การส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่ายคอมพิวเตอร์

MOVING IMAGE TRANSMISSION THROUGH COMPUTER NETWORK

โดย

นาย ชยากริต สรีระศาสตร์

นาย สัญญา โพธิกลัด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

การส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่ายคอมพิวเตอร์

MOVING IMAGE TRANSMISSION THROUGH COMPUTER NETWORK

โดย

นาย ชยากริต สรีระศาสตร์ 43015006

นาย สัญญา โพธิกลัด 43015041

อาจารย์ที่ปรึกษา

รศ.ดร. ยุทธพงษ์ รังสรรค์เสรี

รศ.ดร. ปัญญา ฐิติมัชฌิมา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่ายคอมพิวเตอร์

MOVING IMAGE TRANSMISSION THROUGH COMPUTER NETWORK

ผู้จัดทำ

1. นาย ชยากริต สรีระศาสตร์ 43015006
2. นาย สัญญา โพธิกลัด 43015041

อาจารย์ที่ปรึกษา

(รศ.ดร.ยุทธพงษ์ รังสรรค์เสรี)

การส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่ายคอมพิวเตอร์

MOVING IMAGE TRANSMISSION THROUGH COMPUTER NETWORK

โดย นายชยากริต สรีระศาสตร์ 43015006
นายสัญญา โพธิกลัด 43015041

อาจารย์ที่ปรึกษา รศ.ดร.ยุทธพงษ์ รังสรรค์เสรี
รศ.ดร.ปัญญา ฐิติมัชฌิมา

บทคัดย่อ

โครงงานนี้มีจุดมุ่งหมายในการศึกษาเกี่ยวกับระบบการส่งผ่านข้อมูลที่เป็นภาพเคลื่อนไหวผ่านเครือข่ายคอมพิวเตอร์ โดยศึกษาถึงลักษณะของข้อมูลที่เหมาะสม การบีบอัดข้อมูล (Data Compression) และอัตราการส่งผ่านข้อมูลภายในระบบเครือข่ายคอมพิวเตอร์ที่เหมาะสม เพื่อให้ภาพที่แสดงอยู่ที่ไคลเอนมีความต่อเนื่องใกล้เคียงกับภาพที่แสดงอยู่ที่เซิร์ฟเวอร์ อีกทั้งศึกษาเกี่ยวกับการเรียกใช้งานของฟังก์ชัน API (Application Program Interface) และ Active X Control เพื่อมาสนับสนุนการทำงานให้มีประสิทธิภาพมากขึ้น

Abstract

This project intends to study the system of transferring moving image data through computer network. We study about package of information, data compression and ratio of data transmission in computer network system. Which the picture that show at the client with continuous to be close to the server that showed. Also we study about the usage of function API (Application Program Interface) and Active X Control to support the presentation much more efficiency.

สารบัญ

หน้า

บทคัดย่อ..... I

สารบัญ..... II

สารบัญ(ต่อ) III

สารบัญตาราง.....IV

สารบัญรูปภาพ..... V

สารบัญรูปภาพ(ต่อ).....VI

บทที่ 1 บทนำ..... 1

1.1 ความเป็นมา 1

1.2 วัตถุประสงค์..... 1

บทที่ 2 ทฤษฎีหรือหลักการ.....2

2.1 ทฤษฎีรูปแบบของภาพ..... 2

2.2 ระบบเครือข่าย.....5

2.3 การใช้งาน Visual Basic.....7

2.4 การควบคุมเครือข่ายโดย Winsock Control.....11

2.5 ชุดของฟังก์ชัน API.....20

บทที่ 3 การคำนวณและผลการทดลอง.....28

3.1 ส่วนประกอบของระบบ 28

3.2 โปรโตคอลที่ใช้ในการทดลอง..... 31

3.3 รูปแบบของภาพที่ใช้..... 31

3.4 การทำงานด้านฝั่งเซิร์ฟเวอร์..... 32

3.5 การทำงานทางด้านฝั่งไคลเอน.....35

บทที่ 4 การทดลองและผลการทดลอง.....37

4.1 อุปกรณ์ที่ใช้ในการทดลอง.....37

4.2 การทดลอง.....37

สารบัญ (ต่อ)

หน้า

บทที่ 5 บทสรุปและบทวิจารณ์.....55
5.1 บทสรุปและบทวิจารณ์ผลการทดลอง.....55

หนังสืออ้างอิง..... 57
ภาคผนวก ก. Source Code ด้านส่ง..... 58
ภาคผนวก ข. Source Code ด้านรับ.....66

สารบัญตาราง

ตารางที่ หน้า

2.1 การแบ่งคลาสของ IP Address ในแต่ละคลาส.....6

2.2 ชนิดของแอพพลิเคชันใน Visual Basic.....8

สารบัญรูปภาพ

รูปที่ หน้า

2.1 ลำดับภาพในแต่ละเฟรมที่ถูกเข้ารหัสแบบ MPEG.....4
2.2 การเลือก Network Connection จาก Control Panel..... 14
2.3 การเลือก Properties จาก Local Area Network..... 14
2.4 การเลือก Internet Protocol (TCP/IP) จาก Local Area Network Properties..... 13
2.5 หมายเลขของ IP Address ของเครื่องของเรา..... 13
3.1 บล็อกไดอะแกรมของระบบ 28
3.2 ขั้นตอนการทำงานของระบบ..... 30
3.3 ขั้นตอนการทำงานของฝั่งเซิร์ฟเวอร์..... 32
3.4 ขั้นตอนการทำงานเมื่อทำการติดต่อไปที่ฝั่งไคลเอน 33
3.5 ขั้นตอนของการ Capture..... 34
3.6 ขั้นตอนการทำงานด้านฝั่งไคลเอน..... 35
4.1 สภาวะเริ่มต้นเมื่อทำการรันโปรแกรม 38
4.2 การกดปุ่ม Connect ที่ Manu File 39
4.3 สภาวะลักษณะของการติดต่อ 39
4.4 สภาวะการเตือนไม่มีข้อมูลที่จะส่ง.....40
4.5 สภาวะการการเลือกข้อมูลที่จะส่ง..... 40
4.6 สภาวะการส่งข้อมูลไปทางไคลเอน..... 41
4.7 สภาวะเริ่มต้นเมื่อทำการรันแอพพลิเคชั่น.....41
4.8 สภาวะเมื่อทำการติดต่อได้ 42
4.9 สภาวะเมื่อกำลังทำการติดต่อ..... 42
4.10 สภาวะเมื่อกำลังทำการหาเครื่องตามหมายเลข IP address..... 43
4.11 สภาวะเมื่อฝ่ายไคลเอนทำการปิดแอพพลิเคชั่น.....43
4.12 สถานะรอการติดต่อ..... 44
4.13 สถานะไม่ได้รับการติดต่อ 44
4.14 สถานะได้รับการติดต่อ.....45
4.15 การทำการเลือกไฟล์ที่ต้องการจะส่ง.....45
4.16 การทำงานเมื่อมีการส่งข้อมูล.....46
4.17 การทำงานเมื่อมีการส่งข้อมูลเสร็จสิ้น 46

สารบัญรูปภาพ (ต่อ)

รูปที่ หน้า

4.18 สถานะไม่ได้รับการติดต่อ 47

4.19 การทำการเลือกไฟล์ที่ต้องการจะส่ง 47

4.20 สถานะไม่สามารถส่งข้อมูลได้ 48

4.21 สถานะไม่ใส่ค่า IP ADDRESS 48

4.22 สถานะเตือนให้ใส่ค่า IP ADDRESS 49

4.23 การทำการกดปุ่ม Connect 49

4.24 สถานะเมื่อมีการติดต่อได้แล้ว 50

4.25 สถานการเตือนเมื่อติดต่อกับไดร์เวอร์กล้องไม่ได้ 50

4.26 สถานะเมื่อติดต่อกับไดร์เวอร์กล้องไม่ได้ 50

4.27 สภาวะเมื่อฝ่ายไคลเอนทำการปิดแอพพลิเคชั่น 51

4.28 การทำการกดปุ่ม Connect 51

4.29 สถานะเมื่อมีการติดต่อได้แล้ว 52

4.30 สถานะไม่ได้รับการติดต่อ 52

4.31 สภาวะรอการติดต่อจากเครื่องเซิร์ฟเวอร์ 53

4.32 สภาวะเมื่อมีการติดต่อมาจากเครื่องเซิร์ฟเวอร์ 53

4.33 สภาวะกำลังรับข้อมูล 54

4.34 สภาวะเมื่อทำการรับข้อมูลเสร็จ 54

4.35 สภาวะเมื่อทำการเล่นภาพที่รับมาได้ 54

บทที่ 1
บทนำ

1.1 ความเป็นมา

เนื่องจากปัจจุบันนี้เทคโนโลยีทางด้านคอมพิวเตอร์ได้มีการพัฒนาไปจากเดิมเป็นอย่างมาก อีกทั้งกำลังเป็นที่นิยม เพราะปัจจุบันเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) สามารถหาซื้อได้ง่ายในราคาถูกลง มีขนาดเล็กลงแต่ประสิทธิภาพกลับเพิ่มมากขึ้น และจากเทคโนโลยีที่พัฒนาอย่างต่อเนื่องไม่มีที่สิ้นสุดนี้ทำให้เกิดโปรแกรมตัวใหม่ๆ ที่สามารถใช้งานได้โดยไม่ยากนัก ไม่เหมือนกับสมัยก่อนกว่าจะได้ทำการเขียนโปรแกรมควบคุมอะไรขึ้นมาสักชิ้นหนึ่ง ผู้เขียนจะต้องมีความชำนาญและประสบการณ์พอสมควรจึงจะสามารถสร้างสรรค์โปรแกรมแปลกๆใหม่ๆ ขึ้นมาได้ ด้วยเหตุนี้จึงเป็นเหตุผลให้เกิดโครงงานการส่งภาพเคลื่อนไหวผ่านระบบเครือข่าย (Moving Image Transmission Through Computer Network) ขึ้นมา โดยใช้โปรแกรม Visual Basic ในการพัฒนาแอพพลิเคชั่น เพื่อศึกษาถึงระบบการส่งผ่านข้อมูลที่มีขนาดใหญ่ เช่น ไฟล์รูปภาพ ไฟล์วีดีโอ โดยทำการส่งให้มีลักษณะที่ปรากฏที่ปลายทางใกล้เคียงกับภาพวีดีโอมากที่สุด ซึ่งเราจะเห็นว่าปัจจุบันระบบเครือข่ายที่ติดต่อสื่อสารกันทางคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันของเราเป็นอย่างมาก

