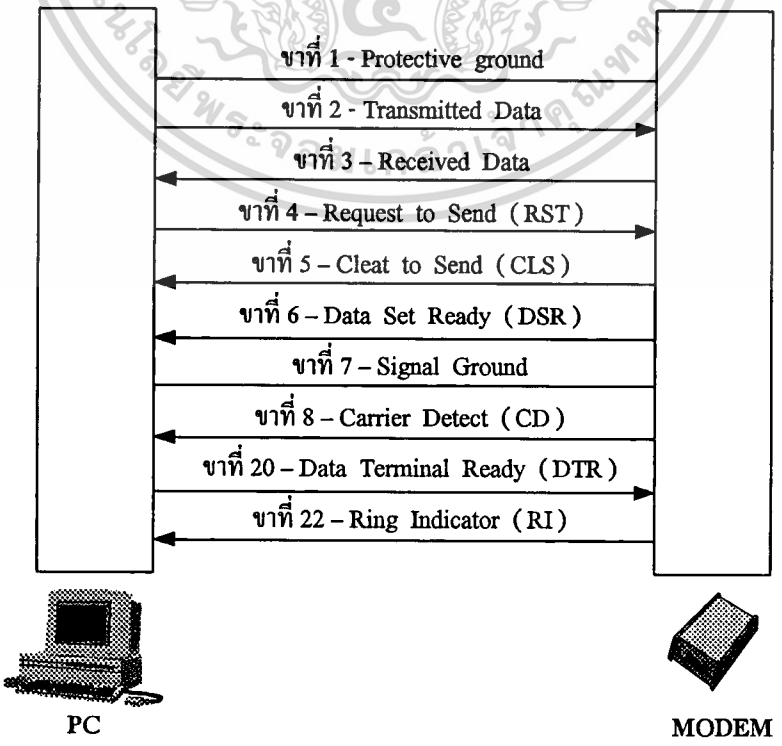
ตารางที่ 2.9 รีจิสเตอร์ของโมเด็ม Hayes ที่ใช้บ่อย

รีจิสเตอร์	ย่าน/หน่วย	ความหมาย
S0	0-255 ครั้ง	จำนวนเสียงกริ่งก่อนที่จะรับโทรศัพท์
S6	2-255 วินาที	เวลาที่รอคอยสัญญาณให้หมุน
S7	1-60 วินาที	เวลาที่รอคอยสัญญาณพาหะ
S13	บิตแมป (Bitmapped)	รีจิสเตอร์สถานะของ UART

2.3.2 โมเด็มอินเตอร์เฟซ (Modem Interface)

อินเตอร์เฟซ RS-232-C, RS-232-D

ในการใช้โมเด็มเพื่อส่ง-รับข้อมูลโมเด็มจะต้องทำการเชื่อมต่อเข้ากับพอร์ต (port) หรือช่อง (slot) ของเทอร์มินัล ไมโครคอมพิวเตอร์ เครื่องมือสื่อสาร หรือคอมพิวเตอร์ส่วนบุคคล โดยอาศัยอุปกรณ์เชื่อมต่อเรียกว่า อินเตอร์เฟซ RS-232-C หรือ RS-232-D เป็นอินเตอร์เฟซแบบธรรมดาที่สุด และเป็นอินเตอร์เฟซแบบอนุกรม (ส่ง 1 บิตต่อครั้ง) มีหัวต่อ 2 ด้าน ด้านหนึ่งเป็นตัวผู้และปลายอีกด้านหนึ่งเป็นตัวเมีย ประกอบด้วยเข็ม (pin) จำนวน 25 เข็ม ด้านหนึ่งเสียบเข้ากับโมเด็ม อีกด้านหนึ่งเสียบเข้ากับพอร์ต หรือช่องของอุปกรณ์คอมพิวเตอร์



รูปที่ 2.12 แสดงอินเตอร์เฟซ RS-232-C , RS-232-D

ทางเดินของข้อมูล (Data Path)

สิ่งสำคัญที่สุดส่วนหนึ่งของการอินเทอร์เฟซแบบ RS-232-C คือเรื่องทางเดินของข้อมูล มีวงจรทางเดินของข้อมูลอยู่ 2 วงจร คือ วงจรหนึ่งจาก DTE ไปยัง DCE และอีกวงจรคือจาก DCE ย้อนกลับมาที่ DTE โดยพีซีจะทำการส่งข้อมูลไปที่โมเด็มบนขา 2 ของเคเบิลและรับข้อมูลจากโมเด็มบนขา 3 ส่วนขา 7 จะทำหน้าที่เป็นกราวของวงจรทั้งสอง

สัญญาณควบคุม

มีสัญญาณควบคุมการทำงานในการอินเทอร์เฟซแบบ RS-232-C ทำหน้าที่ในการควบคุมการเชื่อมโยงและบอกสถานะระหว่างโมเด็มกับพีซี ซึ่งมีรายละเอียด ดังนี้

สัญญาณ Request to Send และ Clear to Send (ขา 4 และ ขา 5)

วงจร RTS และ CTS เตรียมไว้สำหรับใช้ควบคุมการไหลของข้อมูลระหว่างโมเด็มกับพีซี โมเด็มเปิดสัญญาณ CTS ให้ on เมื่อมันพร้อมที่จะรับข้อมูล และปิดมัน off เมื่อมันไม่สามารถรับข้อมูลได้ ในลักษณะที่คล้ายกัน พีซีเปิดสัญญาณ RTS on เมื่อมันสามารถที่จะรับข้อมูลได้ และปิดสัญญาณ RTS off เมื่อพีซีไม่สามารถรับข้อมูลที่กำลังเข้ามาได้ สัญญาณทั้งสองนี้ใช้ควบคุมการไหลของข้อมูลระหว่างโมเด็มกับพีซี มันทำให้โมเด็มสามารถติดต่อสื่อสารได้ที่ความเร็วสูงๆ เพราะมันจะหยุดการไหลของข้อมูลได้เสมอเมื่อจำเป็น ทั้งพีซีและโมเด็มสามารถใช้สัญญาณเหล่านี้เพื่อหลีกเลี่ยงการล้นเกิน (overflowing) ของบัฟเฟอร์ที่เก็บข้อมูลและการสูญหายของข้อมูล

สัญญาณ Data Terminal Ready (ขา 20) และ Data Set Readly (ขา 6)

สัญญาณ DTR on บอกโมเด็มว่าขณะนี้เทอร์มินัลหรือพีซีเปิดอยู่และพร้อมที่จะติดต่อสื่อสารกับโมเด็มแล้ว โมเด็มก็จะตอบรับโดยเปิดสัญญาณ DSR on ซึ่งเป็นการบอกพีซีให้รู้ว่าโมเด็มพร้อมที่จะรับข้อมูลเช่นกัน

สัญญาณ Carrier Detect (ขา 8) และ Ring Indicator (ขา 22)

สัญญาณ Carrier Detect บอกให้รู้ว่าขณะนี้โมเด็มได้รับและรับรู้โทนจากโมเด็มตัวอื่น ส่วนสัญญาณ Ring Indicator บอกให้พีซีรู้ว่าสายโทรศัพท์ของโมเด็มกำลังมีริงกิงอยู่ ในทางปฏิบัติโมเด็มจะมีวงจรตอบรับโดยอัตโนมัติ แต่บางครั้งสัญญาณ RI ถูกใช้เพื่อเริ่มต้นโปรแกรมในพีซีที่กำหนดรูปลักษณ์ในการเชื่อมต่อกับคอลล์ที่เรียกเข้ามา

อินเตอร์เฟซ RS-422-A และ RS-423-A

อินเตอร์เฟซ RS-422-A และ RS-423-A เป็นอินเตอร์เฟซมาตรฐานของ EIA ซึ่งเรียกว่า EIA-530 เพื่อเพิ่มอัตราเร็ว และระยะทางของการส่งข้อมูลให้มากกว่า RS-232-C สำหรับ RS-422-A และ RS-423-A จะเป็นอินเตอร์เฟซขนาด 37 เข็ม สิ่งที่เพิ่มเติมไปจาก RS-232-C คือคุณสมบัติพิเศษของสัญญาณอิเล็กทรอนิกส์ ในขณะที่ RS-232-C ออกแบบมาเพื่อใช้กับอุปกรณ์อิเล็กทรอนิกส์ดิจิตอลแต่ RS-422-A และ RS-423-A ออกแบบมาสำหรับเทคโนโลยีวงจรอินทิเกรต (I.C.)

อินเตอร์เฟซ RS-449 และ X.21

RS-449 เป็นอินเตอร์เฟซที่มีประสิทธิภาพสูงกว่า RS-232-C โดยมีคุณสมบัติเพิ่มขึ้นมาอีก คือ มีโหมดทดสอบเพื่อทดสอบการทำงานของเครื่องโมเด็ม สามารถตรวจสอบได้ว่าเทอร์มินัลปลายทางนั้นว่างอยู่ หรือกำลังใช้งานและสามารถควบคุมการทำงานของโมเด็มได้จากอุปกรณ์อื่น เช่น จากเครื่องส่งข้อมูล หรือจากเทอร์มินัลอื่น

ส่วน X.21 เป็นอินเตอร์เฟซที่ประสิทธิภาพสูงขึ้นไปอีกจาก RS-449 คือสามารถควบคุมการส่งอักขระตรวจสอบความผิดพลาดการส่งข้อมูลและฟังก์ชันอื่นๆโดยมีเข็มเพียง 15 เข็ม อินเตอร์เฟซ X.21 จะใช้ในเครือข่ายสาธารณะแบบเซอร์กิตสวิตช์ (Circuit-Switched Network) เช่น เครือข่าย ATM เครือข่ายดาต้าเนต เป็นต้น

อินเตอร์เฟซ ISDN

ในอนาคตเชื่อว่าอินเตอร์เฟซทั้งหลายที่ได้กล่าวมาจะถูกแทนที่ด้วยอินเตอร์เฟซแบบ ISDN (Intrgrated Services Digital Network) อินเตอร์เฟซ ISDN ใช้ในเครือข่ายดิจิตอล ISDN สามารถส่ง-รับข่าวสารได้ทั้งข้อมูล ข้อความ ภาพ เสียง ภาพและเสียง (วิดีโอ) อย่างไรก็ตามเครือข่าย ISDN ยังอยู่ในขั้นพัฒนาเทคโนโลยีแต่บางประเทศเริ่มเปิดให้ใช้บริการบ้างแล้ว เช่น สหรัฐอเมริกา หรือบางประเทศในยุโรป

2.4 โปรโตคอลในการสื่อสารข้อมูล

2.4.1 โปรโตคอล XMODEM

XMODEM เป็นโปรโตคอลถ่ายโอนไฟล์ที่เรียบง่ายมาก แม้ว่ามันจะมีข้อจำกัด แต่ก็เป็นโปรโตคอลที่ใช้กันกว้างขวางที่สุด XMODEM ถูกใช้บ่อยที่สุดในการดาวน์โหลดไฟล์ทุกชนิดจากระบบคอมพิวเตอร์ตามบ้าน เช่น พีซี นอกจากจะถูกใช้ในการสื่อสารในระหว่างไมโครคอมพิวเตอร์แล้ว XMODEM ยังถูกนำไปใช้กับการสื่อสารของเมนเฟรมด้วย CompuServe เป็นตัวอย่างที่เห็นชัดที่สุดของระบบการสื่อสารของเมนเฟรมที่เสนอการถ่ายโอนไฟล์ด้วย XMODEM

บล็อก

ข้อมูลที่ถูกส่งโดย XMODEM จะถูกแบ่งออกเป็นบล็อก แต่ละบล็อกประกอบด้วยอักขระต่าง ๆ ดังแสดงในตารางที่ 2.10

ตารางที่ 2.10 รูปแบบบล็อกของ XMODEM

ออฟเซต	ความหมาย
0	SOH (start-of-header,ASCII 01)
1	หมายเลขบล็อก เริ่มต้นจาก 1 แต่จะกลับเป็น 0 หลังจาก FF
2	คอมพลีเมนต์ของหนึ่งของหมายเลขบล็อก (255-หมายเลขบล็อก)
3-130	ข้อมูล 128 ไบต์
131	Checksum ผลรวมของไบต์ข้อมูลเท่านั้น

หมายเลขบล็อกเริ่มต้นที่ 1 แต่ถูกคำนวณด้วยมอดูโล 256 (Modulo 256) หมายความว่าหลังจาก 255 (FF ฐานสิบหก) มันจะกลับเป็นศูนย์ คอมพลีเมนต์ที่หนึ่งคำนวณโดยการลบหมายเลขบล็อกจาก 255 หรือโดยการคอมพลีเมนต์ทุกบิตในหมายเลขนั้น (สลับค่าศูนย์กับหนึ่ง) Checksum เป็นไบต์เดียวที่คำนวณโดยการบวกข้อมูล 128 ไบต์ เข้าด้วยกัน โดยไม่สนใจตัวทด

โปรโตคอลในระดับไฟล์

ก่อนที่คอมพิวเตอร์ฝ่ายส่งจะสามารถส่งข้อมูลได้ มันต้องรับอักขระ NAK (negative acknowledgment) จากคอมพิวเตอร์ฝ่ายรับ โปรแกรมผู้รับจะส่ง NAK (15 ฐานสิบหก) หลังจากไทม์เอาต์ทุกๆ 10 วินาที (อย่างเป็นทางการ แต่ในทางปฏิบัติบางครั้งก็ถี่กว่า) ที่ไม่ได้รับข้อมูล NAK ตัวแรกกระตุ้นให้ผู้ส่งเริ่มทำการส่ง

เมื่อโปรแกรมผู้รับเริ่มการรับบล็อกมันจะรายงานข้อผิดพลาดเมื่อมีช่องว่าง 1 วินาที หรือมากกว่า เกิดขึ้นระหว่างตัวอักษรในบล็อก รวมทั้ง Checksum อย่างไรก็ตามมันต้องรอให้สายว่างก่อนที่จะส่ง NAK เพื่อแจ้งข้อผิดพลาดไทม์เอาต์ 1 วินาที ไม่เพียงพอสำหรับการสื่อสารทางไกล จึงมักจะใช้เวลาคอยที่นานกว่านี้

จากนั้นผู้รับจะตรวจสอบหมายเลขบล็อกและรายงานข้อผิดพลาด ถ้ามันไม่เรียงลำดับ ถ้าหมายเลขบล็อกเหมือนกับบล็อกล่าสุด แสดงถึงการส่งซ้ำซึ่งไม่ควรถูกพิจารณาเป็นข้อผิดพลาด หลังจากการรับแต่ละบล็อกผู้รับส่ง ACK (06 ฐานสิบหก) ถ้าบล็อกที่รับมาถูกต้อง หรือ NAK ถ้าไม่ถูกต้อง ในกรณีหลังผู้ส่งจะส่งบล็อกนั้นซ้ำ หลังจากการตอบรับบล็อกนั้นแล้ว บล็อกต่อไปจึงจะถูกส่ง

ในตอนจบของการส่ง ผู้ส่งจะส่ง EOT (04 ฐานสิบหก) และรอคอย ACK มันจะส่ง EOT ซ้ำ ถ้าไม่ได้รับ ACK

ตัวเลือก CRC

เนื่องจาก Checksum หนึ่งไบต์ไม่เพียงพอสำหรับตรวจสอบข้อผิดพลาดทั้งหมด ส่วนขยายของ XMODEM ที่เรียกว่า ตัวเลือก CRC จึงถูกคิดขึ้นโดยใช้ตัวเลขสองไบต์ ตัวเลขนี้เรียกว่า Cyclical redundancy check (CRC-16) ซึ่งสามารถตรวจสอบข้อผิดพลาดได้อย่างน้อย 99.99 เปอร์เซ็นต์

โดยปกติผู้รับต้องบอกผู้ส่งว่าใช้ตัวเลือก CRC โดยการส่งอักขระ C แทน NAK ในการร้องขอให้เริ่มต้นส่ง แต่เนื่องจากไม่ทุกเวอร์ชันของ XMODEM ที่มีตัวเลือก CRC ผู้รับจึงควรเปลี่ยนไปส่ง NAK ถ้าไม่มีการตอบสนองหลังจากการส่ง C

ข้อได้เปรียบและเสียเปรียบของ XMODEM

ข้อได้เปรียบหลักของ XMODEM คือความเรียบง่ายและความเป็นสากล โชคไม่ดีที่ XMODEM มักถูกใช้ในลักษณะที่มันไม่ได้ถูกตั้งใจไว้แต่เดิม

มันไม่สามารถแก้ปัญหาความยาวเวิร์ดได้ ไม่มีการแปลงรูปแบบใดเป็นพิเศษที่จะทำให้ข้อมูลแปดบิตถูกส่งไปกับการสื่อสารเจ็ดบิตได้ ดังนั้นถ้าข้อมูลไบนารีกำลังถูกส่ง ต้องใช้การสื่อสารแปดบิต โปรโตคอลกำหนดให้ใช้แปดบิตข้อมูล ไม่มีพาริตี้ และหนึ่งบิตจบ แม้ว่าจะมีเพียงข้อมูลแบบเท็กซ์กำลังถูกส่งก็ตาม

ไม่มีการป้องกันการแปลความหมายข้อมูลไบนารีผิดเป็นสัญญาณควบคุมเช่นถ้าอุปกรณ์ฝ่ายรับถือว่า ASCII 4 เป็น End of Transmission เสมอ การส่งก็สามารถทำได้เพียงสามบล็อก เนื่องจากบล็อกที่สี่จะมี ASCII 4 เป็นหมายเลขบล็อก ซึ่งหมายความว่า แม้ข้อมูลที่ถูกถ่ายโอนจะอยู่ในรูปแบบ ASCII ที่ยอมรับได้ ตัวโปรโตคอลเองก็อาจเพิ่มรหัสที่ยอมรับไม่ได้เข้าไป

ไม่มีความต้องการการสื่อสารแบบฟูลดูเพล็กซ์ เนื่องจากอุปกรณ์ฝ่ายรับต้องคอยให้การส่งหยุดก่อนที่จะส่ง NAK วิธีนี้ทำให้เกิดเวลาหน่วง โดยเฉพาะอย่างยิ่งบนสายทางไกล โปรโตคอลที่ซับซ้อนกว่านี้อนุญาตให้บล็อกหนึ่งได้รับการตอบรับหรือปฏิเสธในขณะที่บล็อกที่ตามมากำลังถูกส่ง

ข้อบกพร่องหลายๆอย่างของ XMODEM ถูกแก้ไขใน YMODEM และ ZMODEM

2.4.2 โปรโตคอล YMODEM

YMODEM คือ XMODEM ที่มีการขยายความสามารถบางอย่าง ดังนี้

- การตรวจสอบข้อผิดพลาดด้วย CRC-16
- มีบล็อกขนาด 1K เป็นทางเลือก การส่ง STX (02 ฐานสิบหก) ที่จุดเริ่มต้นของแต่ละบล็อก แทน SOH (01 ฐานสิบหก) เป็นสัญลักษณ์ว่าบล็อกที่ตามมายาว 1024 ไบต์ แทน 128 ไบต์ บล็อกขนาด 1024 และ 128 ไบต์ สามารถถูกส่งผสมกันไปได้ในการส่งครั้งหนึ่ง
- การยกเลิกด้วย CAN-CAN อักขระ CAN สองตัวติดกันบอกว่าการถ่ายโอนไฟล์ต้องถูกยกเลิก
- การส่งชื่อไฟล์ บล็อกแรกมีหมายเลขศูนย์ประกอบด้วยชื่อไฟล์ปิดท้ายด้วย ASCII 0 ชื่อไฟล์ควรถูกแปลง เป็นอักษรตัวเล็กถ้าคอมพิวเตอร์ฝ่ายส่งไม่สนับสนุนทั้งชื่อไฟล์ตัวเล็กและตัวใหญ่ ชื่อไฟล์สามารถรวมถึงชื่อเส้นทาง แยกโดย Slash (/) ซึ่งเป็นวิธีปกติในยูนิกซ์ ไม่ใช่ Bacslash (\) ดังที่ใช้ในดอส อย่างไรก็ตามโดยปกติชื่อไฟล์แบบเต็มจะไม่ถูกส่งไป เนื่องจากคอมพิวเตอร์ฝ่ายรับอาจมีโครงสร้างไดเรกทอรีต่างกัน
- การส่งไฟล์แบบแบตช์ (Batch-file transmission) ไฟล์หลายไฟล์สามารถถูกส่งในคราวเดียว ที่จุดสิ้นสุดของแต่ละไฟล์ คอมพิวเตอร์ฝ่ายส่งจะส่ง EOT หนึ่งถึงสิบตัว (ASCII 4 หรือ Ctrl-D) จนกระทั่งได้รับ ACK จากนั้นจึงสามารถส่งไฟล์อื่น (เริ่มต้นด้วยชื่อไฟล์ในบล็อกศูนย์) หรือจบการส่ง โดยการส่งแพคเกตศูนย์ที่ชื่อไฟล์เป็น NULL

นอกจากชื่อไฟล์แล้วบล็อกศูนย์สามารถมีฟิลด์เพิ่มเติมอีกสี่ฟิลด์ ช่องว่างที่ไม่ได้ใช้ควรถูกทำให้เป็น NULL ดังนี้

1. ขนาดของไฟล์เป็นสตริงของเลขฐานสิบตามด้วยช่องว่างหนึ่งตัวถ้าขนาดไฟล์ถูกส่งโปรแกรมฝ่ายรับจะรู้ว่าตัวอักษรเพิ่มเสริมในบล็อกสุดท้าย ตัวไหนที่จะไม่สนใจ ขนาดไฟล์ไม่รวมถึงอักขระควบคุมใดๆที่ถูกใช้เพื่อบอกการจบไฟล์ (เช่น Ctrl-Z)

2. เวลาการปรับปรุงไฟล์ เป็นตัวเลขฐานแปด วัดเป็นวินาทีจากวันที่ 1 มกราคม 1970 ตามเวลามาตรฐานกรีนิช พร้อมด้วยศูนย์ถ้าวันที่ไม่ถูกต้อง ตามด้วยช่องว่างหนึ่งช่อง การใช้เวลากรีนิชหลีกเลี่ยงปัญหาที่จะเกิดขึ้นเมื่อคนที่อยู่คนละเขตเวลากำลังทำงานบนไฟล์ และไม่ชัดเจนว่าไฟล์ไหนใหม่สุด คอมพิวเตอร์ฝ่ายรับควรตั้งวันที่และเวลาของไฟล์ในไดเรกทอรี เมื่อเป็นไปได้

3. ภาวะของไฟล์เป็นสตริงของเลขฐานแปด (ระบบยูนิกซ์เท่านั้น) ตามด้วยช่องว่างหนึ่งช่อง

4. หมายเลขอนุกรม (Serial Number) เป็นสตริงของเลขฐานแปด ฟิลด์นี้ถูกทำให้เป็นศูนย์หากไม่มีหมายเลข โปรแกรมใช้หมายเลขอนุกรมเพื่อตรวจสอบการละเมิดลิขสิทธิ์ โปรแกรมฝ่ายรับสามารถตรวจสอบว่าโปรแกรมฝ่ายส่งมีหมายเลขอนุกรมเหมือนกัน และไม่ควรถูกติดตั้งบนคอมพิวเตอร์พร้อมกันสองเครื่อง

ส่วนที่เหลือของบล็อกถูกทำให้เป็น NULL ไฟล์ต่อไปถูกส่งโดยมีบล็อกชื่อไฟล์ของตัวเอง ชื่อไฟล์ที่เป็น NULL หมายถึงจบการส่งแบบแบตช์

มีข้อสังเกตคือ โฮสต์คอมพิวเตอร์และเครือข่ายจำนวนมากไม่สามารถจัดการกับบล็อกขนาด 1K ที่ต่อเนื่องกันได้

