

รหัสโครงการ SUT7-709-45-12-27



รายงานการวิจัย

เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิตอลผ่านโครงข่ายอินเทอร์เน็ต (Digital Automatic Answering Machine via Internet Networks)

ได้รับทุนอุดหนุนการวิจัยจาก
มหาวิทยาลัยเทคโนโลยีสุรนารี

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

รหัสโครงการ SUT7-709-45-12-27



รายงานการวิจัย

เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านโครงข่ายอินเทอร์เน็ต (Digital Automatic Answering Machine via Internet Networks)

ผู้วิจัย

หัวหน้าโครงการ

อาจารย์เรืออากาศเอกประโยชน์ คำสวัสดิ์

สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พ.ศ. 2545

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

มกราคม 2546

กิตติกรรมประกาศ

ขอขอบคุณ มหาวิทยาลัยเทคโนโลยีสุรนารี ที่ให้ทุนสนับสนุนโครงการวิจัย รวมทั้ง
รศ.น.ท.ดร.สราวุฒิ สุจิตจร และ อ.สมศักดิ์ วาณิชอนันต์ชัย ที่ได้ให้คำแนะนำและมีส่วนช่วย
เหลือในการทำงานวิจัยนี้

บทคัดย่อ

รายงานนี้นำเสนอเครื่องตอบรับโทรศัพท์แบบดิจิทัลที่สามารถทำงานได้บน 3 โครงข่ายหลัก คือ โครงข่ายระบบโทรศัพท์พื้นฐาน (Public Switched Telephone Networks) โครงข่ายโทรศัพท์เคลื่อนที่ (Mobile Phone Networks) และ โครงข่ายอินเทอร์เน็ต (Internet Networks) ผู้ใช้งานสามารถโทรศัพท์เข้าไปทำการฝากหรือตรวจสอบข้อความโดยใช้โทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่เนื่องจากเครื่องตอบรับโทรศัพท์แบบดิจิทัลนี้ถูกออกแบบให้มีการเชื่อมต่อกับเว็บเซิร์ฟเวอร์ส่วนบุคคล (Personal Web Server) ที่สนับสนุนโปรโตคอล HTTP (Hypertext Transfer Protocol) ซึ่งถูกเก็บซ่อนอยู่ในโปรโตคอล SSL (Secure Socket Layer) หรือที่เรียกว่า HTTPS ทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตมีความปลอดภัยสูงมาก และจากการนำเทคโนโลยีจาวาเซิร์ฟเล็ต (Java Servlet Technology) มาใช้ในการเขียนโปรแกรม ทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตได้รับข้อความล่าสุดตลอดเวลาการทำงานของเครื่องตอบรับโทรศัพท์เป็นแบบอัตโนมัติซึ่งใช้ไมโครคอนโทรลเลอร์ควบคุม ผลการทดสอบการใช้งานพบว่าเครื่องตอบรับโทรศัพท์ดังกล่าวสามารถทำงานได้ตามที่ออกแบบไว้อย่างถูกต้อง

ABSTRACT

This report presents a digital answering machine that can function on 3 networks, i.e. Public Switched Telephone Networks, Mobile Phone Networks, and Internet Network. The message of an incoming call via a telephone or mobile one can be left with the machine. Since this machine can be connected to the personal web server that supports the Hypertext Transfer Protocol (HTTP) encapsulated in the SSL (Secure Socket Layer) protocol, called shortly as HTTPS. HTTPS provides high security when it interrogates and transfers voice message via Internet Network. Determining the up-to-date voice message is possible using Java Servlet Technology. The automatic function of the proposed machine is supervised by a microcontroller and the tested results show that it functions properly.

สารบัญ

	หน้า
กิตติกรรมประกาศ	ก
บทคัดย่อ	ข
ABSTRACT	ค
สารบัญ	ง
สารบัญตาราง	ฉ
สารบัญภาพ	ช
บทที่ 1 บทนำ	
1.1 กล่าวนำ	1
1.2 ความสำคัญและที่มาของปัญหาการวิจัย	1
1.3 วัตถุประสงค์ของการวิจัย	2
1.4 ขอบเขตของการวิจัย	2
1.5 ข้อตกลงเบื้องต้น	3
1.6 ประโยชน์ที่ได้รับจากการวิจัย	3
1.7 การจัดรูปแบบของรายงานการวิจัย	3
บทที่ 2 ทฤษฎีที่เกี่ยวข้องและวิธีดำเนินการวิจัย	
2.1 กล่าวนำ	4
2.2 เทคโนโลยีจาวาเซิร์ฟเล็ต	4
2.3 Secure Socket Layer	6
2.4 ไมโครคอนโทรลเลอร์ AVR	7
2.5 การบันทึกข้อมูลเสียงแบบดิจิทัล	10
2.5 วิธีดำเนินการวิจัย	12
2.7 อุปกรณ์ที่จำเป็นในการวิจัย	12
2.8 แผนการดำเนินงานตลอดโครงการวิจัย	13
บทที่ 3 หลักการทำงานและการออกแบบระบบ	
3.1 กล่าวนำ	14
3.2 การทำงานและการออกแบบฮาร์ดแวร์	14
3.3 การทำงานและการออกแบบซอฟต์แวร์	16

สารบัญ (ต่อ)

	หน้า
บทที่ 4 ผลการทดลองและการทดสอบการใช้งาน	
4.1 กล่าวนำ	19
4.2 การบันทึกข้อความเสี่ยงแบบดิจิทัลบนไอซีหน่วยความจำ	19
4.3 การฝากข้อความผ่านโครงข่ายโทรศัพท์	19
4.4 การตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์	20
4.5 การตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต	21
บทที่ 5 บทสรุป	
5.1 สรุปผลการวิจัย	24
5.2 ข้อเสนอแนะ	24
5.3 แนวทางในการพัฒนาต่อไปในอนาคต	24
บรรณานุกรม	25
ภาคผนวก	
ภาคผนวก ก โปรแกรมควบคุมระบบการทำงาน	26
ภาคผนวก ข บทความเผยแพร่งานประชุมวิชาการ EECON-25	94
ภาคผนวก ค รูปภาพวงจรอิเล็กทรอนิกส์และอุปกรณ์การทดลองและวิจัย	100
ประวัติผู้วิจัย	103

สารบัญตาราง

	หน้า
ตารางที่ 2.1 แผนการดำเนินงานตลอดโครงการวิจัย	13
ตารางที่ 4.1 เวลาเฉลี่ยที่ใช้ในการฝากข้อความผ่านโครงข่ายโทรศัพท์	20
ตารางที่ 4.2 เวลาเฉลี่ยที่ใช้ในการตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์	20
ตารางที่ 4.3 เวลาเฉลี่ยที่ใช้ในการฝากและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์	21
ตารางที่ 4.4 ผลการตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต	23

สารบัญภาพ

	หน้า
รูปที่ 1.1 การใช้งานเครื่องตอบรับโทรศัพท์แบบดิจิทัลผ่านอินเทอร์เน็ต	2
รูปที่ 2.1 ขั้นตอนการทำงานของ CGI	5
รูปที่ 2.2 ขั้นตอนการทำงานของ Servlet	5
รูปที่ 2.3 SSL ในโปรโตคอลสแตก TCP/IP	6
รูปที่ 2.4 สถาปัตยกรรมแบบ RISC ของ AT90S8535	8
รูปที่ 2.5 โครงสร้างของหน่วยความจำของ AT90S8535	9
รูปที่ 2.6 โครงสร้างรีจิสเตอร์ใช้งานทั่วไป	10
รูปที่ 2.7 ความสัมพันธ์ระหว่างค่าความต้านทาน Rose กับอัตราการใช้ตัวอย่าง	11
รูปที่ 3.1 แผนภาพบล็อกของระบบ	14
รูปที่ 3.2 วงจรถอดรหัสสัญญาณ DTMF	15
รูปที่ 3.3 วงจรเปิดและบันทึกข้อความเสียงพูด	15
รูปที่ 3.4 โปรแกรมการเชื่อมต่อผ่านพอร์ตอนุกรม	16
รูปที่ 3.5 โปรแกรมบันทึกข้อความบนเครื่องเว็บเซิร์ฟเวอร์	17
รูปที่ 4.1 การล็อกอินเข้าโฮมเพจเพื่อตรวจสอบข้อความ	21
รูปที่ 4.2 การตรวจสอบข้อความโดยใช้ Internet Explorer 6.0	22
รูปที่ 4.3 การฟังข้อความโดยใช้โปรแกรม Windows Media Player 6.0	23
รูปที่ ค.1 วงจรบันทึกข้อความเสียงพูด	101
รูปที่ ค.2 วงจรบันทึกข้อความและเปิดข้อความเสียงพูดผ่านโครงข่ายโทรศัพท์	101
รูปที่ ค.3 วงจรตัดต่อระบบโทรศัพท์และถอดรหัสสัญญาณ DTMF	102
รูปที่ ค.4 การอินเตอร์เฟสไมโครคอนโทรลเลอร์กับวงจรอิเล็กทรอนิกส์	102

บทที่ 1

บทนำ

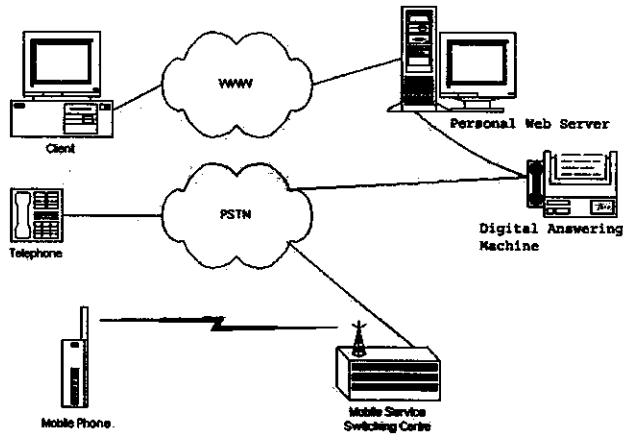
1.1 กล่าวนำ

ในยุคของระบบเทคโนโลยีปัจจุบัน เทคโนโลยีทางด้านอิเล็กทรอนิกส์และคอมพิวเตอร์สามารถนำมาประยุกต์ใช้งานได้อย่างกว้างขวางมากมาย เช่น ทางด้านการแพทย์ ระบบอุตสาหกรรม ระบบการสื่อสารข้อมูลทั้งภาพและเสียง โครงข่ายโทรศัพท์ โครงข่ายคอมพิวเตอร์ เป็นต้น เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ตที่นำเสนอในรายงานการวิจัยนี้ เป็นอีกตัวอย่างหนึ่งของการประยุกต์ใช้งานเทคโนโลยีที่ได้กล่าวถึงข้างต้น

1.2 ความสำคัญและที่มาของปัญหาการวิจัย

ปัจจุบันการติดต่อสื่อสารโดยใช้โทรศัพท์พื้นฐานและโทรศัพท์เคลื่อนที่มีบทบาทสำคัญมาก การพลาดการติดต่อทางโทรศัพท์เพียงครั้งเดียวอาจนำมาซึ่งความสูญเสียเป็นมูลค่ามหาศาล โดยเฉพาะการติดต่อทางธุรกิจ ดังนั้นในแต่ละสำนักงานจะมีเครื่องตอบรับโทรศัพท์ติดตั้งอยู่ด้วยเสมอ เครื่องตอบรับโทรศัพท์แบบดิจิทัลนั้นจะติดต่อสื่อสารกันได้เฉพาะโครงข่ายโทรศัพท์ด้วยกันเท่านั้น จึงเกิดปัญหาในกรณีที่จำเป็นต้องเดินทางไปไกล ๆ เช่นเมื่อเดินทางไปต่างประเทศ หากต้องการจะทราบว่า มีบุคคลใดโทรศัพท์เข้ามายังที่ทำงานและฝากข้อความไว้ ต้องโทรศัพท์ทางไกลระหว่างประเทศกลับมาที่เครื่องตอบรับโทรศัพท์เพื่อตรวจสอบข้อความซึ่งต้องสิ้นเปลืองค่าใช้จ่ายสูง ปัญหานี้อาจแก้ไขได้หากสามารถทำการตรวจสอบข้อความผ่านเว็บเบราว์เซอร์ตามอินเทอร์เน็ตคาเฟ่ที่มีอยู่ทั่วไป โครงการวิจัยนี้จึงให้ความสนใจไปที่การออกแบบเครื่องตอบรับโทรศัพท์แบบดิจิทัลที่สามารถใช้งานได้บน 3 โครงข่ายหลักเพื่อให้สามารถติดต่อสื่อสารกับบุคคลอื่น ๆ ได้ตลอดเวลาโดยสิ้นเปลืองค่าใช้จ่ายน้อยมาก

การประยุกต์ใช้โครงข่ายที่กล่าวถึงแสดงในรูปที่ 1 โดยเครื่องตอบรับโทรศัพท์แบบดิจิทัลจะเชื่อมต่อกับเครื่องคอมพิวเตอร์ที่ทำงานเป็นเว็บเซิร์ฟเวอร์ส่วนบุคคลที่สนับสนุนเทคโนโลยีจาваเซิร์ฟเล็ต ซึ่งสามารถใช้งานได้จริงโดยติดตั้งโปรแกรม PWS4.0 หรือ IIS5.0 พร้อมกับระบบปฏิบัติการ Windows 98 หรือ Windows 2000 ตามลำดับและใช้งานร่วมกับโปรแกรม Jakarta-Tomcat 3.2.3 และ Java 2 SDK 1.3 เครื่องตอบรับโทรศัพท์แบบดิจิทัลก็จะเสมือนเป็นตัวเชื่อมโยงโครงข่ายหลักทั้ง 3 โครงข่ายเข้าด้วยกัน



รูปที่ 1.1 การใช้งานเครื่องตอบรับโทรศัพท์แบบดิจิทัลผ่านอินเทอร์เน็ต

1.3 วัตถุประสงค์ของการวิจัย

1. เพื่อทำการออกแบบ สร้างและพัฒนาเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลโดยใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลหลัก
2. จัดทำโปรแกรมควบคุมเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลให้สามารถใช้งานผ่านโครงข่ายอินเทอร์เน็ต
3. จัดทำโปรแกรมการอินเตอร์เฟสระหว่างเครื่องคอมพิวเตอร์ที่เป็นเครื่องเซิร์ฟเวอร์กับไมโครคอนโทรลเลอร์โดยใช้หลักการจาวาเนทีฟอินเตอร์เฟส (Java Native Interface) เพื่ออัปเดตข้อความเสียงพูดขึ้นสู่โครงข่ายอินเทอร์เน็ต
4. จัดทำโปรแกรมการให้บริการตรวจสอบข้อความเสียงพูดผ่านโครงข่ายอินเทอร์เน็ตโดยใช้เทคโนโลยีจาวาเซิร์ฟเล็ต (Java Servlet Technology) และ SSL (Secure Socket Layer)
5. จัดทำโฮมเพจเพื่อให้บริการตรวจสอบข้อความเสียงพูดผ่านโครงข่ายอินเทอร์เน็ต

1.4 ขอบเขตของการวิจัย

ทำการออกแบบและสร้างเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลให้สามารถใช้งานได้บนโครงข่ายอินเทอร์เน็ต โดยที่การทำงานบนโครงข่ายอินเทอร์เน็ตนั้นจะครอบคลุมถึงเรื่องความปลอดภัยของข้อมูลและการใช้งานในชั้นพื้นฐานเท่านั้น ไม่มีการติดตั้งกำแพงกันไฟ (Fire wall) เพื่อรักษาความปลอดภัยแต่อย่างใด

1.5 ข้อตกลงเบื้องต้น

งานวิจัยนี้เป็นลักษณะของงานวิจัยขั้นพื้นฐาน โดยมุ่งเน้นไปที่มีการนำเอาเทคโนโลยีใหม่ ๆ ทางด้านการสื่อสาร โทรคมนาคมและคอมพิวเตอร์มาประยุกต์ใช้กับการดำเนินชีวิตในปัจจุบัน

1.6 ประโยชน์ที่ได้รับจากการวิจัย

เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลที่สามารถใช้งานผ่านโครงข่ายอินเทอร์เน็ตและรวมทั้งโปรแกรมคอมพิวเตอร์สำหรับควบคุมการทำงานเครื่องตอบรับโทรศัพท์ที่ได้กล่าวถึงข้างต้น

1.7 การจัดรูปแบบของรายงานการวิจัย

รายงานผลการวิจัยเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ต ได้แบ่งออกเป็น 5 บทประกอบด้วย

บทที่ 1 กล่าวถึงความสำคัญและที่มาของปัญหาการวิจัย วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ข้อตกลงเบื้องต้นและประโยชน์ที่ได้รับจากการวิจัย

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานเกี่ยวกับเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ต วิธีดำเนินการวิจัย อุปกรณ์ที่จำเป็นในการวิจัยและแผนการดำเนินงานตลอดโครงการวิจัย

บทที่ 3 กล่าวถึงหลักการทำงานและการออกแบบของระบบ โดยแยกอธิบายเป็นส่วนของฮาร์ดแวร์และส่วนของซอฟต์แวร์

บทที่ 4 เป็นผลการทดลองและการทดสอบการใช้งานจริงของเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ต

บทที่ 5 เป็นบทสรุป ข้อเสนอแนะและแนวทางในการพัฒนา เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ตต่อไปในอนาคต

บทที่ 2

ทฤษฎีที่เกี่ยวข้องและวิธีดำเนินการวิจัย

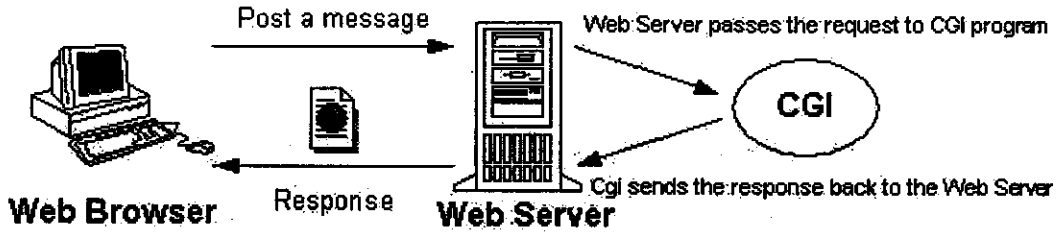
2.1 กล่าวนำ

เครื่องคอมพิวเตอร์ที่ใช้อินเทอร์เน็ตแบบดิจิทัลผ่านอินเทอร์เน็ต ได้รับการออกแบบโดยใช้การผสมผสานกันระหว่างเทคโนโลยีด้านการสื่อสารและโทรคมนาคม เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์ รวมทั้งใช้เทคนิคการเขียนโปรแกรมบนโครงข่ายคอมพิวเตอร์ ในบทนี้จะได้กล่าวถึงทฤษฎีที่เกี่ยวข้องเช่น เทคโนโลยีจาวาเซิร์ฟเล็ต Secure Socket Layer ไมโครคอนโทรลเลอร์ AVR การบันทึกข้อมูลเสียงแบบดิจิทัล นอกจากนี้ยังกล่าวถึงวิธีดำเนินการวิจัย อุปกรณ์ที่จำเป็นในการวิจัย และแผนการดำเนินงานตลอดโครงการวิจัย

2.2 เทคโนโลยีจาวาเซิร์ฟเล็ต

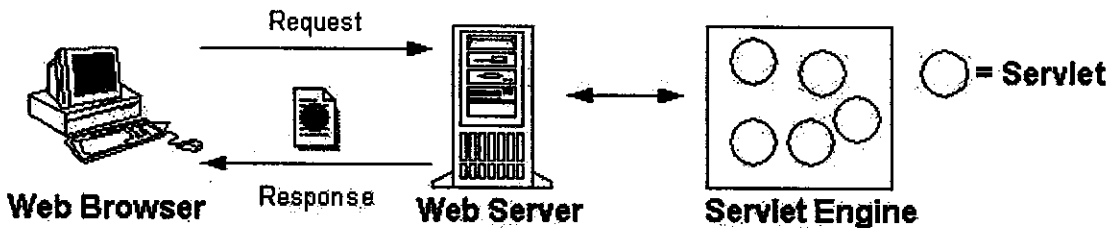
Java Servlet หรือเรียกสั้น ๆ ว่า Servlet เป็นโปรแกรมที่มีการทำงานบนฝั่งเซิร์ฟเวอร์ (Server Side Application) มีหลักการทำงานคล้ายกับ CGI (Common Gateway Interface) แต่ Servlet แตกมีข้อดีกว่า ประการแรกคือตัวภาษาที่ใช้ในการเขียนโปรแกรมเป็นภาษาจาวา (Java) ซึ่งใช้หลักการโปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งสามารถลดความซับซ้อนโครงสร้างของโปรแกรมรวมถึงอำนวยความสะดวกในการนำส่วนประกอบต่าง ๆ ของโปรแกรมที่เขียนไว้แล้วกลับมาใช้ได้ อีก นอกจากนี้จาวายังเป็นภาษาที่ไม่ขึ้นกับแพลตฟอร์ม (Platform Independent) ซึ่งช่วยให้สามารถทำงานหรือพัฒนาโปรแกรมได้ในทุกระบบปฏิบัติการ ประการที่สองเซิร์ฟเล็ตมีความเร็วที่เหนือกว่า CGI เพราะเซิร์ฟเล็ตใช้หลักการของเธรด (Thread) โดยจะทำการสร้าง 1 เธรดต่อหนึ่งการร้องขอ (Request) ที่มาจากไคลเอนต์ (Client) ในขณะที่ CGI จะทำการสร้าง 1 กระบวนการ (Process) ต่อหนึ่งการร้องขอ ซึ่งจะทำให้เปลืองทรัพยากรของระบบมากกว่า และในแต่ละกระบวนการมีการทำงานซ้ำกว่า

ในการที่จะสั่งให้เซิร์ฟเล็ตทำงาน ตัวเว็บเซิร์ฟเวอร์จะไม่สามารถส่งข้อมูลไปให้เซิร์ฟเล็ตได้โดยตรงเหมือนกับ CGI ดังนั้นตัวเว็บเซิร์ฟเวอร์ จะต้องมีส่วนเป็นเสมือนตัวห่อหุ้มเซิร์ฟเล็ตต่าง ๆ ไว้ โดยส่วนนี้เรียกว่า เซิร์ฟเล็ตเอนจิน (Servlet Engine) หรือเซิร์ฟเล็ตคอนเทนเนอร์ (Servlet Container) โดยทั่วไปเซิร์ฟเล็ตเอนจินจะเป็นส่วนที่มี Java Virtual Machine (JVM) อยู่ในตัวเองโดยเซิร์ฟเล็ตเอนจินนี้จะมีหน้าที่รับการร้องขอจากเว็บเบราว์เซอร์แล้วทำการเลือกตัวเซิร์ฟเล็ตขึ้นมาทำการประมวลผลภายใต้ JVM โดยผลลัพธ์ที่ได้จากการประมวลผลของเซิร์ฟเล็ตที่ถูกเลือกจะถูกส่งกลับไปยังเว็บ



รูปที่ 2.1 ขั้นตอนการทำงานของ CGI

เซิร์ฟเวอร์โดยเว็บเซิร์ฟเวอร์นี้จะส่งผล กลับไปยังเว็บเบราว์เซอร์ในฝั่งไคลเอนต์อีกทอดหนึ่ง การส่งผลลัพธ์ต่าง ๆ กลับไปให้ไคลเอนต์นั้น เซิร์ฟเวอร์จะต้องทำการเซต Header เพื่อกำหนดชนิดของ MIME (Multipurpose Internet Mail Extensions) ของสายข้อมูล (Stream) ที่ส่งออกไป โดย Header นี้จะเป็นตัวบอกไคลเอนต์ว่าสายข้อมูลที่กำลังจะได้รับเป็นสายข้อมูลชนิดไหน โดยเซิร์ฟเวอร์สามารถส่งสายข้อมูลออกไปได้ 2 แบบคือ สายข้อมูลเท็กซ์(text stream) และสายข้อมูลไบนารี (byte stream) ในกรณีของงานวิจัยนี้จะมีการส่งสายข้อมูลออกไปเป็นเท็กซ์หรือ HTML (Hyper Text Markup Language) นั่นคือจะมีการสร้างเอกสาร HTML ในขณะที่เซิร์ฟเวอร์ทำงาน การตรวจสอบข้อความผ่านอินเทอร์เน็ตจึงได้รับข้อความที่ทันสมัยตลอดเวลา



รูปที่ 2.2 ขั้นตอนการทำงานของ Servlet

จะเห็นว่าเซิร์ฟเวอร์เป็นโปรแกรมที่มีการประมวลผลบนเว็บเซิร์ฟเวอร์แล้วส่งข้อมูลผลลัพธ์ไปยังไคลเอนต์โดยเซิร์ฟเวอร์จะทำการสร้างเอกสาร HTML กลับไปให้ไคลเอนต์โดยยึดตามมาตรฐาน HTML และ/หรือ XML (Extensible Markup Language) เป็นหลักทำให้เว็บเซิร์ฟเวอร์ใช้งานจากผู้ใช้ได้หลากหลายเพราะไม่อิงกับเครื่องผู้ใช้งาน และเครื่องไคลเอนต์ที่ใช้ติดต่อกับเว็บเซิร์ฟเวอร์ก็ไม่จำเป็นต้องมีทรัพยากรของระบบมากเพราะหน้าที่หลักคือแปล (Render) เอกสาร HTML เพื่อแสดงผลบนเว็บเบราว์เซอร์เท่านั้น เทคโนโลยีจาวาเซิร์ฟเวอร์จึงเหมาะกับการใช้งานในโครงการวิจัยนี้

2.3 Secure Socket Layer

Secure Socket Layer หรือ SSL เป็นเทคโนโลยีที่เอื้ออำนวยให้การเชื่อมต่อบนโครงข่ายคอมพิวเตอร์มีความปลอดภัยสูงมากเพราะ SSL ใช้หลักการของ Cryptography นั่นคือในการเชื่อมต่อ นั้นข้อมูลที่จะถูกส่งออกไปจะถูก Encrypt ก่อนแล้วค่อยส่งผ่านโครงข่ายออกไปและข้อมูลจะถูก Decrypt ในฝั่งรับกลับคืนมาก่อนที่จะถูกประมวลผลใด ๆ SSL ถูกพัฒนาขึ้นมาโดย Netscape ในปี ค.ศ.1994 และต่อมา Internet Engineering Task Force (IETF) ได้เปลี่ยนชื่อ SSL เป็น Transport Layer Security หรือ TLS

สำหรับโครงข่ายอินเทอร์เน็ตแล้ว SSL ทำให้การเชื่อมต่อบนโครงข่ายอินเทอร์เน็ตโดยใช้ โปรโตคอล TCP/IP มีความปลอดภัยสูงขึ้นมากโดย SSL จะถูกเพิ่มเข้าไประหว่าง Transport Layer และ Application Layer ในโปรโตคอลสแตคของ TCP/IP ดังแสดงในรูปที่ 2.3 ในปัจจุบัน SSL มีการใช้งานอย่างกว้างขวางโดยส่วนใหญ่แล้ว SSL จะถูกใช้คู่กับ Hypertext Transfer Protocol (HTTP) ซึ่งเป็นโปรโตคอลสำหรับเว็บเพจบนโครงข่ายอินเทอร์เน็ตเราเรียกโปรโตคอล HTTP ซึ่งถูกเก็บซ่อนอยู่ในโปรโตคอล SSL นี้ว่า HTTPS ดังนั้นการเชื่อมต่อระหว่างเว็บเบราว์เซอร์และเว็บเซิร์ฟเวอร์ โดยใช้ HTTPS นี้จะมีความปลอดภัยสูงมากจึงเป็นที่นิยมใช้ในการลงทะเบียนแบบออนไลน์และการชำระเงินด้วยบัตรเครดิตผ่านโครงข่ายอินเทอร์เน็ต SSL สามารถทำการอิมพีเม้นต์ได้ง่ายและมีโปรแกรมอำนวยความสะดวกมากมายดังเช่น Java Security Socket Extension (JSSE) 1.0.3 ซึ่งเป็นเครื่องมือที่ใช้ในงานวิจัยนี้

TCP/IP Protocol Stack With SSL

TCP/IP Layer	Protocol
Application Layer	HTTP, NNTP, Telnet, FTP, etc.
Secure Sockets Layer	SSL
Transport Layer	TCP
Internet Layer	IP

รูปที่ 2.3 SSL ในโปรโตคอลสแตค TCP/IP

2.4 ไมโครคอนโทรลเลอร์ AVR

ไมโครคอนโทรลเลอร์ที่ใช้ในงานวิจัยนี้เป็นแบบ 8 บิตตระกูล AVR เบอร์ AT90S8535 มีโครงสร้างการทำงานและสถาปัตยกรรมแบบ RISC (Reduced Instruction Set Computer) ประมวลผลด้วยความเร็ว 1 MIPS (Million Instructions Per Second) มีหน่วยความจำขนาด 8 kB แบบ Flash Memory จึงสามารถทำการเขียนหรือลบข้อมูลในหน่วยความจำได้สะดวก ทำให้การพัฒนาหรือเปลี่ยนแปลงระบบควบคุมทำได้ง่ายและรวดเร็ว ส่วนภาษาที่ใช้ในการโปรแกรมเป็นภาษาแอสเซมบลีของ AVR

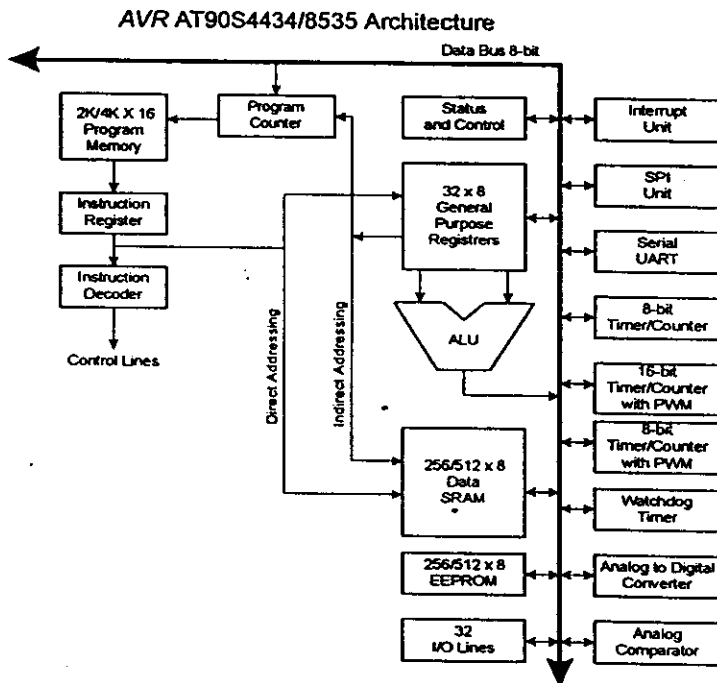
2.4.1 คุณสมบัติของ AVR-AT90S8535

AVR-AT90S8535 เป็นคอนโทรลเลอร์ตัวหนึ่งที่ได้รับการพัฒนาโดยบริษัท ATMEL และกำลังได้รับความนิยมมากในปัจจุบัน ซึ่ง AT90S8535 นี้มีคุณสมบัติที่น่าสนใจมากมาย แต่ในที่นี้จะขอล่าวพอเป็นสังเขปดังนี้

1. เป็นสถาปัตยกรรมที่ภายในถูกออกแบบให้ใช้สถาปัตยกรรมแบบ RISC (Reduce Instruction Set Computer)
2. มีคำสั่งในการควบคุมการทำงานของไมโครคอนโทรลเลอร์ จำนวน 118 คำสั่ง
3. หน่วยความจำแบบ Flash สำหรับบันทึก Program Memory ขนาด 8 KByte
4. หน่วยความจำแบบ EEPROM สำหรับบันทึก Data Memory ขนาด 512 Byte
5. หน่วยความจำแบบ RAM ขนาด 512 Byte
6. ระบบการเปลี่ยนสัญญาณ Analog to Digital ขนาด 10 บิต จำนวน 8 channel
7. รีจิสเตอร์สำหรับใช้งานทั่วไปจำนวน 32 ตัว
8. พอร์ตอินพุต และ เอาต์พุต ขนาด 8 บิต จำนวน 4 พอร์ต
9. ความถี่สัญญาณนาฬิกาสูงสุด 8 MHz
10. ระบบการรีเซ็ตแบบฮาร์ดโนมัลเมื่อเริ่มจ่ายกระแสเข้าไมโครคอนโทรลเลอร์
11. ระบบการอินเตอร์รัพท์จากภายนอก
12. Timer/Counter ขนาด 16 บิต 1 channel
13. Timer/Counter ขนาด 8 บิต 2 channel
14. ไฟเลี้ยงวงจรของไอซี Vcc: 4.0 – 6.0 V

2.4.2 สถาปัตยกรรมภายในและการจัดหน่วยความจำ AVR-AT90S8535

โครงสร้างภายในจะประกอบด้วยรีจิสเตอร์ใช้งานทั่วไปขนาด 8 บิตจำนวน 32 ตัว ที่สามารถเข้าถึงข้อมูลได้ภายใน 1 สัญญาณนาฬิกา ซึ่งหมายความว่า MCU (Micro Controller Unit) สามารถจัดการข้อมูลภายในรีจิสเตอร์ใช้งานทั่วไปได้เสร็จภายใน 1 รอบของสัญญาณนาฬิกา โดยรีจิสเตอร์ R26-R31 เป็นรีจิสเตอร์ขนาด 8 บิต จำนวน 6 ตัว ที่สามารถจับคู่เพื่อใช้เป็นรีจิสเตอร์ขนาด 16 บิต ได้ 3 ตัว ซึ่งจะมีชื่อประจำแต่ละตัวคือ X, Y และ Z ตามลำดับ

