สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ระบบโทรสารแบบจัดเก็บและส่งต่อ

THE STORE-AND-FORWARD FAX SYSTEM

โดย

นาย วรฉัตร สิริยานนท์

นาย วิรุจ สุวรรณปราโมทย์

นาย วีรสิษฐ์ กิติวรรณกุล

b.....
i.....

เลขหมู่.....
เลขทะเบียน 61963
วัน,เดือน,ปี 25 ก.ค. 2549

ภาควิชา
วิศวกรรมโทรคมนาคม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ระบบโทรสารแบบจัดเก็บและส่งต่อ

THE STORE-AND-FORWARD FAX SYSTEM

โดย

นาย วรฉัตร	สิริยานนท์	44010407
นาย วิรุจ	สุวรรณปราโมทย์	44010456
นาย วีรสิษฐ์	กิติวรรณกุล	44010470

อาจารย์ที่ปรึกษา

ดร.พิพัฒน์ พรหมมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบโทรสารแบบจัดเก็บและส่งต่อ**

THE STORE-AND-FORWARD FAX SYSTEM

ผู้จัดทำ

1. นาย วรฉัตร	**สิริยานนท์**	**44010407**
2. นาย วิรุจ	**สุวรรณปราโมทย์**	**44010456**
3. นาย วีรสิษฐ์	**กิติวรรณกุล**	**44010470**

..... อาจารย์ที่ปรึกษา

(ดร. พิพัฒน์ พรหมมี)

ระบบโทรสารแบบจัดเก็บและส่งต่อ

THE STORE-AND-FORWARD FAX SYSTEM

โดย นาย วรฉัตร สิริยานนท์ 44010407

นาย วิรุจ สุวรรณปราโมทย์ 44010456

นาย วีรสิษฐ์ กิติวรรณกุล 44010470

อาจารย์ที่ปรึกษา ดร.พิพัฒน์ พรหมมี

บทคัดย่อ

ปริญญานิพนธ์ นี้เสนอระบบโทรสารแบบจัดเก็บและส่งต่อ โดยมีหลักการ คือ ผู้ใช้บริการจะต้องสมัครเป็นสมาชิกของระบบ โดยระบบจะมีเซิร์ฟเวอร์ในพื้นที่ต่างๆ และ เชื่อมต่อกันด้วยระบบอินเตอร์เน็ท เพื่อวัตถุประสงค์สำหรับการส่งโทรสารทางไกล หรือ ส่งโทรสารข้ามประเทศ ด้วยค่าใช้จ่ายที่ประหยัด โดยสมาชิกของบริการนี้สามารถส่งโทรสารไปยังหมายเลขปลายทางผ่านระบบ จากนั้นระบบจะเก็บโทรสารนั้นๆ เพื่อที่จะส่งต่อโทรสารโดยไปยังเลขหมายปลายทางตามที่ระบุไว้ โดยระบบจะมีการส่งผ่านเส้นทางสำรองกรณีเส้นทางหลักไม่สามารถส่งได้ เช่น จดหมายอิเล็กโทรนิกส์ และระบบจะมีการแจ้งผลการทำงานให้กับสมาชิกทราบด้วยผ่านทางเอสเอ็มเอส

ABSTRACT

A Store and Forward Fax System (SFFS) based on member registration is presented in this project. The system consists of a number of fax servers that located in different areas and also have had the internet connections. The long distance/oversea cost reduction and internet bandwidth utilization are mainly concept of this project. The subscribers who registered to the SFFS can send the fax to different destinations that provided by the SFFS. The SFFS have to store the received fax and then transfer to the other related server by internet communications as soon as possible. The destination fax server analyses the destination number then sent fax through destination using PSTN connection. The alternated routes are used in case of first route is failure connected as well as email. The system can also provide the feature of status updated to the member via by SMS.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ในช่วงทศวรรษที่ผ่านมาการพัฒนาด้านการสื่อสารเป็นไปอย่างก้าวกระโดด ซึ่งทำให้ระบบโทรคมนาคม, ด้านสื่อสารสัญญาณ มีการส่งผ่านข้อมูลที่รวดเร็วและมีความเชื่อถือมากยิ่งขึ้น การเข้ามาของอินเตอร์เน็ต (Internet) ซึ่งเป็นเทคโนโลยีที่ทำให้สามารถสื่อสารกันระยะไกลได้สะดวก เป็นระบบเครือข่ายคอมพิวเตอร์สากลที่มีผู้ใช้งานมากที่สุดในโลก ซึ่งทำให้บุคคลหรือองค์กรทั้งหลายสามารถใช้ประโยชน์จากระบบเครือข่ายนี้ได้อย่างมีประสิทธิภาพ

หากเราจะประยุกต์การใช้งานด้านโทรสารซึ่งค่าบริการนั้นขึ้นอยู่กับขนาดของข้อมูลและระยะทางของการส่งทำให้เกิดปัญหาว่า ถ้าเราต้องการส่งข้อมูลที่มีขนาดใหญ่โดยเชื่อมต่อผ่านโครงข่ายโทรศัพท์พื้นฐานไปยังเครื่องโทรสารปลายทางที่อยู่ไกล จะทำให้เสียค่าใช้จ่ายเป็นจำนวนมาก

โดยโครงงานนี้จึงจัดทำขึ้นเพื่อแก้ไขปัญหาดังที่กล่าวในข้างต้น โดยเป็นการใช้ประโยชน์ของระบบอินเตอร์เน็ตทำการส่งผ่านข้อมูลระยะไกลแทนการเชื่อมต่อเครื่องโทรสารโดยตรงผ่านทางโครงข่ายโทรศัพท์พื้นฐาน ซึ่งแอพพลิเคชั่นนี้มุ่งเน้นการลดค่าใช้จ่ายระยะทางไกลในการส่งโทรสาร พร้อมทั้งยังมีบริการเสริมที่อำนวยความสะดวกทั้งผู้รับ-ส่งโทรสารอีกด้วย

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อที่จะสามารถนำความรู้ที่ศึกษามาประยุกต์ใช้ในงานด้านบริการได้

1.2.2 เพื่อที่จะพัฒนาซอฟต์แวร์ (Software) งานด้านรับ-ส่งโทรสารผ่านเครือข่ายอินเตอร์เน็ตได้

1.2.3 ศึกษาการเขียนโปรแกรมภาษาบอร์แลน เดลฟายเอ็นเตอร์ไพร เวอร์ชัน 7.0 (Borland Delphi Enterprise Version 7.0)

1.2.4 ศึกษาลักษณะการทำงานและคำสั่งใช้งานของระบบแฟกซ์โมเด็ม (Fax Modem)

1.2.5 เพื่อที่จะพัฒนาซอฟต์แวร์ ด้านระบบฐานข้อมูล (Database), การส่งเอสเอ็มเอส (SMS:Short Message Service)ไปยังโทรศัพท์มือถือ (Mobile Phone), ระบบตอบรับด้วยเสียง (IVR:Interactive Voice Response)

1.3 ขอบเขตของโครงงาน

1.3.1 ทำการออกแบบซอฟต์แวร์เพื่อใช้ในการติดต่อระหว่างแฟกซ์โมเด็มและคอมพิวเตอร์

1.3.2 พัฒนาการส่งข้อมูลติดต่อกันระหว่างเครื่องคอมพิวเตอร์บนเครือข่ายอินเตอร์เน็ต

1.3.3 ออกแบบลักษณะการติดต่อระหว่างผู้ใช้งานกับซอฟต์แวร์ (User Interface)

1.3.4 สามารถนำโปรแกรมที่พัฒนาแล้วไปใช้งานได้จริง

1.3.5 พัฒนาระบบตอบรับด้วยเสียง,การส่งเอสเอ็มเอส,ระบบฐานข้อมูล

1.4 วิธีการดำเนินงาน

งานพัฒนาแอพพลิเคชั่นที่สร้างขึ้นนี้มีโปรแกรมสำหรับรับข้อมูลที่ต้องการส่งจากเครื่องโทรสารต้นทางมาเก็บเป็นแฟ้มข้อมูลที่เครื่องผู้ให้บริการซึ่งต่ออยู่กับเครือข่ายอินเตอร์เน็ต แล้วทำการส่งข้อมูลผ่านเครือข่ายอินเตอร์เน็ตไปยังเครื่องให้บริการปลายทางในกลุ่มที่ผู้รับโทรสารอาศัยอยู่ จากนั้นจึงส่งต่อไปยังเครื่องโทรสารปลายทางได้ ซึ่งต้องทำการศึกษาระบบโปรโตคอล TCP/IP การติดต่อส่งข้อมูลข่าวสารผ่านอินเตอร์เน็ต, การเชื่อมต่อและการควบคุมแฟกซ์โมเด็มผ่านคอมพิวเตอร์ให้ทำหน้าที่ตามต้องการ เทคนิคการรับส่งข้อมูลของระบบโทรสารกลุ่ม 3 ซึ่งมีการใช้งานแพร่หลายที่สุดในปัจจุบัน

บทที่ 2
ทฤษฎีและหลักการ

2.1. โทรสาร (FAX : Facsimile)

โทรสารเป็นเครื่องมือสื่อสารที่มีความสามารถในการส่งภาพจากต้นทางไปยังปลายทาง ได้ถูกคิดค้นเมื่อ ปี ค.ศ.1843 ก่อนที่จะมีการคิดค้นโทรพิมพ์ (Type Writer) 60 ปี โทรสารมีการพัฒนามาอย่างต่อเนื่องแต่ก็มักประสบปัญหาด้านโครงสร้างภายในของระบบการสื่อสารสมัยก่อน ในช่วง ปี ค.ศ. 1920 โทรสารมีความสำคัญอย่างยิ่งในด้านงานข่าว โดยสำนักข่าวส่วนใหญ่นำมาใช้ในการส่งภาพข่าว

การใช้โทรสารในการส่งเอกสารและรับเอกสาร ผ่านโครงข่ายการสื่อสาร โดยใช้เทคนิคของการส่งข้อมูลตัวอักษรในแบบเดียวกัน และมีความเหมาะสมกับการส่งเอกสารที่ประกอบด้วยข้อมูลตัวอักษรในแบบเดียวกัน และมีความเหมาะสมกับการส่งเอกสารที่ประกอบด้วยตัวหนังสืออย่างเดียว ส่วนเทคนิคอื่นๆก็คือ การส่งภาพนิ่งของต้นฉบับในแบบอิเล็กทรอนิกส์ โดยจะมีแผนผังบิตเป็นส่วนในการสร้างภาพนิ่งนี้เพื่อใช้ในการสื่อสารผ่านโครงข่าย การบริการของโทรสารจะรวมไปถึงตัวอักษร ภาพกราฟิก (Graphic) หรือ รูปภาพ

ระบบโทรสารจะมีข้อดีเหนือกว่าการส่งอื่นๆ ในรูปแบบของเอกสารหรือส่งข้อความ การใช้งานของโทรสารจะช่วยประหยัดเวลาในการเตรียมเอกสารอีกด้วย เนื่องจากการส่งข่าวสารไม่จำเป็นต้องมาผ่านการส่งโดยการกดแป้นพิมพ์ แต่จะใช้ตัวต้นฉบับของเอกสารนั้นส่งไปยังปลายทางได้เลย โดยต้นฉบับที่เป็นเอกสารก็จะถูกสแกน(Scan) หาความสว่างและความมืดของเอกสาร แล้วจะแปลงเป็นสัญญาณอิเล็กทรอนิกส์เพื่อส่งไปยังปลายทางของระบบโทรสารได้ถูกพัฒนาขึ้นในปี ค.ศ.1843 แต่เพิ่งถูกนำมาใช้งานในตอนกลางปี ค.ศ. 1970 และมีการใช้งานเพิ่มขึ้นอย่างรวดเร็ว

ได้มีการพัฒนาและการยอมรับมาตรฐาน CCITT เกี่ยวกับโทรสาร ทำให้โทรสาร สามารถทำงานร่วมกันระหว่างประเทศได้

เทคโนโลยีของโทรสารมีความก้าวหน้าขึ้น มีความเร็วสูงขึ้น คุณภาพดีกว่าเดิม ขนาดของเครื่องเล็กลง ค่าใช้จ่ายของเครื่องและการส่งโทรสารลดลง และมีการใช้เครื่องที่ง่ายขึ้น

การคิดค้นระบบโทรสารแบบดิจิตอล ทำให้แต่ละหน้าของข่าวสาร สามารถดำเนินการได้อย่างง่ายโดยใช้คอมพิวเตอร์ ทำให้สามารถเก็บข้อมูลข่าวสารลงในดิสก์ (Disk) หรือเทปแม่เหล็ก

โทรสารยังคงสามารถพัฒนาเรื่อยจนถึงปัจจุบัน การพัฒนาของระบบโทรสารดิจิตอล ทำให้บทบาทของโทรสารได้ขยายกว้างออกไปในรูปแบบของดิจิตอลในแง่ของข่าวสารที่สามารถส่งระหว่างเครื่องโทรสารด้วยกันได้โดยตรงโดยไม่ต้องผ่านขั้นตอนบนกระดาษก่อน เอกสารที่ต้องการส่งสามารถถูกเตรียมโดยการใช้โปรแกรมเขียนแบบเวิร์ดโปรเซสเซอร์ (Word Processor) ความสามารถในการส่งตัวหนังสือและกราฟิกร่วมกันได้ ดังนั้นผู้ใช้สามารถแก้ไขตัวหนังสือได้ รวมทั้งรูปแบบกราฟิกต่างๆ ก่อนที่จะมีการส่งโทรสารออกไปยังปลายทาง และเนื่องจากโทรสารสามารถนำมาต่อเข้ากับเครือข่ายโทรศัพท์ ได้ทันที มีการใช้งานได้สะดวกรวดเร็วทำให้ โทรสารเป็นอุปกรณ์สื่อสารที่มีการใช้กันอย่างแพร่หลายในสำนักงานและมีราคาที่ถูกลงอย่างมาก

2.1.1 การแบ่งกลุ่มโทรสาร

CCITT ได้แบ่งอุปกรณ์โทรสารสำหรับใช้บนโครงข่ายสาธารณะออกเป็น 4 แบบ (หรือกลุ่ม)

- กลุ่มที่ 1 (Group 1 Facsimile) จะเป็นโทรสารในแบบอนาล็อก ที่มีความเร็วในการส่งต่ำ การส่งเป็นแบบ FM (Frequency Modulation) ในการแบ่งแยกระดับของสีเทาหลายๆ ระดับ ที่อยู่ระหว่างกลางของสีขาวกับสีดำซึ่งสามารถส่งเอกสารขนาด A4 ISO (210 x 297 มิลลิเมตร หรือ 8.25 x 11.7 นิ้ว) ที่ประมาณ 4 เส้น ต่อ มิลลิเมตร (100 เส้นต่อนิ้ว) ในเวลา ประมาณ 6 นาที และมีข้อกำหนดต่างๆตามข้อกำหนด T.2 ของ CCITT

- กลุ่มที่ 2 (Group 2 Facsimile) เป็นโทรสารแบบอนาล็อกที่มีการปรับปรุงขึ้นมาโดยใช้เทคนิคในการลดแบนด์วิดท์ลงเพื่อให้ได้ความเร็วเพิ่มขึ้นเป็น 2 เท่าหรือมากกว่าที่รายละเอียดของภาพเท่ากันกับกลุ่มที่ 1 การส่งจะเป็นแบบ Duo binary PM (Phase Modulation) และจะยังคงมีระดับของสีเทาระดับต่างๆ ในระหว่างสีขาวกับสีดำ ต้องสามารถทำการส่งเอกสาร A4 ในภายในเวลา 3 นาที และมีข้อกำหนดต่างๆ ตามข้อกำหนด T.3 ของ CCITT

- กลุ่มที่ 3 (Group 3 Facsimile) เป็นโทรสารระบบแรกที่ใช้ดิจิตอล ระบบนี้จะจัดให้มีค่าเพียงขาวกับดำเท่านั้น โดยจะมีการสุ่มสัญญาณที่มีความหนาแน่น 200 พิกเซลต่อนิ้ว ทางแนวนอนขวางของกระดาษ โดยจะมีสแกน 100 หรือ 200 เส้นต่อหนึ่งนิ้วทางแนวตั้งของหน้ากระดาษ โทรสารกลุ่มที่3 จะใช้การเข้ารหัสดิจิตอลและมีการลดค่าเฉลี่ยของข่าวสารที่ซ้ำซ้อนในสัญญาณที่ได้จากการเข้ารหัสของข่าวสารบนเอกสารก่อนที่จะทำการมอดูเลชั่น ถ้ามีการใช้โมเด็มในการส่งสัญญาณผ่านโครงข่ายโทรศัพท์อนาล็อก เวลาในการส่งจะเร็วขึ้นเป็น 3 เท่า หรือมากกว่าเมื่อเปรียบเทียบกับโทรสารในกลุ่มที่ 2 ต้องสามารถทำการส่งเอกสาร A4 ได้ในเวลา 1 นาที และข้อกำหนดต่างๆตาม ข้อกำหนด T.4 ของ CCITT

- กลุ่มที่ 4 (Group 4 Facsimile) เป็นโทรสารระบบดิจิตอล ดำ/ขาว เช่นเดียวกับกลุ่มที่ 3 แต่จะถูกเน้นสำหรับใช้บนโครงข่ายดิจิตอลความเร็วสูงขึ้นไปถึง 64 Kbps และมีการจัดการเกี่ยวกับตรวจสอบความผิดพลาดของข้อมูลที่ได้ รับ ความละเอียดสามารถใช้ได้จาก 200 ถึง 400 พิกเซล ต่อนิ้วการส่งในโทรสารกลุ่มที่ 4 จะใช้เวลาลดลงเหลือเพียงไม่กี่วินาที มาตรฐานของกลุ่มที่ 4 เป็นมาตรฐานดิจิตอลที่ใช้ร่วมกับระบบ ISDN

ปัจจุบันมีการใช้โทรสารทั่วไปจะใช้ตามมาตรฐานโทรสารกลุ่มที่ 3 โดยสามารถใช้การเข้ารหัสข้อมูลแบบดิจิตอล (มาตรฐานกลุ่มที่ 1 และ 2 มีระบบการทำงานแบบอนาล็อก ส่วนมาตรฐานกลุ่มที่ 3 และ 4 มีระบบการทำงานแบบดิจิตอล) โดยตามมาตรฐานโทรสารกลุ่ม 3 จะมีความละเอียดในแนวตั้ง 3.85 เส้นต่อมิลลิเมตร และในแนวนอนจะมีความละเอียดในแนวนอน 1728 พิกเซลต่อความยาว 215 มิลลิเมตร เท่ากับ 8 พิกเซลต่อมิลลิเมตร อย่างไรก็ดีในมาตรฐาน T.4 ให้เพิ่มความละเอียดได้สูงถึง 15.4 เส้นต่อมิลลิเมตร ในแนวตั้ง 16 เส้นต่อมิลลิเมตร ในแนวนอน

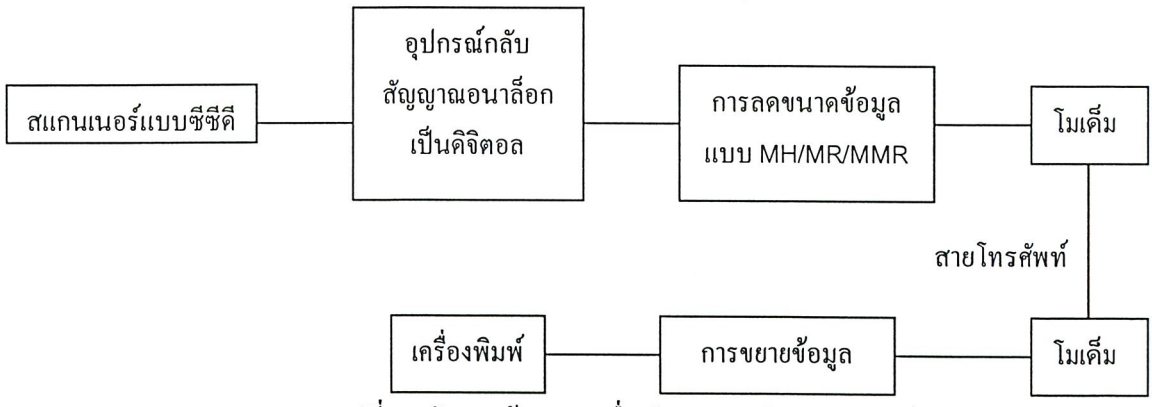
2.1.2 คุณสมบัติของเครื่องโทรสารกลุ่ม 3

คุณสมบัติของเครื่องโทรสารกลุ่ม 3 ตามมาตรฐานของ CCITT แสดงดังต่อไปนี้

Item	Standard	Options		Small Copy {A5&A6}			
Scan Width							
In	8.46	10	11.9	4.2	5.9	5.9	4.2
Mm	215	255	303	107	151	151	107
Pels/line	1728	2048	2432	854	1216	1728	1728
H /in	203			203	203	290	406
/mm	8			8	8	11.4	16
v /in	97.8	196		196/392	138/176	138/176	196/392
/mm	385	7.7		7.7/15.4	5.44/10.9	5.44/10.9	7.7/15.4
Ms/line	20	0,5,10,40					
Coding	Modified	Modified Read, Modified-Modified Read					
Modem							
Fax Signal	V.27ter	V.29, V.17					
Bits/s	2400/4800	9600/7200, 14400, 12000, 9600, 7200					
Handshake	V.21	V.27ter					
Bits/s	(Ch2)	2400					
	300						
Error	None	Error Correction Mode					

ตารางที่ 2.1 คุณสมบัติของเครื่องโทรสารกลุ่ม 3 ตามมาตรฐานของ CCITT

2.1.3 โครงสร้างของเครื่องโทรสารกลุ่ม 3



รูปที่ 2.1 โครงสร้างของเครื่องโทรสารกลุ่ม 3

เครื่องโทรสารกลุ่ม 3 มีการทำงานดังต่อไปนี้

1. สแกนเนอร์แบบซีซีดี (CCD:Charge Coupled Device) ทำหน้าที่ในการเปลี่ยนภาพให้เป็นไฟฟ้าโดยอาศัยโฟโต้เซนเซอร์(Photosensors) เป็นตัวตรวจจับความเข้มของจุดแต่ละจุดแล้วสร้างพัลส์ (Pulse) ขึ้นมาแทนจุดแต่ละจุดนั้นซึ่งใน 1 เส้นจะทำการตรวจจับ 1728 จุด ทำให้ได้พัลส์จำนวน 1728 ลูก

2. ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล (A/D Converter) จะทำการเปลี่ยนสัญญาณอนาล็อกให้เป็นดิจิตอล

3. การลดขนาดข้อมูล อาจเป็นแบบโมดิฟายด์ฮัฟฟ์แมน (MH:Modified Huffman), โมดิฟายด์รีดดิ้ง(MR:Modified Reading) หรือโมดิฟายด์-โมดิฟายด์รีดดิ้ง(MMR:Modified Modified Reading) ซึ่งจะทำการลดจำนวนบิทของข้อมูลให้น้อยลง โดยการใช้รหัสตัวใหม่แทนข้อมูลเดิมทำให้ข้อมูลมีขนาดเล็กกว่าเดิมช่วยให้สามารถส่งข้อมูลได้เร็วขึ้นด้วย

4. โมเด็ม (Modem) จะทำการเปลี่ยนสัญญาณที่ถูกเข้ารหัสเป็นดิจิตอลเรียบร้อยแล้ว ให้เป็นสัญญาณอนาล็อกอีกครั้งหนึ่ง เพื่อให้สามารถส่งไปบนสายโทรศัพท์

ส่วนทางด้านรับจะมีโมเด็มทำการเปลี่ยนสัญญาณอนาล็อกที่รับมาทางด้านส่ง ให้เป็นสัญญาณดิจิตอลแล้วส่งไปยังส่วนขยายข้อมูล ซึ่งจะทำการเปลี่ยนรหัสต่างๆ ให้อยู่ในรูปของจุดขาว, ดำ เพื่อให้เครื่องพิมพ์สามารถแสดงผลออกมาได้เหมือนต้นฉบับเดิม

2.1.4 ขบวนการรับส่งข้อมูลของเครื่องโทรสารกลุ่ม 3

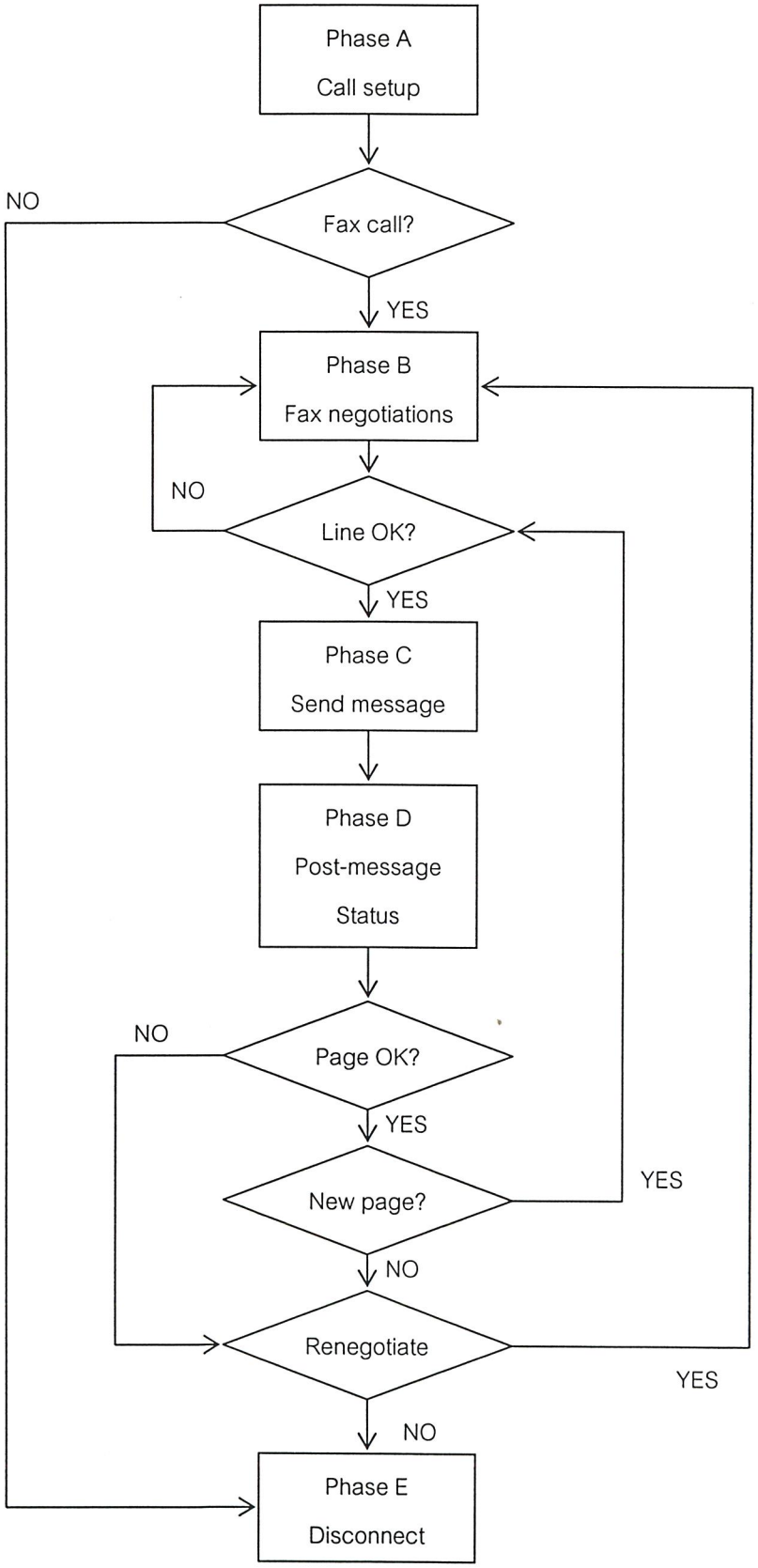
ตามมาตรฐานของ CCITT T.30 ได้แบ่งขั้นตอนในการรับส่งข้อมูลของเครื่องโทรสารกลุ่ม 3 ออกเป็น 5 เฟส แสดงได้ดังรูปที่ 2.2

ช่วงการทำงานตั้งแต่เฟส A-E เรียกว่า แฟกซิไมล์เซชชั่น (Facsimile session)

ช่วงการทำงานตั้งแต่เฟส B-D เรียกว่า แฟกซ์ไมล์โพรซีเดอร์ (Facsimile procedure)

เฟส A ขั้นตอนการติดต่อ (Call Establishment) หรือเรียกว่าการเริ่มต้นการเรียก เป็นขั้นตอนที่จะเชื่อมโยงเครื่องโทรสารต้นทางเข้ากับเครื่องโทรสารปลายทาง โดยผ่านทางข่ายสายโทรศัพท์ อาจเป็นแบบควบคุมโดยโอเปอเรเตอร์ ให้โอเปอเรเตอร์ติดต่อกันได้แล้วจึงทำการส่งข้อมูล หรืออาจเป็นแบบอัตโนมัติคือรับหรือส่งเอกสารโดยอัตโนมัติที่ด้านใดด้านหนึ่งหรือสามารถรับอัตโนมัติและส่งอัตโนมัติทั้งสองด้าน

ทางด้านผู้เรียกจะทำการเรียกไปยังเครื่องโทรสารปลายทางที่ต้องการติดต่อด้วยสัญญาณไดออลโทน (Dial Tone) และส่งสัญญาณ CNG (Calling tone) ไปเพื่อเริ่มการติดต่อ ส่วนทางด้านผู้ถูกเรียกจะตอบสนองต่อการเรียกนั้น โดยส่งสัญญาณ CED (Called station identification) ซึ่งเป็นสัญญาณความถี่ 2100 Hz กลับไปยังเครื่องโทรสารด้านผู้เรียกทุกๆ 3 วินาที



รูปที่ 2.2 ขบวนการในการรับส่งข้อมูลของเครื่องโทรสารกลุ่ม 3

เฟส B ขั้นตอนการส่งข่าวสาร (Pre-message procedure) เป็นขั้นตอนที่เครื่องโทรสารต้นทางเชื่อมโยงกับเครื่องโทรสารปลายทางแล้วต้องทำการตรวจสอบคุณสมบัติและความสามารถในการรับและส่งให้ตรงกันก่อนที่จะเริ่มส่งข่าวสารมี 2 ขั้นตอน คือ