1.2 วัตถุประสงค์

ปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาและทดลองเกี่ยวกับการส่งผ่านข้อมูลและการทำงานของระบบเครือข่ายคอมพิวเตอร์ โดยคำนึงถึงลักษณะของข้อมูลที่สามารถทำการส่งผ่านไปในเครือข่ายได้อย่างเหมาะสม ภายใต้ขีดจำกัดของความเร็วในการส่งผ่านข้อมูลของเครือข่ายที่ใช้ การเขียนแอพพลิเคชั่นในการควบคุมการส่งผ่านข้อมูล

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 ทฤษฎีรูปแบบของภาพ

เมื่อคอมพิวเตอร์ส่วนบุคคลได้กลายเป็นสิ่งจำเป็นในชีวิตประจำวันของเรามากขึ้น ปัจจุบันเราสามารถหาซื้อได้ง่ายขึ้นด้วยราคาที่ถูกลง ด้วยเหตุนี้จึงทำให้มีการนำเอาคอมพิวเตอร์มาประยุกต์ใช้งานในด้านสาขาอาชีพต่างๆ เพิ่มขึ้น โดยเฉพาะงานทางด้านศิลปะ ออกแบบ และงานทางด้านกราฟฟิคการเก็บภาพวีดีโอ

ปัจจุบันเกิดมาตรฐานมากมายหลายมาตรฐานที่เกี่ยวกับภาพ ที่เราได้ยินคุ้นหูที่สุดก็คงจะเป็นมาตรฐาน JPEG และ MPEG ซึ่งเป็นที่นิยมกันในปัจจุบันด้วยข้อดีที่ว่า มันสามารถนำไปเก็บในเครื่องคอมพิวเตอร์ของเราได้ด้วยการใช้พื้นที่ในการเก็บที่น้อยกว่าในอดีตที่ผ่านมา

2.1.1 มาตรฐาน JPEG

JPEG ย่อมาจาก Joint Photographic Expert Group เป็นมาตรฐานที่เกิดขึ้นในปี 1992 จุดมุ่งหมายเพื่อใช้เป็นมาตรฐานภาพนิ่งสำหรับภาพสี อัตราการบีบอัดของไฟล์ JPEG ที่สามารถทำได้ประมาณ 15:1 โดยเฉลี่ย ส่วนมากเราจะนิยมใช้ในงานที่เกี่ยวภาพสีและภาพขาวดำ (grayscale) เช่น ภาพถ่ายดาวเทียม ภาพที่ใช้ทางการแพทย์ บางครั้งเราสามารถใช้มันเป็นภาพวีดีโอได้ หรือเรียกอีกอย่างว่า Motion JPEG หรือ MJPEG เป็นการใช้ฮาร์ดแวร์ชนิดพิเศษเข้าช่วยในการบีบอัดข้อมูลภาพต่อภาพ มาตรฐานการทำงานของการบีบอัดภาพไฟล์ JPEG มีอยู่ด้วยกัน 4 โหมด ได้แก่

โหมดที่ 1 การบีบอัดไฟล์โดยอาศัยวิธีการเข้ารหัสแบบลำดับพื้นฐาน DCT (sequential DCT-based encoding) เป็นวิธีที่ใช้งานมากที่สุด โดยการทำงานจะเริ่มต้นจากซ้ายไปขวา จากบนลงล่าง เป็นวิธีที่ยอมให้เกิดการสูญเสียของข้อมูลในภาพได้ มีอัตราการบีบอัดมาก ซึ่งมีขั้นตอนสำคัญๆ ดังนี้

ขั้นตอนของการเข้ารหัส (Encoder)

1. การแปลงสัญญาณที่ไม่มีความต่อเนื่องให้อยู่ในรูปของโคไซน์หรือ FDCT (Forward Discrete Cosine Transformation)

2. การทำการจัดระดับ (Quantization) เนื่องจากตาของคนเรามีความรู้สึกที่ความต่ำมากกว่าความรู้สึกที่ความถี่สูง จึงทำการจัดระดับโดยการนำค่าความเข้มของแสงที่มีความถี่ต่ำมาไว้ตรงจุดเริ่มต้นของภาพก่อนแล้วจึงตามมาด้วยค่าความเข้มของสีที่มีความถี่สูง โดยการสูญเสียที่เกิดจากการบีบอัดข้อมูลจะเกิดขึ้นในระหว่างขั้นตอนนี้

3. การเข้ารหัสด้วยวิธีการของเอนโทรปี (Entropy Coding) เพื่อทำให้จำนวนบิตที่ทำการเข้ารหัสมีจำนวนที่น้อยลงกว่าการเข้ารหัสด้วยวิธีปกติ

ขั้นตอนการถอดรหัส

1. การถอดรหัสเอนโทรปี (Entropy Decoding)
2. การทำการจัดระดับย้อนกลับ (Dequantization)

3. การแปลงสัญญาณที่ไม่มีความต่อเนื่องให้อยู่ในรูปของโคไซน์ย้อนกลับหรือ IDCT (Inverse Discrete Cosine Transformation)

โหมดที่ 2 เป็นโหมดของการบีบอัดข้อมูลด้วยวิธีการที่ไม่ทำให้เกิดการสูญเสียเลย (Lossless encoding) ใช้งานน้อย วิธีการนี้ให้อัตราการบีบอัดที่น้อย แต่สามารถนำภาพดั้งเดิมกลับมาได้

โหมดที่ 3 เป็นโหมดของการบีบอัดแบบก้าวหน้า (Progressive encoding) ภาพที่ได้จะมีคุณภาพต่ำในตอนแรกที่ถูกแสดงและถูกปรับปรุงขึ้นอย่างต่อเนื่องเพื่อให้ได้ภาพที่สมบูรณ์ในที่สุด โหมดการบีบอัดแบบก้าวหน้าต้องใช้บัฟเฟอร์ในการสะสมข้อมูล โดยมีวิธีการอยู่ 2 วิธีเพื่อให้ได้ภาพที่ดีขึ้นอย่างต่อเนื่อง

1. วิธีการเลือกสเปคตรัม (Spectral selection) ในตอนแรกส่งส่วนประกอบของ DC หลังจากนั้นก็ส่งส่วนประกอบของ ACนิดหน่อย และตามด้วยส่วนของAC ต่อเนื่องมาจนประกอบเป็นภาพโดยสมบูรณ์

2. วิธีประมาณโดยผลสำเร็จ (Successive approximation) ทำการส่งสัมประสิทธิ์ DCT จาก MSB (บิตที่สำคัญมากที่สุด) ถึงLSB (บิตที่สำคัญน้อยที่สุด)

โหมดที่ 4 เป็นโหมดที่มีลักษณะเป็นลำดับชั้น (Hierarchical encoding) ใช้วิธีการสแกนหลายๆ รอบด้วยขนาดที่ต่างกัน

ลำดับภาพวีดีโอ

ลำดับภาพวีดีโอจะแสดงจำนวนของภาพวีดีโอหรือกลุ่มของภาพวีดีโอ ซึ่งจะเป็นเพียงกลุ่มของภาพเพียงกลุ่มหนึ่งเท่านั้นไม่ใช่ภาพทั้งหมดของวีดีโอ ภาพวีดีโอแบบดิจิตอลเรียกได้อีกอย่างว่าเฟรม

แต่ละเฟรมจะบรรจุข้อมูลของความเข้มของสีและความสว่างทั้งหมดที่จำเป็นต่อการแสดงภาพบนหน้าจอแสดงผล ข้อมูลที่เป็นสีและความสว่างนั้นถูกสร้างให้เป็น 3 เมตริกซ์ บรรจุไปด้วยความเข้มของแสงสว่างและ โดยระดับความเข้มของแสงจะเป็นส่วนแรกที่แสดงเป็น Y ตามด้วยความเข้มของสี Cb และCr ด้วยการกระจายของความเข้มของสีที่เท่าๆ กันจะทำให้เกิดการผิดพลาดที่น้อยลง โดยขนาดของเมตริกซ์เหล่านี้จะเปลี่ยนไปขึ้นอยู่กับความระเอียดของภาพ (resolution) และอัตราการแซมปลิ้งที่ใช้