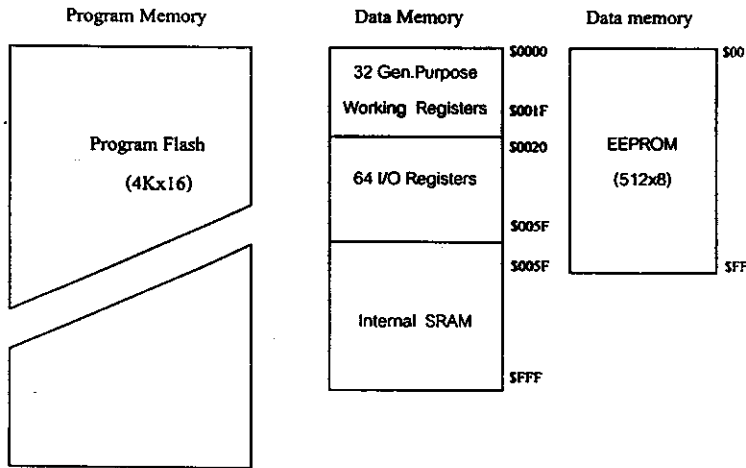


รูปที่ 2.4 สถาปัตยกรรมแบบ RISC ของ AT90S8535

ALU (Arithmetic Unit) จะสนับสนุนการกระทำทางคณิตศาสตร์และลอจิก ระหว่างรีจิสเตอร์กับรีจิสเตอร์ หรือระหว่างรีจิสเตอร์กับค่าคงที่ ซึ่งการเรียกใช้รีจิสเตอร์ในงานทั่วไปสามารถกระทำได้โดยการอ้างหน่วยความจำภายในที่ตำแหน่ง \$00 - \$1F จำนวน 32 ตำแหน่ง และใน MCU ได้จัดแบ่งให้มีรีจิสเตอร์ที่ใช้ควบคุมการทำงานของหน่วยอินพุตและเอาต์พุตต่าง ๆ อีก 64 ตำแหน่ง โดยสามารถเรียกใช้งานได้โดยการอ้างตำแหน่งหน่วยความจำที่ตำแหน่ง \$20 - \$5F

ระบบการทำงานของไมโครคอนโทรลเลอร์ใช้หลักการออกแบบของ HAVARD ด้วยการแยกระบบบัสของโปรแกรมและบัสของข้อมูลออกจากกัน โดยโปรแกรมจะมีการประมวลผลด้วย Single Level Pipelining ซึ่งทำให้ CPU (Central Processing Unit) สามารถ Fetch และ Execute คำสั่ง

ได้ภายใน 1 คาบเวลา ด้วยคำสั่ง JUMP และ CALL แบบ RELATIVE ที่สามารถกระโดดข้ามการทำงานได้ไกลถึง 24k/4k ซึ่งใน 1 คำสั่งจะใช้รหัสการทำงาน 16 บิต หรือ 1 Word โดยทุกครั้งที่มีการอินเตอร์รัพท์ หรือการข้ามไปทำงานในโปรแกรมย่อยค่า ของ PC (Program Counter) จะถูกเก็บลงใน STACK ซึ่งจะใช้พื้นที่หน่วยความจำใน SRAM บางส่วนเพื่อทำเป็นพื้นที่ของ STACK โดยโครงสร้างของหน่วยความจำสามารถแสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 โครงสร้างของหน่วยความจำของ AT90S8535

2.4.3 รีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์ทั้งหมดสามารถใช้ชุดคำสั่งเพื่อเข้าถึงได้โดยตรงและจะใช้เวลาการเข้าถึงเพียง 1 สัญญาณนาฬิกา โดยคำสั่ง SBCI, SUBI, CPI และ ANDI และ ORI ซึ่งกระทำระหว่างรีจิสเตอร์กับค่าคงที่หรือรีจิสเตอร์กับรีจิสเตอร์ และคำสั่ง LDI ที่ใช้โหลดค่าคงที่เข้าในรีจิสเตอร์ จะต้องใช้งานกับรีจิสเตอร์ R16-R31 ส่วนคำสั่ง SBC, SUB, CP, AND และ OR และคำสั่งใช้งานอื่นๆ สามารถใช้งานได้กับรีจิสเตอร์ทั่วไป จากรูปที่ 2.xx เป็นการแสดงการจัดวางตำแหน่งของรีจิสเตอร์ใช้งานทั่วไปทั้งหมด โดยมีรีจิสเตอร์ที่สามารถนำมาใช้งานเป็นรีจิสเตอร์คู่เพื่อทำเป็นตัวชี้ข้อมูลที่อยู่ในหน่วยความจำ ซึ่งรีจิสเตอร์ในกลุ่มนี้จะใช้ชื่อว่า X Y และ Z

2.4.4 หน่วยความจำ SRAM

หน่วยความจำภายใน MCU จัดไว้ 608 คำแหน่ง โดยหน่วยความจำทั้งหมดถูกแบ่งออกเป็นพื้นที่ของรีจิสเตอร์ใช้งานทั่วไป, รีจิสเตอร์ใช้งาน I/O และหน่วยความจำภายใน SRAM โดย 96

ตำแหน่งแรกจะถูกแบ่งเป็นส่วนของรีจิสเตอร์ และอีก 512 ตำแหน่ง ถูกจัดไว้ให้เป็นหน่วยความจำภายใน SRAM ซึ่งการเข้าถึงข้อมูลจะถูกแบ่งเป็น 5 ส่วน คือ Direct , Indirect with Displacement, Indirect with Pre-Decrement และ Indirect with Post-Increment

	7	0	Addr.	
	R0		\$00	
	R1		\$01	
	R2		\$02	
	...			
	R13		\$0D	
	R14		\$0E	
	R15		\$0F	
	R16		\$10	
	R17		\$11	
	...			
	R26		\$1A	X-register low byte
	R27		\$1B	X-register high byte
	R28		\$1C	Y-register low byte
	R29		\$1D	Y-register high byte
	R30		\$1E	Z-register low byte
	R31		\$1F	Z-register high byte

รูปที่ 2.6 โครงสร้างรีจิสเตอร์ใช้งานทั่วไป

2.5 การบันทึกข้อมูลเสียงแบบดิจิทัล

เครื่องตอบรับโทรศัพท์อัตโนมัติที่มีใช้อยู่ทั่วไปนั้น ส่วนมากมีการจัดเก็บข้อมูลเสียงแบบแอนาล็อกซึ่งจัดเก็บโดยการบันทึกลงในเทปคลาสเซ็ท การเปิดฟังข้อความเสียงแต่ละข้อความทำได้ช้าและอายุการใช้งานที่สั้น ข้อจำกัดของเทคโนโลยีแบบเดิมนั้นสามารถแก้ไขด้วยเทคโนโลยีของวงจรรวมที่ทำให้มีไอซีหน่วยความจำความจุสูง ๆ และเหมาะสำหรับการเก็บข้อมูลแบบดิจิทัล

2.5.1 หลักการทำงานของไอซี W51300-701

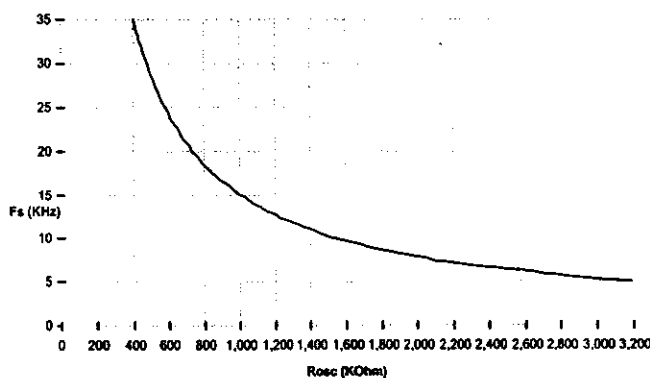
ไอซี W51300-701 เป็นตัวดำเนินการหลักในการบันทึกข้อมูลเสียงพูดและเปิดเสียงพูดในขั้นตอนการตอบรับโทรศัพท์แบบอัตโนมัติ ไอซีดังกล่าวใช้วิธีการเข้ารหัสข้อมูลแบบดิจิทัลที่เรียกว่าการมอดูเลตแบบเดลต้าที่ปรับตัวเองได้ (Adaptive Delta Modulation, ADM) ซึ่งสามารถควบคุมคุณภาพ ความชัดเจนและขนาดของข้อมูลเสียงได้จากการเปลี่ยนแปลงความถี่ของการชักตัวอย่าง ขนาดของข้อมูลจะลดลงเมื่อความถี่ของการชักตัวอย่างลดลงแต่คุณภาพ ความชัดเจนของเสียงก็จะลดลงด้วย ในการใช้งานจึงต้องเลือกทั้งคุณภาพ ความชัดเจนของเสียงและขนาดของข้อมูลให้เหมาะสม โดยในงานวิจัยนี้ใช้ความถี่ของการชักตัวอย่าง 15 kHz ข้อมูลเสียงแบบดิจิทัลที่ได้จากการเข้ารหัส

ข้างต้นจะถูกจัดเก็บในไอซี W55F20 ซึ่งเป็นไอซีหน่วยความจำขนาด 2 เมกกะบิต แบบ Flash EEPROM มีอายุการเก็บรักษาข้อมูล 10 ปี

2.5.2 คุณสมบัติของไอซี W51300-701

ไอซี W51300-701 มีคุณสมบัติที่สำคัญดังนี้

1. มีอัตราการชักตัวอย่างสูงสุดที่ความถี่ 24 กิโลเฮิร์ตซ์เมื่อความต้านทาน R_{osc} เท่ากับ 620 โอห์ม อัตราการชักตัวอย่างนี้สามารถปรับเลือกได้โดยขึ้นอยู่กับค่าความต้านทาน R_{osc} ซึ่งความสัมพันธ์ระหว่างค่าความต้านทาน R_{osc} กับค่าอัตราการชักตัวอย่างแสดงไว้ในรูปที่ 2.7
2. ขาอินพุตควบคุมหน้าที่การทำงาน 8 อินพุตที่มีการแก้ไข Debounced เพื่อให้แน่ใจได้ว่าจะไม่เกิดการผิดพลาดจากสัญญาณรบกวนขณะทำงาน
3. สามารถตั้งโหมดการทำงานได้ทั้งแบบ Single หรือ Multi voice ได้และตั้งการทำงานเป็นแบบปกติหรือจะต่อร่วมกับ CPU ได้
4. สามารถเพิ่มความจุของข้อมูลได้โดยการต่อไอซีที่เป็นหน่วยความจำ (หน่วยความจำภายนอก) โดยสามารถต่อแบบคาสเคดกันได้โดยตรง
5. จัดแบ่งการทำงานในฟังก์ชันบันทึก ลบ เล่นกลับ ได้อย่างอิสระ
6. มีฟังก์ชันตรวจจับแรงดันไปเลี้ยงต่ำที่ค่า 3.0 โวลต์
7. มีเอาต์พุตที่ต่อลำโพงได้โดยตรงและเอาต์พุตกระแสไฟฟ้า 5 มิลลิแอมป์ สามารถขับออกลำโพงได้โดยตรงโดยไม่ต้องมีวงจรขยายสัญญาณ
8. ประหยัดไฟโดยใช้กระแสเพียง 0.01 ไมโครแอมป์ขณะ Stand by และ 5 มิลลิแอมป์ขณะทำงาน



รูปที่ 2.7 ความสัมพันธ์ระหว่างค่าความต้านทาน R_{osc} กับอัตราการชักตัวอย่าง

2.6 วิธีดำเนินการวิจัย

วิธีดำเนินการวิจัยเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ตแบ่งได้เป็นข้อ ๆ ดังนี้

1. ศึกษาการถอดรหัสสัญญาณ DTMF การเขียนโปรแกรมบนโครงข่ายคอมพิวเตอร์เทคโนโลยีจาวาเซิร์ฟเล็ตและจาวาเนทีฟอินเทอร์เน็ตเฟส
2. ออกแบบและสร้างวงจรพร้อมเขียนโปรแกรมควบคุมเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลและทดสอบการทำงาน
3. คิดตั้งเซิร์ฟเวอร์ สร้างโฮมเพจ และเขียนโปรแกรมบนโครงข่ายคอมพิวเตอร์ทั้งฝั่งไคลเอนและเซิร์ฟเวอร์ ทดสอบเครื่องเซิร์ฟเวอร์และปรับปรุงโปรแกรมที่เขียนขึ้น
4. ทำการทดลองและทดสอบการทำงานของระบบและแก้ไขข้อบกพร่องต่าง ๆ ที่เกิดขึ้น
5. สรุปผลการทดลอง จัดทำรายงานวิจัยและบทความวิชาการ

2.7 อุปกรณ์ที่จำเป็นในการวิจัย

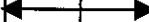
อุปกรณ์ที่จำเป็นในการวิจัยการวิจัยเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านอินเทอร์เน็ต มีดังนี้

1. เครื่องคอมพิวเตอร์
2. ออสซิลโลสโคป
3. ดิจิตอลมัลติมิเตอร์
4. บอร์ดไมโครคอนโทรลเลอร์
5. อุปกรณ์ต่อพ่วงเครื่องคอมพิวเตอร์สำหรับเครื่องไคลเอนและเซิร์ฟเวอร์
6. อุปกรณ์ชิ้นส่วนอิเล็กทรอนิกส์ต่าง ๆ

2.8 แผนการดำเนินงานตลอดโครงการวิจัย

แผนการดำเนินงานตลอดโครงการวิจัย แสดงรายละเอียดได้ดังในตารางที่ 2.1

ตารางที่ 2.1 แผนการดำเนินงานตลอดโครงการวิจัย

กิจกรรม	ผลที่คาดหวัง	ปีงบประมาณ พ.ศ.2545/เดือน					
		1-2	3-4	5-6	7-8	9-10	11-12
1.ศึกษาการถอดรหัสสัญญาณ DTMF การเขียนโปรแกรมบนโครงข่ายคอมพิวเตอร์เทคโนโลยีจาวาเซิร์ฟเล็ทและจาวาเนทีฟอินเตอร์เฟส	1.แนวทางในการวิจัยและข้อมูลประกอบการวิจัย						
2. ออกแบบและสร้างวงจรพร้อมเขียนโปรแกรมควบคุมเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลและทดสอบการทำงาน	2. เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลและโปรแกรมควบคุม						
3. คิดตั้งเซิร์ฟเวอร์ สร้างโฮมเพจและเขียนโปรแกรมบนโครงข่ายคอมพิวเตอร์ทั้งฝั่งไคลเอนและเซิร์ฟเวอร์ ทดสอบเครื่องเซิร์ฟเวอร์และปรับปรุงโปรแกรม	3.โฮมเพจของเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลสำหรับการตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต						
4. ทดสอบระบบทั้งหมด แก้ไขข้อบกพร่องที่เกิดขึ้น	4.เครื่องตอบรับดังกล่าวสามารถใช้งานผ่านโครงข่ายอินเทอร์เน็ตได้						
5. สรุปผลจัดทำรายงานวิจัยและบทความวิชาการ	5.เผยแพร่ข้อมูลในรูปแบบบทความวิชาการและรายงาน						

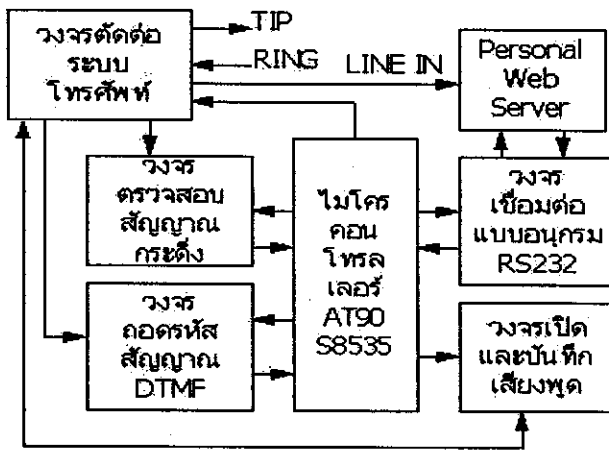
* เดือนที่ 1 หมายถึงเดือนมกราคม 2545

บทที่ 3

หลักการการทำงานและการออกแบบระบบ

3.1 กล่าวนำ

บทนี้จะกล่าวถึงหลักการการทำงานและการออกแบบระบบของเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัล ซึ่งการทำงานของระบบจะเป็นแบบอัตโนมัติ เมื่อมีโทรศัพท์เรียกเข้าระบบจะรอประมาณ 20 วินาทีและเมื่อไม่มีผู้รับสายระบบจะรับสายแทนซึ่งจะมีเสียงคอยแนะนำการใช้งานให้กับผู้ที่โทรศัพท์เข้ามา เช่น การฝากข้อความ การป้อนรหัสผ่าน การตรวจสอบข้อความ และถ้าไม่มีการกดคีย์โทรศัพท์ใด ๆ ภายใน 20 วินาทีระบบจะวางสายเอง สำหรับการตรวจสอบข้อความสามารถเปิดฟังได้ที่เครื่องโดยตรงหรือทำการตรวจสอบผ่านโทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่ นอกจากนี้ยังสามารถตรวจสอบผ่านอินเทอร์เน็ตได้อีก หลักการทำงานและการออกแบบระบบจะแยกอธิบายเป็นส่วนของฮาร์ดแวร์และส่วนของซอฟต์แวร์ ดังนี้



รูปที่ 3.1 แผนภาพบล็อกของระบบ

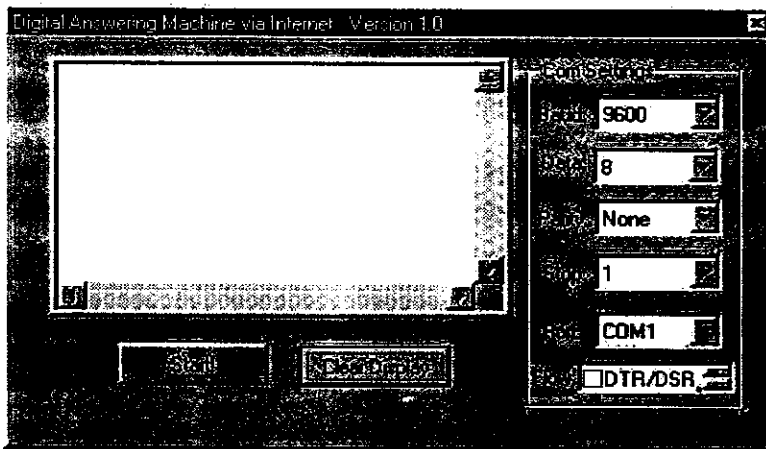
3.2 การทำงานและการออกแบบฮาร์ดแวร์

จากแผนภาพบล็อกในรูปที่ 3.1 ไมโครคอนโทรลเลอร์มีหน้าที่หลักคือ ควบคุมการทำงานของ วงจรตัดต่อระบบโทรศัพท์ วงจรตรวจสอบสัญญาณกระดิ่ง วงจรถอดรหัสสัญญาณ DTMF (Dual Tone Multi Frequency type) วงจรเปิดและบันทึกเสียงพูดและวงจรเชื่อมต่อแบบอนุกรมระหว่างไมโครคอนโทรลเลอร์และเครื่องเว็บเซิร์ฟเวอร์

สำหรับการเชื่อมต่อกับเครื่องเว็บเซิร์ฟเวอร์นั้นใช้การเชื่อมต่อแบบอนุกรมมาตรฐาน RS232 (RS232 Serial Interface) เมื่อมีโทรศัพท์เข้ามาแล้วเครื่องตอบรับโทรศัพท์ถูกเลือกโหมดการทำงานเป็นการฝากข้อความ ระบบจะมีการบันทึกข้อความไว้ที่เครื่องเว็บเซิร์ฟเวอร์ด้วย โดยไมโครคอนโทรลเลอร์จะส่งสัญญาณควบคุมผ่านการเชื่อมต่อดังกล่าวให้เครื่องเว็บเซิร์ฟเวอร์ทำการบันทึกข้อความนั้น โดยสัญญาณเสียงแบบแอนะล็อกจะป้อนผ่านช่อง LINE IN ของการ์ดเสียงและถูกบันทึกเก็บไว้สำหรับการตรวจสอบข้อความทางผ่านอินเทอร์เน็ตต่อไป

3.3 การทำงานและการออกแบบซอฟต์แวร์

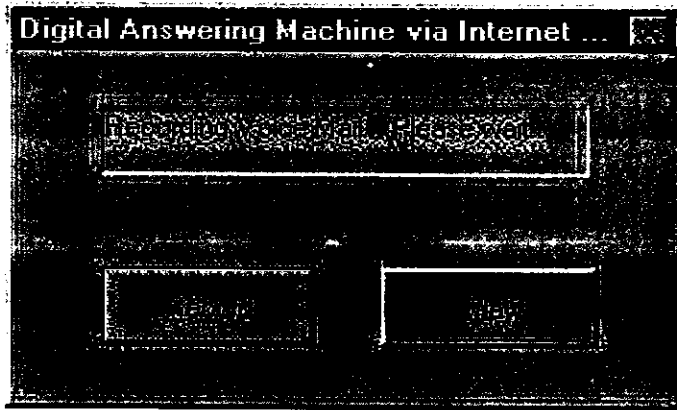
โปรแกรมเชื่อมต่อระหว่างเครื่องเว็บเซิร์ฟเวอร์กับเครื่องตอบรับโทรศัพท์ผ่านพอร์ตอนุกรม (Serial Port) มีการโปรแกรมโดยใช้ภาษา Visual C++ 6.0 โปรแกรมนี้จะทำหน้าที่ส่งผ่านสัญญาณควบคุมการบันทึกข้อความและจะทำงานตลอดเวลาที่เครื่องเว็บเซิร์ฟเวอร์ทำงานเพื่อจะได้ไม่พลาดการติดต่อใด ๆ จากการสื่อสารผ่านโครงข่ายโทรศัพท์



รูปที่ 3.4 โปรแกรมการเชื่อมต่อผ่านพอร์ตอนุกรม

สำหรับโปรแกรมที่ทำหน้าที่บันทึกข้อความบนเครื่องเว็บเซิร์ฟเวอร์นั้นใช้ภาษา Visual C++ 6.0 ซึ่งโปรแกรมหักจะถูกรู้กใ้ใช้งานเมื่อมีการบันทึกข้อความเท่านั้น โดยมีการตั้งเวลาให้โปรแกรมทำการบันทึกข้อความประมาณ 60 วินาที หลังจากนั้นโปรแกรมจะปิดตัวลงเพื่อคืนทรัพยากรให้กับระบบต่อไป การบันทึกข้อความดังกล่าวสามารถบันทึกเก็บไว้บนเครื่องเว็บเซิร์ฟเวอร์ 20 ข้อความ โดยกำหนดรูปแบบเป็นไฟล์มัลติมีเดียของระบบปฏิบัติการ Windows ซึ่งใช้การเข้ารหัสข้อมูลแบบ

PCM (Pulse Code Modulation) มีอัตราการชักตัวอย่าง 8 kHz ใช้จำนวนบิตในการเข้ารหัส 16 บิต และระบบเสียงเป็นแบบโมโน



รูปที่ 3.5 โปรแกรมบันทึกข้อความบนเครื่องเว็บเซิร์ฟเวอร์

งานวิจัยนี้มีการออกแบบให้เว็บเซิร์ฟเวอร์สนับสนุนโปรโตคอล HTTP ซึ่งถูกเก็บซ่อนอยู่ในโปรโตคอล SSL หรือที่เรียกว่า HTTPS ดังนั้นการส่งข้อมูลใด ๆ ก็ตามระหว่างไคลเอนต์กับเซิร์ฟเวอร์จะมีการ Encrypt ข้อมูลก่อนเสมอ ทำให้การส่งข้อมูลผ่านโครงข่ายอินเทอร์เน็ตมีความปลอดภัยสูงมาก

ในการตรวจสอบข้อความผ่านอินเทอร์เน็ตเครื่องไคลเอนต์ต้องมีโปรแกรมเว็บเบราว์เซอร์เพื่อใช้ในการติดต่อไปยังโฮมเพจที่สร้างไว้บนเว็บเซิร์ฟเวอร์โดยใช้ HTTPS และต้องมีโปรแกรมมัลติมีเดียสำหรับเล่นไฟล์เสียง เช่น โปรแกรม Windows Media Player หรือโปรแกรม Winamp รวมทั้งต้องมีการ์ดเสียงและลำโพงหรือหูฟังอยู่ด้วย

เมื่อโปรแกรมเว็บเบราว์เซอร์ ติดต่อเข้าไปที่โฮมเพจของ เครื่องตอบรับโทรศัพท์แบบดิจิทัล เพื่อทำการตรวจสอบข้อความ เว็บเซิร์ฟเวอร์จะมีระบบรักษาความปลอดภัย โดยให้ผู้ใช้งานป้อนชื่อและรหัสผ่านก่อน เมื่อข้อมูลถูกต้องจึงจะสามารถตรวจสอบข้อความได้ ในขั้นตอนการตรวจสอบข้อความนั้น เว็บเบราว์เซอร์จะส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์ แล้วที่ฝั่งเซิร์ฟเวอร์จะมีการทำงานของโปรแกรมที่เขียนโดยใช้ภาษาเซิร์ฟเล็ต ซึ่งโปรแกรมจะทำการตรวจนับไฟล์เสียงพร้อมทั้งเก็บรวบรวมข้อมูลต่าง ๆ ของไฟล์เสียงเพื่อใช้แสดงผล เช่น วัน เดือน ปี และเวลาที่ข้อความถูกฝากไว้ ขณะเดียวกันเว็บเซิร์ฟเวอร์ก็จะส่งสายข้อมูลซึ่งเป็นผลลัพธ์จากการทำงานของโปรแกรมดังกล่าวกลับไปให้ไคลเอนต์เพื่อแสดงผลบนเว็บเบราว์เซอร์ ผู้ใช้งานก็จะทราบข้อมูลล่าสุดจากการแสดงผลนี้ และสามารถคลิกบนลิงค์ที่ภาษาเซิร์ฟเล็ตสร้างให้เพื่อฟังข้อความได้ทันที จะเห็นว่าเอกสาร HTML

ถูกสร้างในขณะที่มีการทำงานของโปรแกรมจาวาเซิร์ฟเล็ค ดังนั้นข้อมูลที่แสดงผลผ่านเว็บเบราว์เซอร์
ออกไปจึงทันสมัยตลอดเวลา ทำให้การตรวจสอบข้อความถูกต้องและทันทั่วทั้งที่

บทที่ 4

ผลการทดลองและการทดสอบการใช้งาน

4.1 กล่าวนำ

เครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลผ่านโครงข่ายอินเทอร์เน็ตที่ได้รับการออกแบบขึ้นได้มีการทดสอบการใช้งานครั้งนี้คือ ทดสอบการบันทึกข้อความเสียงแบบดิจิทัลบนไอซีหน่วยความจำ การฝากข้อความและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์แบบต่าง ๆ และลำดับสุดท้ายเป็นการตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต ผลการทดสอบดังกล่าวแสดงรายละเอียดดังนี้

4.2 การบันทึกข้อความเสียงแบบดิจิทัลบนไอซีหน่วยความจำ

ในการทดลองนี้ได้ทดสอบการบันทึกข้อความเสียงแบบดิจิทัลลงในไอซีหน่วยความจำ W55F20 ขนาด 4 เมกกะบิต โดยกำหนดความถี่ของการชักตัวอย่าง 15 kHz ซึ่งจะได้บิตเรตเท่ากับ 15 kbit/sec ระยะเวลาในการบันทึกข้อมูลเสียงพูดสามารถคำนวณได้จาก $4 \text{ Mbit} / 15 \text{ kbit/sec}$ เท่ากับ 267 วินาที หรือ 4 นาที 27 วินาที ในการออกแบบระบบกำหนดระยะเวลาการบันทึกข้อความไว้ข้อความละ 60 วินาที จำนวน 4 ข้อความ โดยหน่วยความจำส่วนที่เหลือจะถูกใช้เพื่อแยกเซกเมนต์ของข้อความเพื่อให้ง่ายต่อการเข้าถึงข้อความและจากการทดสอบพบว่าสามารถบันทึกข้อความได้ 4 ข้อความตามเวลาที่ออกแบบจริง

4.3 การฝากข้อความผ่านโครงข่ายโทรศัพท์

การทดลองนี้ได้ทำการทดสอบการฝากข้อความผ่านโครงข่ายโทรศัพท์แบบต่าง ๆ เช่น โทรศัพท์บ้าน โทรศัพท์สาธารณะและโทรศัพท์เคลื่อนที่ (DTAC 1800 และ GSM900) โดยทำการโทรศัพท์ผ่านโครงข่ายดังกล่าวข้างต้นเข้ามาที่เครื่องตอบรับโทรศัพท์อัตโนมัติแล้วทำการฝากข้อความมีความยาว 60 วินาทีแล้ววางสายจำนวน 4 ครั้ง เวลาเฉลี่ยที่ใช้ในการฝากข้อความผ่านโครงข่ายโทรศัพท์แบบต่าง ๆ แสดงดังในตารางที่ 4.1

ตารางที่ 4.1 เวลาเฉลี่ยที่ใช้ในการฝากข้อความผ่านโครงข่ายโทรศัพท์

ประเภท ของโทรศัพท์	เวลาในการฝากข้อความ (วินาที)				เวลาเฉลี่ย (วินาที)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	
โทรศัพท์บ้าน	86.44	86.40	86.41	86.43	86.42
โทรศัพท์ สาธารณะ	85.76	85.74	85.78	85.83	85.78
โทรศัพท์เคลื่อน ที่ (DTAC 1800)	82.32	82.35	82.40	82.37	82.36
โทรศัพท์เคลื่อน ที่ (GSM900)	82.84	82.81	82.83	82.80	82.82

4.4 การตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์

การทดลองนี้เป็นการทดสอบการตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์แบบต่าง ๆ เช่น โทรศัพท์บ้าน โทรศัพท์สาธารณะ โทรศัพท์เคลื่อนที่ (DTAC 1800) และโทรศัพท์เคลื่อนที่ (GSM900) โดยทำการโทรศัพท์ผ่านโครงข่ายดังกล่าวข้างต้นเข้ามาทำการตรวจสอบข้อความที่ได้ทำการฝากไว้แล้วในการทดลองก่อนหน้านี้จำนวน 4 ครั้ง เวลาเฉลี่ยที่ใช้ในการฝากข้อความผ่านโครงข่ายโทรศัพท์แสดงดังในตารางที่ 4.2

ตารางที่ 4.2 เวลาเฉลี่ยที่ใช้ในการตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์

ประเภท ของโทรศัพท์	เวลาในการตรวจสอบข้อความ (วินาที)				เวลาเฉลี่ย (วินาที)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	
โทรศัพท์บ้าน	98.16	98.06	98.14	98.08	98.11
โทรศัพท์ สาธารณะ	95.33	95.39	95.31	95.37	95.35
โทรศัพท์เคลื่อน ที่ (DTAC 1800)	85.70	85.63	85.62	85.69	85.66
โทรศัพท์เคลื่อน ที่ (GSM900)	85.50	85.55	85.58	85.53	85.54

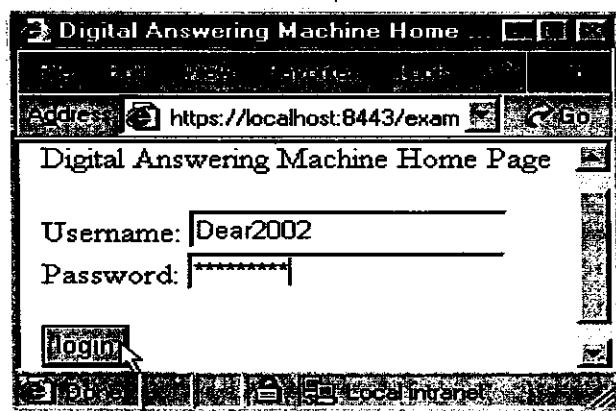
จากการทดสอบการฝากข้อความและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์ เวลาเฉลี่ยที่ใช้ในการฝากข้อความและตรวจสอบข้อความผ่านโครงข่ายดังกล่าว แสดงในตารางที่ 4.3

ตารางที่ 4.3 เวลาเฉลี่ยที่ใช้ในการฝากและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์

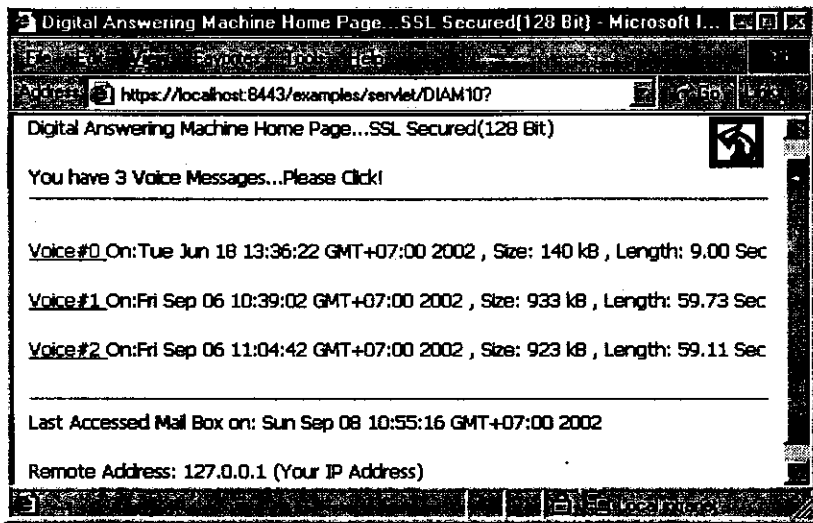
ประเภทของการบริการ	เวลาเฉลี่ย (วินาที)			
	โทรศัพท์บ้าน	โทรศัพท์สาธารณะ	โทรศัพท์เคลื่อนที่ (DTAC 1800)	โทรศัพท์เคลื่อนที่ (GSM900)
ฝากข้อความ	86.42	85.78	82.36	82.82
ตรวจสอบข้อความ	98.11	95.35	85.66	85.54

4.5 การตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต

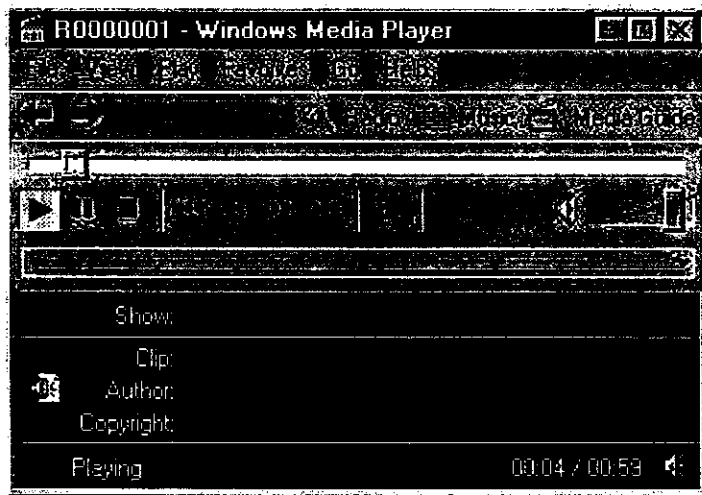
การทดสอบลำดับสุดท้ายเป็นการตรวจสอบข้อความผ่านอินเทอร์เน็ตภายในองค์กร เครื่องเว็บเซิร์ฟเวอร์ที่ใช้ทดสอบเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ใช้ระบบปฏิบัติการ Windows 98 SE และติดตั้งโปรแกรม Jakarta-Tomcat 3.2.3 Java 2 SDK 1.3 และ JSSE 1.0.3 โดยเริ่มจากการล็อกอินเข้าโฮมเพจ ตรวจสอบข้อความและการแสดงผลบนเว็บเบราว์เซอร์ ซึ่งพิจารณาถึงความครบถ้วนและความถูกต้องของข้อมูล เช่น วัน เดือน ปี และเวลาในการสร้างไฟล์เสียงโดยเทียบกับ File Properties ในโปรแกรม Windows Explorer และทดสอบการฟังข้อความผ่านโครงข่ายดังกล่าว ผลการทดสอบแสดงในตารางที่ 4.4 และการแสดงผลบนเว็บเบราว์เซอร์แสดงดังในรูปที่ 4.1 รูปที่ 4.2 และรูปที่ 4.3