2.4.3 โปรโตคอล YMODEM-g

เป็นโปรโตคอลที่พัฒนาจาก YMODEM เนื่องจากคอมพิวเตอร์ที่กำลังส่งไฟล์โดยใช้ XMODEM หรือ YMODEM ต้องรอการตอบรับของแต่ละบล็อกที่จะส่งบล็อกต่อไป ซึ่งทำให้เสียเวลาในการส่งมาก ในบางกรณีที่มั่นใจในความเชื่อถือได้ของสื่อในการสื่อสารมากเพียงพอ ก็น่าที่จะกำจัดโอเวอร์เฮดนี้ได้ วิธีนี้อาจใช้กับการเชื่อมโยงแบบอนุกรมโดยตรง (ไม่ผ่านโมเด็ม) เมื่อใช้โมเด็มที่มีการแก้ไขข้อผิดพลาด หรือเมื่อการส่งข้อมูลเกิดขึ้นบนเครือข่ายโดยใช้แพคเกตที่รวมเอาการแก้ไขข้อผิดพลาดไว้ ในกรณีเช่นนี้อาจต้องใช้โปรโตคอล YMODEM-g

โปรโตคอล YMODEM-g คล้ายกับ YMODEM โดยมีความแตกต่างดังนี้

- คอมพิวเตอร์ฝ่ายรับส่ง G เพื่อร้องขอให้เริ่มต้นทำการส่ง แทนการส่ง C
- คอมพิวเตอร์ฝ่ายรับไม่ส่ง ACK สำหรับแพคเกตที่ถูกต้อง ถ้ามันตรวจพบข้อผิดพลาด มันจะเลิกการถ่ายโอนไฟล์ โดยการส่งอักขระ CAN ติดต่อกัน
- คอมพิวเตอร์ฝ่ายส่งไม่รอให้บล็อกหนึ่งถูกตอบรับ ก่อนจะส่งบล็อกต่อไป

2.4.4 โปรโตคอล ZMODEM

ZMODEM มีการปรับปรุงเพิ่มขึ้นจาก XMODEM และ YMODEM โดยมีข้อได้เปรียบหลักคือ สามารถถ่ายโอนข้อมูลแปดบิตบนช่องทางแบบเจ็ดบิตได้ โดยใช้การเข้ารหัสอักขระควบคุมและอักขระพิเศษตัวอื่น มันสามารถถ่ายโอนไฟล์แบบอัตโนมัติหากซอฟท์แวร์สื่อสารรู้จัก ZMODEM ตัวอย่างเช่น โปรแกรม ProComm Plus รู้จักการเริ่มต้นของการถ่ายโอนไฟล์ด้วย ZMODEM ถ้าร้องขอให้โฮสต์ส่งไฟล์ด้วย ZMODEM ProComm Plus จะเริ่มต้นการดาวน์โหลดโดยอัตโนมัติ เมื่อมันได้รับลำดับที่เริ่มต้นการส่งของ ZMODEM

2.4.5 โปรโตคอล Kermit

โปรโตคอล Kermit ได้รับการพัฒนาขึ้นที่ Columbia University เพื่อใช้ถ่ายโอนไฟล์ระหว่างเมนเฟรมคอมพิวเตอร์ และไมโครคอมพิวเตอร์ Kermit มีประโยชน์กับผู้ที่ต้องการใช้เมนเฟรมเป็นตัวเก็บข้อมูล ผู้ใช้สามารถใช้ Kermit อัปโหลดโปรแกรมที่เขียนขึ้นไปไว้ที่เมนเฟรมให้ผู้ที่ต้องการมาดาวน์โหลดไปได้ แม้ว่าโปรแกรมนั้นทำงานบนเมนเฟรมไม่ได้ก็ตาม

การใช้ Kermit

วิธีปกติในการใช้ Kermit เพื่อถ่ายโอนไฟล์กับเมนเฟรม คือเริ่มต้นด้วยโปรแกรมสื่อสารที่ทำงานบนไมโครคอมพิวเตอร์ที่มีทั้งตัวจำลองเทอร์มินัลและโปรโตคอล Kermit ใช้ไมโครเป็นเทอร์มินัล เพื่อสั่งให้เมนเฟรมใช้ Kermit ของมันเอง จากนั้นไมโครจะถูกเปลี่ยนไปเป็น Kermit Mode และการถ่ายโอนไฟล์ก็จะเกิดขึ้น มีคำสั่งของเซิร์ฟเวอร์ที่ทำให้ไมโครคอมพิวเตอร์สามารถส่งคำสั่งจำนวนหนึ่งไปยังเมนเฟรมได้ ในขณะที่ยังอยู่ใน Kermit Mode แต่ความสามารถนี้ไม่ได้ถูกสร้างขึ้นรองรับเสมอ

Kermit ไม่ค่อยต้องการความสามารถทางการสื่อสารของเครื่องทั้งสองที่ใช้มันนัก มันสามารถใช้กับระบบที่ถูกจำกัดให้ใช้กับตัวอักษรเจ็ดบิต แม้ว่าเมื่อข้อมูลจะถูกส่งในแบบแปดบิต ไม่มีแฮนด์เช็คกิ้งนอกเหนือจากที่มีให้ใน Kermit เอง การทำงานแบบฟูลดูเพล็กซ์ก็ไม่จำเป็นสำหรับการทำงานพื้นฐาน

การเข้ารหัสตัวอักษร

Kermit ต่างจาก XMODEM ตรงที่ทำการแปลงตัวอักษรที่ถูกส่งไปทุกตัวให้เป็นตัวอักษรที่พิมพ์ออกได้ (ASCII 32 ถึง 126) ด้วยวิธีนี้ อักขระที่พิมพ์ไม่ได้ก็สามารถถูกส่งไปได้

อักขระควบคุม

อักขระควบคุมคืออักขระ 0 ถึง 31 และ 127 เมื่อพบอักขระเหล่านี้ในข้อมูลที่ถูกส่ง Kermit จะแปลงพวกมันเป็นตัวอักขระที่พิมพ์ได้โดยการ XOR ด้วย 64 และนำหน้าด้วยอักขระอุปสรรค (Prifix character) ซึ่งปกติเป็น # ดังนั้น Ctrl-A ซึ่งเป็น ASCII 1 จะกลายเป็น A ซึ่งเป็น ASCII 65 และถูกส่งไปเป็น #A เมื่อโปรแกรมฝ่ายรับตรวจพบอักขระอุปสรรคมันจะทิ้งไปและแปลงอักขระที่ตามมากลับเป็นอักขระควบคุมโดย XOR ด้วย 64 ถ้าตัวอักขระอุปสรรคเองถูกส่งออกไปมันจะถูกส่งสองครั้ง เช่น ##

การแปลง 8 บิตเป็น 7 บิต

Kermit สามารถส่งตัวอักษรที่ประกอบด้วยแปดบิต (อักขระเพิ่มเติมหรือข้อมูลไบนารี) และแปลงพวกมันให้เป็นเจ็ดบิตได้ วิธีนี้จำเป็นสำหรับคอมพิวเตอร์ที่ไม่สามารถจัดการกับตัวอักษรแปดบิต Kermit จะตรวจสอบอักขระแต่ละตัว เพื่อดูว่าบิต 7 ถูกเซตหรือไม่ (อักขระที่เกิด 127) ถ้าบิต 7 ถูกเซต ตัวอักษรจะถูกส่งโดยนำหน้าด้วย & ส่วนตัวอักษรเองจะถูกส่งไปเพียงเจ็ดบิต เนื่องจากวิธีนี้ทำให้ความยาวในการส่งเพิ่มขึ้นอย่างมาก จึงควรหลีกเลี่ยงเมื่อสามารถส่งตัวอักษรแปดบิตได้

อักขระข้อมูลของ Kermit

ตัวอักษรที่ไม่ได้เป็นส่วนหนึ่งของข้อมูลที่ถูกส่งแต่เป็นข้อมูลตัวเลขภายในของ Kermit เช่น ความยาวของแพคเกต จะถูกเข้ารหัสโดยการบวกด้วย 32 (ถ้าไม่กำหนดเป็นอย่างอื่น) วิธีนี้รู้จักกันในชื่อฟังก์ชัน tochar() ตัวอย่างเช่น เมื่อบอกความยาวแพคเกตเป็น 26 เลข 26 ต้องถูกใส่ใน

ฟิลด์ LEN ของแพคเกต โดยบวก 32 เข้ากับ 26 และส่งอักขระ ASCII ที่ตรงกับผลรวมซึ่งในกรณีนี้คือ ASCII 58

ฟังก์ชันการถอดรหัสที่สอดคล้องกันคือ unchar() ฟังก์ชันนี้ถอดรหัสตัวอักษรที่รู้ว่าแทนตัวเลขภายในของ Kermit โดยการลบด้วย 32

ตัวเลขที่มากที่สุดที่ tochar() สามารถเข้ารหัสได้คือ 94 เนื่องจาก 95 บวก 32 เท่ากับ 127 ซึ่งเป็นอักขระควบคุม และอักษรใดๆที่เกิน 127 ต้องการแปดบิต ดังนั้น 94 คือ ตัวเลขพิเศษภายใน Kermit เนื่องจากมันจำกัดขนาดของพารามิเตอร์หลายตัว

อักขระที่แทนตัวเอง (เช่น เมื่ออุปกรณ์หนึ่งต้องการบอกอีกอุปกรณ์หนึ่งว่าอักขระตัวไหนที่มันต้องการใช้นำหน้าอักขระควบคุม) จะถูกส่งโดยไม่มีการเข้ารหัส

การบีบข้อมูล

เมื่อตัวอักษรที่เหมือนกันถูกส่งติดต่อกัน Kermit สามารถลดขนาดมันได้ โดยส่งเป็น ~ ตามด้วยจำนวนที่ซ้ำ และอักขระที่ซ้ำ เนื่องจากจำนวนที่ซ้ำต้องถูกเก็บในอักขระตัวเดียวที่ถูกเข้ารหัสด้วย tochar() ดังนั้นจำนวนการซ้ำสูงสุดคือ 94 อักขระควบคุมและอักขระแปดบิตสามารถซ้ำได้เมื่อมันได้ถูกเข้ารหัสเป็นอักขระสองตัว ทั้งสองตัวจะถูกส่งตามหลังจำนวนที่ซ้ำ

แพคเกต

หน่วยของข่าวสารพื้นฐานของ Kermit คือแพคเกต แพคเกตหนึ่งสามารถบรรจุข้อมูลหรือข่าวสารอื่น เช่น การตอบรับหรือข่าวสารแจ้งข้อผิดพลาด ประเภทของแพคเกตถูกแสดงไว้ข้างล่าง แต่ละประเภทถูกกำหนดด้วยอักษรใหญ่ตัวหนึ่งในฟิลด์ type ของแพคเกต

D	แพคเกตข้อมูล
Y	การตอบรับ (ACK)
N	การไม่ตอบรับ (NAK)
S	ส่งค่าเริ่มต้น (พารามิเตอร์แลกเปลี่ยน)
B	การส่งเบรก (EOT)
F	เฮดเดอร์ของไฟล์
Z	การจบไฟล์ (EOF)
E	ข้อผิดพลาด
T	สงวนไว้ใช้ภายใน
I	การเริ่มต้น (พารามิเตอร์แลกเปลี่ยน)

A	ไฟล์แอตตริบิวต์
R	รับค่าเริ่มต้น
C	คำสั่งโฮสต์ บรรจุคำสั่งของเซิร์ฟเวอร์ไปยังเมนเฟรม
K	คำสั่ง Kermit
G	คำสั่ง Generic Kermit

Kermit ใช้แพคเกตเหล่านี้ เพื่อส่งข่าวสารทุกประเภทระหว่างคอมพิวเตอร์ที่สื่อสารกัน จากสัญญาณเริ่มต้นจนถึงข้อมูลการจบไฟล์ เช่นเดียวกับข้อมูลจริง

รูปแบบแพคเกตทั่วไป

รูปแบบแพคเกตทั่วไปมีรายละเอียดดังตารางที่ 2.11 ความยาวของแต่ละแพคเกตแปรเปลี่ยนได้ และแพคเกตที่มีความยาวต่างกันสามารถถูกส่งไปในหนึ่งทรานแซกชัน (Transaction) อย่างไรก็ตาม โปรโตคอลมาตรฐานกำหนดความยาวสูงสุดของแพคเกตไว้ 96 ตัวอักษร ฟิลด์ LEN ซึ่งเป็นฟิลด์ที่สองในแพคเกต บรรจุจำนวนตัวอักษรที่ตามมารวมทั้งฟิลด์ CHECK ด้วย ดังนั้น LEN จะเป็น 94 สำหรับแพคเกตที่มีความยาวสูงสุด 96 ไบต์

ตารางที่ 2.11 รูปแบบแพคเกตของ Kermit

ออฟเซต	ค่าที่เก็บ	ความหมาย
0	MARK	กำหนดจุดเริ่มต้นแพคเกต ปกติเป็น ^A
1	LEN	จำนวนอักขระ ASCII ที่ต่อจากฟิลด์นี้
2	SEQ	หมายเลขลำดับ มอดูโล 64 เริ่มต้นที่ 0
3	TYPE	ชนิดของแพคเกต
4	DATA	ส่วนข้อมูลของแพคเกต
[end]	CHECK	Checksum

แต่ละแพคเกตมีหมายเลขลำดับตัวหนึ่ง เริ่มต้นด้วยศูนย์สำหรับแพคเกตเริ่มต้น หมายเลขจะถูกมอดูโลด้วย 64 หมายความว่า มันจะกลับเป็นศูนย์หลังจาก 63

การเริ่มต้นทรานแซกชัน

ทรานแซกชันเริ่มต้นเมื่ออุปกรณ์ฝ่ายส่งส่งแพคเกตเริ่มต้น (initation packet) ที่บรรจุพารามิเตอร์หลายอย่าง อุปกรณ์ฝ่ายรับจะตอบรับด้วยแพคเกต ACK เพื่อบอกพารามิเตอร์และตัวเลือกที่

มันสนับสนุน ในบางกรณี การแลกเปลี่ยนข้อมูลสามารถเกิดขึ้นโดยไม่ต้องตามด้วยการถ่ายโอนไฟล์ก็ได้ กรณีเช่นนี้แพคเกตเริ่มต้นถูกเรียกว่า แพคเกต Init-Info และเป็นชนิด I

แพคเกตเริ่มต้น

แพคเกตที่ฝ่ายส่งส่งไปเพื่อเริ่มต้นทรานแซกชัน และปกติถูกส่งซ้ำจนกว่าจะได้รับการตอบรับ แพกเกตนี้ประกอบด้วยอักขระพื้นฐานหกตัว และอักขระเพิ่มเติมจำนวนหนึ่งในฟิลด์ข้อมูล แพคเกตตอบรับประกอบด้วยข้อมูลคล้ายกันเกี่ยวกับอุปกรณ์ฝ่ายรับ ฟิลด์ที่ต้องการได้แก่

MAXL ความยาวสูงสุดของแพคเกตที่สามารถรับได้

TIME เวลาเป็นวินาที ที่อุปกรณ์อีกตัวควรรอก่อนที่จะรายงานว่าไทม์เอาต์

NPAD จำนวนของตัวอักษรเพิ่มเสริม (padding characters) ซึ่งควรจะนำหน้าแพคเกตที่เข้ามาแต่ละแพคเกต

PADC อักขระควบคุมที่ต้องการสำหรับการเพิ่มเสริม

EOL อักขระที่ต้องการสำหรับการจบแพคเกตเข้า ถ้ามี

QTCL อักขระที่ถูกใช้เพื่อนำหน้าอักขระควบคุม (โดยปกติคือ #)

แพคเกตเริ่มต้นสามารถบรรจุข้อมูลที่เป็นตัวเลือกเพิ่มเติมดังต่อไปนี้

QBIN อักขระที่ถูกใช้เพื่อนำหน้าอักขระที่มีบิตที่แปดถูกเซตเมื่อบิตพาริตี้ไม่สามารถนำมาใช้สำหรับข้อมูลได้

CHKT บ่งบอกประเภทของการตรวจสอบ ถ้าเป็น 1 คือใช้ Checksum 1 ตัวอักษร ถ้าเป็น 2 คือ ใช้ Checksum 2 ตัวอักษร และ 3 คือ CRC-CCITT 3 ตัวอักษร

REPT ตัวอุปสรรคที่ถูกใช้เพื่อบอกว่าเป็นอักขระซ้ำถ้าเป็นช่องว่าง (32) หมายความว่าไม่มีตัวอักษรซ้ำถ้ามีอักษรซ้ำโดยปกติจะนำหน้าด้วย ~

CAPAS บิตมาสก์ที่บ่งบอกความสามารถหลายอย่างของ Kermit แต่ละบิตถูกเซตเป็น 1 ถ้ามีความสามารถนั้น แต่ละตัวอักษรในฟิลด์ประกอบด้วยหกบิต ซึ่งถูกแปลงเป็นอักขระที่พิมพ์ออกได้

QBIN เป็นตัวอักษรที่ใช้เพื่อนำหน้าตัวอักษรที่บิตที่แปดถูกเซต เมื่อบิตพาริตี้ไม่สามารถนำมาใช้เป็นข้อมูลได้ มันต้องแตกต่างจากอักขระ QTCL การตอบสนองด้วย Y ในฟิลด์นี้เป็นการตกลงให้ทำเครื่องหมายอ้างอิงว่าเป็นแปดบิตได้หากมีการร้องขอ ส่วน N เป็นการปฏิเสธการทำเครื่องหมายอ้างอิง และ & หรืออักขระอื่นๆในช่วง 33-62 หรือ 96-126 หมายถึงทำเครื่องหมายอ้างอิงด้วยตัวอักษรนี้ โดยปกติจะใช้ &

CAPAS บรรจุข้อมูลเกี่ยวกับความสามารถที่มีการร้องขอ (ถ้าถูกส่งโดยอุปกรณ์ฝ่ายส่ง) หรือความสามารถที่มี (ถ้าถูกส่งโดยอุปกรณ์ฝ่ายรับโดยทั่วไปจะมี CAPAS เพียงหนึ่งไบต์ แต่สามารถมีมากกว่านั้นได้ โดยการเซตบิตลำดับต่ำ (low-order bit) ของไบต์

สุดท้าย ตารางที่ 2.12 แสดงรูปแบบของ CAPAS สองไบต์แรก ไบต์ที่สามและไบต์ต่อมามีให้สำหรับผู้ใช้ด้วยเช่นกัน

ตารางที่ 2.12 Kermit Capability Bytes

ฟิลด์ #	บิต	ความหมาย
A. อักษรตัวแรก		
1	5	สงวนไว้
2	4	การทำวินโดวอิ้ง (windowing)
3	3	แพคเกต (ไฟล์แอตตริบิวต์)
4	2	แพคเกตแบบยาว
5	1	สงวนไว้
	0	เป็น 1 ถ้ามี CAPAS ตามมาอีก
B. อักษรตัวที่สอง		
6	5	สงวนไว้
7	4	สงวนไว้
8	3	สำหรับผู้ใช้
9	2	สำหรับผู้ใช้
10	1	สำหรับผู้ใช้
0	0	เป็น 1 ถ้ามี CAPAS ตามมาอีก

การถ่ายโอนข้อมูล

เมื่อมีการแลกเปลี่ยนแพคเกตเริ่มต้นกันแล้ว ข้อมูลจึงถูกส่งเป็นแพคเกตติดต่อกัน อุปกรณ์ฝ่ายรับต้องสนองตอบแต่ละแพคเกตด้วยแพคเกตตอบรับ อุปกรณ์ฝ่ายส่งต้องรอให้แต่ละแพคเกตถูกตอบรับก่อนที่จะส่งแพคเกตต่อไป ถ้าคอมพิวเตอร์ฝ่ายรับรายงานข้อผิดพลาดโดยการส่งแพคเกต NAK คอมพิวเตอร์ฝ่ายส่งจะส่งแพคเกตนั้นใหม่

ถ้าได้รับแพคเกต NAK จะเหมือนกับเป็นการตอบรับแพคเกตอันก่อนถ้ามีแพคเกตหนึ่งมาถึงมากกว่าหนึ่งครั้ง ฝ่ายรับจะส่งแพคเกตตอบรับออกไปแพคเกตเดียวและไม่สนใจแพคเกตซ้ำซ้อนนั้น

ไฟล์หลายไฟล์สามารถถูกส่งไปในหนึ่งทรานแซกชันได้ สำหรับแต่ละไฟล์ Kermit จะส่งแพคเกตไฟล์เฮดเดอร์ (file header packet) แพคเกตข้อมูล และแพคเกต end-of-file Kermit บางตัวส่งแพคเกต type-A หลังจากแพคเกตไฟล์เฮดเดอร์ แพคเกตนี้บรรจุข้อมูลเพิ่มเติมเกี่ยวกับไฟล์ เช่น วันที่สร้าง ขนาด และชนิด (ตัวอย่างเช่น ASCII, EBCDIC หรือ ไบนารี) เมื่อไม่มีไฟล์ที่จะส่งแล้ว

คอมพิวเตอร์ฝ่ายส่งจะส่งแพคเกต End-of-Transmission การแลกเปลี่ยนแพคเกตมีขั้นตอนดังในตารางที่ 2.13

การขัดจังหวะการถ่ายโอน

Kermit มีคุณสมบัติที่เป็นทางเลือกเพื่อทำให้การถ่ายโอนไฟล์สามารถถูกขัดจังหวะได้ การส่งแพคเกต EOF พร้อมด้วย D ในฟิลด์ข้อมูล หมายถึง "ทิ้งไฟล์นั้น" แพคเกต EOF ยังต้องได้รับการตอบรับ การส่งแพคเกต ACK พร้อมด้วย Z ในฟิลด์ข้อมูลหมายถึง "ขัดจังหวะการส่งไฟล์ทั้งหมด"

ข้อผิดพลาดร้ายแรงที่ฝั่งใดฝั่งหนึ่งถูกแจ้งโดยแพคเกตข้อผิดพลาด ซึ่งยกเลิกทรานแซกชันนั้น แพคเกตนี้ควรถูกส่งถ้ากระบวนการขัดจังหวะไม่ทำงาน เนื่องจากอุปกรณ์อีกฝ่ายหนึ่งไม่สนับสนุนตัวเลือกในการขัดจังหวะ

ข้อมูลระหว่างแพคเกต

อักขระการยกเลิก (Terminating character) ใด ๆ ที่ระบบต้องการสามารถถูกเพิ่มเข้าที่ท้ายแพคเกต โดยปกติจะเป็น Carriage return มันจะไม่ถูกพิจารณาเป็นส่วนหนึ่งของแพคเกตตอนที่คำนวณความยาวหรือตรวจสอบทั้งหมด อักขระอื่นๆ เช่น อักขระแฮนเช็คกิ้ง สามารถถูกส่งไประหว่างแพคเกตได้เช่นกัน

ตารางที่ 2.13 ลำดับทรานเซกชันของ Kermit

เหตุการณ์	ผู้ส่งถึงผู้รับ	ผู้รับถึงผู้ส่ง
SEND INIT (ส่งการเริ่มต้น)	แพคเกตพารามิเตอร์	
ACK (การตอบรับ)		แพคเกตพารามิเตอร์
HEADER (เฮดเดอร์ของไฟล์)	แพคเกตเฮดเดอร์	
ACK		แพคเกตตอบรับ
DATA (แพคเกตข้อมูล)	แพคเกตข้อมูล	
ACK		แพคเกตตอบรับ
[วนกลับไปยัง DATA ถ้ายังมีข้อมูลที่จะส่ง]		
EOF(end-of-file)	แพคเกต EOF	
ACK		แพคเกตตอบรับ
[วนกลับไปยัง HEADER ถ้ายังมีไฟล์ที่จะส่ง]		
EOT(end-of-transaction)	แพคเกต EOT	
ACK		แพคเกตตอบรับ

การตรวจสอบข้อผิดพลาด

มีวิธีตรวจสอบข้อผิดพลาดอยู่สามวิธีในระหว่างการเริ่มต้น คอมพิวเตอร์ที่สื่อสารกันต้องตกลงวิธีที่จะใช้ โปรแกรมฝ่ายรับจะคำนวณส่วนตรวจสอบด้วยวิธีเดียวกันสำหรับแต่ละแพคเกต และส่งแพคเกต NAK ถ้าผลการคำนวณไม่ตรงกับจำนวนในฟิลด์ CHECK