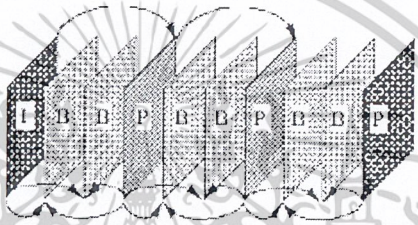
2.1.2 มาตรฐาน MPEG

เป็นมาตรฐานการบีบอัดข้อมูลที่เป็นเสียงและเป็นวีดีโอ โดยใช้เทคนิคของ JPEG ในการบีบอัดภาพแรก (I-Frame) และภาพต่อๆไป (B-Frame หรือ P-Frame) โดยจะทำการเก็บเฉพาะค่าความแตกต่างของภาพก่อนหน้านี้ สามารถบีบอัดข้อมูลภาพและเสียงพร้อมกันได้ MPEG จะประกอบไปด้วยระดับการไหลของเฟรมที่ต่างกัน 3ชนิด คือ

- เฟรมที่ถูกเข้ารหัสภายในตัว (Intra-coded Frame: I-Frame) เป็นเฟรมที่ถูกเข้ารหัสโดยไม่ต้องอาศัยเฟรมอื่นในลำดับวีดีโอมาใช้ในการเข้ารหัส ใช้เทคนิคการเข้ารหัสเดียวกันกับ JPEG จึงทำให้เป็นเฟรมแรกในลำดับวีดีโอที่ถูกใช้เป็นเฟรมอ้างอิงเสมอ เฟรมที่ถูกเข้ารหัสภายในตัวมันจำเป็นต้องใช้บิตในการแสดงภาพมากที่สุด ทำให้เป็นเฟรมที่มีอัตราการบีบอัดน้อยที่สุดใน 3 เฟรมนี้

- เฟรมที่ถูกเข้ารหัสโดยการทำนาย (Predictive coded Frame: P-Frame) เป็นเฟรมที่ถูกเข้ารหัสโดยการอาศัยข้อมูลจากเฟรมอื่นที่ถูกแสดงมาเป็นลำดับก่อนหน้านี้ ซึ่งจะถูกใช้เป็นเฟรมอ้างอิง โดยเฟรมนี้จะมาจากเฟรมที่ถูกเข้ารหัสภายในตัว หรือเฟรมที่ถูกเข้ารหัสโดยการทำนายก็ได้ เฟรมที่ถูกเข้ารหัสโดยการทำนายจะมีขนาดประมาน 50-30 เปอร์เซ็นต์ของเฟรมที่ถูกเข้ารหัสภายในตัว และให้อัตราการบีบอัดที่มากกว่าเฟรมที่ถูกเข้ารหัสภายในตัว

- เฟรมที่ถูกเข้ารหัสแบบทำนายสองทาง (Bidirectional-coded Frame: B-Frame) เป็นเฟรมที่ถูกเข้ารหัสโดยอาศัยเฟรมอื่นเช่นเดียวกับเฟรมที่ถูกเข้ารหัสโดยการทำนาย โดยใช้เฟรมที่เข้ารหัสก่อนหน้านี้กับเฟรมล่วงหน้าซึ่งอาจจะเป็นเฟรมที่เข้ารหัสภายในตัวหรือเฟรมที่เข้ารหัสแบบทำนายก็ได้ เป็นเฟรมที่เข้ารหัสโดยอาศัยการทำนายสองทิศทาง เฟรมที่เข้ารหัสแบบการทำนายสองทิศทางนี้มีขนาดประมาณ 50 เปอร์เซ็นต์ของเฟรมที่เข้ารหัสแบบทำนาย และให้อัตราการบีบอัดที่มากที่สุดในสามเฟรมนี้



รูปที่ 2.1 ลำดับภาพในแต่ละเฟรมที่ถูกเข้ารหัสแบบ MPEG

รูปแบบของรหัสเมื่อทำการเข้ารหัสที่ลำดับเฟรมต่อเนื่องกันจะมีลักษณะดังนี้ IBBPBBPBB IBBPBBPBB IBBPBBPBB ซึ่งรูปแบบนี้จะขึ้นอยู่กับตัวเข้ารหัสที่ใช้และไม่จำเป็นต้องมีกฎเกณฑ์

เราอาจจะทำการแบ่งมาตรฐานของ MPEG ออกเป็น 3 ส่วนด้วยกัน

1. ส่วนของภาพวีดีโออยู่บนพื้นฐานของ H.261 และ JPEG
2. ส่วนของเสียงอยู่บนพื้นฐานของเทคโนโลยี MUSICAM
3. ส่วนของระบบที่ใช้เป็นการควบคุมการไหลของข้อมูลอยู่บนพื้นฐานของเครือข่ายเทคนิค

การประมาณการเคลื่อนไหว

เทคนิคที่ถูกใช้ในวีดีโอ MPEG อยู่บนพื้นฐานของการประมาณการเคลื่อนไหว ซึ่งจะเกิดขึ้นที่กระบวนการของการเข้ารหัสแบบทำนายและการเข้ารหัสแบบทำนายสองทิศทาง เป็นการประมาณการเคลื่อนไหวเบื้องต้นที่มีลักษณะต่อเนื่องกัน ยกเว้นการเปลี่ยนแปลงที่เกิดจากวัตถุภายในเฟรม การเคลื่อนไหวที่เป็นศูนย์หรือไม่เกิดการเคลื่อนไหวนั้น จะเกิดขึ้นระหว่างเฟรมจะทำให้การเข้ารหัสเป็นไปได้โดยง่ายอย่างมีประสิทธิภาพโดยการอาศัยเฟรมที่ถูกการเข้ารหัสก่อนหน้านี้และเฟรมในอนาคต เมื่อมาถึงตัวถอดรหัส ข้อมูลเดียวที่จำเป็นสำหรับตัวถอดรหัสก็คือส่วนของเฟรมแรกที่จำเป็นมากต่อการนำภาพต้นกำเนิดกลับคืนมาเพื่อใช้เป็นเฟรมอ้างอิงในภาพต่อๆ ไป แต่ในเวลาที่เกิดการเคลื่อนไหวเกิดขึ้นการประมาณตำแหน่งที่เป็นไปได้จะทำได้ไม่ง่ายนัก

หนทางของการทำนายการเคลื่อนไหวที่ใช้ในมาตรฐานของ MPEG เพื่อที่จะแก้ปัญหาของการประมาณการเคลื่อนไหวนี้คือ การกระทำโดยการใช้การค้นหาที่ว่าง 2 มิติ ที่แต่ละความเข้มของแสง (luminance) ในบล้อกของมาโคร การประมาณการเคลื่อนไหวจะไม่ถูกใช้โดยตรงความเข้มของสี (chrominance) ในวีดีโอMPEG แต่จะทำโดยการสมมุติว่าการเคลื่อนไหวของความเข้มของสีมีลักษณะที่เช่นเดียวกับการเคลื่อนไหวของความเข้มของแสง การค้นหาที่ว่านี้ จะถูกกระทำหรือดำเนินการในแนวทางใดนั้น ขึ้นอยู่กับความใส่ใจในเรื่องความซับซ้อนและคุณภาพของภาพที่ต้องการ

2.2 ระบบเครือข่าย

ระบบเครือข่ายจะเกิดจากคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไปถูกเชื่อมโยงเข้าด้วยกัน ซึ่งในการเชื่อมโยงนี้ต้องอาศัยสายสัญญาณ (Cables), ตัวต่อเชื่อม (Connectors) และตัวแปลงสัญญาณเครือข่าย (network adapters) หรือการ์ดเครือข่ายที่อยู่ภายในเครื่องคอมพิวเตอร์จะเป็นอุปกรณ์ที่ช่วยในการส่งหรือรับข้อมูลจากคอมพิวเตอร์เครื่องอื่นได้

เครือข่าย Ethernet เป็นเครือข่ายที่นิยมใช้ที่สุดอีกประเภทหนึ่งในปัจจุบัน เป็นที่รู้จักกันในอีกชื่อหนึ่งว่าเครือข่าย LAN มีความเร็วในการส่งผ่านข้อมูล 10 Mbps ใช้สายโคเอคเชี่ยนเป็นสายนำสัญญาณ โดยแต่ละสถานีในเครือข่ายจะใช้ตัวกลางร่วมกัน ดังนั้นแต่ละสถานีต้องทำการตรวจดูก่อนว่าตัวกลางนั้นว่างหรือไม่ก่อนที่ทำการส่งข้อมูลมีตัวตรวจสอบการชนกันของข้อมูล

โปรโตคอล TCP/IP

TCP/IP ย่อมาจาก Transmission Control Protocol/Internet Protocol เป็นข้อตกลงร่วมกันในระบบเครือข่ายที่ใช้ในการติดต่อสื่อสารข้อมูลระหว่างเครือข่ายที่เชื่อมถึงกัน โดยมีโครงสร้างฮาร์ดแวร์หรือระบบปฏิบัติการที่แตกต่างกัน

2.2.1 การทำงานของ TCP

TCP เป็นโปรโตคอลที่มุ่งเน้นในการเชื่อมต่อ (connection – oriented protocol) ระหว่างระบบเครือข่ายเพื่อทำการแลกเปลี่ยนข้อมูล เนื่องจากระบบเครือข่ายจะใช้ตัวกลางตัวเดียวกัน (share media) ตัวอย่างเช่นการใช้สายเคเบิ้ลร่วมกัน ซึ่งจำเป็นที่จะต้องทำการแบ่งข้อมูลออกเป็นชิ้นๆ และทำให้อยู่ในรูปแบบที่สามารถส่งผ่านตัวนำในระบบเครือข่ายได้ ซึ่งเราจะเรียกข้อมูลลักษณะนี้ว่าแพ็กเก็ท (Packet) เมื่อแอพพลิเคชันต้องการที่จะทำการส่งข้อมูลๆ หนึ่งก็จะทำการแบ่งข้อมูลนั้นออกเป็นแพ็กเก็ทที่มีขนาดที่เหมาะสมตามระบบเครือข่าย แล้วจึงทำการส่งต่อไป