รูปที่ 4.1 การล็อกอินเข้าโฮมเพจเพื่อตรวจสอบข้อความ



รูปที่ 4.2 การตรวจสอบข้อความโดยใช้ Internet Explorer 6.0



รูปที่ 4.3 การฟังข้อความโดยใช้โปรแกรม Windows Media Player 6.0

ตารางที่ 4.4 ผลการตรวจสอบข้อความผ่านโครงข่ายอินเทอร์เน็ต

โปรแกรม Web Browser ที่ใช้ทดสอบ	การ แสดงผล	ความถูกต้อง ของข้อมูล	การฟังข้อความเสียง	
			Windows Media Player 6.0	Winamp 2.71
Internet Explorer 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน
Opera 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน
Netscape 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน

บทที่ 5

บทสรุป

5.1 สรุปผลการวิจัย

โครงการวิจัยนี้ได้ดำเนินการศึกษา วิเคราะห์และออกแบบเครื่องตอบรับโทรศัพท์อัตโนมัติแบบดิจิทัลที่สามารถทำงานได้บน 3 โครงข่ายหลัก ผู้ใช้งานสามารถโทรศัพท์เข้าไปทำการฝากหรือตรวจสอบข้อความโดยใช้โทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่ โดยสามารถฝากข้อความได้สูงสุด 24 ข้อความ เครื่องตอบรับโทรศัพท์ดังกล่าวนี้สามารถเชื่อมต่อกับเครื่องเว็บเซิร์ฟเวอร์ส่วนบุคคล ทำให้สามารถทำการตรวจสอบข้อความผ่านอินเทอร์เน็ตได้และมีความปลอดภัยของข้อมูลสูงมากเนื่องจากมีการใช้ HTTPS ในการเชื่อมต่อระหว่างเว็บเบราว์เซอร์และเว็บเซิร์ฟเวอร์ และจากการนำเทคโนโลยีจาวาเซิร์ฟพลิเคชันมาใช้ในการเขียนโปรแกรมเพื่อทำงานบนฝั่งเซิร์ฟเวอร์ ทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตได้รับข้อความล่าสุดตลอดเวลาและผลการทดสอบการใช้งานพบว่าเครื่องตอบรับโทรศัพท์ดังกล่าวสามารถทำงานได้ตามที่ออกแบบไว้อย่างถูกต้อง

5.2 ข้อเสนอแนะ

5.2.1 ข้อเสนอแนะทั่วไป

1. ควรมีการศึกษาการทำงานของระบบอย่างละเอียด โดยการจำลองการทำงานของระบบในสถานะการทำงานต่าง ๆ ขึ้น โดยใช้โปรแกรม Petri Net หรือ โปรแกรม Colored Petri Net เพื่อป้องกันการเกิด Deadlock ที่อาจจะเกิดขึ้นได้ในขณะใช้งาน
2. ควรได้มีการศึกษาและออกแบบระบบให้สามารถทำงานได้บนระบบปฏิบัติการ Linux เพื่อจะได้ลดค่าใช้จ่ายเกี่ยวกับระบบปฏิบัติการของตัวเซิร์ฟเวอร์ลง

5.2.2 ข้อเสนอแนะสำหรับการศึกษาค้างต่อไป

1. พัฒนาระบบให้สามารถทำการฝากข้อความจากโสมเพจบนโครงข่ายอินเทอร์เน็ตมายังเครื่องตอบรับโทรศัพท์แบบดิจิทัลและสามารถรองรับการทำงานแบบผู้ใช้งานหลาย ๆ คน(Multi user) ได้
2. เชื่อมต่อเว็บเซิร์ฟเวอร์ส่วนกับฐานข้อมูลภายในองค์กรโดยใช้เทคโนโลยี JDBC (Java Database Connectivity) เพื่อให้สามารถใช้งานฐานข้อมูลดังกล่าวได้ทุกแห่งที่มีอินเทอร์เน็ตใช้งาน

บรรณานุกรม

- [1] ประกมลฤช แสงชูวงศ์ สุธี คนกระโทก ทวีศักดิ์ ศรีโปดก. 2543. เครื่องตอบรับโทรศัพท์และเปิด-ปิดอุปกรณ์ไฟฟ้าผ่านโครงข่ายโทรศัพท์. รายงานโครงการงานวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี.
- [2] B. Glassmeyer, E. Melin. 2000, Digital Answering Machine (Online). Available URL: <http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/>
- [3] G. Shachor . 2001, Tomcat IIS Howto (Online). Available URL:<http://apache.org/tomcat/>
- [4] Sun Microsystems. 2001, Java Servlet Technology (Online). Available URL: <http://java.sun.com/products/servlet/>
- [5] A. Patzer et. al. Professional Java Server Programming, Wrox Press Ltd, 2000.
- [6] Deitel H.M., Deitel P.J. Java How to Program, O'Reily&Associates Inc, 1999.
- [7]Sun Microsystems. 2000, Java Secure Socket Extension (Online). Available URL: <http://java.sun.com/security/>
- [8] ทีมงานอีทีที. 2542, คู่มือ AVR Enhanced RISC Microcontrollers AT90S/LS4434 AT90S/LS8535. บริษัทอีทีทีจำกัด, กรุงเทพมหานคร.

ภาคผนวก ก

โปรแกรมควบคุมระบบการทำงาน

```

// Project: Voice Recording for Digital Automatic Answering
// Machine via Internet Network
// Date: Dec 19, 2002
// File: PlayMMWave.cpp
// Title: Implementation file

#include "stdafx.h"
#include "record2.h"
#include "PlayWave.h"
#include "PlayMMWave.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////
// CPlayMMSound
UINT PlaySound( LPVOID pParam );
IMPLEMENT_DYNCREATE(CPlayMMSound, CWinThread)
CPlayMMSound::CPlayMMSound()
{
    m_pPlaySound = NULL;
    strcpy(m_FileName, "");
    ZeroMemory(&m_MMCKInfoParent, sizeof
(MMCKINFO));
    ZeroMemory(&m_MMCKInfoChild, sizeof
(MMCKINFO));
    m_SoundBuffer = NULL;
    m_pSoundThread = NULL;
    m_pStopEvent = new CEvent(FALSE, TRUE);
}

CPlayMMSound::~CPlayMMSound()
{
    if(m_SoundBuffer)
        delete m_SoundBuffer;
}

BOOL CPlayMMSound::InitInstance()
{
    // TODO: perform and per-thread initialization here
    return TRUE;
}

int CPlayMMSound::ExitInstance()
{
    // TODO: perform any per-thread cleanup here
    return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(CPlayMMSound, CWinThread)
//{{AFX_MSG_MAP(CPlayMMSound)
// NOTE - the ClassWizard will add and remove mapping
// macros here.
//}}AFX_MSG_MAP
ON_THREAD_MESSAGE(WM_PLAYMMSOUND_P
LAYFILE, PlaySoundFile)
ON_THREAD_MESSAGE(WM_PLAYMMSOUND_C
LOSEFILE, CloseSoundFile)
ON_THREAD_MESSAGE(WM_PLAYMMSOUND_P
LAYSOUNDPTR, OnPlaySoundPtr)
ON_THREAD_MESSAGE(WM_PLAYMMSOUND_E
NDTHREAD, OnEndThread)
END_MESSAGE_MAP()
////////////////////
// CPlayMMSound message handlers
LRESULT CPlayMMSound::PlaySoundFile(WPARAM
wParam, LPARAM lParam)
{
    char *pFileName = (char*) lParam;
    strcpy(m_FileName, pFileName);
    if(!OpenSoundFile(m_FileName))
        return FALSE;
    return TRUE;
}

BOOL CPlayMMSound::OpenSoundFile(CString csFileName)
{
    // code taken from Visual C++ Multimedia -- Aitken and Jarol p 122
    CloseSoundFile(0, 0);
    m_hmmio = mmioOpen((LPSTR)(LPCTSTR)
csFileName, NULL, MMIO_READ);
    if(!m_hmmio)
    {
        AfxMessageBox("unable to open Sound MM
File");
        return FALSE;
    }
    m_MMCKInfoParent.fccType = mmioFOURCC
('W', 'A', 'V', 'E');
}

```

```

int errorCode = mmioDescend(m_hmmio,
&m_MMCKInfoParent,NULL,MMIO_FINDRIFF);
if(errorCode)
{
    AfxMessageBox("Error descending into
file"),

    mmioClose(m_hmmio,0);
    m_hmmio = NULL;
    return FALSE;
}
m_MMCKInfoChild.ckid = mmioFOURCC('f','m','t',' ');
errorCode =
mmioDescend(m_hmmio,&m_MMCKInfoChild,&m_MMCKInfoPar
ent,MMIO_FINDCHUNK);
if(errorCode)
{
    AfxMessageBox("Error descending in file");
    mmioClose(m_hmmio,0);
    m_hmmio = NULL;
    return FALSE;
}
DWORD bytesRead = mmioRead(m_hmmio,
(LPSTR)&m_PCMWaveFmtRecord,m_MMCKInfoChild.cksize);
if(bytesRead < 0)
{
    AfxMessageBox("Error reading PCM wave
format record");

    mmioClose(m_hmmio,0);
    return FALSE;
}
// open output sound file
errorCode =
mmioAscend(m_hmmio,&m_MMCKInfoChild,0);
if(errorCode)
{
    AfxMessageBox("Error ascending in File");
    mmioClose(m_hmmio,0);
    m_hmmio = NULL;
    return FALSE;
}
m_MMCKInfoChild.ckid = mmioFOURCC('d','a','t','a');

```

```

errorCode =
mmioDescend(m_hmmio,&m_MMCKInfoChild,
&m_MMCKInfoParent,MMIO_FINDCHUNK);
if(errorCode)
{
    AfxMessageBox("error reading data chunk");
    mmioClose(m_hmmio,0);
    m_hmmio = NULL;
    return FALSE;
}
m_BytesToRead = m_PCMWaveFmtRecord.nChannels
(m_PCMWaveFmtRecord.wBitsPerSample/sizeof
(BYTE))
*m_PCMWaveFmtRecord.nBlockAlign*]
if(m_BytesToRead >
    m_MMCKInfoChild.cksize)
    m_BytesToRead =
m_MMCKInfoChild.cksize;
m_bContinuePlaying = TRUE;
m_pStopEvent->ResetEvent();
m_pSoundThread = AfxBeginThread(PlaySound,this
return TRUE;
}
BYTE* CPlayMMSound::ReadSoundFile(int*dwBytesRead)
{
    if(m_BytesToRead <= 0)
        return NULL;
    if(!m_hmmio)
        return NULL;
    TRACE("ReadSoundFile\n");
    BYTE * pSoundBuffer = new BYTE[m_BytesToRead];
    if(!pSoundBuffer)
        return NULL;
    DWORD dwRetc = mmioRead(m_hmmio,(LPSTR)
pSoundBuffer, m_BytesToRead);
    if(dwRetc == -1)
    {
        AfxMessageBox("Error reading WAVE
file\n");
        if(pSoundBuffer)

```

```

        delete pSoundBuffer;
    }
    return NULL;
}

else if(dwRetc == 0)
{
    CloseSoundFile(0,0);
    m_bContinuePlaying = FALSE;
}

if(dwBytesRead)
    *dwBytesRead = dwRetc;

char debugBuffer[MAX_PATH];
sprintf(debugBuffer,"read %d bytes\n",dwRetc);
TRACE(debugBuffer);
return pSoundBuffer;
}

LRESULT CPlayMMSound::CloseSoundFile(WPARAM
wParam,LPARAM lParam)
{
    if(m_pStopEvent)
        m_pStopEvent->SetEvent();
    TRACE("STOP EVENT SET\n");
    m_bContinuePlaying = FALSE;
    if(m_hmmio)
    {
        mmioClose(m_hmmio,0);
        m_hmmio = NULL;
    }
    return TRUE;
}

LRESULT CPlayMMSound::OnPlaySoundPtr(WPARAM
wParam,LPARAM lParam)
{
    CPlaySound*pPlaySound = (CPlaySound*)lParam;
    m_pPlaySound = pPlaySound;
    return TRUE;
}

LRESULT CPlayMMSound::OnEndThread(WPARAM wParam,
LPARAM lParam)
{
    CloseSoundFile(0,0);
    ::PostQuitMessage(0);
    return TRUE;
}

}

UINT CPlayMMSound::PlaySound( LPVOID pParam )
{
    CPlayMMSound* pPlayMMSound = (CPlayMMSound*)
pParam;

    if(pPlayMMSound &&
        pPlayMMSound->m_pPlaySound)
    {
        pPlayMMSound->m_pPlaySound->
PostThreadMessage(WM_PLAY_SOUND_STARTPLAYING,GetCu
rrentThreadId(),(LPARAM)0);
    }
    if(!pPlayMMSound)
        return FALSE;

    if(pPlayMMSound && !pPlayMMSound->
m_pPlaySound)
        return FALSE;

    HANDLE hHandle[2];
    hHandle[1] = (HANDLE) pPlayMMSound->
m_pPlaySound->m_pSemaphore->m_hObject;
    hHandle[0] = (HANDLE) pPlayMMSound->
m_pStopEvent->m_hObject;

    int dwBytesRead = 0;
    BYTE* pSoundBuffer = pPlayMMSound->
ReadSoundFile(&dwBytesRead);
    while(pPlayMMSound->m_bContinuePlaying)
    {
        DWORD dwRetc = WaitForMultipleObjects
(2,hHandle,FALSE,INFINITE);

        if(dwRetc == WAIT_FAILED)
        {
            DWORD dwRetc = ::GetLastError();
            char errorBuffer[MAX_PATH];

            sprintf(errorBuffer,"WaitForMultipleObjects failed:
%d\n",dwRetc);

            TRACE(errorBuffer);
        }
        dwRetc = WAIT_OBJECT_0;
        if(dwRetc == 0) // stop event;
        {
            TRACE("STOP Event seen\n");

```

```

        break;
    }
    if(!pPlayMMSound->m_bContinuePlaying)
        break;
    WAVEHDR* pWaveHdr = new WAVEHDR;
    ZeroMemory(pWaveHdr,sizeof
(WAVEHDR));

    pWaveHdr->lpData = (char*)pSoundBuffer;
    pWaveHdr->dwBufferLength =
dwBytesRead;

    if(pPlayMMSound &&
        pPlayMMSound->m_pPlaySound)
    {
        pPlayMMSound->m_pPlaySound-
>PostThreadMessage(WM_PLAYSOUND_PLAYBLOCK,GetCurr
entThreadId(),(LPARAM)pWaveHdr);
    }

    pSoundBuffer = pPlayMMSound->
ReadSoundFile(&dwBytesRead);
}

if(pPlayMMSound &&
    pPlayMMSound->m_pPlaySound)
{
    (WM_PLAYSOUND_STOPPLAYING,GetCurrentThreadId(),
(LPARAM)0);
}

return TRUE;
}

//End of file.

```

```

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:  Dec 19, 2002
// File:  PlayMMWave.h
#if
!defined(AFX_PLAYMMSOUND_H_F3383123_E3DD_11D3_
85_806A49C10000__INCLUDED_)
#define
AFX_PLAYMMSOUND_H_F3383123_E3DD_11D3_8685_8
49C10000__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// PlayMMSound.h : header file
#define WM_PLAYMMSOUND_PLAYFILE WM_USER+301
#define WM_PLAYMMSOUND_CLOSEFILE WM_USER+302
#define WM_PLAYMMSOUND_PLAYSOUNDPTR
WM_USER+303
#define WM_PLAYMMSOUND_ENDTHREAD WM_USER+3
////////////////////////////////////
// CPlayMMSound thread
class CPlayMMSound : public CWinThread
{
    DECLARE_DYNCREATE(CPlayMMSound)
public:
    BYTE* ReadSoundFile(int*dwBytesRead);
    CPlayMMSound();    // protected constructor used
dynamic creation
protected:
    BOOL OpenSoundFile(CString csFileName);
    LRESULT OnEndThread(WPARAM wParam,
LPARAM lParam);
// Attributes
public:
    CEvent* m_pStopEvent;
    CPlaySound* m_pPlaySound;
    BOOL m_bContinuePlaying;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CPlayMMSound)

```

```

public:
virtual BOOL InitInstance();
virtual int ExitInstance();
//}}AFX_VIRTUAL

static UINT PlaySound( LPVOID pParam );

// Implementation
protected:

virtual ~CPlayMMSound();

// Generated message map functions
//{{AFX_MSG(CPlayMMSound)

// NOTE - the ClassWizard will add and
remove member functions here.

//}}AFX_MSG

afx_msg LRESULT PlaySoundFile(WPARAM wParam,
LPARAM lParam);

afx_msg LRESULT CloseSoundFile(WPARAM
wParam, LPARAM lParam);

afx_msg LRESULT OnPlaySoundPtr(WPARAM
wParam, LPARAM lParam);

DECLARE_MESSAGE_MAP()
char m_FileName[MAX_PATH];
MMCKINFO m_MMCKInfoParent;
MMCKINFO m_MMCKInfoChild;
HMMIO m_hmmio;
WAVEFORMATEX m_PCMWaveFmtRecord;
BYTE m_WaveFormatExtras[MAX_PATH]; // for non
PCM wavformatex there are extra bytes at the end

BYTE * m_SoundBuffer;
CWinThread* m_pSoundThread;
int m_BytesToRead;

};

////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous //line.

#endif //

#ifdef(AFX_PLAYMMSOUND_H_F3383123_E3DD_11D3_86
85_806A49C10000_INCLUDED_)

//End of file.

```

```

// Project: Voice Recording for Digital Automatic Answering
// Machine via Internet Network
// Date: Dec 19, 2002
// File: PlayWave.cpp
// Title: Implementation file

#ifndef __COMPLEX__
#define __COMPLEX__
typedef struct
{
    float real;
    float imag;
}
COMPLEXNUMBER;
#endif

#include "stdafx.h"
#include "winbase.h"
#include <mmsystem.h>
#include <math.h>
#include "record2.h"
#include "PlayWave.h"
#include "RecordDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CPlaySound
IMPLEMENT_DYNCREATE(CPlaySound, CWinThread)

#define BLOCKSAHEAD 3

CPlaySound::CPlaySound(int iHertz)
{
    m_nOutputBuffers = 0;
    m_nMaxOutputBuffers = 0;
    memset(&m_WaveFormatEx, 0x00, sizeof
(m_WaveFormatEx));

    m_WaveFormatEx.wFormatTag =
WAVE_FORMAT_PCM;

    m_WaveFormatEx.nChannels = 1;
    m_WaveFormatEx.wBitsPerSample = 16;
    m_WaveFormatEx.cbSize = 0;
    m_WaveFormatEx.nSamplesPerSec = iHertz; // 20050;

```

```

        m_WaveFormatEx.nAvgBytesPerSec =
m_WaveFormatEx.nSamplesPerSec
            *(m_WaveFormatEx.wBitsPerSample/8);
        m_WaveFormatEx.nBlockAlign =
            (m_WaveFormatEx.wBitsPerSample/8)*
            m_WaveFormatEx.nChannels;

        m_pSemaphore = new CSemaphore
(BLOCKS_AHEAD,BLOCKS_AHEAD);
        m_bPlay = FALSE;
    }

CPlaySound::~CPlaySound()
{
}

BOOL CPlaySound::InitInstance()
{
    // TODO: perform and per-thread initialization here
    return TRUE;
}

int CPlaySound::ExitInstance()
{
    // TODO: perform any per-thread cleanup here
    return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(CPlaySound, CWinThread)
    //{{AFX_MSG_MAP(CPlaySound)
        // NOTE - the ClassWizard will add and
remove mapping macros here.
        //}}AFX_MSG_MAP
        ON_THREAD_MESSAGE(WM_PLAY_SOUND_STAR
TPLAYING, OnStartPlaying)
        ON_THREAD_MESSAGE(WM_PLAY_SOUND_STOP
PLAYING, OnStopPlaying)
        ON_THREAD_MESSAGE(WM_PLAY_SOUND_PLAY
BLOCK, OnWriteSoundData)
        ON_THREAD_MESSAGE(MM_WOM_DONE,
OnEndPlaySoundData)
        ON_THREAD_MESSAGE(WM_PLAY_SOUND_ENDT
HREAD, OnEndThread)
    END_MESSAGE_MAP()
    ////////////////////////////////////////////
    // CPlaySound message handlers

```

```

LRESULT CPlaySound::OnStartPlaying(WPARAM wParam
LPARAM lParam)
{
    DWORD mmReturn = 0;
    if(m_bPlay)
        return FALSE;
    if(!m_bPlay)
    {
        // open wavein device
        MMRESULT mmReturn = ::waveOutOpen(&m_hP
WAVE_MAPPER,
&m_WaveFormatEx, ::GetCurrentThreadId(), 0,
CALLBACK_THREAD);
        if(mmReturn)
        {
            char errorbuffer[MAX_PATH];
            char errorbuffer1[MAX_PATH];
            waveOutGetErrorText(
mmReturn, errorbuffer, MAX_PATH);
            sprintf(errorbuffer1, "WAVEOUT:%x:%s", mmReturn
orbuffer);
            AfxMessageBox(errorbuffer1);
        }
        if(!mmReturn)
        {
            m_bPlay = TRUE;
        }
    }
    return TRUE;
}

LRESULT CPlaySound::OnStopPlaying(WPARAM wParam,
LPARAM lParam)
{
    MMRESULT mmReturn = 0;
    if(!m_bPlay)
        return FALSE;
    if(m_bPlay)
    {
        mmReturn = ::waveOutReset(m_hPlay);
        if(!mmReturn)
            m_bPlay = FALSE;
        Sleep(500);
    }
}

```

```

        if(!mmReturn)
            mmResult = ::waveOutWrite
            (m_hPlay, lpHdr, sizeof(WAVEHDR));
            if(mmResult)
                TRACE("error from waveoutwrite\n");
                m_nOutputBuffers++;
        }
    }
    return TRUE;
}

LRESULT CPlaySound::OnEndPlaySoundData(WPARAM
wParam, LPARAM lParam)
{
    LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;
    if(lpHdr)
    {
        ::waveOutUnprepareHeader(m_hPlay, lpHdr,
sizeof(WAVEHDR));
        if(lpHdr->lpData)
            delete ((BYTE*) lpHdr->lpData);
        delete lpHdr;
        m_pSemaphore->Unlock();
    }
    return ERROR_SUCCESS;
}

LRESULT CPlaySound::OnWriteSoundData(WPARAM wParam,
LPARAM lParam)
{
    LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;
    MMRESULT mmResult = 0;
    if(lpHdr)
    {
        char debugbuffer[256];
        sprintf(debugbuffer, "SOUND BUFFER
written: %d, %d\n", lpHdr->dwBufferLength, m_nOutputBuffers);
        TRACE(debugbuffer);
        if(m_bPlay)
        {
            short int* lpInt = (short int*) lpHdr->lpData;
            DWORD dwSamples = lpHdr->
dwBufferLength/sizeof(short int);
            ProcessSoundData(lpInt, dwSamples);
            mmResult = ::waveOutPrepareHeader
(m_hPlay, lpHdr, sizeof(WAVEHDR));
            if(mmResult)
                TRACE("error from waveoutprepareheader\n");
        }
    }
}

mmResult = ::waveOutWrite
(m_hPlay, lpHdr, sizeof(WAVEHDR));
if(mmResult)
    TRACE("error from waveoutwrite\n");
    m_nOutputBuffers++;
}
}
return ERROR_SUCCESS;
}

LPWAVEHDR CPlaySound::CreateWaveHeader()
{
    LPWAVEHDR lpHdr = new WAVEHDR;
    ZeroMemory(lpHdr, sizeof(WAVEHDR));
    BYTE* lpByte = new
BYTE[(m_WaveFormatEx.nBlockAlign*SOUNDSAMPLES)];
    lpHdr->lpData = (char *) lpByte;
    lpHdr->dwBufferLength =
(m_WaveFormatEx.nBlockAlign*SOUNDSAMPLES);
    return lpHdr;
}

void CPlaySound::ProcessSoundData(short int *sound, DWORD
dwSamples)
{
    LRESULT CPlaySound::OnEndThread(WPARAM wParam,
LPARAM lParam)
    {
        if(m_bPlay)
            OnStopPlaying(0,0);::PostQuitMessage(0);
        return TRUE;
    }
}

//End of file.

```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:   Dec 19, 2002
// File:   PlayWave.h
//if
#define(AFX_PLAYSOUND_H__5260C01C_03B2_11D2_A421_
FC4B2C882A60__INCLUDED_)
#define
AFX_PLAYSOUND_H__5260C01C_03B2_11D2_A421_FC4B2C
882A60__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// PlayWave.h : header file
#include <afxmt.h>
#ifndef __COMPLEX__
#define __COMPLEX__
typedef struct
{
    float real;
    float imag;
}
COMPLEXNUMBER;
#endif
////////////////////////////////////
// CPlaySound thread
#define WM_PLAYSOUND_STARTPLAYING WM_USER+600
#define WM_PLAYSOUND_STOPPLAYING WM_USER+601
#define WM_PLAYSOUND_PLAYBLOCK WM_USER+602
#define WM_PLAYSOUND_ENDTHREAD WM_USER+603
class CRecordDlg;
class CPlaySound : public CWinThread
{
    DECLARE_DYNCREATE(CPlaySound)
public:
    CPlaySound(int iHertz=8000);
// protected constructor used by dynamic creation
// Attributes
public:
// Operations
public:
// Overrides
```

```
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CPlaySound)
public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
//}}AFX_VIRTUAL
    virtual void ProcessSoundData(short int *sound,
DWORD dwSamples);
// Implementation
protected:
    virtual ~CPlaySound();
// Generated message map functions
//{{AFX_MSG(CPlaySound)
// NOTE - the ClassWizard will add and remove member
functions here.
//}}AFX_MSG
    afx_msg LRESULT OnStartPlaying(WPARAM wParam,
LPARAM lParam);
    afx_msg LRESULT OnStopPlaying(WPARAM wParam,
LPARAM lParam);
    afx_msg LRESULT OnEndPlaySoundData(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnWriteSoundData(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnEndThread(WPARAM wParam,
LPARAM lParam);
    DECLARE_MESSAGE_MAP()
protected:
    int m_nOutputBuffers;
    int m_nMaxOutputBuffers;
    WAVEFORMATEX m_WaveFormatEx;
    BOOL m_bPlay;
    HWAVEOUT m_hPlay;
public:
    CSemaphore* m_pSemaphore;
protected:
    LPWAVEHDR CreateWaveHeader();
};
////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous line.
```

```

#endif //
!defined(AFX_PLAYSOUND_H_5260C01C_03B2_11D2_A421_
FC4B2C882A60_INCLUDED_)
//End of file.

```

```

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:  Dec 19, 2002
// File:  record2.cpp
// Title:  Defines the class behaviors for the application
#include "stdafx.h"
#include <nmsystem.h>
#include "record2.h"
#include "recordDlg.h"
#include "PlayWave.h"
#include "RecordWave.h"
#include "WriteWaveFile.h"
#include "PlayMMWave.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CRecordApp
BEGIN_MESSAGE_MAP(CRecordApp, CWinApp)
   //{{AFX_MSG_MAP(CRecordApp)
// NOTE - the ClassWizard will add and remove mapping macros
//here.
//  DO NOT EDIT what you see in these blocks of generated code!
    }}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
////////////////////////////////////
// CRecordApp construction
CRecordApp::CRecordApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}
////////////////////////////////////
// The one and only CRecordApp object
CRecordApp theApp;
////////////////////////////////////
// CRecordApp initialization
int CRecordApp::ExitInstance()
{

```

```

        if(m_pPlayMMSound)
        {
            m_pPlayMMSound->PostThreadMessage
(WM_PLAYMMSOUND_ENDTHREAD,0,0);
            ::WaitForSingleObject(m_pPlayMMSound->
m_hThread, 5000);
            m_pPlayMMSound = NULL;
        }
        if(m_pPlaySound)
        {
            m_pPlaySound->PostThreadMessage
(WM_PLAYSOUND_ENDTHREAD,0,0);
            ::WaitForSingleObject(m_pPlaySound->
m_hThread, 5000);
            m_pPlaySound = NULL;
        }
        if(m_pRecordSound)
        {
            m_pRecordSound->PostThreadMessage
(WM_RECORDSOUND_ENDTHREAD,0,0);
            ::WaitForSingleObject(m_pRecordSound->
m_hThread, 5000);
            m_pRecordSound = NULL;
        }
        if(m_pWriteSound)
        {
            m_pWriteSound->PostThreadMessage
(WM_WRITESOUNDFILE_ENDTHREAD,0,0);
            ::WaitForSingleObject(m_pWriteSound->
m_hThread, 5000);
            m_pWriteSound = NULL;
        }
        return 0;
    }

    BOOL CRecordApp::InitInstance()
    {
        AfxEnableControlContainer();
        // Standard initialization

#ifdef _AFXDLL
        Enable3dControls(); // Call this when using MFC in a
shared DLL
#else

```

```

        Enable3dControlsStatic();
        // Call this when linking to MFC statically
#endif

        InitRecording();
        InitPlaying();
        InitWriting();
        InitPlayMMSound();
        if(m_pRecordSound)
        {
            m_pRecordSound->PostThreadMessage
(WM_RECORDSOUND_SOUNDPLAYER,(WPARAM)0,
(LPARAM)

            m_pPlaySound);
            m_pRecordSound->PostThreadMessage
(WM_RECORDSOUND_WRITERTHREAD,(WPARAM)0,
(LPARAM)

            m_pWriteSound);
        }

        CRecordDlg dlg;
        m_pMainWnd = &dlg;
        dlg.m_RecordThread = m_pRecordSound;
        dlg.m_PlayThread = m_pPlaySound;
        dlg.m_PlayMMSound = m_pPlayMMSound;
        int nResponse = dlg.DoModal();
        if (nResponse == IDOK)
        {
            // TODO: Place code here to handle when the dialog is
            // dismissed with OK
        }
        else if (nResponse == IDCANCEL)
        {
            // TODO: Place code here to handle when the dialog is
            // dismissed with Cancel
        }

        return FALSE;
    }

    BOOL CRecordApp::InitRecording()
    {
        m_pRecordSound = new CRecordSound();
        m_pRecordSound->CreateThread();
        return TRUE;
    }

```

```

BOOL CRecordApp::InitPlayMMSound()
{
    m_pPlayMMSound = new CPlayMMSound();
    m_pPlayMMSound->CreateThread();
    return TRUE;
}

BOOL CRecordApp::InitWriting()
{
    m_pWriteSound = new CWriteSoundFile();
    m_pWriteSound->CreateThread();
    return TRUE;
}

BOOL CRecordApp::InitPlaying()
{
    m_pPlaySound = new CPlaySound();
    m_pPlaySound->CreateThread();
    return TRUE;
}

//End of file.