สังเกตว่าแพคเกต ACK และ NAK มีส่วนตรวจสอบเช่นกัน และคอมพิวเตอร์ฝ่ายส่งควรแน่ใจว่ามันถูกต้อง อย่างไรก็ตาม ถ้าแพคเกต ACK หรือ NAK มี Checksum ไม่ถูกต้อง คอมพิวเตอร์ฝ่ายส่งจะไม่ส่ง NAK กลับไป โดยปกติมันจะทำเหมือนว่าไม่เคยได้รับแพคเกตนั้นมาก่อน

- Checksum 1 ตัวอักษร

ส่วนตรวจสอบที่ง่ายที่สุดในการคำนวณคือ Checksum 1 ตัวอักษร ซึ่งถูกคำนวณดังนี้

- บวกค่า ASCII ของตัวอักษรทั้งหมดในบล็อก ถ้าเป็นตัวอักษร 8 บิต บิตที่แปดจะถูกรวมเข้าไปด้วย
- จากนั้นบิต 6 และ 7 จะถูกบวกเข้ากับจำนวนที่สร้างจากบิต 0 ถึง 5
- ผลลัพธ์ถูกแปลงให้เป็นตัวอักษรที่พิมพ์ได้

ดังนั้นถ้า s เป็นผลรวมเลขคณิตของตัวอักษร ASCII แล้ว

check - tochar(32 + ((s +((s AND 192)/64)) AND 63))

- Checksum 2 ตัวอักษร

สำหรับ Checksum 2 ตัวอักษร ผลรวมเลขคณิตของตัวอักษรถูกคำนวณในแบบ 16 บิต และ 12 บิตล่างถูกส่งเป็นตัวอักษรสองตัว โดยใช้ฟังก์ชัน tochar() (บิต 6 ถึง 11 ตามด้วย บิต 0 ถึง 5)

- CRC 3 ตัวอักษร

CRC-16 ถูกคำนวณและถูกส่งเป็นตัวอักษรสามตัวติดต่อกัน โดยใช้ฟังก์ชัน tochar() กับบิต 12 ถึง 15, บิต 6 ถึง 11 และบิต 0 ถึง 5 ตามลำดับ

คุณสมบัติที่เป็นทางเลือก

มีคุณสมบัติที่เป็นทางเลือกอยู่สองคุณสมบัติที่ทำให้ Kermit มีประโยชน์มากขึ้นไปอีก แต่โชคไม่ดีที่มันไม่ได้ถูกรวมเข้าใน Kermit ทุกเวอร์ชันและไม่สามารถถูกใช้บนคอมพิวเตอร์ทุกประเภท แต่พวกมันกำลังมีการใช้งานเพิ่มขึ้นอย่างรวดเร็ว

1. การขยายความยาวแพคเกต

คุณสมบัติหนึ่งให้แพคเกตมีความยาวได้มากกว่าที่ถูกกำหนดไว้ดั้งเดิมการร้องขอแพคเกตแบบยาวกระทำโดยอุปกรณ์ฝ่ายส่ง ด้วยการเซตหมายเลขความสามารถเป็น 4 ซึ่งก็คือบิต 2 ของอักขระ CAPAS ตัวแรกในแพคเกตการเริ่มต้น โปรแกรมฝ่ายรับที่สามารถใช้แพคเกตแบบยาวจะตอบสนองโดยการยอมรับการร้องขอนั้น

โปรแกรมฝ่ายส่งสามารถระบุให้แพคเกตที่ยาวที่สุดของอินพุตที่มันยอมรับได้ โดยที่ความยาวถูกแจ้งในสองไบต์ คือ MAXL1 และ MAXL2

MAXL 1 บรรจุตัวอักษรที่แทนด้วย

32+(length/95)

MAXL 2 บรรจุตัวอักษรที่แทนด้วย

32+(length MOD 95)

MAXL 1 และ MAXL 2 อยู่ในไบต์ที่สองและสามของ CAPAS ไบต์สุดท้าย ถ้ามันหายไปแต่บิต long-packets ถูกเซต ความยาว 500 ไบต์จะถูกนำมาใช้ ไบต์ MAXL แบบปกติต้องถูกเซตให้มีความยาวตามโปรโตคอลดั้งเดิม เนื่องจากไม่อาจทราบได้ว่าคอมพิวเตอร์ฝ่ายรับมีความสามารถในการใช้แพคเกตแบบยาวหรือไม่ จนกว่าจะเสร็จสิ้นกระบวนการเริ่มต้น

เมื่อตกลงใช้แพคเกตแบบยาวแล้ว แต่ละแพคเกตจะถูกปฏิบัติเสมือนแพคเกตแบบยาว ถ้าฟิลด์ LEN ของมันเป็นศูนย์ รูปแบบของแพคเกตแบบยาวถูกแสดงในตารางที่ 2.14

2. สไลดิ้งวินโดวส์

ข้อต่ำหนิอย่างหนึ่งของโปรโตคอล Kermit คือมันทำงานช้าเมื่อใช้กับระยะทางไกล เนื่องจากความจำเป็นที่ต้องรอการตอบรับหลังจากการส่งแต่ละแพคเกต ซึ่งทำให้เกิดโอเวอร์เฮดมาก โดยเฉพาะอย่างยิ่งเมื่อคอมพิวเตอร์สองเครื่องอยู่ห่างไกลกัน และมีสื่อกลางอื่น เช่น เครือข่ายแทรกอยู่ตรงกลาง ในกรณีเช่นนั้น เวลารอคอยอาจเกินกว่าเวลาในการส่งได้

เพื่อแก้ปัญหานี้มีการเสนอการปรับปรุงด้วยการใช้สไลดิ้งวินโดวส์ (Sliding Windows) เพื่อทำให้แพคเกตต่อเนื่องกันถูกส่งโดยไม่มีการรอคอยการตอบรับ อย่างไรก็ตามยังคงต้องมีการตอบรับในเวลาที่เหมาะสม ระบบสื่อสารไม่ทุกระบบที่สามารถทำสไลดิ้งวินโดวส์ได้ เนื่องจากคอมพิวเตอร์แต่ละตัวต้องมีความสามารถในการสื่อสารแบบฟูลดูเพล็กซ์

ตารางที่ 2.14 แพคเกตแบบยาวของ Kermit

ออฟเซต	ค่าที่เก็บ	ความหมาย
0	MARK	กำหนดจุดเริ่มต้นแพคเกต ปกติเป็น^A
1	LEN	เซตเป็นศูนย์เพื่อบอกว่าเป็นแพคเกตแบบยาว
2	SEQ	หมายเลขลำดับ, มอดูโล 64, เริ่มต้นที่ 0
3	TYPE	ชนิดของแพคเกต
4	LENX1	ตัวอักษรคำนวณจาก (32 + INT (length/95))
3	LENX2	ตัวอักษรคำนวณจาก (32 + (lenght MODULO 95))
3	HCHECK	Checksum ของ LEN, SEQ, TYPE, LENX1, LENX2
4	DATA	เนื้อหาของแพคเกต
[end]	CHECK	Checksum ของทุกไบต์นับจาก LEN เป็นต้นไป

การร้องขอสไลดิ้งวินโดวส์

การร้องขอวินโดวส์ทำได้โดยการเซตบิต 4 ของ CAPAS ไบต์แรกในแพคเกตเริ่มต้น ในเวลาเดียวกัน ขนาดของวินโดวส์ (จำนวนของแพคเกตข้อมูลที่ถูกส่งติดต่อกัน) จะถูกใส่ไว้ในฟิลด์แรกต่อจาก CAPAS ไบต์สุดท้าย จำนวนแพคเกตสูงสุดในวินโดวส์คือ 31 จากนั้นอุปกรณ์ฝ่ายรับจึงแจ้งให้ผู้ส่งรู้ว่ามันสามารถใช้ความสามารถนี้หรือไม่

เนื่องจากจำนวนของแพคเกตในไฟล์อาจเกินจำนวนของแพคเกตในวินโดวส์ วินโดวส์จึงถูกมองว่าเลื่อนผ่านไฟล์ไปเรื่อยๆ มันเคลื่อนย้ายเมื่อแพคเกตแรกในวินโดวส์ถูกส่งไปเสร็จสมบูรณ์แล้ว (ได้รับการตอบรับจากอุปกรณ์ฝ่ายรับ) แนวความคิดของสไลดิ้งวินโดวส์ดังแสดงในรูปที่ 2.13

Packet Type	Packet Number	Sent?	ACK Received?
Init	0		
File header	1		
Data	2		
Data	3		
Data	4		
Data	5		
Data	6		
Data	7		
Data	8		
Data	9		
Data	10		
Data	11		
Data	12		
Data	13		
Data			
Data	N		
End of file	+1		

Size = 5

Current Window

Limits of Window

รูปที่ 2.13 แสดงแนวความคิดของสไลดิ้งวินโดวส์

สังเกตว่าแพคเกตแบบยาวและสไลดิ้งวินโดวส์สามารถใช้ร่วมกันได้ ดังในตารางที่ 2.15 แสดงแพคเกตเริ่มต้นการส่งที่รวมส่วนขยายทั้งสอง

ตารางที่ 2.15 แพคเกต Send-Init ที่ใช้ Long Packet และ Sliding Windows

ออฟเซต	ค่าที่เก็บ	ความหมาย
0	MARK	กำหนดจุดเริ่มต้นแพคเกต ปกติเป็น^A
1	LEN	จำนวนอักขระแอสกีหลังจากฟิลด์นี้
2	SEQ	หมายเลขลำดับ, มอดูโล 64, เริ่มต้นที่ 0
3	TYPE	ชนิดของแพคเกต (ดูด้านล่าง)
4	MAXL	ความยาวสูงสุดของแพคเกตที่สามารถรับได้

ตารางที่ 2.15 (ต่อ) แพคเกต Send-Init ที่ใช้ Long Packet และ Sliding Windows

5	TIME	เวลาไทม์เอาต์
6	NPAD	จำนวนของตัวอักษรเพิ่มเสริม
7	PADC	อักขระควบคุมที่ต้องการสำหรับการเพิ่มเสริม
8	EOL	อักขระการยกเลิก ถ้ามี
9	QTCL	อักขระที่ใช้นำอักขระควบคุม
10	QBIN	อักขระที่ใช้นำอักขระที่เกิน 127
11	CHKT	ประเภทของการตรวจสอบ
12	REPT	อักขระนำหน้าอักขระที่ซ้ำ
13	CAPAS	บิตมาสก์กำหนดความสามารถ
14	WINDW	จำนวนของแพคเกตในวินโดวส์
15	MAXL 1	ไบต์สูง (MOD 95) ของความยาวสูงสุด
16	MAXL 2	ไบต์ต่ำ (MOD 95) ของความยาวสูงสุด
17	CHECK	Checksum

การเริ่มต้นการถ่ายโอน

ผู้ส่งส่งแพคเกตไฟล์เฮดเดอร์ด้วยวิธีปกติ และรอจนกระทั่งมันได้รับการตอบรับก่อนการส่งแพคเกตข้อมูล เนื่องจากอุปกรณ์ฝ่ายรับถูกคาดการณ์ให้จัดเก็บข้อมูลลงไฟล์แต่มันไม่สามารถทำได้จนกว่าจะได้รับชื่อไฟล์ซึ่งเป็นส่วนหนึ่งของแพคเกตไฟล์เฮดเดอร์

การถ่ายโอนข้อมูล

ในแต่ละแพคเกตที่ได้รับ อุปกรณ์ฝ่ายรับจะส่ง ACK หรือ NAK ตามความเหมาะสม มันเก็บรายการแพคเกตที่ได้รับอย่างสมบูรณ์แล้วไว้ เช่นเดียวกับที่อุปกรณ์ฝ่ายส่งเก็บรายการแพคเกตที่ถูกตอบรับโดยอุปกรณ์ฝ่ายรับ

เมื่อได้รับแพคเกต NAK แพคเกตที่ถูกอ้างถึงจะถูกส่งใหม่ ถ้ามันไม่อยู่นอกวินโดวส์ในขณะนั้น

Checksum ไม่ถูกต้อง

ปัญหาหนึ่งกับระบบนี้คือ ถ้าแพคเกตถูกรับโดยมี Checksum ไม่ถูกต้องการส่ง NAK อาจสร้างความสับสน NAK อาจกำลังบอกความเสียหายในฟิลด์หมายเลขแพคเกต ซึ่งในกรณีนี้ NAK จะอ้างถึงแพคเกตผิดตัวได้

Kermit แก้ปัญหานี้โดยการให้อุปกรณ์ฝ่ายส่งไม่สนใจแพคเกต ACK หรือ NAK ที่มี Checksum ไม่ถูกต้อง ถ้าอุปกรณ์ฝ่ายรับได้รับแพคเกตข้อมูลที่มี Checksum ไม่ถูกต้อง มันควรส่ง NAK สำหรับแพคเกตที่ยังไม่ได้ตอบรับที่เก่าที่สุด หรือไม่สนใจแพคเกตที่เสีย เมื่อแพคเกตต่อไปมาถึง มันจะดูจากหมายเลขแพคเกตว่า แพคเกตใดที่หายไป และส่ง NAK สำหรับแพคเกตนั้น

การสิ้นสุดการถ่ายโอน

อุปกรณ์ฝ่ายส่งจะรอการตอบรับแพคเกตทั้งหมดก่อนที่จะส่งแพคเกต End-of-File จากนั้นจึงขอให้ตอบรับแพคเกตนั้นก่อนการส่งไฟล์ต่อไป

ไทม์เอาต์

ถ้ามีแพคเกตหายไปอาจทำให้อุปกรณ์ต้องรอคอยซึ่งกันและกัน ดังนั้นอุปกรณ์ฝ่ายส่งควรตอบสนองช่วงเวลาที่ยาวนานระหว่างการรับด้วยการส่งแพคเกตที่ไม่ได้รับการตอบรับที่เก่าที่สุดและอุปกรณ์ฝ่ายรับควรส่ง NAK สำหรับแพคเกตที่ไม่ได้ตอบรับที่เก่าที่สุด (ถ้ามี) หรือสำหรับแพคเกตที่จะมาถึงต่อไป บางทีควรจะมีขีดจำกัดของการพยายามนี้ เพื่อให้การถ่ายโอนถูกยกเลิกถ้าเกิดปัญหามากเกินไป

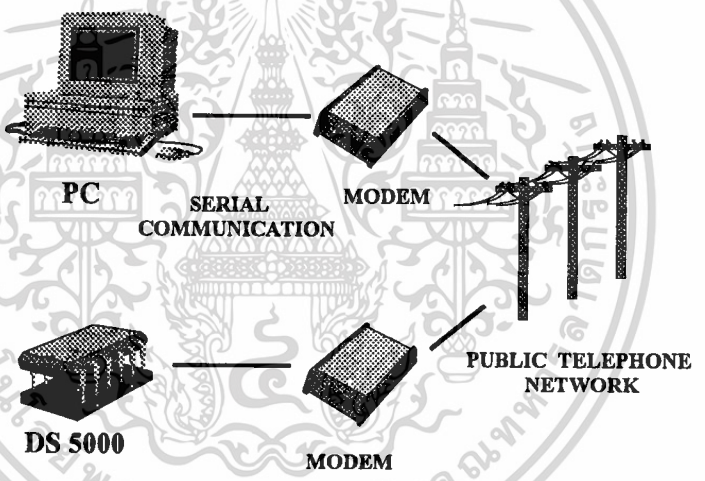
การยกเลิกการถ่ายโอนไฟล์

ผู้ส่งสามารถยุติการถ่ายโอนโดยการส่งแพคเกต End-of-File พร้อมด้วย D ในฟิลด์ข้อมูล ผู้รับสามารถหยุดการถ่ายโอนไฟล์โดยการใส่ X ในฟิลด์ข้อมูลของแพคเกต ACK มันสามารถหยุดการถ่ายโอนไฟล์ทั้งหมดโดยการใส่ Z ในฟิลด์ข้อมูล จากนั้นผู้ส่งจะส่งแพคเกต End-of-File พร้อมด้วย D ในฟิลด์ข้อมูลและหมายเลขลำดับที่มากกว่าจำนวนของ ACK ที่ร้องขอให้หยุดการถ่ายโอน

บทที่ 3
การดำเนินการวิจัย

3.1 โมดูลส่งไฟล์ไบนารีระยะไกล

โมดูลส่งไฟล์ไบนารีได้มีการออกแบบและพัฒนาโดยจะใช้ไมโครคอนโทรลเลอร์ในการที่จะทำการส่งไฟล์ให้กับคอมพิวเตอร์ที่มีการติดต่อสื่อสารในระยะทางไกล ซึ่งจะมีส่วนประกอบและลักษณะการทำงาน ดังรูปที่ 3.1



รูปที่ 3.1 แสดงส่วนประกอบและลักษณะการทำงานของระบบรับ-ส่งไฟล์ไบนารี

ในการทำงานของระบบการส่งไฟล์ไบนารีสามารถแบ่งส่วนประกอบออกได้เป็น 3 ส่วนใหญ่ ๆ ได้แก่

3.1.1 ส่วนที่ทำหน้าที่ในการรับ – ส่งไฟล์ไบนารี

เป็นส่วนของอุปกรณ์ที่ทำหน้าที่ในการรับส่งไบนารีไฟล์ ซึ่งประกอบไปด้วย

1.) อุปกรณ์ส่งไฟล์ไบนารี เป็นวงจรไมโครคอนโทรลเลอร์ DS5000 มีหน้าที่ในการถ่ายโอนข้อมูลที่เก็บมาได้จากอุปกรณ์อิเล็กทรอนิกส์เข้ามาเก็บไว้ในหน่วยความจำแบบ NVRAM ที่มีความจุขนาด 32 Kbyte เพื่อรอการถ่ายโอนข้อมูลนั้นไปยังเครื่องรับที่ทำการติดต่อเข้ามาทางการ

สื่อสารอนุกรม ในการส่งไบนารีไฟล์เครื่องส่งจะทำการส่งค่า วัน/เดือน/ปี ที่ทำการเก็บข้อมูลแล้วตามด้วยชื่อของไฟล์ที่เก็บไว้ให้เครื่องรับก่อนเพื่อทำการตรวจสอบว่าจะไม่ทำการรับไฟล์ซ้ำ จากนั้นจึงจะทำการส่งไฟล์ข้อมูลที่เก็บไว้ให้กับเครื่องรับโดยจะมีการควบคุมการส่งไฟล์ด้วยโปรโตคอลแบบ XMODEM ซึ่งรายละเอียดขั้นตอนการส่งไฟล์ด้วยโปรโตคอลแบบ XMODEM นั้นได้กล่าวไว้แล้วในบทที่ 2

นอกจากที่เครื่องส่งไฟล์ไบนารีจะทำการส่งไบนารีไฟล์ได้แล้วยังสามารถที่จะทำการรับไบนารีไฟล์และทำการจัดเก็บลงหน่วยความจำ NVRAM ที่ทำการส่งมาจากเครื่องคอมพิวเตอร์ที่ทำการติดต่อเข้ามาได้อีกด้วย ซึ่งในการรับไฟล์ไบนารีนั้นก็มีการควบคุมด้วยโปรโตคอล XMODEM ด้วยเช่นกัน

2.) อุปกรณ์รับไฟล์ไบนารี เป็นอุปกรณ์ที่ทำหน้าที่ในการรับไฟล์ข้อมูลหรือไบนารีไฟล์ที่ทำการส่งมาจากอุปกรณ์ส่งไฟล์ โดยในระบบนี้จะใช้งานเป็นไมโครคอมพิวเตอร์ที่มีการอาศัยโปรแกรมการสื่อสารอนุกรม CROSSTALK หรือ PROCOMM PLUS ซึ่งจะทำการติดต่อรับไฟล์แล้วทำการเก็บบันทึกลงในลักษณะโครงสร้างของไบนารีไฟล์

3.1.2 ส่วนที่ทำหน้าที่ในการปรับสภาพสัญญาณในการเชื่อมโยงเครือข่าย

เป็นส่วนที่ทำหน้าที่ในการปรับสภาพของสัญญาณทางลอจิกของการสื่อสารอนุกรมมาตรฐาน RS232 ให้มีสภาพของสัญญาณเหมาะสมกับตัวกลางที่มีความสามารถในการส่งผ่านคลื่นสัญญาณได้ในช่วงของความถี่ที่มีค่าจำกัดอย่างเช่นในสายโทรศัพท์ที่สามารถนำสัญญาณได้อยู่ในช่วงความถี่ตั้งแต่ 300 – 3000 Hz เท่านั้น จึงจำเป็นที่จะต้องทำการปรับสภาพของสัญญาณให้เป็นสัญญาณความถี่ในช่วงดังกล่าวก่อนทำการส่ง ส่วนฝ่ายรับก็จะต้องทำการเปลี่ยนสัญญาณความถี่นี้กลับให้เป็นสัญญาณทางลอจิกอีกครั้งซึ่งจะมีขบวนการที่ตรงข้างกับฝ่ายส่ง อุปกรณ์ที่ทำหน้าที่ในการแปลงสัญญาณนี้คือ โมเด็ม (MODEM) โดยจะมีหลักการทำงานดังได้กล่าวไว้แล้วในบทที่ 2

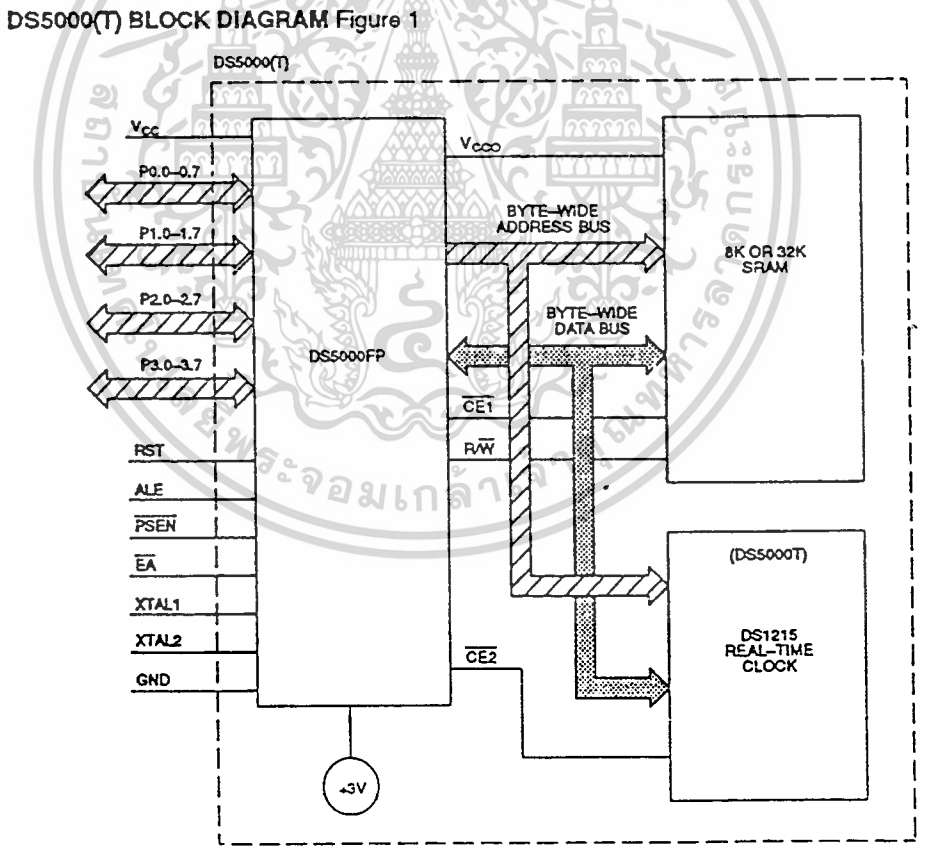
3.1.3 ส่วนที่ทำหน้าที่เป็นตัวกลางในการสื่อสาร

เป็นส่วนที่ทำหน้าที่ในการเป็นตัวกลางในการส่งผ่านข้อมูลภายในเครือข่ายการสื่อสารของส่วนส่งกับส่วนรับซึ่งในระบบการทำงานของโครงงานได้ใช้เครือข่ายระบบโทรศัพท์ที่มีให้บริการทั้งภายในและภายนอกประเทศ จึงทำให้การติดต่อสื่อสารในระยะทางไกลสามารถทำได้สะดวกยิ่งขึ้น