เนื่องจากได้มีการแบ่งข้อมูลที่เราจะทำการส่งออกเป็นแพ็กเก็ท ดังนั้น TCP จะต้องทำการทำเครื่องหมาย (Sequence Numbers) เพื่อระบุถึงระดับของข้อมูลที่จะทำการส่ง เพื่อที่จะสามารถเรียงข้อมูลกลับได้ในตอนรับ และ TCP ยังมีการคำนวณ checksum เพื่อใช้ในการตรวจสอบหาความผิดพลาดระหว่างการส่ง สุดท้าย TCP จะใช้ port id ในการระบุว่าจะให้แอพพลิเคชันใดที่อยู่ในเครือข่ายนั้นทำการรับส่ง

ข้อมูลนั้นๆ ซึ่งทั้ง Sequence Number, Checksum และ Port id จะถูกแทรกลงใน TCP แพ็กเก็ท ในส่วนที่เรียกว่าเฮดเดอร์ (Header) ซึ่งจะถูกแทรกอยู่ในส่วนแรกของแพ็กเก็ทที่ต้องการจะส่ง

อินเตอร์เน็ตแอดเดรส (Internet Address)

ทุกอินเตอร์เฟสที่ต่ออยู่บนเครือข่ายจะต้องมีหมายเลขประจำตัวเพื่อใช้ในการติดต่อสื่อสารข้อมูล หมายเลขนี้เราเรียกว่าอินเตอร์เน็ตแอดเดรส (Internet Address) หรือเรียกย่อๆว่า IP Address โดยค่า IP Address นี้จะเป็นเลขจำนวน 32 บิต ที่ไม่ถูกนับต่อเนื่องกันไป แต่จะใช้วิธีแบ่งหมายเลขดังกล่าวออกเป็นกลุ่มขนาด 8 บิตจำนวน 4 ชุด และคั่นแต่ละชุดด้วยจุด ดังตัวอย่างเช่น 192.168.13.204

นอกจากนี้ใน IP Address นั้นยังถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นที่อยู่ของเน็ตเวิร์ค (Network ID) และส่วนที่เป็นที่อยู่ของโฮส (Host ID) ซึ่งข้อมูลในส่วนนี้จะถูกใช้สำหรับการค้นหาเส้นทางของอุปกรณ์ปลายทาง เพื่อที่จะขนส่งข้อมูลจากต้นทางไปถึงปลายทางอย่างถูกต้อง

หมายเลข IP Address ยังทำการแบ่งแต่ละช่วงออกเป็นคลาส (class) ต่างๆกัน เพื่อเป็นกำหนดขนาดของเครือข่ายที่ใช้จากคลาส A ถึง E เพื่อจะได้ทำการกำหนด IP Address ได้อย่างเหมาะสมกับขนาดของเครือข่าย

ตารางที่ 2.1 การแบ่งคลาสของ IP Address ในแต่ละคลาส

Class	Range
A	0.0.0.0 – 127.255.255.255
B	128.0.0.0 – 191.255.255.255
C	192.0.0.0 – 223.255.255.255
D	224.0.0.0.0 – 239.255.255.255
E	240.0.0.0 – 255.255.255.255

2.2.2 การทำงานของ IP (Internet Protocol)

IP เป็นทำหน้าที่ในการนำข้อมูลไปส่งยังปลายทาง โปรโตคอลต่างๆใน TCP/IP Suite ต่างก็อาศัยระบบนี้ เนื่องจากตัว IP มีกลไกในการหาเส้นทางการขนส่งข้อมูล สามารถที่จะหาช่องทางทุกทางที่สามารถจะเป็นไปได้ มีความเชี่ยวชาญในการขนส่งข้อมูลได้ในระยะทางไกลๆ จุดด้อยของ IP คือ ความไม่น่าเชื่อถือ (unreliable) และ การไม่มีการเชื่อมต่อ (connectionless) ดังนั้นเพื่อทำการลดข้อด้อยนี้ จึงต้องใช้ร่วมกับ TCP ที่มีกลไกในรับประกันการรับ-ส่งข้อมูล เพื่อทำให้ข้อมูลที่รับ-ส่งกันมีความน่าเชื่อถือได้

ความไม่น่าเชื่อถือ (Unreliable) ตัว IP เองมีกลไกในการหาเส้นทางของข้อมูล (routing) ไปยังปลายทาง แล้วจึงทำการส่งข้อมูลไปยังปลายทาง แต่ไม่มีการตรวจสอบว่าข้อมูลถึงปลายทางหรือไม่ มีแต่การแจ้งกลับเมื่อเกิดปัญหาขึ้นระหว่างการส่งข้อมูล ดังนั้นการส่งด้วย IP อาจเป็นได้ทั้งข้อมูลถึงปลายทางแล้วและข้อมูลไม่ถึงแต่ไม่สามารถส่งข้อความผิดพลาดกลับมาได้

ไม่มีการเชื่อมต่อ (Connectionless) หมายถึง IP จะไม่มีสถานะเหมือนการเชื่อมต่อระหว่างต้นทางกับปลายทาง การเชื่อมต่อที่ว่าคือการเชื่อมต่อทางลอจิคัล ไม่ใช่สายจริงๆ (Physical) การที่บอกว่า IP มีการเชื่อมต่อแบบ Connectionless

การหาเส้นทางของIP (IP Routing) หมายถึง IP มีกระบวนการค้นหาเส้นทางในการส่งผ่านข้อมูลจากต้นทางไปยังปลายทางผ่านการส่งต่อของอุปกรณ์ IP ที่อยู่ในเครือข่ายที่โดยช่วยกันทำการส่งต่อข้อมูลไปจนถึงปลายทาง

อุปกรณ์ที่ใช้ในเครือข่าย

โฮสเป็นอุปกรณ์ที่ทำหน้าที่กำเนิดข้อมูลในกรณีเป็นผู้ส่ง หรือทำหน้าที่รับข้อมูลในกรณีเป็นผู้รับ การส่งข้อมูลใดๆ จะต้องเป็นการส่งข้อมูลจากโฮสไปยังโฮสเสมอ สำหรับที่อยู่ของ IP แพ็คเก็ทแล้วจะเป็นข้อมูลที่ปรากฏอยู่ในส่วนที่อยู่ต้นทาง (Source Address) และ ที่อยู่ปลายทาง (Destination Address) หรืออีกความหมายหนึ่งก็คือ IP แอดเดรสเป็นหมายเลขที่ระบุตำแหน่งของโฮสต้นทางและโฮสปลายทางเท่านั้น

เราเตอร์ทำหน้าที่ส่งผ่านข้อมูลจากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่ง ตำแหน่งของเราเตอร์จะอยู่ระหว่างจุดเชื่อมต่อของเครือข่ายทั้งสอง ด้วยข้อกำหนดของ IP ข้อมูลจะส่งโดยตรงข้ามเครือข่ายไม่ได้ ต้องอาศัยเราเตอร์เป็นผู้ทำหน้าที่ส่งผ่านข้อมูลไปให้

เครือข่ายในที่นี้คือเครือข่ายที่เป็น IP เท่านั้นไม่รวมเครือข่ายประเภทอื่น โดย IP จะระบุหมายเลขประจำโฮสโดยใช้ IP แอดเดรสเพื่อระบุตำแหน่งของโฮสต้นทางและปลายทางและระบุตำแหน่งของเครือข่ายที่โฮสนั้นใช้เชื่อมต่อ โดยโปรโตคอล IP มีกระบวนการที่จะแยกหมายเลขประจำตัวของโฮสและของเครือข่ายออกจากกัน เพื่ออุปกรณ์เหล่านั้นจะสามารถพิจารณาได้ว่าควรส่งข้อมูลที่รับมานั้นไปในทิศทางใด

2.3 การใช้งาน Visual Basic

Visual Basic เป็นเครื่องมือที่ใช้ในการพัฒนาโปรแกรมบนวินโดวส์ เป็นภาษาที่ใช้ในการเขียนแอปพลิเคชั่นที่มีประสิทธิภาพ สามารถเรียนรู้ได้ง่าย มีขั้นตอนการทำงานที่ไม่สลับซับซ้อนทั้งในส่วนของการออกแบบที่ติดต่อกับผู้ใช้ และในส่วนของโปรแกรมมิ่ง ทั้งยังเป็นภาษาที่ได้รับความนิยมเป็นอย่างมากในหมู่นักพัฒนาโปรแกรมในปัจจุบัน

จากที่ได้ทำการศึกษาโครงสร้างของการทำงานของเครื่องมือที่ใช้ในการพัฒนาโปรแกรมบนระบบการส่งข้อมูลผ่านเครือข่าย จึงได้เลือก Visual Basic เป็นเครื่องมือในการพัฒนาดังกล่าว เนื่องจากเหตุผลดังต่อไปนี้

1. สร้างแอปพลิเคชั่นได้ง่ายและรวดเร็ว
2. มีทูลต่างๆที่ช่วยในการควบคุมการทำงานร่วมกับอุปกรณ์อื่นๆ มากมาย เช่น การ์ดเสียง วีดีโอการ์ด