// Project: Voice Recording for Digital Automatic Answering
// Machine via Internet Networks
// Date: Dec 19, 2002
// File: record2.h
// Title: Main header file for the Voice recording application
#ifdef
!defined(AFX_RECORD_H__5260C00F_03B2_11D2_A421_FC4B2C882A60__INCLUDED_)
#define
AFX_RECORD_H__5260C00F_03B2_11D2_A421_FC4B2C882A60__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif
#include "resource.h"
#include "RecordWave.h"
#include "PlayWave.h"
#include "PlayMMWave.h"
////////////////////////////////////
// CRecordApp:
// See record2.cpp for the implementation of this class
class CRecordApp : public CWinApp
{
public:
    CRecordApp();

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CRecordApp)
public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL
    virtual int ExitInstance();

// Implementation
    //{{AFX_MSG(CRecordApp)
// NOTE - the ClassWizard will add and remove member functions
//here.
// DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

```

```

protected:
    BOOL m_bRecording;
    CRecordSound* m_pRecordSound;
    CPlaySound* m_pPlaySound;
    CWriteSoundFile* m_pWriteSound;
    CPlayMMSound* m_pPlayMMSound;

protected:
    BOOL InitRecording();
    BOOL InitPlaying();
    BOOL InitWriting();
    BOOL InitPlayMMSound();

};

////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
// immediately before the previous //line.
#endif//

!defined(AFX_RECORD_H__5260C00F_03B2_11D2_A421_FC4B
2C882A60__INCLUDED_)
//End of file.

```

```

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Networks
// Date:  Dec 19, 2002
// File:  recordDlg.cpp
// Title:  Implementation file
#include "stdafx.h"
#include <mmsystem.h>
#include "record2.h"
#include "recordDlg.h"
#include "RecordWave.h"
#include "WriteWaveFile.h"
#include "PlayMMWave.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);

// DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:

//{{AFX_MSG(CAboutDlg)
//}}AFX_MSG
    DECLARE_MESSAGE_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
   //{{AFX_DATA_INIT(CAboutDlg)

```

```

    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CRecordDlg dialog
CRecordDlg::CRecordDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CRecordDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CRecordDlg)
// NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent
    DestroyIcon in Win32

    m_hIcon = AfxGetApp()->LoadIcon
(IDR_MAINFRAME);
}

void CRecordDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CRecordDlg)
    DDX_Control(pDX, IDC_BUTTON1, m_Recording);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CRecordDlg, CDialog)
    //{{AFX_MSG_MAP(CRecordDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
    ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
    ON_BN_CLICKED(IDC_TRAIN, OnTrain)
    ON_WM_TIMER()

```

```

    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CRecordDlg message handlers
BOOL CRecordDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // Add "About..." menu item to system menu.
    // TODO: Add extra initialization here
    m_bRecording = FALSE;
    SetTimer(START_RECORDTIME, START_TIME,
    NULL);
    SetTimer(STOP_RECORDTIME, STOP_TIME,
    NULL);
    SetTimer(CLOSE_RECORDPROG, CLOSE_TIME,
    NULL);
    return TRUE;
    // return TRUE unless you set the focus to a control
}

void CRecordDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

void CRecordDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for
        painting
        SendMessage(WM_ICONERASEBKGND,
        (LPARAM) dc.GetSafeHdc(), 0);
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics
        (SM_CXICON);

```

```

        int cylcon = GetSystemMetrics
(SM_CYICON);

        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cylcon + 1) / 2;
        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

HCURSOR CRecordDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

#define WM_RECORDSOUND_STARTRECORDING
WM_USER+500
#define WM_RECORDSOUND_STOPRECORDING
WM_USER+501
void CRecordDlg::OnTrain()
{
}

void CRecordDlg::OnButton1()
{
    if(m_bRecording)
    {
        m_Recording.SetWindowText("&Start
Recording");
        m_RecordThread->PostThreadMessage
(WM_RECORDSOUND_STOPRECORDING, 0, 0L);
        m_bRecording = FALSE;
    }
    else
    {
        m_Recording.SetWindowText("&Stop
Recording");
        m_RecordThread->PostThreadMessage
(WM_RECORDSOUND_STARTRECORDING, 0, 0L);
        m_bRecording = TRUE;
    }
}

void CRecordDlg::OnButton2()
{
    // TODO: Add your control notification handler code here
    if(m_PlayMMSound)
    {
        m_PlayMMSound->PostThreadMessage
        WM_PLAYMMSOUND_PLAYSOUND
        ,0,(LPARAM)m_PlayThread);
        m_PlayMMSound->PostThreadMessage
        (WM_PLAYMMSOUND_PLAYFILE,0
        (LPARAM)"sound.wav");
    }
}

void CRecordDlg::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    if(nIDEvent == START_RECORDTIME)
    {
        m_RecordThread->PostThreadMessage
        (WM_RECORDSOUND_STARTRECORDING, 0, 0L);
        KillTimer(START_RECORDTIME);
    }
    if(nIDEvent == STOP_RECORDTIME)
    {
        m_RecordThread->PostThreadMessage
        (WM_RECORDSOUND_STOPRECORDING, 0, 0L);
        KillTimer(STOP_RECORDTIME);
    }
    if(nIDEvent == CLOSE_RECORDPROG)
    {
        KillTimer(CLOSE_RECORDPROG);
        DestroyWindow();
    }
    CDialog::OnTimer(nIDEvent);
}

//End of file.

```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:   Dec 19, 2002
// File:   recordDlg.h
#if
#ifdef(AFX_RECORDDLG_H_5260C011_03B2_11D2_A421_
FC4B2C882A60__INCLUDED_)
#define
AFX_RECORDDLG_H_5260C011_03B2_11D2_A421_FC4B2C8
82A60__INCLUDED_
#define START_TIME 1000    // 1 SEC
#define STOP_TIME 60000    // 60 SEC
#define CLOSE_TIME 61000  // 61 SEC
#define START_RECORDTIME 1000
#define STOP_RECORDTIME 1001
#define CLOSE_RECORDPROG 1002
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#include "RecordWave.h"
#include "PlayWave.h"
////////////////////////////////////
// CRecordDlg dialog
class CRecordDlg : public CDialog
{
// Construction
public:
    CRecordDlg(CWnd* pParent = NULL);

// standard constructor

// Dialog Data
//{{AFX_DATA(CRecordDlg)
enum { IDD = IDD_RECORD_DIALOG };
CButton  m_Recording;
//}}AFX_DATA
CButton m_Train;
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CRecordDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);
// DDX/DDV support
//}}AFX_VIRTUAL
// Implementation
```

```
protected:
    HICON m_hIcon;
    // Generated message map functions
//{{AFX_MSG(CRecordDlg)
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM
IParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnButton1();
afx_msg void OnButton2();
afx_msg void OnTimer(UINT nIDEvent);
//}}AFX_MSG
afx_msg void OnTrain();
DECLARE_MESSAGE_MAP()
BOOL m_bRecording;

public:
    CRecordSound* m_RecordThread;
    CPlaySound* m_PlayThread;
    CPlayMMSound* m_PlayMMSound;
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous //line.
#endif //
#ifdef(AFX_RECORDDLG_H_5260C011_03B2_11D2_A421_
FC4B2C882A60__INCLUDED_)
//End of file.
```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:   Dec 19, 2002
// File:   RecordWave.cpp
// Title:  Implementation file

#include "stdafx.h"
#include <mmsystem.h>
#include "record2.h"
#include "RecordWave.h"
#include "PlayWave.h"
#include "WriteWaveFile.h"
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////
// CRecordSound
IMPLEMENT_DYNCREATE(CRecordSound, CWinThread)
#define MAXINPUTBUFFERS 25
#define FILECOUNT 10
CRecordSound::CRecordSound(int iHertz)
{
    m_nInputBuffers = 0;
    m_nMaxInputBuffers = MAXINPUTBUFFERS;
    memset(&m_WaveFormatEx, 0x00, sizeof
(m_WaveFormatEx));
    m_WaveFormatEx.wFormatTag =
WAVE_FORMAT_PCM;
    m_WaveFormatEx.nChannels = 1;
    m_WaveFormatEx.wBitsPerSample = 16;
    m_WaveFormatEx.cbSize = 0;
    m_WaveFormatEx.nSamplesPerSec = iHertz//20050;
    m_WaveFormatEx.nAvgBytesPerSec =
m_WaveFormatEx.nSamplesPerSec
        *(m_WaveFormatEx.wBitsPerSample/8);
    m_WaveFormatEx.nBlockAlign =
        (m_WaveFormatEx.wBitsPerSample/8)*
        m_WaveFormatEx.nChannels;
```

```
    m_bRecording = FALSE;
    m_Player = NULL;
    m_Writer = NULL;
}
CRecordSound::~CRecordSound()
{
}
BOOL CRecordSound::InitInstance()
{
    // TODO: perform and per-thread initialization here
    return TRUE;
}
int CRecordSound::ExitInstance()
{
    // TODO: perform any per-thread cleanup here
    return CWinThread::ExitInstance();
}
BEGIN_MESSAGE_MAP(CRecordSound, CWinThread)
    {{{AFX_MSG_MAP(CRecordSound)
// NOTE - the ClassWizard will add and remove mapping macros
//here.
}}}AFX_MSG_MAP
    ON_THREAD_MESSAGE(WM_RECORDSOUND_
ARTRECORDING, OnStartRecording)
    ON_THREAD_MESSAGE(WM_RECORDSOUND_
OPRECORDING, OnStopRecording)
    ON_THREAD_MESSAGE(MM_WIM_DATA,
OnSoundData)
    ON_THREAD_MESSAGE(WM_RECORDSOUND_
UNDPLAYER, OnPtrSoundPlayer)
    ON_THREAD_MESSAGE(WM_RECORDSOUND_
DTHREAD, OnEndThread)
    ON_THREAD_MESSAGE(WM_RECORDSOUND_
RITERTHREAD, OnPtrSoundWriter)
END_MESSAGE_MAP()
////////////////////////////////////
// CRecordSound message handlers
LRESULT CRecordSound::OnPtrSoundPlayer(WPARAM wPar
LPARAM lParam)
{
    m_Player = (CPlaySound*) lParam;
    return ERROR_SUCCESS;
```

```

}

LRESULT CRecordSound::OnStartRecording(WPARAM wParam,
LPARAM lParam)
{
    DWORD mmReturn = 0;
    if(m_bRecording)
        return FALSE;
    if(!m_bRecording)
    {
        // open wavein device
        MMRESULT mmReturn = ::waveInOpen(
&m_hRecord, WAVE_MAPPER,
        &m_WaveFormatEx,
::GetCurrentThreadId(), 0, CALLBACK_THREAD);
        if(mmReturn )
        {
            char errorbuffer[MAX_PATH];
            char errorbuffer1[MAX_PATH];
            waveInGetErrorText( mmReturn,
errorbuffer,MAX_PATH);
            sprintf(errorbuffer1,"WAVEIN:%x:%s",mmReturn,error
buffer);

            AfxMessageBox(errorbuffer1);
        }
        if(!mmReturn)
        {
            for(int i=0; i <
m_nMaxInputBuffers; i++)
            {
                LPWAVEHDR lpHdr =
CreateWaveHeader();

                mmReturn =
::waveInPrepareHeader(m_hRecord,lpHdr, sizeof(WAVEHDR));

                mmReturn = ::waveInAddBuffer
(m_hRecord, lpHdr, sizeof(WAVEHDR));

                m_nInputBuffers++;
            }

            mmReturn = ::waveInStart
(m_hRecord);

            if(mmReturn )
            {
                char errorbuffer[MAX_PATH];

```

```

char errorbuffer1[MAX_PATH];
            waveInGetErrorText( mmReturn,
errorbuffer, MAX_PATH);
            sprintf(errorbuffer1,"WAVEIN:%x:%s",mmReturn,errorbuffer);
            AfxMessageBox(errorbuffer1);
        }
        if(!mmReturn)
        {
            m_bRecording = TRUE;
            if(m_Player)
                m_Player->PostThreadMessage
(WM_PLAYSOUND_STARTPLAYING,0, 0);
            if(m_Writer)
            {
                PWRITESOUNDFILE pwsf=
(PWRITESOUNDFILE)
new WRITESOUNDFILE;

                ZeroMemory(pwsf,sizeof(WRITESOUNDFILE));
                char *p = pwsf->lpszFileName;
                char filename[256];
                char a[256];
                int ii = 0;
                int idNum;
                {
                    char ifilename[] = "c:/file_in.txt";
                    char ofilename[] = "c:/file_out.txt";
                    FILE *ofp, *ifp;
                    ifp = fopen(ifilename,"r");
                    fscanff(ifp,"%d",&idNum);
                    fclose(ifp);
                    ofp = fopen(ofilename,"w");
                    idNum = idNum + 1;
                    strcpy(a, filename);
                    fprintf(ofp,"%d",idNum);
                    ii = ii + idNum;
                    fclose(ofp);
                }
                wsprintf(filename, "R%07d.wav", ii);strcpy(a, filename);
                strcpy(p,a);
                memcpy(&pwsf->waveFormatEx,&m_WaveFormatEx,sizeof
(m_WaveFormatEx));
                m_Writer->PostThreadMessage(

```

```

        WM_WRITE SoundFile.FileName, 0,
(LPParam) pwsf);
    }
    return ERROR_SUCCESS;
}

}

return TRUE;
}

LRESULT CRecordSound::OnStopRecording(WPARAM wParam,
LPARAM lParam)
{
    MMRESULT mmReturn = 0;
    if(!m_bRecording)
        return FALSE;
    if(m_bRecording)
    {
        mmReturn = ::waveInStop(m_hRecord);
        if(!mmReturn)
            mmReturn = ::waveInReset(m_hRecord);
        if(!mmReturn)
            m_bRecording = FALSE;
        Sleep(500);
        if(!mmReturn)
            mmReturn = ::waveInClose(m_hRecord);
        if(m_Player)
            m_Player->PostThreadMessage
(WM_PLAY Sound.StopPlaying, 0, 0);
        if(m_Writer)
            m_Writer->PostThreadMessage
(WM_WRITE SoundFile.CloseFile, 0, 0);
        return mmReturn;
    }
    return TRUE;
}

LRESULT CRecordSound::OnSoundData(WPARAM wParam,
LPARAM lParam)
{
    LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;
    if(lpHdr)
    {
        short int * lpInt = (short int*) lpHdr->lpData;

```

```

        DWORD cbRecorded = lpHdr->
dwBytesRecorded;
        ::waveInUnprepareHeader(m_hRecord,
lpHdr, sizeof(WAVEHDR));
        ProcessSoundData(lpInt, cbRecorded/sizeof
(short int));
        if(m_Writer)
        {
            WAVEHDR* pWriteHdr = new
WAVEHDR;
            if(!pWriteHdr)
                return FALSE;
            memcpy(pWriteHdr, lpHdr, sizeof
(WAVEHDR));
            BYTE * pSound = new BYTE
[lpHdr->dwBufferLength];
            if(!pSound)
            {
                delete pWriteHdr;
                return FALSE;
            }
            memcpy(pSound, lpHdr->
lpData, lpHdr->dwBufferLength);
            pSound;
            m_Writer->PostThreadMessage
(WM_WRITE SoundFile.WriteBlock,
GetCurrentThreadId(),
(LPParam) pWriteHdr);
        }
        if(m_Player)
        {
            m_Player->PostThreadMessage
(WM_PLAY Sound.PlayBlock,
GetCurrentThreadId(),
(LPParam) lpHdr);
        }
        else
        {
            delete lpInt;
            delete lpHdr;
        }
    }
}

```

```

        char debugbuffer[256];
        sprintf(debugbuffer, "SOUND BUFFER
returned: %d\n",cbRecorded);
        TRACE(debugbuffer);
        if(m_bRecording)
        {
            LPWAVEHDR lpHdr =
CreateWaveHeader();

            ::waveInPrepareHeader
(m_hRecord,lpHdr, sizeof(WAVEHDR));

            ::waveInAddBuffer(m_hRecord,
lpHdr, sizeof(WAVEHDR)); m_nInputBuffers++;
        }
    }
    return ERROR_SUCCESS;
}

LPWAVEHDR CRecordSound::CreateWaveHeader()
{
    LPWAVEHDR lpHdr = new WAVEHDR;
    ZeroMemory(lpHdr, sizeof(WAVEHDR));
    BYTE* lpByte = new
BYTE[(m_WaveFormatEx.nBlockAlign*SOUNDSAMPLES)];
    lpHdr->lpData = (char *) lpByte;
    lpHdr->dwBufferLength =
(m_WaveFormatEx.nBlockAlign*SOUNDSAMPLES);
    return lpHdr;
}

void CRecordSound::ProcessSoundData(short int* sound, DWORD
dwSamples)
{
}

LRESULT CRecordSound::OnEndThread(WPARAM wParam,
LPARAM lParam)
{
    if(m_bRecording)
    {
        OnStopRecording(0, 0);
    }
    ::PostQuitMessage(0);
    return TRUE;
}

LRESULT CRecordSound::OnPtrSoundWriter(WPARAM wParam,
LPARAM lParam)

```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:   Dec 19, 2002
// File:   RecordWave.h
#ifdef _AFX_RECORDSOUND_H__5260C01B_03B2_11D2_A4
21_FC4B2C882A60_INCLUDED_
#define
AFX_RECORDSOUND_H__5260C01B_03B2_11D2_A421_FC4B
2C882A60_INCLUDED_
#ifdef _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// RecordWave.h : header file
#define WM_RECORDSOUND_STARTRECORDING
WM_USER+500
#define WM_RECORDSOUND_STOPRECORDING
WM_USER+501
#define WM_RECORDSOUND_SOUNDPLAYER
WM_USER+502
#define WM_RECORDSOUND_ENDTHREAD WM_USER+503
#define WM_RECORDSOUND_WRITERTHREAD
WM_USER+504
////////////////////////////////////
// CRecordSound thread
#include <mmsystem.h>
#include "PlayWave.h"
#include "WriteWaveFile.h"
#define SOUNDSAMPLES 1000
class CRecordSound : public CWinThread
{
    DECLARE_DYNCREATE(CRecordSound)
public:
    CRecordSound(int iHertz=8000);
// protected constructor used by dynamic creation
    LPWAVEHDR CreateWaveHeader();
    virtual void ProcessSoundData(short int* sound,
DWORD dwSamples);
// Attributes
public:
// Operations
public:
```

```
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CRecordSound)
public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
//}}AFX_VIRTUAL
// Implementation:
protected:
    virtual ~CRecordSound();
// Generated message map functions
//{{AFX_MSG(CRecordSound)
// NOTE - the ClassWizard will add and remove member functions
//here.
//}}AFX_MSG
    afx_msg LRESULT OnStartRecording(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnStopRecording(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnSoundData(WPARAM wParam
LPARAM lParam);
    afx_msg LRESULT OnPtrSoundPlayer(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnPtrSoundWriter(WPARAM
wParam, LPARAM lParam);
    afx_msg LRESULT OnEndThread(WPARAM wParam
LPARAM lParam);
    DECLARE_MESSAGE_MAP()
protected:
    HWAVEIN m_hRecord;
    int m_nInputBuffers;
    int m_nMaxInputBuffers;
    WAVEFORMATEX m_WaveFormatEx;
    BOOL m_bRecording;
    CPlaySound* m_Player;
    CWriteSoundFile* m_Writer;
    CWinThread *m_pSoundThread;
};
////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous //line.
```

```

#endif//

!defined(AFX_RECORDSOUND_H__5260C01B_03B2_11D2_A4
21_FC4B2C882A60_INCLUDED_)

//End of file.

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network
// Date:  Dec 19, 2002
// File:  resource.h
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
#define IDM_ABOUTBOX                0x0010
#define IDD_ABOUTBOX                100
#define IDS_ABOUTBOX                101
#define IDD_RECORD_DIALOG            102
#define IDR_MAINFRAME                128
#define IDD_TRAIN                    129
#define IDC_BUTTON1                  1000
#define IDC_STATIC_SILENCE           1001
#define IDC_STATIC_ZEROCROSSINGS     1002
#define IDC_TRAIN                    1003
#define IDC_UP                       1004
#define IDC_DOWN                     1005
#define IDC_BUTTON2                  1005
#define IDC_RIGHT                    1006
#define IDC_LEFT                     1007
#define IDC_CLICK                    1008
#define IDC_STOPTRAINING             1009

// Next default values for new objects
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        130
#define _APS_NEXT_COMMAND_VALUE        32771
#define _APS_NEXT_CONTROL_VALUE        1006
#define _APS_NEXT_SYMED_VALUE          101
#endif
#endif

//End of file.

```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   stdafx.cpp
// Title:   Source file that includes just the standard includes
//      record.pch will be the pre-compiled header
//      stdafx.obj will contain the pre-compiled type information
#include "stdafx.h"
//End of file.
```

```
// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   stdafx.h
// Title:   Include file for standard system include files
#if !defined(AFX_STDAFX_H__5260C013_03B2_11D2_A421_FC4B2C882A60__INCLUDED_)
#define AFX_STDAFX_H__5260C013_03B2_11D2_A421_FC4B2C882A60__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#define VC_EXTRALEAN
// Exclude rarely-used stuff from Windows headers
#include <afxwin.h>      // MFC core and standard components
#include <afxext.h>      // MFC extensions
#include <afxdisp.h>     // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>      // MFC support for Windows Common
//Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous //line.
#endif //
#define(AFX_STDAFX_H__5260C013_03B2_11D2_A421_FC4B2C882A60__INCLUDED_)
//End of file.
```

```

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   WriteWaveFile.cpp
// Title:   Implementation file
#include "stdafx.h"

#include <mmsystem.h>
#include "record2.h"
#include "WriteWaveFile.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////

// CWriteSoundFile
IMPLEMENT_DYNCREATE(CWriteSoundFile, CWinThread)

CWriteSoundFile::CWriteSoundFile()
{
}

CWriteSoundFile::~CWriteSoundFile()
{
}

BOOL CWriteSoundFile::InitInstance()
{
    // TODO: perform and per-thread initialization here
    return TRUE;
}

int CWriteSoundFile::ExitInstance()
{
    // TODO: perform any per-thread cleanup here
    return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(CWriteSoundFile, CWinThread)
    //{AFX_MSG_MAP(CWriteSoundFile)

// NOTE - the ClassWizard will add and remove mapping macros
//here.

    }AFX_MSG_MAP

    ON_THREAD_MESSAGE(WM_WRITESOUNDFILE_
FILENAME, CreateWaveFile)

    ON_THREAD_MESSAGE(WM_WRITESOUNDFILE_
WRITEBLOCK, WriteToSoundFile)

    ON_THREAD_MESSAGE(WM_WRITESOUNDFILE_
CLOSEFILE, CloseSoundFile)

```

```

    ON_THREAD_MESSAGE(WM_RECORDSOUND_EN
DTHREAD, OnEndThread)
END_MESSAGE_MAP()

LRESULT CWriteSoundFile::CreateWaveFile(WPARAM wParam,
LPARAM lParam)
{
    PWRITESOUNDFILE pWriteSoundFile =
(PWRITESOUNDFILE) lParam;

    int cbWaveFormatEx = sizeof(WAVEFORMATEX) +
pWriteSoundFile->waveFormatEx.cbSize;

    m_hFile = ::mmioOpen(pWriteSoundFile->
lpzFileName, NULL,
MMIO_CREATE|MMIO_WRITE|MMIO_EXCLUSIVE |
MMIO_ALLOCBUF);

    if(!m_hFile)
        return TRUE;

    ZeroMemory(&m_MMCKInfoParent, sizeof
(MMCKINFO));

    m_MMCKInfoParent.fccType = mmioFOURCC
('W','A','V','E');

    MMRESULT mmResult = ::mmioCreateChunk(
m_hFile, &m_MMCKInfoParent,

    MMIO_CREATERIFF);

    ZeroMemory(&m_MMCKInfoChild, sizeof
(MMCKINFO));

    m_MMCKInfoChild.ckid = mmioFOURCC('f','m','t',' ');
    m_MMCKInfoChild.cksize = cbWaveFormatEx;
    mmResult = ::mmioCreateChunk(m_hFile,
&m_MMCKInfoChild, 0);

    mmResult = ::mmioWrite(m_hFile,
(char*)&pWriteSoundFile->waveFormatEx, cbWaveFormatEx);

    mmResult = ::mmioAscend(m_hFile,
&m_MMCKInfoChild, 0);

    m_MMCKInfoChild.ckid = mmioFOURCC('d','a','t','a');

    mmResult = ::mmioCreateChunk(m_hFile,
&m_MMCKInfoChild, 0);

    return ERROR_SUCCESS;
}

LRESULT CWriteSoundFile::WriteToSoundFile(WPARAM
wParam, LPARAM lParam)

```

```

{
    LPWAVEHDR lpHdr = (LPWAVEHDR) lParam;
    int cbLength = lpHdr->dwBufferLength;
    if(lpHdr)
    {
        char *soundbuffer = (char*) lpHdr->lpData;
        if(m_hFile && soundbuffer)
            ::mmioWrite(m_hFile,
soundbuffer, cbLength);
        if(soundbuffer)
            delete (BYTE*) soundbuffer;
        if(lpHdr)
            delete lpHdr;
    }
    return ERROR_SUCCESS;
}

LRESULT CWriteSoundFile::CloseSoundFile(WPARAM wParam,
LPARAM lParam)
{
    if(m_hFile)
    {
        ::mmioAscend(m_hFile,
&m_MMCKInfoChild, 0);
        ::mmioAscend(m_hFile,
&m_MMCKInfoParent, 0);
        ::mmioClose(m_hFile, 0);
        m_hFile = NULL;
    }
    return ERROR_SUCCESS;
}

LRESULT CWriteSoundFile::OnEndThread(WPARAM wParam,
LPARAM lParam)
{
    CloseSoundFile(0,0);
    ::PostQuitMessage(0);
    return TRUE;
}

////////////////////////////////////
// CWriteSoundFile message handlers
//End of file.

```

```

// Project: Voice Recording for Digital Automatic Answering
//      Machine via Internet Network.
// Date:  Dec 19, 2002
// File:  WriteWaveFile.h
#ifdef
#ifndef(AFX_WRITESOUNDFILE_H_A0EDF320_057F_11D2_
A421_E60ECC112860_INCLUDED_)
#define
AFX_WRITESOUNDFILE_H_A0EDF320_057F_11D2_A421_E
0ECC112860_INCLUDED_
#ifdef _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// WriteSoundFile.h : header file
#include <mmsystem.h>
#define WM_WRITESOUNDFILE_FILENAME WM_USER+700
#define WM_WRITESOUNDFILE_WRITEBLOCK
WM_USER+701
#define WM_WRITESOUNDFILE_CLOSEFILE WM_USER+702
#define WM_WRITESOUNDFILE_ENDTHREAD
WM_USER+703
typedef struct writesoundfile_tag {
    char lpszFileName[MAX_PATH];
    WAVEFORMATEX waveFormatEx;
    TCHAR buffer[100];
} WRITESOUNDFILE, *PWRITESOUNDFILE;
////////////////////////////////////
// CWriteSoundFile thread
class CWriteSoundFile : public CWinThread
{
    DECLARE_DYNCREATE(CWriteSoundFile)
public:
    CWriteSoundFile();
protected:
    // protected constructor used by dynamic creation
    // Attributes
public:
    // Operations
public:
    // Overrides

    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CWriteSoundFile)

```

```

public:
virtual BOOL InitInstance();
virtual int ExitInstance();
//}}AFX_VIRTUAL

// Implementation
protected:
virtual ~CWriteSoundFile();
BOOL IsSpeech();
// Generated message map functions
//{{AFX_MSG(CWriteSoundFile)
// NOTE - the ClassWizard will add and remove member functions
//here.
//}}AFX_MSG
afx_msg LRESULT CreateWaveFile(WPARAM
wParam, LPARAM lParam);
afx_msg LRESULT WriteToSoundFile(WPARAM
wParam, LPARAM lParam);
afx_msg LRESULT CloseSoundFile(WPARAM
wParam, LPARAM lParam);
afx_msg LRESULT OnEndThread(WPARAM wParam,
LPARAM lParam);
DECLARE_MESSAGE_MAP()
HMMIO m_hFile;
MMCKINFO m_MMCKInfoData;
MMCKINFO m_MMCKInfoParent;
MMCKINFO m_MMCKInfoChild;

protected:
};
///////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous //line.
#endif//
#ifdef AFX_WRITESOUNDFILE_H__A0EDF320_057F_11D2_
A421_E60ECC112860__INCLUDED_
//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
// Machine via Internet Network
// Date: Dec 19, 2002
// File: serial_mon.cpp
#include "stdafx.h"
#include "serial_mon.h"
#include "serialspy.h"
GCommMonitor::GCommMonitor()
{
m_hConn = NULL;
m_hMPP = NULL;
m_pConn = NULL;
m_pMPP = NULL;
}
GCommMonitor::~GCommMonitor()
{
if(m_pConn && m_pConn->bConnected)
{
TRACE(_T("Warning: terminating COMM connection in
GCommMonitor::~GCommMonitor(\n"));
Disconnect();
}
if(m_hConn)
{
::GlobalUnlock(m_hConn);
::GlobalFree(m_hConn);
}
if(m_hMPP)
{
::GlobalUnlock(m_hMPP);
::GlobalFree(m_hMPP);
}
}
BOOL GCommMonitor::IsConnected() const
{
return (m_pConn && m_pConn->bConnected);
}
BOOL GCommMonitor::Connect(LPTTYSTRUCT lpTTY)
{
if(m_pConn && m_pConn->bConnected)
return TRUE;
if(!m_hConn)

```

```

{
    m_hConn = ::GlobalAlloc(GMEM_MOVEABLE, sizeof
(CONNECTION));
    m_pConn = (LPCONNECTION)::GlobalLock(m_hConn);
    ZeroMemory(m_pConn, sizeof(CONNECTION));
}

// Create I/O event used for overlapped reads/writes
if(m_pConn->osRead.hEvent = ::CreateEvent(NULL, TRUE,
FALSE, NULL)) == NULL)
    return FALSE;

if(m_pConn->osWrite.hEvent = ::CreateEvent(NULL, TRUE,
FALSE, NULL)) == NULL)
    return FALSE;

m_pConn->hCommDev = ::CreateFile(strPort,
GENERIC_READ|GENERIC_WRITE, 0,
    NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL|FILE_FLAG_OVERLAPPED,
    NULL);

if(m_pConn->hCommDev == INVALID_HANDLE_VALUE)
    return FALSE;

BOOL bPurge = FALSE;
// Get any early notifications
if(::SetCommMask(m_pConn->hCommDev, EV_RXCHAR))
{
    // Setup device buffers
    if(::SetupComm(m_pConn->hCommDev, 4096, 4096))
    {
        // Purge any information in the buffer
        if(::PurgeComm(m_pConn->hCommDev,
PURGE_TXABORT|PURGE_RXABORT|
PURGE_TXCLEAR|PURGE_RXCLEAR))
        {
            bPurge = TRUE;
        }
    }
}

if(!bPurge)
{
    ::CloseHandle(m_pConn->hCommDev);
    return FALSE;
}

// Set up for overlapped I/O

```

```

COMMTIMEOUTS CommTimeOuts;
CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF;
CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
CommTimeOuts.ReadTotalTimeoutConstant = 1000;
CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
CommTimeOuts.WriteTotalTimeoutConstant = 1000;
if(!::SetCommTimeouts(m_pConn->hCommDev,
&CommTimeOuts))
{
    ::CloseHandle(m_pConn->hCommDev);
    return FALSE;
}

// Set up the connect
DCB dcb;
ZERO_MEMORY(dcb);
dcb.DCBlength = sizeof(DCB);
if(!::GetCommState(m_pConn->hCommDev, &dcb))
{
    ::CloseHandle(m_pConn->hCommDev);
    return FALSE;
}

dcb.BaudRate = lpTTY->dwBaudRate;
dcb.ByteSize = lpTTY->byByteSize;
dcb.Parity = lpTTY->byParity;
dcb.StopBits = lpTTY->byStopBits;
// Setup hardware flow control
BOOL bSet = (BYTE)((lpTTY->byFlowCtrl & FC_DTRDSR) !=
0);

dcb.fOutxDsrFlow = bSet;
if(bSet)
    dcb.fDtrControl = DTR_CONTROL_HANDSHAKE;
else
    dcb.fDtrControl = DTR_CONTROL_ENABLE;
bSet = (BYTE)((lpTTY->byFlowCtrl & FC_RTSCTS) != 0);
dcb.fOutxCtsFlow = bSet;
if(bSet)
    dcb.fRtsControl = RTS_CONTROL_HANDSHAKE;
else
    dcb.fRtsControl = RTS_CONTROL_ENABLE;
// Setup software flow control
bSet = (BYTE)((lpTTY->byFlowCtrl & FC_XONXOFF) != 0);
dcb.fInX = dcb.fOutX = bSet;

```

```

dcb.XonChar = ASCII_XON;
dcb.XoffChar = ASCII_XOFF;
dcb.XonLim = 100;
dcb.XoffLim = 100;
// other various settings
dcb.fBinary = TRUE;
dcb.fParity = TRUE;
if(!::SetCommState(m_pConn->hCommDev, &dcb))
{
    ::CloseHandle(m_pConn->hCommDev);
    return FALSE;
}
m_pConn->bConnected = TRUE;
return m_pConn->bConnected;
}

BOOL GCommMonitor::Disconnect()
{
    if(!m_pConn)
        return TRUE;

    // Set connected flag to false
    m_pConn->bConnected = FALSE;

    // Disable event notification and wait for thread to halt
    SetCommMask(m_pConn->hCommDev, 0);
    EscapeCommFunction(m_pConn->hCommDev, CLRDR);
    // Purge any outstanding reads/writes and close device handle
    PurgeComm(m_pConn->hCommDev,
PURGE_TXABORT|PURGE_RXABORT|
        PURGE_TXCLEAR|PURGE_RXCLEAR);
    ::CloseHandle(m_pConn->hCommDev);
    ::CloseHandle(m_pConn->osRead.hEvent);
    ::CloseHandle(m_pConn->osWrite.hEvent);
    return TRUE;
}

BOOL GCommMonitor::Monitor(LPMONITORPROC
lpMonitorProc, LPARAM lParam)
{
    BOOL bMonitor = FALSE;

    // Create monitoring thread
    if(m_pConn->bConnected)
    {
        if(lpMonitorProc)
        {

```

```

            if(!m_hMPP)
            {
                m_hMPP = ::GlobalAlloc(GMEM_MOVEABLE, sizeof
(MONITORPROCPARAMS));

                m_pMPP = (LPMONITORPROCPARAMS)::GlobalLock
(m_hMPP);
            }
            m_pMPP->lpConn = m_pConn;
            m_pMPP->lpCallback = lpMonitorProc;
            m_pMPP->lpCallbackParam = lParam;
            // Create a secondary thread to watch for an event
            DWORD dwThreadId;
            if(::CreateThread(NULL, 0,
(LPTHREAD_START_ROUTINE)CommMonitorProc,
                (LPVOID)m_pMPP, 0, &dwThreadId))
            {
                m_pConn->dwThreadId = dwThreadId;
                ::EscapeCommFunction(m_pConn->hCommDev, SETDTR);
                bMonitor = TRUE;
            }
        }
    }

    return bMonitor;
}

DWORD CALLBACK GCommMonitor::CommMonitorProc
(LPVOID lpThreadParameter)
{
    HGLOBAL hParams = (HGLOBAL)lpThreadParameter;
    LPMONITORPROCPARAMS pParams =
(LPMONITORPROCPARAMS)::GlobalLock(hParams);
    if(!pParams)
        return 0;

    HGLOBAL hBuf = ::GlobalAlloc(GMEM_MOVEABLE,
MAX_RXBLOCK + 1);
    if(!hBuf)
    {
        ::GlobalUnlock(hParams);
        return 0;
    }

    LPBYTE lpBuffer = (LPBYTE)::GlobalLock(hBuf);
    if(!lpBuffer)
    {

```

```

::GlobalUnlock(hParams);
::GlobalFree(hBuf);
return 0;
}

// Create I/O event used for overlapped read
OVERLAPPED os;
::ZeroMemory(&os, sizeof(os));
os.hEvent = ::CreateEvent(NULL, TRUE, FALSE, NULL);
if(os.hEvent == NULL || !SetCommMask(pParams->lpConn->
hCommDev, EV_RXCHAR))
{
    ReportLastError();
    return 0;
}
while(pParams->lpConn->bConnected)
{
    DWORD dwEvtMask = 0;
    int nLength = 0;
    OVERLAPPED ol;
    ::ZeroMemory(&ol, sizeof(ol));
    ol.hEvent = os.hEvent;
    WaitCommEvent(pParams->lpConn->hCommDev,
&dwEvtMask, &ol);
    if((dwEvtMask & EV_RXCHAR) == EV_RXCHAR)
    {
        do
        {
            if((nLength = CommReadBlock(pParams->lpConn, lpBuffer,
MAX_RXBLOCK)) != 0)
            {
                if(pParams->lpCallback)
                    pParams->lpCallback(hBuf, nLength, pParams->
lpCallbackParam);
            }
        } while(nLength > 0);
    }
}
::GlobalUnlock(hBuf);
::GlobalFree(hBuf);
// Get rid of event handles
::CloseHandle(os.hEvent);

```

```

// Clear information in structure (indicates the thread has
//terminated)
pParams->lpConn->dwThreadId = NULL;
::GlobalUnlock(hParams);
return 1;
}

BOOL CALLBACK GCommMonitor::CommReadBlock
(LPCONNECTION lpConn, LPBYTE lpszBlock, int nLen)
{
    // Only try to read number of bytes in queue
    DWORD dwErrorFlags;
    COMSTAT ComStat;
    ::ClearCommError(lpConn->hCommDev, &dwErrorFlags,
&ComStat);
    DWORD dwLength = min((DWORD)nLen, ComStat.cbInQue);
    if(dwLength > 0)
    {
        BOOL bReadStat = ReadFile(lpConn->hCommDev, lpszBlock,
dwLength,
                                &dwLength, &lpConn->osRead);
        if(!bReadStat)
        {
            if(GetLastError() == ERROR_IO_PENDING)
            {
                OutputDebugString("\n\rIO Pending");
                // We have to wait for read to complete. This function will time
                while(!GetOverlappedResult(lpConn->hCommDev,
&lpConn->osRead,
                                &dwLength, TRUE))
                {
                    DWORD dwError = GetLastError();
                    if(dwError == ERROR_IO_INCOMPLETE)
                    {
                        // Normal result if not finished
                        continue;
                    }
                    else
                    {
                        // An error occurred, try to recover
                        ::ClearCommError(lpConn->hCommDev,
&dwErrorFlags, &ComStat);
                        if(dwErrorFlags > 0)

```

```

        break;
    }
}
else
{
    // Some other error occurred
    dwLength = 0;
    ClearCommError(lpConn->hCommDev, &dwErrorFlags,
&ComStat);
    if(dwErrorFlags > 0)
    {
    }
}
return dwLength;
}
//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network
// Date:    Dec 19, 2002
// File:    serial_mon.h
#ifndef _INC_COMM_MON_
#define _INC_COMM_MON_
#define ZERO_MEMORY(s) ::ZeroMemory(&s, sizeof(s))
#define MAX_RXBLOCK 80
// flow control
#define FC_DTRDSR 0x01
#define FC_RTSCS 0x02
#define FC_XONXOFF 0x04
#define ASCII_XON 0x11
#define ASCII_XOFF 0x13
typedef struct _TTYSTRUCT
{
    BYTE byCommPort;
// zero based port - 3 or higher implies TELNET
    BYTE byXonXoff;
    BYTE byByteSize;
    BYTE byFlowCtrl;
    BYTE byParity;
    BYTE byStopBits;
    DWORD dwBaudRate;
} TTYSTRUCT, *LPTTYSTRUCT;
typedef struct _CONNECTION
{
    HANDLE hCommDev;
    BOOL bConnected;
    OVERLAPPED osWrite;
    OVERLAPPED osRead;
    DWORD dwThreadId;
} CONNECTION, *LPCONNECTION;
typedef int CALLBACK MONITORPROC(HGLOBAL, int,
LPARAM);
typedef MONITORPROC *LPMONITORPROC;
typedef struct _MONITORPROCPARMS
{
    LPCONNECTION lpConn;
    LPMONITORPROC lpCallback;
    LPARAM lpCallbackParam;
} MONITORPROCPARMS, *LPMONITORPROCPARMS;

```

```

class GCommMonitor
{
public:
    GCommMonitor();

    // connect to a comm port
    BOOL Connect(LPTTYSTRUCT lpTTY);

    // disconnect
    BOOL Disconnect();

    // monitor (read) data at open connection
    BOOL Monitor(LPMONITORPROC lpMonitorProc, LPARAM
lpParam=NULL);

    BOOL IsConnected() const;
protected:
    // thread entry point
    static DWORD CALLBACK CommMonitorProc(LPVOID
lpThreadParameter);

    // helper function to monitor comm port
    static BOOL CALLBACK CommReadBlock(LPCONNECTION
lpConn, LPBYTE lpszBlock, int nLen);

    // information about the current connection and
    // monitoring thread
    LPCONNECTION m_pConn;
    HGLOBAL m_hConn;
    LPMONITORPROC_PARAMS m_pMPP;
    HGLOBAL m_hMPP;
public:
    virtual ~GCommMonitor();
};

#endif

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network
// Date:    Dec 19, 2002
// File:    serialspy.cpp
// Title:    Defines the class behaviors for the application.