3.2 ลักษณะการออกแบบและขั้นตอนการทำงาน

3.2.1 รายละเอียดของวงจร

โมดูลส่งไฟล์ไบนารีสร้างขึ้นจากอุปกรณ์ ไมโครคอนโทรลเลอร์ DS5000 ซึ่งเป็นชิพไมโครคอมพิวเตอร์ที่ผลิตขึ้นโดยบริษัท DALLAS SEMICONDUCTOR โดยมีชุดคำสั่งและสถาปัตยกรรมภายในที่มีลักษณะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ผลิตโดยอินเทล โดยจะมีลักษณะของขาสัญญาณและโครงสร้างภายใน ดังรูปที่ 3.2



รูปที่ 3.2 แสดงขาสัญญาณและไดอะแกรมวงจรภายในของ DS5000

ลักษณะเด่นต่าง ๆ ของชิพ DS5000 มีดังต่อไปนี้

- หน่วยความจำ โปรแกรม /ข้อมูล ภายในมีขนาด 8 หรือ 32 Kbyte และเป็นแบบ NONVOLATILE CMOS SRAM
- รีจีสเตอร์ภายในล้วนเป็นแบบ NONVOLATILE ทั้งสิ้น
- มี real-time clock อยู่ภายในชิพ
- มีตัวนับและตั้งเวลาขนาด 16 บิต จำนวน 2 ตัว
- มีพอร์ทขนานที่สามารถทำการรับหรือส่งข้อมูลได้ 2 ทิศทาง จำนวน 4 พอร์ท ๆ ละ 8 บิต หรือสามารถใช้งานเป็นพอร์ทขนาด 1 บิตได้จำนวน 32 พอร์ท
- มีพอร์ทอนุกรมที่สามารถทำการโปรแกรมให้มีการรับ-ส่งแบบ full duplex ที่ความเร็วสูงได้
- โครงสร้างอินเทอร์รัพท์ติดต่อได้จาก 5 แหล่ง พร้อมด้วยการจัด priority ได้ 2 ระดับ
- สามารถทำการโปรแกรมได้ทั้งแบบขนานและอนุกรม

เนื่องมาจากชิพมีหน่วยความจำภายในเป็นแบบ NONVOLATILE จึงทำให้สามารถทำการเก็บรักษาข้อมูลไว้ได้นานเป็นระยะเวลา 10 ปีโดยปราศจาก Vcc ในการใช้งานชิพ DS5000 จะมีลักษณะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทั้งลักษณะของขาสัญญาณและชุดคำสั่ง แต่จะมีส่วนการใช้ที่ต่างออกไป คือ

1.) การจัดการหน่วยความจำ

ชิพ DS5000 จะมีการแบ่งหน่วยความจำภายในขนาด 8 หรือ 32 Kbyte ออกเป็น 2 ส่วนเพื่อทำหน้าที่เป็นหน่วยความจำโปรแกรมและข้อมูล ดังในรูปที่ 3.3 โดยพื้นที่ของหน่วยความจำภายในเมื่อถูกทำการแบ่งแล้ว ส่วนที่ทำหน้าที่เป็นหน่วยความจำโปรแกรมจะเป็นส่วนทางด้านล่างโดยจะทำหน้าที่ในการเก็บโปรแกรมประยุกต์ที่จะใช้งาน และสามารถใช้คำสั่งเพื่อทำการอ่านได้เพียงอย่างเดียวเท่านั้นไม่สามารถใช้คำสั่งเขียนลงไปในหน่วยความจำส่วนนี้ได้ ในส่วนของหน่วยความจำข้อมูลจะเป็นส่วนที่อยู่ทางด้านบนสามารถทำการอ่านและเขียนลงบนหน่วยความจำส่วนนี้ได้อย่างปรกติด้วยคำสั่ง MOVX

ในการแบ่งหน่วยความจำนี้ทำได้ด้วยการโปรแกรมที่รีจีสเตอร์ MCON ซึ่งเป็นรีจีสเตอร์ที่ทำหน้าที่ในการควบคุมการแบ่งหน่วยความจำโดยจะมีรายละเอียดภายในของรีจีสเตอร์ ดังนี้

MEMORY CONTROL REGISTER

D7	D6	D5	D4	D3	D2	D1	D0
PA3	PA2	PA1	PA0	RA32/8	ECE2	PAA	-

MCON.7-4 : PA3–0

Partition Address : ใช้ในการเลือกตำแหน่งที่จะทำการแบ่งพื้นที่ออกเป็นหน่วยความจำโปรแกรม และหน่วยความจำข้อมูล ซึ่งสามารถกำหนดค่าต่าง ๆ ได้ดังตารางที่ 3.1

ตารางที่ 3.1 แสดงการโปรแกรมค่า Partition Address ที่รีจีสเตอร์ MCON.7-4

PA3	PA2	PA1	PA0	Partition Address
0	0	0	0	0000H
0	0	0	1	0800H
0	0	1	0	1000H
0	0	1	1	1800H
0	1	0	0	2000H
0	1	0	1	2800H
0	1	1	0	3000H
0	1	1	1	3800H
1	0	0	0	4000H
1	0	0	1	4800H
1	0	1	0	5000H
1	0	1	1	5800H
1	1	0	0	6000H
1	1	0	1	6800H
1	1	1	0	7000H
1	1	1	1	8000H

MCON.3 : RA32/8

Range Address : ใช้เลือกค่าที่มากที่สุดของหน่วยความจำภายใน โดยเมื่อ

RA32/8 = 0 หมายความว่าชิพจะมีหน่วยความจำภายในขนาด 8 Kbyte

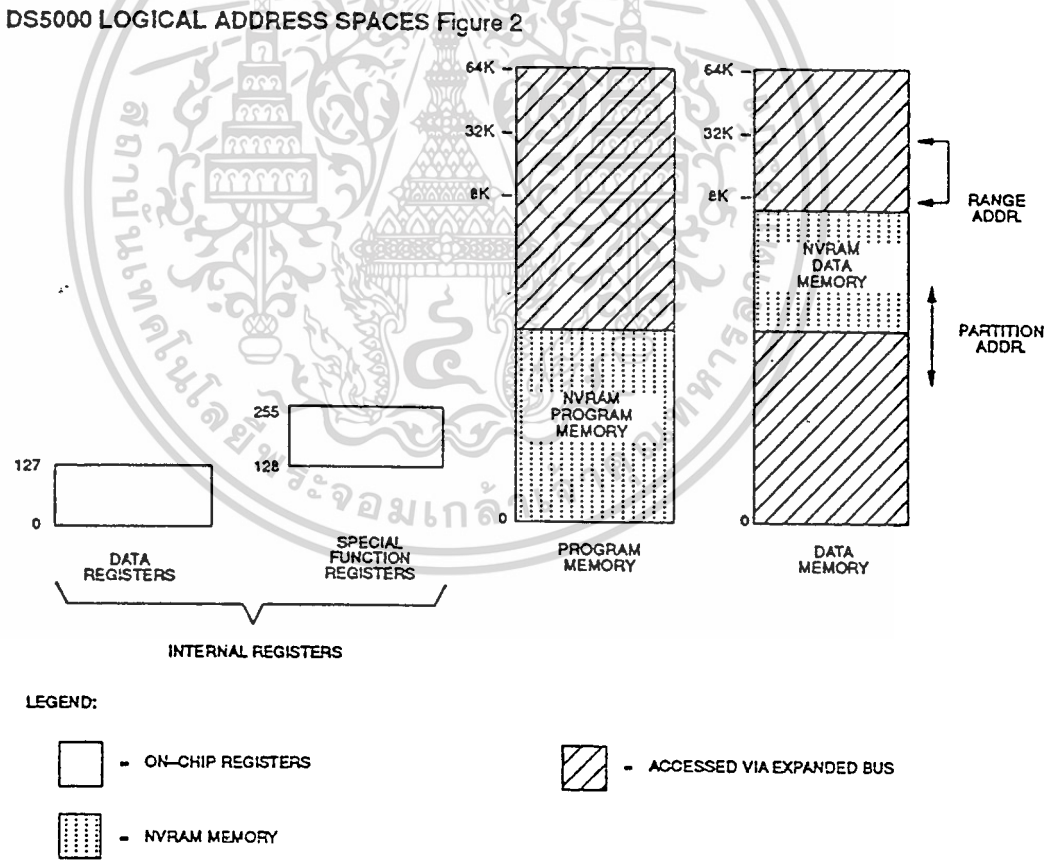
RA32/8 = 1 หมายความว่าชิพจะมีหน่วยความจำภายในขนาด 32 Kbyte

MCON.2 : ECE2

Enable Chip Enable 2 : ใช้ในการ Enable หรือ Disable สัญญาณ CE2* ที่ใช้ในสัญญาณการเพิ่มหน่วยความจำข้อมูลภายในโดยปกติจะมีค่าเป็น 0

MCON.1 : PAA

Partition Address Access : ใช้ในการควบคุมการเขียนและอ่านหน่วยความจำข้อมูล



รูปที่ 3.3 แสดงการแบ่งหน่วยความจำภายในของ DS5000

2.) การทำการโปรแกรมชิพ DS5000

ในการทำการโปรแกรมชิพ DS5000 เราสามารถทำการโปรแกรมได้ 2 วิธี

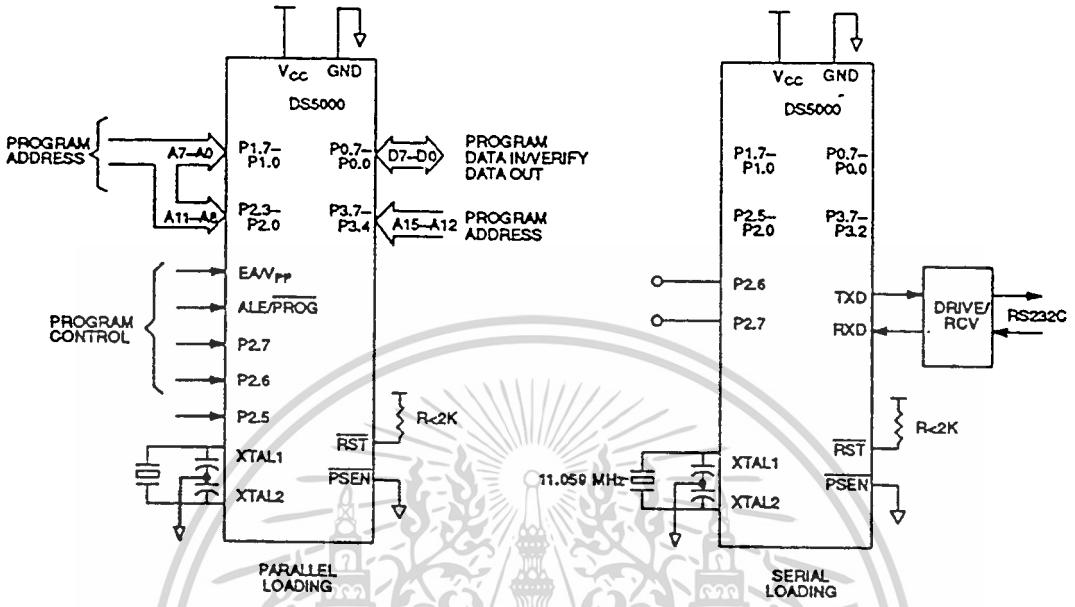
- **การโหลดโปรแกรมแบบอนุกรม** จะเป็นการโหลดโปรแกรมประยุกต์เข้ามาเก็บไว้ในหน่วยความจำโปรแกรมโดยผ่านทางพอร์ทอนุกรมมาตรฐาน RS232C ซึ่งในการโหลดโปรแกรมจะต้องทำการปรับระดับลอจิกที่ขา $\overline{RST}$ ให้เป็น high และที่ขา $\overline{PSEN}$ ให้เป็น low ดังแสดงในรูปที่ 3.4 ก. ซึ่งในการโหลดโปรแกรมประยุกต์ Serial Bootstrap Loader จะเป็นตัวจัดการการโหลดโปรแกรม

- **การโหลดโปรแกรมแบบขนาน** จะเป็นการโหลดโปรแกรมประยุกต์เข้ามาเก็บไว้ในหน่วยความจำโปรแกรมโดยผ่านทางพอร์ทขนาน แสดงดังรูปที่ 3.4 ข. โดยจะต้องทำการปรับระดับสัญญาณที่ขาต่าง ๆ เป็นตารางที่ 3.2

ตารางที่ 3.2 แสดงการปรับสัญญาณที่ขาต่าง ๆ ในการโปรแกรมแบบขนาน

MODE	RST	PSEN	PROG	EA	P2.7	P2.6	P2.5
Program	1	0	0	Vpp	1	0	X
Security Set	1	0	0	Vpp	1	1	X
Verify	1	X	X	1	0	0	X
Prog Expanded	1	0	0	Vpp	0	1	0
Verify Expanded	1	0	1	1	0	1	0
Prog MCON or Key registers	1	0	0	Vpp	0	1	1
Verify MCON registers	1	0	1	1	0	1	1

PROGRAM LOADING CONFIGURATIONS Figure 3



รูปที่ 3.4 แสดงลักษณะการโปรแกรมทั้งแบบขนานและอนุกรมของชิพ DS5000

ในการใช้งานชิพ DS5000 ทำหน้าที่เป็นเครื่องส่งไบนารีไฟล์ได้มีการใช้สัญญาณในพอร์ท 1 ในการควบคุมการติดต่อสื่อสารตามมาตรฐานการสื่อสารอนุกรม RS232C โดยได้ทำการต่อขาสัญญาณของพอร์ท 1 กับเข้ากับขาสัญญาณ RS232C และอุปกรณ์ต่าง ๆ ดังตารางที่ 3.3

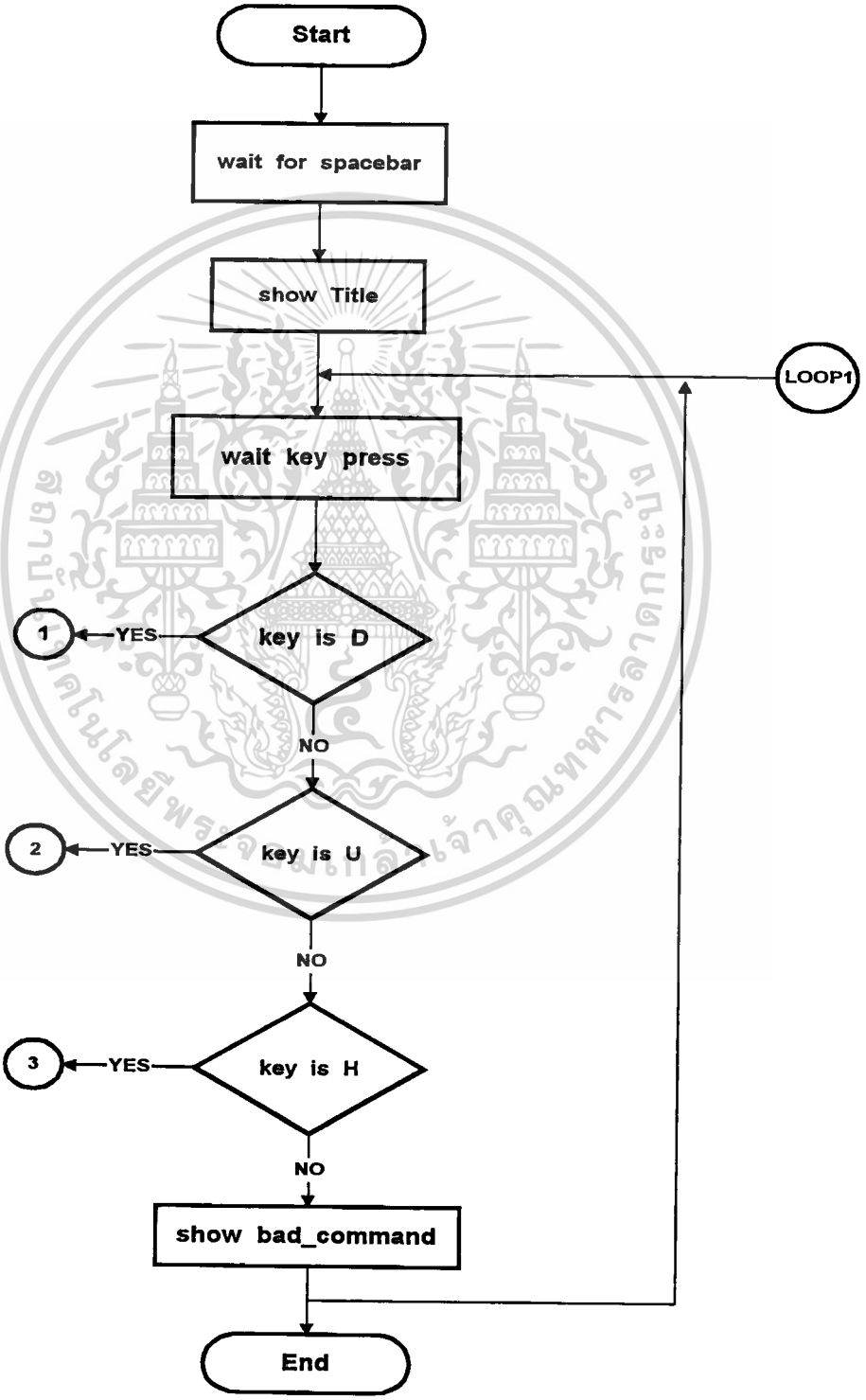
ตารางที่ 3.3 แสดงการต่อขาสัญญาณของพอร์ท 1 กับขาสัญญาณมาตรฐาน RS232C และอุปกรณ์ต่าง ๆ

ขาสัญญาณของพอร์ท 1	ขาสัญญาณ RS232C หรือ อุปกรณ์
P1.0	LED
P1.1	CD
P1.2	CTS
P1.3	DSR
P1.4	DTR
P1.5	DTS
P1.6	LED
P1.7	Not use

3.2.2 รายละเอียดของโปรแกรม

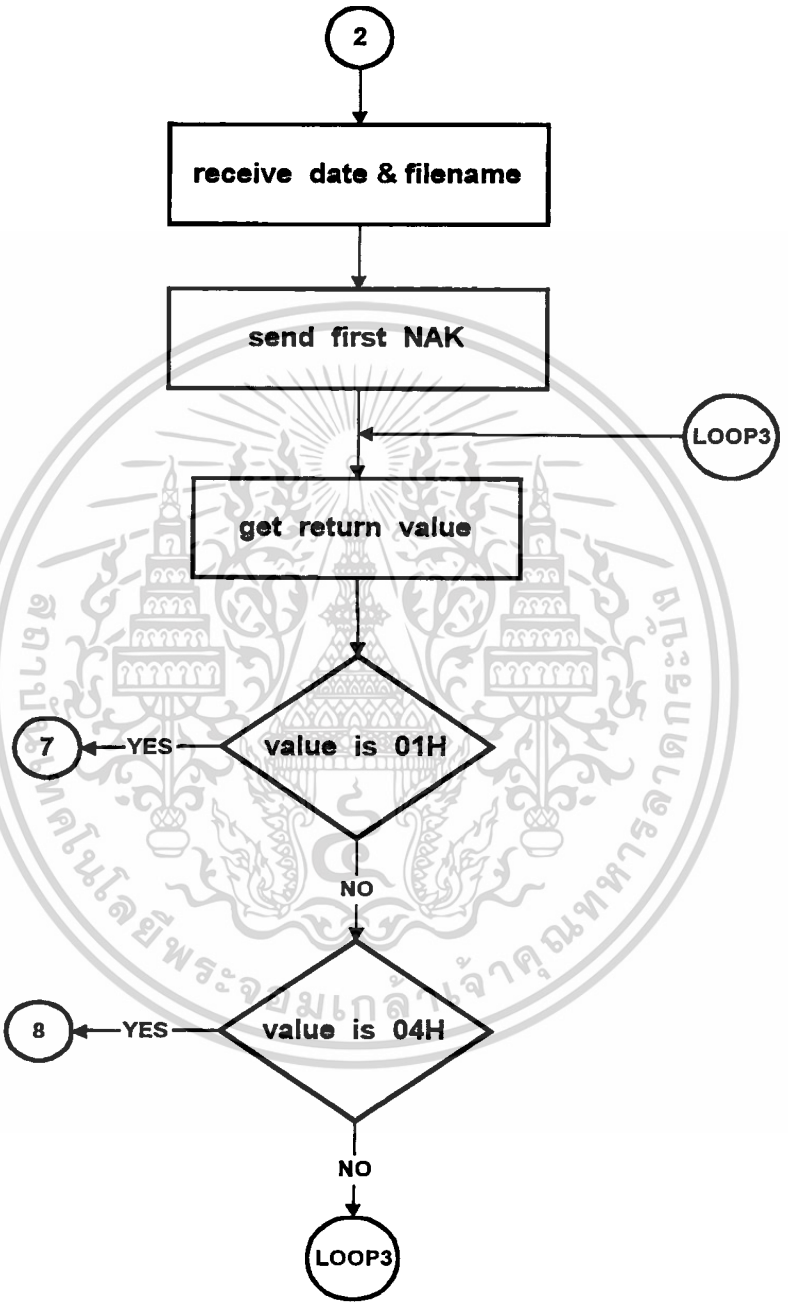
โปรแกรมในการควบคุมการรับ-ส่งไฟล์มีขั้นตอนการออกแบบและการทำงาน ดังนี้

- ส่วนของ MAIN PROGRAM



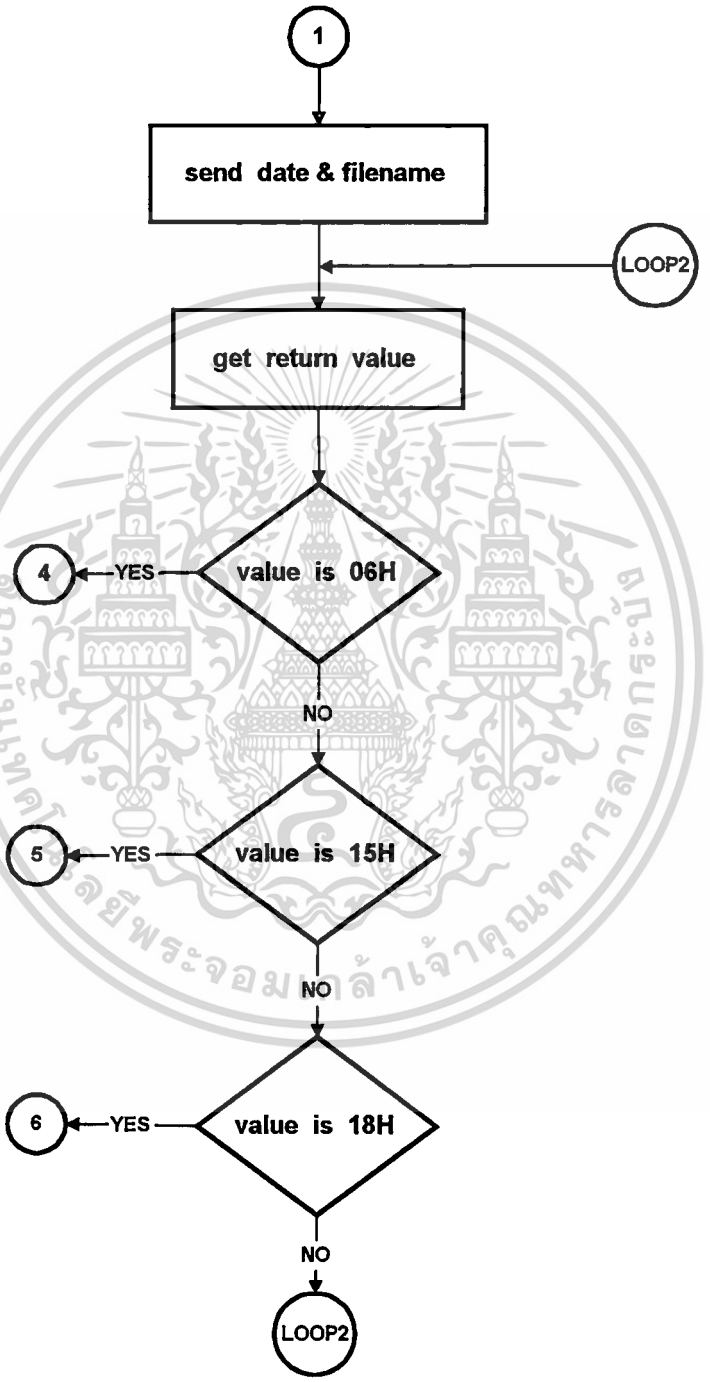
รูปที่ 3.5 แสดงขั้นตอนการทำงานในส่วนของโปรแกรมหลัก