3. ยอมให้ผู้พัฒนาแอปพลิเคชั่นสามารถเรียกใช้ความสามารถของระบบปฏิบัติการได้อย่างเต็มที่ โดยผ่านทาง Window API
4. มีเครื่องมือในการควบคุมการทำงานร่วมกับแอปพลิเคชั่นอื่นๆ เช่น OLE control, Data control เป็นต้น
5. สามารถติดต่อกับระบบฐานข้อมูลที่มีอยู่ในปัจจุบันได้ เช่น FoxPro หรือ Microsoft Access

เข้าสู่ Visual Basic

เมื่อเราเปิด Visual Basic ขึ้นมา เราก็จะพบกับไดอะล็อกบ็อกซ์โปรเจคใหม่ (New Project) ที่แสดงประเภทของแอปพลิเคชั่นที่เราสร้างได้ แสดงดังตารางที่ 2.2

ตารางที่ 2.2 ชนิดของแอพพลิเคชันใน Visual Basic

ชนิดของแอปพลิเคชั่น	คำอธิบาย
Standard EXE	เป็นแอพพลิเคชั่นทั่วไปที่มีการใช้งานในวินโดว์ เมื่อสร้างแล้วจะได้ไฟล์ที่มีนามสกุลเป็น .EXE
Active EXE	เป็นการสร้างแอพพลิเคชั่นที่เรียกว่า out - off - process OLE server
ActiveX DLL	เป็นการสร้างแอพพลิเคชั่นที่เรียกว่า in - process OLE server
ActiveX EXE	เป็นการสร้าง ActiveX control เมื่อสร้างแล้วจะได้ไฟล์นามสกุลเป็น .OCX
VB Application Wizard	เป็นการสร้างวิซาร์ดเพื่อใช้งานร่วมกับแอปพลิเคชั่น ซึ่งมักเป็นแอพพลิเคชั่นที่มีความซับซ้อน จึงต้องมีวิซาร์ดมาช่วยลดความยุ่งยากในการใช้งาน
VB Wizard Manager	เป็นเครื่องมือที่ใช้สร้างวิซาร์ด
Data Project	เป็นการสร้างแอพพลิเคชั่นเพื่อให้ทำงานร่วมกับ Data object
DHTML Application	เป็นการสร้างแอพพลิเคชั่นอินเตอร์เน็ตในฝั่งไคลเอนท์ โดยใช้ความสามารถของ Dynamic HTML ซึ่งจะรันบนบราวเซอร์ Internet Explorer
IIS Application	เป็นการสร้างแอพพลิเคชั่นอินเตอร์เน็ตในฝั่งเซิร์ฟเวอร์ โดยใช้ความสามารถของอินเตอร์เน็ต เซิร์ฟเวอร์อย่าง IIS เช่น Active Server Pages
Addin	เป็นเครื่องมือช่วยในการสร้าง Add - in ซึ่ง Visual Basic เปิดโอกาสให้เราสร้างเครื่องมือส่วนตัวขึ้นมาเฉพาะกับงานของเราได้
ActiveX Document DLL	เป็นการสร้างแอพพลิเคชั่นที่เรียกว่า in - process ActiveX Document (ซึ่งเป็นวิธีการที่ไม่นิยมแล้วในปัจจุบัน)
ActiveX Document EXE	เป็นการสร้างแอพพลิเคชั่นที่เรียกว่า out -of - process ActiveX Document (ซึ่งเป็นวิธีการที่ไม่นิยมแล้วในปัจจุบัน)

ในการเขียนโปรแกรมเพื่อสร้างแอปพลิเคชั่นจาก Visual Basic นั้น จะมีวิธีการเขียนโปรแกรมที่แตกต่างจากการเขียนโปรแกรมแบบโครงสร้างทั่วไปที่เราอาจจะเคยได้เรียนรู้มาก่อน เช่น ภาษาพาสคาล (Pascal), ภาษา C นั่นเพราะการเขียนโปรแกรมกับ Visual Basic นั้น จะใช้การเขียนแบบ Event-Driven (อาจแปลได้ว่าเหตุการณ์พาไปก็ได้)

Event-Driven จริงๆก็คือ การเขียนโปรแกรมในลักษณะว่า "ถ้าเหตุการณ์นี้เกิดขึ้น เราควรจะดำเนินการอย่างไร" เมื่อคิดออกเราก็เขียนโปรแกรมมารองรับกับเหตุการณ์ต่างๆ เหล่านั้น ซึ่งโปรแกรม Visual Basic จะมีเครื่องมือที่ช่วยให้เราจัดการกับเหตุการณ์ต่างๆที่น่าจะเกิดกับแอปพลิเคชั่นของเราได้อย่างสะดวก

วิธีสร้างแอปพลิเคชั่นด้วย Visual Basic

ขั้นที่ 1: การออกแบบแอปพลิเคชั่น

ก่อนสร้างแอปพลิเคชั่น หรือเขียนโปรแกรมนั้นสิ่งแรกที่ควรต้องทำก็คือ ต้องทราบให้แน่ชัดก่อนว่าแอปพลิเคชั่นที่เราจะสร้างนั้นจะใช้ประโยชน์อะไร แล้วจึงทำการเขียนโฟลว์ชาร์ทเพื่อช่วยให้เราเรียบเรียงความคิดอย่างเป็นระเบียบ ทำให้มองเห็นการทำงานได้อย่างชัดเจนยิ่งขึ้น

ขั้นที่ 2: ตกแต่งหน้าตาแอปพลิเคชั่น

เป็นการตกแต่งหน้าตาแอปพลิเคชั่นที่ได้ออกแบบไว้ พร้อมกับกำหนดค่าพร็อพเพอร์ตี้ต่างๆ ให้กับคอนโทรลแต่ละตัวในแอปพลิเคชั่น

ขั้นที่ 3: เขียนโค้ดกำกับการทำงานของแอปพลิเคชั่น

เป็นการเขียนโปรแกรมเพื่อควบคุมการทำงานต่างๆเพื่อรองรับกับเหตุการณ์ที่จะเกิดขึ้นกับคอนโทรลต่างๆในแอปพลิเคชั่นของเรา

ขั้นที่ 4: ทดสอบการทำงานของแอปพลิเคชั่น

หลังจากเขียนโค้ดเสร็จ ก็ต้องทดสอบแอปพลิเคชั่นที่เราสร้างขึ้น ซึ่งประกอบด้วยคอนโทรลต่างๆที่เราปรับแต่งไว้ เพื่อดูว่าโปรแกรมที่เราสร้างขึ้นสามารถทำงานได้อย่างที่คาดหมายเอาไว้รึเปล่า เพื่อทำการแก้ไขต่อไป

ขั้นที่ 5: บันทึกเก็บไว้ในคอมพิวเตอร์

หลังจากแน่ใจว่าโปรแกรมของเราทำงานได้ถูกต้องตามวัตถุประสงค์ที่ได้ออกแบบไว้ จึงทำการบันทึกเก็บไว้ซึ่งสามารถเพิ่มเติมปรับเปลี่ยนได้ในภายหลัง

ขั้นที่ 6: การสร้างไฟล์ .EXE

เมื่อเราทำการสร้างแอปพลิเคชั่นเสร็จแล้ว เราอาจต้องการนำแอปพลิเคชั่นที่สร้างขึ้นเรียกใช้งานได้เองโดยไม่ต้องเรียกผ่าน Visual Basic หรือต้องการนำไปใช้งานกับคอมพิวเตอร์เครื่องอื่นๆ เราก็สามารถทำได้โดยสร้างไฟล์เอ็กซ์คิวต์

การคอมไพล์แบบ Native (Native Code Compiling)

จากที่ในอดีต Visual Basic ใช้ตัวแปลภาษาที่เรียกว่า อินเตอร์พรีตเตอร์ (Interpreter) คือต้องแปลคำสั่งใน Visual Basic ทุกครั้งที่จะต้องรันโปรแกรม และจะต้องทำการแปลบรรทัดต่อบรรทัด ทำให้เกิดการทำงานที่ช้ากว่าตัวแปลภาษาที่เรียกว่าคอมไพเลอร์ (Compiler) ซึ่งจะแปลภาษาที่เราเขียนเป็นภาษาระดับต่ำที่พร้อมใช้งานได้ตลอดเวลา ข้อบกพร่องตรงนี้ทำให้ Visual Basic รุ่นใหม่นั้นถูกแก้ไข โดยเราสามารถเลือกที่จะให้แปลภาษาได้ทั้ง 2 แบบ คือ ทั้งแบบอินเตอร์พรีตเตอร์ และแบบคอมไพเลอร์