1. ส่วนการตรวจสอบสัญญาณ ทำการส่งสัญญาณเพื่อบอกให้ทราบว่าเป็นเครื่องโทรสารอยู่ในกลุ่มใด, พร้อมที่จะรับข้อมูลหรือไม่, หมายเลขโทรศัพท์ของเครื่องโทรสารต้นทาง รวมถึงความสามารถอื่นๆที่ไม่ใช่มาตรฐานของ CCITT เป็นทางเลือก

2. ส่วนคำสั่ง ส่งสัญญาณเพื่อตรวจสอบระบบสื่อสัญญาณ DIS (Digital identification signal) ไปยังด้านผู้เรียกเพื่อแสดงคุณสมบัติและความสามารถต่างๆของเครื่อง ซึ่งเมื่อทางด้านผู้เรียกได้รับสัญญาณนี้แล้วก็จะส่งสัญญาณ DCS (Digital Command Signal) มายังด้านผู้ถูกเรียกเพื่อแจ้งให้ทราบว่าได้เลือกใช้คุณสมบัติและความสามารถใดของเครื่องที่ถูกเรียกเป็นข้อกำหนดในการรับส่งข้อมูลครั้งนี้ ซึ่งเครื่องโทรสารด้านถูกเรียกก็ต้องเลือกโหมดของตนเองให้เป็นไปตามข้อกำหนดนั้น จากนั้นเครื่องโทรสารด้านผู้เรียกจะส่งสัญญาณความเร็วสูงที่เรียกว่า เทรนนิ่ง (Training)เพื่อตรวจสอบระบบสื่อสาร ซึ่งหลังจากเครื่องโทรสารด้านผู้ถูกเรียกได้รับสัญญาณเทรนนิ่งและเมื่อตรวจสอบเรียกร้อยแล้ว ก็จะส่งสัญญาณ CFR (Confirmation to Receive) กลับไปยังเครื่องโทรสารต้นทางเพื่อยืนยันความพร้อมในการรับข้อมูล

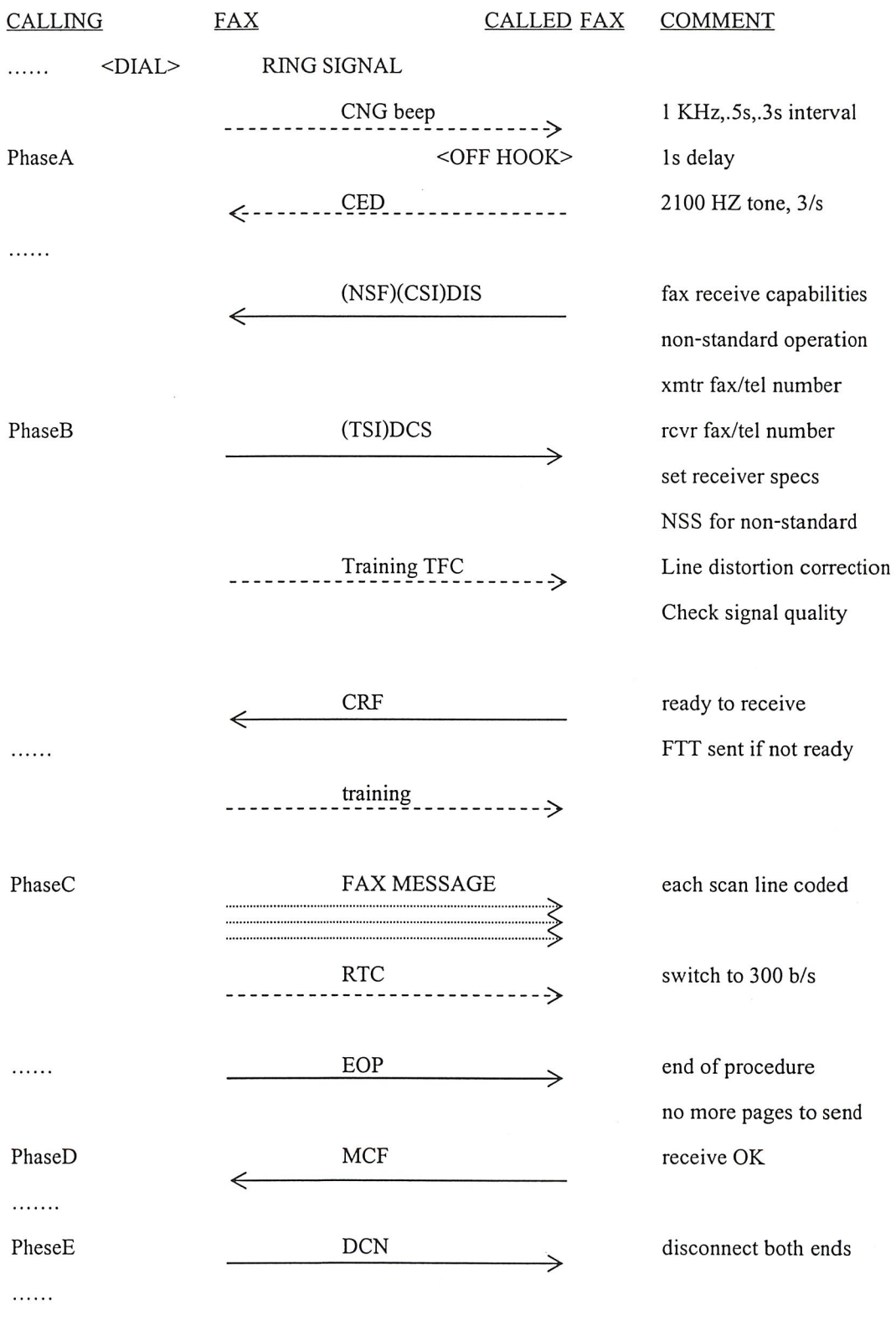
เฟส C ขั้นตอนการส่งข้อมูล (Message Transmission) ประกอบด้วย

1. เฟส C1 เป็นขั้นตอนส่งสัญญาณควบคุมเพื่อการซิงโครไนซ์ การตรวจสอบข้อผิดพลาด การแก้ไขข้อผิดพลาด และตรวจสอบสถานะของระบบสื่อสัญญาณในขณะที่กำลังส่งข่าวสาร

2. เฟส C2 เป็นขั้นตอนการส่งข่าวสาร โดยมีรูปแบบตามแต่ชนิดของเครื่องโทรสารแต่ละกลุ่ม

เฟส D ขั้นตอนหลังการส่งข่าวสาร (Post-message procedure) เครื่องโทรสารด้านผู้เรียกจะส่งสัญญาณ RTC(Return to Control) ไปยังด้านถูกเรียกเพื่อทำการปรับโมเด็มให้กลับไปอยู่ที่ความเร็ว 300 บิท/วินาที จากนั้นจึงส่งสัญญาณ EOP (End Of Procedure) ซึ่งทางด้านถูกเรียกก็จะส่งสัญญาณ MCF (Message Confirmation) กลับมาให้ทางด้านต้นทางเพื่อยืนยันการรับข้อมูลที่ถูกต้องเรียบร้อย ในกรณีที่มีการส่งข้อมูลหลายๆหน้า เมื่อจบหน้าแรกจะมีการส่งสัญญาณอื่นๆ ที่แสดงว่ามีข้อมูลที่จะส่งต่อไปอีกแทนการส่งสัญญาณ EOP เช่น EOM(End of Message) หรือ MPS (Multi Page Signal)

เฟส E ปลดสาย (Call release) เป็นขั้นตอนเลิกการติดต่อระหว่างเครื่องโทรสารต้นทาง และเครื่องโทรสารปลายทาง อาจเป็นแบบใช้อัตโนมัติหรือไม่ก็ได้ โดยเครื่องโทรสารด้านผู้เรียกจะส่งสัญญาณ DCN (Disconnect) ไปยังด้านปลายทางเพื่อยกเลิกการติดต่อ



รูปที่ 2.3 แสดงขั้นตอนการรับส่งข้อมูลระหว่างเครื่องโทรสารจำนวน 1 หน้า

2.2 โมเด็ม (MODEM)

โมเด็ม (MODEM) เป็นคำที่ย่อมาจากคำว่า MOdulate-DEModulate ซึ่งเป็นหน้าที่การทำงานของโมเด็ม คือ ทำการมอดูเลชัน (Modulation) และ ดีมอดูเลชัน (Demodulation) สัญญาณ โดยโมเด็มจะทำหน้าที่แปลงสัญญาณดิจิตอล (Digital) จากคอมพิวเตอร์ที่ส่งมาทาง RS-232 ให้กลายเป็นสัญญาณอนาล็อก (Analog)แล้วส่งออกไปตามสายส่ง ขบวนการนี้เราเรียกว่าการมอดูเลท (Modulate)สัญญาณ ในทางกลับกัน เมื่อโมเด็มได้รับสัญญาณเข้ามา ก็จะแปลงสัญญาณอนาล็อกที่ได้กลับมาเป็นสัญญาณดิจิตอลแล้วจึงส่งให้คอมพิวเตอร์ในรูปของสัญญาณดิจิตอลผ่านทาง RS-232 เช่นกัน ขบวนการแปลงสัญญาณกลับนี้เรียกว่า การดีมอดูเลต(Demodulate)

2.2.1 โมเด็มกับการสื่อสารข้อมูล

ข้อมูลที่ใช้อยู่ในระบบคอมพิวเตอร์เป็นข้อมูลแบบดิจิตอล มีลักษณะเป็นสัญญาณแบบไม่ต่อเนื่อง คือเป็นสัญญาณของข้อมูล "0" และ "1" ส่วนสัญญาณไฟฟ้าอีกรูปแบบหนึ่งเป็นสัญญาณที่มีความต่อเนื่องเรียกว่าสัญญาณอนาล็อก

ในการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ในระยะทางไม่ไกลมากนัก อาจใช้การรับส่งแบบอนุกรมแบบตามมาตรฐาน RS-232C ซึ่งส่งข้อมูลดิจิตอล ของคอมพิวเตอร์ไปตามสายจนถึงผู้รับ กรณีนี้จะสามารถรับส่งข้อมูลได้ไกลถึง 35 เมตร ตามคุณสมบัติของ RS-232C ซึ่งขึ้นอยู่กับคุณภาพของสายส่งด้วย ถ้าสายส่งคุณภาพดี อาจส่งได้ไกลถึง 150 เมตร ที่ความเร็ว 9600 บิต(Bit)ต่อวินาที สำหรับการส่งข้อมูลระยะไกล การส่งข้อมูลดิจิตอลโดยตรงอาจเกิดปัญหาหลายอย่าง ปัญหาที่สำคัญก็คือสัญญาณรูปสี่เหลี่ยมของสัญญาณดิจิตอลเมื่อส่งไปไกลๆ สัญญาณจะเพี้ยนหรือมีรูปร่างผิดไปจากเดิมได้ง่าย ทำให้สายส่งและวงจรรับส่งสัญญาณดิจิตอลต้องถูกออกแบบมาอย่างดี ราคาของสายส่งสัญญาณแบบดิจิตอลจึงมีราคาแพงกว่าสายส่งสัญญาณแบบอนาล็อกมาก ในทางปฏิบัติเราอาจรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์สองเครื่องโดยใช้สัญญาณดิจิตอลผ่านสายส่งได้ ซึ่งทั้งสายส่งและวงจรเชื่อมต่อทั้งหมดเป็นแบบดิจิตอล แต่ค่าใช้จ่ายของทั้งระบบจะสูงมากจึงไม่คุ้มกับการลงทุน วิธีที่จะสามารถหลีกเลี่ยงคือ การส่งข้อมูลไปตามสายส่งในแบบอนาล็อกแทน การทำเช่นนี้เราจำเป็นต้องใช้อุปกรณ์เข้าช่วยแปลงสัญญาณในการรับส่งข้อมูลทั้งสองด้าน ซึ่งเป็นที่มาของโมเด็มนั่นเอง

จากการที่โมเด็มแปลงสัญญาณดิจิตอลจากคอมพิวเตอร์ให้กลายเป็นสัญญาณอนาล็อกในการรับส่งข้อมูลนี้เอง ถ้าโมเด็มแปลงสัญญาณออกมาอยู่ในรูปเสียง ซึ่งเป็นสัญญาณอนาล็อกแบบหนึ่ง เราก็สามารถรับส่งข้อมูลผ่านทางสายโทรศัพท์ได้ โมเด็มทั่วๆไปที่เราใช้งานจะเป็นโมเด็มที่แปลงสัญญาณจากคอมพิวเตอร์ให้อยู่ในรูปของคลื่นเสียงทั้งนั้น เมื่อโมเด็มได้รับข้อมูลดิจิตอลจากคอมพิวเตอร์มันจะเปลี่ยนให้กลายเป็นสัญญาณอนาล็อก จากนั้นก็นำสัญญาณอนาล็อกที่ได้นี้มารวมเข้ากับสัญญาณพาหะ (Carrier Wave) แล้วส่งออกไปทางสายส่งข้อมูล

โดยปกติเมื่อโมเด็มติดต่อกันได้แล้วจะยังไม่สามารถรับส่งข้อมูลได้ทันที แต่จะต้องตกลงรายละเอียดของวิธีการรับส่งข้อมูลกันก่อน เรียกว่าการทำ Handshaking ซึ่งมีขั้นตอนดังนี้คือ เมื่อโมเด็มปลายทางตอบรับแล้ว โมเด็มต้นทางจะทำการทดสอบสภาพสายก่อนว่าสามารถรับส่งข้อมูลได้ดีเพียงใด

เพื่อที่จะได้เลือกความเร็วที่สูงที่สุดเท่าที่สายจะรับได้ จากนั้นจะทดสอบความเพี้ยนของสัญญาณต่อไป หลังจากขั้นตอนนี้โมเด็มจะตกลงกับปลายทางได้ว่าจะใช้ความเร็วในการรับส่งข้อมูลเท่าไหร่ ใช้การผสมสัญญาณแบบไหน ที่ความถี่เท่าใด ถ้าใช้ความเร็วสูงสุดตามที่กำหนดไม่ได้โมเด็มก็จะทำการลดความเร็วลงและทดสอบสภาพสายใหม่จนกระทั่งได้ความเร็วที่สามารถรับส่งข้อมูลได้อย่างถูกต้อง ซึ่งการลดความเร็วของโมเด็มจะลดลงเป็นขั้นๆ ตามแต่ละมาตรฐานที่กำหนดไว้ จากนั้นโมเด็มจะตกลงกับปลายทางว่าจะใช้วิธีการตรวจสอบความผิดพลาด(Error Detection) แบบใด และใช้การลดขนาดข้อมูล (Data Compression) หรือไม่ ขั้นตอนในการทำ Handshaking ของโมเด็มทั้งหมดจะใช้เวลาหลายสิบวินาที เมื่อเสร็จเรียบร้อยแล้วจึงสามารถรับส่งข้อมูลกันได้

2.2.2 โมเด็มสำหรับสายตรงและสายโทรศัพท์

โมเด็มแบ่งตามการใช้งานได้สองแบบคือ โมเด็มที่ใช้กับสายตรง (Leased Line) และโมเด็มที่ใช้กับสายโทรศัพท์ (Dial-up line) โมเด็มที่ใช้กับสายตรงหรือสายเช่าจะส่งข้อมูลด้วยความเร็วสูงจนถึงสูงมาก (9600 บิตต่อวินาทีไปจนถึง 2 ล้านบิตต่อวินาที) ผ่านสายที่ลากตรงไปยังจุดหมายปลายทางตายตัวซึ่งเป็นการติดต่อในลักษณะจุดถึงจุด (Point to Point) จะต่อไปยังจุดอื่นๆไม่ได้ ส่วนมากจะเป็นการใช้ติดต่อส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่มีข้อมูลส่งไปมาจำนวนมาก เช่น เครือข่ายเอทีเอ็ม(ATM) ของธนาคาร, เทอร์มินอล (Terminal) ของคอมพิวเตอร์ขนาดใหญ่ ซึ่งอาจจะเป็นมินิคอมพิวเตอร์(Mini Computer) หรือเมนเฟรม (Mainframe)เป็นต้น การส่งข้อมูลมักจะส่งเป็นกลุ่มและมีซอฟต์แวร์(Software) ควบคุมการรับส่งโดยเฉพาะ ข้อดีของโมเด็มแบบที่ใช้กับสายตรง คือส่งข้อมูลได้ที่ความเร็วสูง เนื่องจากสายส่งจะมีคุณภาพดีเหมาะกับงานส่งข้อมูลจำนวนมากระหว่างจุดสองจุด ข้อเสียก็คือไม่สามารถเปลี่ยนจุดรับข้อมูลไปตามที่ต่างๆได้จึงขาดความคล่องตัว

โมเด็มแบบใช้กับสายโทรศัพท์ จะรับส่งข้อมูลด้วยความเร็วตั้งแต่ 300 บิตต่อวินาทีจนถึง 56000 บิตต่อวินาทีผ่านเครือข่ายโทรศัพท์ เราสามารถรับส่งข้อมูลไปยังที่ต่างๆได้ตามต้องการ โดยไม่กำหนดจุดรับข้อมูลตายตัวเหมือนอย่างโมเด็มสายตรง เครื่องคอมพิวเตอร์ส่วนบุคคล (PC) และบริการรับส่งข้อมูลต่างๆจะใช้โมเด็มแบบนี้ในการทำงานเกือบทั้งหมด การรับส่งข้อมูลจะเป็นการรับส่งทีละหนึ่งตัวอักษรไม่ส่งเป็นกลุ่ม เรียกว่าการส่งแบบ อะซิงโครนัส (Asynchronous) ซึ่งโดยปกติแล้วการส่งข้อมูลผ่านสายโทรศัพท์จะส่งด้วยความเร็ว9600ถึง28800 บิตต่อวินาทีเท่านั้น

ความเร็วสูงสุดที่เชื่อถือได้ในการรับส่งข้อมูลของโมเด็มคือ 28800 บิตต่อวินาที ถ้าใช้โมเด็มความเร็วสูงกว่านี้รับส่งข้อมูลผ่านสายโทรศัพท์เช่น 33600 บิตต่อวินาทีไปจนถึง 56000 บิตต่อวินาทีอาจทำได้ในบางพื้นที่ แต่ถ้าส่งข้อมูล ไกลๆระหว่างจังหวัดหรือประเทศอาจจะมีความผิดพลาดสูง จึงถือความเร็ว 28800 บิตต่อวินาทีเป็นความเร็วสูงสำหรับส่งข้อมูลผ่านสายโทรศัพท์ในปัจจุบัน ข้อดีของการใช้โมเด็มแบบนี้คือ มีความคล่องตัวสูงสามารถรับส่งข้อมูลไปยังที่ต่างๆได้ไม่จำกัด และไม่จำเป็นต้อง จัดหาวงจรสายตรงมาเป็นพิเศษเพื่อส่งข้อมูล ข้อเสียของโมเด็มประเภทนี้คือ ความเร็วในการส่งข้อมูลต่ำกว่าโมเด็มแบบสายตรง และการส่งข้อมูลเป็นจำนวนมากไปยังจุดปลายจุดเดียวในระยะทางไกลค่าโทรศัพท์ทางไกลอาจจะแพงกว่าการเช่าวงจรสายตรงมาใช้ก็ได้

โมเด็มที่มีขายในท้องตลาดปัจจุบัน บางรุ่นอาจทำงานได้เฉพาะกับสายตรงหรือบางรุ่นอาจทำงานได้กับสายโทรศัพท์เท่านั้น โมเด็มที่ใช้กับสายตรงได้เพียงอย่างเดียวจะนำมาต่อใช้กับสายโทรศัพท์ไม่ได้ และโมเด็มที่ใช้กับสายโทรศัพท์ได้เพียงอย่างเดียวก็จะนำมาใช้งานต่อกับสายตรงไม่ได้เช่นกัน ยกเว้นโมเด็มบางชนิด ซึ่งเลือกการทำงานในตัวได้ว่าจะต่อกับสายตรงหรือสายโทรศัพท์ โมเด็มที่ใช้งานได้ทั้งสองอย่างก็มักจะมีราคาแพงกว่าโมเด็มที่ใช้กับสายโทรศัพท์เพียงอย่างเดียว ในกรณีที่ไม่ต้องการใช้การต่อเครื่องคอมพิวเตอร์ส่วนบุคคลเข้ากับเมนเฟรมหรือมินิคอมพิวเตอร์ การใช้โมเด็มแบบต่อกับสายโทรศัพท์ได้อย่างเดียวก็นับว่าเพียงพอแล้วสำหรับรับส่งข้อมูลทั่วไป

2.2.3 โมเด็มในยุคแรก

โมเด็มรุ่นแรกๆที่ใช้รับส่งข้อมูล จะทำหน้าที่แปลงสัญญาณจากคอมพิวเตอร์ให้กลายเป็นความถี่เสียงแล้วส่งไปตามสายเท่านั้น ไม่มีความสามารถในการหมุนโทรศัพท์อัตโนมัติอย่างโมเด็มในปัจจุบัน โมเด็มแบบที่ไม่มีการต่อเข้ากับสายโทรศัพท์โดยตรงเราเรียกว่าโมเด็มแบบ Acoustic Coupler ซึ่งจะมีรูปร่างรองรับกับหูฟัง(Handset)ของโทรศัพท์แบบมาตรฐาน เมื่อคอมพิวเตอร์ส่งข้อมูลไปให้ผู้รับ Acoustic Coupler จะเปลี่ยนข้อมูลนั้นให้กลายเป็นสัญญาณเสียง แล้วส่งออกทางลำโพงของมัน หูฟังของเครื่องโทรศัพท์จะรับสัญญาณเสียงนี้ส่งไปให้ปลายทางอีกทีหนึ่ง เมื่อเครื่องโทรศัพท์ส่งสัญญาณเสียงให้ Acoustic Coupler มันก็รับสัญญาณเสียงนี้ผ่านไมโครโฟน และนำมาเปลี่ยนเป็นสัญญาณดิจิตอลให้คอมพิวเตอร์ต่อไป Acoustic Coupler มีข้อดีคือ มันไม่จำเป็นต้องต่อเข้ากับสายโทรศัพท์ ทำให้เราใช้ที่ไหนก็ได้ที่ส่งข้อมูล เช่น โทรศัพท์สาธารณะ โทรศัพท์ในห้องพักของโรงแรม ฯลฯ ซึ่งที่ต่างๆเหล่านี้บางครั้งเราจะไปตัดต่อสายโทรศัพท์ให้มาเข้ากับโมเด็มโดยตรงไม่ได้

สำหรับข้อเสียของโมเด็มชนิดนี้ก็คือรับส่งข้อมูลได้ช้า เนื่องจากความเร็วมาตรฐานของ Acoustic Coupler อยู่ที่ 300บิตต่อวินาทีเท่านั้น และจากการที่มันฟังเสียงจากเครื่องโทรศัพท์ ทำให้เสียงรบกวนจากภายนอกเข้ารบกวนรับส่งข้อมูลได้ง่าย จึงไม่เหมาะที่จะใช้งานในสถานที่ที่มีเสียงดัง และข้อเสียสำคัญอีกข้อก็คือ ไม่สามารถใช้งานกับหูฟังของโทรศัพท์ที่มีขนาดและรูปร่างไม่ได้มาตรฐาน เช่นโทรศัพท์แบบแบนๆ เป็นต้น

โมเด็มในปัจจุบันจะรับส่งข้อมูลผ่านสายโทรศัพท์ โดยต่อสายโดยตรงไม่ต้องส่งเสียงผ่าน Acoustic Coupler สายโทรศัพท์จะต่อเข้าโมเด็มโดยใช้ปลั๊กโทรศัพท์ที่เรียกว่า RJ-11 การต่อสายโทรศัพท์เข้ากับโมเด็มโดยตรงจะช่วยป้องกันสัญญาณรบกวนได้ดีขึ้น และทำให้การส่งข้อมูลได้เร็วกว่าแบบเก่า แต่การใช้งานลักษณะนี้มีข้อเสียคือถ้าโทรศัพท์ที่ใช้ไม่มีปลั๊กแบบ RJ-11 เพื่อต่อเข้ากับสายโทรศัพท์ โมเด็มก็จะใช้สายโทรศัพท์นั้นรับส่งข้อมูลไม่ได้ ทำให้ยุ่งยากเวลาต้องการรับส่งข้อมูลจากนอกสถานที่ เช่น จากที่พักในโรงแรม เป็นต้น

2.2.4 องค์ประกอบในการสื่อสารข้อมูล

จากการที่โมเด็มต่อกับคอมพิวเตอร์ผ่านทาง RS-232C และส่งข้อมูลไปยังเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งดังนั้นองค์ประกอบในการรับส่งข้อมูล (Communication Parameter) จึงต้องตรงกันตลอด

เส้นทางที่รับส่ง องค์ประกอบดังกล่าวคือ ความเร็ว (Speed) จำนวนบิตของข้อมูล (Data Bit) จำนวนบิตเริ่มและบิตจบ (Start Bit, Stop Bit) การตรวจสอบพาริตี้บิต (Parity Bit) และการเลือกใช้ Echo(Duplex)ในการรับส่ง

ความเร็ว คืออัตราการรับส่งข้อมูลเป็นจำนวนบิตต่อวินาที โมเด็มที่ด้านรับและคอมพิวเตอร์เครื่องรับจะต้องมีความเร็วในการรับส่งข้อมูลเท่ากัน ความเร็วที่ใช้กันทั่วไปมีตั้งแต่ 2400 4500 9600 19200 บิตต่อวินาทีไปจนถึง 38400 57600 หรือ 115000 บิตต่อวินาที ข้อสังเกตคือความเร็วในการส่งผ่านข้อมูลตลอดทั้งเส้นทางนั้นกับ การส่งข้อมูลจริงในสายแต่ละช่วง ระหว่างโมเด็มสองตัวอาจไม่สูงเท่านี้ เนื่องจากมีการใช้เทคโนโลยีในการย่อขนาดของข้อมูลเข้ามาช่วย ทำให้ปริมาณข้อมูลที่ต้องส่งลดลง

จำนวนบิตของข้อมูล คือการกำหนดใช้จำนวนข้อมูลกี่บิตในการส่ง ปกติจะเลือกได้สองแบบคือ 7 บิต ต่อหนึ่งตัวอักษร หรือ 8 บิตต่อหนึ่งตัวอักษร การรับส่งข้อมูลภาษาไทย เราจำเป็นต้องใช้แบบ 8 บิตต่อหนึ่งตัวอักษรเท่านั้น เนื่องจากภาษาไทยเราใช้รหัสครบทั้ง 8 บิต ถ้าหากส่งในแบบ 7 บิตต่อหนึ่งตัวอักษรรหัสภาษาไทยจะถูกตัดบิตที่ 8 ทิ้งไป กลายเป็นตัวภาษาอังกฤษ ส่วนจำนวนบิตเริ่มและบิตจบนั้นกำหนดไว้ให้ตัวส่งแยกออกว่าข้อมูลจะเริ่มต้นเมื่อไหร่ และจบลงเมื่อใด

บิตเริ่มต้น มักจะใช้หนึ่งบิตเสมอ และบิตจบข้อมูล จะมีสองแบบคือ 1 บิตและ 2 บิตการใช้งานทั่วไปมักจะใช้แบบ 1 บิต

การตรวจสอบพาริตี เป็นการตรวจสอบความถูกต้องของการรับส่งข้อมูลที่ส่งมา โดยสามารถเลือกให้ตรวจสอบว่าข้อมูลที่เป็น "1" เป็นจำนวนคู่(Even)หรือจำนวนคี่(Odd) หรือไม่ต้องตรวจสอบ (None) เช่น ถ้าส่งข้อมูล 11000001 ไปให้ผู้รับ และมีการตรวจสอบพาริตีแบบจำนวนคู่ พาริตีบิตในกรณีนี้จะมีค่าเป็น "1" เพื่อให้จำนวน "1" ทั้งหมดมีจำนวนเป็นเลขคู่ ถ้ากำหนดการตรวจสอบพาริตีเป็นจำนวนคี่ พาริตีก็จะเป็น "0" ให้จำนวน"1" ของข้อมูลทั้งหมดเป็นเลขคี่ และถ้าไม่มีการตรวจสอบเลย เครื่องส่งก็จะส่งบิตจบปิดท้ายข้อมูลทันที ไม่มีการส่งพาริตีบิตไปให้ผู้รับถ้ารับส่งข้อมูลแบบ 7 บิต เรามักตั้งการตรวจสอบพาริตีเอาไว้ แต่ถ้าส่งรับข้อมูลแบบ 8 บิต ก็จะไม่มีการตรวจสอบพาริตีบิต สำหรับการเลือกใช้ Echo ในการรับส่งก็เป็นการเลือกให้สอดคล้องกับ Full Duplex หรือ Half Duplex นั่นเอง ถ้าเป็นการรับส่งแบบ Full Duplex ก็ต้องเลือกใช้ Echo off และถ้าเป็นแบบ Half Duplex เราก็เลือกใช้ Echo on เมื่อองค์ประกอบนี้ทั้งหมดตรงกัน การรับส่งข้อมูลก็จะทำได้ถูกต้อง ถ้าส่วนใดส่วนหนึ่งไม่ว่าจะเป็นเครื่องคอมพิวเตอร์ด้านส่ง โมเด็มด้านส่ง โมเด็มด้านรับ หรือคอมพิวเตอร์ด้านรับ เพียงส่วนเดียวใช้องค์ประกอบในการรับส่งข้อมูลผิดไป การรับส่งข้อมูลจะผิดพลาดได้ ทั้งผู้รับและผู้ส่งต้องตกลงกันให้แน่นอนว่าจะใช้องค์ประกอบในการรับส่งข้อมูลแต่ละตัวอย่างไร

2.2.5 มาตรฐานของโมเด็มตาม CCITT (ITU-T) V-Series

โมเด็มจำเป็นต้องมีมาตรฐานเช่นเดียวกับอุปกรณ์อื่นๆ เพื่อให้ผู้ผลิตแต่ละบริษัทผลิตโมเด็มออกมาแล้วใช้รับส่งข้อมูลกับบริษัทอื่นๆได้ มาตรฐานของโมเด็มจะแบ่งออกเป็นสองส่วน คือมาตรฐานในส่วนของฮาร์ดแวร์ที่โมเด็มใช้ และอีกส่วนหนึ่งคือมาตรฐานในส่วนของซอฟต์แวร์ที่ควบคุมการทำงานของโมเด็มหรือคำสั่งของโมเด็ม โดยมาตรฐานนี้กำหนดขึ้นจากองค์การมาตรฐานสื่อสารสากล หรือ