#include "stdafx.h"
#include "serialspy.h"
#include "cs_dlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CCommSpyApp
BEGIN_MESSAGE_MAP(CCommSpyApp, CWinApp)
   //{{AFX_MSG_MAP(CCommSpyApp)
// NOTE - the ClassWizard will add and remove mapping macros
//here.
// DO NOT EDIT what you see in these blocks of generated code
    //}}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

////////////////////////////////////
// CCommSpyApp construction
CCommSpyApp::CCommSpyApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

////////////////////////////////////
// The one and only CCommSpyApp object
CCommSpyApp theApp;    ///Mian Object
////////////////////////////////////
// CCommSpyApp initialization
BOOL CCommSpyApp::InitInstance()
{
    AfxEnableControlContainer();

    // Standard initialization

#ifdef _AFXDLL
    Enable3dControls();
// Call this when using MFC in a shared DLL

```

```

#else
    Enable3dControlsStatic();
// Call this when linking to MFC statically
#endif
SetRegistryKey("Commspy");
    CCommspyDlg dlg;
    m_pMainWnd = &dlg; // main windows
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
// TODO: Place code here to handle when the dialog is dismissed
//with OK
    }
    else if (nResponse == IDCANCEL)
    {
// TODO: Place code here to handle when the dialog is
// dismissed with Cancel
    }
    return FALSE;
}

void ReportLastError()
{
    LPVOID lpMsgBuf = NULL;
    BOOL bFormat =
::FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER|F
ORMAT_MESSAGE_FROM_SYSTEM,
    NULL, GetLastError(),
    MAKELANGID(LANG_NEUTRAL,
    SUBLANG_DEFAULT),
    (LPTSTR) &lpMsgBuf, 0, NULL);
    if (bFormat && lpMsgBuf)
    {
        AfxMessageBox((LPCTSTR)lpMsgBuf);
        LocalFree(lpMsgBuf);
    }
}

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network
// Date:  Dec 19, 2002
// File:  serialspy.h
// Title:  Main header file for the serialspy application
#ifndef _INC_COMMSPY_
#define _INC_COMMSPY_
#include "resource.h"

void ReportLastError();
////////////////////////////////////
// CCommspyApp:
class CCommspyApp : public CWinApp
{
public:
    CCommspyApp();

public:
    virtual BOOL InitInstance();
    DECLARE_MESSAGE_MAP()
};

#endif
//End of file.

```



```

BOOL CCommspyDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_fontDisplay.CreatePointFont(m_settings.nPointSize,
m_settings.strFontName);
    m_display.SetFont(&m_fontDisplay);
    m_display.LimitText(LONG_MAX);
    m_display.GetWindowRect(&m_rectDisplayStop);
    ScreenToClient(&m_rectDisplayStop);
    m_rectDisplayStart = m_rectDisplayStop;
    CRect rectEdge;
    GetDlgItem(IDC_GROUP_DISPLAY)->GetWindowRect
(&rectEdge);
    ScreenToClient(&rectEdge);
    m_rectDisplayStart.right = rectEdge.right;
    // update background color control
    CRect rect;
    CWnd *pWnd = GetDlgItem(IDC_COLOR_BK);
    pWnd->SetDlgCtrlID(0xffff);
    pWnd->GetWindowRect(rect);
    ScreenToClient(&rect);
    m_colorBK.SetColor(m_settings.clBK);
    m_colorBK.Create(WS_VISIBLE|WS_CHILD, rect, this,
IDC_COLOR_BK);
    // update foreground color control
    pWnd = GetDlgItem(IDC_COLOR_FG);
    pWnd->GetWindowRect(rect);
    pWnd->SetDlgCtrlID(0xffff);
    ScreenToClient(&rect);
    m_colorFG.SetColor(m_settings.clFG);
    m_colorFG.Create(WS_VISIBLE|WS_CHILD, rect, this,
IDC_COLOR_FG);

    // Add "About..." menu item to system menu.
    // IDM_ABOUTBOX must be in the system command
range.

    ASSERT((IDM_ABOUTBOX & 0xFFF0) ==
IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);
    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {

```

```

        CString strAboutMenu;
        strAboutMenu.LoadString
(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu
(MF_SEPARATOR);
            pSysMenu->AppendMenu
(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }
    SetIcon(m_hIcon, TRUE);
    SetIcon(m_hIcon, FALSE);

    // Reposition the window
    UINT nFlags = SWP_NOSIZE;
    CPoint pt(0,0);
    if(m_settings.dwPos == 0xFFFFFFFF)
        nFlags |= SWP_NOMOVE;
    else
        pt = CPoint(m_settings.dwPos);
    const CWnd *pWndInsertAfter = NULL;
    if(m_settings.bOnTop)
        pWndInsertAfter = &wndTopMost;
    SetWindowPos(pWndInsertAfter, pt.x, pt.y, 0, 0, nFlags);
    return TRUE;
}

void CCommspyDlg::OnClose()
{
    if(m_comm.IsConnected() && !m_comm.Disconnect())
        return;
    UpdateData();
    CRect wr;
    GetWindowRect(&wr);
    m_settings.dwPos = MAKELPARAM(wr.left, wr.top);
    StoreSettings();
    CDialog::OnClose();
}

void CCommspyDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this);

```

```

// Device context for painting
        SendMessage(WM_ICONERASEBKGND,
(WPARAM) dc.GetSafeHdc(), 0);

        int cxIcon = GetSystemMetrics
(SM_CXICON);

        int cyIcon = GetSystemMetrics
(SM_CYICON);

        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;
        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

HCURSOR CCommSpyDlg::OnQueryDragIcon()
{
    return (HCURSOR)m_hIcon;
}

void CCommSpyDlg::OnFont()
{
    CFontDialog dlg;
    if(dlg.DoModal() == IDOK)
    {
        m_fontDisplay.DeleteObject();
        m_settings.nPointSize = dlg.GetSize();
        m_settings.strFontName = dlg.GetFaceName();
        m_fontDisplay.CreatePointFont(m_settings.nPointSize,
m_settings.strFontName);
        m_display.SetFont(&m_fontDisplay);
    }
}

int _nDataValues[] = {5,6,7,8};
int _nBaudRates[] = {CBR_110, CBR_300, CBR_600, CBR_1200,
CBR_2400,
        CBR_4800, CBR_9600, CBR_14400, CBR_19200,
        CBR_38400, CBR_56000, CBR_57600, CBR_115200,
        CBR_128000, CBR_256000};

```

```

void CCommSpyDlg::OnStart()
{
    BOOL bDisconnect = m_comm.IsConnected();
    if(!bDisconnect)
    {
        UpdateData();
        // Initialize info
        TTYSTRUCT tty;
        ZERO_MEMORY(tty);
        tty.byCommPort = (BYTE)m_settings.nPort;
        tty.byXonXoff = FALSE;
        tty.byByteSize = (BYTE)_nDataValues[m_settings.nData];
        tty.byFlowCtrl = (BYTE)m_settings.nFlow;;
        tty.byParity = (BYTE)m_settings.nParity;;
        tty.byStopBits = (BYTE)m_settings.nStop;;
        tty.dwBaudRate = (DWORD)_nBaudRates[m_settings.nBaud];
        if(m_comm.Connect(&tty))
        {
            ShowControls(SW_HIDE, &m_rectDisplayStart);
            MSG msg;
            while(PeekMessage(&msg, NULL, WM_PAINT,
WM_PAINT, PM_NOREMOVE))
            {
                ::TranslateMessage(&msg);
                ::DispatchMessage(&msg);
            }
            // Begin monitoring
            if(!m_comm.Monitor(&MonitorCallback, (LPARAM)
m_hWnd))
            {
                bDisconnect = TRUE;
            }
        }
        else
        {
            ReportLastError();
        }
    }
    if(bDisconnect)
    {
        m_comm.Disconnect();
        ShowControls(SW_SHOW, &m_rectDisplayStop);
    }
}

```

```

    }
}

void CCommspyDlg::ShowControls(int nShow, CRect
*pNewDisplayRect)
{
    int nControls[] = {IDC_COLOR_FG, IDC_COLOR_BK,
IDC_BAUD, IDC_DATA, IDC_PARITY,
        IDC_STOP, IDC_PORT, IDC_FONT, IDC_HEX,
IDC_DECIMAL,
        IDC_OCTAL, IDC_ASCII, IDC_ONTOP};

    int nCount = sizeof(nControls)/sizeof(nControls[0]);
    for(int i = 0; i < nCount; i++)
    {
        CWnd *pWnd = GetDlgItem(nControls[i]);
        if(pWnd)
            pWnd->ShowWindow(nShow);
    }
    if(pNewDisplayRect)
    {
        m_display.MoveWindow(pNewDisplayRect);
    }
    CString strStartStop(nShow == SW_SHOW ? _T("Start!") : _T
("Stop!"));
    CWnd *pWnd = GetDlgItem(IDC_START);
    if(pWnd)
        pWnd->SetWindowText(strStartStop);
}

LRESULT CCommspyDlg::OnRX(WPARAM wLen, LPARAM
hBuf)
{
    LPBYTE lpBuf = (LPBYTE)::GlobalLock((HGLOBAL)hBuf);
    if(!lpBuf)
        return 1;

    CString strOutput;
    if(m_settings.nFormat == IDC_ASCII)
    {
        strOutput = lpBuf;
    }
    else
    {
        CString strCvt;
        for(int i = 0; i < (int)wLen; i++)

```

```

    {
        switch(m_settings.nFormat)
        {
            case IDC_DECIMAL:
                strCvt.Format("%3d ", (BYTE)lpBuf[i]);
                break;

            case IDC_OCTAL:
                strCvt.Format("%3o ", (BYTE)lpBuf[i]);
                break;

            default:
                strCvt.Format("%02x ", (BYTE)lpBuf[i]);
                break;
        }

        strOutput += strCvt;
    }
}

//
//start1
/* strOutput += "\r\n";
m_display.SetSel(0,0);
m_display.ReplaceSel(strOutput);
if (strOutput == "70")
    MessageBox("Match");
else
    MessageBox(" not match");
m_display.SetSel(0,0);
strOutput += "\r\n";
m_display.SetSel(0,0);
m_display.ReplaceSel(strOutput);
if (!strcmp(strOutput, "r\r\n")) //compare TRUE ---> record voice
{
    WinExec("record.exe", SW_SHOW); //Open program
RECORD VOICE
    m_display.SetSel(0,0);
    m_display.ReplaceSel("Recording Voice Mail...\r\n");
}
else
{
    m_display.SetSel(0,0);
    m_display.ReplaceSel("Unknown data...\r\n");
}
m_display.SetSel(0,0);

```

```

strOutput=""; //clear
::GlobalUnlock((HGLOBAL)hBuf);

return 1;
}

int CALLBACK MonitorCallback(HGLOBAL hBuf, int nLen,
LPARAM lParam)
{
    HWND hDlg = (HWND)lParam;
    ::SendMessage(hDlg, IDM_RX, nLen, (LPARAM)hBuf);

    return 1;
}

void CCommspyDlg::OnOnTop()
{
    UpdateData();
    if(m_settings.bOnTop)
    {
        SetWindowPos(&wndTopMost, 0, 0, 0, 0,
SWP_NOMOVE|SWP_NOSIZE);
    }
    else
    {
        SetWindowPos(&wndBottom, 0, 0, 0, 0,
SWP_NOMOVE|SWP_NOSIZE|SWP_NOREDRAW);
        BringWindowToTop();
    }
}

void CCommspyDlg::OnClear()
{
    m_display.SetWindowText("");
}

HBRUSH CCommspyDlg::OnCtlColor(CDC* pDC, CWnd* pWnd,
UINT nCtlColor)
{
    if(pWnd->m_hWnd == m_display.m_hWnd)
    {
        pDC->SetTextColor(m_settings.clFG);
        pDC->SetBkColor(m_settings.clBK);
        if(m_brushDisplay.m_hObject)
            m_brushDisplay.DeleteObject();
        m_brushDisplay.CreateSolidBrush(m_settings.clBK);
        return m_brushDisplay;
    }

    return CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
}

void CCommspyDlg::OnBkColorChange()
{
    m_settings.clBK = m_colorBK.GetColor();
    m_display.Invalidate();
}

void CCommspyDlg::OnFgColorChange()
{
    m_settings.clFG = m_colorFG.GetColor();
    m_display.Invalidate();
}

void CCommspyDlg::LoadSettings()
{
    CWinApp *pApp = AfxGetApp();
    m_settings.clBK = (COLORREF)pApp->GetProfileInt
(CS_REGKEY_SETTINGS, CS_REGENTRY_BKCOLOR, (int)
RGB(0,0,255));
    m_settings.clFG = (COLORREF)pApp->GetProfileInt
(CS_REGKEY_SETTINGS, CS_REGENTRY_FGCOLOR, (int)
RGB(0,255,255));
    m_settings.nPointSize = pApp->GetProfileInt
(CS_REGKEY_SETTINGS, CS_REGENTRY_POINTSIZE, 90);
    m_settings.strFontName = pApp->GetProfileString
(CS_REGKEY_SETTINGS, CS_REGENTRY_FACENAME, _T
("Terminal"));
    m_settings.bOnTop = pApp-
>GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_ONTOP, FALSE);
    m_settings.nPort = pApp-
>GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_PORT, 1);
    m_settings.nParity = pApp-
>GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_PARITY, 0);
    m_settings.nBaud = pApp-
>GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_BAUD, 11);
    m_settings.nData = pApp-
>GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_DATABITS, 3);
}

```

```

    m_settings.nStop = pApp->GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_STOPBITS, 0);

    m_settings.nFlow = pApp->GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_FLOW, 0);

    m_settings.nFormat = pApp->GetProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_FORMAT, IDC_HEX);

    m_settings.dwPos = (DWORD)pApp->GetProfileInt
(CS_REGKEY_SETTINGS, CS_REGENTRY_POS,
0xFFFFFFFF);
}

void CCommspyDlg::StoreSettings()
{
    CWinApp *pApp = AfxGetApp();

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_BKCOLOR, (int)m_settings.clBK);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_FGCOLOR, (int)m_settings.clFG);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_POINTSIZE, (int)m_settings.nPointSize);

    pApp->WriteProfileString(CS_REGKEY_SETTINGS,
CS_REGENTRY_FACENAME, m_settings.strFontName);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_ONTOP, m_settings.bOnTop);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_PORT, m_settings.nPort);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_PARITY, m_settings.nParity);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_BAUD, m_settings.nBaud);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_DATABITS, m_settings.nData);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_STOPBITS, m_settings.nStop);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_FLOW, m_settings.nFlow);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_FORMAT, m_settings.nFormat);

    pApp->WriteProfileInt(CS_REGKEY_SETTINGS,
CS_REGENTRY_POS, m_settings.dwPos);

```

```

}

void CCommspyDlg::OnSysCommand(UINT nID, LPARAM
lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CDialog dlg(IDD_ABOUTBOX);
        dlg.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

//End of file.

```

```
// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   cs_dlg.h
// Title:   Header file
#ifndef _INC_CS_DLG_
#define _INC_CS_DLG_
#include "util_cir.h"
#include "serial_mon.h"
typedef struct _CSSETTINGS
{
    COLORREF clBK;
    COLORREF clFG;
    DWORD dwPos;
    int nPointSize;
    CString strFontName;
    BOOL bOnTop;
    int nFormat;
    int nPort;
    int nParity;
    int nBaud;
    int nData;
    int nStop;
    int nFlow;
} CSSETTINGS, *LPCSETTINGS;
// Registry stuff
#define CS_REGKEY_SETTINGS    _T("Settings")
#define CS_REGENTRY_BKCOLOR   _T("okcolor")
#define CS_REGENTRY_FGCOLOR   _T("fgcolor")
#define CS_REGENTRY_POINTSIZ _T("pointsize")
#define CS_REGENTRY_FACENAME  _T("facename")
#define CS_REGENTRY_ONTOP     _T("ontop")
#define CS_REGENTRY_PORT      _T("port")
#define CS_REGENTRY_PARITY     _T("parity")
#define CS_REGENTRY_BAUD      _T("baud")
#define CS_REGENTRY_DATABITS   _T("databits")
#define CS_REGENTRY_STOPBITS   _T("stopbits")
#define CS_REGENTRY_FLOW       _T("flow")
#define CS_REGENTRY_FORMAT     _T("format")
#define CS_REGENTRY_POS        _T("pos")
////////////////////////////////////
// CCommspyDlg dialog
```

```
class CCommspyDlg : public CDialog
{
// Construction
public:
    // Standard constructor
    CCommspyDlg(CWnd* pParent = NULL);

protected:
    // DDX/DDV support
    virtual void DoDataExchange(CDataExchange* pDX);
    void ShowControls(int nShow, CRect *pNewDisplayRect);
    void LoadSettings();
    void StoreSettings();
    // Called when data received from communication port
    LRESULT OnRX(WPARAM wLen, LPARAM lBuf);

// Implementation
protected:
    HICON m_hIcon;
    CRect m_rectDisplayStart;
    CRect m_rectDisplayStop;
    // Subclassed controls
    GColorComboBox m_colorBK;
    GColorComboBox m_colorFG;
    CCheckBox m_flow;
    CEdit m_display;
    // Brushes, fonts, settings
    CFont m_fontDisplay;
    CBrush m_brushDisplay;
    CSSETTINGS m_settings;
    // Monitoring object
    GCommMonitor m_comm;

    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();

    afx_msg void OnFont();
    afx_msg void OnStart();
    afx_msg void OnStop();
    afx_msg void OnOnTop();
    afx_msg void OnClose();
    afx_msg void OnClear();
```

```

afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd *pWnd, UINT
nCtlColor);

afx_msg void OnBkColorChange();

afx_msg void OnFgColorChange();

    DECLARE_MESSAGE_MAP()
};

#endif

//End of file.

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network
// Date:   Dec 19, 2002
// File:   resource.h
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by commspy.rc

#define IDM_ABOUTBOX                0x0010
#define IDD_ABOUTBOX                100
#define IDS_ABOUTBOX                101
#define IDD_COMMSPY_DIALOG          102
#define IDR_MAINFRAME               128
#define IDB_BITMAP_BLACK            129
#define IDB_BITMAP_DKGRAY          130
#define IDC_DISPLAY                  1000
#define IDC_START                    1001
#define IDC_COLOR_BK                 1002
#define IDC_COLOR_FG                 1003
#define IDC_BAUD                     1004
#define IDC_DATA                     1005
#define IDC_PARITY                    1006
#define IDC_STOP                     1007
#define IDC_STOPBITS                 1007
#define IDC_PORT                     1008
#define IDC_FONT                     1009
#define IDC_CLEAR                    1010
#define IDC_HEX                      1011
#define IDC_DECIMAL                  1012
#define IDC_OCTAL                    1013
#define IDC_ASCII                    1014
#define IDC_RESUME_SCROLL            1015
#define IDC_TRIGGER                  1016
#define IDC_PRINT                    1017
#define IDC_GROUP_DISPLAY            1018
#define IDC_ONTOP                    1019
#define IDC_BUTTON1                  1020
#define IDC_FLOW                     1022
#define IDOK                         1025
#define IDCANCEL                     1026
#define IDC_EDIT1                    1027

// Next default values for new objects
#ifndef APSTUDIO_INVOKED

```

```

#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        132
#define _APS_NEXT_COMMAND_VALUE        32771
#define _APS_NEXT_CONTROL_VALUE        1028
#define _APS_NEXT_SYMED_VALUE          101
#endif
#endif

//End of file.

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network
// Date:    Dec 19, 2002
// File:    stdafx.cpp
// Title:    Source file that includes just the standard includes
//          commspy.pch will be the pre-compiled header
//          stdafx.obj will contain the pre-compiled type information
#include "stdafx.h"

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   stdafx.h
//Title:   Include file for standard system include files
#if
!defined(AFX_STDAFX_H__4C9D49E6_9A7C_11D1_B12E_00
C04FD636E6__INCLUDED_)
#define
AFX_STDAFX_H__4C9D49E6_9A7C_11D1_B12E_00C04FD63
6E6__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#define VC_EXTRALEAN    // Exclude rarely-used stuff from
Windows headers
#include <afxwin.h>    // MFC core and standard components
#include <afxext.h>    // MFC extensions
#include <afxdisp.h>    // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>
// MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
// 4100 = 'identifier' : unreferenced formal parameter
#pragma warning ( disable : 4100 )
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
//immediately before the previous //line.
#endif //
!defined(AFX_STDAFX_H__4C9D49E6_9A7C_11D1_B12E_00
C04FD636E6__INCLUDED_)
//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   util_clr.cpp
#include "stdafx.h"
#include "util_clr.h"
#include "util_mdc.h"
#include "util_gdi.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
GColorComboBox::GColorComboBox()
{
    m_cl = RGB(0,0,0);
}
GColorComboBox::~GColorComboBox()
{
}
void GColorComboBox::SetColor(COLORREF clInit)
{
    m_cl = clInit;
}
COLORREF GColorComboBox::GetColor() const
{
    return m_cl;
}
void GColorComboBox::UpdateColor(COLORREF clUpdate)
{
    if(clUpdate != m_cl)
    {
        m_cl = clUpdate;
        Invalidate();
        int nChildID = GetDlgCtrlID();
        WPARAM wParam = MAKEWPARAM((WORD)nChildID,
CCN_SELCHANGE);
        GetParent()->SendMessage(WM_COMMAND, wParam,
(LPARAM)m_hWnd);
    }
}
BEGIN_MESSAGE_MAP(GColorComboBox, CWnd)

```

```

ON_WM_LBUTTONDOWN()

ON_WM_PAINT()

ON_WM_ERASEBKGD()

END_MESSAGE_MAP()

BOOL GColorComboBox::Create(DWORD dwStyle, const
RECT& rect,

        CWnd *pParentWnd, UINT nID)
{
    return CWnd::Create(NULL, NULL, dwStyle, rect, pParentWnd,
nID);
}

void GColorComboBox::OnLButtonDown(UINT nFlags, CPoint
pt)
{
    CRect cr;

    GetClientRect(cr);

    cr.right -= CCBDIM_CXEDGE;

    cr.left = cr.right - CCBDIM_CXARROW -

        CCBDIM_CXBORDER - CCBDIM_CXSPACERIGHT;

    cr.top += CCBDIM_CYEDGE;

    cr.bottom -= CCBDIM_CYEDGE;

    if(cr.PtInRect(pt))
    {
        CRect wr;

        GetWindowRect(wr);

        WNDCLASS wndcls;

        memset(&wndcls, 0, sizeof(wndcls));

        wndcls.lpszClassName = "GPopupColorCtrl";

        wndcls.lpfnWndProc = AfxWndProc;

        wndcls.hInstance = AfxGetInstanceHandle();

        if(AfxRegisterClass(&wndcls))
        {
            GPopupColorCtrl *pPopup = new GPopupColorCtrl(m_cl,
this);

            pPopup->CreateEx(0, wndcls.lpszClassName, NULL,

WS_BORDER|WS_POPUP,

                wr.left, wr.bottom+2, 0, 0, AfxGetMainWnd()->

m_hWnd, 0, NULL);
        }
    }
}

void GColorComboBox::OnPaint()

```

```

{
    CPaintDC dc(this);

    GMemDC memDC;

    CBrush brush(GetSysColor(COLOR_BTNFACE));

    CRect cr;

    GetClientRect(cr);

    if(memDC.Create(&dc, cr, &brush))
    {
        CDC *pDC = &memDC;

        CRect rectDraw(cr);

        rectDraw.DeflateRect(1,1);

        CPen penBlack(PS_SOLID, 1, RGB(0,0,0));

        CPen penDarkGray(PS_SOLID, 1, GetSysColor
(COLOR_BTNSHADOW));

        CPen penWhite(PS_SOLID, 1, RGB(255,255,255));

        CPen penGray(PS_SOLID, 1, GetSysColor
(COLOR_BTNFACE));

        // draw the borders

        CPen *pOldPen = pDC->SelectObject(&penGray);

        pDC->MoveTo(rectDraw.left, rectDraw.top);

        pDC->LineTo(rectDraw.right, rectDraw.top);

        pDC->MoveTo(rectDraw.left, rectDraw.top);

        pDC->LineTo(rectDraw.left, rectDraw.bottom);

        pDC->SelectObject(&penWhite);

        rectDraw.DeflateRect(1, 1);

        pDC->MoveTo(rectDraw.left, rectDraw.top);

        pDC->LineTo(rectDraw.right, rectDraw.top);

        pDC->MoveTo(rectDraw.left, rectDraw.top);

        pDC->LineTo(rectDraw.left, rectDraw.bottom);

        pDC->SelectObject(&penBlack);

        rectDraw.InflateRect(1, 1);

        pDC->MoveTo(rectDraw.right, rectDraw.top);

        pDC->LineTo(rectDraw.right, rectDraw.bottom+1);

        pDC->MoveTo(rectDraw.left, rectDraw.bottom);

        pDC->LineTo(rectDraw.right, rectDraw.bottom);

        pDC->SelectObject(&penDarkGray);

        rectDraw.DeflateRect(1, 1);

        pDC->MoveTo(rectDraw.right, rectDraw.top);

        pDC->LineTo(rectDraw.right, rectDraw.bottom+1);

        pDC->MoveTo(rectDraw.left, rectDraw.bottom);

        pDC->LineTo(rectDraw.right, rectDraw.bottom);

        pDC->SelectObject(&penBlack);
    }
}

```

```

int nEnd = rectDraw.right - CCBDIM_CXEDGE -
    CCBDIM_CXBORDER;
int nStart = nEnd - CCBDIM_CXARROW;
int nMiddle = cr.top + (cr.Height()/2);
pDC->MoveTo(nStart, nMiddle-1);
pDC->LineTo(nEnd, nMiddle-1);
nStart++;
nEnd--;
pDC->MoveTo(nStart, nMiddle);
pDC->LineTo(nEnd, nMiddle);
nStart++;
nEnd--;
pDC->MoveTo(nStart, nMiddle+1);
pDC->LineTo(nEnd, nMiddle+1);
// draw dividers
nEnd = rectDraw.right - CCBDIM_CXEDGE -
    CCBDIM_CXBORDER - CCBDIM_CXARROW -
    CCBDIM_CXSPACERIGHT - 1;
pDC->SelectObject(&penWhite);
int nTop = cr.top + CCBDIM_CYEDGE +
CCBDIM_CYBORDER;
int nBottom = (cr.bottom - CCBDIM_CYEDGE) -
CCBDIM_CYBORDER;
pDC->MoveTo(nEnd, nTop);
pDC->LineTo(nEnd, nBottom);
pDC->SelectObject(&penDarkGray);
nEnd--;
pDC->MoveTo(nEnd, nTop);
pDC->LineTo(nEnd, nBottom);
// draw color rectangle
nEnd = CCBDIM_CXSPACELEFT;
CRect rectColor(cr);
rectColor.DeflateRect(1,1);
rectColor.right = nEnd;
rectColor.left += cr.left + CCBDIM_CXEDGE +
CCBDIM_CXBORDER;
rectColor.top = nTop;
rectColor.bottom = nBottom;
pDC->SelectObject(&penBlack);
CBrush brush(m_cl);
CBrush *pOldBrush = pDC->SelectObject(&brush);
pDC->Rectangle(rectColor);

```

```

pDC->SelectObject(pOldPen);
pDC->SelectObject(pOldBrush);
memDC.Copy(&dc, cr);
memDC.Release();
}
}
BOOL GColorComboBox::OnEraseBkgnd(CDC *pDC)
{
    return TRUE;
}
////////////////////////////////////
// GPopupColorCtrl
COLORREF g_clArray[] = {RGB(0,0,255), RGB(0,255,0),
    RGB(0,255,255), RGB(255,0,0),
    RGB(255,0,255), RGB(255,255,0),
    RGB(255,255,255), RGB(0,0,0),
    RGB(0,0,127), RGB(0,127,0),
    RGB(0,127,127), RGB(127,0,0),
    RGB(127,0,127), RGB(127,127,0),
    RGB(127,127,127), RGB(63,63,63)};
GPopupColorCtrl::GPopupColorCtrl(COLORREF clInit,
    GColorComboBox *pBuddyCombo)
{
    m_pBuddyCombo = pBuddyCombo;
    m_nActiveSquare = 0;
    m_nCurHoverSquare = 0;
    for(int nColor = 0; nColor < CCDIM_TOTALINDECES;
nColor++)
    {
        if(g_clArray[nColor] == clInit)
        {
            m_nActiveSquare = nColor;
            m_nCurHoverSquare = nColor;
            break;
        }
    }
    m_clSelected = g_clArray[m_nActiveSquare];
}
GPopupColorCtrl::~GPopupColorCtrl()
{
}
int GPopupColorCtrl::CursorToColorIndex(CPoint ptTest)

```

```

{
    CPoint pt(CCDIM_CXSPACE, CCDIM_CYSYSPACE);
    int nIndex = 0;
    for(int nRow = 0; nRow < 4; nRow++)
    {
        for(int nCol = 0; nCol < 4; nCol++)
        {
            CRect rectSquare(pt, CSize(CCDIM_SQUAREWIDTH,
            CCDIM_SQUAREHEIGHT));
            if(rectSquare.PtInRect(ptTest))
                return nIndex;
            pt.x += (CCDIM_SQUAREWIDTH + CCDIM_CXSPACE);
            nIndex++;
        }
        pt.x = CCDIM_CXSPACE;
        pt.y += (CCDIM_SQUAREHEIGHT + CCDIM_CYSYSPACE);
    }
    return -1;
}

void GPopupColorCtrl::DrawColorSquare(CDC *pDC, int nIndex,
CPoint pt)
{
    CRect rectDraw(pt, CSize(CCDIM_SQUAREWIDTH,
    CCDIM_SQUAREHEIGHT));
    _3DPENS pens;
    int nThickness = GfxDraw3DBorder(pDC, &pens, rectDraw);
    rectDraw.DeflateRect(nThickness, nThickness);
    CBrush brush(g_clArray[nIndex]);
    pDC->FillRect(rectDraw, &brush);
    if(nIndex == m_nCurHoverSquare)
    {
        rectDraw.InflateRect(1, 1);
        GfxDrawBorder(pDC, rectDraw, pens.pPenWhite);
        rectDraw.DeflateRect(1, 1);
        GfxDrawBorder(pDC, rectDraw, pens.pPenBlack);
    }
}

BEGIN_MESSAGE_MAP(GPopupColorCtrl, GPopupWindow)
    ON_WM_CREATE()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONDOWN()
    ON_WM_PAINT()

```