2.3.1 ส่วนประกอบต่างๆ ของ Visual Basic

มีวินโดวส์เกิดขึ้น 5 วินโดวส์ คือ

1. วินโดว์หลักของ Visual Basic ซึ่งใช้ในการสั่งงานต่างๆ เช่น เกี่ยวกับไฟล์ คอมไพล์ การรันแอปพลิเคชั่นหรือควบคุมวินโดวส์อื่นๆ ซึ่งอาจจะสั่งงานผ่าน Toolbar ก็ได้
2. วินโดว์ Toolbox จะประกอบไปด้วยออปเจ็คต่างๆ ที่เรานำมาสร้างเป็นแอปพลิเคชั่น
3. วินโดว์โปรเจค(Project) จะเป็นตัวควบคุมไฟล์ต่างๆ ที่จะประกอบกันเป็นแอปพลิเคชั่น จะมีส่วนขยายเป็น .mak ซึ่งจะประกอบไปด้วย Forms(frm), Module(.bas) และCustom Control (.vbx)
   - Form เป็นส่วนที่ติดต่อกับผู้ใช้ ประกอบไปด้วยโปรเจ็คต่างๆ
   - Modules เป็นส่วนที่กำหนดตัวแปรและ โพรซีเจอร์ต่างๆ
   - Custom Controls เป็นส่วนของออปเจ็คต่างๆที่อยู่ในทูลบ็อก
4. วินโดว์ Form เป็นส่วนที่นำเอาออปเจ็คมาวางไว้ ซึ่งในครั้งแรกจะมีชื่อว่า form1 และสามารถที่จะปรับเปลี่ยนตำแหน่งและขนาดของฟอร์มได้
5. วินโดว์ Properties เป็นส่วนที่ใช้กำหนดคุณสมบัติต่างๆ ของออปเจ็คที่ปรากฏอยู่บนฟอร์มซึ่งเราสามารถเลือกได้ โดยการกดปุ่ม F4 หรือเลือกจากวินโดว์บนเมนูก็ได้

เครื่องมือใน ActiveX Control

เป็นองค์ประกอบทางซอฟต์แวร์ที่สามารถเห็นเป็นรูปธรรม ทำให้การเขียนโปรแกรมเพื่อสร้างแอพพลิเคชั่นจาก Visual Basic นั้นเป็นเรื่องง่าย โดยการนำเอา ActiveX มาวางบนฟอร์ม และเขียนโค้ดกำกับการทำงานเอาไว้ ActiveX Control มิได้มีเพียงที่เรามองเห็นใน Toolbox เท่านั้น แต่ยังมี ActiveX Control อื่นๆ อีกมากมายที่เราสามารถเปิดดูและเพิ่มเติมเข้าไปใน Toolbox ได้จากการเลือก Project >Component... พื้นฐานของ ActiveX Control ที่เราพบเห็นอยู่เสมอๆ ในการใช้แอปพลิเคชั่นทั่วไป มีดังนี้

Lable: แถบอักษร

Lable เป็นแถบตัวอักษร หรือป้ายตัวอักษรที่เรากำหนดข้อความลงไปได้ในขณะที่ใช้งานผู้จะแก้ข้อความนี้ไม่ได้ ซึ่งเรามักใช้คู่กับ ActiveX Control ตัวอื่นๆ เพื่อให้ผู้ใช้งานอ่านข้อความข้างในเพียงอย่างเดียว

CommandButton: ปุ่มกด

CommandButton หรือเรียกสั้นๆ ว่า Button เป็นปุ่มที่ให้เรากด <Enter> หรือคลิกเมาส์ เพื่อเลือกตัวเลือกนั้น ซึ่งเราใช้ CommandButton แทนหนึ่งคำสั่ง

TextBox: กรอบข้อความ

TextBox จะยอมให้เราเพิ่มเติมหรือแก้ไขข้อความที่อยู่ใน ActiveX Control ได้

OptionButton: ตัวเลือก

เป็นตัวเลือกที่เลือกได้เพียงตัวเดียวเท่านั้น ซึ่งบางคนอาจเรียกว่า Radio Button เพราะเหมือนกับเราฟังวิทยุจากสถานีวิทยุ ที่เลือกฟังได้เพียงทีละสถานี

CheckBox: ตัวเลือก

เป็นตัวเลือกที่สามารถเลือกได้มากกว่า 1 ตัว โดยการคลิกเลือกตัวเลือกที่ต้องการ

Frame: กรอบ

Frame ทำหน้าที่แยกกลุ่มของ ActiveX Control ออกเป็นกลุ่มๆ แต่ยังอยู่ในฟอร์มเดียวกัน การที่เฟรมสามารถบรรจุเอา ActiveX Control ต่างๆอยู่ภายในได้ เราเรียกความสามารถนี้ว่า คอนเทนเนอร์ (Container) ซึ่งฟอร์มเองก็มีความสามารถนี้อยู่แล้ว

ListBox: รายการข้อมูล

เป็นรายการข้อมูลที่ให้เราเลือก จากตัวเลือกที่อยู่ภายใน

ComboBox: รายการข้อมูลชนิดพิเศษ

เป็นรายการข้อมูลชนิดพิเศษที่รวมเอาความสามารถของ TextBox และ ComboBox ไว้ด้วยกัน คือ นอกจากจะคลิดเลือกรายการที่มีแล้ว ยังสามารถเลือกโดยการพิมพ์ชื่อข้อมูลที่ต้องการก็ได้เช่นกัน

ScrollBar: แถบเลื่อน

แถบเลื่อนที่เราใช้กันค่อนข้างบ่อยในการใช้งานมีอยู่ 2 ประเภทคือ แถบเลื่อนแนวตั้ง (HscrollBar) และแถบเลื่อนแนวนอน (VscrollBar) ซึ่งมีการทำงานคล้ายกัน

2.4 การควบคุมเครือข่ายโดย Winsock control

Winsock control เป็นเครื่องมือที่ช่วยให้เราสามารถเขียนโปรแกรมติดต่อสื่อสารข้อมูลผ่านระบบเครือข่ายทั้งโปรโตคอล TCP (transmission Control Protocol) และ โปรโตคอล UDP (User Datagram Protocol) โดยทั้งสองโปรโตคอลสามารถที่จะใช้สร้างเป็นแอพพลิเคชั่นเซิร์ฟเวอร์ (Server) และไคลเอน (Client) ได้

การเลือกใช้โปรโตคอล

การเลือกใช้โปรโตคอลมีลักษณะแตกต่างกันดังนี้

- โปรโตคอล TCP เป็นโปรโตคอลที่มีฐานในการเชื่อมต่อที่แน่นอน มีลักษณะคล้ายกับการใช้งานของโทรศัพท์ ผู้ใช้จะกระทำการสร้างการเชื่อมต่อขึ้นก่อนๆ ที่จะกระทำการใดๆ ต่อไป

- โปรโตคอล UDP เป็นโปรโตคอลที่ไม่ต้องการการเชื่อมต่อที่แน่นอน การปฏิบัติต่อกันระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องจะเป็นลักษณะการส่งโน้ต ข้อความจะถูกส่งจากคอมพิวเตอร์เครื่องหนึ่งไปยังคอมพิวเตอร์อีกเครื่องหนึ่ง โดยมีลักษณะการเชื่อมต่อระหว่างกันที่ไม่แน่นอน ขนาดของข้อมูลที่ใหญ่ที่สุดที่จะสามารถส่งไปได้จะถูกกำหนดโดยเครือข่าย

โดยธรรมดาแล้วโปรโตคอลที่ถูกเลือกจะถูกกำหนดจากตัวโปรแกรมที่เราสร้างขึ้น โดยเราจะมีข้อกำหนดบางอย่างที่ช่วยในการเลือกใช้งานโปรโตคอลให้เหมาะสม

1. โปรแกรมที่ต้องการบอกให้ฝั่งเซิร์ฟเวอร์ และฝั่งไคลเอนรู้ว่ามีการรับส่งข่าวสารนั้นแล้วจริง ถ้าเป็นเช่นนั้น โปรโตคอล TCP ก็เหมาะสมเพราะมีการสร้างการเชื่อมต่อที่แน่ชัดก่อนการส่งหรือรับข้อมูล
2. ข้อมูลมีขนาดใหญ่มาก (เช่น ไฟล์รูปภาพ หรือ ไฟล์เสียง) หรือไม่? การกระทำการเชื่อมต่อที่เกิดขึ้นครั้งหนึ่ง โปรโตคอล TCP จะยังคงรักษาการเชื่อมต่อนั้น และรับประกันความครบถ้วนของข้อมูล อย่างไรก็ตาม โปรโตคอล TCP มีการคำนวณเส้นทางที่มากกว่าโปรโตคอล UDP ทำให้ข้อมูลที่ใช้โปรโตคอล TCP ในการส่งมีมูลค่าที่มากกว่า
3. ถ้าข้อมูลถูกส่งเป็นพักๆ หรือเป็นช่วงระยะเวลาหนึ่ง และการส่งข้อมูลเป็นการส่งที่มีลักษณะการส่งแบบจำนวนเล็กๆ โปรโตคอล UDP ก็จะเหมาะสมกว่าโปรโตคอล TCP

การตั้งค่าโปรโตคอล

เราสามารถตั้งค่าโปรโตคอลโดยผ่านทางหน้าต่างพร็อพเพอตี้ (Property) โดยคลิกที่ Protocol แล้วเลือกระหว่าง sckTCPProtocol หรือ sckUDPProtocol เราสามารถที่จะตั้งค่าโปรโตคอลได้อีกทางหนึ่งภายในโค้ดของเรา ดังตัวอย่าง

```
Winsock1.Protocol = sckTCPProtocol
```