CCITT (International Telephone and Telegraph Consultative Committee) ที่ปัจจุบันเปลี่ยนชื่อเป็น ITU-T (International Telecommunication Union-Telecommunication) ซึ่งจะครอบคลุมเกี่ยวกับความเร็วในการรับส่งข้อมูล, ความถี่ที่ใช้ และเทคนิคการผสมสัญญาณในสาย ฯลฯ

มาตรฐานทางการสื่อสารและรับส่งข้อมูล เนื่องจากมีสมาชิกอยู่เกือบทุกประเทศทั่วโลก ผู้ผลิตรายใหญ่จึงปฏิบัติตามมาตรฐานของ CCITT หรือ ITU-T ทำให้โมเด็มคนละยี่ห้อกันที่ใช้มาตรฐานเดียวกันสามารถรับส่งข้อมูลได้ทันที

ความเร็วในการรับส่งข้อมูลของโมเด็มนั้นกำหนดเป็น บิตต่อวินาที (bit per second, bps) หรือ บิตเรท(bit rate) ซึ่งแตกต่างจากอัตราการเปลี่ยนแปลงของสัญญาณไฟฟ้าในสายส่งหรือที่เรียกกันว่า Baud Rate ในสมัยก่อนการรับส่งข้อมูลใช้เทคนิคการผสมสัญญาณแบบง่ายๆ เช่นการเปลี่ยนแปลงความถี่ตามข้อมูล “0” และ “1” ที่ได้รับ อัตราการส่งข้อมูลละอัตราการเปลี่ยนแลงของสัญญาณในสายส่งจึงมีค่าเท่ากัน หรือพูดอีกอย่างว่า สัญญาณรูปคลื่นเพียงหนึ่งลูก จะแทนข้อมูลเพียง 1 บิต เราจึงถือว่าอัตราการเปลี่ยนแปลงข้อมูลของสัญญาณในสายส่ง ก็คืออัตราการส่งข้อมูลนั่นเอง ต่อมาเทคนิคการผสมสัญญาณซับซ้อนมากขึ้น ทำให้เราสามารถส่งข้อมูลได้มากขึ้นกว่าเดิม โดยอัตราการเปลี่ยนแปลงของสัญญาณในสายยังคงเท่าเดิม ดังนั้นเมื่อเราพูดว่า โมเด็มรับส่งข้อมูลที่ความเร็ว 1200 Baud เราจะไม่ทราบเลยว่าโมเด็มนั้นรับส่งข้อมูลได้กี่บิตต่อวินาที เนื่องจากว่าถ้าโมเด็มผสมสัญญาณ 1 บิตต่อหนึ่งลูกคลื่น โมเด็มนั้นจะรับส่งข้อมูลที่ความเร็ว 1200 บิตต่อวินาที หรือถ้าโมเด็มผสมสัญญาณ 4 บิตต่อหนึ่งลูกคลื่นที่เปลี่ยนแปลงในสายส่ง โมเด็มจะรับส่งข้อมูลได้เร็วถึง 4800 บิตต่อวินาที โดยยังคงมีอัตราการเปลี่ยนแปลงสัญญาณในสายส่งเท่ากับ 1200 Baud เหมือนเดิม เพราะฉะนั้นเราจึงเลิกใช้คำว่า Baud Rate สำหรับบอกความเร็วการรับส่งข้อมูลของโมเด็ม และหันมาใช้คำว่า บิตเรท หรืออัตราการส่งข้อมูลเป็นบิตต่อวินาทีแทน ซึ่งสื่อความหมายได้เข้าใจตรงกันมากกว่า

มาตรฐานของโมเด็มที่ใช้อยู่ในปัจจุบัน เป็นไปตามที่องค์การมาตรฐานสื่อสารสากล หรือ CCITT (ITU-T) เป็นผู้กำหนดขึ้น โดยมีชื่อเรียกแต่ละมาตรฐานของโมเด็มขึ้นต้นด้วยตัวอักษร “V” และตามด้วยตัวเลข จึงมีการเรียกมาตรฐานนี้อีกชื่อหนึ่งว่า V-Series นอกจากมาตรฐานของโมเด็มแล้ว CCITT หรือ ITU-T ยังเป็นผู้กำหนดมาตรฐานทางการสื่อสารอื่นๆ เช่น มาตรฐานของการสื่อสารผ่านดาวเทียม, มาตรฐานของโทรสาร (Fax :Facsimile) มาตรฐานการสื่อสารข้อมูลต่างๆ ทั้งในแบบดิจิตอล และอนาล็อก รวมถึงมาตรฐานเกี่ยวกับระบบโทรศัพท์อีกด้วย มาตรฐานที่ CCITT หรือ ITU-T เป็นผู้กำหนดได้รับการยอมรับกันทั่วโลก การติดต่อสื่อสารระหว่างประเทศจึงเป็นไปได้อย่างไม่มีปัญหา เนื่องจากทุกๆคนต่างก็ทำตามมาตรฐานเดียวกัน

มาตรฐานที่ขึ้นต้นด้วยตัวอักษร V ไม่ใช่มาตรฐานของโมเด็มทั้งหมด บางมาตรฐานอาจหมายถึงการเชื่อมต่อแบบอื่นๆก็ได้ เช่น V.24 เป็นมาตรฐานการรับส่งข้อมูลแบบอนุกรมเทียบได้กับ RS-232C นั่นเอง และ V.35 หมายถึงการรับส่งข้อมูลแบบอนุกรมความเร็วสูงเป็นต้น ในที่นี้จะกล่าวถึงมาตรฐานของโมเด็มแบบต่างๆ ที่ใช้กันมากตาม CCITT (ITU-T) V-Series ตั้งแต่ความเร็วต่ำไปจนถึงความเร็วสูงและของแต่ละมาตรฐานเป็นดังนี้

V.90 พัฒนาจากมาตรฐาน V.34bis ซึ่งใช้สำหรับโมเด็มความเร็ว 56000 บิตต่อวินาที มาตรฐานนี้ ITU-T ประกาศออกใช้งาน เมื่อเดือนกันยายน ปี 1998 โดยอาศัยหลักการที่เครือข่ายดิจิตอลสามารถรองรับการจราจรทางเสียงที่มีความเร็ว 64 Kbps ต่อสายแต่ละเส้น และการที่จะ digitize (แปลงสัญญาณอนาล็อกเป็นดิจิตอล) สัญญาณเสียงอนาล็อกที่มีความถี่ 3 KHz นั้น จำเป็นต้องทำการสุ่มสัญญาณ(Sampling) เสียงด้วยอัตราสองเท่าของความถี่นั้น โมเด็มความเร็ว 56 Kbps ใช้ประโยชน์จากข้อเท็จจริงที่ว่า AOL(American OnLine – ผู้ให้บริการอินเตอร์เน็ตรายใหญ่ที่สุดในสหรัฐอเมริกา) ผู้ให้บริการอินเตอร์เน็ตรายใหญ่อื่นๆ ล้วนแต่มีการเชื่อมโยงระบบดิจิตอลโดยตรงกับเครื่องข่ายโทรศัพท์อยู่แล้ว และไม่จำเป็นต้องทำการแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลด้วย ในขั้นตอนการส่งขึ้น(Upstream) นี้จะช่วยสงวนรักษาแบนด์วิดท์ของช่องทางดิจิตอลโดยสมบูรณ์ตลอดทางไปยังจุดสุดท้าย ข้ามเครือข่ายโทรศัพท์ไปยังผู้ใช้ด้วยความเร็ว 56 Kbps หรือก็คือช่วงขาลง (Downstream)

V.92 เป็นข้อกำหนดหรือมาตรฐานใหม่สำหรับ Dial-up โมเด็มความเร็ว 56000 บิตต่อวินาที จะใช้เทคนิคการผสมสัญญาณแบบใหม่ที่แตกต่างไปจากเดิมอย่างสิ้นเชิง คือใช้หลักการของการเปลี่ยนสัญญาณจากดิจิตอลเป็นสัญญาณอนาล็อก (Digital to Analog Converter) มาใช้แทนการผสมสัญญาณในแบบ Trellis Encoding การรับส่งข้อมูลจะแปลงข้อมูลดิจิตอล ครั้งละ 7 บิตส่งออกไป 8000 ครั้งต่อวินาที (7 x 8000 = 56000 บิตต่อวินาที) แล้วส่งออกไปเป็นสัญญาณอนาล็อกตามสายโทรศัพท์ โมเด็มชนิดนี้จะมีความเร็วในการรับส่งข้อมูลไม่เท่ากันทั้งสองด้าน คือด้านรับข้อมูลเข้ามาจะมีความเร็ว 56000 บิตต่อวินาที แต่ด้านส่งข้อมูลออกไปจะมีความเร็วเพียง 28800 บิตต่อวินาที หรือ 33600 บิตต่อวินาทีเท่านั้น ส่วนโมเด็มด้านส่งก็จะกลับกันคือ ส่งข้อมูลที่ความเร็ว 56000 บิตต่อวินาที ละรับข้อมูลเข้ามาที่ความเร็ว 28800 หรือ 33600 บิตต่อวินาทีตามมาตรฐาน V.34 ซึ่งเป็นมาตรฐาน V.92 ที่กำหนดโดย ITU-T ประกอบด้วยคุณสมบัติใหม่ 3 ประการ ได้แก่ Quick Connect , Modem on Hold , V.PCM Upstream

มาตรฐานของโมเด็ม V-Series ที่กล่าวถึงทั้งหมดนี้ เป็นมาตรฐานที่พบเห็นได้ทั่วไป ซึ่งยังมีบางมาตรฐานไม่ได้นำมาพูดถึง เนื่องจากมีที่ใช้งานพิเศษเฉพาะเท่านั้น ส่วนมาตรฐานของโมเด็มตามแบบสหรัฐอเมริกา หรือที่เราเรียกว่า Bell Standard ปัจจุบันค่อยๆ ลดความนิยมลง เนื่องจากประเทศต่างๆพากันใช้ตามมาตรฐานของ CCITT หรือ ITU-T เป็นหลัก และในประเทศไทยก็ใช้ตามมาตรฐานของ CCITT หรือ ITU-T เช่นกัน

2.2.6 มาตรฐานคำสั่งของโมเด็ม

โมเด็มในยุคแรกๆทำหน้าที่เปลี่ยนสัญญาณข้อมูลของคอมพิวเตอร์เป็นสัญญาณเสียงแล้วส่งออกไปทางสายโทรศัพท์และรับกลับมาเท่านั้น ไม่มีฟังก์ชันพิเศษช่วยการทำงานให้สะดวกขึ้นแต่อย่างใด ต่อมาจึงมีผู้คิดเพิ่มเติมความสามารถให้กับโมเด็มขึ้นให้เราใช้งานง่ายขึ้นและสะดวกรวดเร็วกว่าเดิม โดย

การส่งคำสั่งไปให้โมเด็มทำฟังก์ชันต่างๆ เมื่อมีคำสั่งของโมเด็มเกิดขึ้นก็เป็นเรื่องของมาตรฐานที่จะต้องติดตามมาเพื่อควบคุมให้การสั่งงานโมเด็มเป็นไปอย่างเดียวกัน เหมือนอย่างในเรื่องของความเร็วในการรับส่งข้อมูลและเทคนิคการผสมสัญญาณ

โมเด็มในยุคแรกการใช้งานต้องอาศัยคนเข้าช่วยเหลือเกือบทุกอย่าง เนื่องจากโมเด็มทำหน้าที่รับส่งสัญญาณ และเปลี่ยนแปลงสัญญาณที่ได้รับมาให้เครื่องคอมพิวเตอร์เท่านั้นเอง ส่วนการหมุนโทรศัพท์และการติดต่อกับอีกด้านหนึ่งเพื่อส่งข้อมูลไปจนถึงการรับโทรศัพท์สำหรับโมเด็มด้านรับนั้นต้องใช้คนทำสิ่งเหล่านี้ทั้งหมด โมเด็มแบบนี้จึงใช้งานลำบาก ผู้ใช้ต้องมีความเข้าใจระบบการทำงานของโมเด็มพอสมควร การปรับฟังก์ชันการทำงานแบบต่างๆ ของโมเด็มมักจะใช้วิธีผลักสวิตช์เล็กๆ (DIP Switch) บนตัวโมเด็ม ถ้าเราไปดูโมเด็มแบบเก่า จะเห็นสวิตช์ตัวไหนใช้ทำอะไรและต้องผลักสวิตช์ไปทางไหน การสั่งงานโมเด็มโดยใช้คำสั่งจากคอมพิวเตอร์จึงเกิดขึ้น

โดยปกติโมเด็มจะติดต่อกับเครื่องคอมพิวเตอร์เพื่อทำหน้าที่รับส่งข้อมูลอยู่แล้วดังนั้นขณะที่โมเด็มยังไม่ได้รับการติดต่อกับปลายทางเพื่อส่งข้อมูล คอมพิวเตอร์จะสามารถส่งคำสั่งต่างๆให้โมเด็มโดยไม่ได้รบกวนการส่งข้อมูลแต่อย่างใด ในขั้นแรก เมื่อเปิดสวิตช์ให้โมเด็มทำงาน สัญญาณทุกอย่างที่ได้รับจากคอมพิวเตอร์ จะถือว่าเป็นคำสั่งทั้งหมดจนกว่าจะติดต่อกับโมเด็มปลายทางได้ และเมื่อโมเด็มเริ่มทำการส่งข้อมูลผ่านสายส่งแล้วสัญญาณต่างๆ ที่คอมพิวเตอร์ส่งให้โมเด็มจะถือว่าเป็นข้อมูลทั้งหมดจนกว่าจะหยุดส่งข้อมูลโดยการเลิกการติดต่อกับปลายทางหรือวางสายโทรศัพท์แล้ว โมเด็มจึงกลับมาอยู่ในภาวะที่คอยรับคำสั่งจากคอมพิวเตอร์อีกครั้งหนึ่ง

บริษัท Hayes Microcomputer Product Inc. เป็นผู้คิดชุดคำสั่งชุดหนึ่งขึ้นมาเพื่อสั่งงานโมเด็มสำหรับเครื่องคอมพิวเตอร์ส่วนบุคคลและได้รับความนิยมเป็นอย่างมากจนถือเป็นมาตรฐานอันหนึ่ง มาตรฐานคำสั่งนี้เรียกว่า Hayes Command Set เป็นคำสั่งที่ช่วยให้ผู้ใช้สามารถกำหนดการทำงานต่างๆ บนโมเด็มได้ โดยใช้ซอฟต์แวร์สั่งจากคอมพิวเตอร์ไปยังโมเด็มโดยตรง ทำให้เราไม่ต้องปรับสวิตช์เพื่อเลือกการทำงานแบบต่างๆของโมเด็มอีกต่อไป โมเด็มที่เราใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลหรือแมคอินทอช (Macintosh) เกือบทั้งหมดจะรับคำสั่งตามมาตรฐานของ Hayes

เมื่อคำสั่งของโมเด็มเป็นสิ่งจำเป็นสำหรับการใช้งานโมเด็มสมัยใหม่ ทาง CCITT หรือ ITU-T ก็ได้มีการกำหนดมาตรฐานของคำสั่งโมเด็มขึ้นเช่นกัน เรียกว่ามาตรฐานคำสั่งแบบ V.25 bis แต่เนื่องจากว่าโมเด็มที่ใช้กับเครื่องคอมพิวเตอร์ และซอฟต์แวร์สั่งงานโมเด็มเกือบทั้งหมดใช้คำสั่งตามมาตรฐานของ Hayes ดังนั้นมาตรฐานคำสั่งโมเด็มแบบ V.25 bis จึงไม่ค่อยมีใช้กันเท่าใดนัก เรียกได้ว่าเครื่องคอมพิวเตอร์ติดต่อกับโมเด็มโดยใช้คำสั่งแบบ Hayes Command ทั้งหมด ส่วนคำสั่งแบบ V.25 bis มีใช้บ้างเหมือนกัน แต่เป็นในระบบชุมสายตามมาตรฐานของ CCITT หรือ ITU-T แต่โมเด็มบางแบบสามารถแบบคำสั่งแบบ Hayes Command และแบบ V.25 bis ของ CCITT หรือ ITU-T แต่โมเด็มราคาถูกที่ใช้กับคอมพิวเตอร์ส่วนมากมักจะรับคำสั่งได้แต่ Hayes Command เท่านั้น ผู้ผลิตโมเด็มบริษัทอื่นๆ ได้ใช้มาตรฐานคำสั่งตามบริษัท Hayes ซึ่งเป็นต้นฉบับ โดยรับคำสั่งจากเครื่องคอมพิวเตอร์ตามแบบคำสั่งของ Hayes ทุกประการ เรียกว่าเป็นโมเด็มแบบ Hayes Compatible ซึ่งเป็นโมเด็มที่ใช้คำสั่งแบบ Hayes นั้นนอกจากจะรับคำสั่งจากคอมพิวเตอร์เหมือนคำสั่งของบริษัท Hayes แล้ว โมเด็มยังต้องตอบสนองต่อคำสั่ง

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

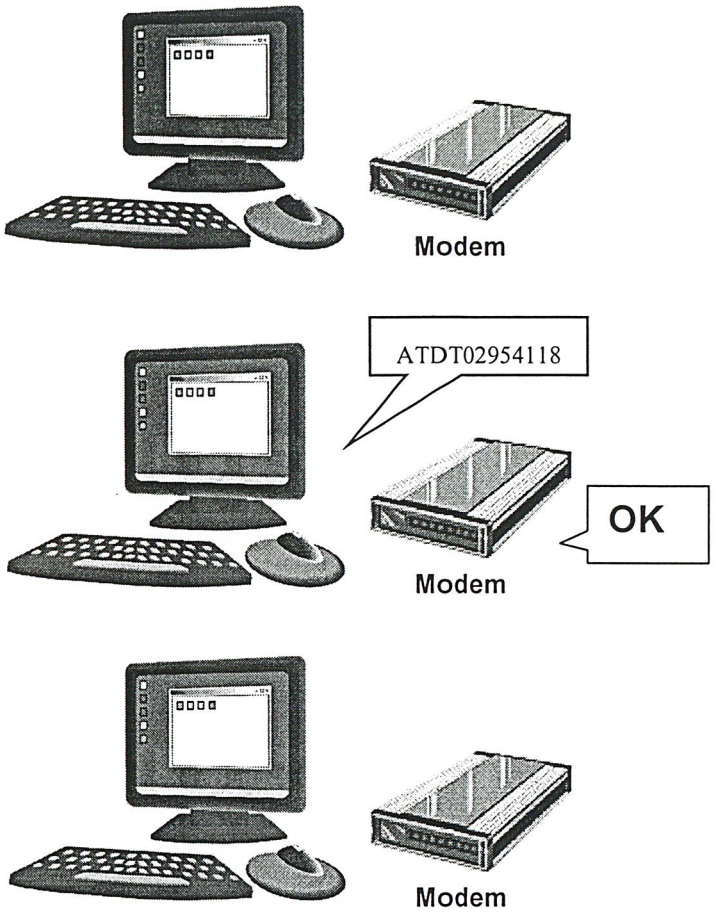
ต่างๆ เหมือนที่บริษัท Hayes ใช้ รวมถึงมีที่เก็บตัวแปรต่างๆ เหมือนกันอีกด้วยเพื่อให้ซอฟต์แวร์ทำงานโต้ตอบกับโมเด็มได้ถูกต้อง

คำสั่งโมเด็มจะควบคุมการทำงานที่จำเป็นทั้งหมดของโมเด็ม เช่น ตอบรับสัญญาณโทรศัพท์ที่เรียกเข้ามา ต่อเข้าสายโทรศัพท์หรือวางสายโทรศัพท์ การตั้งค่าโมเด็มใหม่ สั่งให้โมเด็มหมุนโทรศัพท์ตามเบอร์ที่กำหนด ปรับพารามิเตอร์ต่างๆ ของโมเด็ม ฯลฯ ซึ่งถ้าหากไม่ใช้คำสั่งโมเด็มแล้ว ผู้ใช้จะต้องกำหนดตัวแปรเหล่านั้นด้วยวิธีการผลักสวิตช์บนโมเด็มตามที่ได้กล่าวมาข้างต้น การใช้คำสั่งจึงสะดวกและง่ายต่อการใช้งานมาก ข้อดีอันหนึ่งของการใช้คอมพิวเตอร์ส่งคำสั่งให้โมเด็มคือ ซอฟต์แวร์สำหรับติดต่อสื่อสารสามารถปรับตัวแปรต่างๆ ของโมเด็มให้เป็นไปอย่างที่ต้องการได้ โดยที่ผู้ใช้ไม่ต้องรู้รายละเอียดใดๆ เลย โปรแกรมจะจัดการให้เสร็จและติดต่อส่งข้อมูลทันที โปรแกรมคนละโปรแกรมอาจใช้งานโมเด็มไม่เหมือนกัน ซึ่งแต่ละโปรแกรมก็จะปรับโมเด็มให้ทำงานต่างกันได้โดยไม่ต้องแก้ไขส่วนที่เป็นฮาร์ดแวร์ของโมเด็ม การใช้งานโมเด็มจงมีความคล่องตัวมากกว่าการใช้สวิตช์เลือกแบบเก่า ซึ่งถ้ามีการเปลี่ยนแปลงอะไรเราก็ต้องปรับสวิตช์กันทีหนึ่งทุกครั้งไป และอาจเกิดความผิดพลาดได้ง่ายกว่าการใช้คำสั่งงานโมเด็ม

ภายในโมเด็มจะมีหน่วยความจำพิเศษสำหรับเก็บตัวแปรในการทำงานที่สวิตช์แบบเก่าหน่วยความจำนี้ยังคงเก็บค่าต่างๆ เอาไว้ได้ แม้ว่าเปิดโมเด็มหรือดึงปลั๊กโมเด็มออกก็ตาม โมเด็มที่ใช้คำสั่งของ Hayes เรียกหน่วยความจำส่วนนี้ว่า S-Register เอาไว้เก็บพารามิเตอร์ในการทำงานของโมเด็ม เช่น จำนวนครั้งที่จะตอบรับสัญญาณเรียกเข้า , ช่วงเวลาสำหรับรอสัญญาณก่อนหมุนโทรศัพท์ ฯลฯ

ซอฟต์แวร์ที่ใช้สื่อสารสามารถเปลี่ยนค่าตัวแปรเหล่านั้นชั่วคราว หรือเปลี่ยนค่าถาวรไปเลยก็ได้ โดยคำสั่งเก็บค่าตัวแปรเอาไว้ หน่วยความจำพิเศษนี้บางชนิดใช้แบตเตอรี่เล็กๆ คอยจ่ายไฟให้เวลาที่เราปิดโมเด็ม เพื่อป้องกันค่าต่างๆ หายไปจากหน่วยความจำ ดังนั้นเมื่อใช้โมเด็มไปนานๆ แบตเตอรี่ดังกล่าวจะหมดลง จำเป็นต้องเปลี่ยนอันใหม่ให้ ไม่เช่นนั้นการทำงานของโมเด็มจะผิดพลาดได้ เนื่องจากค่าของตัวแปรในหน่วยความจำหายไป โมเด็มบางชนิดเก็บค่าตัวแปรในหน่วยความจำแบบที่ไม่ต้องใช้แบตเตอรี่จ่ายไฟสำรองให้ โมเด็มแบบนี้เราก็ไม่จำเป็นต้องเปลี่ยนแบตเตอรี่ภายใน และโมเด็มบางแบบก็ไม่ยอมให้เราเปลี่ยนค่าตัวแปรตามที่เราสั่งมาจะเป็นไปชั่วคราวเท่านั้น เมื่อเปิดปิดโมเด็มใหม่ ค่าต่างๆ จะกลับเหมือนเดิมตามที่ผู้ผลิตกำหนดเอาไว้ใน ROM (Read only Memory) ของโมเด็มนั้นเอง

61963



รูปที่ 2.4 แสดงการทำงานของโมเด็มโดยใช้คำสั่ง AT (AT Command)

Hayes Command เป็นคำสั่งที่ใช้สั่งงานโมเด็ม มีอีกชื่อหนึ่งเรียกว่า AT Command เพราะคำสั่งทุกคำสั่งจะต้องขึ้นต้นด้วยอักษร AT เสมอ เมื่อจบคำสั่งให้ปิดท้ายด้วยรหัส ASCII ตัวที่ 13 คือ Carriage Return หรือกดปุ่ม Enter โมเด็มก็จะรับคำสั่งนั้นไปทำงานทันที และตอบคำว่า OK กลับมา บางคำสั่งจะสังเกตว่ามีรหัสหรือตัวเลขต่อท้ายเพื่อระบุวิธีการทำงานโดยละเอียดอีกทีหนึ่ง เช่น ATB อาจตามด้วย 0 หรือ 1 ดังนั้นคำสั่งที่ใช้จริงจึงอาจเป็น ATB1 ก็ได้ ในทำนองเดียวกันบางคำสั่งก็ตามด้วยข้อมูล เช่น ATDT 023266232 คือคำสั่ง ATDT ที่ให้หมุนโทรศัพท์หมายเลข 023266232 เป็นต้น ตัวอย่างคำสั่ง AT

2.3 แฟกซ์โมเด็ม (Fax Modem)

โมเด็มที่ใช้ในเครื่องโทรสารกลุ่ม 3 ได้กำหนดไว้ตามมาตรฐานของ CCITT ดังตารางต่อไปนี้

Bits per Second	Baud Rate	Bits per Sample	Type	Carrier Frequency	Bandwidth in Hertz
14400	2400	6	V.17	1800	550-3050
12000	2400	5	V.17	1800	550-3050
9600	2400	4	V.29	1700	450-2950
7200	2400	3	V.29	1700	450-2950
4800	1600	3	V.27ter	1800	950-2650
2400	1200	2	V.27ter	1800	1150-2450

ตารางที่ 2.2 มาตรฐานของโมเด็มที่ใช้ในเครื่องโทรสารกลุ่ม 3 ตามข้อกำหนดของ CCITT

โมเด็มที่ใช้ในเครื่องโทรสารกลุ่ม 3 จะตรวจสอบการติดต่อระหว่างเครื่องโทรสารทั้ง 2 ด้านก่อนการส่งข้อมูล และโมเด็มจะพยายามส่งข้อมูลด้วยความเร็วสูงสุดเท่าที่จะเป็นไปได้ ส่วนใหญ่โมเด็มในเครื่องโทรสารได้ถูกกำหนดไว้ตามมาตรฐาน V.27ter และ V.29 โดยส่วนใหญ่ในการส่งข้อมูลของเครื่องโทรสารกลุ่ม 3 นั้น จะเริ่มต้นที่ความเร็ว 9600 บิทต่อวินาที แต่ถ้าหากไม่สามารถส่งข้อมูลด้วยความเร็วนี้ได้ โมเด็มจะลดความเร็วในการส่งลงเป็น 7200,4800 หรือ 2400 บิทต่อวินาที แทนตามความเหมาะสม

2.3.1 กลุ่มคำสั่งที่ใช้ควบคุมแฟกซ์โมเด็ม

คำสั่งสำหรับควบคุมแฟกซ์โมเด็มระดับ 1 และ 2 โดย EIA ถูกออกแบบเพิ่มเติมจากกลุ่มคำสั่ง AT ซึ่งสายของอักขระที่ถูกส่งไปยังโมเด็มจะเรียกว่า แถวคำสั่ง (Command line) และจะต้องขึ้นต้นด้วยอักษร AT หรือ at แถวคำสั่งประกอบด้วยสัญลักษณ์ใน รหัสแอสกี้ (ASCII) และลงท้ายด้วยเครื่องหมายแสดงการสิ้นสุดคำสั่งที่เรียกว่า รหัสแอสกี้ 13 (Carriage return)

มีเพียง บิทต่ำ 7 บิทของแต่ละอักษรเท่านั้นที่จะถูกตรวจสอบในขณะที่โมเด็มแปลความหมายของคำสั่งคำสั่งที่เขียนด้วยอักษรตัวใหญ่หรือตัวเล็กจะมีความหมายเหมือนกัน ช่องว่างและเครื่องหมายควบคุมอื่นๆที่นอกเหนือไปจาก รหัสแอสกี้ 13 และ รหัสแอสกี้ 8 (ช่องว่าง) ที่ปรากฏในคำสั่งจะถูกละเลยไป และจะถูกแสดงค่าด้วย 0 ด้วยเหตุ นี้แฟกซ์โมเด็มทุกเครื่องจึงต้องมี การควบคุมการไหล (Flow control) แนบ Xon/Xoff และอาจมีการควบคุมการไหลอื่นๆอีกก็ได้

2.3.2 การเข้ารหัส

ในเครื่องโทรสารกลุ่ม 3 กำหนดให้มีการเข้ารหัสเพื่อให้ข้อมูลที่ต้องส่งผ่านระบบสื่อสัญญาณมีจำนวนน้อยลง ช่วยให้การส่งเร็วขึ้น การเข้ารหัสมี ให้เลือก 2 แบบคือ 1 มิติ, 2 มิติ แต่จะแสดงเฉพาะรายละเอียดของการเข้ารหัสแบบ 1 มิติซึ่งใช้หลักการเข้ารหัสแบบโมดิฟายด์ฮัฟฟ์แมน ซึ่งข้อมูลที่มีค่าความน่าจะเป็นสูง (เกิดขึ้นบ่อย) จะถูกแทนด้วยรหัสที่มีความยาวน้อย ส่วนข้อมูลที่มีความน่าจะเป็นต่ำ (เกิดขึ้นน้อย) จะถูกแทนด้วยรหัสที่มีความยาวมาก ซึ่งในมาตรฐานของ CCITT ได้กำหนดรหัสที่ใช้ในการแทนมาอย่างแน่นอนแล้วดังในตารางที่ 2.3

ข้อกำหนดในการเข้ารหัสดังต่อไปนี้

1. การเข้ารหัสจะเริ่มขึ้นด้วยจุดขาวก่อนเสมอ กรณีที่จุดแรกเป็นจุดดำ รหัสของจุดขาวที่มีความยาวเท่ากับ 0 จะถูกส่งออกไป

2. ข้อมูลจุดขาวหรือจุดดำสามารถมีความยาวได้ถึง 1728 พิคเซล ซึ่งเป็นความยาวสูงสุดสำหรับเส้นแกนมาตรฐาน 1 เส้น