- ส่วนของการ UPLOAD FILE



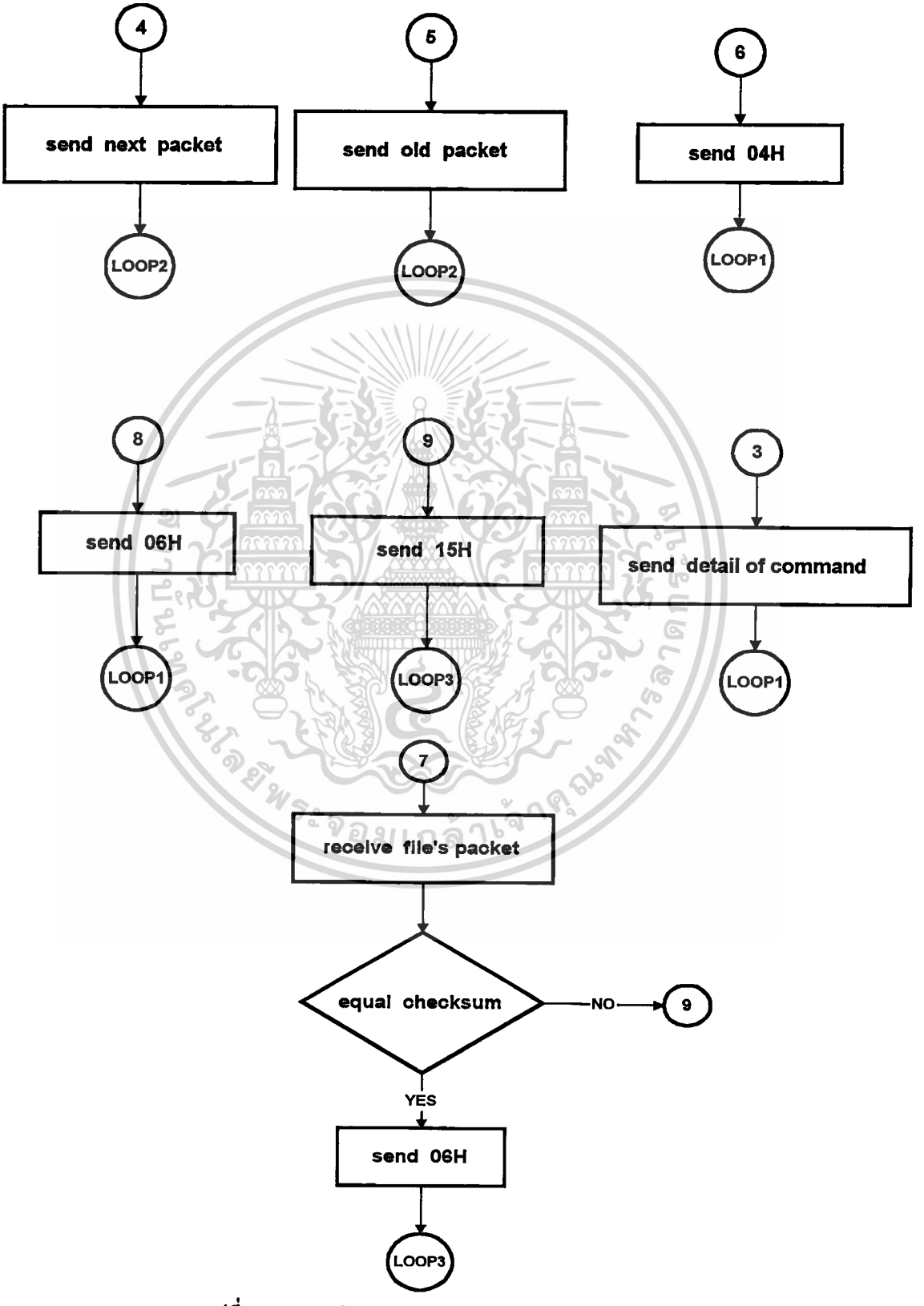
รูปที่ 3.6 แสดงขั้นตอนการทำงานในส่วนของการ UPLOAD FILE

- ส่วนของการ DOWNLOAD FILE



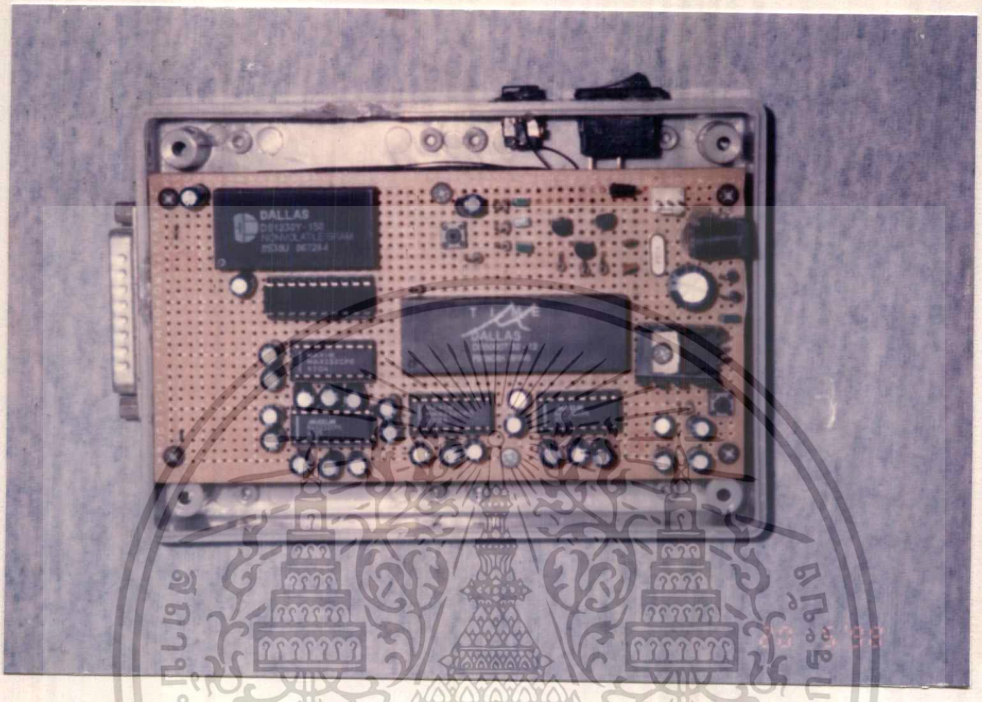
รูปที่ 3.7 แสดงขั้นตอนการทำงานในส่วนของการ DOWNLOAD FILE

- ส่วนประกอบต่าง ๆ ของโปรแกรม



รูปที่ 3.8 แสดงส่วนประกอบของโปรแกรมในส่วนต่าง ๆ

3.2.3 ลักษณะของเครื่องส่งไฟล์ไบนารี



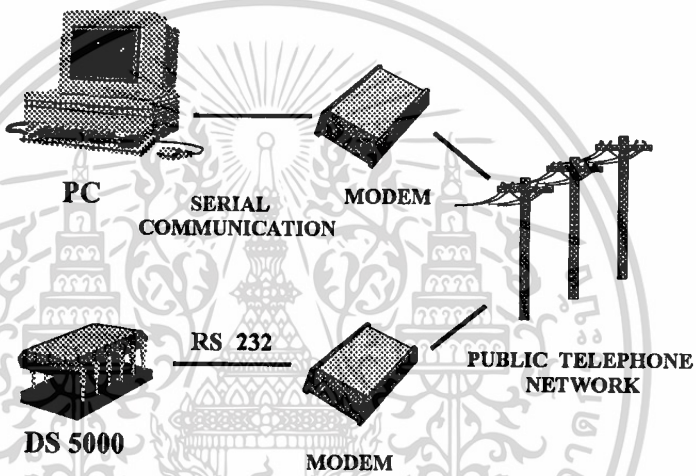
รูปที่ 3.9 แสดงเครื่องส่งไฟล์ไบนารีที่สมบูรณ์

บทที่ 4

การทดลอง

การทำงานของอุปกรณ์

ในการใช้งานโมดูลส่งไฟล์ไบนารี เราทำการต่อระบบดังแสดงในรูปที่ 4.1



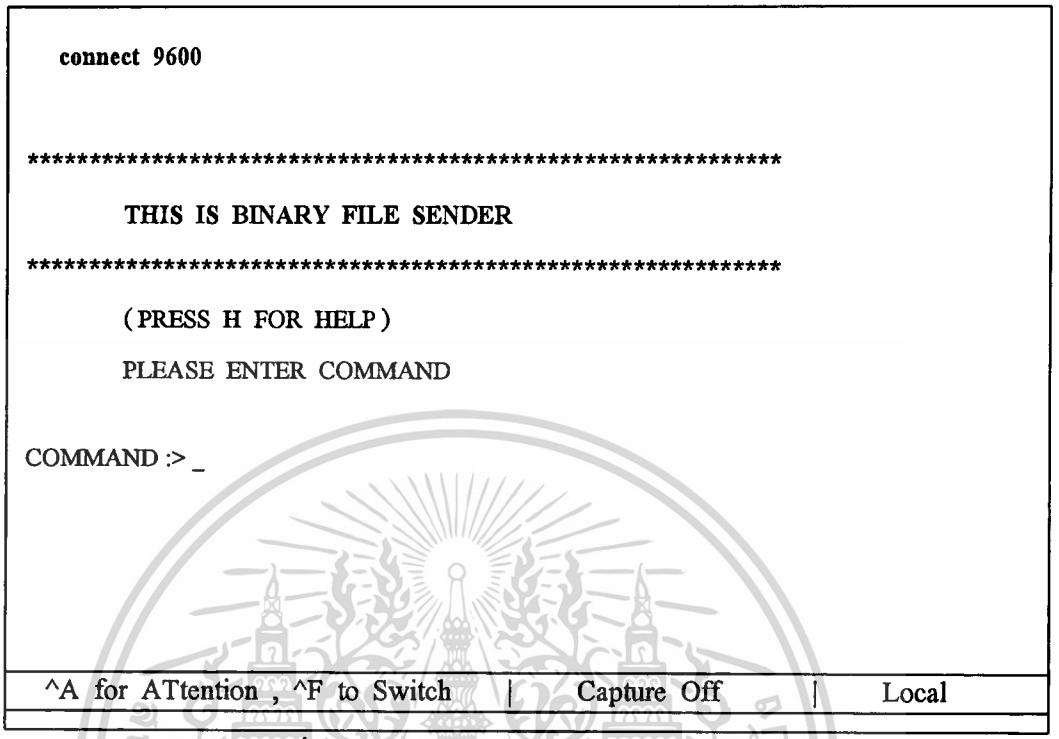
รูปที่ 4.1 แสดงอุปกรณ์และการทำงานของระบบ

โดยจะมีการแสดงขั้นตอนในการทำงาน ดังต่อไปนี้

4.1 สถานะพร้อมทำงาน

เมื่อเราทำการต่ออุปกรณ์ส่งไฟล์ไบนารีเข้ากับระบบแล้วระบบมีการเชื่อมโยงของเครือข่าย ระบบจะเข้าสู่สถานะพร้อมรับคำสั่งในการทำงานโดยแสดงจอภาพบน PC ดังรูปที่ 4.2 ซึ่งในสถานะนี้เราสามารถสั่งงานอุปกรณ์ได้ ซึ่งมีคำสั่งในการทำงาน ดังนี้

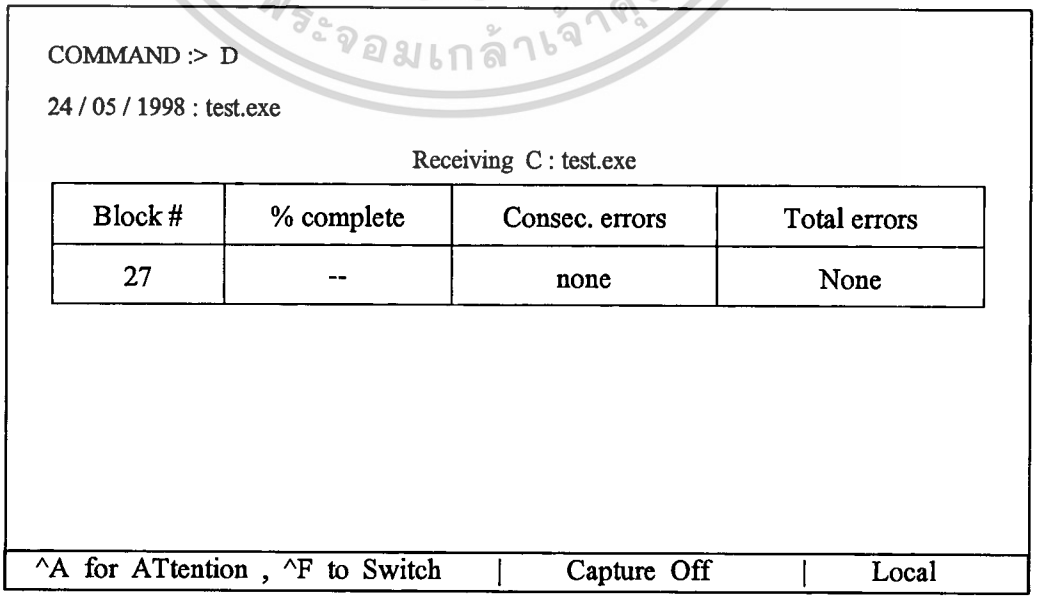
D	ทำการ DOWNLOAD ไฟล์จากหน่วยความจำของอุปกรณ์ส่งไฟล์ไบนารีมาเก็บยัง PC
U	ทำการ UPLOAD ไฟล์จาก PC ไปเก็บไว้ในหน่วยความจำของอุปกรณ์ส่งไบนารีไฟล์
H	ทำการแสดงคำสั่งต่าง ๆ พร้อมรายละเอียดในการทำงาน



รูปที่ 4.2 แสดงหน้าต่างพร้อมรับคำสั่งบนหน้าจอ PC

4.2 การ DOWNLOAD ไบนารีไฟล์

เมื่อเราทำการป้อนคำสั่ง D ให้กับอุปกรณ์ส่งไฟล์ไบนารี อุปกรณ์ส่งไฟล์ไบนารีจะทำการส่ง วัน / เดือน / ปี ของที่ทำการเก็บไฟล์และชื่อของไฟล์ที่ได้ทำการเก็บไว้ในหน่วยความจำมา

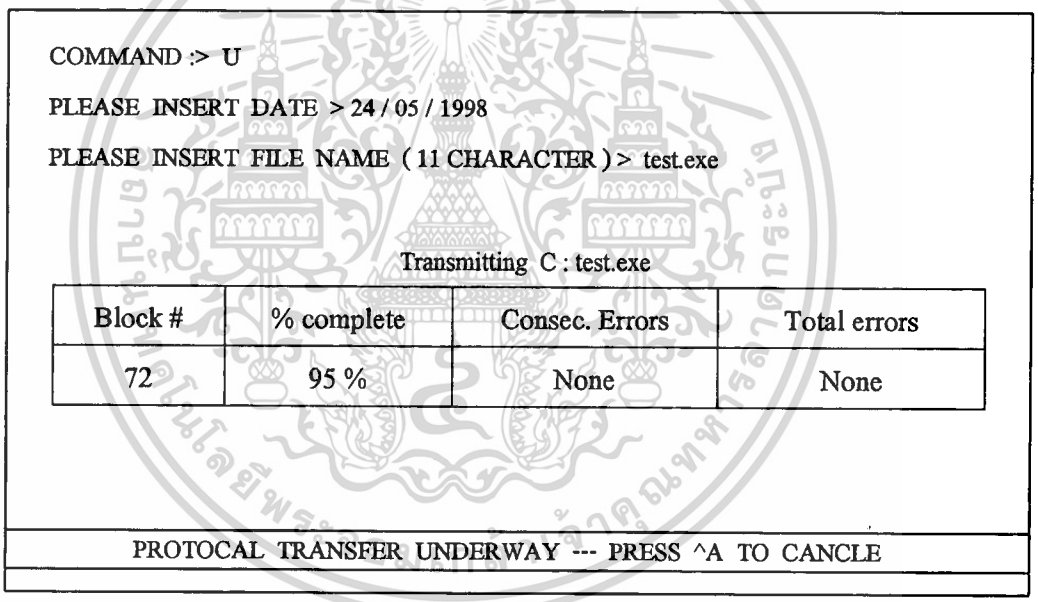


รูปที่ 4.3 แสดงหน้าต่างในการส่งไฟล์ test.exe มายัง PC

ให้ยัง PC แล้วรอจนกว่า PC เราจะเข้าสู่การ DOWNLOAD ไฟล์จึงจะทำการส่งไฟล์มาให้ ดังแสดงหน้าต่างในการ DOWNLOAD ไฟล์ดังรูปที่ 4 .3

4.3 การ UPLOAD ไบนารีไฟล์

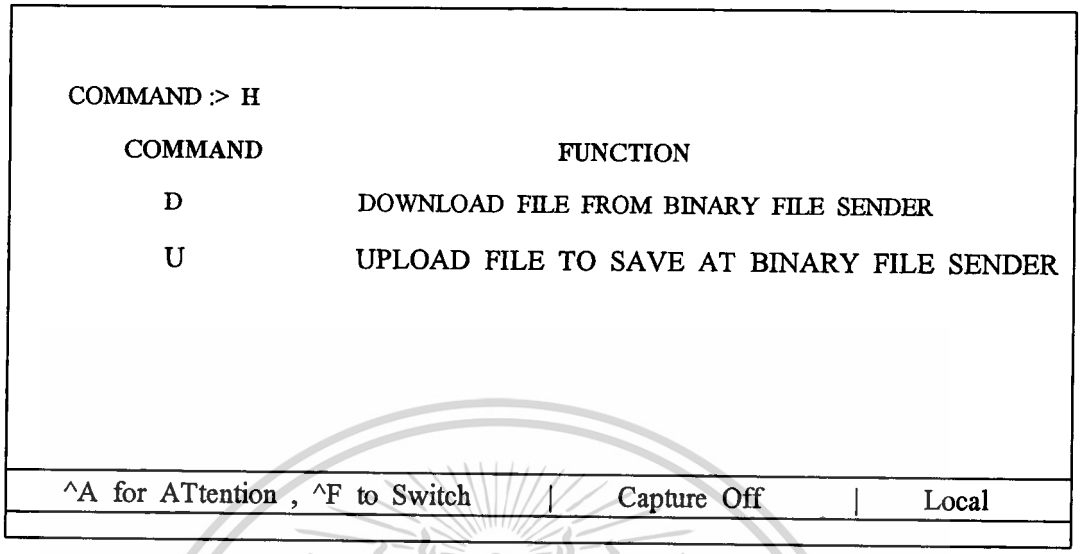
เมื่อเราทำการป้อนคำสั่ง U ให้กับอุปกรณ์ส่งไฟล์ไบนารี อุปกรณ์ส่งไฟล์ไบนารีจะทำการถาม วัน / เดือน / ปี ในการทำการ UPLOAD ไฟล์ พร้อมทั้งชื่อไฟล์ที่จะทำการ UPLOAD เพื่อจะทำการบันทึกไว้ในหน่วยความจำ แล้วจะทำการรับไฟล์ไบนารีจาก PC เพื่อนำไปเก็บไว้ในหน่วยความจำของเครื่อง ดังแสดงหน้าต่างในการ UPLOAD ดังรูปที่ 4.4



รูปที่ 4.4 แสดงหน้าต่างในการส่งไฟล์ test.exe จาก PC มายัง อุปกรณ์ส่งไบนารีไฟล์

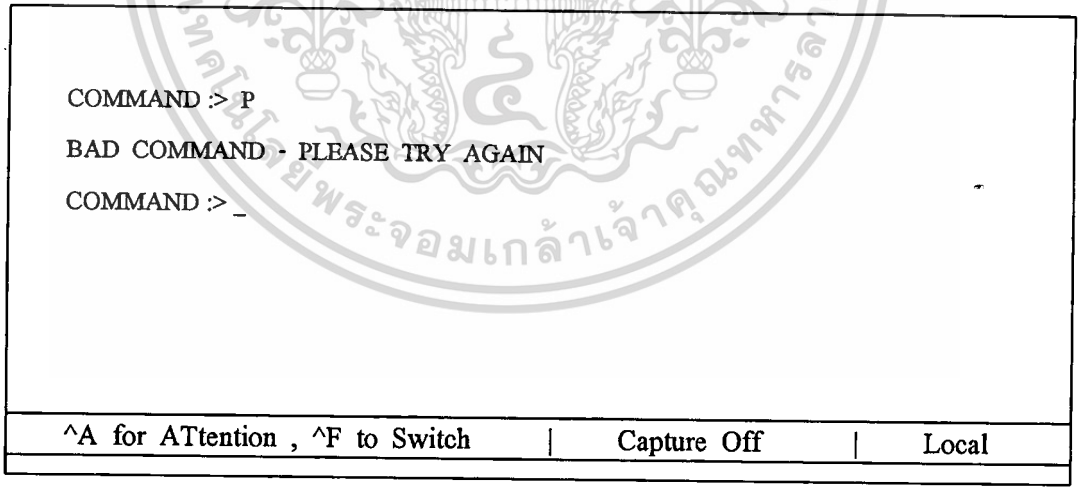
4.4 หน้าต่างช่วยเหลือ (HELP)

เมื่อทำการป้อนคำสั่ง H ให้กับอุปกรณ์ส่งไฟล์ไบนารี อุปกรณ์ส่งไฟล์ไบนารีจะทำการแสดงคำสั่งและการทำงานของคำสั่งนั้น ๆ ดังแสดงในรูปที่ 4.5



รูปที่ 4.5 แสดงหน้าต่างช่วยเหลือที่แสดงคำสั่งและหน้าที่ของคำสั่งนั้น ๆ

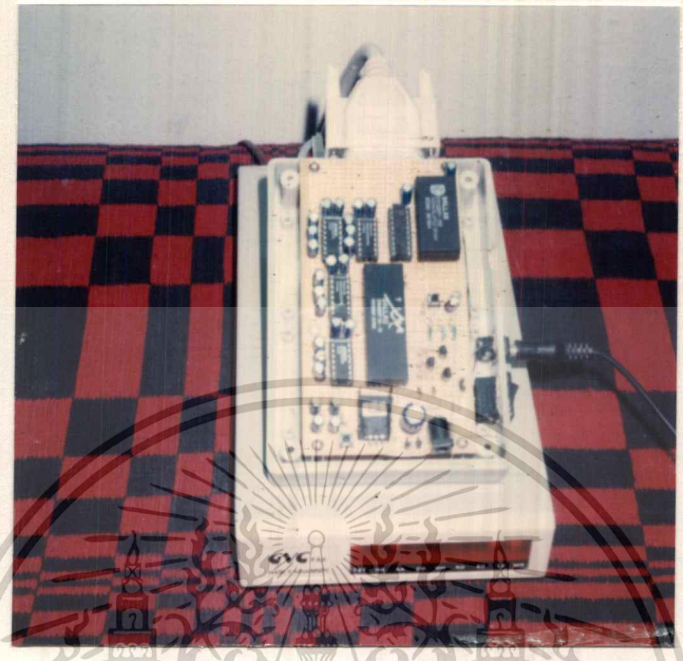
แต่ในกรณีที่ทำการป้อนคำสั่งผิดจะทำการแจ้งความผิดพลาดและให้ป้อนคำสั่งใหม่ดังแสดงในรูปที่ 4.6



รูปที่ 4.6 แสดงหน้าต่างการทำงานเมื่อมีการป้อนคำสั่งผิดพลาด

4.5 การทดลองส่งไฟล์ไบนารีระยะไกล

ในการทดลองจะทำการรับและส่งไฟล์ label.exe ระหว่างคอมพิวเตอร์กับอุปกรณ์ส่งไฟล์ไบนารี แล้วทำการรันโปรแกรมเพื่อตรวจสอบความผิดพลาด ซึ่งแสดงดังในรูปที่ 4.7



ก. แสดงการทำงานของอุปกรณ์ส่งไฟล์ไบนารีในขณะรับและส่งไฟล์

ข. แสดงการทำงานของคอมพิวเตอร์ในขณะรับและส่งไฟล์

รูปที่ 4.7 แสดงการทำการทดลองรับและส่งไฟล์ระยะไกลระหว่างคอมพิวเตอร์กับอุปกรณ์ส่งไฟล์ไบนารี

บทที่ 5
บทสรุป

5.1 สรุปผลและการประยุกต์ใช้งาน

โครงงานพิเศษโมดูลส่งไฟล์ไบนารีระยะไกล เป็นโครงงานที่จัดทำขึ้นเพื่อใช้ประโยชน์ในการส่งไฟล์ไบนารีที่เก็บมาได้จากอุปกรณ์อิเล็กทรอนิกส์จากสถานที่ต่าง ๆ มายังคอมพิวเตอร์ที่ต้องการข้อมูลนั้น โดยผ่านเครือข่ายระบบโทรศัพท์ภายใต้การควบคุมของโปรโตคอล XMODEM ทำให้สามารถลดค่าใช้จ่ายและเวลาที่จะต้องใช้ในการเดินทางไปเก็บข้อมูล ณ สถานที่เหล่านั้นลงได้

ในการส่งข้อมูล โมดูลส่งไฟล์ไบนารีสามารถทำการส่งไฟล์ที่มีขนาดไม่เกิน 32 Kbyte ได้ ซึ่งในการส่งไฟล์ภายใต้การควบคุมของโปรโตคอล XMODEM ที่มีการตรวจสอบโดยวิธี checksum พบว่าการส่งไฟล์มีประสิทธิภาพถึง 99.5 เปอร์เซ็นต์ ดังนั้นข้อมูลจึงมีโอกาสผิดพลาดน้อยมาก

5.2 แนวทางในการพัฒนาโครงงาน

เนื่องจากโปรโตคอลแบบ XMODEM เป็นโปรโตคอลที่ได้พัฒนาขึ้นมาใช้งานเป็นรุ่นแรก ดังนั้นจึงมีข้อเสีย คือช้าและมีข้อผิดพลาดมากกว่าโปรโตคอลรุ่นหลัง ดังนั้นหากมีการพัฒนาปรับปรุงใช้โปรโตคอลรุ่นใหม่ก็จะทำให้มีประสิทธิภาพในการทำงานสูงขึ้น

ภาคผนวก ก.