หรือ `Winsock1.Protocol = sckUDPProtocol`

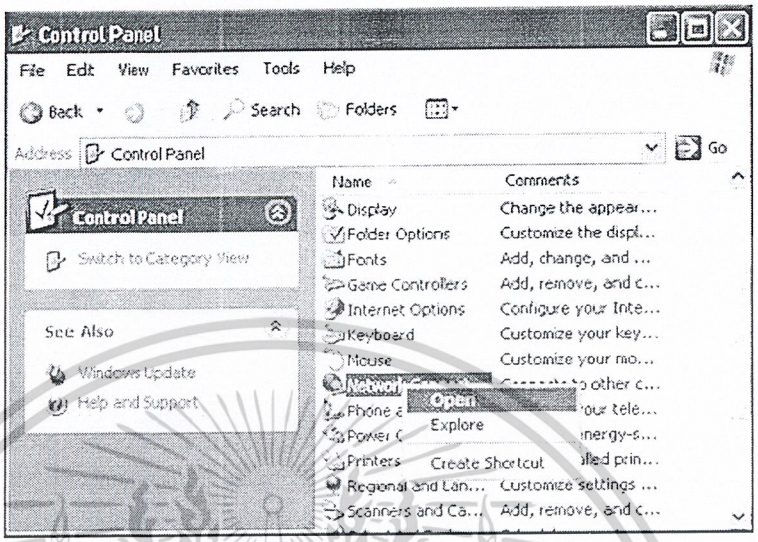
การหาชื่อของเครื่องคอมพิวเตอร์

การติดต่อภายในเครือข่ายคอมพิวเตอร์ เราจะต้องรู้ IP Address หรือ “Friendly name” ของคอมพิวเตอร์เครื่องที่เราต้องการทำการติดต่อ โดยหมายเลข IP Address นี้ จะเป็นหมายเลขดิจิตอลสามตัวที่ต่อเนื่องกันคั่นด้วย “.” (xxx.xxx.xxx.xxx)

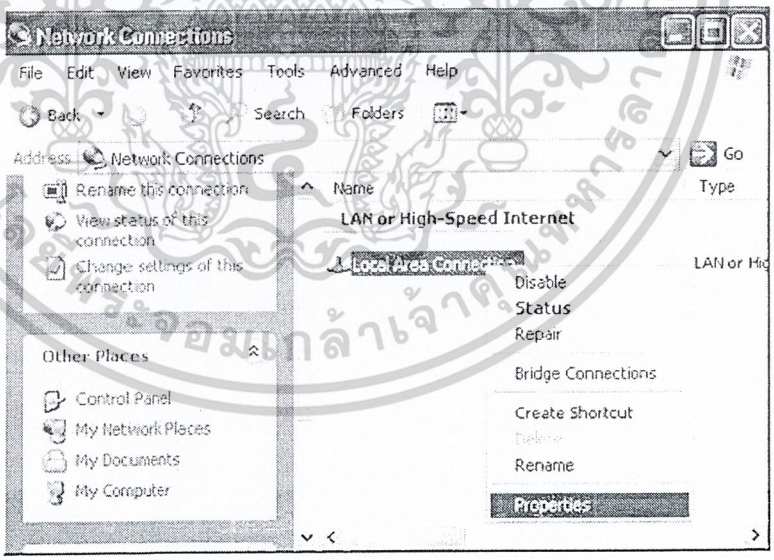
ชื่อของ IP address สามารถหาได้จาก

1. คลิก Start บน taskbar ของคอมพิวเตอร์
2. คลิก Setting จากนั้น ทำการเลือกที่ Control Panel

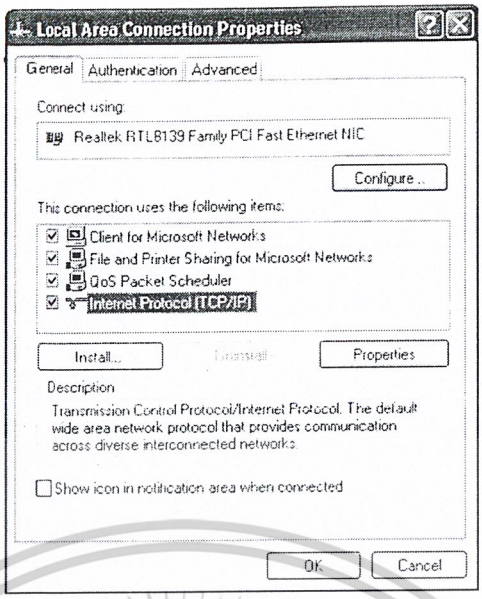
3. เลือกตรง Network Connection
4. คลิก Local Area Network เลือก Properties



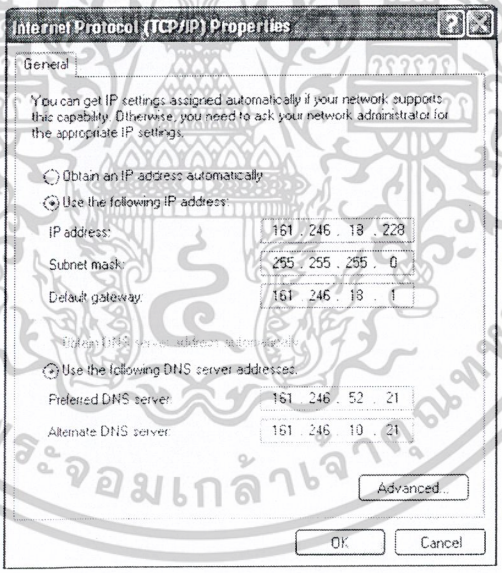
รูปที่ 2.2 การเลือก Network Connection จากControl Panel



รูปที่ 2.3 การเลือก Properties จาก Local Area Network



รูปที่ 2.4 การเลือก Internet Protocol (TCP/IP)
จาก Local Area Network Properties



รูปที่ 2.5 หมายเลขของ IP Address ของเครื่องของเรา

5. ทำการคลิกที่ เลือก Internet Protocol (TCP/IP)

6. เราจะเห็นชื่อ IP Address ของเราได้ที่ช่อง IP Address เมื่อเราพบชื่อ IP Address เรา สามารถใช้เป็นค่าสำหรับ RemoteHost property

2.4.1 ขั้นตอนของการใช้งาน Winsock Control

การสร้างการเชื่อมต่อ

ซ้อคเกต (Socket) การเชื่อมต่อสามารถถูกสร้างขึ้นจากอุปกรณ์เป้าหมายในเครือข่ายโดยใช้การควบคุมจาก Winsock การควบคุมโดยWinsockจะสามารถถูกใช้ในการติดต่อสื่อสารทั้งโปรโตคอลTCP หรือUDP โดยค่าเริ่มต้น (default) จะถูกตั้งค่าไปที่โปรโตคอล TCP หมายเลข IP Address และพอร์ทของอุปกรณ์เป้าหมายจะถูกเก็บในพร็อพเพอตี้ของ "RemoteHost" และ "RemotePort" ซึ่งค่าพร็อพเพอตี้เหล่านี้สามารถจะตั้งค่าทั้งในขั้นตอนของการออกแบบและการรันโปรแกรม ดังตัวอย่าง

```
Winsock1.Protocol = sckTCPProtocol
Winsock1.RemoteHost = "192.168.168.50"
Winsock1.RemotePort = 1070
Winsock1.Connect
```

เหตุการณ์ "Connect" เป็นไปอย่างรวดเร็วเมื่อการเชื่อมต่อถูกสร้างขึ้น เหตุการณ์ยังสามารถใช้ประโยชน์เพื่อบอกว่าการเชื่อมต่อนั้นพร้อมที่จะใช้แล้ว ดังตัวอย่าง

```
Private Winsock1_Connect ()
    Msgbox "Connection established"
End Sub
```

การส่งข้อมูล

เมื่อการเชื่อมต่อถูกสร้างขึ้นครั้งหนึ่ง ข้อมูลจะสามารถส่งไปยังอุปกรณ์เป้าหมายโดยใช้เมธอด "SendData" อีเวน "SendComplete" และ "SendProgress" จะถูกใช้เพื่อแสดงสถานะของการส่งข้อมูล

```
Winsock1.SendData "Hello World" vbCr
```

การรับข้อมูล

โดยทั่วไปอีเวน "DataArrival" ถูกใช้เมื่อข้อมูลถูกส่งโดยอุปกรณ์เป้าหมาย ภายในอีเวนจะมีเมธอด "GetData" เพื่อใช้ในการจับข้อมูลที่รับเข้ามา โดยทั่วไปข้อมูลที่รับเข้าจะถูกเก็บไว้ในบัฟเฟอร์ชั่วคราว

```
Private Sub Winsock1_DataArrival (ByVal Bytes Total As Long)
Dim buf As String
    Winsock1.GetData buf
    ProcessingRoutine buf
End Sub
```

การปิดการเชื่อมต่อ

ซ็อคเก็ตของการเชื่อมต่อที่ถูกสร้างไว้นั้น โดยทั่วไปจะยังคงเปิดไว้นานเท่านานตราบเท่าที่มันยังต้องการการติดต่อสื่อสาร กับอุปกรณ์เป้าหมาย ซ็อคเก็ตของการเชื่อมต่อที่ถูกสร้างไว้จะปิดลงเมื่อการติดต่อสื่อสาร ไม่เป็นที่ต้องการแล้ว แอพพลิเคชั่นสามารถใช้เมธอด "Close" เพื่อทำการปิดการเชื่อมต่อ ถ้าต้องการทำการเชื่อมต่อใหม่ ควรจะต้องทำการปิดการเชื่อมต่อก่อนแล้วจึงทำการสร้างการเชื่อมต่อใหม่ดังที่กล่าวมาแล้ว

```
Winsock1.Close
```