3. รหัสที่ใช้แทนข้อมูลมี 2 ชนิดคือ

- เทอร์มิเนติ้งโค้ด (Terminationg Code) ใช้แทนค่าพิคเซลที่มีความยาวตั้งแต่ 0 ถึง 63
- เมคอัพโค้ด (Make-up Code) ใช้แทนค่าพิคเซลที่มีความยาวเป็นจำนวนเท่าของ 64 จนถึง 1728 นั่นคือมีเมคอัพโค้ดแทนรหัสได้ 27 ชุด

4. กรณีของพิคเซลซึ่งมีความยาวตั้งแต่ 64-1728 จะแทนด้วยเมคอัพโค้ดซึ่งมีค่าเท่ากับหรือน้อยกว่าความยาวของพิคเซลก่อน โดยความยาวของพิคเซลที่เหลือจากการแทนด้วยเมคอัพโค้ด จะมีค่าไม่เกิน 63 พิคเซล ซึ่งสามารถแทนค่าที่เหลือด้วยเทอร์มิเนติ้งโค้ดได้

5. จุดสิ้นสุดของเส้นสแกน (EOL:End of line) เมื่อสิ้นสุดการสแกนข้อมูลครบ 1 เส้น ข้อมูลจะตั้งตามด้วยรหัส EOL เพื่อแสดงจุดสิ้นสุดของเส้นสแกน นอกจากนี้รหัส EOL ยังใช้นำหน้าเส้นสแกนเส้นแรกของแต่ละหน้าด้วย รูปแบบของ EOL คือ 0000 0000 0001

6. บิทเติม(Fill)เป็นบิทที่เติมขึ้นระหว่างบิทของข้อมูลกับบิทสิ้นสุด เพื่อให้เวลาในการส่งข้อมูลในเส้นสแกนนั้นไม่ต่ำกว่าเวลาต่ำสุดที่เครื่องพิมพ์ด้านรับสามารถทำงานได้ทันรูปแบบของบิทเติมคือ 0 ซึ่งจำนวนบิทสามารถแปรค่าได้

7. เมื่อจบการส่งข้อมูลแต่ละแผ่นจะต้องตามด้วยรหัสรีเทิร์นทูคอนโทรล (RTC:Return To Control) เพื่อเป็นการกลับเข้าสู่โหมดการควบคุมเครื่องโทรสาร รูปแบบของรหัส RTC คือรหัส EOL ติดต่อกันจำนวน 6 ชุด

Run	White	Black	Run	White	Black
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	10000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101101
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000011000	50	01010101	000001010010
19	0001100	00001100111	51	01010011	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

ตารางที่ 2.3 เทอร์มิเนติ้งโค้ด

2.4 ระบบเครือข่าย และการโปรแกรม

การพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายจำเป็นต้องมีความรู้พื้นฐานทางด้านระบบเครือข่ายพอสมควรในหัวข้อจะอธิบายคำศัพท์พื้นฐาน และหลักการของระบบเครือข่ายที่จำเป็นสำหรับการทำความเข้าใจในการพัฒนาโปรแกรมประยุกต์บนเครือข่ายดังจะกล่าวต่อไป

2.4.1. ความหมายของระบบเครือข่ายคอมพิวเตอร์และอินเทอร์เน็ตเวิร์คกิ้ง (Internet working)

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) คือระบบการเชื่อมต่อระหว่างระบบปลายทาง (End-System) ซึ่งระบบปลายทางเป็นระบบอิสระจากกัน(Autonomous)ระบบปลายทางสามารถเป็นได้ตั้งแต่ไมโครคอมพิวเตอร์(Microcomputer) ไปจนกระทั่งซุปเปอร์คอมพิวเตอร์ (Supercomputer) ขนาดใหญ่เพื่อจุดมุ่งหมายในการแลกเปลี่ยนข้อมูล และการแบ่งข้อมูลในการแบ่งปันทรัพยากรของระบบเช่น ไฟล์(File)ข้อมูล ,เครื่องพิมพ์(Printer) ,โมเด็ม(Modem), ตลอดจนการให้บริการฐานข้อมูลร่วม(Sharing database)

อินเทอร์เน็ตเวิร์คกิ้งหรืออินเทอร์เน็ต(Internet) คือการเชื่อมต่อของระบบเครือข่าย 2 เครือข่ายขึ้นไปดังนั้นคอมพิวเตอร์บนระบบเครือข่ายหนึ่งก็สามารถติดต่อกับคอมพิวเตอร์บนระบบเครือข่ายอื่นๆได้

2.4.2 องค์ประกอบของอินเทอร์เน็ต

อินเทอร์เน็ตเป็นเครือข่ายคอมพิวเตอร์ชนิดหนึ่งที่ใช้โปรโตคอล TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นมาตรฐานการทำงานของระบบ ดังนั้นถ้ามีเครือข่ายคอมพิวเตอร์ที่ใช้โปรโตคอล TCP/IP อยู่แล้วก็จะเป็นการสะดวกและง่ายต่อการเชื่อมต่อเข้ากับระบบอินเตอร์เน็ต ระบบการทำงานของเครือข่ายโปรโตคอล TCP/IP โดยเฉพาะสำหรับเครือข่ายอินเทอร์เน็ตนั้นจะแบ่งกลุ่มของแฟกเตอร์หรือฟังก์ชั่นการทำงานออกเป็น 6 กลุ่มใหญ่ๆ ซึ่งการติดตั้งอุปกรณ์ต่างๆ เองก็ต้องคำนึงถึงแฟกเตอร์ 6 กลุ่มเช่นเดียวกันคือ

1. <u>ชนิดของสถานี</u>

ระบบเครือข่ายโปรโตคอล TCP/IP ส่วนใหญ่จะประกอบด้วยเครื่องคอมพิวเตอร์ (สถานี) ที่ทำหน้าที่แตกต่างกันอยู่ 2 ชนิด คือเครื่องที่เป็นสถานีที่ให้บริการที่เรียกว่าโฮสต์(Host)หรือเซิรฟ์เวอร์(Server)และเครื่องคอมพิวเตอร์สำหรับผู้ใช้ทั่วไปซึ่งเรียกว่าเทอร์มินัล(Terminal)หรือ ไคลเอ็นต์(Client) โดยที่เครื่องคอมพิวเตอร์ที่เป็นสถานีให้บริการนั้นจะเป็นเครื่องที่คอยให้บริการแก่ผู้ใช้ในด้านต่างๆไม่ว่าเป็นแหล่งเก็บรวบรวมข้อมูล (Data Sharing) การให้บริการโปรแกรมประยุกต์ต่างๆ (Application) หรือการให้บริการการใช้งานระบบประมวลผลกลาง(CPU Time Sharing) เป็นต้น ดังนั้นคุณสมบัติโดยทั่วไปทั้งด้านฮาร์ดแวร์และด้านซอฟต์แวร์ของเครื่องโฮสต์หรือเซิรฟ์เวอร์จึงมีคุณสมบัติที่ดีกว่าเครื่องคอมพิวเตอร์สำหรับผู้ใช้มาก

2. <u>ระบบไอพีแอดเดรส (IP Address)</u>

การสื่อสารข้อมูลในระบบเครือข่ายคอมพิวเตอร์ เป็นการสื่อสารในลักษณะที่เฟรมข้อมูลของแต่ละการสื่อสารเป็นคนกำหนดเส้นทางที่สื่อสารเองคือเมื่อมีการขอติดต่อสื่อสารข้อมูลของเครื่องคอมพิว-เตอร์คู่ใดเกิดขึ้น เครื่องคอมพิวเตอร์เครื่องนั้นก็จะทำการสร้างเฟรมข้อมูลขึ้นมาแล้วค่อยส่งออกไปในระบบเครือข่ายโดยที่เฟรมข้อมูลนั้นจะมีส่วนของแอดเดรสที่อยู่ในส่วนอ้างอิงกำกับการสื่อสาร(Header) ที่จะบอกว่า เฟรมข้อมูลนี้เป็นของเครื่องคอมพิวเตอร์เครื่องใดที่กำลังส่ง และจะส่งไปยังเครื่องใด

ดังนั้นการที่เครื่องคอมพิวเตอร์ต่างๆ จะติดต่อสื่อสารกันในระบบเครือข่ายโปรโตคอล TCP/IP จะต้องมีการกำหนดค่าไอพีแอดเดรสให้แต่ละสถานีที่จะสื่อสารกันด้วย (นอกเหนือจากค่า MAC-Address ที่มีอยู่ในแต่ละเครื่อง) เพราะค่าไอพีแอดเดรสนั้นจะเป็นค่าอ้างอิงในเฟรมข้อมูลที่สื่อสารในเครือข่าย ซึ่งจะมี 2 ชนิด คือไอพีแอดเดรสต้นทาง (Source IP Address) และไอพีแอดเดรสปลายทาง (Destination IP Address)

3. ระบบโปรโตคอลหาเส้นทาง (IP Routing Protocols)

ลักษณะการติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์ใดๆ ในระบบเครือข่ายอินเตอร์เน็ตนั้น โดยทั่วไปแล้วมี 2 ลักษณะคือ

- การเชื่อมต่อภายในเครือข่ายท้องถิ่น (LAN)

- การเชื่อมต่อระหว่างเครือข่ายท้องถิ่นหนึ่งกับอีกเครือข่ายท้องถิ่นหนึ่ง โดยอาจมีการบริการของระบบเครือข่ายระยะไกล (WAN) เข้ามาเกี่ยวข้องด้วย

ในการเชื่อมต่อจำเป็นต้องใช้อุปกรณ์ที่เรียกว่าเราท์เตอร์ (Router) โดยเราท์เตอร์จะเกี่ยวข้องกับระบบทำงานที่เรียกว่า โปรโตคอลหาเส้นทาง (Routing Protocol) ซึ่งจำทำหน้าที่ตรวจสอบและจัดการเกี่ยวกับเส้นทางในการสื่อสารข้อมูลทั้งหมดของระบบ

4. ระบบชื่อกลุ่ม (Domain Name System)

มีการออกแบบระบบชื่อของสถานีบริการต่างๆบนเครือข่ายอินเทอร์เน็ตในรูปลักษณะตัวอักษร เพื่ออำนวยความสะดวกต่อการใช้งานของยูสเซอร์ ระบบ DNS(Domain Name System) เป็นระบบซอฟแวร์ที่ทำหน้าที่ในการจัดสรรและบริการในส่วนการเปรียบเทียบค่าระหว่างชื่อตัวอักษรกับค่าไอพีแอดเดรสของเครื่องสถานีต่างๆบนอินเทอร์เน็ต

5. โปรแกรมประยุกต์บนเครือข่ายอินเทอร์เน็ต (Application)

ระบบเครือข่ายอินเทอร์เน็ตเป็นระบบเครือข่ายขนาดใหญ่ที่มีการเชื่อมโยงกันทั่วโลก ดังนั้นการใช้งานโปรแกรมประยุกต์ต่างๆ บนระบบอินเทอร์เน็ตจึงมีลักษณะพิเศษแตกต่างจากการใช้งานบนระบบเครือข่ายท้องถิ่นทั่วไป คือจะมีโปรแกรมประยุกต์มากมายหลายชนิด เช่น ระบบจดหมายอิเล็กโทรนิกส์ (E-Mail) ระบบข่าวสารร่วม (Usenet) ระบบอุโมงค์อินเทอร์เน็ต(Gopher) และระบบเครือข่ายใยแมงมุม (World-Wide-Web) เป็นต้น โดยที่แต่ละชนิดมีลักษณะการใช้งานที่แตกต่างกันมาก

6. ระบบความปลอดภัย (Security)

มีหน้าที่ป้องกันไม่ให้มีการลักลอบเข้ามาใช้หรือทำลายข้อมูลที่สำคัญ

2.5 ความรู้เบื้องต้นเกี่ยวกับโปรโตคอล TCP/IP

2.5.1 ข้อแตกต่างระหว่างชุดโปรโตคอล TCP/IP และรูปแบบ OSI

1. ลำดับการติดต่อสื่อสารของชั้นเลเยอร์ ในรูปแบบ OSI นั้นจะกำหนดลำดับขั้นการสื่อสารที่เป็นลำดับขั้นตอนการติดต่อที่แน่นอน โดยเฉพาะการอินเตอร์เฟซระหว่างชั้นเลเยอร์ ซึ่งทำให้รูปแบบ OSI สามารถเป็นระบบเปิดสำหรับระบบเครือข่ายคอมพิวเตอร์ทั่วไป เพราะไม่ว่าจะมีการเปลี่ยนแปลงโปรโตคอลในเลเยอร์ชั้นใดก็ตามจะไม่มีผลกระทบต่อการสื่อสารเลเยอร์ชั้นถัดไป ในขณะที่ชุดโปรโตคอล TCP/IP จะไม่มีการกำหนดรูปแบบการติดต่อที่ตายตัว เพื่อในผู้ออกแบบเครือข่ายมีอิสระสามารถเปลี่ยนแปลงโครงสร้างของเครือข่ายได้ง่ายขึ้น

2. การติดต่อสื่อสารระหว่างเครือข่ายหรืออินเทอร์เน็ต คือ การติดต่อสื่อสารข้อมูลระหว่างระบบคอมพิวเตอร์ 2 ระบบที่ไม่สามารถติดต่อสื่อสารกันได้โดยผ่านทางเครือข่ายการสื่อสารข้อมูลเพียงเครือข่ายเดียวได้ต้องอาศัยเครือข่ายขึ้นไปในการติดต่อสื่อสารกัน และเครือข่ายเหล่านี้อาจจะมีลักษณะของเครือข่ายที่ต่างกันได้

ความแตกต่างในเรื่องของอินเทอร์เน็ตระหว่างชุดโปรโตคอล TCP/IP กับรูปแบบ OSI ก็คือในชุดโปรโตคอล TCP/IP จะใช้ โปรโตคอลสำหรับอินเทอร์เน็ตที่เรียกว่า โปรโตคอล IP (Internet Protocol) ซึ่งในรูปแบบ OSI จะเรียกว่าโปรโตคอลสำหรับอินเทอร์เน็ตว่า โปรโตคอลเน็ตเวิร์ค

3. การบริการการเชื่อมต่อการสื่อสาร (Connection Service) ในชุดโปรโตคอล TCP/IP นั้นจะมีการบริการการเชื่อมต่อการสื่อสารระหว่างต้นทางและปลายทาง 2 แบบ คือการบริการแบบ Connection-Oriented เท่านั้น

4. โปรโตคอลควบคุมการจัดการสื่อสาร ในชุดโปรโตคอล TCP/IP จะใช้โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลสำหรับควบคุมการสื่อสาร กำหนดตำแหน่งต้นทางและปลายทาง และ อื่นๆ กับข้อมูล ซึ่งในรูปแบบ OSI นั้นจะแบบแยกการควบคุมการสื่อสารออกจากกันโดยใช้โปรโตคอลเซสชั่นและโปรโตคอล ทรานสปอร์ตตามลำดับ

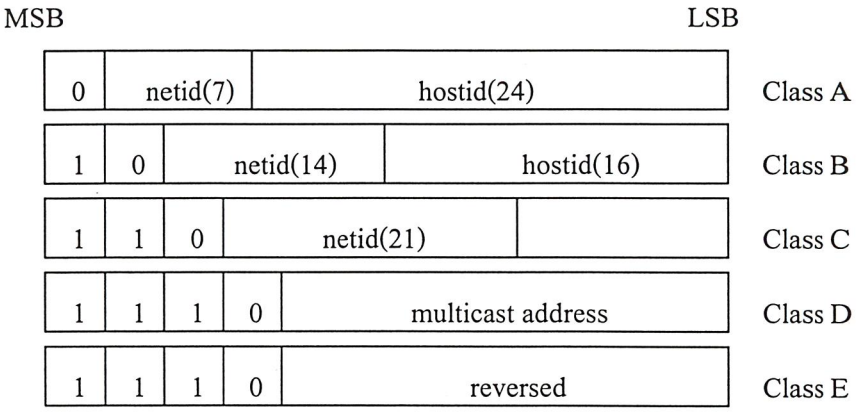
2.5.2 ลักษณะของการติดต่อ แบ่งออกเป็น 2 ชนิดคือ

1. Connection-Oriented คือการติดต่อที่ต้องมีการเชื่อมโปรเซสที่จะทำการติดต่อก่อนที่จะมีการส่งหรือรีบข้อมูล ซึ่งสามารถใช้คำว่าวงจรเสมือน (Virtual Circuit) เพราะว่าจะทำงานเสมือนมีวงจรต่ออยู่ระหว่างโปรเซสถึงแม้ว่าข้อมูลนี้อาจจะผ่าน Packet –Switching Network บริการชนิดนี้ส่วนมากจะใช้ในกรณีที่มีข่าวสารต้องการมากกว่าหนึ่งข่าวสาร ดังนั้นสามารถแบ่งขั้นการทำงานออกเป็น

- ขั้นการสร้างการติดต่อ (Connection establishment)
- ขั้นการส่งผ่านข้อมูล (Data Transfer)
- ขั้นยกเลิกการติดต่อ (Connection termination)

2. Connectionless หรือดาต้าแกรม (Datagram) คือจะไม่มีขั้นการสร้างการติดต่อและขั้นการยกเลิกการติดต่อ แต่จะมีขั้นการส่งผ่านข้อมูลเพียงอย่างเดียว โดยข้อมูลซึ่งเรียกว่าดาต้าแกรม จะถูกส่งจากระบบหนึ่งไปสู่ระบบหนึ่งอย่างเป็นอิสระโดยไม่ขึ้นอยู่กับดาต้าแกรมอื่น

2.5.3 ไอพีแอดเดรส (IP Address) เป็นหัวใจสำคัญของโปรโตคอล IP เพราะเป็นแอดเดรสที่บ่งบอกถึงสถานีปลายทางจริงๆ ภายในระบบเครือข่ายขนาดใหญ่ที่มีการเชื่อต่อทั้ง LAN และ WAN ทำให้ผู้ใช้งานมองระบบเครือข่ายเสมือนเป็นระบบเครือข่ายเดียวกันได้ และมีวัตถุประสงค์ให้ผู้ใช้งานสามารถกำหนดแอดเดรสของสถานีได้ตามต้องการ



รูปที่ 2.5 แสดงการแบ่งประเภทของไอพีแอดเดรส

ไอพีแอดเดรสเป็นแอดเดสขนาด 32 บิต แบ่งออกเป็น 5 ประเภทคือประเภท A,B,C,D และ E ดังแสดงอยู่ในรูปที่ แอดเดรสที่ใช้งานโดยทั่วไปคือ 3 ประเภทแรก ประเภทที่ 4 ใช้ในกรณีพิเศษ ส่วนประเภทสุดท้ายสำรองไว้ใช้ในอนาคต ช่วงของไอพีแอดเดรสแต่ละประเภทแสดงดังรูปที่ 2.6

0000000……..……00000 0111111……..……11111	Class A
1000000……..……00000 1011111……..……11111	Class B
1100000……..……00000 1101111……..……11111	Class C
1110000……..……00000 1110111……..……11111	Class D
1111000……..……00000 1111011……..……11111	Class E

รูปที่ 2.6 ช่องของไอพีแอดเดรสแต่ละประเภท

ไอพีแอดเดรสแต่ละประเภทมีหมายเลขที่ไม่ซ้ำกัน ซึ่งก็หมายความว่า ถ้ากำหนดสถานีใดๆ ด้วยไอพีแอดเดรส หมายเลขของสถานีก็ไม่ซ้ำด้วยเช่นกัน เมื่อมองแค่ 3 ประเภทแรกจะเห็นได้ว่าภายในแอดเดรสขนาด 32 บิต แบ่งออกเป็น 2 ส่วนย่อย คือหมายเลขเครือข่าย (NetID : Network identification) และหมายเลขสถานี(HostID : Host identification) แต่ละประเภทมีขนาด NetID และ HostID ไม่เท่ากันดังในรูปที่ 2.5

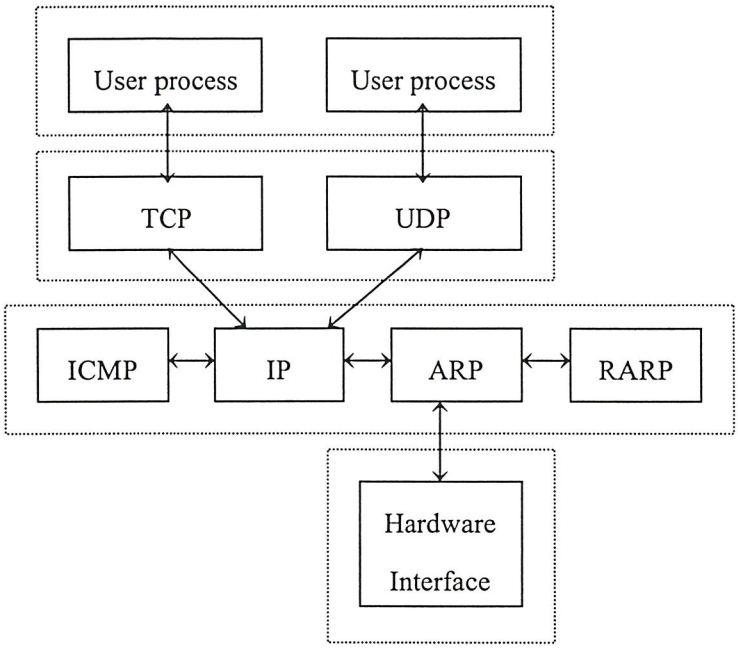
ในการอ่าน/เขียนไอพีแอดเดรส แทนที่จะมองเป็นเลขฐานสอง กลับมองเป็นเลขฐานสิบ โดยการแบ่ง 32 บิต ออกเป็น 4 ไบต์ย่อย แต่ละไบต์แทนด้วยเลขฐานสิบ 1 ตัว เรียกวิธีการเช่นนี้ว่า "dotted decimal"ดังนั้นเลขแต่ละตัวของไอพีแอดเดรสมีค่า 0-255 เช่น $(10011110011011000000010011100010)_B$ เขียนในรูป "dotted decimal" ได้เป็น 158.108.4.226 เมื่อกล่าวถึงเฉพาะ NetID ของไอพีแอดเดรสประเภท A,B และ C จะใช้เลขฐานสิบจำนวน 1,2 และ 3 หลักในการอ้างถึงตามลำดับ และปล่อยให้ส่วนที่เป็น hostid มีค่าเป็น "0" เช่น1.0.0.0,158.108.0.0 และ 195.200.10.0 เป็น NetID ประเภท A,B และ C ตามลำดับ

การนำไอพีแอดเดรสมากำหนดเป็นหมายเลขของเครือข่ายหรือสถานีนั้นมีข้อกำหนดปลีกย่อยอยู่หลายประการ ซึ่งผู้วางระบบจำเป็นต้องเข้าใจในส่วนนี้ จึงจะกำหนดแอดเดรสในการใช้งานได้ไม่ผิดพลาด

2.5.4 โครงสร้างของชุดโปรโตคอล TCP/IP

โครงสร้างของสถาปัตยกรรมของชุดโปรโตคอล TCP/IP นั้นแบ่งออกเป็น 3 ส่วนหลักๆ คือส่วนของกรรมวิธีปฏิบัติการหรือโปรเซส (Process) โฮสต์(Host) และเครือข่าย (Network) ในส่วนของโปรเซสก็ได้แก่เอนทิลิตี้หรือแอปพลิเคชั่นที่ต้องการติดต่อสื่อสาร ทุกโปรเซสจะกระทำในเครื่องของโฮสต์ (หรือเวอร์กสเตชั่น) ซึ่งในแต่ละโฮสต์สามารถจะมีหลายๆ เอนทิตี้ (หมายถึงแอพพลิเคชั่น เช่นโปรแกรมระบบจัดการฐานข้อมูล ไฟล์ข้อมูล) ได้พร้อมกัน การสื่อสารกันระหว่างเอนทิตี้ของโฮสต์เครื่องหนึ่งกับเอนทิตี้ของโฮสต์อีกเครื่องหนึ่งหรือหลายเครื่องจะกระทำโดยผ่านทางเครือข่ายที่โฮสต์เชื่อมต่ออยู่

การทำงานที่สัมพันธ์กันระหว่างโปรเซส โฮสต์ และเครือข่ายของสถาปัตยกรรม TCP/IP ทำให้สามารถจัดรูปแบบของสถาปัตยกรรม TCP/IP ได้เป็น 4 ชั้น และสามารถกำหนดชนิดของโปรโตคอลที่ทำงานในแต่ละชั้นได้เป็น 4 แบบโปรโตคอลเช่นกัน ดังที่ได้กล่าวมาแล้วว่าในชุดโปรโตคอล TCP/IP นั้น เอนทิตี้แต่ละชั้นอาจจะติดต่อสื่อสารข้อมูลโดยผ่านเอนทิตี้ในชั้นเดียวกันหรือเอนทิตี้ในชั้นล่างลงไปซึ่งไม่จำเป็นต้องเป็นชั้นที่ติดกัน



รูปที่ 2.7 ความสัมพันธ์ระหว่างชุดโปรโตคอล TCP/IP

1. ชั้นแอพพลิเคชั่น ในชั้นนี้ประกอบด้วยโปรแกรมประยุกต์ที่ใช้ในเครือข่าย เช่นโปรแกรมส่งถ่ายข้อมูล (File-transfer program) และอาจกล่าวได้ว่าโปรโตคอล TCP/IP ก็คือโปรโตคอลในชั้นแอพพลิเคชั่นรวมกับชั้น พรีเซนเตชั่นของ OSI โมเดลนั่นเอง

2. ชั้นทรานสปอร์ต(Transport Layer) ในชั้นนี้เป็นชั้นที่ให้การส่งข้อมูลจากปลายถึงจุดปลาย เปรียบเทียบได้กับชั้นเซสชั่นรวมกับทรานสปอร์ตเลเยอร์นั่นเอง โดยโปรโตคอล TCP/IP มีซอกเก็ต (Socket) เป็นจุดปลาย (End-Point) ในการสื่อสาร ซึ่งซอกเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์และหมายเลขพอร์ต (Port) ของเครื่องที่ต้องการส่งข้อมูลไปถึง ในชั้นนี้มีการรับรองให้ถึงที่หมาย และลำดับของข้อมูลที่ส่งโดยไม่ซ้ำและความผิดพลาดของข้อมูล

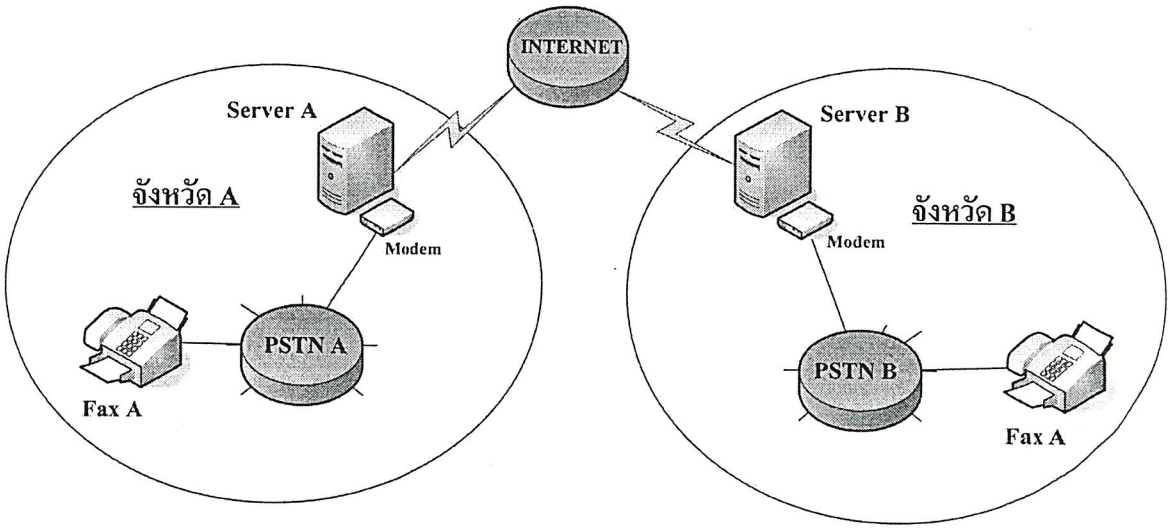
3. ชั้นอินเทอร์เน็ต(Internet layer) ในชั้นนี้มีการกำหมดดาต้าแกรมและทำการหาเส้นทางการส่ง หน้าที่ของ เลเยอร์นี้เทียบเท่ากับชั้นเน็ตเวอร์คเลเยอร์และดาต้าลิ้งของ OSI โมเดล

4. ชั้นฟิสิคัล โปรโตคอล TCP/IP ไม่ได้กำหนดรูปแบบของการเชื่อมต่อในระดับนี้ขึ้นใหม่ แต่ใช้มาตรฐานที่มีอยู่เดิมที่กำหนดไว้ก่อน เช่น RS232, อีเทอร์เน็ต (Ethernet) เป็นต้น

บทที่ 3
การออกแบบและการสร้าง

โครงงานนี้พัฒนาขึ้นโดย โปรแกรมภาษาบอร์แลนเดลฟายเอ็นเตอร์ไพร เวอร์ชัน 7.0 (Borland Delphi Enterprise Version 7.0) โดยใช้ระบบปฏิบัติการไมโครซอฟต์ วินโดวส์ เอ็มอี (Microsoft Windows ME)

โครงงานนี้จะเชื่อมต่อระหว่าง 2 โครงข่าย คือ เครือข่ายอินเทอร์เน็ต และเครือข่ายโทรศัพท์ (PSTN) ซึ่งผ่านคอมพิวเตอร์ที่มีโปรแกรมทำหน้าที่รับส่งโทรสาร ระบบตอบรับด้วยเสียง (Interactive Voice Response) ไว้โต้ตอบกับผู้ใช้ (User) ในการใช้บริการระบบฐานข้อมูล



PSTN: Public Switching Telephone Network
(ระบบโทรศัพท์พื้นฐาน)

รูปที่ 3.1 ภาพรวมของลักษณะการสื่อสารทั้งระบบ

3.1 การทำงานของระบบตอบรับด้วยเสียง (IVR: Interactive Voice Response)