```

    ON_WM_ERASEBKGD()
END_MESSAGE_MAP()

int GPopupColorCtrl::OnCreate(LPCREATESTRUCT lpcs)
{
    int nCreate = GPopupWindow::OnCreate(lpcs);
    if(nCreate == 0)
    {
        int cx = CCDIM_CXSPACE * 5;
        cx += GetSystemMetrics(SM_CXBORDER) * 2;
        cx += (CCDIM_SQUAREWIDTH * 4);
        int cy = CCDIM_CYSYSPACE * 5;
        cy += GetSystemMetrics(SM_CYBORDER) * 2;
        cy += (CCDIM_SQUAREHEIGHT * 4);
        SetWindowPos(&wndTopMost, 0, 0, cx, cy,

        SWP_NOMOVE|SWP_NOZORDER|SWP_SHOWWINDOW|SW
        P_NOACTIVATE);
    }
    return nCreate;
}

void GPopupColorCtrl::OnPaint()
{
    CPaintDC dc(this);
    GMemDC memDC;
    CBrush brush(GetSysColor(COLOR_BTNFACE));
    CRect cr;
    GetClientRect(cr);
    if(memDC.Create(&dc, cr, &brush))
    {
        CPoint pt(CCDIM_CXSPACE, CCDIM_CYSYSPACE);
        int nIndex = 0;
        for(int nRow = 0; nRow < 4; nRow++)
        {
            for(int nCol = 0; nCol < 4; nCol++)
            {
                DrawColorSquare(&memDC, nIndex, pt);
                pt.x += (CCDIM_SQUAREWIDTH +
                CCDIM_CXSPACE);
                nIndex++;
            }
            pt.x = CCDIM_CXSPACE;
            pt.y += (CCDIM_SQUAREHEIGHT + CCDIM_CYSYSPACE);

```

```

    }
    memDC.Copy(&dc, cr);
    memDC.Release();
}
}

BOOL GPopupColorCtrl::OnEraseBkgnd(CDC *pDC)
{
    return TRUE;
}

void GPopupColorCtrl::OnMouseMove(UINT nFlags, CPoint pt)
{
    int nIndex = CursorToColorIndex(pt);
    if(nIndex != -1 && nIndex != m_nCurHoverSquare)
    {
        m_nCurHoverSquare = nIndex;
        Invalidate();
    }
}

void GPopupColorCtrl::OnLButtonUp(UINT nFlags, CPoint pt)
{
    if(!OnLButtonDownCheck(nFlags, pt))
    {
        int nIndex = CursorToColorIndex(pt);
        // find out what color mouse is over
        if(nIndex != -1)
        {
            if(m_pBuddyCombo)
                m_pBuddyCombo->UpdateColor(g_clArray[nIndex]);
            m_nActiveSquare = nIndex;
            ::ReleaseCapture();
        }
    }
}

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network.
// Date:    Dec 19, 2002
// File:    util_clr.h
#ifndef _INC_UTIL_CLR_
#define _INC_UTIL_CLR_
#include "util_pop.h"

// notifications
#define CCN_SELCHANGE 1
#define CCBDIM_CXEDGE 2
#define CCBDIM_CYEDGE 2
#define CCBDIM_CYBORDER 3
#define CCBDIM_CXBORDER 2
#define CCBDIM_CXSPACELEFT 4
#define CCBDIM_CXSPACERIGHT 1
#define CCBDIM_CXCOLORRECT 32
#define CCBDIM_CXDIVIDER 2
#define CCBDIM_CXARROW 5
class GColorComboBox : public CWnd
{
    DECLARE_MESSAGE_MAP()
public:
    GColorComboBox();
    BOOL Create(DWORD dwStyle, const RECT& rect,
                CWnd *pParentWnd, UINT nID);
    void SetColor(COLORREF clInit);
    COLORREF GetColor() const;
    virtual void UpdateColor(COLORREF clUpdate);
protected:
    afx_msg void OnLButtonDown(UINT nFlags, CPoint pt);
    afx_msg int OnCreate(LPCREATESTRUCT lpcs);
    afx_msg BOOL OnEraseBkgnd(CDC *pDC);
    afx_msg void OnPaint();
    COLORREF m_cl;
public:
    virtual ~GColorComboBox();
};

#define CCDIM_SQUAREWIDTH 16
#define CCDIM_SQUAREHEIGHT 16
#define CCDIM_CYSPEC 3
#define CCDIM_CXSPACE 3
#define CCDIM_TOTALINDECES 16

```

```

class GPopupColorCtrl : public GPopupWindow
{
    DECLARE_MESSAGE_MAP();

public:
    GPopupColorCtrl(COLORREF clInit,
                    GColorComboBox *pBuddyCombo);

    COLORREF GetSelectedColor();

protected:
    afx_msg void OnPaint();
    afx_msg BOOL OnEraseBkgnd(CDC *pDC);
    afx_msg void OnMouseMove(UINT nFlags, CPoint pt);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint pt);
    afx_msg int OnCreate(LPCREATESTRUCT lpcs);
    void DrawColorSquare(CDC *pDC, int nIndex, CPoint pt);
    int CursorToColorIndex(CPoint ptTest);
    COLORREF m_clSelected;
    int m_nActiveSquare;
    int m_nCurHoverSquare;
    GColorComboBox *m_pBuddyCombo;

public:
    virtual ~GPopupColorCtrl();
};

#endif

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network.
// Date:    Dec 19, 2002
// File:    util_gdi.cpp
#include "stdafx.h"
#include "util_gdi.h"

tag3DPens::tag3DPens()
{
    pPenBlack = new CPen(PS_SOLID, 1, RGB(0,0,0));
    pPenDarkGray = new CPen(PS_SOLID, 1, GetSysColor
(COLOR_BTNSHADOW));
    pPenWhite = new CPen(PS_SOLID, 1, RGB(255,255,255));
    pPenGray = new CPen(PS_SOLID, 1, GetSysColor
(COLOR_BTNFACE));
}

tag3DPens::~tag3DPens()
{
    delete pPenBlack;
    delete pPenDarkGray;
    delete pPenWhite;
    delete pPenGray;
}

int GfxDraw3DBorder(CDC *pDC, LP3DPENS p3DPens, CRect
rectBorder, BOOL bInside)
{
    if(bInside)
    {
        CPen *pOldPen = pDC->SelectObject(p3DPens->
pPenDarkGray);

        pDC->MoveTo(rectBorder.left, rectBorder.top);
        pDC->LineTo(rectBorder.right, rectBorder.top);
        pDC->MoveTo(rectBorder.left, rectBorder.top);
        pDC->LineTo(rectBorder.left, rectBorder.bottom);
        pDC->SelectObject(p3DPens->pPenBlack);
        rectBorder.DeflateRect(1, 1);
        pDC->MoveTo(rectBorder.left, rectBorder.top);
        pDC->LineTo(rectBorder.right, rectBorder.top);
        pDC->MoveTo(rectBorder.left, rectBorder.top);
        pDC->LineTo(rectBorder.left, rectBorder.bottom);
        pDC->SelectObject(p3DPens->pPenWhite);
        rectBorder.InflateRect(1, 1);
        pDC->MoveTo(rectBorder.right-1, rectBorder.top);

```

```

pDC->LineTo(rectBorder.right-1, rectBorder.bottom);
pDC->MoveTo(rectBorder.left, rectBorder.bottom-1);
pDC->LineTo(rectBorder.right, rectBorder.bottom-1);
pDC->SelectObject(p3DPens->pPenGray);
rectBorder.DeflateRect(1, 1);
pDC->MoveTo(rectBorder.right-1, rectBorder.top);
pDC->LineTo(rectBorder.right-1, rectBorder.bottom);
pDC->MoveTo(rectBorder.left, rectBorder.bottom-1);
pDC->LineTo(rectBorder.right, rectBorder.bottom-1);
pDC->SelectObject(pOldPen);
}
else
{
    CPen *pOldPen = pDC->SelectObject(p3DPens->pPenGray);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.right, rectBorder.top);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.left, rectBorder.bottom);
    pDC->SelectObject(p3DPens->pPenWhite);
    rectBorder.DeflateRect(1, 1);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.right, rectBorder.top);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.left, rectBorder.bottom);
    pDC->SelectObject(p3DPens->pPenBlack);
    rectBorder.InflateRect(1, 1);
    pDC->MoveTo(rectBorder.right, rectBorder.top);
    pDC->LineTo(rectBorder.right, rectBorder.bottom+1);
    pDC->MoveTo(rectBorder.left, rectBorder.bottom);
    pDC->LineTo(rectBorder.right, rectBorder.bottom);
    pDC->SelectObject(p3DPens->pPenDarkGray);
    rectBorder.DeflateRect(1, 1);
    pDC->MoveTo(rectBorder.right, rectBorder.top);
    pDC->LineTo(rectBorder.right, rectBorder.bottom+1);
    pDC->MoveTo(rectBorder.left, rectBorder.bottom);
    pDC->LineTo(rectBorder.right, rectBorder.bottom);
    pDC->SelectObject(pOldPen);
}
return 2;
}

void GfxDrawBorder(CDC *pDC, CRect rectBorder,
    CPen *pPen)

```

```

{
    CPen *pOldPen = NULL;
    if(pPen)
        pOldPen = pDC->SelectObject(pPen);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.right-1, rectBorder.top);
    pDC->LineTo(rectBorder.right-1, rectBorder.bottom-1);
    pDC->LineTo(rectBorder.left, rectBorder.bottom-1);
    pDC->LineTo(rectBorder.left, rectBorder.top);
    if(pOldPen)
        pDC->SelectObject(pOldPen);
}

void GfxDrawBorder(CDC *pDC, CRect rectBorder, COLORREF
    clFill, CPen *pPen)
{
    CPen *pOldPen = NULL;
    if(pPen)
        pOldPen = pDC->SelectObject(pPen);
    pDC->MoveTo(rectBorder.left, rectBorder.top);
    pDC->LineTo(rectBorder.right-1, rectBorder.top);
    pDC->LineTo(rectBorder.right-1, rectBorder.bottom-1);
    pDC->LineTo(rectBorder.left, rectBorder.bottom-1);
    pDC->LineTo(rectBorder.left, rectBorder.top);
    if(pOldPen)
        pDC->SelectObject(pOldPen);
    CBrush brush(clFill);
    rectBorder.DeflateRect(1, 1);
    pDC->FillRect(rectBorder, &brush);
}

//End of file.

```

```
// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   util_gdi.h
#ifndef _INC_UTIL_GDI_
#define _INC_UTIL_GDI_

typedef struct tag3DPens
{
    CPen *pPenBlack;
    CPen *pPenDarkGray;
    CPen *pPenWhite;
    CPen *pPenGray;
    tag3DPens();
    ~tag3DPens();
} _3DPENS, *LP3DPENS;

int GfxDraw3DBorder(CDC *pDC, LP3DPENS p3DPens,
    CRect rectBorder, BOOL bInside=TRUE);
void GfxDrawBorder(CDC *pDC, CRect rectBorder, CPen
    *pPen=NULL);
void GfxDrawBorder(CDC *pDC, CRect rectBorder,
    COLORREF cFill, CPen *pPen=NULL);
#endif
//End of file.
```

```
// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   util_mdc.cpp
#include "stdafx.h"
#include "util_mdc.h"

GMemDC::GMemDC() : pOldBitmap(NULL),
bPaintStructInitialized(FALSE)
{
}

GMemDC::~GMemDC()
{
    ASSERT(pOldBitmap == NULL);
    ASSERT(bPaintStructInitialized == FALSE);
}

BOOL GMemDC::Create(CDC *pDC, const RECT& rDest, CBru
    *pBkBrush, const PAINTSTRUCT *pPS)
{
    BOOL bOK = FALSE;
    pOldBitmap = NULL;
    ASSERT(pDC != NULL);
    ASSERT(m_hDC == NULL);
    if(pDC != NULL)
    {
        if(CreateCompatibleDC(pDC) != 0)
        {
            CRect rDestRect(rDest);
            #ifndef _MFC_DOCS_HAVE_NO_ERRORS_
            int iBitsPerPixel = pDC->GetDeviceCaps(BITSPIXEL);
            #ifndef _DEBUG
            int iThisBitsPerPixel = GetDeviceCaps(BITSPIXEL);
            // these should be the same
            ASSERT(iThisBitsPerPixel == iBitsPerPixel);
            #endif
            int iWidth = rDestRect.Width() * iBitsPerPixel;
            int iHeight = rDestRect.Height() * iBitsPerPixel;
            CSize sizeBitmap(iWidth, iHeight);
            #endif
            CSize sizeBitmap(rDestRect.Width(), rDestRect.Height());
            // Create an uninitialized bitmap
            if(Bitmap.CreateCompatibleBitmap(pDC, sizeBitmap.cx,
                sizeBitmap.cy) != 0)
```

```

{
    pOldBitmap = SelectObject(&Bitmap);
    if(pOldBitmap != NULL)
    {
        CBrush *pBrush = pBkBrush;
        if(pBrush == NULL)
            pBrush = pDC->GetCurrentBrush();
        ASSERT(pBrush != NULL);
        // Initialize background
        if(pBrush != NULL)
            FillRect(rDestRect, pBrush);
        if(pPS != NULL)
        {
            bPaintStructInitialized = TRUE;
            memcpy((void *)&m_ps, (void *)pPS, sizeof(m_ps));
        }
        bOK = TRUE;
    }
}

ASSERT(bOK == TRUE);
return bOK;
}

BOOL GMemDC::Copy(CDC *pDC, const RECT& rDest, const
POINT *pptSource)
{
    BOOL bOK = FALSE;
    CRect r(rDest);
    ASSERT(this != NULL);
    ASSERT(pDC != NULL);
    // Must be true in order to use BitBlt()
    ASSERT(pDC->GetDeviceCaps(RASTERCAPS) &
RC_BITBLT);
    POINT ptSrcTopLeft;
    if(pptSource != NULL)
        ptSrcTopLeft = (*pptSource);
    else // Default to destination rect upper-left
    {
        ptSrcTopLeft.x = r.left;
        ptSrcTopLeft.y = r.top;
    }
}

```

```

if(pDC != NULL)
    bOK = pDC->BitBlt(r.left, r.top, r.Width(), r.Height(),
        this, ptSrcTopLeft.x, ptSrcTopLeft.y, SRCCOPY);
    ASSERT(bOK == TRUE);
    return bOK;
}

BOOL GMemDC::Release()
{
    BOOL bOK = FALSE;
    ASSERT(pOldBitmap != NULL);
    if(pOldBitmap != NULL)
    {
        SelectObject(pOldBitmap);
        pOldBitmap = NULL;
        bOK = TRUE;
    }
    bPaintStructInitialized = FALSE;
    return bOK;
}

const PAINTSTRUCT *GMemDC::GetPaintStruct() const
{
    const PAINTSTRUCT *pPS;
    if(bPaintStructInitialized == TRUE)
        pPS = (const PAINTSTRUCT *)(&m_ps);
    else
        pPS = NULL;
    return pPS;
}

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   util_mdc.h

#ifndef _INC_UTIL_MDC_
#define _INC_UTIL_MDC_

class GMemDC : public CDC
{
public:
    BOOL Create(CDC *pDC,
               const RECT& rClient,
               CBrush *pBkBrush=NULL, of 'pDC'
               const PAINTSTRUCT *pPS=NULL); // Paint struct if
'pDC' is actually a CPaintDC
    BOOL Copy(CDC *pDC,
              const RECT& rDraw,
              const POINT *ppt=NULL);

    BOOL Release();
    // Return the internal paint structure
    const PAINTSTRUCT *GetPaintStruct() const;

public:
    GMemDC();
    ~GMemDC();

private:
    PAINTSTRUCT m_ps;
    BOOL bPaintStructInitialized;
    CBitmap *pOldBitmap;
    CBitmap Bitmap;
    GMemDC( const GMemDC& );
    GMemDC& operator=( const GMemDC& );
};

#endif
//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//      Machine via Internet Network.
// Date:   Dec 19, 2002
// File:   util_pop.cpp

#include "stdafx.h"
#include "util_pop.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

GPopupWindow::GPopupWindow()
{
}

GPopupWindow::~GPopupWindow()
{
}

void GPopupWindow::PostNcDestroy()
{
    delete this;
}

BOOL GPopupWindow::OnLButtonDownCheck(UINT nFlags,
CPoint pt)
{
    CRect cr;
    GetClientRect(cr);
    if(!cr.PtInRect(pt))
        return TRUE;
    return FALSE;
}

BEGIN_MESSAGE_MAP(GPopupWindow, CWnd)
    ON_WM_CREATE()
    ON_WM_DESTROY()
    ON_WM_CAPTURECHANGED()
    ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()

void GPopupWindow::OnCaptureChanged(CWnd *pWnd)
{
    PostMessage(WM_CLOSE);
}

int GPopupWindow::OnCreate(LPCREATESTRUCT lpcs)
{

```

```

int nCreate = CWnd::OnCreate(lpcs);
SetCapture();
return nCreate;
}

void GPopupWindow::OnLButtonDown(UINT nFlags, CPoint pt)
{
    if(OnLButtonDownCheck(nFlags, pt))
    {
        ::ReleaseCapture();
    }
}

//End of file.

```

```

// Project: RS232 Interface for Digital Automatic Answering
//          Machine via Internet Network
// Date:    Dec 19, 2002
// File:    util_pop.h
#ifndef _INC_UTIL_POP_
#define _INC_UTIL_POP_
class GPopupWindow : public CWnd
{
    DECLARE_MESSAGE_MAP()
public:
    GPopupWindow();
protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpcs);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint pt);
    afx_msg void OnCaptureChanged(CWnd *pWnd);
    virtual BOOL OnLButtonDownCheck(UINT nFlags, CPoint pt);
    virtual void PostNcDestroy();
public:
    virtual ~GPopupWindow();
};
#endif
//End of file.

```

```

;*****
;PROJECT:      DIGITAL AUTOMATIC ANSWERING
;              MACHINE VIA INTERNET.
;DATE:         JULY 19, 2002
;FILE:         DIAM2002.ASM
;TITLE:        MAIN PROGRAM FOR DIGITAL
;              AUTOMATIC ANSWERING MACHINE.
;MANY THANKS TO MY STUDENTS IN
;TELECOMMUNICATION ENGINEERING PROJECTS.
;              PRAKOLKRIT SEANGCHUWONG
;              TAVEESAK SRIPODOK
;              SUTEE KONKRATOKE
;              POLLAKRIT TOONKUM
;              CHONSAWAT SONGSRI
;*****

.INCLUDE"8535DEF.INC"

.CSEG
.ORG    $0000

        RJMP    RESET
;Reset entry vector
        RETI
        RETI
        RETI
        RETI    ;timer2
        RETI
        RETI    ;timer/counter1
                ;compare matched
        RETI
        RETI    ;timer1
        RETI
        RETI    ;timer0
        RETI    ;RXdone
        RETI    ;TXempty
        RETI    ;TXdone
        RETI
        RETI
        RETI    ;INT Number 17
        .DEF    LED_REC    =R28
        .DEF    SEC_TIME   =R16
        .DEF    TEMP       =R20

;GLOBAL REGISTER
        .DEF    DATA_PASSWORD=R29

;*****
;DEF    DATA_ETC=R30
;DEF    ADDR_EEP=R31
RESET:   LDI    TEMP,LOW(RAMEND)
;INITIAL STACK POINTER
        OUT    SPL,TEMP
        LDI    TEMP,HIGH(RAMEND)
        OUT    SPL+1,TEMP
;DEFINE for RS232
.EQU     PC_PLAY = 0x70
        ;PC_PLAY
.EQU     PC_REC  = 0x72
        ;PC_REC
.EQU     PC_STOPREC = 0x73
        ;PC_STOP
;*****INITIAL PORT OF AVR8535
;DEFINE PORTA CONTROLS IC RECORDER
        .EQU     PORTA0 = E_PLAY_T
        ;IC TRANSMIT
        .EQU     PORTA1 = RESET_T
        ;IC TRANSMIT
        .EQU     PORTA2 = L_REC
        ;IC RECEIVE
        .EQU     PORTA3 = E_PLAY_R
        ;IC RECEIVE
        .EQU     PORTA4 = RESET_R
        ;IC RECEIVE
        .EQU     PORTA5 = INT_T
        ;IC TRANSMIT
        .EQU     PORTA6 = INT_R
        ;IC RECEIVE
        .EQU     PORTA7 = SWITCH_PLA'
        ;IC RECEIVE
;DEFINE PORTB
        .EQU     PORTB0 =
        SWITCH_PASSWORD
        .EQU     PORTB1 = LED_STATUS
        .EQU     PORTB2 = CHANNEL
        .EQU     PORTB3 = CHANNEL
        .EQU     PORTB4 = CHANNEL 1
        .EQU     PORTB5 = CHANNEL 2
        .EQU     PORTB6 = CHANNEL 3

```

```

.EQU      PORTB7 = CHANNEL 4
;DEFINE PORTC
.EQU      PORTC0 =
CONTROL_SUPPLY
.EQU      PORTC1 = ON/OFF_HOOK
.EQU      PORTC2 =
CHECK_RINGING&HOLD_TONE
.EQU      PORTC3 = INT_8870
.EQU      PORTC4 = Q1
.EQU      PORTC5 = Q2
; UPPERBIT
.EQU      PORTC6 = Q3
; "IC MT8870 RECEIVE DTMF"
.EQU      PORTC7 = Q4
;DEFINE PORTD
.EQU      PORTD0 = RXD
;RS232 - SERIAL COMMUNICATION
.EQU      PORTD1 = TXD
.EQU      PORTD2 = SWITCH_1
.EQU      PORTD3 = A
.EQU      PORTD4 = B
.EQU      PORTD5 = C
;"LED 7 SEGMENTS"
.EQU      PORTD6 = D
INIT_PORT:  LDI      TEMP,0B00011111
            OUT      DDRA,TEMP
            LDI      TEMP,0B11100000
            OUT      PORTA,TEMP

            LDI      TEMP,0B11111110
            OUT      DDRB,TEMP
            LDI      TEMP,0B00000001
            OUT      PORTB,TEMP
            LDI      TEMP,0B00000011
            OUT      DDRC,TEMP
            LDI      TEMP,0B11111100
            OUT      PORTC,TEMP
STORE_PASSW: LDI
            DATA_PASSWORD,0B00010000;PWD
CONSTANT=1
MOV        R11,DATA_PASSWORD

LDI
            DATA_PASSWORD,0B00100000 ;PWD
CONSTANT=2
MOV        R12,DATA_PASSWORD
LDI
            DATA_PASSWORD,0B00110000 ;PWD
CONSTANT=3
MOV        R13,DATA_PASSWORD
LDI
            DATA_PASSWORD,0B01000000 ;PWD
CONSTANT=4
MOV        R14,DATA_PASSWORD
LDI        TEMP,0X02
SHOW_RESET: SBI        PORTB,1
;DISPLAY LED
RCALL      DELAYAA
CBI        PORTB,1
RCALL      DELAYAA
DEC        TEMP
BRNE       SHOW_RESET
LDI        SEC_TIME,0X04
;DELAY 4 SEC
RCALL      TIME_SET
;BEFORE SUPPLY IC RECORDED
SBI        PORTC,0
;ON SUPPLY IC RECORDED
LDI        SEC_TIME,0X01
;DELAY 1 SEC
RCALL      TIME_SET
;BEFORE SUPPLY IC RECORDED
INIT_PORTD: LDI        TEMP,0X04
LOOP_I_ST:  SBI        PORTB,1
;SHOW INITIAL START
RCALL      DELAYAA
CBI        PORTB,1
RCALL      DELAYAA
DEC        TEMP
BRNE       LOOP_I_ST
LDI        R16,0X7F
OUT        DDRD, R16
INIT_PROG:

```

```

        SBI        PORTA,1
;RESET IC SENT MESSAGE
        RCALL DELAYAA
        CBI        PORTA,1
;***** INITIAL FOR RS232 INTERFACE
*****
        CLI
;Clear golbal interrupt
        CBI        UCR, RXCIE
;Clear TX interrupt
        CBI        UCR, TXCIE
;Clerr RX interrupt
        CBI        UCR, UDRIE
;Clear data empty interrupt
        CBI        UCR, CHR9
;Send 8 data
        LDI        TEMP, 51
        OUT        UBRR, TEMP
;Baud 9600 at 8MHz XTAL
        SBI        UCR, TXEN
;Set pin TX as serial TX
        SBI        UCR, RXEN
;Set pin RX as serial RX
;*****
;*****" DETECT RINGING TONE "*****
;*****
START_PROG: .DEF        TEMP1    =R21
            .DEF        CHECKCON=R22
            .DEF        CHECKVAR=R23
;****SET REGISTER AS VARIABLE***
            .DEF        C_H      =R24
            .DEF        C_H1     =R25
            .DEF        C_H2     =R26
            .DEF        L_SW     =R27
            SER        C_H
            SER        C_H1
            CLR        C_H2
            CLR        TEMP1
;/**INITIAL NUMBER 'HOLD ON'**
;*****
START:      SER        TEMP
;INITIAL BOUNCE RINGINGTONE

```

```

RECEIVE_RING: MOV    CHECKVAR,TEMP1
;VARIABLE VALUE
            RCALL    CHANGE_PASSWORD
;**** SUB ****
            RCALL    CHECK_HOLD
;**** SUB ****
            RCALL    LISTEN_RECORD
;**** SUB ****
            RCALL    SHOW_RECEIVE
;**** SUB ****
            SBIC     PINC,2
;IF "HAVE RINGING" CROSS
            RJMP     RECEIVE_RING
            SBI      PORTB,1
;DISPLAY RECEIVE RINGING TONE
            SER      C_H
            SER      C_H1
            CLR      C_H2
            DEC      TEMP
            BRNE     RECEIVE_RING
            INC      TEMP1
            MOV      CHECKCON,TEMP1
            CPI      TEMP1,0X03
;NUMBER RING = 1 TIMES
            BRNE     START
;*****
;*****" OPERATION " *****
;*****
HOLD_ON:   .DEF      INTP_RECV  =R8
            .DEF      NUM_REPEAT =R20
            .DEF      RECV_DTMF  =R21
            .DEF      NUM_PASSWORD=R22
            .DEF      CONTROL_MESS=R23
            .DEF      LOOP_MESS  =R24
            .DEF      CORRECT     =R25
            .DEF      NUM_LOOP    =R26
            .DEF      VAR_CONMES  =R27
            CBI      PORTB,1
;OFF DISPLAY RINGING TONE
            SBI      PORTC,1
;*****ON HOOK*****

```

```

        SBI      PORTA,0
;SENT MESSAGE 1
        RCALL   DELAYAA
        RCALL   DELAYAA
        CBI      PORTA,0
        RCALL   DELAYAA
        NOP
        LDI      SEC_TIME,0X01
;DELAY ABOUT 1 SEC FOR
        RCALL   TIME_SET
;RANGE BEFORE MESSAGE'
        RCALL   DELAYAA
;RCALL DELAYAA
;****CAN FILL IMPROVE DELAY
        LDI      VAR_CONMES,0X0F
;VARIABLE CONTROL
        LDI      SEC_TIME,0X05
;ADD DELAY FOR MESSAGE
        RCALL   TIME_SETKEY
        CPI      VAR_CONMES,0X00
;IF EQUAL JUMP
        BREQ     RECEIVE_KEY
        LDI      SEC_TIME,0X01
;DELAY ABOUT 1 SEC FOR
        RCALL   TIME_SET
; RANGE AFTER MESSAGE'
        RCALL   DELAYAA
;DELAYAA= 0.5 Sec
;RCALL DELAYAA
;****CAN FILL IMPROVE DELAY
        RCALL   TIME_OUT
;*SET TIME_OUT-WAIT ANY KEY
        RCALL   DELAY_KEY
RECEIVE_KEY: IN      RECV_DTMF,PINC
        ANDI     RECV_DTMF,0B11110000
;*****
; ***** START PROGRAM NO.1 2 *****
;*****
        .INCLUDE"SUB_1_21.SUB"
;-----
        .INCLUDE"SUB_2_21.SUB"
;*****

```

```

; ***** CLOSE *****
;*****
CLOSE: CBI      PORTA,0
        CBI      PORTA,1
        CBI      PORTA,2
        CBI      PORTA,3
;SAFTY IC RECORDED
        CBI      PORTA,4
        LDI      SEC_TIME,0X03
;DELAY BEFORE OFF HOOK 3 SEC
        RCALL   TIME_SET
        CBI      PORTC,1
;OFF HOOK
        RJMP    INIT_PORTD
;*****
; "SUB NO.1 PROGRAM CHANGE PASSWORD"
;*****
CHANGE_PASSWORD: CPI      TEMP1,0X00
        BREQ     NEXT_CHANGE
        RJMP     CH_P_BACK
NEXT_CHANGE: SBIC      PINB,0
        RJMP     CH_P_BACK
        SBIC      PINB,0
        RJMP     CH_P_BACK
.DEF      INTP_RECV  =R8
.DEF      NUM_REPEAT =R20
.DEF      RECV_DTMF  =R21
.DEF      NUM_PASSWORD =R22
.DEF      LOOP_YES_NO =R23
.DEF      SHOW_LED   =R24
.DEF      CORRECT    =R25
.DEF      LOOP_TEMP  =R26
        SBI      PORTB,1
;SHOW LED 'RED' UNTIL FINISH
        SBI      PORTC,1
;ON HOOK
        LDI      SHOW_LED,0B00000000
;SHOW '0' (bit 6,5,4,3)
        OUT      PORTD,SHOW_LED
        RCALL   DELAYAA
        RCALL   DELAYAA

```

```

        LDI    SHOW_LED,0B01111000
;SHOW '_' (bit 6,5,4,3)
        OUT    PORTD,SHOW_LED
INIT_D_OUT: LDI    NUM_REPEAT,0X03
OLD_PASSWORD: LDI    NUM_PASSWORD,0X04
;AMOUNT PASSWORD=4
        LDI    CORRECT,0X00
;NO OF TRUE PASSWORD
OLD_1:    CPI    NUM_PASSWORD,0X04
        BRNE   OLD_2
        MOV    DATA_PASSWORD,R11
        RJMP   WAIT_K_O_PASS
OLD_2:    CPI    NUM_PASSWORD,0X03
        BRNE   OLD_3
        MOV    DATA_PASSWORD,R12
        RJMP   WAIT_K_O_PASS
OLD_3:    CPI    NUM_PASSWORD,0X02
        BRNE   OLD_4
        MOV    DATA_PASSWORD,R13
        RJMP   WAIT_K_O_PASS
OLD_4:    MOV    DATA_PASSWORD,R14
WAIT_K_O_PASS: RCALL TIME_K_PULL
        RCALL  DELAY_KEY
        RCALL  TIME_K_PULL
        RCALL  DELAY_KEY
        RCALL  TIME_OUT
;*SET TIME_OUT-WAIT ANY KEY
        RCALL  DELAY_KEY
        RCALL  TIME_OUT
        RCALL  DELAY_KEY
        RCALL  TIME_K_PULL
        RCALL  DELAY_KEY
        RCALL  TIME_K_PULL
        RCALL  DELAY_KEY
        IN     RECV_DTMF,PINC
        ANDI   RECV_DTMF,0B11110000
        CP
        DATA_PASSWORD,RECV_DTMF
        BREQ   WAIT_PASS_OK
WAIT_PU_KEYNO: RCALL SHOW_KEY
        LDI    CORRECT,0X00
;RESET NUM OF CORRECT

```

```

        DEC    NUM_PASSWORD
        BRNE   OLD_1
        LDI    LOOP_TEMP,0X02
SHOW_WRONG: LDI    SHOW_LED,0B00000000
;SHOW '0' (bit 6,5,4,3)
        OUT    PORTD,SHOW_LED
;BECAUSE WRONG PASSWORD
        RCALL  DELAYAA
        RCALL  DELAYAA
        LDI    SHOW_LED,0B01111000
        OUT    PORTD,SHOW_LED
        RCALL  DELAYAA
        RCALL  DELAYAA
        DEC    LOOP_TEMP
        BRNE   SHOW_WRONG
        DEC    NUM_REPEAT
        BREQ    BACK_NEW_P1
        RJMP   OLD_PASSWORD
WAIT_PASS_OK: RCALL SHOW_KEY
        RCALL  DELAYAA
        INC    CORRECT
        CPI    CORRECT,0X04
        BREQ   SHOW_TRUE
        DEC    NUM_PASSWORD
        BRNE   OLD_1
        RJMP   SHOW_WRONG
        RJMP   SHOW_TRUE
BACK_NEW_P1:  RJMP   BACK_NEW_P
SHOW_TRUE:   LDI    LOOP_TEMP,0X02
SHOW_TR_O:   LDI    SHOW_LED,0B00101000
;SHOW '5' (bit 6,5,4,3)
        OUT    PORTD,SHOW_LED
;BECAUSE TRUE PASSWORD
        RCALL  DELAYAA
        RCALL  DELAYAA
        LDI    SHOW_LED,0B01111000
        OUT    PORTD,SHOW_LED
        RCALL  DELAYAA
        RCALL  DELAYAA
        DEC    LOOP_TEMP
        BRNE   SHOW_TR_O
NEW_PASSWORD: LDI    NUM_REPEAT,0X03

```