โปรแกรมการทำงาน

```
;   Filename        bft.asm
;   Description     send and receive binary file with XMODEM
                    protocol
;   Hardware        89c51 , DS5000T
;   Assembler       sxa51
;   Start – date    November 1997
;   Modify – date   March 1998

                ORG     0000H

BS      EQU     08H
CR      EQU     0DH
LF      EQU     0AH
EOS     EQU     10H
XON     EQU     11H
XOFF    EQU     13H
MCON    EQU     0C6H
TA      EQU     0C7H

                LJMP    MAIN

                ORG     0100H

INIT:           MOV     TMOD,#00100000B
                MOV     SCON,#01010010B
                MOV     TCON,#01101001B
                MOV     TH1,#0FDH
                SETB    TR1
                RET
```

```
SEND:         JNB     TI,SEND
              CLR     TI
              MOV     SBUF,A
              RET

RECEIVE:      JNB     RI,RECEIVE
              CLR     RI
              MOV     A,SBUF    ; GET IT
              RET

SEND_CR:      MOV     A,#CR
              LCALL   SEND
              MOV     A,#LF
              LCALL   SEND
              RET

BACK_SP:      MOV   A,#BS
              LCALL SEND
              MOV   A,#BS
              LCALL SEND
              RET

SEND_STRING:        CLR A
                    MOVC A,@A+DPTR
                    CJNE A,#EOS,SEND_STRING1
                    RET
```

```
SEND_STRING1:    PUSH DPL
                 PUSH DPH
                 LCALL  SEND
                 POP   DPH
                 POP   DPL
                 INC   DPTR
                 SJMP  SEND_STRING

FIR_NAK:     MOV    A,#15H
             LCALL  SEND
             LCALL  WAIT
             JNB    RI,FIR_NAK
             CLR    RI
             MOV    A,SBUF
             RET

COM_SEQ:     MOV    60H,R0
             ANL    60H,#0FFH
             XRL    60H,#0FFH
             MOV    R1,60H
             RET

SEND_END:    MOV    A,#04H
             LCALL  SEND
             RET
```

```
SUMC:       ADD   A,R4
            MOV   R4,A
            RET

EXR_SA:     MOV   TA,#0AAH
            MOV   TA,#055H
            MOV   MCON,#00000010B
            MOVX  @DPTR,A
            RET

EXR_RE:     MOV   TA,#0AAH
            MOV   TA,#055H
            MOV   MCON,#00000010B
            MOV   A,#00H
            MOVX  A,@DPTR
            RET

SEND_FILE:  PUSH  MCON
            MOV   A,#01H      ;SOH IS 0x01
            LCALL SEND
            CALL  SUMC
            MOV   A,R0        ;R0 IS SEQ
            LCALL SEND
            CALL  SUMC
            MOV   A,R1        ;R1 IS CSEQ
            LCALL SEND
            CALL  SUMC
```

```
SEND_FILE1:  MOV    TA,#0AAH
             MOV    TA,#055H
             MOV    MCON,#00000010B
             MOV    A,#00H
             MOVX   A,@DPTR
             LCALL  SEND
             CALL   SUMC
             INC    DPTR
             INC    R5
             CJNE   R5,#80H,SEND_FILE1
             MOV    A,R4
             LCALL  SEND
             POP    MCON
             RET

DELAY:       MOV    R6,#00H
DE:          MOV    R7,#00H
             DJNZ   R7,$
             DJNZ   R6,DE
             RET

WAIT:        MOV    R2,#0CH
WA2:         MOV    R1,#09DH
             DJNZ   R1,$
             DJNZ   R2,WA2
             RET
```

```
;*********************************************************************
;******************** MAIN PROGRAM ***********************************
;*********************************************************************

MAIN:       MOV   SP,#50H
            LCALL INIT

BOOT:       LCALL RECEIVE
            CJNE  A,#20H,BOOT
            LCALL SEND_CR
            MOV   DPTR,#TITLE1
            LCALL SEND_STRING

SHOW1:      SETB  P1.0
            SETB  P1.6
            LCALL SEND_CR
            MOV   DPTR,#TITLE2
            LCALL SEND_STRING

SHOW2:      LCALL RECEIVE
            LCALL SEND
            CJNE  A,#44H,NEXT1
            LJMP  BOOK1

NEXT1:      CJNE  A,#55H,NEXT2
            LJMP  BOOK2
```

```
NEXT2:      CJNE   A,#48H,NEXT3
            LJMP   BOOK5

NEXT3:      MOV    DPTR,#BAD_COMM
            LCALL  SEND_STRING
            LJMP   SHOW1

;*********************************************************************
;******************** DOWNLOAD FILE SECTION ******************
;*********************************************************************

BOOK1:      PUSH   MCON
            MOV    DPH,#00H
            MOV    DPL,#00H
            MOV    R7,#00H
            LCALL  SEND_CR
RT1:        LCALL  EXR_RE
            LCALL  SEND
            INC    DPTR
            INC    R7
            CJNE   R7,#0AH,RT1
            MOV    A,#3AH
            LCALL  SEND
            MOV    R7,#00H

RT2:        LCALL  EXR_RE
            LCALL  SEND
            INC    DPTR
```

```
                INC    R7
                CJNE   R7,#0EH,RT2
                POP    MCON

BOOK3:          MOV    R0,#01H
                MOV    R2,#00H
                MOV    R3,#00H
                MOV    61H,#1AH
                MOV    62H,#00H
                MOV    63H,#1AH
                MOV    64H,#00H
                MOV    DPTR,#SE_BEGIN
                LCALL  SEND_STRING

LOOP:           LCALL  RECEIVE

CHECK1:         CJNE   A,#06H,CHECK2
                LJMP   SD_NEW

CHECK2:         CJNE   A,#15H,CHECK3
                LJMP   SD_AGAIN

CHECK3:         CJNE   A,#18H,LOOP
                LJMP   EN_CON

SD_NEW:         INC    R0
                INC    R3
                MOV    61H,63H
```

```
                MOV    62H,64H

SD_AGAIN:       CLR    P1.0
                MOV    R4,#00H
                MOV    R5,#00H
                MOV    DPH,62H
                MOV    DPL,61H
                LCALL  COM_SEQ
                LCALL  SEND_FILE
                MOV    64H,DPH
                MOV    63H,DPL
                MOV    A,R3
                CJNE   A,65H,LOOP
                LCALL  SEND_END
                LCALL  RECEIVE
                LCALL  RECEIVE
EN_CON:         CLR    P1.0
                CPL    P1.0
                LCALL  DELAY
                CPL    P1.0
                LCALL  DELAY
                CPL    P1.0
                LCALL  DELAY
                CPL    P1.0
                LJMP   SHOW1
```

```
;*********************************************************************
;******************** UPLOAD FILE SECTION ************************
;*********************************************************************

BOOK2:      PUSH   MCON
            MOV    DPTR,#TI_BEGIN
            LCALL  SEND_STRING
            MOV    R7,#00
            MOV    DPH,#00
            MOV    DPL,#00H

TI1:        LCALL  RECEIVE
            CJNE   A,#0DH,TI2
            PUSH   DPH
            PUSH   DPL
            MOV    DPTR,#TI_READY
            LCALL  SEND_STRING
            POP    DPL
            POP    DPH
            POP    MCON
            LJMP   TI3

TI2:        LCALL  EXR_SA
            LCALL  SEND
            INC    R7
            INC    DPTR
            CJNE   R7,#0AH,TI1
            PUSH   DPH
```

```
                PUSH   DPL
                MOV    DPTR,#TI_READY
                LCALL  SEND_STRING

TI3:            LCALL  SEND_CR
                MOV    DPTR,#NA_BEGIN
                LCALL  SEND_STRING
                POP    DPL
                POP    DPH
                MOV    R7,#00

NA1:            LCALL  RECEIVE
                CJNE   A,#0DH,NA3

NA2:            MOV    A,#00
                LCALL  EXR_SA
                INC    R7
                INC    DPTR
                CJNE   R7,#0EH,NA2
                JMP    NA4

NA3:            LCALL  EXR_SA
                LCALL  SEND
                INC    R7
                INC    DPTR
                CJNE   R7,#0EH,NA1
```

```
NA4:        MOV   DPTR,#NA_READY
            LCALL SEND_STRING
            POP   MCON

BOOK4:      CLR   P1.6

RC_FILE:    MOV   DPTR,#RC_BEGIN
            LCALL SEND_STRING
            MOV   R3,#00H
            MOV   61H,#1AH
            MOV   62H,#00H
            MOV   63H,#00H
            MOV   65H,#00H
            PUSH  MCON

RC_LOOP:    LCALL FIR_NAK

RC_LOOP1:   MOV   DPH,62H
            MOV   DPL,61H

RC_P1:      CJNE  A,#04H,RC_P2
            LJMP  RC_END

RC_P2:      CJNE  A,#01H,RC_P3

RC_LOOP2:   MOV   R4,#00H
            MOV   R5,#00H
            LCALL SUMC
```

```
                LCALL  RECEIVE
                LCALL  SUMC
                LCALL  RECEIVE
                LCALL  SUMC

RC_LOOP3:       LCALL  RECEIVE
                CPL    P1.6
                MOV    TA,#0AAH
                MOV    TA,#055H
                MOV    MCON,#00000010B
                MOVX   @DPTR,A
                LCALL  SUMC
                INC    DPTR
                INC    R5
                CJNE   R5,#80H,RC_LOOP3
                LCALL  RECEIVE
                MOV    63H,R4
                CJNE   A,6AH,RC_LOOP4

                INC    R3
                MOV    62H,DPH
                MOV    61H,DPL
                MOV    A,#06H
                LCALL  SEND

RC_P3:          LCALL  RECEIVE
                LJMP   RC_P1
```

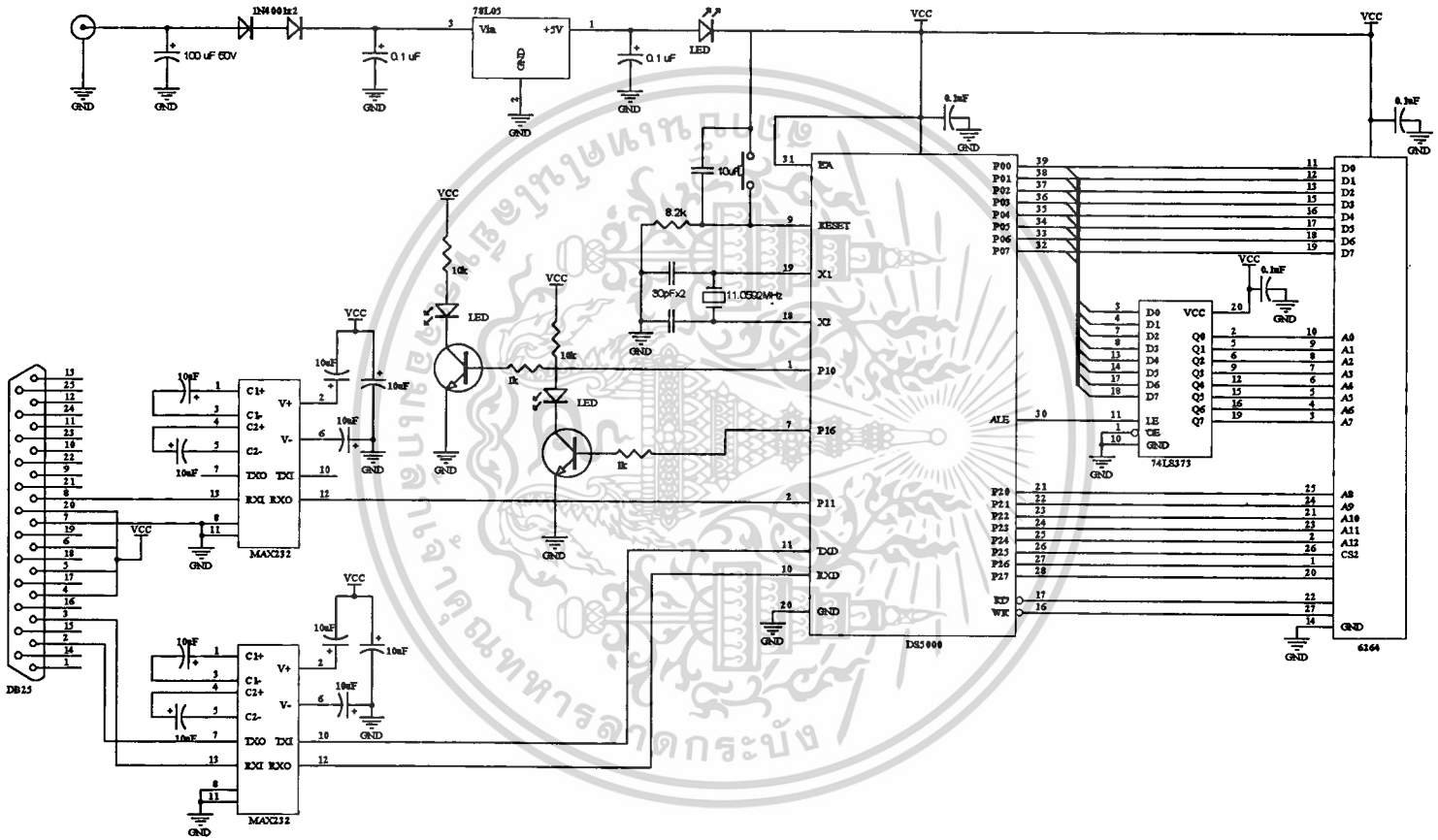
```
RC_LOOP4:   MOV   A,#15H
            LCALL  SEND
            JMP   RC_LOOP1

RC_END:     MOV   A,#06H
            LCALL  SEND
            MOV   65H,R3
            POP   MCON
            LCALL  DELAY
            LCALL  DELAY
            LCALL  DELAY
            MOV   DPTR,#RC_COMP
            LCALL  SEND_STRING
            JMP   SHOW1

;*********************************************************************
;******************** HELP SECTION ***********************************
;*********************************************************************

BOOK5:      PUSH   DPH
            PUSH   DPL
            MOV   DPTR,#H_DETAIL
            LCALL  SEND_STRING
            POP   DPL
            POP   DPH
            LJMP   SHOW1
```

```
;*********************************************************************
;******************** MESSAGE SECTION ****************************
;*********************************************************************

TITLE1:     DB CR,LF,"*************************************************************"
            DB CR,LF,"************ THIS IS BINARY FILE SENDER ************"
            DB CR,LF,"*************************************************************"
            DB CR,LF," "
            DB CR,LF,"      ( PRASS H FOR HELP ) "
            DB CR,LF,"    PLEASE ENTER COMMAND "
            DB CR,LF,EOS

TITLE2:     DB CR,LF," COMMAND :> ",EOS

BAD_COMM:   DB CR,LF," *** BAD COMMAND !!! PLEASE TRY AGAIN ***"
            DB CR,LF,EOS

SE_BEGIN:   DB CR,LF," *** YOU CAN DOWNLOAD BINARY FILE NOW ***"
            DB CR,LF," **** PLEASE COME TO RECEIVE FILE ****"
            DB CR,LF,EOS

RC_BEGIN:   DB CR,LF," *** YOU CAN UPLOAD BINARY FILE NOW ***"
            DB CR,LF," **** WAITING FOR MOVE FILE ****"
            DB CR,LF,EOS

RC_COMP:    DB CR,LF," *** READY FOR UPLOAD BINARY FILE ***"
            DB CR,LF,EOS
```

```
SE_COMP:    DB CR,LF," *** READY FOR DOWNLOAD BINARY FILE ***"
            DB CR,LF,EOS

TI_BEGIN:   DB CR,LF," PLEASE INSERT DATE > ",EOS

TI_READY:   DB CR,LF," DATE ARE SAVE SEADY ",EOS

NA_BEGIN:   DB CR,LF," PLEASE INSERT FILE NAME (11 CHARACTER)> ",EOS

NA_READY:   DB CR,LF," NAME ARE SAVE READY "
            DB CR,LF,EOS

H_DETAIL:   DB CR,LF," COMMAND          FUNCTION "
            DB CR,LF,"   -------          -------- "
            DB CR,LF,"   D    DOWNLOAD FILE FROM BINARY FILE SENDER "
            DB CR,LF,"   U    UPLOAD FILE TO SAVE AT BINARY FILE SENDER "
            DB CR,LF,EOS

END
```

ภาคผนวก ข.

วงจรการทำงาน


1N4001x2
7805
Vin
+5V
GND
100 uF 50V
0.1 uF
LED
VCC
0.1mF
10uF
8.2k
RESET
X1
X2
30pFx2
11.0592MHz
10k
1k
P10
P16
P11
TXD
RXD
ALE
EA
P00
P01
P02
P03
P04
P05
P06
P07
P20
P21
P22
P23
P24
P25
P26
P27
RD
WR
DS5000
74LS373
6264
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
CS2
OE
LE
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
DB25
MAX232
C1+
C1-
C2+
C2-
V+
V-
TXO
TXI
RXI
RXO
10nF
Title
Size
B
Number
Revision
Date: 20-Jun-1998
File: C:\PFWSCH\CIRCUIT1.SCH
Sheet of
Drawn By:

ภาคผนวก ค.
ข้อมูลอุปกรณ์

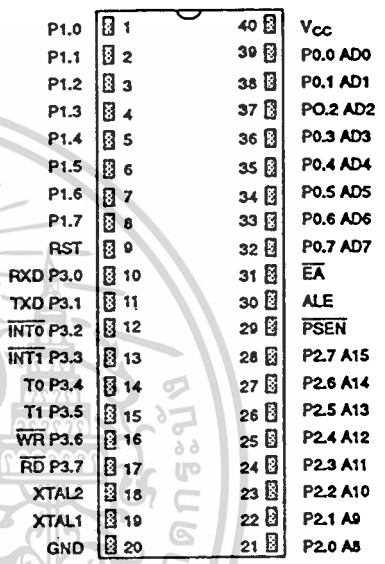
DALLAS SEMICONDUCTOR

DS5000(T)
Soft Microcontroller

FEATURES

- 8–bit 8051 compatible uC adapts to task–at–hand:
  - 8 or 32 Kbytes of nonvolatile RAM for program and/or data memory storage
  - Initial downloading of software in end system via on–chip serial port
  - Capable of modifying its own program and/or data memory in end use
- Crashproof operation:
  - Maintains all nonvolatile resources for 10 years in the absence of V_{CC}
  - Power–fail reset
  - Early warning power–fail interrupt
  - Watchdog timer
- Software Security Feature:
  - Executes encrypted software to prevent unauthorized disclosure
- On–chip, full–duplex serial I/O ports
- Two on–chip timer/event counters
- 32 parallel I/O lines
- Compatible with industry standard 8051 instruction set and pinout
- Optional Permanently Powered Real–Time Clock (DS5000T)

PIN ASSIGNMENT



40-Pin Encapsulated Package

DESCRIPTION

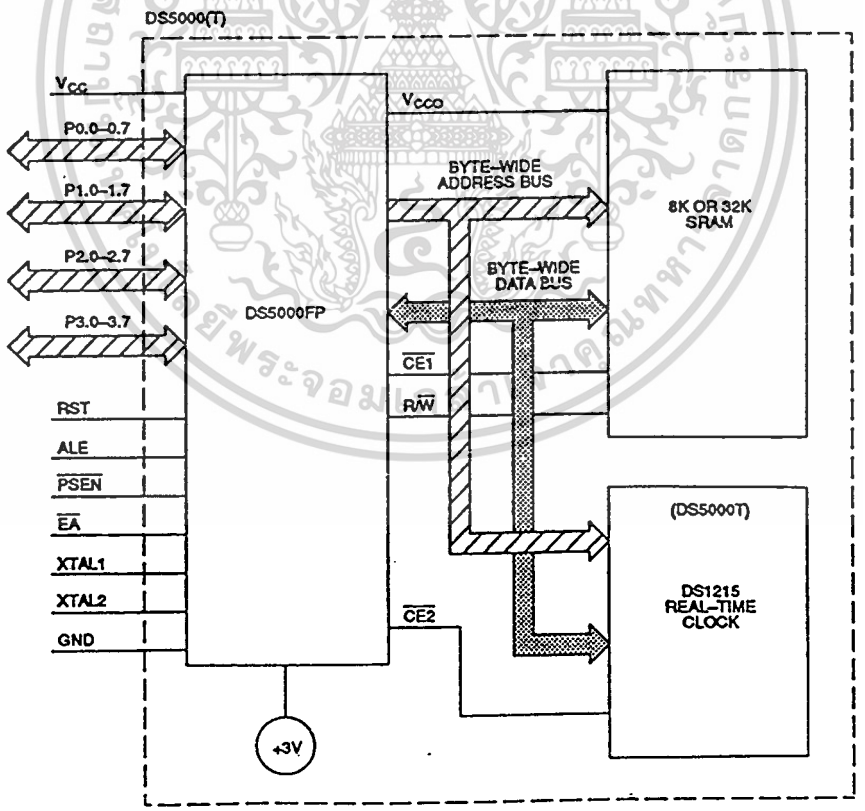
The DS5000(T) Soft Microcontroller is a fully 8051 compatible 8–bit CMOS microcontroller that offers "softness" in all aspects of its application. This is accomplished through the comprehensive use of nonvolatile technology to preserve all information in the absence of system V_{CC}. The internal program/data memory space is implemented using either 8K or 32K bytes of nonvolatile CMOS SRAM. Furthermore, internal data registers and key configuration registers are also nonvolatile. An optional real–time clock gives permanently powered timekeeping. The clock keeps time to a hundredth of a second using an on–board crystal.