โดยจะแบ่งเป็น 2 ส่วนคือ ด้านผู้ส่งโทรสาร และ ด้านผู้ไปรับโทรสาร

3.1.1 ด้านผู้ส่งโทรสาร

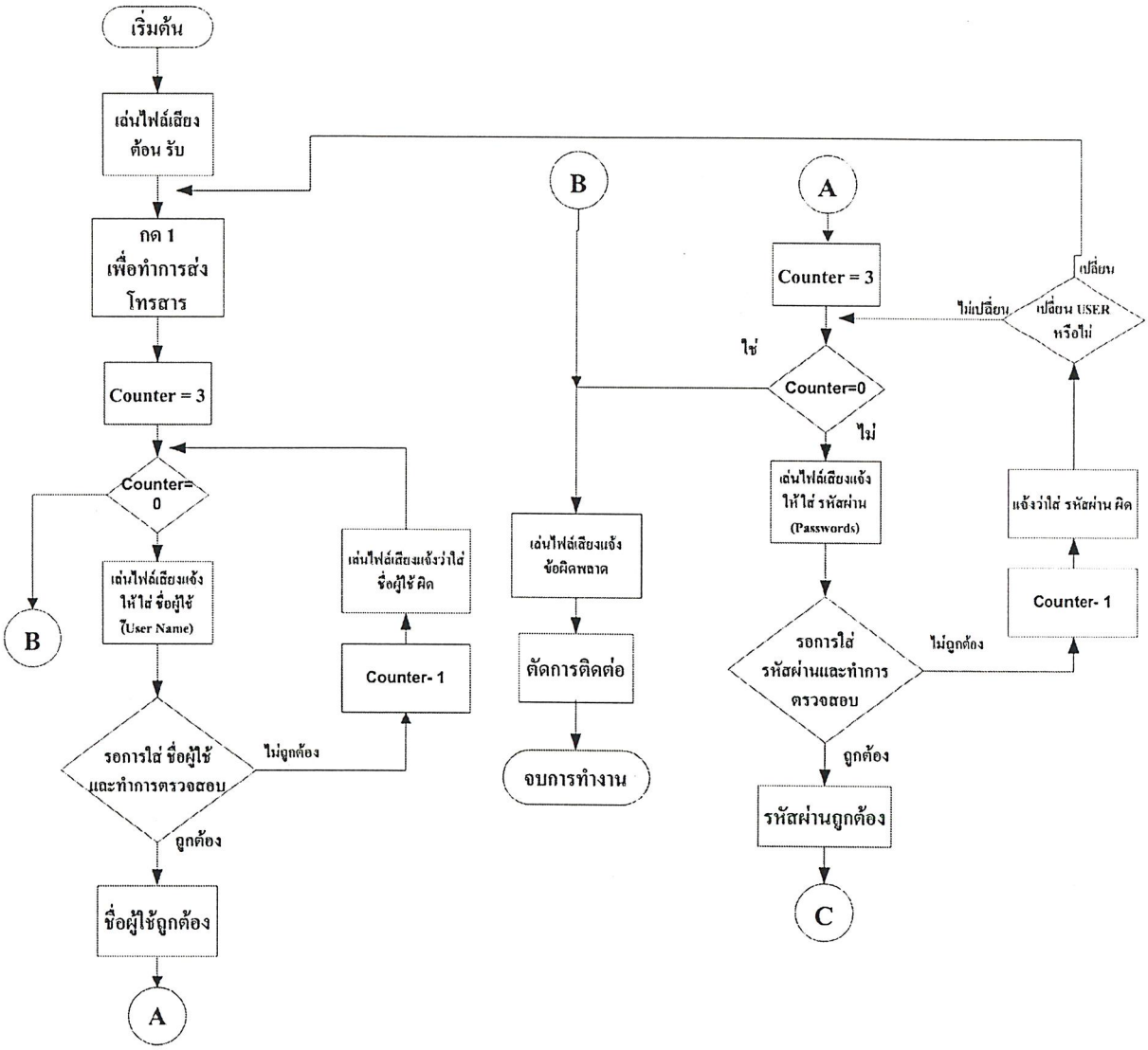
จากรูปที่ 3.2.1 และ 3.2.2 การทำงานจะเป็นแบบ อัตโนมัติเมื่อมีผู้เรียกโดยการเรียกเข้ามาใช้บริการ ต้องใช้เครื่องโทรสารโทรเรียกเข้ามา เข้าทำมาในหมายเลขโทรศัพท์ ก็ จะทำยกหูทันที แล้วเล่นไฟล์เสียงต้อนรับ โดยการเล่นไฟล์เสียงจะเล่นโดยใช้ ไฟล์สกุล .wav

หลังจากทำการเล่นเสียงต้อนรับ เซิร์ฟเวอร์ก็จะทำการเล่นไฟล์เสียง เพื่อบอกผู้ใช้ให้ ใส่รหัสผู้ใช้บริการ โดยผู้ใช้จะใส่รหัสดิจิตอล 4 ตัว โดยกดบนแป้นคีย์บนเครื่องโทรสาร โดย รับค่าการแปลงสัญญาณ DTMF มาเป็นสัญญาณตัวเลขที่โปรแกรม เข้า

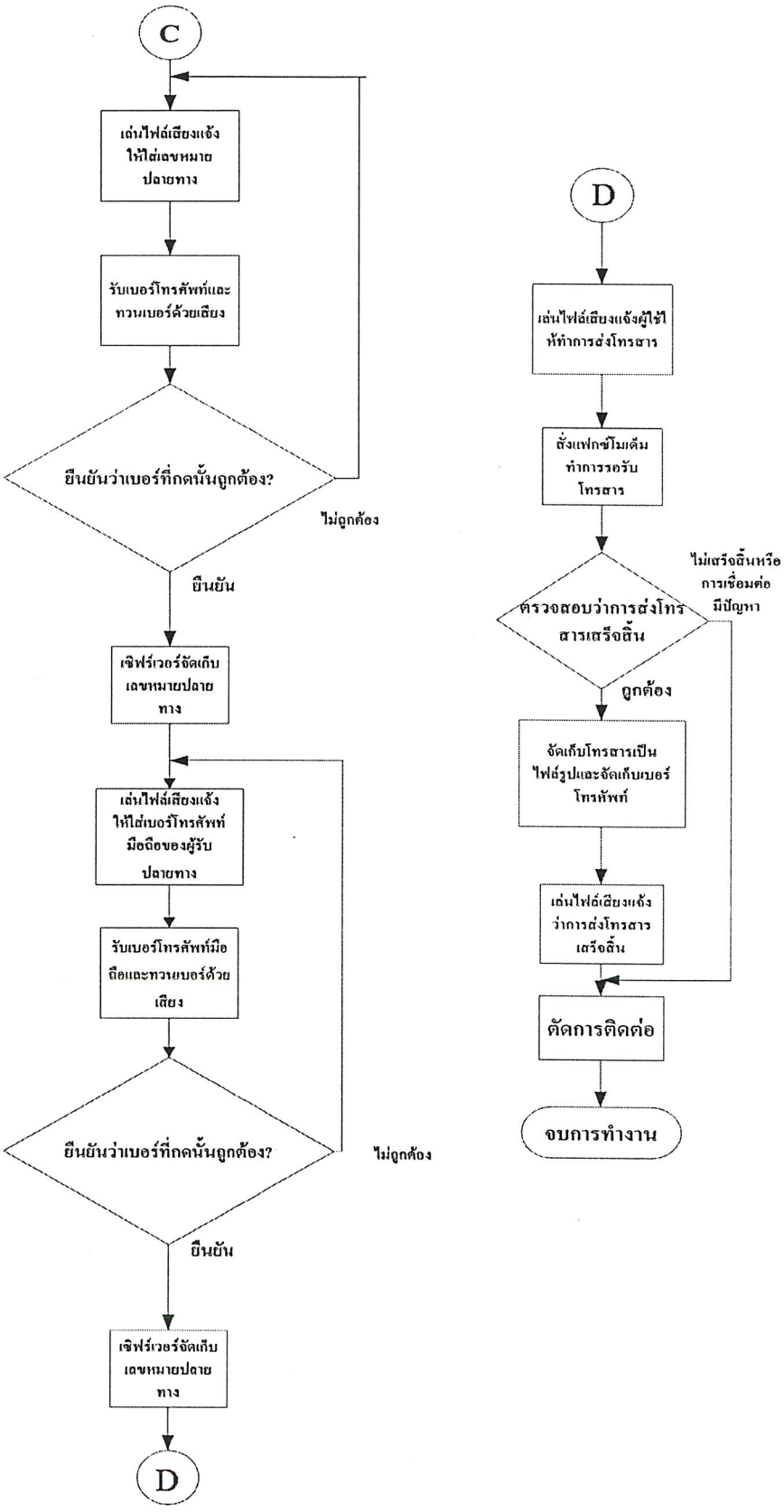
แล้วจะตัวเลขที่ได้ ไปเทียบกับ ชื่อตัวเลขของผู้ใช้บริการในฐานข้อมูลของสมาชิก ถ้าไม่ตรง ก็จะให้โอกาสใส่ผิดได้อีก 3 ครั้งแล้วจะทำการเล่นไฟล์เสียงแจ้งความผิดพลาด และ ตัดการติดต่อ แต่ถ้าตรงกัน ก็จะทำการถามหารหัสผ่านต่อไป โดยจะให้ โอกาสใส่ผิดได้ 3 ครั้ง หรือจะทำการเปลี่ยนผู้ใช้บริการ

หลังจากใส่รหัสผู้ใช้บริการและรหัสผ่านถูกต้อง ระบบก็จะเล่นไฟล์เสียงแจ้งให้ผู้ใช้บริการใส่หมายเลขเครื่องโทรสารปลายทางที่ผู้ใช้ต้องการจะส่ง โดยระบบจะให้ใส่หมายเลข แล้วทวนหมายเลขที่ผู้ใช้กดมาด้วยเสียง แล้วให้ผู้ใช้ยืนยันว่าเบอร์ที่กดนั้นถูกต้อง ถ้าไม่ก็จะให้ใส่เบอร์โทรศัพท์ใหม่

จากนั้นระบบจะทำการแจ้งให้ผู้ใช้บริการส่งโทรสาร โดยผู้ใช้เพียงกด ปุ่มส่งโทรสารบนเครื่องโทรสารเท่านั้น และถ้าเกิดระหว่างการส่งโทรสารผิดพลาดก็จะตัดการติดต่อ หลังจากเสร็จสิ้นการส่งโทรสาร เครื่องจะทำการแปลงโทรสารที่ได้รับเป็นไฟล์รูปภาพ ทำการจัดเก็บเพื่อทำการส่งให้เครื่องเซิร์ฟเวอร์ปลายทาง แล้วตัดการติดต่อ



รูปที่ 3.2.1 โฟลว์ชาร์ตแสดงการทำงานของระบบตอบรับอัตโนมัติด้านผู้ส่ง

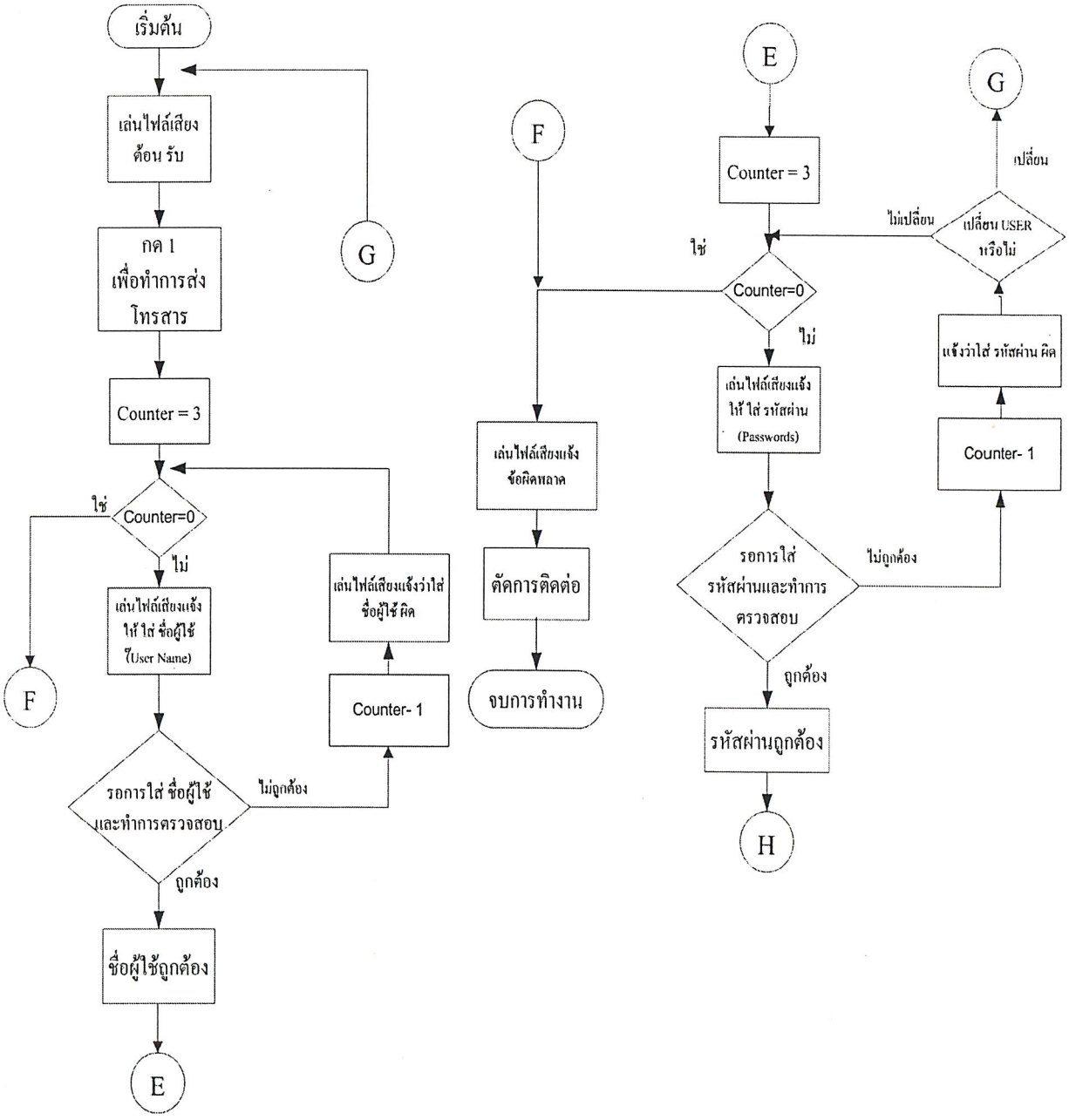


รูปที่ 3.2.2 โฟลว์ชาร์ตแสดงการทำงานของระบบตอบรับอัตโนมัติด้านผู้ส่ง

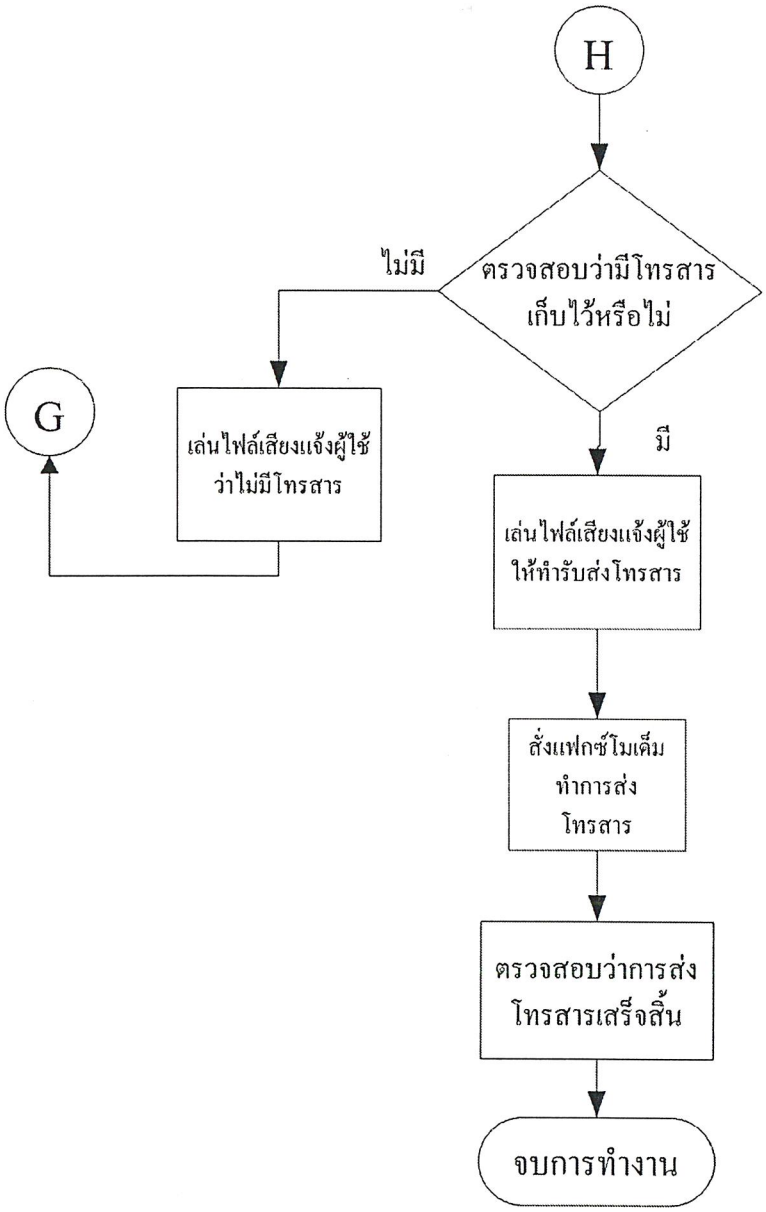
3.1.2 ด้านผู้รับโทรสาร

จากรูปที่ 3.3.1 และ 3.3.2 จะเป็นบริการที่มีให้ผู้รับปลายทางเข้ามารับโทรสารด้วยตนเอง ถ้าเซิร์ฟเวอร์ไม่สามารถส่งโทรสารไปที่เครื่องโทรสารผู้รับได้

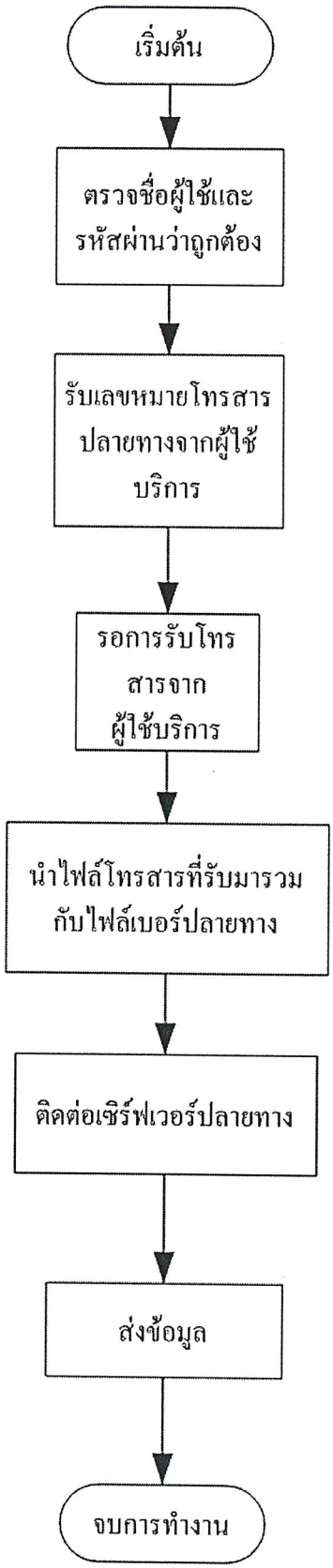
การเริ่มต้นจะเหมือนกันกับด้านผู้ส่ง จะมีการขอรหัสผู้ใช้บริการ กับรหัสผ่าน โดยรหัสผ่านนี้จะไปเทียบกับรหัสผู้ใช้ใน ล็อกเกอร์โทรสาร เมื่อทั้งรหัสผู้ใช้และรหัสผ่านถูกต้อง ระบบก็จะเล่นไฟล์เสียงแจ้งให้ผู้รับเตรียมรับโทรสาร ระบบจะทำการส่งโทรสาร



รูปที่ 3.3.1 โฟลว์ชาร์ตแสดงการทำงานของระบบตอบรับด้วยเสียงด้านผู้รับ



รูปที่ 3.3.2 โฟลว์ชาร์ตแสดงการทำงานของระบบตอบรับด้วยเสียงด้านผู้รับ



รูปที่ 3.4 โฟลว์ชาร์ตแสดงการทำงานของส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ต้นทาง

3.2 ส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ต้นทาง

ในส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ต้นทางจะทำหน้าที่เป็นตัวกลางในการติดต่อสื่อสารระหว่างผู้ส่งโทรสารกับเซิร์ฟเวอร์ปลายทาง ซึ่งกระบวนการนี้จะเป็นไปอย่างอัตโนมัติ ดังรูปที่ 3.4 จะเห็นว่าในเบื้องต้นนั้นโปรแกรมสามารถที่จะทำการตรวจสอบรหัสผู้ใช้งานกับรหัสผ่านเทียบกับฐานข้อมูลที่ผู้ใช้งานเคยลงทะเบียนมาแล้วก่อนหน้านี้ จากนั้นทำการรับข้อมูลเบื้องต้นในการส่งโทรสาร เช่น เลขหมายโทรสารปลายทางที่ต้องการส่ง และ รับข้อมูลโทรสารของผู้ใช้บริการ

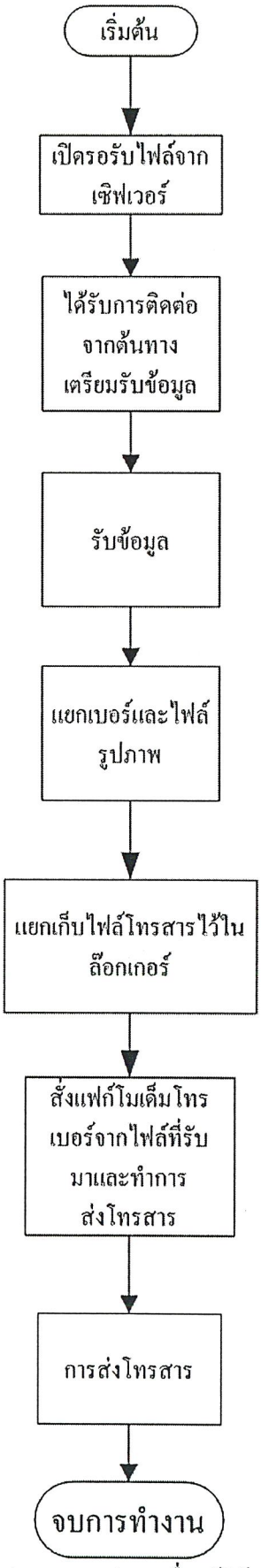
เมื่อได้ข้อมูลครบแล้วทางเครื่องเซิร์ฟเวอร์ต้นทางจะทำการรวมข้อมูลทั้งหมด เพื่อทำการส่งต่อไปยังเครื่องเซิร์ฟเวอร์ปลายทางผ่านทางเครือข่ายอินเตอร์เน็ต

3.3 ส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ปลายทาง

ในส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ปลายทาง รับข้อมูลทางเครือข่ายอินเตอร์เน็ต เมื่อข้อมูลได้รับมาเรียบร้อยแล้วทางเซิร์ฟเวอร์ต้นทางจะทำการแยกข้อมูลออกเป็นหมายเลขโทรสารปลายทางและข้อมูลโทรสาร นำไปเก็บไว้ในฐานข้อมูลเพื่อรอคิวในการส่งโทรสารต่อไป

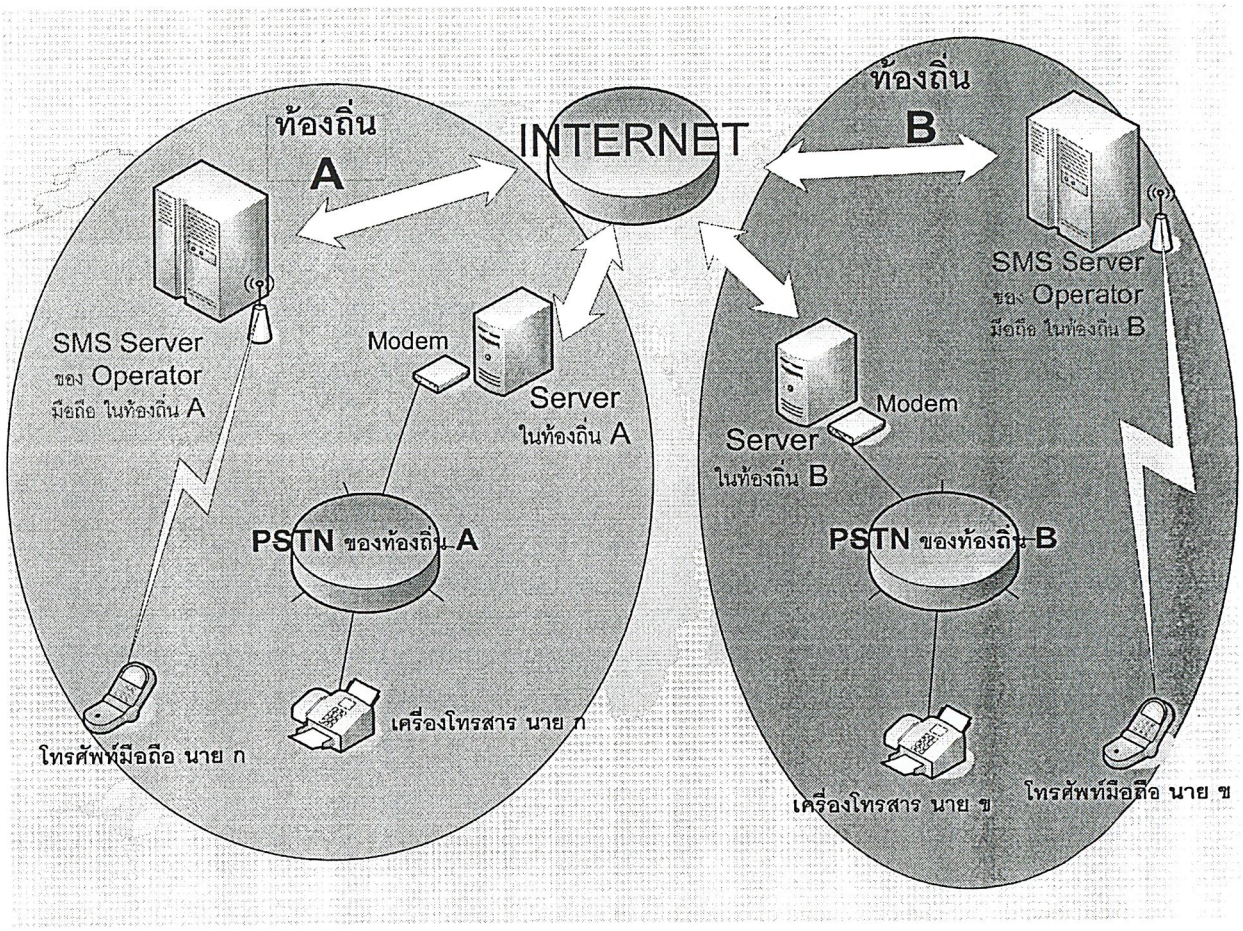
ในขั้นตอนระบบการทำงานเมื่อเครื่องผู้ให้บริการเซิร์ฟเวอร์ปลายทางทำการส่งโทรสาร ดังรูปที่ 3.5 จะเห็นว่าเมื่อทำการส่งโทรสารสำเร็จ ทางเซิร์ฟเวอร์ มาดึงข้อมูลโทรสารในเซิร์ฟเวอร์โดยติดต่อผ่านทางระบบตอบรับด้วยเสียง ของเครื่องผู้ให้บริการ

จากโฟลว์ชาร์ตทั้งหมดที่กล่าวมาแล้วสรุปเป็นภาพรวมของระบบได้ ดังภาพที่ 3.6



รูปที่ 3.5โฟลว์ชาร์ตแสดงการทำงานของส่วนเครื่องผู้ให้บริการเซิร์ฟเวอร์ปลายทาง

จากโฟลว์ชาร์ตทั้งหมดที่กล่าวมาแล้วสรุปเป็นภาพรวมของระบบได้ ดังภาพที่ 3.6



รูปที่ 3.6 ภาพรวมทั้งหมดของระบบ

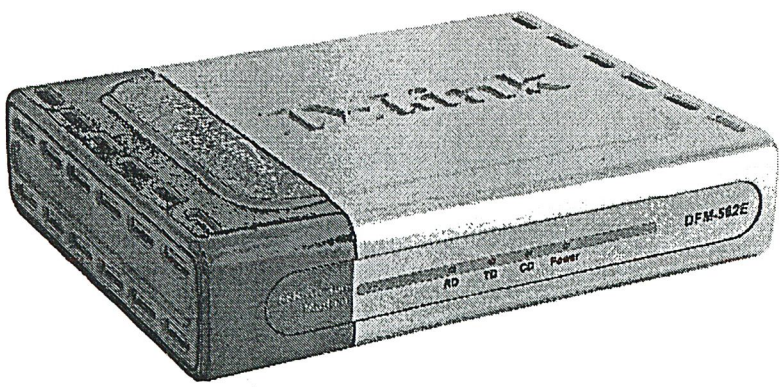
3.4 การทำงานของโปรแกรมรวม

โดยจะอธิบายการทำงานโดย สมมุติ ให้ นาย ก ที่อยู่กรุงเทพ ประเทศไทย ต้องการจะส่งโทรสารไป ซานฟรานซิสโก ประเทศอเมริกา โดยมาใช้บริการของโครงงานนี้