```

;INPUT NEW PWD 3 TIMES
KEY_NEW:
SEQ_NEW1:    RCALL WAIT_KEYNEW
              MOV  R1,RECV_DTMF
              RCALL SHOW_KEYNEW
SEQ_NEW2:    RCALL WAIT_KEYNEW
              MOV  R2,RECV_DTMF
              RCALL SHOW_KEYNEW
SEQ_NEW3:    RCALL WAIT_KEYNEW
              MOV  R3,RECV_DTMF
              RCALL SHOW_KEYNEW
SEQ_NEW4:    RCALL WAIT_KEYNEW
              MOV  R4,RECV_DTMF
              RCALL SHOW_KEYNEW
              LDI  SEC_TIME,0X01
;DELAY 1 SECONDS'
              RCALL TIME_SET
              RCALL SHOW_NEW
              LDI  LOOP_YES_NO,0X20
;/CONTROL LOOP_SHOW 5&0
;/ AND WAIT KEY FOR
;/CONFIRM NEW PASSWORD
;CONFIRM_NEWPASS:
              SBIC  PINC,3
;SKIP IF PRESS KEY
              RJMP  YES_NO_NEWPASS
              LDI  SHOW_LED,0B00101000
;SHOW 'S' (bit 6,5,4,3)
              OUT  PORTD,SHOW_LED
;BECAUSE TRUE PASSWORD
              RCALL DELAYAA
              LDI  SHOW_LED,0B00000000
;SHOW '0' (bit 6,5,4,3)
              OUT  PORTD,SHOW_LED
;BECAUSE TRUE PASSWORD
              RCALL DELAYAA
              DEC  LOOP_YES_NO
              BREQ  BACK_NEW_P
              RJMP  CONFIRM_NEWPASS
YES_NO_NEWPASS: LDI  SHOW_LED,0B01111000
;SHOW '_'
              OUT  PORTD,SHOW_LED
              RCALL DELAY_KEY
IN          RECV_DTMF,PINC
ANDI       RECV_DTMF,0B11110000
CPI        RECV_DTMF,0B01010000
;IF EQUAL 'S' GO O.K.
              BREQ  OK_NEWPASSWORD
              CPI  RECV_DTMF,0B10100000
              BRNE  BACK_NEW_P
              DEC  NUM_REPEAT
              BRNE  KEY_NEW
              RJMP  BACK_NEW_P
OK_NEWPASSWORD: MOV  R11,R1
              MOV  R12,R2
              MOV  R13,R3
              MOV  R14,R4
              RCALL SHOW_NEW
BACK_NEW_P:  LDI  SEC_TIME,0X02
;DELAY 2 SECONDS'
              RCALL TIME_SET
              CBI  PORTB,1
;OUT OF CHANGE PASSWORD
              CBI  PORTC,1
;ON HOOK
              RJMP  INIT_PORTD
CH_P_BACK:  RET
WAIT_KEYNEW: RCALL TIME_OUT
;*SET TIME_OUT-WAIT ANY KEY
              RCALL DELAY_KEY
              RCALL TIME_OUT
              RCALL DELAY_KEY
              RCALL TIME_K_PULL
              RCALL DELAY_KEY
              RCALL TIME_K_PULL
              RCALL DELAY_KEY
IN          RECV_DTMF,PINC
ANDI       RECV_DTMF,0B11110000
RET
SHOW_KEY:   LSR  RECV_DTMF
              CPI  RECV_DTMF,0B01010000
;NUMBER" 0 "
              BRNE OUT_KEY

```

```

        CLR     RECV_DTMF
;SHOW NUMBER "0"
OUT_KEY:  OUT    PORTD,RECV_DTMF
        RCALL  DELAYAA
        RCALL  DELAYAA
        LD1    SHOW_LED,0B01111000
;SHOW '_'
        OUT    PORTD,SHOW_LED
        CPI    NUM_PASSWORD,0X01
        BRNE   BACK_SHOW
        RCALL  DELAYAA
;DELAY FOR SHOW LAST NUMBER
        RCALL  DELAYAA
BACK_SHOW:  RET
SHOW_KEYNEW: LSR    RECV_DTMF
        CPI    RECV_DTMF,0B01010000
;NUMBER "0 "
        BRNE   OUT_KEY_N
        CLR    RECV_DTMF
OUT_KEY_N:  OUT    PORTD,RECV_DTMF
        RCALL  DELAYAA
        RCALL  DELAYAA
        LD1    SHOW_LED,0B01111000
;SHOW '_'
        OUT    PORTD,SHOW_LED
        RET
SHOW_NEW:  MOV    RECV_DTMF,R1
        RCALL  SHOW_KEY
        MOV    RECV_DTMF,R2
        RCALL  SHOW_KEY
        MOV    RECV_DTMF,R3
        RCALL  SHOW_KEY
        MOV    RECV_DTMF,R4
        RCALL  SHOW_KEY
        RET
*****
'SUB NO.2 PROGRAM CHECK "HOLD ON BEFORE WORK"
*****
HECK_HOLD:  CPI    TEMP1,0X00
        BREQ   COMEBACK
        CP     CHECKCON,CHECKVAR
;* COMPARE 'HOLD OFF *
        BRNE   COMEBACK
        DEC    C_H
        BRNE   COMEBACK
        SER    C_H
        DEC    C_H1
        BRNE   COMEBACK
        CBI    PORTB,1
;RECEIVE RINGING TONE
        SER    C_H
        SER    C_H1
        INC    C_H2
        CPI    C_H2,0X22
        BRNE   COMEBACK
        RJMP   INIT_PORTD
COMEBACK:  RET
;*****
;*****SUB NO.4 PROGRAM LISTEN_RECORD*****
;*****
LISTEN_RECORD: CPI    TEMP1,0X00          ;/IF THERE
ARE RINGING TONES GO
        BRNE   RET_L_RE
;RETURN TO MAIN PROGRAM
        SBIC   PINA,7
;READ KEY PLAY MESSAGE
        RJMP   OFF_S_REC
        RJMP   ON_S_REC
OFF_S_REC:  CBI    PORTA,3
;NO TELL MESSAGE THAT RECORDED
        RJMP   RET_L_RE
ON_S_REC:  SBI    PORTA,3
;TELL MESSAGE THAT RECORDED
        BRCS   RET_L_RE
;IF CARRY SET DON'T WRITE EEPROM
        LD1    LED_REC,0B01111000
;GET VALUE '_'
        RCALL  EEP_WRITEL
        SEC
;SET C/FLAG FOR REDUCE WRITING
RET_L_RE:  RET
;*****

```

```

;*****
; "SUB NO.5 PROGRAM RESET IC SENDING"
;*****
RESET_IC:    SBI    PORTA,1
             ;RESET IC SENT MESSAGE
             RCALL  DELAYAA
             CBI    PORTA,1
             RCALL  DELAYAA
             RET

;*****
; "SUB NO.6 LOOP_MESSAGE"
;*****
S_LOOP_MESS: SBI    PORTA,0
             RCALL  DELAYAA
             RCALL  DELAYAA
             CBI    PORTA,0
             RCALL  DELAYAA
L_MESS: SBI    PORTA,0
             RCALL  DELAYAA
             CBI    PORTA,0
             RCALL  DELAYAA
             DEC    LOOP_MESS
             BRNE   L_MESS
             RET

;*****
CLOSE3:      RJMP   CLOSE
;*****
; "SUB NO.7 READ&CHECK PASSWORD"
;*****
RD_PASSWORD_P: RCALL  DELAYAA
               LDI    NUM_PASSWORD,0X04
               ;AMOUT PASSWORD=4
SEQ_1:       CPI    NUM_PASSWORD,0X04
               BRNE   SEQ_2
               MOV    DATA_PASSWORD,R11
               RJMP   WAIT_KEY_PASS
SEQ_2:       CPI    NUM_PASSWORD,0X03
               BRNE   SEQ_3
               MOV    DATA_PASSWORD,R12
               RJMP   WAIT_KEY_PASS
SEQ_3:       CPI    NUM_PASSWORD,0X02
               BRNE   SEQ_4
               MOV    DATA_PASSWORD,R13
               RJMP   WAIT_KEY_PASS
SEQ_4:       MOV    DATA_PASSWORD,R14
WAIT_KEY_PASS: RCALL  TIME_K_PULL
               RCALL  DELAY_KEY
               RCALL  TIME_K_PULL
               RCALL  DELAY_KEY
               RCALL  TIME_EEP
               ;/WAIT KEY
               RCALL  DELAY_KEY
               RCALL  TIME_EEP
               RCALL  DELAY_KEY
               RCALL  TIME_K_PULL
               RCALL  DELAY_KEY
               RCALL  TIME_K_PULL
               RCALL  DELAY_KEY
               IN     RECV_DTMF,PINC
               ANDI    RECV_DTMF,0B11110000
               CP
               DATA_PASSWORD,RECV_DTMF
               BREQ   WAIT_PU_KECOR
WAIT_PU_KERROR: LDI    CORRECT,0X00
               ;RESET NUM OF CORRECT
               RCALL  TIME_K_PULL
               RCALL  DELAYAA
               RCALL  TIME_K_PULL
               DEC    NUM_PASSWORD
               BRNE   SEQ_1
MESS4:       DEC    NUM_REPEAT
               CPI    NUM_REPEAT,0X02
               BRNE   RE_MESS4
               MOV    LOOP_MESS,CONTROL_MESS
               ;SENT MESSAGE 4
               RCALL  S_LOOP_MESS
               ;WRONG PASSWORD
               ;****CAN FILL IMPROVE DELAY 2 SEC
               RJMP   RD_PASSWORD_P
RE_MESS4:    CPI    NUM_REPEAT,0X00
               BREQ   CLOSE3
               SBI    PORTA,0
               RCALL  DELAYAA

```

```

        RCALL DELAYAA
;REVIEW MESSAGE 4 (16)
        CBI     PORTA,0
;/"WRONG PWD TRY AGAIN"
        RCALL DELAYAA
;****CAN FILL IMPROVE DELAY 2 SEC
        RJMP    RD_PASSWORD_P
WAIT_PU_KECOR: RCALL TIME_K_PULL
        RCALL DELAYAA
        RCALL TIME_K_PULL
        INC     CORRECT
        CPI     CORRECT,0X04
        BREQ    BACK
        DEC     NUM_PASSWORD
        BREQ    MESS4
        RJMP    SEQ_1
BACK:      RET
;*****
;      "SUB PROGRAM DELAY"
;*****
DELAYAA:   LDI     R17, 0X08
DLY1A:     LDI     R18, 0X0FF
DLY2A:     LDI     R19, 0X0FF
DLY3A:     DEC     R19
            BRNE   DLY3A
            DEC     R18
            BRNE   DLY2A
            DEC     R17
            BRNE   DLY1A
            RET
;*****
; "SUB PROGRAM DELAY FOR BEFORE AND AFTER
;MESSAGE"
;*****
DELAY_MESS: LDI     R17, 0X06
DLY1M:     LDI     R18, 0X0FF
DLY2M:     LDI     R19, 0X0FF
DLY3M:     DEC     R19
            BRNE   DLY3M
            DEC     R18
            BRNE   DLY2M
            DEC     R17

```

```

        BRNE    DLY1M
        RET
;*****
;      "SUB PROGRAM DELAY KEY_DTMF"
;*****
DELAY_KEY:  LDI     R17, 0X01
DLY1K:      LDI     R18, 0X0FF
DLY2K:      LDI     R19, 0X0FF
DLY3K:      DEC     R19
            BRNE   DLY3K
            DEC     R18
            BRNE   DLY2K
            DEC     R17
            BRNE   DLY1K
            RET
;*****
;      "SUB PROGRAM TIME_SET --IN SECOND --"
;      'INPUT R16 IS NUMBER OF SECOND'
;*****
TIME_SET :
TIME_SET1:  LDI     R17, 0X2B
TIME_SET2:  LDI     R18, 0X0FA
TIME_SET3:  LDI     R19, 0X0FF
TIME_SET4:  DEC     R19
            BRNE   TIME_SET4
            DEC     R18
            BRNE   TIME_SET3
            DEC     R17
            BRNE   TIME_SET2
            DEC     SEC_TIME
            BRNE   TIME_SET1
            RET
;*****
;      "SUB PROGRAM TIMEOUT"
;*****
TIME_OUT:   LDI     SEC_TIME, 0X0A
            ;SET TIME =10 SECONDS
TIME1:      LDI     R17, 0X02B
TIME2:      LDI     R18, 0X0FA
TIME3:      LDI     R19, 0X0FF
TIME4:      IN      INTP_RECV, P1NC

```

```

        SBRC    INTP_RECV,3
; **INTERUPT MT8870 **
        RJMP    KEY_ON
        DEC     R19
        BRNE    TIME4
        DEC     R18
        BRNE    TIME3
        DEC     R17
        BRNE    TIME2
        DEC     SEC_TIME
        BRNE    TIME1
        NOP
        RJMP    CLOSE
KEY_ON:   RET
;*****
;      "SUB PROGRAM TIME EEPROM"
;*****
TIME_EEP:  LDI     SEC_TIME,0X0A
           ;SET TIME =10 SECONDS
TIME_E1:  LDI     R17,0X02B
TIME_E2:  LDI     R18,0X0FA
TIME_E3:  LDI     R19,0X0FF
TIME_E4:  IN      INTP_RECV,PINC
        SBRC    INTP_RECV,3
        RJMP    R_KEY_EEP
        DEC     R19
        BRNE    TIME_E4
        DEC     R18
        BRNE    TIME_E3
        DEC     R17
        BRNE    TIME_E2
        DEC     SEC_TIME
        BRNE    TIME_E1
        NOP
        RCALL   EEP_WRITES
;WRITE DATA IN EEPROM
        LDI     SEC_TIME,0B01000000
        OUT     PORTD,SEC_TIME
        RJMP    CLOSE
R_KEY_EEP:  RET
;*****

```

```

;*****
      "SUB PROGRAM TIME KEY PULL"
;*****
TIME_K_PULL:  LDI  SEC_TIME,0X0A
              ;SET TIME =10 SECONDS
TIMEK1:  LDI  R17,0X02B
TIMEK2:  LDI  R18,0X0FA
TIMEK3:  LDI  R19,0X0FF
TIMEK4:
              IN      INTP_RECV,PINC
              SBRS   INTP_RECV,3
; **INTERUPT MT8870 **
        RJMP    KEY_P_OFF
        DEC     R19
        BRNE    TIMEK4
        DEC     R18
        BRNE    TIMEK3
        DEC     R17
        BRNE    TIMEK2
        DEC     SEC_TIME
        BRNE    TIMEK1
        NOP
        RJMP    CLOSE
KEY_P_OFF:  RET
;*****
;      "SUB PROGRAM TIME LISTEN MESSAGE"
;*****
TIME_MESS:  LDI     SEC_TIME,0X019
           ;SET TIME = 25 SECONDS (25)
TIMEM1:  LDI     R17,0X02B
TIMEM2:  LDI     R18,0X79
TIMEM3:  LDI     R19,0X0FF
TIMEM4:  IN      INTP_RECV,PINC
        SBRC    INTP_RECV,3
; **INTERUPT MT8870 **
        RJMP    KEY_ON5
        DEC     R19
        BRNE    TIMEM4
        DEC     R18
        BRNE    TIMEM3
        DEC     R17
        BRNE    TIMEM2

```

```

        DEC     SEC_TIME
        BRNE    TIMEM1
        LDI     LED_REC,0B01001000
;GET VALUE '9'
        RCALL   EEP_WRITE
        CLC
;INITAIL FOR RE WRITE EEPROM
        RJMP    CLOSE
KEY_ON5:  RET
;*****
;      "SUB PROGRAM TIME SET FOR MESSAGE IF KEY
;INTERRUPT "1ST BY DELAY FOR MESSAGE"
;*****
TIME_SETKEY:
T_SETKEY1:  LDI   R17,0X02B
T_SETKEY2:  LDI   R18,0X0FA
T_SETKEY3:  LDI   R19,0X0FF
T_SETKEY4:  IN     INTP_RECV,PINC
        SBRC    INTP_RECV,3
;**INTERRUPT MT8870**
        RJMP    DEC_VAR
        DEC     R19
        BRNE    T_SETKEY4
        DEC     R18
        BRNE    T_SETKEY3
        DEC     R17
        BRNE    T_SETKEY2
        DEC     SEC_TIME
        BRNE    T_SETKEY1
        NOP
        RJMP    BACKKEY
DEC_VAR:   RCALL   DELAYAA
        IN      RECV_DTMF,PINC
        ANDI    RECV_DTMF,0B11110000
        LDI     VAR_CONMES,0X00
BACKKEY:   RET
;*****
; SUB PROGRAM WRITE & READ DATA IN EEPROM
;*****
EEP_WRITES:  SBIC    EECR,EWE
        RJMP    EEP_WRITES
        LDI     ADDR_EEP,$00
        OUT     EEARH,ADDR_EEP
        LDI     ADDR_EEP,$15
        OUT     EEARL,ADDR_EEP
        OUT     EEDR,DATA_ETC
        SBI     EECR,EEMWE
        SBI     EECR,EWE
        RET
EEP_READS:  SBIC    EECR,EWE
        RJMP    EEP_READS
        LDI     ADDR_EEP,$00
        OUT     EEARH,ADDR_EEP
        LDI     ADDR_EEP,$15
        OUT     EEARL,ADDR_EEP
        SBI     EECR,EERE
        IN      DATA_ETC,EEDR
        RET
;*****
;      SUB PROGRAM FOR RS232 INTERFACE
;*****
TX232:      SBIS     USR,UDRE
;Wait until UDRE = 1
        RJMP     TX232
        OUT      UDR,R16
;Send data to serial
        RET
RX232:      SBIS     USR,RXC
;Wait until have data in buffer
        RJMP     RX232
        IN       R20,UDR
;Read data from buffer
        MOV      TEMP,R20
        RET
;*****
;      .INCLUDE"SUB_3_21.SUB"
;*****
;      *** END OF MAIN PROGRAM DIAM2002.ASM ***
;*****

```

```

;*****
;*PROJECT:      DIGITAL AUTOMATIC ANSWERING
;               MACHINE VIA INTERNET.
;*DATE:         JULY 19, 2002
;*FILE:         SUB_1_21.SUB
;*TITLE:        SUB PROGRAM FOR DIGITAL
;               AUTOMATIC ANSWERING MACHINE.
;*****
; ***** SUB PROGRAM FOR RECORD SOUND*****
;*****
READ_DTMF_NO1: CPI   RECV_DTMF,0B00010000
               ;*COMPARE 'RECEIVE IF IS NO_1'*
               BRNE   READ_DTMF_NO2
               CPI    VAR_CONMES,0X00
;IF EQUAL '0' JUMP
               BREQ   DECCONME_ONE
;SENT MESS IF PRESS KEY INTERRUPT
               LDI    LOOP_MESS,0X01
;SENT MESSAGE 2 (01)
               RCALL  S_LOOP_MESS
;**** SUB PRO ****
               RJMP   RECORD_SOUND
DECCONME_ONE:
               SBI    PORTA,0
               RCALL  DELAYAA
;/ SENT MESSAGE 2
               RCALL  DELAYAA
;/IF PRESS KEY INTERRUPT
               CBI    PORTA,0
               RCALL  DELAYAA
RECORD_SOUND: LDI    SEC_TIME,0X08
               ;'DELAY BEFORE RECORD'(6)
               RCALL  TIME_SET
;***** RS232 INTERFACE *****
               LDI    TEMP, 0X72
;PC_REC
               MOV    R16,TEMP
;Put data to RS232
               RCALL  TX232
               SBI    PORTA,2
; 'RECORD MESSAGE'

```

```

RCALL  TIME_MESS
;STOP RECORD IF PRESS ANY KEY
LDI    SEC_TIME,0X05
;'DELAY 5 SEC AFTER PRESS KEY '
RCALL  TIME_SET
CBI    PORTA,2
RCALL  DELAYAA
NOP
LDI    LED_REC,0B01001000
; GET VALUE '9'
RCALL  EEP_WRITEL
CLC
;INITIAL FOR RE WRITING
NOP
RAMP   CLOSE

```

```

;*****
;      *** END OF SUB PROGRAM SUB_1_21.SUB ***
;*****

```

```

;*****
;PROJECT:      DIGITAL AUTOMATIC ANSWERING
;              MACHINE VIA INTERNET.
;DATE:        JULY 19, 2002
;FILE:        SUB_2_21.SUB
;TITLE:       SUB PROGRAM FOR DIGITAL
;              AUTOMATIC ANSWERING MACHINE.
;*****
; START PROGRAM SENT MESSAGE THAT RECORDED****
;*****
READ_DTMF_NO2: CPI   RECV_DTMF,0B00100000
                BRNE  CLOSE1
                ;LAST LINE FOR END
;*****CHECK PASSWORD 4 DIGITS*****
SENT_MESSP:    LDI    NUM_REPEAT,0X03
;*****NUMBER REPEAT *****
                LDI    CORRECT,0X00
;INITIAL NO. PASSWORD CORRECT
                CPI    VAR_CONMES,0X00
;IF EQUAL '0' JUMP
                BREQ   DECCONME_TWO
;SENT MESS IF PRESS KEY INTERRUPT
                LDI    LOOP_MESS,0X02
; SENT MESSAGE 3 (02)
                RCALL  S_LOOP_MESS
;"PLEASE PUT YOUR PASSWORD "
                RJMP   AFTERMES3
DECCONME_TWO: LDI    LOOP_MESS,0X01
;SENT MESSAGE 3
                RCALL  S_LOOP_MESS
;IF PRESS KEY INTERRUPT
AFTERMES3:    LDI    SEC_TIME,0X01
;"DELAY MESSAGE 15 = 1 SEC '
                RCALL  TIME_SET
                LDI    CONTROL_MESS,0X01
;CONTROL MESSAGE 4
                RCALL  RD_PASSWORD_P
LISTEN_MESS:  CPI    NUM_REPEAT,0X03
                BRNE  JUMP1
                LDI    LOOP_MESS,0X02
;SENT MESSAGE 5

```

```

                RCALL  S_LOOP_MESS
;"YOUR MESSAGE IS"
                LDI    SEC_TIME,0X07
;"DELAY MESSAGE 5 = 7 SEC '(6)
                RCALL  TIME_SET
                SBI    PORTA,3
                RCALL  DELAYAA
                RCALL  DELAYAA
;/**SENT MESSAGE THAT RECORDED**
                CBI    PORTA,3
                RCALL  DELAYAA
                RJMP   INF_LOOP
JUMP1:        LDI    LOOP_MESS,0X01
;SENT MESSAGE 5
                RCALL  S_LOOP_MESS
                LDI    SEC_TIME,0X07
;"DELAY MESSAGE 5 = 7 SEC '(6)
                RCALL  TIME_SET
                SBI    PORTA,3
                RCALL  DELAYAA
                RCALL  DELAYAA
;/**SENT MESSAGE THAT RECORDED**
                CBI    PORTA,3
                RCALL  DELAYAA
INF_LOOP:     RCALL  TIME_MESS_S
                IN     RECV_DTMF,PINC
                ANDI   RECV_DTMF,0B11110000
                CPI    RECV_DTMF,0B01010000
;****COMPARE 'KEY=5'****
                BRNE  INF_LOOP
                SBI    PORTA,3
                RCALL  DELAYAA
                RCALL  DELAYAA
;/**SENT MESSAGE THAT RECORDED**
                CBI    PORTA,3
                RCALL  DELAYAA
                RJMP   INF_LOOP
;*****
;              "SUB PROGRAM TIME LISTEN MESSAGE"
;*****
TIME_MESS_S:  LDI    SEC_TIME,0X01A
                ;SET TIME = 25 SECONDS (25)

```

```

TIMEM1_S:    LDI    R17,0X02B
TIMEM2_S:    LDI    R18,0X0FA
TIMEM3_S:    LDI    R19,0X0FF
TIMEM4_S:    IN     INTP_RECV,PINC
              SBRC   INTP_RECV,3

```

```

; **INTERUPT MT8870 **

```

```

      RJMP    KEY_ON5_OK
      DEC     R19
      BRNE    TIMEM4_S
      DEC     R18
      BRNE    TIMEM3_S
      DEC     R17
      BRNE    TIMEM2_S
      DEC     SEC_TIME
      BRNE    TIMEM1_S
      RJMP    CLOSE

```

```

KEY_ON5_OK:  RET

```

```

CLOSE1:     RJMP    CLOSE

```

```

;*****
;      *** END OF SUB PROGRAM SUB_2_21.SUB ***
;*****

```

```

;*****
; *PROJECT:    DIGITAL AUTOMATIC ANSWERING
;              MACHINE VIA INTERNET .
; *DATE:       JULY 19, 2002
; *FILE:       SUB_3_21.SUB
; *TITLE:      SUB PROGRAM FOR DIGITAL
;              AUTOMATIC ANSWERING MACHINE.
;*****
; "SUB PROGRAM SHOW LED AFTER RECORDED MESSAGE "
;*****
SHOW_RECEIVE: CPI    TEMP1,0X00
              BRNE    RET_SHOW_15
              RCALL   EEP_READL
              ORI     LED_REC,0B10000111
              OUT     PORTD,LED_REC
              RJMP    RET_SHOW

RET_SHOW_15:
              SBI     PORTD,3
              SBI     PORTD,4
              SBI     PORTD,5
              SBI     PORTD,6

RET_SHOW:    RET
;*****
; SUB PROGRAM WRITE & READ DATA LED IN EEPROM
;*****
EEP_WRITEL:  SBIC     EECR,EWE
              RJMP    EEP_WRITEL
              LDI     ADDR_EEP,$00
              OUT     EEARH,ADDR_EEP
              LDI     ADDR_EEP,$16
              OUT     EEARL,ADDR_EEP
              OUT     EEDR,LED_REC
              SBI     EECR,EEMWE
              SBI     EECR,EWE
              RET

EEP_READL:   SBIC     EECR,EWE
              RJMP    EEP_READL
              LDI     ADDR_EEP,$00
              OUT     EEARH,ADDR_EEP
              LDI     ADDR_EEP,$16
              OUT     EEARL,ADDR_EEP
              SBI     EECR,EERE

```

```
IN      LED_REC,EEDR
RET
```

```
*****
;
;      *** END OF SUB PROGRAM SUB_3_21.SUB ***
;
*****
```

```
*****
// Project: Digital Automatic Answering Machine via Internet
//      Network.
// Date:   June 29, 2002
// File:   DIAM10.JAVA
// Title:  Java Servlet programming for checks voice
//      messages via internet.
//*****

import java.io.*;
import java.text.*;
import java.util.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

public class DIAM10 extends HttpServlet {
    ResourceBundle rb = ResourceBundle.getBundle("LocalStrings");
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Digital Answering Machine Home
Page...SSL Secured(128 Bit)</TITLE>");
        out.println("</head>");
        out.println("<body>");
        // Create an object: directory
        File myDir = new File("C:/jakarta-tomcat-
3.2.3/webapps/ROOT/Voice");
        out.println("<a href='\"/Project/index.html\"'>");
        out.println("<img src='\"/examples/images/return.gif\"' height=24
" +
            "width=24 align=right border=5 alt='\"return\"'></a>");
        out.println("Digital Answering Machine Home Page...SSL
Secured(128 Bit)");
        out.println("<p>");
        HttpSession session = request.getSession(true);
        // print session info
        Date created = new Date(session.getCreationTime());
        Date accessed = new Date(session.getLastAccessedTime());
```

```

// Get the contents of the directory
File[] contents = myDir.listFiles();
// List the contents of the directory
if(contents!=null)
{
    out.println("\nYou have " + contents.length
        + " Voice Messages ...Please Click");
    out.println("<HR>");
}

String title = rb.getString("requestinfo.title");
out.println("<title>" + title + "</title>");
out.println("</head>");
out.println("<body bgcolor=\\"white\\">");
out.println("<p>");
out.println("Created: " + created);
out.println("<p>");
out.println("Last Accessed: " + accessed);
out.println("<p>");
out.println("<p>");
int number = 24;//number of message in voice mail box
double play_time = 0.0;
for(int i = 0; i < contents.length ; i++)/(int i = 1; i < number; i++)
{
    File f = new File("R0000000.wav");
    if (f.exists() ){
        out.println("<a href=\\"/Voice/R0000000"+ i +".wav\\">Voice#" + i
+ " ");
        out.println("</a>");
        out.println(" On:"+ new Date(contents[i].lastModified()));
        out.println(",Size: " + ((contents[i].length()/1024)+ " kB");
        DecimalFormat fnt = new DecimalFormat("0.00");
        play_time = (2.250001)*((double)(contents[i].length()*(double)
(0.0000277689071546)));
        out.println(",Length: "+ fnt.format(play_time) + " Sec");
        out.println("<p>");
        out.println("</a>");
        out.println("<p>");
        out.println("</a>");
        out.println("<p>");
        out.println("<p>");
    }
}

```

```

out.println("<HR>");
out.println("</table>");
out.println("Last Accessed Mail Box on: " + accessed);
out.println("<p>");
out.println(rb.getString("requestinfo.label.remoteaddr"));
out.println("</td><td>");
out.println(request.getRemoteAddr());
out.println(" (Your IP Address)</p>");
Enumeration e = request.getHeaderNames();
while (e.hasMoreElements()) {
    String headerName = (String)e.nextElement();
    String headerValue = request.getHeader(headerName);
}
}

public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException
{
    doGet(request, response);
}
}