ORDERING INFORMATION

PART NUMBER	RAM SIZE	MAX CRYSTAL SPEED	TIMEKEEPING?
DS5000–8–8	8K bytes	8 MHz	No
DS5000–8–12	8K bytes	12 MHz	No
DS5000–8–16	8K bytes	16 MHz	No
DS5000–32–8	32K bytes	8 MHz	No
DS5000–32–12	32K bytes	12 MHz	No
DS5000–32–16	32K bytes	16 MHz	No
DS5000T–8–8	8K bytes	8 MHz	Yes
DS5000T–8–12	8K bytes	12 MHz	Yes
DS5000T–8–16	8K bytes	16 MHz	Yes
DS5000T–32–8	32K bytes	8 MHz	Yes
DS5000T–32–12	32K bytes	12 MHz	Yes
DS5000T–32–16	32K bytes	16 MHz	Yes

Operating information is contained in the User's Guide section of the Soft Microcontroller Data Book. This data sheet provides ordering information, pinout, and electrical specification.

DS5000(T) BLOCK DIAGRAM Figure 1



PIN DESCRIPTION

PIN NUMBER	DESCRIPTION
1–8	P1.0 – P1.7 General purpose I/O Port 1
9	RST Active high reset input. A logic 1 applied to this pin will activate a reset state. This pin is pulled down internally so this pin can be left unconnected if not used.
10	P3.0 RXD General purpose I/O port pin 3.0. Also serves as the receive signal for the on board UART. This pin should not be connected directly to a PC COM port.
11	P3.1 TXD General purpose I/O port pin 3.1. Also serves as the transmit signal for the on board UART. This pin should not be connected directly to a PC COM port.
12	P3.2 $\overline{\text{INT0}}$ General purpose I/O port pin 3.2. Also serves as the active low External Interrupt 0.
13	P3.3 $\overline{\text{INT1}}$ General purpose I/O port pin 3.3. Also serves as the active low External Interrupt 1.
14	P3.4 T0 General purpose I/O port pin 3.4. Also serves as the Timer 0 input.
15	P3.5 T1 General purpose I/O port pin 3.5. Also serves as the Timer 1 input.
16	P3.6 $\overline{\text{WR}}$ General purpose I/O port pin. Also serves as the write strobe for Expanded bus operation.
17	P3.7 $\overline{\text{RD}}$ General purpose I/O port pin. Also serves as the read strobe for Expanded bus operation.
18, 19	XTAL2, XTAL1 Used to connect an external crystal to the internal oscillator. XTAL1 is the input to an inverting amplifier and XTAL2 is the output.
20	GND Logic ground.
21–28	P2.0–P2.7 General purpose I/O Port 2. Also serves as the MSB of the Expanded Address bus.

PIN NUMBER	DESCRIPTION
29	$\overline{PSEN}$ Program Store Enable. This active low signal is used to enable an external program memory when using the Expanded bus. It is normally an output and should be unconnected if not used. $\overline{PSEN}$ also is used to invoke the Bootstrap Loader. At this time, $\overline{PSEN}$ will be pulled down externally. This should only be done once the DS5000 is already in a reset state. The device that pulls down should be open drain since it must not interfere with $\overline{PSEN}$ under normal operation.
30	ALE Address Latch Enable. Used to de–multiplex the multiplexed Expanded Address/Data bus on Port 0. This pin is normally connected to the clock input on a '373 type transparent latch. When using a parallel programmer, this pin also assumes the $\overline{PROG}$ function for programming pulses.
31	$\overline{EA}$ External Access. This pin forces the DS5000 to behave like an 8031. No internal memory (or clock) will be available when this pin is at a logic low. Since this pin is pulled down internally, it should be connected to +5V to use NVRAM. In a parallel programmer, this pin also serves as V_{PP} for super voltage pulses.
32–39	P0.7–P0.0 General purpose I/O Port 0. This port is open–drain and can not drive a logic 1. It requires external pull–ups. Port 0 is also the multiplexed Expanded Address/Data bus. When used in this mode, it does not require pull–ups.
40	V_{CC} +5 volts.

INSTRUCTION SET

The DS5000 executes an instruction set which is object code compatible with the industry standard 8051 microcontroller. As a result, software development packages which have been written for the 8051 are compatible with the DS5000, including cross–assemblers, high–level language compilers, and debugging tools.

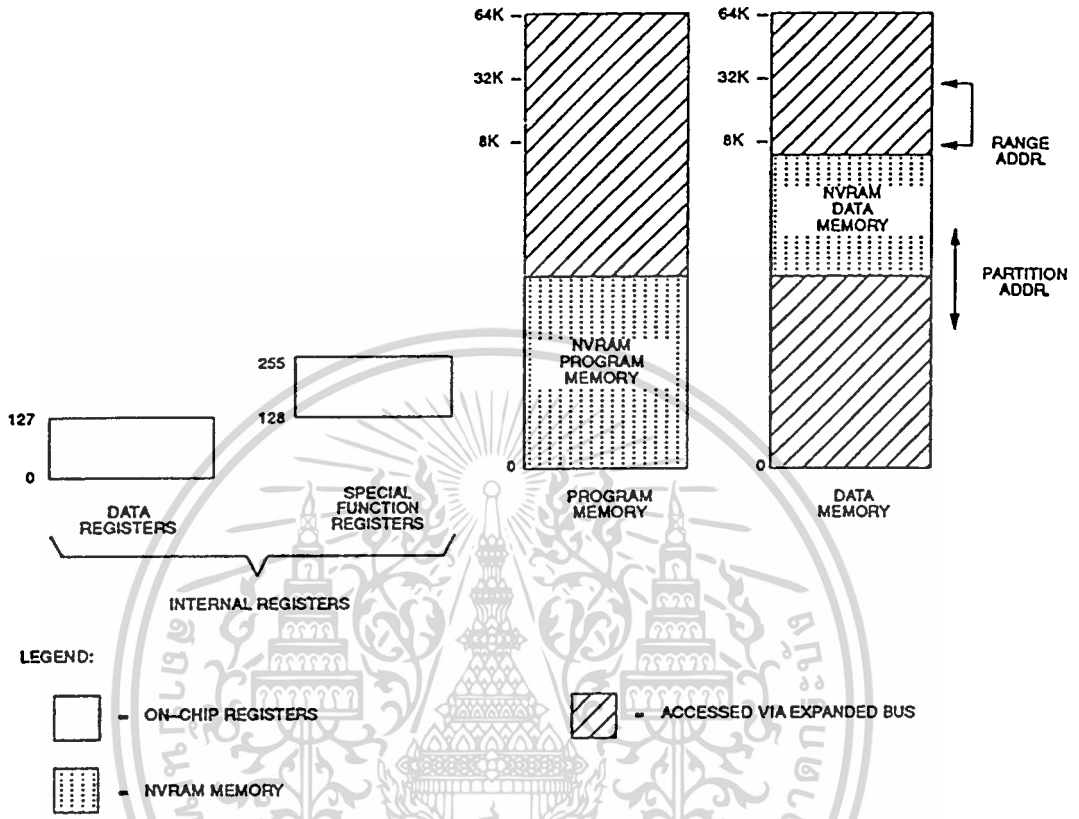
A complete description for the DS5000 instruction set is available in the User's Guide section of the Soft Microcontroller Data Book.

MEMORY ORGANIZATION

Figure 2 illustrates the address spaces which are accessed by the DS5000. As illustrated in the figure, separate address spaces exist for program and data memory. Since the basic addressing capability of the machine is 16 bits, a maximum of 64 Kbytes of program memory and 64 Kbytes of data memory can be accessed by the DS5000 CPU. The 8K or 32K byte RAM area inside of the DS5000 can be used to contain both program and data memory.

The Real–time Clock (RTC) in the DS5000T is reached in the memory map by setting a SFR bit. The MCON.2 bit (ECE2) is used to select an alternate data memory map. While ECE2=1, all MOVXs will be routed to this alternate memory map. The real–time clock is a serial device that resides in this area. A full description of the RTC access and example software is given in the User's Guide section of the Soft Microcontroller Data Book. If the ECE2 bit is set on a DS5000 without a timekeeper, the MOVXs will simply go to a nonexistent memory. Software execution would not be affected otherwise.

DS5000 LOGICAL ADDRESS SPACES Figure 2



PROGRAM LOADING

The Program Load Modes allow initialization of the NVRAM Program/Data Memory. This initialization may be performed in one of two ways:

1. Serial Program Loading which is capable of performing Bootstrap Loading of the DS5000(T). This feature allows the loading of the application program to be delayed until the DS5000(T) is installed in the end system.
2. Parallel Program Load cycles which perform the initial loading from parallel address/data information presented on the I/O port pins. this mode is timing-set compatible with the 8751H microcontroller programming mode.

The DS5000 is placed in its Program Load configuration by simultaneously applying a logic 1 to the RST pin and forcing the $\overline{\text{PSEN}}$ line to a logic 0 level. Immediately following this action, the DS5000 will look for a parallel Program Load pulse, or a serial ASCII carriage return (0DH) character received at 9600, 2400, 1200, or 300 bps over the serial port.

The hardware configurations used to select these modes of operation are illustrated in Figure 3.

PROGRAM LOADING CONFIGURATIONS Figure 3

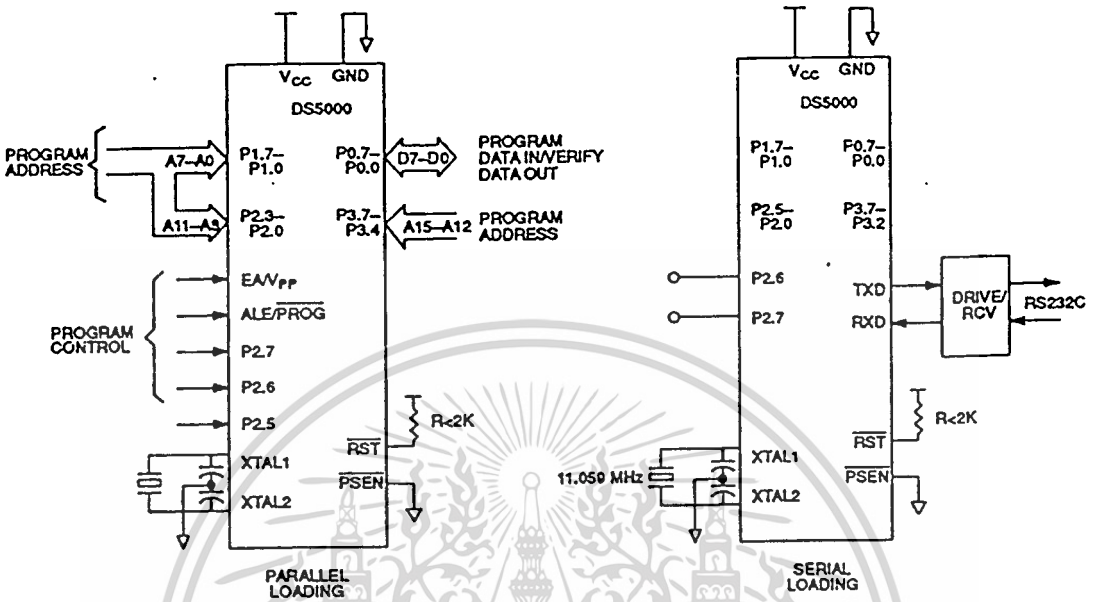


Table 1 summarizes the selection of the available Parallel Program Load cycles. The timing associated with these cycles is illustrated in the electrical specs.

SERIAL BOOTSTRAP LOADER

The Serial Program Load Mode is the easiest, fastest, most reliable, and most complete method of initially loading application software into the DS5000 nonvolatile RAM. Communication can be performed over a standard asynchronous serial communications port. A typical application would use a simple RS232C serial interface to program the DS5000 as a final production procedure. The hardware configuration which is required for the Serial Program Load mode is illustrated in Figure 3. Port pins 2.7 and 2.6 must be either open or pulled high to avoid placing the DS5000 in a parallel load cycle. Although an 11.0592 MHz crystal is shown in Figure 3, a variety of crystal frequencies and loader baud rates are supported, shown in Table 2. The serial loader is designed to operate across a three–wire interface from a standard UART. The receive, transmit, and ground wires are all that are necessary to establish communication with the DS5000.

The Serial Bootstrap Loader implements an easy–to–use command line interface which allows an application program in an Intel hex representation to be loaded into and read back from the device. Intel hex is the typical format which existing 8051 cross–assemblers output. The serial loader responds to single character commands which are summarized below:

COMMAND	FUNCTION
C	Return CRC–16 checksum of embedded RAM
D	Dump Intel Hex File
F	Fill embedded RAM block with constant
K	Load 40–bit Encryption Key
L	Load Intel Hex File
R	Read MCON register
T	Trace (Echo) incoming Intel Hex data
U	Clear Security Lock
V	Verify Embedded RAM with incoming Intel Hex
W	Write MCON register
Z	Set Security Lock
P	Put a value to a port.
G	Get a value from a port.

PARALLEL PROGRAM LOAD CYCLES Table 1

MODE	RST	$\overline{PSEN}$	$\overline{PROG}$	$\overline{EA}$	P2.7	P2.6	P2.5
Program	1	0	0	V_{PP}	1	0	X
Security Set	1	0	0	V_{PP}	1	1	X
Verify	1	X	X	1	0	0	X
Prog Expanded	1	0	0	V_{PP}	0	1	0
Verify Expanded	1	0	1	1	0	1	0
Prog MCON or Key registers	1	0	0	V_{PP}	0	1	1
Verify MCON registers	1	0	1	1	0	1	1

The Parallel Program Cycle is used to load a byte of data into a register or memory location within the DS5000. The Verify Cycle is used to read this byte back for comparison with the originally loaded value to verify proper loading. The Security Set Cycle may be used to enable and the Software Security feature of the DS5000. One may also enter bytes for the MCON register or for the five encryption registers using the Program MCON cycle. When using this cycle, the absolute register address must be presented at Ports 1 and 2 as in the normal program cycle (Port 2 should be 00H). The MCON contents can likewise be verified using the Verify MCON cycle.

When the DS5000 first detects a Parallel Program Strobe pulse or a Security Set Strobe pulse while in the Program Load Mode following a Power On Reset, the internal hardware of the DS5000 is initialized so that an existing 4 Kbyte program can be programmed into a DS5000 with little or no modification. This initialization automatically sets the Range Address for 8 Kbytes and maps the lowest 4 Kbyte bank of Embedded RAM as program memory. The next 4 Kbytes of Embedded RAM are mapped as Data Memory.

In order to program more than 4 Kbytes of program code, the Program/Verify Expanded cycles can be used. Up to 32 Kbytes of program code can be entered and verified. Note that the expanded 32 Kbyte Program/Verify cycles take much longer than the normal 4 Kbyte Program/Verify cycles.

A typical parallel loading session would follow this procedure. First, set the contents of the MCON register with the correct range and partition only if using expanded programming cycles. Next, the encryption registers can be loaded to enable encryption of the program/data memory (not required). Then, program the DS5000 using either normal or expanded program cycles and check the memory contents using Verify cycles. The last operation would be to turn on the security lock feature by either a Security Set cycle or by explicitly writing to the MCON register and setting MCON.0 to a 1.

SERIAL LOADER BAUD RATES FOR DIFFERENT CRYSTAL FREQUENCIES Table 2

CRYSTAL FREQ (MHz)	BAUD RATE					
	300	1200	2400	9600	19200	57600
14.7456		Y	Y	Y	Y	
11.0592	Y	Y	Y	Y	Y	Y
9.21600	Y	Y	Y	Y		
7.37280	Y	Y	Y	Y		
5.52960	Y	Y	Y	Y		
1.84320	Y	Y	Y	Y		

ADDITIONAL INFORMATION

A complete description for all operational aspects of the DS5000, is provided in the User's Guide section of the Soft Microcontroller Data Book.

DEVELOPMENT SUPPORT

Dallas Semiconductor offers a kit package for developing and testing user code. The DS5000TK Evaluation Kit allows the user to download Intel hex formatted code directly to the DS5000 from a PC–XT/AT or compatible computer. The kit consists of a DS5000T–32–12, an interface pod, demo software, and an RS232 connector that attaches to the COM1 or COM2 serial port of a PC. See the User's Guide for further details.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.3V to 7.0V
Operating Temperature	0°C to +70°C
Storage Temperature	–40°C to 70°C
Soldering Temperature	260°C for 10 seconds

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

DC CHARACTERISTICS

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

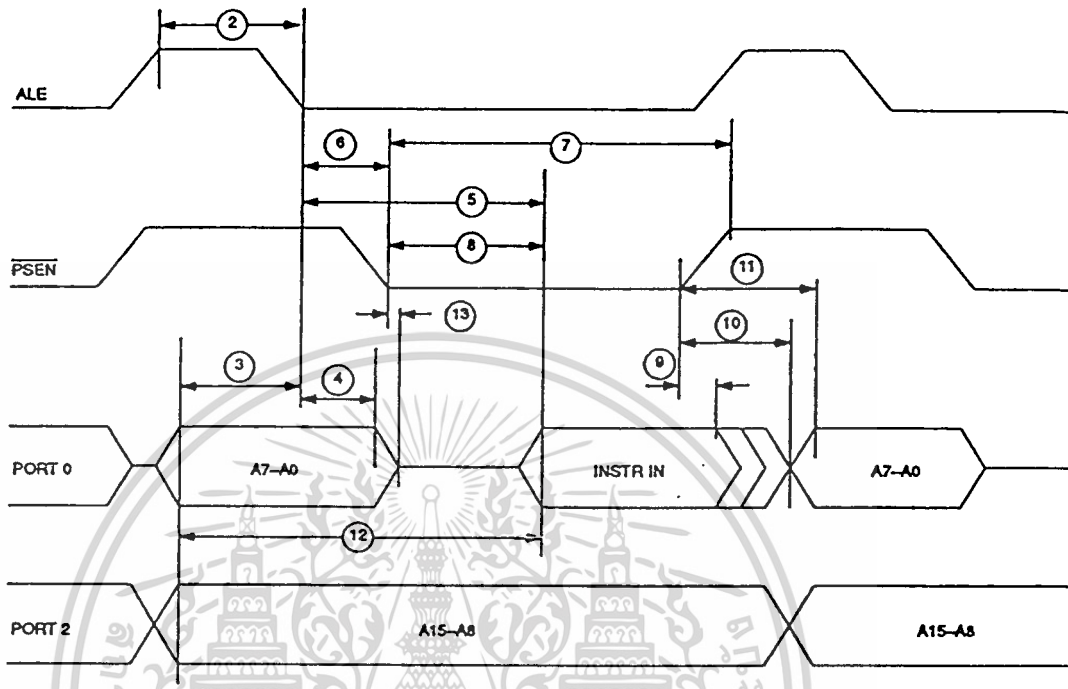
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Low Voltage	V_{IL}	-0.3		0.8	V	1
Input High Voltage	V_{IH1}	2.0		V_{CC} +0.3	V	1
Input High Voltage RST, XTAL2	V_{IH2}	3.5		V_{CC} +0.3	V	1
Output Low Voltage @ I_{OL}=1.6mA (Ports 1, 2, 3)	V_{OL1}		.15	0.45	V	
Output Low Voltage @ I_{OL}=3.2mA (Ports 0, ALE, $\overline{PSEN}$)	V_{OL2}		.15	0.45	V	1
Output High Voltage @ I_{OH}=-80µA (Ports 1, 2, 3)	V_{OH1}	2.4	4.8		V	1
Output High Voltage @ I_{OH}=-400 µA (Ports 0, ALE, $\overline{PSEN}$)	V_{OH2}	2.4	4.8		V	1
Input Low Current V_{IN} = 0.45V (Ports 1, 2, 3)	I_{IL}			-50	µA	
Transition Current; 1 to 0 V_{IN} = 2.0V (Ports 1, 2, 3)	I_{TL}			-500	µA	
Input Leakage Current 0.45 < V_{IN} < V_{CC} (Port 0)	I_L			±10	µA	
RST, $\overline{EA}$ Pulldown Resistor	R_{RE}	40		125	KΩ	
Stop Mode Current	I_{SM}			80	µA	4
Power Fail Warning Voltage	V_{PFW}	4.15	4.6	4.75	V	1
Minimum Operating Voltage	V_{CCmin}	4.05	4.5	4.65	V	1
Programming Supply Voltage (Parallel Program Mode)	V_{PP}	12.5		13	V	1
Program Supply Current	I_{PP}		15	20	mA	
Operating Current DS5000–8K@8 MHz DS5000–32K @ 12 MHz DS5000T–32–16 @ 16 MHz	I_{CC}		25.2 35.7 45.6	43 48 54	mA	2
Idle Mode Current @ 12 MHz	I_{CC}		4.5	6.2	mA	3

AC CHARACTERISTICS
EXPANDED BUS MODE TIMING SPECIFICATIONS

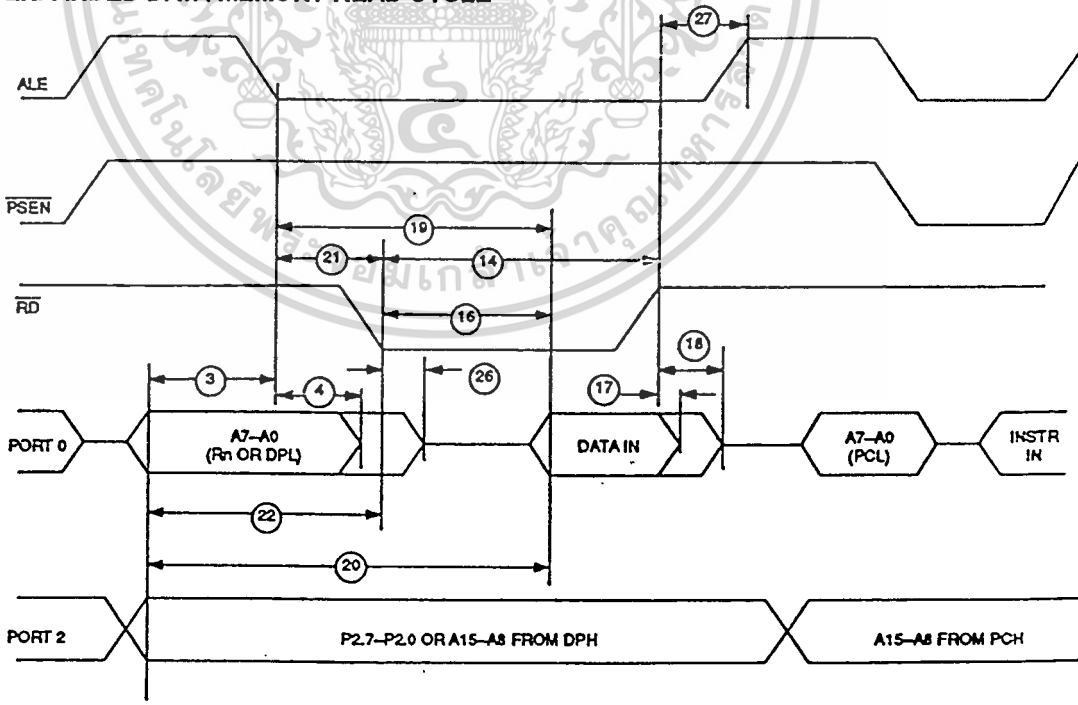
(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