โดยนาย ก จะต้องทำการโทรติดต่อกับศูนย์บริการ ที่มีสาขาอยู่ที่กรุงเทพ ซึ่งเป็นระบบ เซิรฟ์เวอร์ โดยจะมีระบบ ตอบรับอัตโนมัติ โดยระบบตอบรับก็ จะสอบถามผู้ใช้งาน และ รหัสผ่าน โดย นาย ก ก็ใส่รหัสผู้ใช้งานและ รหัสผ่านโดย ใช้แป้นพิมพ์บนเครื่องโทรสาร หลังจากเซิร์ฟเวอร์ อนุญาตให้เข้าไปใช้ ก็จะสอบถาม ถาม เบอร์เรื่องโทรสารปลายทาง,เบอร์โทรศัพท์มือถือของผู้รับโทรสารปลายทาง เมื่อนาย ก กดเบอร์ เครื่องโทรสารปลายทาง,เบอร์โทรศัพท์มือถือของผู้รับโทรสารปลายทาง เรียบร้อย ระบบตอบรับ ก็จะให้ นาย ก กด ส่งโทรสารเซิร์ฟเวอร์ ก็จะรับ เอกสารที่นาย ก ส่ง เมื่อ การส่งโทรสาร เสร็จสิ้นเซิร์ฟเวอร์ ก็จะนำ เอกสารแปลงเป็นไฟล์ภาพ แล้ว ทำการส่ง ต่อรูปภาพเอกสารนี้ ไปทางอินเตอร์เน็ตไปยังเซิร์ฟเวอร์เมือง ซานฟรานซิสโก ประเทศสหรัฐอเมริกา เมื่อเซิร์ฟเวอร์ที่ซานฟรานซิสโกได้รับเอกสาร ก็จะทำการ โทรไปยังปลายทางที่นาย ก ต้องการ โดยใช้แฟกซ์โมเด็มติดต่อ หากทำการส่งโทรสารสำเร็จทางเซิร์ฟเวอร์ปลายทางจะทำการส่งเอสเอ็มเอสไปยังผู้ส่งโทรสารว่าโทรสารได้ทำการส่งเรียบร้อยแล้ว แต่ถ้าหากไม่สามารถส่งโทรสารไปยังผู้รับโทรสารปลายทางได้ ทางเซิร์ฟเวอร์จำทำการส่งเอสเอ็มเอสไปยังผู้รับโทรสารปลายทางว่า ให้โทรติดต่อมายังเซิร์ฟเวอร์ปลายทางเพื่อรับโทรสารผ่านทางระบบตอบรับด้วยเสียง และส่งเอสเอ็มเอสไปยังผู้ส่งโทรสารว่าส่งโทรสารไม่สำเร็จ

3.5 การสร้างโปรแกรม

โปรแกรมนี้ได้พัฒนาโดยใช้ โปรแกรมภาษาบอร์แลนเดลฟายเอ็นเตอร์ไพร เวอร์ชัน 7.0 (Borland Delphi Enterprise Version 7.0) ซึ่งติดตั้ง TurboPower Async Professional Version 4.06 เป็นคอมโพเนนท์ที่รองรับการเชื่อมต่อผ่านพอร์ทอนุกรม (Serial Port) ,การเชื่อมต่อผ่านเครือข่ายโทรศัพท์ (TAPI:Telephony Application Programming Interface), การรับ-ส่งโทรสาร ซึ่งเราสามารถนำมาประยุกต์ใช้สร้างระบบตอบรับด้วยเสียง (IVR: Interactive Voice Response) โดยที่โมเด็มที่ใช้ต้องมีการเชื่อมต่อผ่านพอร์ทอนุกรมแล้วรองรับ รองรับทั้งระบบโทรสารและระบบเสียง ดังนั้นโครงงานนี้จึงเลือกใช้ ดี-ลิงค์โมเด็มดาต้า/แฟกซ์/วอยส์ DFM-562E



รูปที่ 3.7 โครงงานนี้ใช้ ดี-ลิงค์โมเด็มดาต้า/แฟกซ์/วอยส์ DFM-562E
(D-Link Data/Fax/Voice Modem) เนื่องด้วยเป็นโมเด็มที่รองรับทั้งระบบโทรสารและระบบเสียง

โปรแกรมนี้ประกอบไปด้วยส่วนสำคัญ 4 ส่วนด้วยกันดังนี้

1. ระบบตอบรับด้วยเสียง (IVR: Interactive Voice Response)
2. ส่วนทำการรับ-ส่งโทรสาร
3. ส่วนทำหน้าที่ส่งผ่านข้อมูลผ่านเครือข่ายอินเตอร์เน็ต
4. ส่วนการจัดการฐานข้อมูลและส่วนการส่งเอสเอ็มเอส

3.5.1 โปรแกรมระบบตอบรับด้วยเสียง (IVR: Interactive Voice Response)

การสร้างส่วนของระบบตอบรับด้วยเสียงมี 2 คอมโพเนนท์ที่สำคัญดังนี้

1. TApdComport ทำหน้าที่ในการจัดการการรับส่งข้อมูลผ่านทางพอร์ทอนุกรม โดยทำหน้าที่ในการเชื่อมต่อระหว่างโมเด็ม(Modem)กับเครื่องคอมพิวเตอร์
2. TApdTapiDevice ทำหน้าที่จัดการการสื่อสารระหว่างเครื่องคอมพิวเตอร์กับระบบเครือข่ายโทรศัพท์ ซึ่งในการออกแบบเราได้กำหนดอีเวนต์ (Event) ที่สำคัญของโปรแกรมดังนี้
   - OnTapiConnect ทำหน้าที่ในการเริ่มการเชื่อมการทำงานของ TAPI โดยที่จะเริ่มทำงานเมื่อมีผู้ใช้บริการเข้ามาที่เครื่องคอมพิวเตอร์และทำการรับสายเรียบร้อยแล้ว ซึ่งจำเป็นต้องกำหนดให้โมเด็มนั้นเปิดการทำงานในรูปแบบเสียงก่อน โดยใช้โพรซีเยอร์ EnableVoice กำหนดให้มีค่าเป็น True
   - OnTapiDTMF ทำหน้าที่รับเสียงกดปุ่มโทรศัพท์จากผู้ใช้บริการแปลงเป็นตัวเลขดิจิตอลได้ (DTMF DETECTION) โดยค่าที่ได้จะเป็นตัวอักษร 0-9, *, # ตามการกดปุ่มโทรศัพท์ ซึ่งมีการรับค่าได้เพียงครั้งละ 1 ตัวอักษรเท่านั้น การออกแบบจึงต้องทำให้สามารถรับค่าเป็นชุดตัวอักษรได้ โดยเราใช้คำสั่ง Case เข้ามาช่วยในการรับค่า การเล่นไฟล์เสียงนั้นเราใช้โพรซิเยอร์ PlayWaveFile ซึ่งสามารถกำหนดให้มีการขัดจังหวะการเล่นไฟล์เสียงโดยใช้โพรซิเยอร์ InterruptWave โดยให้กำหนดให้ค่าเป็น True
     โปรแกรมนี้จะมีสถานะการทำงานอยู่ 7 สถานะด้วยกันดังนี้
     - StateIdle สถานะของโปรแกรมขณะที่ไม่มีการทำงานใดๆ
     - StateGreeting สถานะของโปรแกรมในการเริ่มต้นการตอบรับเมื่อมีผู้ใช้บริการโทรเข้ามา
     - StateGetUser สถานะของโปรแกรมทำการรับรหัสผู้ใช้งาน
     - StateGetPass สถานะของโปรแกรมทำการรับรหัสผ่าน
     - StateGetNum สถานะของโปรแกรมทำการรับเลขหมายโทรศัพท์ผู้ใช้ปลายทาง
     - StateSendFax สถานะของโปรแกรมขณะที่ทำการส่งโทรสาร
     - StateReceiveFax สถานะของโปรแกรมขณะที่ทำการรับโทรสาร
     - StateEndCall สถานะของโปรแกรมทำการตัดการเชื่อมต่อ

- OnTapiWaveNotify ทำหน้าที่ในการตรวจสอบไฟล์เสียงที่เล่นผ่านวอยส์โมเด็มอยู่ในขณะนั้นว่ากำลังเล่นไฟล์เสียงอะไรอยู่เพื่อตรวจสอบความถูกต้องจะมี TWaveMessages เป็นส่วนตรวจสอบ Message ที่ส่งมา

3.5.2 โปรแกรมทำการรับ-ส่งโทรสาร

- การสร้างส่วนการรับ-ส่งโทรสารมี 3 คอมโพเนนท์ที่สำคัญดังนี้

1. TApdComport ทำหน้าที่ในการจัดการการรับส่งข้อมูลผ่านทางพอร์ทอนุกรม โดยทำหน้าที่ในการเชื่อมต่อระหว่างโมเด็ม(Modem)กับเครื่องคอมพิวเตอร์

2. TApdSendFax ทำหน้าที่ในการควบคุมการทำงานต่างๆที่เกี่ยวกับการส่งโทรสาร

3. TApdReceiveFax ทำหน้าที่ในการควบคุมการทำงานต่างๆที่เกี่ยวกับการรับโทรสาร

- ส่วนการรับโทรสารซึ่งมีลักษณะการทำงานดังนี้

1. เนื่องจากการรับโทรสารต้องทำงานหลังจากการทำงานของระบบตอบรับด้วยเสียงจึงต้องทำการบอกโมเด็มให้รับรู้ว่าเราต้องการเปลี่ยนจากสถานะการติดต่อสื่อสารในรูปแบบเสียงให้เป็นการติดต่อสื่อสารในรูปของข้อมูล ซึ่งเราใช้โพรซีเยอร์ AutomatedVoiceToComms ในการเปลี่ยนโดยที่ไม่ต้องทำการจัดการเกี่ยวกับพอร์ทอนุกรมเพราะค่าที่ใช้เหมือนกับตอนใช้ระบบตอบรับด้วยเสียง

2. ทำการกำหนดค่าต่างๆที่จำเป็นต้องใช้ เช่น การกำหนดชื่อไฟล์สำหรับบันทึกข้อมูล

3. ทำการรับไฟล์รูปภาพจากเครื่องโทรสารของผู้ใช้บริการโดยใช้ StartManualReceive เป็นโพรซีเยอร์ในการรับโทรสาร

4. อีเวนท์ที่สำคัญในกระบวนการรับโทรสาร

- OnFaxError ทำหน้าที่ในการตรวจสอบข้อผิดพลาดที่เกิดขึ้นขณะรับโทรสาร จะมี ErrorCode บ่งบอกว่าข้อผิดพลาดที่เกิดขึ้นเนื่องจากสาเหตุใด
- OnFaxFinish ทำหน้าที่กำหนดการทำงานในขั้นตอนต่อไปเมื่อมีการรับโทรสารเสร็จเรียบร้อย ค่า ErrorCode จะมีค่าเท่ากับศูนย์ ในที่นี้จะส่งการทำงานต่อให้ส่วนส่งผ่านข้อมูลผ่านอินเตอร์เน็ต

- ส่วนของการส่งโทรสาร ซึ่งแบ่งได้เป็น 2 ชนิดดังนี้

1. การส่งโทรสารเมื่อมีการร้องขอจากระบบตอบรับด้วยเสียง ซึ่งมีขั้นตอนการทำงานดังนี้
   - ใช้โพรซีเยอร์ AutomatedVoiceToComms ในการเปลี่ยนสถานะการทำงานจากเสียงเป็นข้อมูล เหมือนกับขั้นตอนการรับโทรสารผ่านทางระบบตอบรับด้วยเสียง
   - ทำการกำหนดค่าต่างๆที่จำเป็นต้องใช้ เช่น กำหนดชื่อไฟล์สำหรับส่งโทรสาร
   - ทำการส่งไฟล์รูปภาพจากเครื่องคอมพิวเตอร์ของผู้ให้บริการโดยใช้ StartManualTransmit เป็นโพรซีเยอร์ในการรับโทรสาร

2. การส่งโทรสารไปยังผู้ใช้ปลายทาง ซึ่งมีขั้นตอนการทำงานดังนี้

- ใช้คอมโพเนนท์ TapdComport ซึ่งทำหน้าที่ในการจัดการการรับส่งข้อมูลผ่านทางพอร์ทอนุกรมโดยทำหน้าที่ในการเชื่อมต่อระหว่างโมเด็ม(Modem)กับเครื่องคอมพิวเตอร์
- เมื่อรับข้อมูลรูปภาพ, หมายเลขปลายทาง จากเครื่องคอมพิวเตอร์ต้นทาง จึงทำการกำหนดค่าที่สำคัญ เช่น การเลือกไฟล์รูปภาพที่ทำการส่งเราใช้พร๊อพเพอร์ตี้ FaxFile , การกำหนดหมายเลขปลายทางทำได้โดยใช้พร๊อพเพอร์ตี้ PhoneNumber ซึ่งกำหนดค่าเป็นสตริง
- ทำการส่งโทรสารไปยังผู้รับโทรสารปลายทางโดยใช้ StartTransmit เป็นโพรซีเยอร์ในการส่งโทรสาร

อีเวนท์ที่สำคัญในกระบวนการส่งโทรสาร

- OnFaxError ทำหน้าที่ในการตรวจสอบข้อผิดพลาดที่เกิดขึ้นขณะทำการส่งโทรสาร จะมี ErrorCode บ่งบอกว่าข้อผิดพลาดที่เกิดขึ้นเนื่องจากสาเหตุใด
- OnFaxFinish ทำหน้าที่กำหนดการทำงานในขั้นตอนต่อไปเมื่อมีการส่งโทรสารเสร็จเรียบร้อย ค่า ErrorCode จะมีค่าเท่ากับศูนย์

3.5.3 ส่วนทำหน้าที่ส่งผ่านข้อมูลผ่านเครือข่ายอินเตอร์เน็ต

การส่งไฟล์ผ่าน Internet จะทำการส่งอย่างอัตโนมัติ เมื่อมีไฟล์แฟกซ์ถูกส่งเข้ามาในโฟลเดอร์ของโปรแกรม โดยจะมองเป็น server ทั้งสองฝั่งคือฝั่งส่งและฝั่งรับ ซึ่งแต่ละฝั่งจะใช้คอมโพเนนท์ของภาษาเดลไฟล์ที่สำคัญๆคือ

1.ฝั่งส่ง

1.1 TIdTCPServer - เป็นคอมโพเนนท์ TCP (Transmission Control Protocol) server ที่รองรับ multi-thread คือ จะสามารถ listen หรือรองรับการเชื่อมต่อได้จากหลายๆ client มี Properties ที่สำคัญๆคือ

- Active บอกสถานะของ server มีค่าเป็น Boolean

และ Event ที่สำคัญคือ

- OnConnect เมื่อมีการเชื่อมต่อมาจากอีกฝั่ง
- OnDisconnect เมื่ออีกฝั่งทำการตัดการเชื่อมต่อ
- OnExecute เมื่อฝั่ง client ใช้ Method "Run"

2.ฝั่งรับ

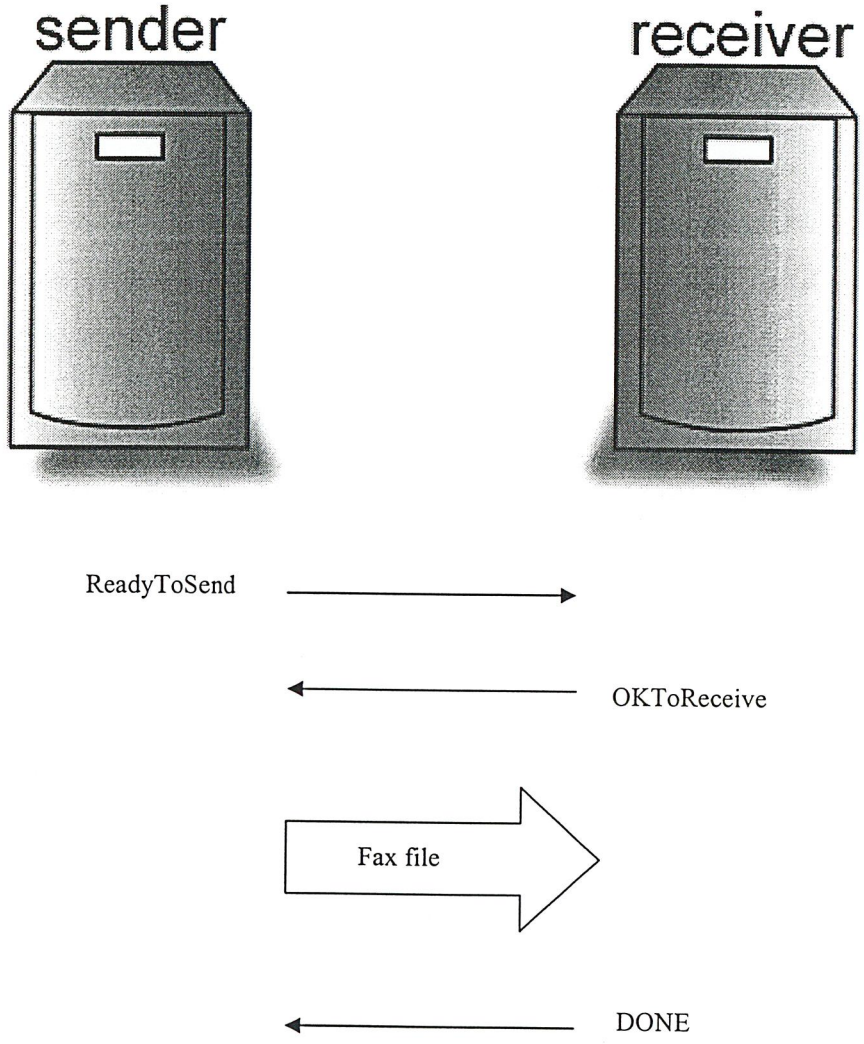
2.1 TIdTCPClient - เป็นคอมโพเนนท์ TCP (Transmission Control Protocol) client ทำหน้าที่เชื่อมต่อไปยัง TCPServer ที่ต้องการ โดย properties ดังต่อไปนี้

- Host เป็น properties ที่ระบุ server ปลายทาง ด้วย computer name หรือ ip address
- Port เป็น properties ที่ระบุ port ของ server ปลายทาง

และ Event ที่สำคัญคือ

- Connect ทำการเชื่อมต่อไปยัง server ปลายทาง ซึ่งจะต้องมีการระบุ IP server ปลายทาง บน properties Host และ port ของ server ปลายทาง บน properties Port

ในการ Transfer File จำเป็นต้องมีการสร้าง protocol เพื่อให้ server ทั้งสองฝั่งรับรู้ว่าจะมีการส่งไฟล์กัน โดยนี่จะเป็น protocol ที่สร้างขึ้นมาเพื่อส่งไฟล์ผ่าน internet



รูปที่ 3.8 แสดงโปรโตคอล ของการส่งข้อมูล

3.5.5 ส่วนการส่งข้อความเอสเอ็มเอส

ในการส่งเอสเอ็มเอส นั้น เราจะใช้การส่งผ่านทาง เอสเอ็มเอสเกตเวย์ (SMS Gateway) โดยไปใช้ของ www.clickatell.com โดยการส่งนั้นจะใช้การส่งผ่านทางเอฟทีพี (FTP) โดยจะทำการเขียนเอฟทีพีไคลเอนท์(FTP Client) เพื่อทำการติดต่อกับเซิร์ฟเวอร์ของทาง Clickatell โดยจะมีคอมโพเนนท์ในภาษาเดลไฟล์ดังนี้

TIdFTP เป็นคอมโพเนนท์ใช้ในการสร้างไคล์เอนท์เอฟทีพี โดยมี มีพร็อพเพอร์ตี้ ที่สำคัญๆคือ

-Host เป็นพร็อพเพอร์ตี้ที่ระบุ server ปลายทาง ด้วย computer name หรือ ip address เป็นตัว

-UserName เป็นพร็อพเพอร์ตี้ใส่ username ในการ Login

-Password เป็นพร็อพเพอร์ตี้ใส่ Password ในการ Login

-Put เป็นพร็อพเพอร์ตี้สำหรับการส่งไฟล์ข้อมูล ในที่นี้เราส่งไฟล์ที่เป็นข้อความ .txt โดยไฟล์ .txt จะมีฟอร์มในการเขียนดังนี้

api_id:527688
user:warcraft3
password:warcraft3 } (ส่วน username และ Password และ api_id ที่ได้จากการลงทะเบียน)

to:6667765319 (เบอร์ผู้ที่ต้องการจะส่ง แต่ต้องมีรหัสประเทศนำหน้า โดย 06-776-5319 รหัสประเทศ)

From:SFFS Server (หลัง From: ใส่ชื่อผู้ส่ง ในกรณีนี้ชื่อผู้ส่งคือ “SFFS Server”)

Text:Your Fax has been Sent !!23/3/2005 11:52:23 (หลัง Text: คือข้อความที่ต้องการจะส่ง)

3.5.6 ส่วนการจัดการฐานข้อมูล

ระบบฐานข้อมูลจะเก็บ user name และ password ของสมาชิก smart store forward fax system โดยระบบฐานข้อมูลจะใช้ Microsoft Access และในการตรวจสอบ user name และ password จะใช้คอมโพเนนท์ในภาษาเดลไฟล์ดังนี้

1. TADOConnection ทำหน้าที่ในการติดต่อกับแหล่งข้อมูล ควบคุมเงื่อนไข และสถานะการติดต่อกับฐานข้อมูล มีพร็อพเพอร์ตี้ ที่สำคัญคือ

- Connected กำหนดการเชื่อมต่อกับแหล่งข้อมูล มีค่าเป็น Boolean

Method ที่สำคัญ

- Open เปิดการเชื่อมต่อคอมโพเนนท์ ADOConnection กับฐานข้อมูล มีค่าเป็น Boolean

2. TADOTable ใช้เข้าถึงข้อมูลในตารางจากฐานข้อมูลที่ต้องการติดต่อ มี พร็อพเพอร์ตี้ที่สำคัญคือ

- Active เป็นค่าที่บ่งบอกว่าสามารถใช้งานข้อมูลในตารางได้หรือไม่ มีค่าเป็น Boolean
- Connecttion กำหนดคอมโพเนนท์ที่ต้องการใช้ในการเชื่อมต่อกับฐานข้อมูล
- TableName กำหนดชื่อตารางที่ต้องการนำมาใช้จากฐานข้อมูล

Method ที่สำคัญ

- Locate ใช้ค้นหาข้อมูลที่ต้องการในตาราง

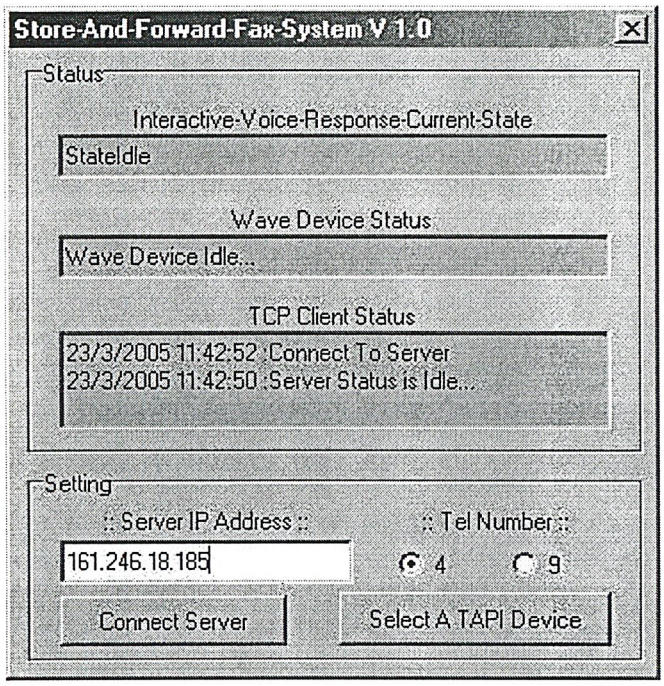
บทที่ 4

การทดลองและการทำงาน

4.1 การตั้งค่าเริ่มต้น

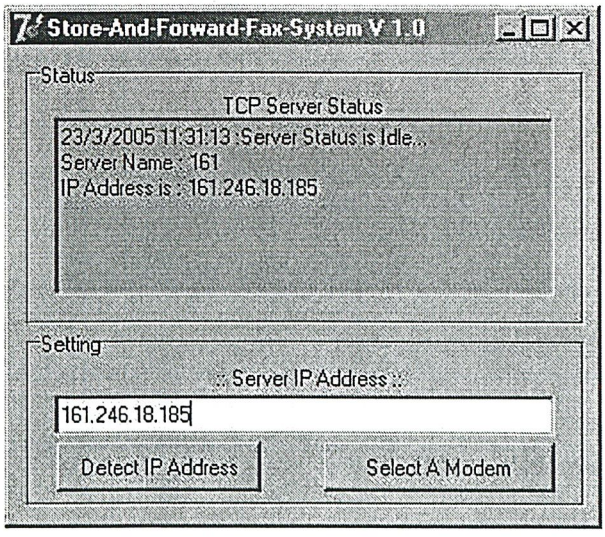
โครงงานนี้ประกอบไปด้วยโปรแกรม 2 โปรแกรมด้วยกันดังนี้

4.1.1 โปรแกรมลูกข่าย (Client SFFS)



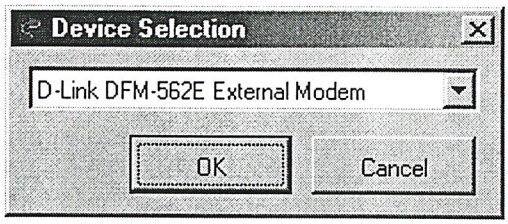
รูปที่ 4.1 แสดงหน้าจอส่วนควบคุมหลักของโปรแกรมลูกข่าย(Client SFFS)

4.1.2 โปรแกรมแม่ข่าย (Server SFFS)



รูปที่ 4.2 แสดงหน้าจอส่วนควบคุมหลักของโปรแกรมแม่ข่าย(Server SFFS)

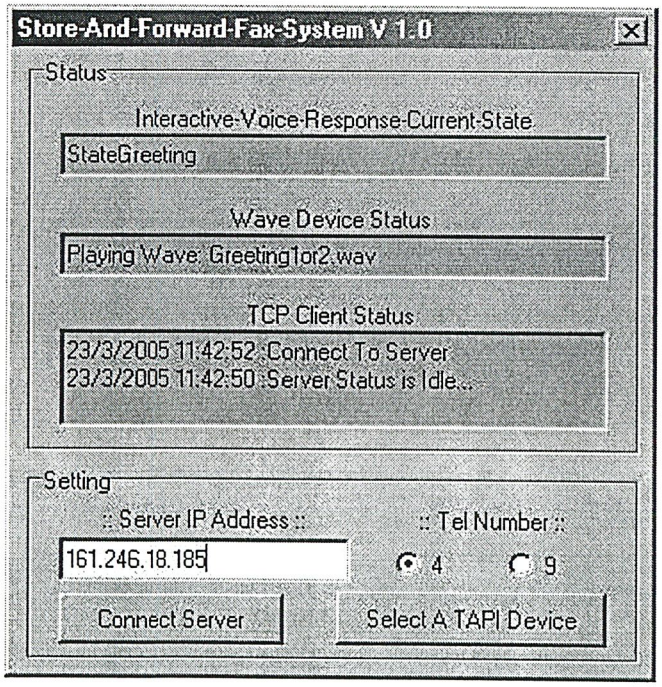
ซึ่งการตั้งค่าเริ่มต้นจะต้องกำหนดค่าไอพีแอดเดรส(IP Address) ซึ่งจะต้องกำหนดเป็นค่าของทางเครื่องคอมพิวเตอร์ฝั่งแม่ข่ายซึ่งจะต้องกำหนดเหมือนกันทั้ง 2 ฝั่ง ต่อมาจึงกำหนดค่าพอร์ทอนุกรมที่ใช้อยู่ติดต่อกับอุปกรณ์ใดบ้าง โดยเลือกที่ปุ่ม Select A TAPI Device จะขึ้นหน้าจอตามรูปที่ 4.3 แล้วจึงทำการเลือกโมเด็มที่ต้องการใช้งานในโปรแกรม



รูปที่ 4.3 แสดงส่วนหน้าจอการเลือกกำหนดโมเด็มที่ใช้งาน

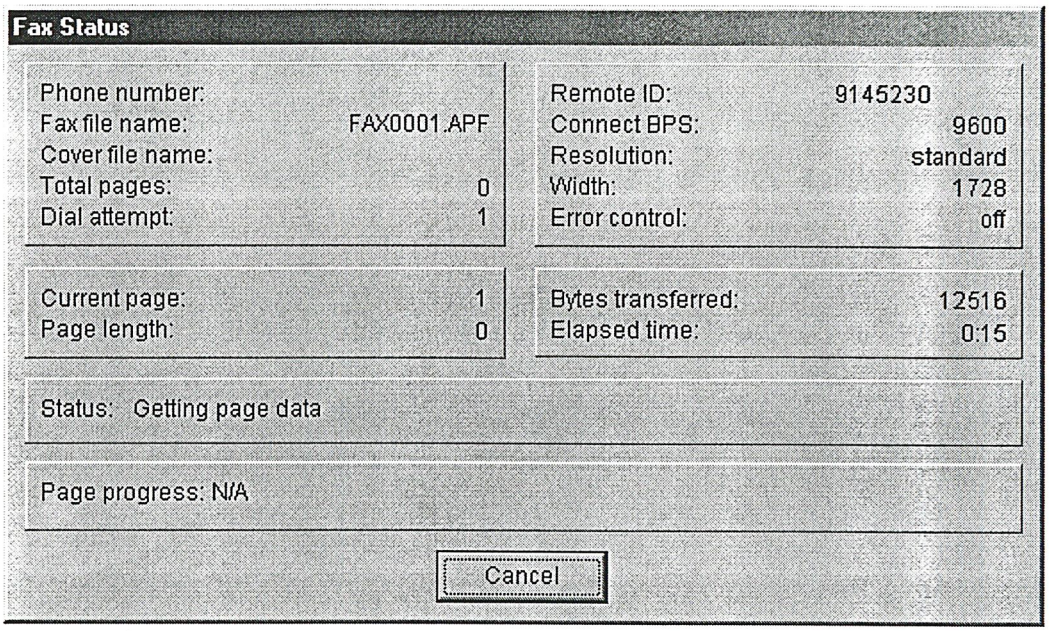
4.2 ขั้นตอนการทำงาน

ก่อนให้โปรแกรมเริ่มทำงาน ควรตรวจสอบว่าการเชื่อมต่ออินเตอร์เน็ตและการต่ออุปกรณ์โมเด็มได้ทำอย่างถูกต้องแล้ว เมื่อสั่งให้โปรแกรมเริ่มทำงานโดยในที่นี่จะหมายถึงโปรแกรมทางฝั่งลูกข่าย โปรแกรมจะทำการรอตอบรับสายโทรศัพท์จากผู้ใช้งาน ดังรูปที่ 4.1 เมื่อมีผู้ใช้งานโทรศัพท์เข้ามาที่โปรแกรมลูกข่าย โปรแกรมจะทำการรับสายแล้วเล่นไฟล์เสียงตอบรับ ดังรูปที่ 4.4