//End of file.

```

ภาคผนวก ข

บทความเผยแพร่งานประชุมวิชาการ EECON-25

เครื่องตอบรับโทรศัพท์แบบดิจิทัลผ่านอินเทอร์เน็ต

Digital Answering Machine via Internet

ประโยชน์ คำสวัสดิ์

สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

อ. เมือง จ.นครราชสีมา 30000

โทร 0-4422-4394 โทรสาร 0-4422-4220 E-mail: prayoth@ccs.sut.ac.th

บทคัดย่อ

บทความนี้นำเสนอเครื่องตอบรับโทรศัพท์แบบดิจิทัลที่สามารถทำงานได้บน 3 โครงข่ายหลัก คือ โครงข่ายระบบโทรศัพท์พื้นฐาน (Public Switched Telephone Networks) โครงข่ายโทรศัพท์เคลื่อนที่ (Mobile Phone Networks) และโครงข่ายอินเทอร์เน็ต (Internet Networks) ผู้ใช้งานสามารถโทรศัพท์เข้าไปทำการฝากหรือตรวจสอบข้อความโดยใช้โทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่ เนื่องจากเครื่องตอบรับโทรศัพท์แบบดิจิทัลนี้ถูกออกแบบให้มีการเชื่อมต่อกับเว็บเซิร์ฟเวอร์ส่วนบุคคล (Personal Web Server) ที่สนับสนุนโปรโตคอล HTTP (Hypertext Transfer Protocol) ซึ่งถูกเก็บซ่อนอยู่ในโปรโตคอล SSL (Secure Socket Layer) หรือที่เรียกว่า HTTPS ทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตมีความปลอดภัยสูงมาก และจากการนำเทคโนโลยีจาวาเซิร์ฟเล็ต (Java Servlet Technology) มาใช้ในการเขียนโปรแกรม ทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตได้รับข้อความล่าสุดตลอดเวลาการทำงานของเครื่องตอบรับโทรศัพท์เป็นแบบอัตโนมัติซึ่งใช้ไมโครคอนโทรลเลอร์ควบคุม ผลการทดสอบการใช้งานพบว่าเครื่องตอบรับโทรศัพท์ดังกล่าวสามารถทำงานได้ตามที่ออกแบบไว้อย่างถูกต้อง

Abstract

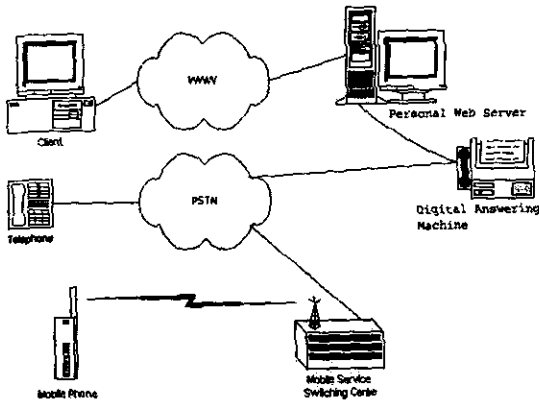
This paper presents a digital answering machine that can function on 3 networks, i.e. Public Switched Telephone Networks, Mobile Phone Networks, and Internet Network. The message of an incoming call via a telephone or mobile one can be left with the machine. Since this machine can be connected to the personal web server that supports the Hypertext Transfer Protocol (HTTP) encapsulated in the SSL (Secure Socket Layer) protocol, called shortly as HTTPS. HTTPS provides high security when it interrogates and transfers voice message via Internet Network. Determining the up-to-date voice message is possible using Java Servlet Technology. The automatic function of the proposed machine is supervised by a microcontroller and the tested results show that it functions properly.

Keywords: Digital Answering Machine, Java Servlet Technology, HTTPS, SSL.

1. บทนำ

ปัจจุบันการติดต่อสื่อสารโดยใช้โทรศัพท์พื้นฐานและโทรศัพท์เคลื่อนที่มีบทบาทสำคัญมาก การพลาดการติดต่อทางโทรศัพท์เพียงครั้งเดียวอาจนำมาซึ่งความสูญเสียเป็นมูลค่ามหาศาลโดยเฉพาะการติดต่อทางธุรกิจ ดังนั้นในแต่ละสำนักงานจะมีเครื่องตอบรับโทรศัพท์ติดตั้งด้วยเสมอ เครื่องตอบรับโทรศัพท์แบบดิจิทัล [1] [2] นั้นจะติดต่อสื่อสารกันได้เฉพาะโครงข่ายโทรศัพท์ด้วยกันเท่านั้น จึงเกิดปัญหาในกรณีที่จะต้องเดินทางไปไกล ๆ เช่นเมื่อเดินทางไปต่างประเทศ หากต้องการจะทราบว่าบุคคลใดโทรศัพท์เข้ามายังที่ทำงานและฝากข้อความไว้ ต้องโทรศัพท์ทางไกลระหว่างประเทศกลับมาที่เครื่องตอบรับโทรศัพท์เพื่อตรวจสอบข้อความซึ่งต้องสิ้นเปลืองค่าใช้จ่ายสูง ปัญหานี้ อาจแก้ไขได้หากสามารถทำการตรวจสอบข้อความผ่านเว็บเบราว์เซอร์ตามอินเทอร์เน็ตคาเฟ่ที่มีอยู่ทั่วไป โครงการวิจัยนี้จึงให้ความสนใจไปที่การออกแบบเครื่องตอบรับโทรศัพท์แบบดิจิทัลที่สามารถใช้งานได้บน 3 โครงข่ายหลักเพื่อให้สามารถติดต่อสื่อสารกับบุคคลอื่น ๆ ได้ตลอดเวลาโดยสิ้นเปลืองค่าใช้จ่ายน้อยมาก

การประยุกต์ใช้โครงข่ายที่กล่าวถึงแสดงในรูปที่ 1 โดยเครื่องตอบรับโทรศัพท์แบบดิจิทัลจะเชื่อมต่อกับเครื่องคอมพิวเตอร์ที่ทำงานเป็นเว็บเซิร์ฟเวอร์ส่วนบุคคลที่สนับสนุนเทคโนโลยีจาวาเซิร์ฟเล็ต ซึ่งสามารถใช้งานได้จริง [3] โดยติดตั้งโปรแกรม PWS4.0 หรือ IIS5.0 พร้อมกับระบบปฏิบัติการ Windows 98 หรือ Windows 2000 ตามลำดับ และใช้งานร่วมกับโปรแกรม Jakarta-Tomcat 3.2.3 และ Java 2 SDK 1.3 ซึ่งสามารถทำการดาวน์โหลดได้โดยไม่ต้องเสียค่าใช้จ่ายจาก <http://jakarta.apache.org/downloads/binindex.html> และ <http://java.sun.com/downloads.html> ตามลำดับ เครื่องตอบรับโทรศัพท์แบบดิจิทัลก็จะเสมือนเป็นตัวเชื่อมโยงโครงข่ายหลักทั้ง 3 โครงข่ายเข้าด้วยกัน



รูปที่ 1 การใช้งานเครื่องคอมพิวเตอร์รับโทรศัพท์แบบดิจิทัล

2. ทฤษฎีที่เกี่ยวข้อง

2.1 เทคโนโลยีภาษาเซิร์ฟเล็ต

Java Servlet หรือเรียกสั้น ๆ ว่า Servlet เป็นโปรแกรมที่มีการทำงานบนฝั่งเซิร์ฟเวอร์ (Server Side Application) มีหลักการทำงานคล้ายกับ CGI (Common Gateway Interface) แต่ Servlet ไม่มีข้อเสีย [4] [5] ประการแรกคือตัวภาษาที่ใช้ในการเขียนโปรแกรมเป็นภาษาจาวา (Java) ซึ่งใช้หลักการโปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งสามารถลดความซับซ้อนโครงสร้างของโปรแกรมรวมไปถึงอำนวยความสะดวกในการนำส่วนประกอบต่าง ๆ ของโปรแกรมที่เขียนไว้แล้วกลับมาใช้ได้อีก นอกจากนี้จาวายังเป็นภาษาที่ไม่ขึ้นกับแพลตฟอร์ม (Platform Independent) ซึ่งช่วยให้สามารถทำงานหรือพัฒนาโปรแกรมได้ในทุกระบบปฏิบัติการ ประการที่สองเซิร์ฟเล็ตมีความเร็วที่เหนือกว่า CGI เพราะเซิร์ฟเล็ตใช้หลักการของเธรด (Thread) โดยจะทำการสร้าง 1 เธรดต่อหนึ่งการร้องขอ (Request) ที่มาจากไคลเอนต์ (Client) ในขณะที่ CGI จะทำการสร้าง 1 กระบวนการ (Process) ต่อหนึ่งการร้องขอ ซึ่งจะทำให้เปลืองทรัพยากรของระบบมากกว่า และในแต่ละกระบวนการนี้การทำงานช้ากว่า

ในการที่จะสั่งให้เซิร์ฟเล็ตทำงาน ตัวเว็บเซิร์ฟเวอร์จะไม่สามารถส่งข้อมูลไปให้เซิร์ฟเล็ตได้โดยตรงเหมือนกับ CGI ดังนั้นตัวเว็บเซิร์ฟเวอร์ จะต้องมีส่วนเป็นเสมือนตัวต่อเชื่อมเซิร์ฟเล็ตต่าง ๆ ไว้โดยส่วนนี้เรียกว่า เซิร์ฟเล็ตเอนจิน (Servlet Engine) หรือเซิร์ฟเล็ตคอนเทนเนอร์ (Servlet Container) โดยทั่วไปเซิร์ฟเล็ตเอนจินจะเป็นส่วนที่มี Java Virtual Machine (JVM) อยู่ในตัวเองโดยเซิร์ฟเล็ตเอนจินจะมีหน้าที่รับการร้องขอจากเว็บเบราว์เซอร์แล้วทำการเลือกตัวเซิร์ฟเล็ตขึ้นมาทำการประมวลผลภายใต้ JVM โดยผลลัพธ์ที่ได้จากการประมวลผลของเซิร์ฟเล็ตที่ถูกเลือกจะถูกส่งกลับไปยังเว็บเซิร์ฟเวอร์โดยเว็บเซิร์ฟเวอร์นี้จะส่งผลกลับไปยังเว็บเบราว์เซอร์ในฝั่งไคลเอนต์อีกทอดหนึ่ง การส่งผลลัพธ์ต่าง ๆ กลับไปให้ไคลเอนต์นั้น เซิร์ฟเล็ตจะต้องทำการเซต Header เพื่อกำหนดชนิดของ MIME (Multipurpose Internet Mail

Extensions) ของสายข้อมูล (Stream) ที่ส่งออกไป โดย Header นี้จะเป็นตัวบอกไคลเอนต์ว่าสายข้อมูลที่กำลังจะได้รับเป็นสายข้อมูลชนิดไหน โดยเซิร์ฟเล็ตสามารถส่งสายข้อมูลออกไปได้ 2 แบบคือ สายข้อมูลเท็กซ์ (text stream) และสายข้อมูลไบนารี (byte stream) ในกรณีของงานวิจัยนี้จะมีการส่งสายข้อมูลออกไปเป็นเท็กซ์หรือ HTML (Hyper Text Markup Language) นั่นคือจะมีการสร้างเอกสาร HTML ในขณะที่เซิร์ฟเล็ตทำงาน การตรวจสอบข้อความผ่านอินเทอร์เน็ตจึงได้รับข้อความที่ทันสมัยตลอดเวลา

จะเห็นว่าเซิร์ฟเล็ตเป็นโปรแกรมที่มีการประมวลผลบนเว็บเซิร์ฟเวอร์แล้วส่งข้อมูลผลลัพธ์ไปยังไคลเอนต์โดยเซิร์ฟเล็ตจะทำการสร้างเอกสาร HTML กลับไปให้ไคลเอนต์โดยยึดตามมาตรฐาน HTML และ/หรือ XML (Extensible Markup Language) เป็นหลัก [4] ทำให้เว็บเซิร์ฟเวอร์ใช้งานจากผู้ใช้ได้หลากหลายเพราะไม่อิงกับเครื่องผู้ใช้ และเครื่องไคลเอนต์ที่ใช้ติดต่อกับเว็บเซิร์ฟเวอร์ก็ไม่จำเป็นต้องมีทรัพยากรของระบบมากเพราะหน้าที่หลักคือแปล (Render) เอกสาร HTML เพื่อแสดงผลบนเว็บเบราว์เซอร์เท่านั้น เทคโนโลยีภาษาเซิร์ฟเล็ตจึงเหมาะกับการใช้งานในโครงการวิจัยนี้

2.2 การบันทึกข้อมูลเสียงแบบดิจิทัล

เครื่องคอมพิวเตอร์ที่บันทึกข้อมูลที่มีอยู่ทั่วไปนั้น ส่วนมากมีการจัดเก็บข้อมูลเสียงแบบแอนะล็อกซึ่งจัดเก็บโดยการบันทึกลงในเทปคาสเซต การเปิดฟังข้อความเสียงแต่ละข้อความทำได้ช้าและอายุการใช้งานที่สั้น ข้อจำกัดของเทคโนโลยีแบบเดิมนี้สามารถแก้ไขด้วยเทคโนโลยีของวงจรรวมที่ทำให้อิทธิพลหน่วยความจำความจุสูง ๆ และเหมาะสำหรับการเก็บข้อมูลแบบดิจิทัล เช่น ไอซี W51300 ไอซีดังกล่าวใช้วิธีการเข้ารหัสข้อมูลแบบดิจิทัลที่เรียกว่า การมอดูเลตแบบลดคาที่ปรับตัวเองได้ (Adaptive Delta Modulation, ADM) ซึ่งสามารถควบคุมคุณภาพ ความชัดเจนและขนาดของข้อมูลเสียงได้จากการเปลี่ยนแปลงความถี่ของการสุ่มตัวอย่าง ขนาดของข้อมูลจะลดลงเมื่อความถี่ของการสุ่มตัวอย่างลดลงแต่คุณภาพ ความชัดเจนของเสียงก็จะลดลงด้วย ในการใช้งานจึงต้องเลือกทั้งคุณภาพ ความชัดเจนของเสียงและขนาดของข้อมูลให้เหมาะสม [1] โดยในงานวิจัยนี้ใช้ความถี่ของการสุ่มตัวอย่าง 15 kHz ข้อมูลเสียงแบบดิจิทัลที่ได้จากการเข้ารหัสข้างต้นจะถูกจัดเก็บในไอซี W55F20 ซึ่งเป็นไอซีหน่วยความจำขนาด 2 เมกะบิต แบบ Flash EEPROM มีอายุการเก็บรักษาข้อมูล 10 ปี

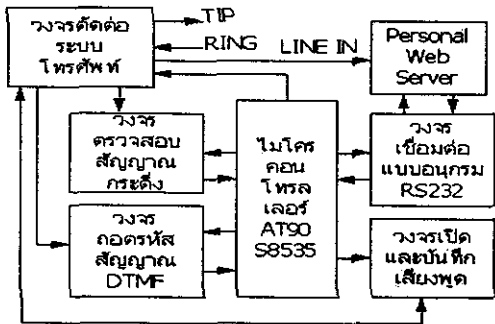
2.3 ไมโครคอนโทรลเลอร์ AVR

ไมโครคอนโทรลเลอร์ที่ใช้ในงานวิจัยนี้เป็นแบบ 8 บิต กระตุล AVR เบอร์ AT90S8535 มีโครงสร้างการทำงานและสถาปัตยกรรมแบบ RISC (Reduced Instruction Set Computer) ประมวลผลด้วยความเร็ว 1 MIPS (Million Instructions Per Second) มีหน่วยความจำขนาด 8 kB แบบ Flash Memory จึงสามารถทำการเขียนหรือลบข้อมูลในหน่วยความจำได้สะดวก ทำให้การพัฒนาหรือเปลี่ยนแปลงระบบควบคุม

ทำได้ง่ายและรวดเร็ว ส่วนภาษาที่ใช้ในการโปรแกรมเป็นภาษาเอซเซมบลีของ AVR

1. หลักการทำงานและการออกแบบระบบ

การทำงานของระบบเป็นแบบอัตโนมัติเมื่อมีโทรศัพท์เรียกเข้ามาในระบบจะรอประมาณ 20 วินาทีและเมื่อไม่มีผู้รับสายระบบจะรับสายแทนซึ่งจะมีเสียงคอยแนะนำการใช้งานให้กับผู้ที่โทรศัพท์เข้ามา เช่น การฝากข้อความ การป้อนรหัสผ่าน การตรวจสอบข้อความ และถ้าไม่มีการกดคีย์ใด ๆ ภายใน 20 วินาทีระบบจะวางสายเอง สำหรับการตรวจสอบข้อความสามารถเปิดฟังได้ที่เครื่องโดยตรงหรือทำการตรวจสอบผ่านโทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่ นอกจากนี้ยังสามารถตรวจสอบผ่านอินเทอร์เน็ตได้อีก หลักการทำงานและการออกแบบระบบจะแยกอธิบายเป็นส่วนของฮาร์ดแวร์และส่วนของซอฟต์แวร์ ดังนี้



รูปที่ 2 แผนภาพบล็อกของระบบ

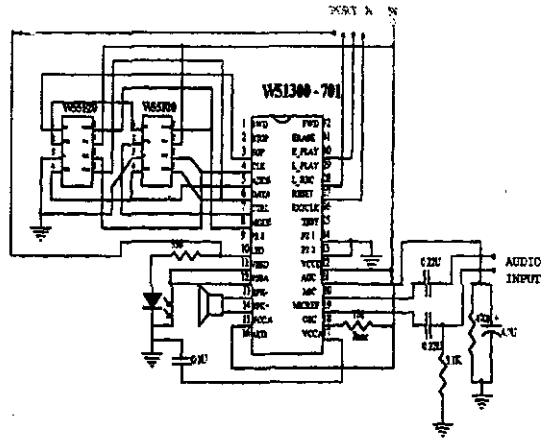
3.1 การทำงานและการออกแบบฮาร์ดแวร์

จากแผนภาพบล็อกในรูปที่ 2 ไมโครคอนโทรลเลอร์มีหน้าที่หลักคือ ควบคุมการทำงานของวงจรติดต่อระบบโทรศัพท์ วงจรตรวจสอบสัญญาณกระดิ่ง วงจรถอดรหัสสัญญาณ DTMF (Dual Tone Multi Frequency type) วงจรเปิดและบันทึกเสียงพูดและวงจรเชื่อมต่อแบบอนุกรมระหว่างไมโครคอนโทรลเลอร์และเครื่องเว็บเซิร์ฟเวอร์

วงจรถอดรหัสสัญญาณ DTMF จะทำหน้าที่แปลงสัญญาณความถี่ที่เกิดจากการกดคีย์โทรศัพท์ให้เป็นเลขฐาน 2 ความยาว 4 บิตแล้วส่งให้ไมโครคอนโทรลเลอร์ประมวลผล ทำให้ทราบว่าผู้ที่โทรศัพท์เข้ามาต้องการรับบริการฝากหรือตรวจสอบข้อความ แล้วระบบก็จะให้บริการและคอยส่งเสียงแนะนำการใช้งานตามบริการนั้น ๆ

วงจรเปิดและบันทึกเสียงพูดนั้นใช้ไอซี WS1300 เป็นตัวดำเนินการหลักโดยมีการควบคุมการทำงานผ่านพอร์ต A ของไมโครคอนโทรลเลอร์ วงจรนี้ทำหน้าที่เปิดเสียงแนะนำการใช้งานและบันทึกข้อความเสียงแบบดิจิทัลเก็บไว้ได้ 4 ข้อความ ซึ่งข้อความส่วนนี้จะถูก

เก็บไว้ใช้สำหรับการตรวจสอบข้อความที่ตัวเครื่องโดยตรงและผ่านโครงข่ายโทรศัพท์พื้นฐานและโทรศัพท์เคลื่อนที่

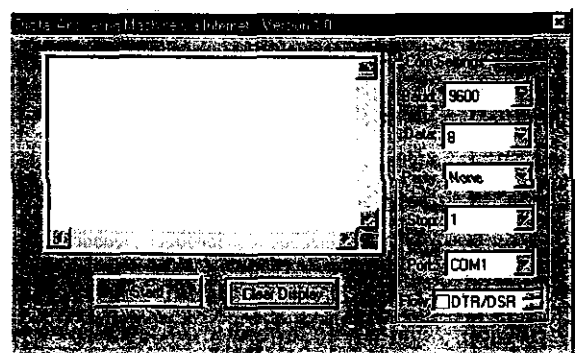


รูปที่ 3 วงจรเปิดและบันทึกข้อความเสียงพูด

สำหรับการเชื่อมต่อกับเครื่องเว็บเซิร์ฟเวอร์นั้นใช้การเชื่อมต่อแบบอนุกรมมาตรฐาน RS232 (RS232 Serial Interface) เมื่อมีโทรศัพท์เข้ามาแล้วเครื่องตอบรับโทรศัพท์ถูกเลือกโหมดการทำงานเป็นการฝากข้อความ ระบบจะมีการบันทึกข้อความไว้ที่เครื่องเว็บเซิร์ฟเวอร์ด้วย โดยไมโครคอนโทรลเลอร์จะส่งสัญญาณควบคุมผ่านการเชื่อมต่อดังกล่าวให้เครื่องเว็บเซิร์ฟเวอร์ทำการบันทึกข้อความนั้น โดยสัญญาณเสียงแบบแอนะล็อกจะป้อนผ่านช่อง LINE IN ของการ์ดเสียงและถูกบันทึกเก็บไว้สำหรับการตรวจสอบข้อความทางผ่านอินเทอร์เน็ตต่อไป

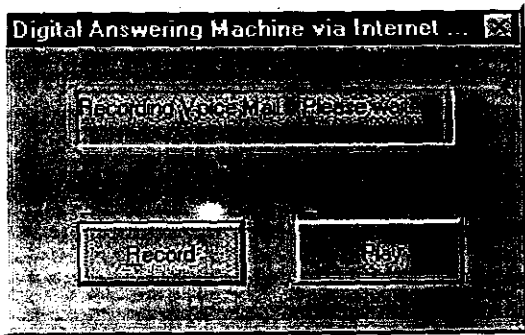
3.2 การทำงานและการออกแบบซอฟต์แวร์

โปรแกรมเชื่อมต่อระหว่างเครื่องเว็บเซิร์ฟเวอร์กับเครื่องคอมพิวเตอร์โทรศัพท์ผ่านพอร์ตอนุกรม (Serial Port) มีการโปรแกรมโดยใช้ภาษา Visual C++ 6.0 โปรแกรมนี้จะทำหน้าที่ส่งผ่านสัญญาณควบคุมการบันทึกข้อความและจะทำงานตลอดเวลาที่เครื่องเว็บเซิร์ฟเวอร์ทำงานเพื่อจะได้ไม่พลาดการติดต่อใด ๆ จากการสื่อสารผ่านโครงข่ายโทรศัพท์



รูปที่ 4 โปรแกรมการเชื่อมต่อผ่านพอร์ตอนุกรม

สำหรับโปรแกรมที่ทำหน้าที่บันทึกข้อความบนเครื่องเว็บเซิร์ฟเวอร์นั้นใช้ภาษา Visual C++ 6.0 ซึ่งโปรแกรมดังกล่าวจะถูกเรียกใช้งานเมื่อมีการบันทึกข้อความเท่านั้น โดยมีการตั้งเวลาให้โปรแกรมทำการบันทึกข้อความประมาณ 60 วินาที หลังจากนั้นโปรแกรมจะปิดตัวลงเพื่อคืนทรัพยากรให้กับระบบต่อไป การบันทึกข้อความดังกล่าวสามารถบันทึกเก็บไว้บนเครื่องเว็บเซิร์ฟเวอร์ 20 ข้อความ โดยกำหนดรูปแบบเป็นไฟล์มัลติมีเดียของระบบปฏิบัติการ Windows ซึ่งใช้การเข้ารหัสข้อมูลแบบ PCM (Pulse Code Modulation) มีอัตราการชักตัวอย่าง 8 kHz ใช้จำนวนบิตในการเข้ารหัส 16 บิตและระบบเสียงเป็นแบบโมโน



รูปที่ 5 โปรแกรมบันทึกข้อความบนเครื่องเว็บเซิร์ฟเวอร์

งานวิจัยนี้มีการออกแบบให้เว็บเซิร์ฟเวอร์สนับสนุนโปรโตคอล HTTP ซึ่งถูกเก็บซ่อนอยู่ในโปรโตคอล SSL หรือที่เรียกว่า HTTPS ดังนั้นการส่งข้อมูลใด ๆ ก็ตามระหว่างไคลเอนต์กับเซิร์ฟเวอร์จะมีการ Encrypt ข้อมูลก่อนเสมอ ทำให้การส่งข้อมูลผ่านโครงข่ายอินเทอร์เน็ตมีความปลอดภัยสูงมาก

ในการตรวจสอบข้อความผ่านอินเทอร์เน็ตเครื่องไคลเอนต์ต้องมีโปรแกรมเว็บเบราว์เซอร์เพื่อใช้ในการติดต่อไปยังโฮมเพจที่สร้างไว้บนเว็บเซิร์ฟเวอร์โดยใช้ HTTPS และต้องมีโปรแกรมมัลติมีเดียสำหรับเล่นไฟล์เสียง เช่นโปรแกรม Windows Media Player หรือโปรแกรม Winamp รวมทั้งต้องมีการเชื่อมต่อเสียงและลำโพงหรือหูฟังอยู่ด้วย

เมื่อโปรแกรมเว็บเบราว์เซอร์ ติดต่อก้าวเข้าไปที่โฮมเพจของเครื่องตอบรับโทรศัพท์แบบดิจิทัลเพื่อทำการตรวจสอบข้อความเว็บเซิร์ฟเวอร์จะมีระบบรักษาความปลอดภัย โดยให้ผู้ใช้งานป้อนชื่อและรหัสผ่านก่อน เมื่อข้อมูลถูกต้องจึงจะสามารถตรวจสอบข้อความได้ ในขั้นตอนการตรวจสอบข้อความนั้น เว็บเบราว์เซอร์จะส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์ แล้วที่ฝั่งเซิร์ฟเวอร์จะมีการทำงานของโปรแกรมที่เขียนโดยใช้ภาษาซี++ ซึ่งโปรแกรมจะทำการตรวจนับไฟล์เสียงพร้อมทั้งเก็บรวบรวมข้อมูลต่าง ๆ ของไฟล์เสียงเพื่อใช้แสดงผล เช่น วัน เดือน ปี และเวลาที่ข้อความถูกฝากไว้ ขณะเดียวกันเว็บเซิร์ฟเวอร์ก็จะส่งสายข้อมูลซึ่งเป็นผลลัพธ์จากการทำงานของโปรแกรมดังกล่าวกลับไปยังไคลเอนต์เพื่อแสดงผลบนเว็บเบราว์เซอร์ ผู้ใช้งานก็จะทราบข้อมูลล่าสุด

จากการแสดงผลนี้ และสามารถคลิกบนลิงค์ที่ภาษาซี++ ให้เพื่อฟังข้อความได้ทันที จะเห็นว่าเอกสาร HTML ถูกสร้างในขณะที่มีการทำงานของโปรแกรมภาษาซี++ ดังนั้นข้อมูลที่แสดงผลผ่านเว็บเบราว์เซอร์ออกไปจึงทันสมัยตลอดเวลา ทำให้การตรวจสอบข้อความถูกต้องและทันทั่วถึง

4. ผลการทดสอบการใช้งาน

ในขั้นต้นได้ทดสอบการบันทึกข้อความเสียงแบบดิจิทัลลงในไอซีหน่วยความจำ W55F20 ขนาด 4 เมกกะบิต โดยกำหนดความถี่ของการชักตัวอย่าง 15 kHz ซึ่งจะได้บิตเรตเท่ากับ 15 kbit/sec ระยะเวลาในการบันทึกข้อมูลเสียงพูดสามารถคำนวณได้จาก 4 Mbit / 15 kbit/sec เท่ากับ 267 วินาที หรือ 4 นาที 27 วินาที ในการออกแบบกำหนดเวลาการบันทึกข้อความไว้ที่ 60 วินาที จำนวน 4 ข้อความ โดยหน่วยความจำส่วนที่เหลือจะใช้ในการแยกแยะขนาดของข้อความ ผลการทดสอบพบว่าสามารถบันทึกข้อความได้ 4 ข้อความตามเวลาที่ออกแบบจริง

ลำดับต่อมาได้ทดสอบการฝากและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์ โดยฝากข้อความที่มีความยาว 60 วินาทีแล้ววางสายจำนวน 4 ครั้ง จากนั้นโทรศัพท์เข้ามาตรวจสอบข้อความที่ฝากไว้ครั้งละ 1 ข้อความ เวลาเฉลี่ยในการฝากและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์ที่แสดงในตารางที่ 1

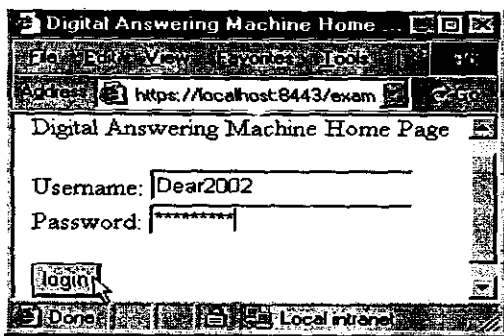
ตารางที่ 1 เวลาเฉลี่ย (วินาที) ที่ใช้ในการฝากและตรวจสอบข้อความผ่านโครงข่ายโทรศัพท์

บริการ	โทรศัพท์บ้าน	โทรศัพท์สาธารณะ	โทรศัพท์เคลื่อนที่ (DTAC1800)	โทรศัพท์เคลื่อนที่ (GSM900)
ฝากข้อความ	86.42	85.78	82.36	82.82
เช็คข้อความ	98.11	95.35	85.66	85.54

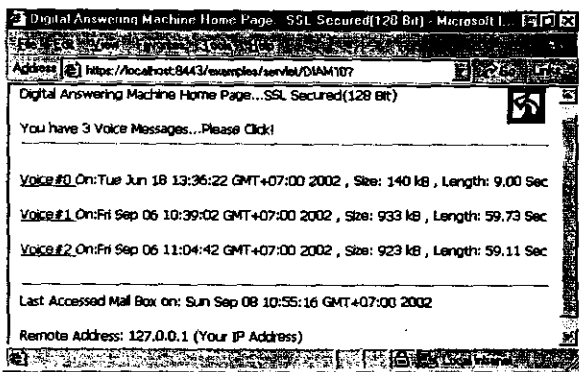
ลำดับสุดท้ายเป็นการทดสอบการตรวจสอบข้อความผ่านอินเทอร์เน็ตภายในองค์กร เครื่องเว็บเซิร์ฟเวอร์ที่ใช้ทดสอบเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ใช้ระบบปฏิบัติการ Windows 98 SE และติดตั้งโปรแกรม Jakarta-Tomcat 3.2.3 Java 2 SDK 1.3 และ JSSE 1.0.3 โดยเริ่มจากการล็อกอินเข้าโฮมเพจ ตรวจสอบข้อความและการแสดงผลบนเว็บเบราว์เซอร์ ซึ่งพิจารณาถึงความครบถ้วนและความถูกต้องของข้อมูล เช่น วัน เดือน ปี และเวลาในการสร้างไฟล์เสียงโดยเทียบกับ File Properties ในโปรแกรม Windows Explorer และทดสอบการฟังข้อความผ่านโครงข่ายดังกล่าว ผลการทดสอบแสดงในตารางที่ 2 และการแสดงผลบนเว็บเบราว์เซอร์แสดงดังในรูปที่ 6 และ 7

ตารางที่ 2 ผลการตรวจสอบข้อความโดยใช้โปรแกรมต่างกัน

โปรแกรม ที่ใช้ทดสอบ	การ แสดงผล	ความ ถูกต้อง ของข้อมูล	การฟังข้อความเสียง	
			Media Player 6.0	Winamp 2.71
Internet Explorer 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน
Opera 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน
Netscape 6.0	ครบถ้วน	ถูกต้อง	ได้ ชัดเจน	ได้ ชัดเจน



รูปที่ 6 การล็อกอินเข้าโฮมเพจเพื่อตรวจสอบข้อความ



รูปที่ 7 การตรวจสอบข้อความโดยใช้ Internet Explorer 6.0

5. สรุป

เครื่องตอบรับโทรศัพท์แบบดิจิทัล สามารถทำงานได้บน 3 โครงข่ายหลัก ผู้ใช้งานสามารถโทรศัพท์เข้าไปทำการฝากหรือตรวจสอบข้อความโดยใช้โทรศัพท์พื้นฐานหรือโทรศัพท์เคลื่อนที่ สามารถฝากข้อความได้สูงสุด 24 ข้อความ เครื่องตอบรับโทรศัพท์ดังกล่าวนี้สามารถเชื่อมต่อกับเครื่องเว็บเซิร์ฟเวอร์ส่วนบุคคล ทำให้สามารถทำการตรวจสอบข้อความผ่านอินเทอร์เน็ตได้และมีความปลอดภัยของข้อมูลสูงมาก และจากการนำเทคโนโลยีจาวาเซิร์ฟเวอร์เข้ามาใช้ในการเขียนโปรแกรมบน

ฝั่งเซิร์ฟเวอร์ จึงทำให้การตรวจสอบข้อความผ่านอินเทอร์เน็ตได้รับข้อความล่าสุดตลอดเวลาและผลการทดสอบการใช้งานพบว่าเครื่องตอบรับโทรศัพท์ดังกล่าวสามารถทำงานได้ตามที่ออกแบบไว้อย่างถูกต้อง

สำหรับการพัฒนาต่อไปในอนาคต คือ

- 1.ปรับปรุงการทำงานให้สามารถทำการฝากข้อความจากสมมติพจนานุกรมข้อมูลอินเทอร์เน็ตมาซึ่งเครื่องตอบรับโทรศัพท์แบบดิจิทัล
- 2.เชื่อมต่อเว็บเซิร์ฟเวอร์ส่วนบุคคลกับฐานข้อมูลภายในองค์กรโดยใช้เทคโนโลยี JDBC (Java Database Connectivity) เพื่อให้สามารถใช้งานฐานข้อมูลดังกล่าวได้ทุกแห่งที่มีอินเทอร์เน็ตใช้งาน

6. กิตติกรรมประกาศ

ขอขอบคุณ มหาวิทยาลัยเทคโนโลยีสุรนารี ที่ให้ทุนสนับสนุนโครงการวิจัย รวมทั้ง รศ.น.ท.ดร.สราวุฒิ สุจิตจร และ อ.สมศักดิ์ วาณิชนันต์ชัย ที่ได้ให้คำแนะนำและมีส่วนช่วยเหลือในการเขียนบทความนี้

เอกสารอ้างอิง

- [1] ประภทกลุข แสงชูวงศ์ สุธี คนกระโทก ทวีศักดิ์ ศรีโปดก “เครื่องตอบรับโทรศัพท์และเปิด-ปิดอุปกรณ์ไฟฟ้าผ่านโครงข่ายโทรศัพท์” รายงานโครงงานวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี 2543, หน้า 29-39.
- [2] B. Glassmeyer, E. Melin “Digital Answering Machine” <http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/>
- [3] G. Shachor “Tomcat IIS Howto” <http://apache.org/>
- [4] Sun Microsystems “Java Servlet Technology” <http://java.sun.com/products/servlet/index.html>
- [5] A. Patzer et al “Professional Java Server Programming” Wrox Press Ltd, 2000, pp 7-317.

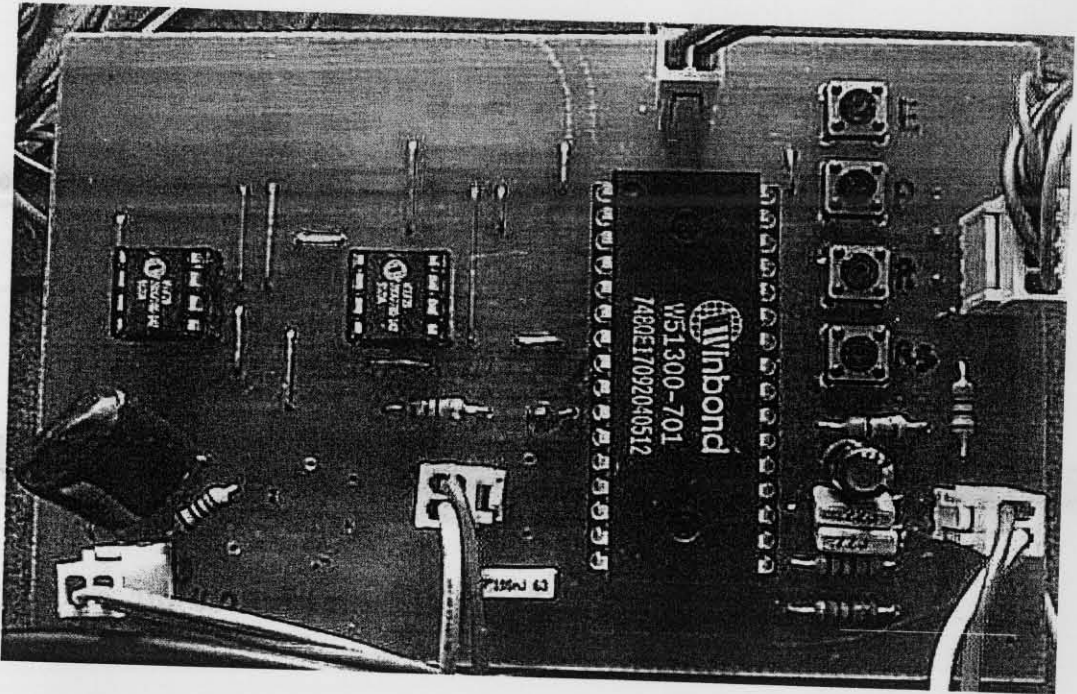


ประวัติผู้เขียนบทความ

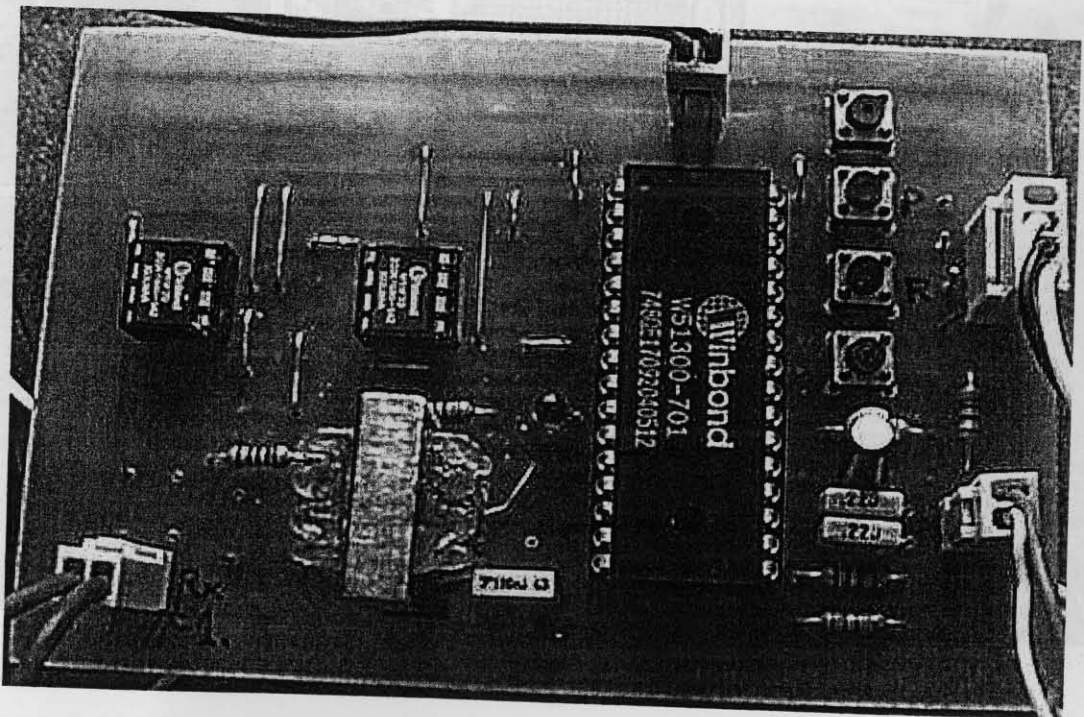
ประโชชน์ คำสวัสดิ์ สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิตสาขาวิศวกรรมไฟฟ้าจากโรงเรียนนายเรืออากาศและวิศวกรรมศาสตรมหาบัณฑิตสาขาเดียวกันจากมหาวิทยาลัยเกษตรศาสตร์ เมื่อปี พ.ศ. 2537 และ 2541 ตามลำดับ ปัจจุบันเป็นอาจารย์ประจำสาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี สนใจงานวิจัยเกี่ยวกับ Digital Signal and Image Processing และ Embedded and Real-time Systems.

ภาคผนวก ค

รูปภาพวงจรอิเล็กทรอนิกส์และอุปกรณ์การทดลองและวิจัย



รูปที่ ค.1 วงจรบันทึกข้อความเสียงพูด



รูปที่ ค.2 วงจรบันทึกข้อความและเปิดข้อความเสียงพูดผ่านโครงข่ายโทรศัพท์