#	PARAMETER	SYMBOL	MIN	MAX	UNITS
1	Oscillator Frequency	$1/t_{CLK}$	1.0	8 (-8) 12 (-12) 16 (-16)	MHz
2	ALE Pulse Width	t_{ALPW}	$2*t_{CLK}$		ns
3	Address Valid to ALE Low	t_{AVALL}	t_{CLK} -40		ns
4	Address Hold After ALE Low	t_{AVAAV}	t_{CLK} -35		ns
5	ALE Low to Valid Instr. In @16 MHz	t_{ALLVI}		$4t_{CLK}$ -150 $4t_{CLK}$ -90	ns ns
6	ALE Low to $\overline{PSEN}$ Low	t_{ALLPSL}	t_{CLK} -25		ns
7	$\overline{PSEN}$ Pulse Width	t_{PSPW}	$3t_{CLK}$ -35		ns
8	$\overline{PSEN}$ Low to Valid Instr. In @16 MHz	t_{PSLVI}		$3t_{CLK}$ -150 $3t_{CLK}$ -90	ns ns
9	Input Instr. Hold after $\overline{PSEN}$ Going High	t_{PSIV}	0		ns
10	Input Instr. Flat after $\overline{PSEN}$ Going High	t_{PSIX}		t_{CLK} -20	ns
11	Address Hold after $\overline{PSEN}$ Going High	t_{PSAV}	t_{CLK} -8		ns
12	Address Valid to Valid Instr. In @16 MHz	t_{AVVI}		$5t_{CLK}$ -150 $5t_{CLK}$ -90	ns ns
13	$\overline{PSEN}$ Low to Address Float	t_{PSLAZ}	0		ns
14	$\overline{RD}$ Pulse Width	t_{RDPW}	$6t_{CLK}$ -100		ns
15	$\overline{WR}$ Pulse Width	t_{WRPW}	$6t_{CLK}$ -100		ns
16	$\overline{RD}$ Low to Valid Data In @16 MHz	t_{RDLDV}		$5t_{CLK}$ -165 $5t_{CLK}$ -105	ns ns
17	Data Hold after $\overline{RD}$ High	t_{RDHDV}	0		ns
18	Data Float after $\overline{RD}$ High	t_{RDHDZ}		$2t_{CLK}$ -70	ns
19	ALE Low to Valid Data In @16 MHz	t_{ALLVD}		8_{CLK} -150 $8t_{CLK}$ -90	ns ns
20	Valid Addr. to Valid Data In @16 MHz	t_{AVDV}		$9t_{CLK}$ -165 $9t_{CLK}$ -105	ns ns
21	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	t_{ALLRDL}	$3t_{CLK}$ -50	$3t_{CLK}$ +50	ns
22	Address Valid to $\overline{RD}$ or $\overline{WR}$ Low	t_{AVRDL}	$4t_{CLK}$ -130		ns
23	Data Valid to $\overline{WR}$ Going Low	t_{DVWRL}	t_{CLK} -60		ns
24	Data Valid to $\overline{WR}$ High @16 MHz	t_{DVWRH}	$7t_{CLK}$ -150 $7t_{CLK}$ -90		ns ns
25	Data Valid after $\overline{WR}$ High	t_{WRHDV}	t_{CLK} -50		ns
26	$\overline{RD}$ Low to Address Float	t_{RDLAZ}		0	ns
27	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{RDHALH}	t_{CLK} -40	t_{CLK} +50	ns

EXPANDED PROGRAM MEMORY READ CYCLE

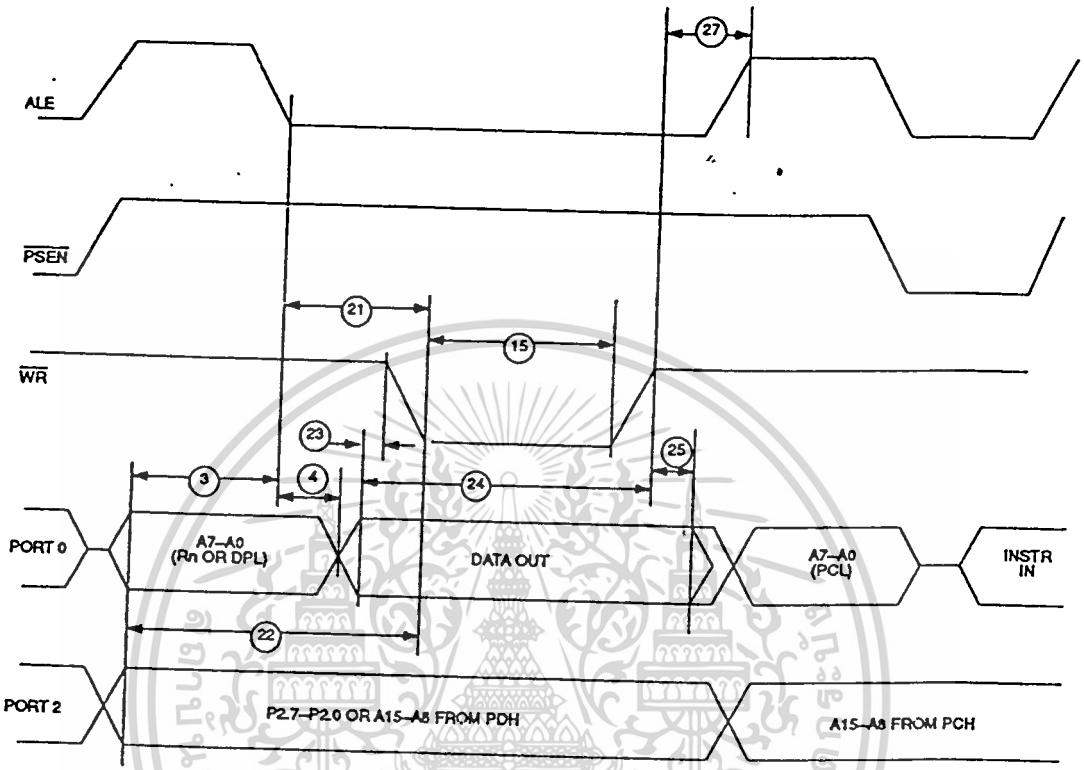


EXPANDED DATA MEMORY READ CYCLE

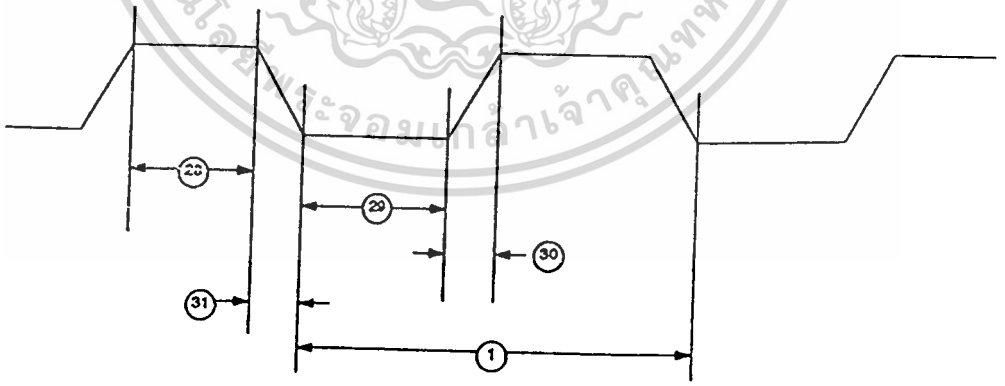


042193 11/18

EXPANDED DATA MEMORY WRITE CYCLE



EXTERNAL CLOCK TIMING

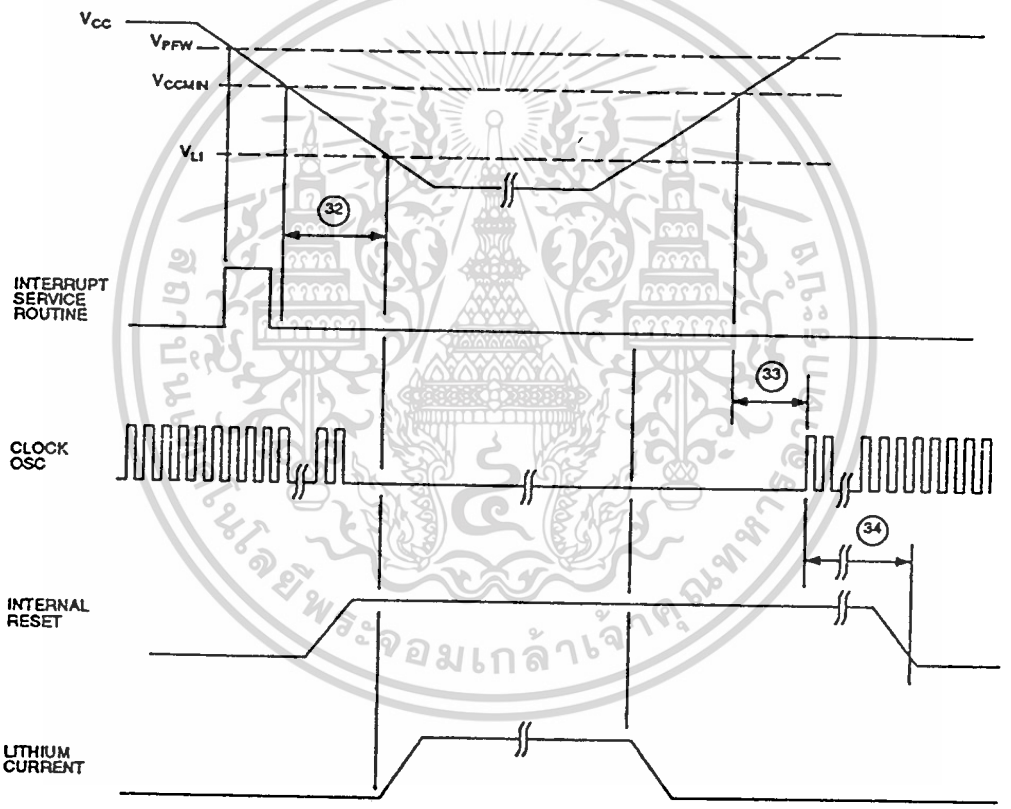


AC CHARACTERISTICS (cont'd)
SERIAL PORT TIMING – MODE 0

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

#	PARAMETER	SYMBOL	MIN	MAX	UNITS
35	Serial Port Cycle Time	t_{SPCLK}	$12t_{CLK}$		µs
36	Output Data Setup to Rising Clock Edge	t_{DOCH}	$10t_{CLK}$ -133		ns
37	Output Data Hold after Rising Clock Edge	t_{CHDO}	$2t_{CLK}$ -117		ns
38	Clock Rising Edge to Input Data Valid	t_{CHDV}		$10t_{CLK}$ -133	ns
39	Input Data Hold after Rising Clock Edge	t_{CHDIV}	0		ns

POWER CYCLE TIMING



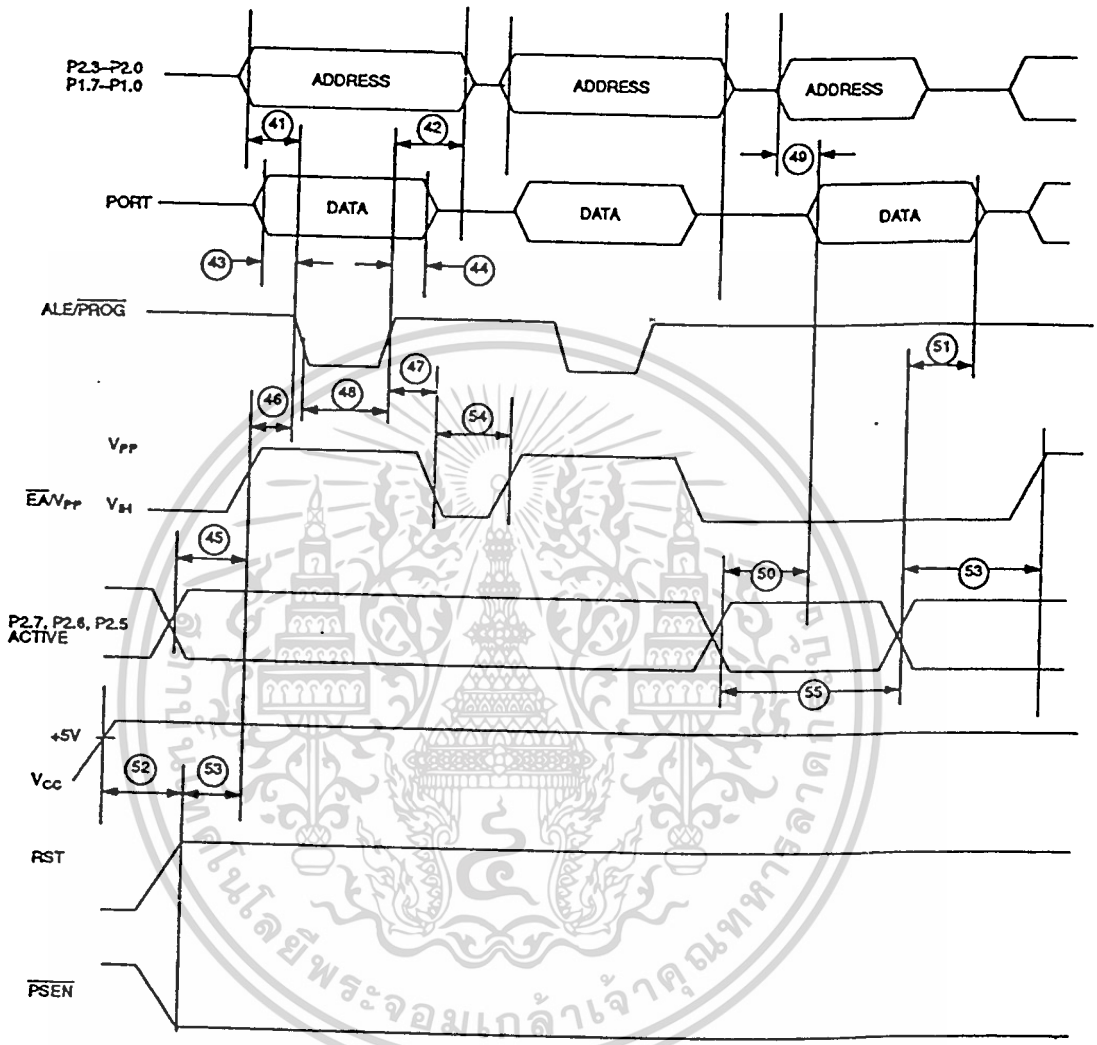
AC CHARACTERISTICS (cont'd)
PARALLEL PROGRAM LOAD TIMING

(t_A = 0°C to70°C; V_{CC} = 5V ± 5%)

#	PARAMETER	SYMBOL	MIN	MAX	UNITS
40	Oscillator Frequency	$1/t_{CLK}$	1.0	12.0	MHz
41	Address Setup to $\overline{PROG}$ Low	t_{AVPRL}	0		
42	Address Hold after $\overline{PROG}$ High	t_{PRHAV}	0		
43	Data Setup to $\overline{PROG}$ Low	t_{DVPRL}	0		
44	Data Hold after $\overline{PROG}$ High	t_{PRHDV}	0		
45	P2.7, 2.6, 2.5 Setup to V_{PP}	t_{P27HVP}	0		
46	V_{PP} Setup to $\overline{PROG}$ Low	t_{VPHPRL}	0		
47	V_{PP} Hold after $\overline{PROG}$ Low	t_{PRHVPL}	0		
48	$\overline{PROG}$ Width Low	t_{PRW}	2400		t_{CLK}
49	Data Output from Address Valid	t_{AVDV}		48 1800*	t_{CLK}
50	Data Output from P2.7 Low	t_{DVP27L}		48 1800*	t_{CLK}
51	Data Float after P2.7 High	t_{P27HDZ}	0	48 1800*	t_{CLK}
52	Delay to Reset/$\overline{PSEN}$ Active after Power On	t_{PORPV}	21504		t_{CLK}
53	Reset/$\overline{PSEN}$ Active (or Verify Inactive) to V_{PP} High	t_{RAVPH}	1200		t_{CLK}
54	V_{PP} Inactive (Between Program Cycles)	t_{VPPPC}	1200		t_{CLK}
55	Verify Active Time	t_{VFT}	48 2400 t_{CLK}		t_{CLK}

* Second set of numbers refers to expanded memory programming up to 32K bytes.

PARALLEL PROGRAM LOAD TIMING

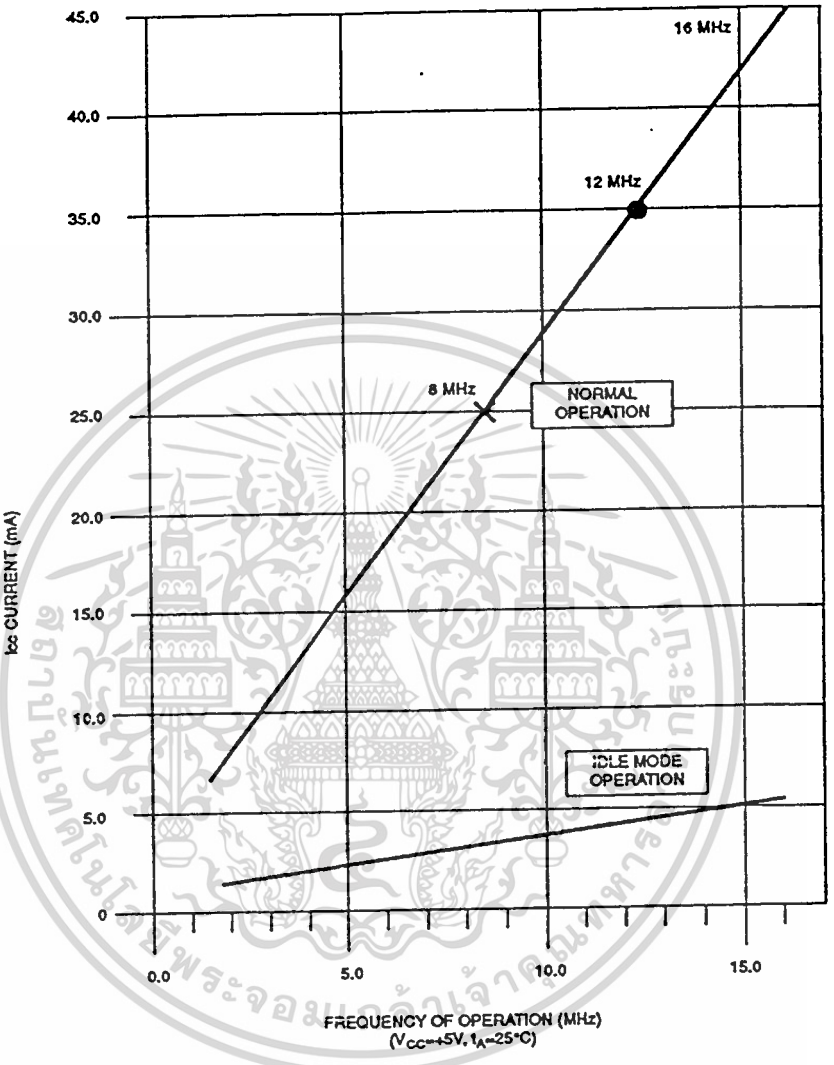


CAPACITANCE

(test frequency = 1 MHz; t_A = 25°C)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Output Capacitance	C_O			10	pF	
Input Capacitance	C_I			10	pF	

DS5000 TYPICAL I_{CC} VS. FREQUENCY



Normal operation is measured using:

1) External crystals on XTAL1 and 2
2) All port pins disconnected
3) RST=0 volts and EA=V_{CC}
4) Part performing endless loop writing to internal memory.

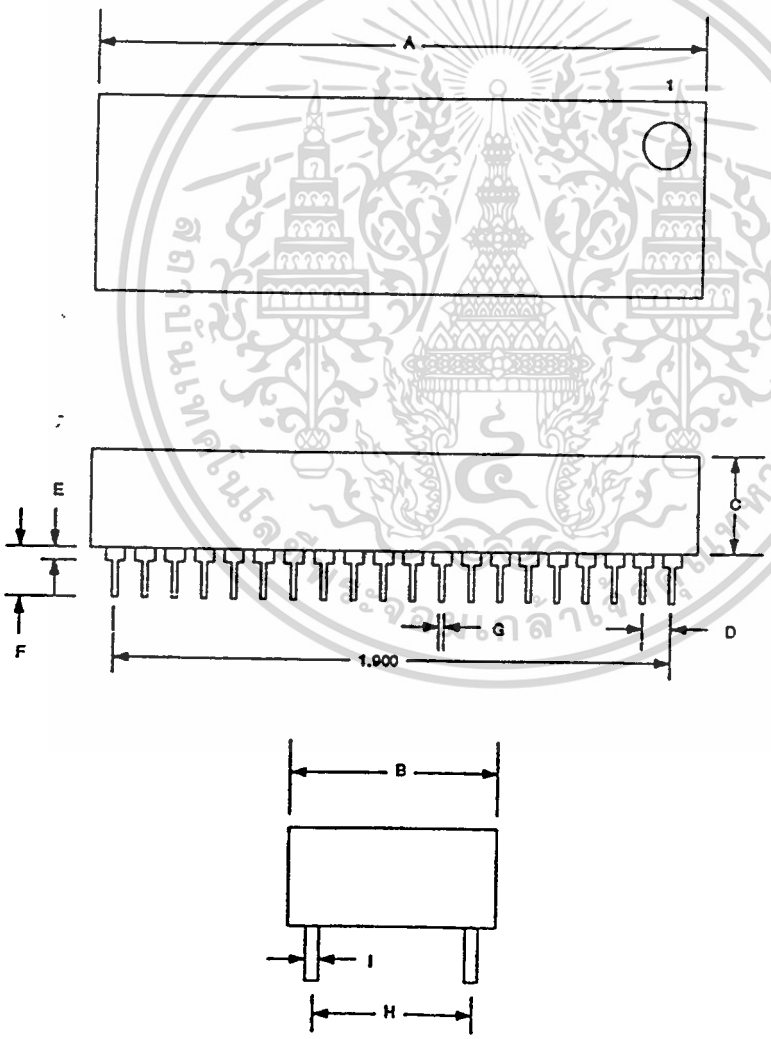
Idle mode operation is measured using:

1) External clock source at XTAL1; XTAL2 floating
2) All port pins disconnected
3) RST=0 volts and EA=V_{CC}
4) Part set in IDLE mode by software.

NOTES:

1. All voltages are referenced to ground.
2. Maximum operating I_{CC} is measured with all output pins disconnected; XTAL1 driven with tCLKR, t_{CLKF}=10 ns, V_{IL} = 0.5V; XTAL2 disconnected; $\overline{EA}$ = RST = PORT0 = V_{CC}.
3. Idle mode I_{CC} is measured with all output pins disconnected; XTAL1 driven with t_{CLKR}, t_{CLKF} = 10 ns, V_{IL} = 0.5V; XTAL2 disconnected; $\overline{EA}$ = PORT0 = V_{CC}, RST = V_{SS}.
4. Stop mode I_{CC} is measured with all output pins disconnected; $\overline{EA}$ = PORT0 = V_{CC}; XTAL2 not connected; RST = V_{SS}.
5. Crystal start up time is the time required to get the mass of the crystal into vibrational motion from the time that power is first applied to the circuit until the first clock pulse is produced by the on-chip oscillator. The user should check with the crystal vendor for the worst case spec on this time.

PACKAGE DRAWING



DIM	INCHES	
	MIN	MAX
A IN.	2.080	2.100
B IN.	0.680	0.700
C IN.	0.290	0.310
D IN.	0.090	0.110
E IN.	0.040	0.060
F IN.	0.165	0.185
G IN.	0.016	0.020
H IN.	0.590	0.610
I IN.	0.009	0.012

เอกสารอ้างอิง

1. Kenneth J. Ayala, **The 8051 Microcontroller : Architecture, Programming and Applications**, West Publishing Company, USA, 1991.
2. **ปรเมษฐ์ ประนยานันทน์, ปิยพงศ์ เผ่าวาณิช, คู่มือและการประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์ MCS-51**, บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน).
3. Timothy S. Monk, **Guide to Serial Communications**, pp. 299-353, Sams Publishing, 1992.
4. **น.ต. ฉัตรชัย สุมามาลย์, การสื่อสารข้อมูลคอมพิวเตอร์และระบบเครือข่าย, หน้า** 17-75, **บริษัท ด่านสุทธาการพิมพ์ จำกัด**
5. Peter W. Gofton **เรียบเรียงโดย จิรศักดิ์ เหลืองอุไร, การสื่อสารอนุกรมบน PC,** หน้า 312-350, **บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน).**
6. **ประสิทธิ์ วิทยธีราภรณ์, เทคนิคการสื่อสารด้วยโมเด็ม, หน้า** 19-80, **หจก.สำนักพิมพ์ฟิสิกส์เซ็นเตอร์**

ประวัติผู้เขียน

นายภูเมศร์ จินดาจิธาวัฒน์ เกิดวันที่ 24 พฤษภาคม พ.ศ.2520 สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนศึกษานารีวิทยา ในปี พ.ศ.2537 ได้เข้ารับการศึกษาต่อที่ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง สำเร็จการศึกษาในปี พ.ศ.2541