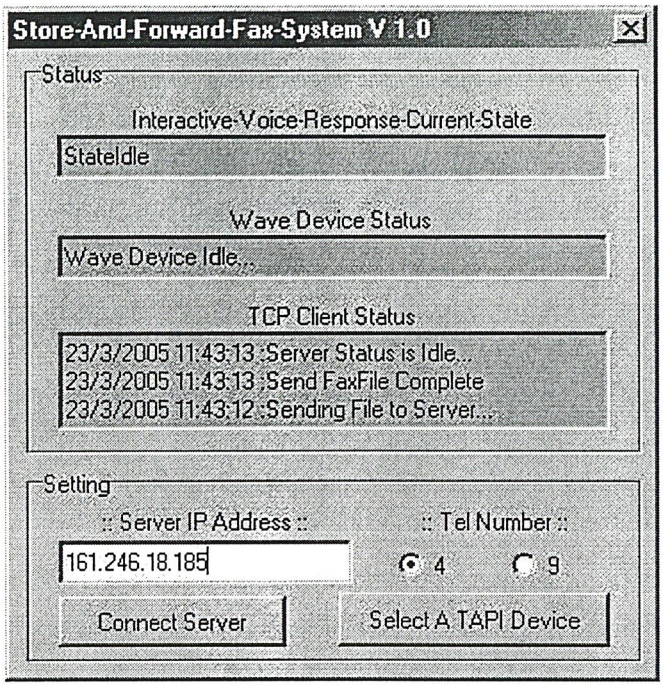
รูปที่ 4.4 แสดงหน้าจอส่วนโปรแกรมลูกข่ายขณะทำการเล่นไฟล์เสียงต้อนรับ

จากนั้นจึงผู้ใช้งานต้องทำการเลือกว่าจะทำการส่งโทรสารหรือทำการขอรับโทรสาร ซึ่งต่อมามีกระบวนการในการยืนยันผู้ใช้งานด้วยรหัสผู้ใช้งานและรหัสผ่าน ว่าได้ลงทะเบียนกับทางผู้ให้บริการแล้วหรือไม่ ถ้าทำการส่งโทรสารต้องกดหมายเลขโทรศัพท์ผู้รับโทรสารปลายทาง เมื่อหมายเลขปลายทางถูกต้องแล้ว โปรแกรมจะทำการรอรับโทรสารจากผู้ใช้งาน เมื่อผู้ใช้งานกดส่งโทรสารจากเครื่องโทรสาร โปรแกรมจะแสดงสถานะ การทำงานขณะรับโทรสารดังรูปที่ 4.5



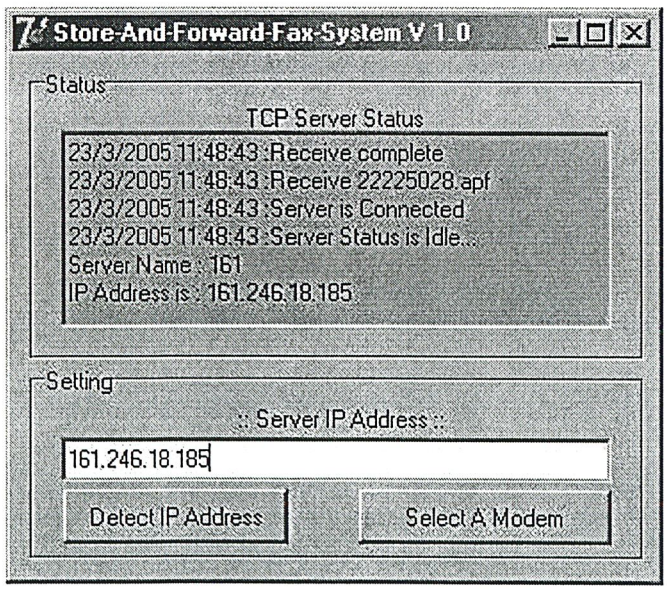
รูปที่ 4.5 แสดงหน้าจอส่วนโปรแกรมลูกข่ายขณะแสดงสถานะ การรับโทรสาร

เมื่อทำการรับโทรสารเสร็จแล้ว โปรแกรมจะทำการตัดการติดต่อ แล้วกลับไปเปิดระบบตอบรับด้วยเสียงเพื่อรอตอบรับสายโทรศัพท์จากผู้ใช้งานต่อไป แต่โปรแกรมยังต้องทำการเปลี่ยนชื่อไฟล์รูปภาพที่รับมาให้เป็นชื่อของรหัสผู้ใช้งานรวมกับหมายเลขโทรศัพท์ปลายทาง จากนั้นจึงทำการส่งไฟล์ข้อมูลรูปภาพไปยัง เครื่องคอมพิวเตอร์แม่ข่าย (Server SFFS) โดยมีสถานะ การส่งไฟล์ข้อมูลตามรูปที่ 4.6

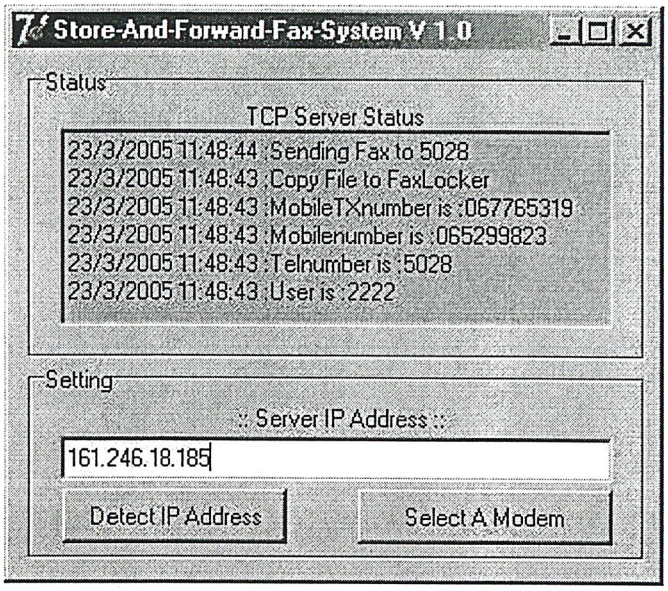


รูปที่ 4.6 แสดงหน้าจอส่วนโปรแกรมลูกข่ายขณะส่งไฟล์โทรสารไปยังโปรแกรมแม่ข่าย

ทางโปรแกรมแม่ข่าย(Server SFFS) จึงทำการรับข้อมูลไฟล์รูปภาพจากคอมพิวเตอร์ฝั่งลูกข่าย ดังรูปที่ 4.7 แล้วนำข้อมูลที่ได้ไปทำการวิเคราะห์ชื่อไฟล์ข้อมูลเพื่อหารหัสผู้ใช้ กับเบอร์โทรศัพท์ผู้รับโทรสารปลายทาง จากนั้นโปรแกรมจึงนำข้อมูลที่ผ่านการวิเคราะห์แล้วไปเก็บไว้ในระบบล็อกเกอร์โทรสารตามรหัสผู้ใช้งาน ซึ่งมีสถานะ การรับข้อมูลโทรสารดังรูปที่ 4.8

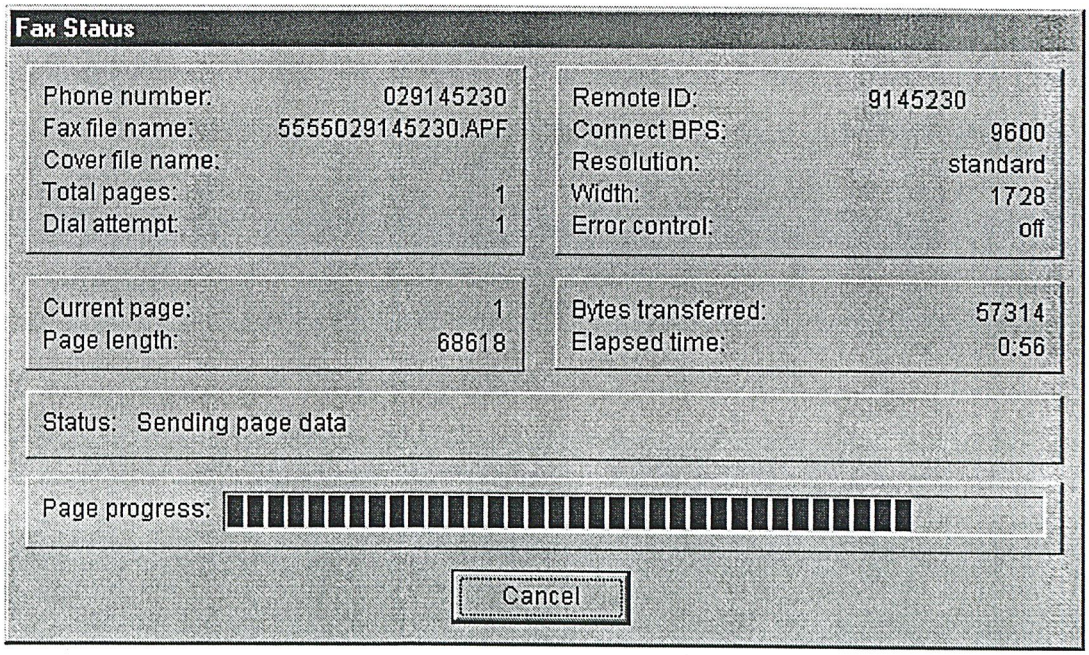


รูปที่ 4.7 แสดงหน้าจอส่วนโปรแกรมแม่ข่ายขณะแสดงสถานะ การรับข้อมูลโทรสาร



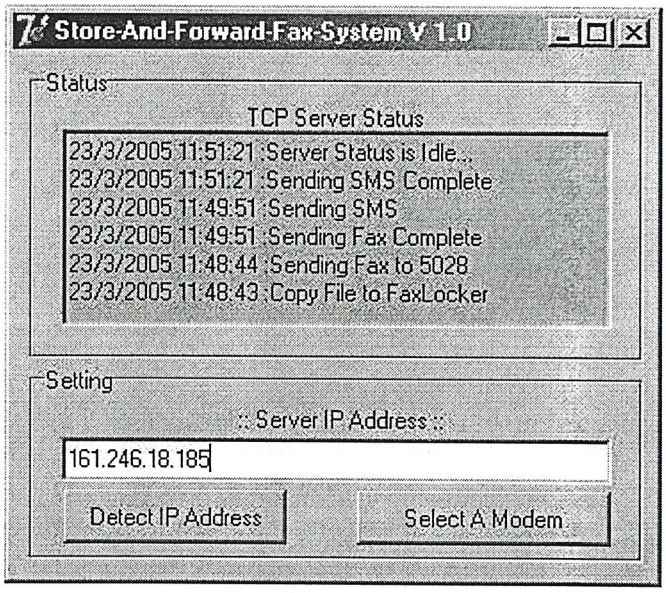
รูปที่ 4.8 แสดงสถานะของโปรแกรมแม่ข่ายขณะจัดเก็บเอกสารไปยังล็อกเกอร์โทรสารแล้วส่งโทรสาร

ต่อมาโปรแกรมจึงทำการส่งโทรสารตามหมายเลขผู้ใช้บริการปลายทางที่วิเคราะห์ไว้ซึ่งมีสถานะการส่งโทรสารไปยังผู้ใช้บริการปลายทางตามรูปที่ 4.9



รูปที่ 4.9 แสดงหน้าจอส่วนโปรแกรมแม่ข่ายขณะแสดงสถานะการส่งโทรสารไปยังผู้ใช้บริการปลายทาง

เมื่อทำการส่งโทรสารเสร็จเรียบร้อยแล้ว โปรแกรมแม่ข่ายจึงทำการส่งเอสเอ็มเอสไปยังผู้ส่งโทรสารเพื่อบอกสถานะว่าตอนนี้ผู้รับโทรสารปลายทางได้รับโทรสารเรียบร้อยแล้ว ซึ่งสถานะ การทำงานเป็นไปตามรูปที่ 4.10



รูปที่ 4.10 แสดงสถานะของโปรแกรมแม่ข่ายขณะส่งเอสเอ็มเอสแล้วการรอรับโทรสารครั้งต่อไป

เมื่อส่งโทรสารจากเครื่องคอมพิวเตอร์แม่ข่ายของผู้รับโทรสารปลายทาง เสร็จเรียบร้อยแล้ว จากนั้นโปรแกรมแม่ข่ายจึงกลับเข้าสู่กระบวนการรอรับไฟล์ข้อมูลจากเครื่องคอมพิวเตอร์ต้นทาง ต่อไป โดยเอกสารที่นำมาส่งโทรสารแสดงดังรูปที่ 4.11 ซึ่งโทรสารที่ได้รับแสดงดังรูปที่ 4.12 ความละเอียดภาพที่ได้รับถึงจะลดลงแต่ยังสามารถสื่อสารได้อย่างสมบูรณ์

(ปิดอากรแสตมป์ 10 บาท ตามประมวลรัษฎากร)

หนังสือมอบอำนาจ

(บุคคลทั่วไป)

เรื่อง.....

เขียนที่.....

วันที่.....เดือน.....พ.ศ. .....

โดยหนังสือฉบับนี้ ข้าพเจ้า.....

อายุ.....ปี เชื้อชาติ.....สัญชาติ.....อยู่บ้านเลขที่.....

ตรอก, ซอย.....ถนน.....ตำบล/แขวง.....

อำเภอ/เขต.....จังหวัด.....

ได้มอบอำนาจให้.....อายุ.....ปี

เชื้อชาติ.....สัญชาติ.....อยู่บ้านเลขที่.....

ตรอก, ซอย.....ถนน.....ตำบล/แขวง.....

อำเภอ/เขต.....จังหวัด.....เป็นผู้มีอำนาจ

จัดการ.....

แทนข้าพเจ้าจนเสร็จการและข้าพเจ้ายอมรับผิดชอบในการที่ผู้รับมอบอำนาจของข้าพเจ้าได้ทำไปตามที่มอบอำนาจนี้เสมือนหนึ่งข้าพเจ้าได้ทำการเองด้วยตนเอง เพื่อเป็นหลักฐานข้าพเจ้าได้ลงลายมือชื่อไว้เป็นสำคัญต่อหน้าพยานแล้ว

.....***ผู้มอบอำนาจ***

.....***ผู้รับมอบอำนาจ***

ข้าพเจ้าขอรับรองว่าเป็นลายมือ หรือ ลายนิ้วมืออันแท้จริงของผู้มอบอำนาจกับผู้รับมอบอำนาจ และผู้รับมอบอำนาจกับผู้มอบอำนาจได้ลงลายมือชื่อต่อหน้าข้าพเจ้า

.....***พยาน***

.....***พยานและผู้เขียนข้อความ***

บัตรประจำตัวของผู้มอบอำนาจ

เลขที่.....

บ้านเลขที่.....ตรอก, ซอย.....

ถนน.....

ตำบล/แขวง.....

อำเภอ/เขต.....

จังหวัด.....

ออกให้ ณ วันที่.....

บัตรประจำตัวของผู้รับมอบอำนาจ

เลขที่.....

บ้านเลขที่.....ตรอก, ซอย.....

ถนน.....

ตำบล/แขวง.....

อำเภอ/เขต.....

จังหวัด.....

ออกให้ ณ วันที่.....

รูปที่ 4.11 เอกสารที่นำมาส่งโทรสาร

(ปิดอากรแสตมป์ 10 บาท ตามประมวลรัษฎากร)

หนังสือมอบอำนาจ

(บุคคลทั่วไป)

เรื่อง .....

เขียนที่ .....

วันที่ ..... เดือน ..... พ.ศ. .....

โดยหนังสือฉบับนี้ ข้าพเจ้า .....

อายุ ..... ปี เชื้อชาติ ..... สัญชาติ ..... อยู่บ้านเลขที่ .....

ตรอก/ซอย ..... ถนน ..... ตำบล/แขวง .....

อำเภอ/เขต ..... จังหวัด .....

ได้มอบอำนาจให้ ..... อายุ ..... ปี

เชื้อชาติ ..... สัญชาติ ..... อยู่บ้านเลขที่ .....

ตรอก/ซอย ..... ถนน ..... ตำบล/แขวง .....

อำเภอ/เขต ..... จังหวัด ..... เป็นผู้มีอำนาจ

จัดการ .....

แทนข้าพเจ้าจนเสร็จการและข้าพเจ้ายอมรับผิดชอบในการที่ผู้รับมอบอำนาจของข้าพเจ้าได้ทำไปตามที่มอบอำนาจนี้เสมือนหนึ่งข้าพเจ้าได้ทำการเองด้วยตนเอง เพื่อเป็นหลักฐานข้าพเจ้าได้ลงลายมือชื่อไว้เป็นสำคัญต่อหน้าพยานแล้ว

..... ***ผู้มอบอำนาจ***

..... ***ผู้รับมอบอำนาจ***

ข้าพเจ้าขอรับรองว่าเป็นลายมือ หรือ ลายนิ้วมืออันแท้จริงของผู้มอบอำนาจกับผู้รับมอบอำนาจ และผู้รับมอบอำนาจกับผู้มอบอำนาจได้ลงลายมือชื่อต่อหน้าข้าพเจ้า

..... ***พยาน***

..... ***พยานและผู้เขียนข้อความ***

บัตรประจำตัวของผู้มอบอำนาจ

เลขที่ .....

บ้านเลขที่ ..... ตรอก/ซอย .....

ถนน .....

ตำบล/แขวง .....

อำเภอ/เขต .....

จังหวัด .....

ออกให้ ณ วันที่ .....

บัตรประจำตัวของผู้รับมอบอำนาจ

เลขที่ .....

บ้านเลขที่ ..... ตรอก/ซอย .....

ถนน .....

ตำบล/แขวง .....

อำเภอ/เขต .....

จังหวัด .....

ออกให้ ณ วันที่ .....

รูปที่ 4.12 เอกสารที่ได้รับจากเซิร์ฟเวอร์ปลายทาง

ในกรณีที่ผู้รับได้รับโทรสารผ่านทางเครื่องโทรสาร เครื่องแม่ข่ายจะทำการส่งข้อความเอสเอ็มเอส(SMS Messenger) เข้าโทรศัพท์มือถือของผู้ส่งโทรสารเพื่อแจ้งให้ทราบว่าผู้รับได้รับโทรสารผ่านทางเครื่องโทรสารดังรูปที่ 4.13

รูปที่ 4.13 แสดงข้อความเอสเอ็มเอสแจ้งผู้ส่งว่าผู้รับได้รับโทรสารผ่านทางเครื่องโทรสาร

และในกรณีที่เครื่องแม่ข่ายส่งโทรสารไปหาผู้รับ แต่การส่งไม่สมบูรณ์ หรือผู้รับไม่ได้รับโทรสารผ่านทางเครื่องโทรสาร เครื่องแม่ข่ายก็จะแจ้งผู้ส่งผ่านทาง เอสเอ็มเอสว่าการส่งโทรสารล้มเหลว ดังรูปที่ 4.14

รูปที่ 4.14 แสดงข้อความเอสเอ็มเอสแจ้งผู้ส่งว่าผู้รับไม่ได้รับโทรสารผ่านทางเครื่องโทรสาร

แล้วทางเครื่องแม่ข่ายก็จะนำเบอร์โทรศัพท์มือถือข้อผู้รับ เพื่อนำมาส่งเอสเอ็มเอสหาผู้รับ เพื่อแจ้งให้ผู้รับทราบว่ามีโทรสารถึงผู้รับ แต่ให้ผู้รับเข้าไปรับโทรสารด้วยตนเอง โดยจะแจ้งเบอร์ติดต่อ ชื่อผู้ใช้(Username) และรหัสผ่าน (Passwords) ดังรูปที่ 4.15

รูปที่ 4.15 แสดงข้อความเอสเอ็มเอสแจ้งผู้รับว่าให้ผู้รับเข้าไปรับโทรสารด้วยตนเอง

บทที่ 5
บทวิจารณ์ และ บทสรุป

ปริญญานิพนธ์ระบบจัดเก็บและส่งต่อโทรสาร ยังมีปัญหาในส่วนระบบตอบรับด้วยเสียง โดยที่ยังไม่สามารถตรวจจับการวางหูของผู้ใช้บริการได้ แต่ก็แก้ปัญหาได้โดยวิธีการตั้งเวลาตัดการเชื่อมต่อ เมื่อไม่มีการตอบสนองจากผู้ใช้งาน โปรแกรมยังมีบั๊กเล็กน้อยในระบบตอบรับด้วยเสียงเช่นในบางครั้งใช้เวลาในการประมวลเสียงที่ใช้นานเกินไปทำให้ต้องรอสายประมาณ 1-2 วินาที ยังมีปัญหาในการตรวจจับสัญญาณแป้นโทรศัพท์บ้างในบางครั้งถ้าหากผู้ใช้บริการกดแป้นโทรศัพท์เร็วเกินไป เวลาส่งโทรสารไปตามบ้านผู้รับบริการสามารถใช้กับเครื่องโทรสารแบบระบบอัตโนมัติเท่านั้นเพราะเมื่อเซิร์ฟเวอร์ปลายทางทำการติดต่อไปยังผู้รับโทรสารปลายทางนั้น ไม่สามารถเล่นไฟล์เสียงบอกผู้รับโทรสารว่ามีโทรสารเข้ามาให้กดปุ่มเพื่อรอรับโทรสารได้ ซึ่งผู้รับโทรสารอาจไม่ทราบว่ามีโทรสารเข้ามาเลยทำให้รับโทรสารไม่สำเร็จ

แนวทางในการพัฒนาโครงงานสำหรับผู้ที่สนใจ นำไปพัฒนาให้สามารถรับข้อมูลรูปภาพหรือจดหมายอิเล็กทรอนิกส์ แล้วส่งเป็นโทรสาร นำระบบตอบรับด้วยเสียงไปพัฒนาในไปใช้ประโยชน์อย่างอื่น เช่น CallCenter ระบบจองบัตรชมภาพยนต์ เป็นต้น หรืออาจจะดัดแปลงการส่ง เอสเอ็มเอสในการใช้งานด้านอื่นๆ

ภาคผนวก

```
//Program SFFS Server
unit Server_Beta;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, IdBaseComponent, IdComponent, IdTCPServer, IdThreadMgr,
  IdThreadMgrDefault, StdCtrls, Winsock, AdPort, AdFax, AdFStat,
OoMisc, AdTapi, ExtCtrls, IdTCPConnection, IdTCPClient, IdFTP, DB,
ADODB;

type
  TForm1 = class(TForm)
    TCPServer: TIdTCPServer;
    ThreadManager: TIdThreadMgrDefault;
    GroupBox1: TGroupBox;
    mmStatus: TMemo;
    Label4: TLabel;
    GroupBox2: TGroupBox;
    Label3: TLabel;
    btnSelTapi: TButton;
    btnDetect: TButton;
    Edit3: TEdit;
    ApdSendFax1: TApdSendFax;
    ApdFaxStatus1: TApdFaxStatus;
    ComPort1: TApdComPort;
    ApdTapiDevice1: TApdTapiDevice;
    ReSendTimer1: TTimer;
    SmsFTP: TIdFTP;
    SMSmemo: TMemo;
    ADOConnection1: TADOConnection;
    ADOQuery1: TADOQuery;
    procedure TCPServerExecute(AThread: TIdPeerThread);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure TCPServerConnect(AThread: TIdPeerThread);
    procedure TCPServerDisconnect(AThread: TIdPeerThread);
    procedure btnDetectClick(Sender: TObject);
    procedure ApdSendFax1FaxFinish(CP: TObject; ErrorCode: Integer);
    procedure btnSelTapiClick(Sender: TObject);
    procedure ReSendTimer1Timer(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  FaxLockerDir : String;
  ResendCount : integer;
  MobileTx,MobileRx : String;
```

```
  User,Pass : string[4];

implementation

{$R *.dfm}

function GetIPFromHost
(var HostName, IPaddr, WSAErr: string): Boolean;
type
  Name = array[0..100] of Char;
  PName = ^Name;
var
  HEnt: pHostEnt;
  HName: PName;
  WSAData: TWSAData;
  i: Integer;
begin
  Result := False;
  if WSAStartup($0101, WSAData) <> 0 then begin
    WSAErr := 'Winsock is not responding."';
    Exit;
  end;
  IPaddr := '';
  New(HName);
  if GetHostName(HName^, SizeOf(Name)) = 0 then
  begin
    HostName := StrPas(HName^);
    HEnt := GetHostByName(HName^);
    for i := 0 to HEnt^.h_length - 1 do
     IPaddr :=
      Concat(IPaddr,
      IntToStr(Ord(HEnt^.h_addr_list^[i])) + '.');
    SetLength(IPaddr, Length(IPaddr) - 1);
    Result := True;
  end
  else begin
   case WSAGetLastError of
    WSANOTINITIALISED:WSAErr:='WSANotInitialised';
    WSAENETDOWN       :WSAErr:='WSAENetDown';
    WSAEINPROGRESS    :WSAErr:='WSAEInProgress';
   end;
  end;
  Dispose(HName);
  WSACleanup;
end;

procedure TForm1.TCPServerExecute(AThread: TIdPeerThread);
var
    Size,accSize : integer;
    FaxPathName,TempPathName : string;
    xxxStream,accStream : TFileStream;
    RxCommand,Rx2Command : string;
    accName,buf,FaxLock,CreateFolder : string;
    Telnumber  : string[9];
    TexttoRead : textFile;
begin
```

```
  Telnumber := '';
  if not AThread.Terminated and AThread.Connection.Connected then
  begin
    RxCommand := AThread.Connection.ReadLn;
    if RxCommand = 'ReadyToSend' then
    begin
      AThread.Connection.WriteLn('OKToReceive');
      TempPathName := GetCurrentDir + '\' + 'Temp.txt';
      if FileExists (TempPathName) then DeleteFile(TempPathName);
      accStream := TFileStream.Create(TempPathName,fmCreate);
      accSize := AThread.Connection.ReadInteger;
      AThread.Connection.ReadStream(accStream,accSize,False);
      FreeAndNil(accStream);


      {Process FTP}
      AssignFile(TexttoRead,TempPathName);
      Reset(TexttoRead);
      ReadLn(TexttoRead,buf);
      mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :DaTa_in_TextFile
is :'+buf);
      User := Copy(buf,1,4);
      MobileTx :=Copy(buf,5,9);
      MobileRX :=Copy(buf,14,9);
      Telnumber :=Copy(buf,23,4);
      accName := User+Telnumber;
      CloseFile(TexttoRead);
      DeleteFile(TempPathName);

      {Next File}
      AThread.Connection.WriteLn('NeedData');
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Receive'+'
'+accName+'.apf');
        FaxPathName := GetCurrentDir + '\' + accName+'.apf';
        if FileExists (FaxPathName) then DeleteFile(FaxPathName);
        xxxStream := TFileStream.Create(FaxPathName,fmCreate);
        Size := AThread.Connection.ReadInteger;
        AThread.Connection.ReadStream(xxxStream,Size,False);
        FreeAndNil(xxxStream);
        AThread.Connection.WriteLn('DONE');
      Rx2Command := AThread.Connection.ReadLn;
      if Rx2Command = 'Bye' then
        begin
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Receive complete');
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :User is :'+User);
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Telnumber is
:'+Telnumber);
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Mobilenumber is
:'+MobileRx);
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :MobileTXnumber is
:'+MobileTx);
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Copy File to
FaxLocker');
   MobileTx :='66'+Copy(MobileTx,2,8);
   MobileRX :='66'+Copy(MobileRx,2,8);
```

```
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Mobilenumber is
:'+MobileRx);
   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :MobileTXnumber is
:'+MobileTx);

          {FAXLOCKER}
          CreateFolder := FaxLockerDir+user+'\';
          CreateDirectory(pchar(CreateFolder), nil);
          FaxLock := FaxLockerDir+User+'\'+User+Telnumber+'.apf';
          if FileExists (Faxlock) then DeleteFile(Faxlock);
          MoveFile(PChar(FaxPathName), PChar(FaxLock));
          DeleteFile(FaxPathName);

          {SendFAX}
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending Fax
to '+Telnumber);
          ApdSendFax1.DialRetryWait := 30; {if line busy}
          ApdSendFax1.DialAttempts  := 3;
          ApdSendFax1.PhoneNumber   := Telnumber;
          ApdSendFax1.FaxFile       := FaxLock;
          comport1.Open := true;
          ApdSendFax1.StartTransmit;
        end;
    end;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  Host, IP, Err: string;
begin
  User := '';
  ReSendTimer1.Interval :=10000;
  ResendCount := 1;
  {SMS int MODULE}
  smsftp.Host := 'ftpupload.clickatell.com' ;
  smsFtp.Username := '527688' ;
  smsftp.Password := 'warcraft3' ;
  {end sms module}
  if GetIPFromHost(Host, IP, Err) then
    begin
      TCPServer.Bindings.Add.IP := IP;
      TCPServer.Bindings.Add.Port := 17777 ;
      TCPServer.Active := True;
      Edit3.Text := IP ;
      mmStatus.Lines.Insert(0,'IP Address is : '+IP);
      mmStatus.Lines.Insert(0,'Server Name : '+Host);
      mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status is
Idle...');
      FaxLockerDir := GetCurrentDir+'\FaxLocker\';
    end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
```

```
begin
  if TCPServer.Active = true then
  TCPServer.Active := False;
end;

procedure TForm1.TCPServerConnect(AThread: TIdPeerThread);
begin
  mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server is
Connected');
end;

procedure TForm1.TCPServerDisconnect(AThread: TIdPeerThread);
begin
  mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server is
Disconnected');
end;
```

```
procedure TForm1.btnDetectClick(Sender: TObject);
var
  Host, IP, Err: string;
begin
  TCPServer.Active := False;
  TCPServer.Bindings.Clear ;
  if GetIPFromHost(Host, IP, Err) then
    begin
      TCPServer.Bindings.Add.IP := IP;
      TCPServer.Bindings.Add.Port := 17777 ;
      TCPServer.Active := True;
      Edit3.Text := IP ;
      mmStatus.Lines.Insert(0,'IP Address is : '+IP);
      mmStatus.Lines.Insert(0,'Server Name : '+Host);
      mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status is
Idle...');
    end;
end;

procedure TForm1.ApdSendFax1FaxFinish(CP: TObject; ErrorCode:
Integer);
var
    a : integer;
    b : string;
begin
  Randomize;
  a := random(1000000); {random because SMS file must unique}
  b := inttostr(a);
  if ErrorCode = 0 then
      begin
        ResendCount := 1;
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending Fax
Complete');
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending SMS ');
```

```
        {send SMS To FAX USER To Know that fax send complete}
        try
        SmsMemo.Lines.Delete(3);
        SmsMemo.Lines.Delete(3);
        SmsMemo.Lines.Delete(3);
        SmsMemo.Lines.Delete(3);
        smsftp.Connect;
        DeleteFile(GetCurrentDir+'\smstx.txt');
        SmsMemo.Lines.Insert(3,'to:'+MobileTx);
        SmsMemo.Lines.Insert(4,'From:SFFS Server');
        SmsMemo.Lines.Insert(5,'Text:Your Fax has been Sent
!!'+DateTimeToStr(Now));
        SmsMemo.Lines.SaveToFile(GetCurrentDir+'\smstx.txt');
        smsftp.Put(GetCurrentDir+'\smstx.txt','smstx'+b+'.txt');
        smsftp.Quit;
        smsftp.Disconnect;
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending SMS
Complete');
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status
is Idle...');
        except on E:Exception do
         begin
         mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t connect
SMS Server');
         smsftp.Disconnect;
         smsftp.Abort;
         mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status
is Idle...');
         end;
        end;
       {end SMSTX}
       end
       else
       begin
         mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending Fax
Not Complete');
         ResendCount := ResendCount+1  ;
         if ResendCount <4 then
         begin
         ResendTimer1.Enabled := true;
         end
           else  {SMS to Receive Fax USER}
           begin
                   with ADOQuery1 do
           begin
               Close;
               With SQL do
                 begin
                 Clear;
                 Add('SELECT *');
                 Add('FROM account');
                 Add('WHERE username=(:Username)');
                 Parameters[0].Value := user;
                 Prepared:=True;
                 end;
```

```
              Active:=True;
              Open;//alternatively can use ExecSQL
              if RecordCount=1 then
              begin
                Pass := FieldByName('Password').AsString;
                mmStatus.Lines.Insert(0,DateTimeToStr(Now)+'
:PassWord is : '+pass);
              end
              else mmStatus.Lines.Insert(0,DateTimeToStr(Now)+'
:Fail to access the SFFS Database');
            end;
            DeleteFile(GetCurrentDir+'\smsRx.txt');
            try
          SmsMemo.Lines.Delete(3);
          SmsMemo.Lines.Delete(3);
          SmsMemo.Lines.Delete(3);
          SmsMemo.Lines.Delete(3);
          smsftp.Connect;
          DeleteFile(GetCurrentDir+'\smstxfail.txt');
          SmsMemo.Lines.Insert(3,'to:'+MobileTx);
          SmsMemo.Lines.Insert(4,'From:SFFS Server');
          SmsMemo.Lines.Insert(5,'Text:Your Fax has been Sent
Failure!!'+DateTimeToStr(Now));
          SmsMemo.Lines.SaveToFile(GetCurrentDir+'\smstxfail.txt');

smsftp.Put(GetCurrentDir+'\smstxfail.txt','smstxfail'+b+'.txt');
          smsftp.Quit;
          smsftp.Disconnect;
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending SMS
Complete');
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status
is Idle...');
          except on E:Exception do
           begin
           mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t connect
SMS Server');
           smsftp.Disconnect;
           smsftp.Abort;
           mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status
is Idle...');
           end;
          end; {TX FAIL SMS}
             try
             smsftp.Connect;
             DeleteFile(GetCurrentDir+'\smsrx.txt');
             SmsMemo.Lines.Delete(3);
             SmsMemo.Lines.Delete(3);
             SmsMemo.Lines.Delete(3);
             SmsMemo.Lines.Delete(3);
             SmsMemo.Lines.Insert(3,'to:'+MobileRx);
             SmsMemo.Lines.Insert(4,'From:SFFS Server');
             SmsMemo.Lines.Insert(5,'Text:Your have New Fax To You Plz
Call SFFS server Telnumber:3420 Username is :'+User+' Password is
:'+Pass);
             SmsMemo.Lines.SaveToFile(GetCurrentDir+'\smsrx.txt');
```

```
          smsftp.Put(GetCurrentDir+'\smsrx.txt','smsrx'+b+'.txt');
          smsftp.Quit;
          smsftp.Disconnect;
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Sending SMS
To RxUser Complete');
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server
Status is Idle...');
          except on E:Exception do
            begin
            mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t
connect SMS Server');
            smsftp.Disconnect;
            smsftp.Abort;
            mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server
Status is Idle...');
            end;
          end;
          end;  {RX FAIL SMS}
      end;
end;

procedure TForm1.btnSelTapiClick(Sender: TObject);
begin
  ApdTapiDevice1.SelectDevice;
end;

procedure TForm1.ReSendTimer1Timer(Sender: TObject);
begin
    ApdSendFax1.StartTransmit;
    ReSendTimer1.Enabled := false;
end;
end.


//Program SFFS Client
unit IVRFAXS;

interface

uses
  WinTypes,
  WinProcs,
  Messages,
  SysUtils,
  Classes,
  Graphics,
  Controls,
  StdCtrls,
  ExtCtrls,
  Forms,
  Dialogs,
  AdFax,
  AdFStat,
  AdTapi,
  AdPort,
  AdTUtil,
```

```
  OOMisc, XPMan, IdAntiFreezeBase, IdAntiFreeze, IdBaseComponent,
  IdComponent, IdTCPConnection, IdTCPClient, DB, ADODB;

type
  TForm1 = class(TForm)
    ApdTapiDevice1: TApdTapiDevice;
    ApdComPort1: TApdComPort;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    ApdFaxStatus1: TApdFaxStatus;
    ApdReceiveFax1: TApdReceiveFax;
    ApdFaxStatus2: TApdFaxStatus;
    ApdSendFax1: TApdSendFax;
    mmStatus: TMemo;
    Label4: TLabel;
    GroupBox2: TGroupBox;
    Button1: TButton;
    edtServer: TEdit;
    Label3: TLabel;
    TCPClient: TIdTCPClient;
    IdAntiFreeze1: TIdAntiFreeze;
    btnConnect: TButton;
    Timer1: TTimer;
    Label5: TLabel;
    Telnum4: TRadioButton;
    Telnum9: TRadioButton;
    OpenDialog1: TOpenDialog;
    ADOConnection1: TADOConnection;
    ADOQuery1: TADOQuery;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ApdTapiDevice1TapiDTMF(CP: TObject; Digit: Char;
      ErrorCode: Longint);
    procedure ApdTapiDevice1TapiConnect(Sender: TObject);
    procedure ApdTapiDevice1TapiWaveNotify(CP: TObject; Msg:
TWaveMessage);
    procedure ApdReceiveFax1FaxFinish(CP: TObject; ErrorCode:
Integer);
    procedure ApdSendFax1FaxFinish(CP: TObject; ErrorCode: Integer);
    procedure ApdReceiveFax1FaxError(CP: TObject; ErrorCode:
Integer);
    procedure btnConnectClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
  StateIdle             = 0;
  StateGreeting         = 1;
  StateReceiveFax       = 2;
  StateEndCall          = 3;
```

```
  StateGetUser          = 4;
  StateGetPass          = 5;
  StateGetNum           = 6;
  StateSendFax          = 7;
  StateGetMobile        = 8;
  StateTelconfirm       = 9;
const
  DefaultPort = 7676;

var
  User,Pass,Telnumber,MobileNum,MobileTX : string;
  CurrentState,missuser,misspass,send1_receive2,countTime,Telnum4or9
: Integer;
  Form1: TForm1;
  WaveFileDir,FaxFileDir,FaxLockerDir,FaxFile,accPath,filename2 :
String;
  sr: TSearchRec;
  ClientForm: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  ApdTapiDevice1.SelectDevice;
  ApdTapiDevice1.EnableVoice := True;
end;

procedure FaxFTP;
var
    filename,CreateDir : string;
    OnlyFileName  : string;
    TxTextFile    : TextFile;
begin
          CreateDir := FaxFileDir+user+'\';
          CreateDirectory(pchar(CreateDir), nil);
RenameFile(FaxFileDir+'FAX0001.APF',FaxFileDir+user+Telnumber+'.APF'
);
          filename:= FaxFileDir+user+Telnumber+'.APF';
          filename2:=FaxFileDir+user+'\'+user+Telnumber+'.APF';
          MoveFile(PChar(filename), PChar(filename2));
          //Create temporary text file with acc inside
          OnlyFileName := User+MobileTX+MobileNum+Telnumber;
          AssignFile(TxTextFile,accPath);
          ReWrite(TxTextFile);
          WriteLn(TxTextFile,OnlyFileName);
          CloseFile(TxTextFile);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Server Status is
Idle...');
 counttime := 0;
 Timer1.Enabled := false;
 accPath := GetCurrentDir+'Temp.txt';
 WaveFileDir := GetCurrentDir+'\Wave\';
```

```
FaxFileDir := GetCurrentDir+'\FAX\';
FaxLockerDir := 'C:\WINDOWS\Desktop\SFFS\Server\FaxLocker\' ;
Apdreceivefax1.DestinationDir := FaxFileDir;
if not ApdTapiDevice1.EnableVoice then
   ApdTapiDevice1.EnableVoice := True;
 if ApdTapiDevice1.EnableVoice then begin
   ApdTapiDevice1.AutoAnswer;
 end else
   MessageDlg('The Selected device does not support Voice
Extensions.', mtInformation, [mbOk], 0);
end;

procedure TForm1.ApdTapiDevice1TapiDTMF(CP: TObject; Digit: Char;
  ErrorCode: Longint);
var
    i : integer ;
begin
    case CurrentState of
    StateGreeting :
      begin
          if telnum4.Checked then Telnum4or9 := 4;
          if telnum9.Checked then Telnum4or9 := 9;
          case Digit of
          '1':
            begin
              User := '';
              Pass := '';
              Telnumber := '';
              send1_receive2 := 1;
              CurrentState := StateGetUser;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'User.wav');
            end;


            '2':
            begin
              User := '';
              Pass := '';
              Telnumber := '';
              send1_receive2 := 2;
              CurrentState := StateGetUser;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'User.wav');
            end;

          '*':
            begin
              ApdTapiDevice1.InterruptWave := true;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Goodbye.wav');
              CurrentState := StateEndCall;
            end;

          '#':
            begin
              CurrentState := StateGreeting;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
            end;
```

```
          else
            begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Play'+Digit+'.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
        ApdTapiDevice1.PlayWaveFile(WaveFileDir+'misschoice.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
            end;
          end;
      end; {end greeting}
    StateGetUser :
      begin
        case Digit of
          '*':
            begin
              CurrentState := StateGreeting;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
            end;
          '#':
            begin
              if User = '' then
                begin
                CurrentState := StateGetUser;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'user_retry.wav');
                end
              else
                  begin
                      if Length(User) <> 4 then
                        begin
                        User := '';
                        missuser := missuser+1;
                        if missuser < 3 then
                            begin
                            CurrentState := StateGetUser;
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'user_retry.wav');
                            end {end if}
                            else
                            begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'3times_miss.wav');
                            CurrentState := StateEndCall;
                            end; {end else}
                        end {emd if length(User)}
                    else
                    begin
                         for i := 1 to Length(User) do begin
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'play'+Copy(User,
i, 1)+'.wav');
                     while ApdTapiDevice1.WaveState = wsPlaying do
                          begin
                         ApdTapiDevice1.InterruptWave :=
false;
                          Application.ProcessMessages;
                          end;
```

```
                                    ApdTapiDevice1.InterruptWave :=
true;
                        end;

                                {DATABASE INT Module}
                                with ADOQuery1 do
                                  begin
                                  Close;
                                  With SQL do begin
                                    Clear;
                                    Add('SELECT *');
                                    Add('FROM account');
                                    Add('WHERE username=(:Username)');
                                    Parameters[0].Value := user;
                                    Prepared:=True;
                                  end;
                                  Active:=True;
                                  Open;//alternatively can use ExecSQL
                                  if RecordCount=1 then
                                  begin
                                  CurrentState := StateGetPass;

      ApdTapiDevice1.PlayWaveFile(WaveFileDir+'pass.wav');
                                end
                              else
                                begin
                                  missuser := missuser+1;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'invaliduser.wav');
                          while ApdTapiDevice1.WaveState = wsPlaying do
                                  Application.ProcessMessages;
                                  User := '';
                                  if missuser < 3 then
                                    begin
                                    CurrentState := StateGetUser;
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'user_retry.wav');
                                    end {end if}
                                    else  begin
                                    ApdTapiDevice1.InterruptWave :=
true;

      ApdTapiDevice1.PlayWaveFile(WaveFileDir+'3times_miss.wav');
                                    CurrentState := StateEndCall;
                                    end; {end else}
                                end; {emd else user}
                            end;  {end for}
                      end; {end else length(User)}
                      end; {ADO}
            end;  {end #}
          else
            begin
              User := User+Digit;
              end;
    end;  {end case Digit}
  end;   {end GetUser}

  StateGetPass :
  begin
```

```
        case Digit of
          '*':
            begin
              CurrentState := StateGetUser;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'User.wav');
            end;
          '#':
            begin
              if Pass = '' then
                begin
                CurrentState := StateGetPass;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Pass_retry.wav');
                end
              else
                  begin
                      if Length(Pass) <> 4 then
                        begin
                        Pass := '';
                        misspass := misspass + 1;
                         if misspass < 3 then
                            begin
                            CurrentState := StateGetPass;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Pass_retry.wav');
                            end {end if}
                            else  begin
                            ApdTapiDevice1.InterruptWave := true;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'3times_miss.wav');
                            CurrentState := StateEndCall;
                            end; {end else}
                        end
                      else
                        begin
                          for i := 1 to Length(Pass) do begin
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'play'+Copy(Pass,
i, 1)+'.wav');
                         while ApdTapiDevice1.WaveState = wsPlaying do
                                  begin
                                  ApdTapiDevice1.InterruptWave := false;
                                      Application.ProcessMessages;
                                      end;
                                  ApdTapiDevice1.InterruptWave := true;
                                  end;

                        {DATABASE PASS INT MODULE}
                              with ADOQuery1 do
                              begin
                                Close;
                                With SQL do
                                  begin
                                  Clear;
                                  Add('SELECT *');
                                  Add('FROM account');
          Add('WHERE username=(:Username) and
password=(:Password)');
                                  Parameters[0].Value := user;
```

```
                                    Parameters[1].Value := pass;
                                    Prepared:=True;
                                    end;
                                  Active:=True;
                                  open;
                                   if RecordCount=1 then
                                   begin
                            MobileTX :=
FieldByName('TelNumber').AsString;
                            mmStatus.Lines.Insert(0,DateTimeToStr(Now)+'
:User''s MobileNumber is : '+MobileTX);
                                        case send1_receive2 of
                                          1 :
                                              begin
                                             CurrentState :=
StateGetNum;
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum.wav');
                                              end;
                                          2 :
                                              begin
                                            CurrentState :=
StateSendFax;
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Sendfax.wav');
                                              end;
                                    end; {end case}
                                end
                               else
                                  begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'invalidpass.wav');
                                  Pass := '';
                                  misspass := misspass + 1 ;
                                  if misspass < 3 then
                                    begin
                                    CurrentState := StateGetPass;
          ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Pass_retry.wav');
                                    end {end if}
                                    else  begin
                                    ApdTapiDevice1.InterruptWave :=
true;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'3times_miss.wav');
                                    CurrentState := StateEndCall;
                                    end; {end else}
                                  end;
                            end;
                            end;
                     end;
            end;  {end #}
          else
            begin
              Pass := Pass+Digit;
            end;
        end;  {end case Digit}
      end;   {end GetPass}

  StateGetNum :
      begin
```

```
        case Digit of
          '*':
            begin
              CurrentState := StateGreeting;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
            end;
          '#':
            begin
              if Telnumber = '' then
                begin
                CurrentState := StateGetNum;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum_retry.wav');
                end
              else
                  begin
                      if Length(Telnumber) <> Telnum4or9  then
                        begin
                          Telnumber := '';
                          CurrentState := StateGetNum;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum_retry.wav');
                        end
                      else
                        begin
         ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Repeattel.wav');
                        while ApdTapiDevice1.WaveState = wsPlaying
do
                          Application.ProcessMessages;
                          for i := 1 to Length(Telnumber) do
                            begin
      ApdTapiDevice1.PlayWaveFile(WaveFileDir+'play'+Copy(Telnumber,
i, 1)+'.wav');
                              while ApdTapiDevice1.WaveState =
wsPlaying do
                                begin
                                 ApdTapiDevice1.InterruptWave :=
false;
                                Application.ProcessMessages;
                                end;
                              end;
                            ApdTapiDevice1.InterruptWave := true;
      ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum_confirm.wav');
{¡´ confirm}
                          CurrentState := StateTelconfirm;
                        end; {end else length(Telnumber)}
                  end;{end else Telnumber}
            end;  {end #}
          else
            begin
              Telnumber := Telnumber+Digit;
              end;
        end;  {end case Digit}
      end;    {end GetNum}

   StateTelconfirm :
    begin
        case Digit of
```

```
          '*':
            begin
               Telnumber := '';
               CurrentState := StateGetNum;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum.wav');
            end;
          '1':
            begin
              CurrentState := StateGetMobile;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Mobile.wav');
            end;
           else
            begin
              ApdTapiDevice1.InterruptWave := false;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Play'+Digit+'.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
        ApdTapiDevice1.PlayWaveFile(WaveFileDir+'misschoice.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
              ApdTapiDevice1.InterruptWave := true;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum_confirm.wav');
            end;
        end;
      end;

   StateGetMobile :
      begin
        case Digit of
          '*':
            begin
              CurrentState := StateGetNum;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum.wav');
            end;
          '#':
            begin
              if Mobilenum = '' then
                begin
                CurrentState := StateGetMobile;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Mobile_retry.wav');
                end
              else
                  begin
                      if Length(Mobilenum) <> 9  then
                        begin
                          MobileNum := '';
                          CurrentState := StateGetMobile;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Mobile_retry.wav');
                        end
                      else
                        begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Repeatmobile.wav');
```

```
                        while ApdTapiDevice1.WaveState = wsPlaying
do
                          Application.ProcessMessages;
                          for i := 1 to Length(Mobilenum) do
                            begin
ApdTapiDevice1.PlayWaveFile(WaveFileDir+'play'+Copy(Mobilenum, i,
1)+'.wav');
                       while ApdTapiDevice1.WaveState = wsPlaying do
                             begin
                              ApdTapiDevice1.InterruptWave :=
false;
                             Application.ProcessMessages;
                             end;
                           end;
                          ApdTapiDevice1.InterruptWave := true;
ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Mobile_confirm.wav');
                          CurrentState := StateReceiveFax;
                        end;  {end else length(Telnumber)}
                      end; {end else MobileNum}
            end;  {end #}
          else
            begin
              MobileNum := MobileNum+Digit;
              end;
        end;  {end case Digit}
      end;   {end GetMobileNum}


    StateReceiveFax :
      begin
        case Digit of
          '*':
            begin
               Mobilenum := '';
               CurrentState := StateGetMobile;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Mobile.wav');
            end;
          '1':
            begin
              CurrentState := StateReceiveFax;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'fsndprmt.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
                Application.ProcessMessages;
                timer1.Enabled := false;
              ApdTapiDevice1.AutomatedVoiceToComms;
              ApdReceiveFax1.FaxFile := FaxFile;
              DeleteFile(FaxFileDir+'FAX0001.APF');
              ApdReceiveFax1.StartManualReceive(True);
            end;
           else
            begin
              ApdTapiDevice1.InterruptWave := false;
ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Play'+Digit+'.wav');
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
ApdTapiDevice1.PlayWaveFile(WaveFileDir+'misschoice.wav');
```

```
              while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Telnum_confirm.wav');
            end;
        end;
      end;

    StateSendFax :
      begin
        case Digit of
          '*':
            begin
              CurrentState := StateGreeting;
              ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
            end;
          '1':
            begin
         if FindFirst(FaxLockerDir+User+'\*.*',faArchive,sr) = 0
then
                begin
                     FaxFile := FaxLockerDir+User+'\'+sr.Name;
                     CurrentState := StateSendFax;

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'frcvprmt.wav');
                     while ApdTapiDevice1.WaveState = wsPlaying do
                         Application.ProcessMessages;
                         timer1.Enabled := false;
                     ApdTapiDevice1.AutomatedVoiceToComms;
                     ApdSendFAX1.FaxFile := Faxfile;
                     ApdSendFax1.StartManualTransmit;
                end
                else
                  begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'nofile.wav');
               CurrentState := StateGreeting;
               while ApdTapiDevice1.WaveState = wsPlaying do
               Application.ProcessMessages;
               ApdTapiDevice1.PlayWaveFile(WaveFileDir+'1or2.wav');
                    end;
            end;
        else
        begin

ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Play'+Digit+'.wav');
        while ApdTapiDevice1.WaveState = wsPlaying do
        Application.ProcessMessages;
        ApdTapiDevice1.PlayWaveFile(WaveFileDir+'misschoice.wav');
        while ApdTapiDevice1.WaveState = wsPlaying do
        Application.ProcessMessages;
        ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Sendfax.wav');
        end;
        end;
      end;
  end;
end;
procedure TForm1.ApdTapiDevice1TapiConnect(Sender: TObject);
begin
```

```
  {Play Greeting}
  User            := '';
  Pass            := '';
  Telnumber       := '';
  MobileNum       := '';
  missuser        := 0;
  misspass        := 0;
  send1_receive2  := 0 ;
  FaxFile         := '';
  ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Greeting1or2.wav');
  CurrentState := StateGreeting;
end;
procedure TForm1.ApdTapiDevice1TapiWaveNotify(CP: TObject;
  Msg: TWaveMessage);
begin
  case CurrentState of
    StateIdle        : Edit1.Text := 'StateIdle';
    StateGreeting    : Edit1.Text := 'StateGreeting';
    StateReceiveFax  : Edit1.Text := 'StateReceiveFax';
    StateEndCall     : Edit1.Text := 'StateEndCall';
    StateGetUser     : Edit1.Text := 'StateGettingUserName';
    StateGetPass     : Edit1.Text := 'StateGettingPassword';
    StateGetNum      : Edit1.Text := 'StateGettingNumber';
    StateTelconfirm  : Edit1.Text := 'StateTelNumberconfirm';
    StateGetMobile   : Edit1.Text := 'StateGettingMobileNumber';
    StateSendFax     : Edit1.Text := 'StateSendFax';

  end;
  if Msg = waPlayOpen then
    begin
      timer1.Enabled := false;
      Edit2.Text := 'Playing Wave:
'+ExtractFileName(ApdTapiDevice1.WaveFileName)
    end
    else if Msg = waPlayDone then
      begin
      timer1.Enabled := true;
      Edit2.Text := 'Wave Device Idle...';
      end;
  case CurrentState of
    StateEndCall:
      begin
        Edit1.Text := 'StateEndCall';
        if Msg = waPlayDone then begin
          timer1.Enabled := false;
          ApdTapiDevice1.CancelCall;
          if not ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.EnableVoice := True;
          if ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.AutoAnswer;
          Edit1.Text := 'StateIdle';
        end;
      end;
  end;
end;

procedure TForm1.ApdReceiveFax1FaxFinish(CP: TObject; ErrorCode:
Integer);
```

```
var
  tempFile,accFile : TFileStream;
  FileTestTextName : string;
  sCommand,s1Command: string;
  s2Command : string;
begin
    if ErrorCode = 0 then
      begin
          ApdTapiDevice1.InterruptWave := true;
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Receive Fax
Complete');
          ApdTapiDevice1.CancelCall;
          if not ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.EnableVoice := True;
          if ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.AutoAnswer;
          Edit1.Text := 'StateIdle';
          FaxFTP;
          {TCP SENDING}
          if not TCPClient.Connected then ShowMessage('Please
connect first');
          TCPClient.WriteLn('ReadyToSend');
          sCommand := TCPClient.ReadLn;
          if sCommand = 'OKToReceive' then
          begin
              accFile := TFileStream.Create(accPath,fmOpenRead);
              TCPClient.WriteInteger(accFile.Size);
              TCPClient.OpenWriteBuffer;
              TCPClient.WriteStream(accFile);
              TCPClient.CloseWriteBuffer;
              FreeAndNil(accFile);
              s1Command := TCPClient.ReadLn;
              if s1Command = 'NeedData' then
              begin
                   FileTestTextName := filename2;
                   tempFile :=
TFileStream.Create(FileTestTextName,fmOpenRead);
                   mmStatus.Lines.Insert(0,DateTimeToStr(Now)+'
:Sending File to Server...');
                   TCPClient.WriteInteger(tempFile.Size);
                   TCPClient.OpenWriteBuffer;
                   TCPClient.WriteStream(tempFile);
                   TCPClient.CloseWriteBuffer;
                   FreeAndNil(tempFile);
              end;
              s2Command := TCPClient.ReadLn;
              if s2Command = 'DONE' then
              begin
                  mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Send
FaxFile Complete');
                  TCPClient.WriteLn('Bye');
              end;

          end;  {TCP Finish}
      end    {End Fax}
      else
        begin
```

```
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t
Receive Fax');
          ApdTapiDevice1.CancelCall;
          if not ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.EnableVoice := True;
          if ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.AutoAnswer;
          Edit1.Text := 'StateIdle';
        end
end;

procedure TForm1.ApdSendFax1FaxFinish(CP: TObject; ErrorCode:
Integer);
begin
     if ErrorCode = 0 then
      begin
        mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Send Fax
Complete');
        ApdTapiDevice1.CancelCall;
        apdcomport1.Open := false;
        if not ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.EnableVoice := True;
        if ApdTapiDevice1.EnableVoice then
              ApdTapiDevice1.AutoAnswer;
              Edit1.Text := 'StateIdle';
      end
      else
        begin
          mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t Sent
Fax');
          ApdTapiDevice1.CancelCall;
          if not ApdTapiDevice1.EnableVoice then
                ApdTapiDevice1.EnableVoice := True;
          if ApdTapiDevice1.EnableVoice then
                ApdTapiDevice1.AutoAnswer;
          Edit1.Text := 'StateIdle';
        end
end;

procedure TForm1.ApdReceiveFax1FaxError(CP: TObject; ErrorCode:
Integer);
begin
    mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Can''t Receive
Fax');
          ApdTapiDevice1.CancelCall;
          if not ApdTapiDevice1.EnableVoice then
                ApdTapiDevice1.EnableVoice := True;
          if ApdTapiDevice1.EnableVoice then
                ApdTapiDevice1.AutoAnswer;
          Edit1.Text := 'StateIdle';
end;

procedure TForm1.btnConnectClick(Sender: TObject);
begin
   try
    TCPClient.Host := edtServer.Text;
    TCPClient.Port := DefaultPort;
    TCPClient.Connect;
```

```
    mmStatus.Lines.Insert(0,DateTimeToStr(Now)+' :Connect To
Server');
  except
    on E: Exception do MessageDlg ('Error while connecting to
ScreenThiefSERVER:'+#13+E.Message, mtError, [mbOk], 0);
  end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  if TCPClient.Connected then TCPClient.Disconnect;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
   timer1.Enabled := false;
   ApdTapiDevice1.InterruptWave := true;
   ApdTapiDevice1.PlayWaveFile(WaveFileDir+'Goodbye.wav');
        while ApdTapiDevice1.WaveState = wsPlaying do
              Application.ProcessMessages;
              timer1.Enabled := false;
              ApdTapiDevice1.CancelCall;
        if not ApdTapiDevice1.EnableVoice then
               ApdTapiDevice1.EnableVoice := True;
        if ApdTapiDevice1.EnableVoice then
               ApdTapiDevice1.AutoAnswer;
               Edit1.Text := 'StateIdle';
end;
end.
```

หมายเหตุ: Program SFFS, Source Code และComponent TurboPower Async Professional Version 4.06 ได้รวบรวมเป็นแผ่น CD ถ้าผู้ใดสนใจ ติดต่อ ดร.พิพัฒน์ พรหมมี ห้อง T-108A หรือ ติดต่อทีมพัฒนาได้ที่ E-mail:worachat_22@yahoo.com,werasit@hotmail.com,legendary_midfielder@hotmail.com

หนังสืออ้างอิง

[1] จิราภรณ์ จุติสุขสันติ์ และ ฐิตินันท์ห ลิมปาภินันท์ , "บริการสั่งโทรสารผ่านเครือข่ายอินเทอร์เน็ต = Fax server on Internet" , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,2539

[2] เกียรติ อินทุวรรณรัตน์ ธิติ พัฒนไพจิตต์กุล , "การส่งข้อมูลจากจดหมายอิเล็กทรอนิกส์เพื่อแสดงบนเครื่องโทรสาร = Data transferring from E-mail to facsimile (Fax)" , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2544

[3] Kenneth R. McConnell , Dennis Bodson , Richard Schaphorst , "FAX : digital facsimile technology and applications Edition 2nd ed" , Imprint Boston : Artech House, 1992

[4] สัจจะ จรัสรุ่งรวีวร, เริ่มต้นอย่างมืออาชีพด้วย Delphi 7 ฉบับสมบูรณ์.นนทบุรี:อินโฟเพรส, 2546

[5] Kenneth R. McConnell , Dennis Bodson , Richard Schaphorst , "FAX : digital facsimile technology and applications Edition 2nd ed" , Imprint Boston : Artech House, 1992

[6] http://www.thaidelphi.com

[7] http://www.delphibasics.co.uk/ByFunction.asp?Main=PgmStruct&Sub=Data

[8] http://docs.projectindy.org/online/frames.html?frmname=topic&frmfile=IdTCPServer_pas.html

[9] Async Professional a comprehensive communications toolkit for Borland Delphi , Http://sourceforge.net/projects/tpapro/

กิตติกรรมประกาศ

ปริญญานิพนธ์ระบบโทรสารแบบจัดเก็บและส่งต่อ(THE STORE-AND-FORWARD FAX SYSTEM) สำเร็จลุล่วงไปด้วยดีทั้งนี้ ด้วยความอนุเคราะห์ช่วยเหลือจากผู้มีพระคุณหลายท่านโดยเฉพาะท่านอาจารย์ที่ปรึกษา ดร. พิพัฒน์ พรหมมี ที่กรุณาช่วยเหลือให้คำแนะนำในเรื่องเนื้อหา การดำเนินงานจนถึงช่วยแก้ปัญหาต่างๆให้สำเร็จลุล่วงด้วยดี รวมทั้งห้อง ผศ.ดร. สุทธิชัย นพนาคีพงษ์ T-302 ที่เอื้อเฟื้อคู่สายโทรศัพท์ภายใน เพื่อใช้ในการทดลองและส่งชิ้นงาน

นาย วรฉัตร สิริยานนท์
นาย วิรุจ สุวรรณปราโมทย์
นาย วีรสิษฐ์ กิติวรรณกุล