การแสดงสถานะของการเชื่อมต่อ

การควบคุมโดยใช้ Winsock Control จะประกอบด้วยพร็อพเพอตี้ "State" ซึ่งสามารถอ่านได้ทุกเวลาเพื่อจะบอกถึงสถานะในขณะนั้น อีเวน "Error" จะเกิดขึ้นเมื่อพบความผิดพลาดของการเชื่อมต่อบ่อยครั้งที่พร็อพเพอตี้ "State" จะถูกใช้ร่วมกับอีเวน "Error" เพื่อแสดงถึงสถานะที่เป็นอยู่

```
Private Sub Winsock1_Error (ByVal Number As Integer, Description As String...)
Dim ms As String
    ms = "Error=" &Number& "" &Description& vbCrLf
    ms = ms & Winsock1.State
    Msgbox ms,, "Connection Error"
End Sub
```

พื้นฐานการเชื่อมต่อโปรโตคอล TCP

เมื่อเราทำการสร้างโปรแกรมที่ใช้โปรโตคอล TCP เราจะต้องทำการระบุก่อนเลยว่าแอพพลิเคชั่นของเราจะเป็นเซิร์ฟเวอร์หรือไคลเอน ถ้าเป็นเซิร์ฟเวอร์นั้นหมายความว่าโปรแกรมของเราจะต้องรอการติดต่อ "listen" จากพอร์ทที่ระบุลงไป ถ้าเป็นไคลเอนเราจะต้องทำการร้องขอการติดต่อจากเซิร์ฟเวอร์ เซิร์ฟเวอร์สามารถที่จะยอมรับการร้องขอนั้นและการเชื่อมต่อก็จะเกิดขึ้นโดยสมบูรณ์ จากนั้นเซิร์ฟเวอร์และไคลเอนจึงสามารถติดต่อกันอย่างอิสระระหว่างกันได้

ขั้นตอนการสร้าง TCP Server

1. สร้างโปรเจ็คขึ้นใหม่
2. เปลี่ยนชื่อจากค่าดีฟอร์ท (มีชื่อว่า from) เป็น frmServer
3. เปลี่ยนแคปชั่น (Caption) ของฟอร์มเป็น "TCP Server"
4. เรียกใช้ Winsock และเปลี่ยนชื่อของมันเป็น tcpserver
5. เพิ่มโค้ดเข้าไปยังฟอร์ม

```
Private Sub Form_Load ()
    'ตั้งค่าพร็อพเพอตี้โลคัลพอร์ทโดยตัวเลข
    'จากนั้นรอการติดต่อ
    tcpServer.LocalPort = 1001
    tcpServer.Listen
    frmClient.Show
End Sub
Private Sub tcpServer_ConnectionRequest (ByVal requestID As Long)
    'ตรวจเช็คสถานะการควบคุมว่าปิดอยู่หรือไม่ ถ้าไม่
    'ทำการปิดการเชื่อมต่อก่อนรับการเชื่อมต่อใหม่
    If tcpServer.State <> sckClosed Then
    tcpServer.Close
    'ยอมรับการร้องขอด้วยพารามิเตอร์ requestID
    tcpServer.Accept requestID
End Sub

Private Sub txtSendData_Change()
    'ชื่อของ TextBox ชื่อ txtSendData จะบรรจุด้วยข้อมูลที่จะส่ง
    'ผู้ใช้สามารถใส่ค่าต่างๆลงใน TextBox
    'ในกรณีนี้ทำการส่งค่าที่เป็นตัวอักษรใช้เมธอด SendData
    tcpServer.SendData txtSendData.Text
End Sub

Private Sub tcpServer_DataArrival (ByVal Bytes Total As Long)
'ประกาศตัวแปรสำหรับข้อมูลที่เข้ามา
'ร้องขอเมธอด GetData และตั้งค่าพร็อพเพอตี้ของ TextBox ชื่อ txtOutput
Dim strData As String
    tcpServer.GetData strData
    txtOutput.text = strData
End Sub
```

ขั้นตอนการสร้างไคลเอน TCP

1. ทำการเพิ่มformใหม่ และตั้งชื่อ frmClient
2. เปลี่ยนชื่อแคปชั่นของformเป็น TCP Client

3. เรียกใช้Winsock และให้ตั้งชื่อให้ tcpClient
4. เพิ่ม TextBox สองตัวลงใน frmClient และตั้งชื่อให้ว่า txtSend และ txtOutput
5. เขียนปุ่ม CommandButton ลงบน form และตั้งให้ว่า cmdConnect
6. เปลี่ยนชื่อ Capture ของปุ่ม CommandButton เป็น Connect
7. เพิ่มโค้ดข้างล่างลงใน form

```
Private Sub Form_Load()
    'ชื่อของWinsock คือ tcpClient
    'การระบุ remote host สามารถระบุได้ทั้ง IP Address (เช่น "121.121.121.121")
    'หรือใช้ชื่อของคอมพิวเตอร์ก็ได้ "friendly"
    tcpClient.RemoteHost = "RemoteComputertName"
    tcpClient.RemotePort = 1001
End Sub


Private Sub cmdConnect_Click()
    'ร้องขอการเชื่อมต่อด้วยเมธอด Connect
    tcpClient.Connect
End Sub

Private Sub txtSend_Change()
    tcpClient.SendData txtSend.Text
End Sub

Private Sub tcpServer_DataArrival (ByVal Bytes Total As Long)
Dim strData As String
    tcpClient.GetData strData
    txtOutput.text = strData
End Sub
```

โค้ดข้างบนเป็นการสร้างแอพพลิเคชั่นเซิร์ฟเวอร์และไคลเอนอย่างง่ายๆ เพื่อใช้ในการส่งข้อความติดต่อสื่อสารกัน โดยขณะที่ใช้โปรโตคอล UDP เราสามารถที่จะมีอิสระในการเปลี่ยนพร็อพเพอตี้ RemoteHost และ RemotePort ขณะที่ยังคงอยู่ใน LocalPort เดียวกัน แต่ถ้าเป็นโปรโตคอล TCP เราจะต้องทำการปิดการเชื่อมต่อก่อนการเปลี่ยนพร็อพเพอตี้ RemoteHost และ RemotePort

พื้นฐานการเชื่อมต่อโปรโตคอล UDP

การสร้างโปรแกรมที่ใช้โปรโตคอล UDP จะง่ายกว่าการสร้างโปรแกรมที่ใช้โปรโตคอล TCP เพราะโปรโตคอล UDP ไม่ต้องการเชื่อมต่อที่ชัดเจน ในแอพพลิเคชั่นที่ใช้โปรโตคอล TCP โดย Winsock Control หนึ่งตัวจะต้องถูกตั้งค่าอย่างชัดเจนไปที่ "listen" เพื่อรับการติดต่อ ขณะที่ตัวอื่นจะต้องเริ่มทำการเชื่อมต่อด้วยวิธีเดียวกัน

Winsock Procedure

ใน Winsock จะมี procedure ที่สำคัญ ที่ใช้สำหรับกระทำการติดต่อสื่อสาร ทั้งทางฝ่ายเซิร์ฟเวอร์ (Server) และฝ่ายไคลเอน (Client) ดังนี้

- Close คือ เหตุการณ์เมื่อมีการหยุดหรือยกเลิกการติดต่อสื่อสารของฝ่าย Server หรือฝ่าย Client โดย Function Winsock.Close ซึ่งทำให้เราสามารถตรวจสอบฝ่ายตรงข้ามได้ว่ามีการติดต่ออยู่หรือไม่ โดยอาจใช้ข้อความเตือน เป็นต้น
- Connect เป็นเหตุการณ์ฝ่าย Client มีการส่งสัญญาณการติดต่อมายังฝ่าย Server ทำให้ procedure ของฝ่าย Server ทำงาน และเราก็สามารถใส่ข้อความเตือนได้เช่นเดียวกัน
- ConnectionRequest เป็นเหตุการณ์ที่ฝ่าย Client ส่งสัญญาณติดต่อกลับมายัง Server procedure พร้อมกับการร้องขอ ID (Request Identification) ซึ่งเป็นค่าที่สร้างขึ้นมาใหม่ (Generate) เพื่อให้ Server สามารถรู้ได้ว่าใช้ ID จากคอนโทรลตัวใด เพื่อจะได้ทำการติดต่อสื่อสารได้ถูก
- DataArrival เกิดขึ้นเมื่อมีการส่งข้อมูลระหว่าง Server และ Client Procedureนี้ก็จะทำงาน
- Error เป็นเหตุการณ์ที่เกิดขึ้นเมื่อเกิดความผิดพลาดระหว่างการทำงานของ Server และ Client โดยจะทำการส่งหมายเลขของความผิดพลาดที่เกิดขึ้นพร้อมกับข้อความในเหตุการณ์นั้น
- SendProgress เกิดขึ้นในขณะที่มีการส่งข้อมูล เมื่อส่งข้อมูลเสร็จจะมีข้อความแสดงบอก เช่น SendComplete
- SendComplete เกิดขึ้นเมื่อทำการส่งข้อมูลไปยังฝ่ายตรงข้ามเสร็จเรียบร้อย

Winsock Properties & Events

- Accept (RequestID) คือการตกลงกันระหว่าง Server และ Client ในการเลือกหมายเลขคอนโทรลให้ตรงกัน
- Close เป็นการส่งสัญญาณยกเลิกการติดต่อ ซึ่งจะเป็นฝ่าย Server หรือ Client ก็ได้ที่จะใช้ Function นี้ทำงาน ส่งผลให้ Procedure close ทางฝั่งตรงข้ามทำงาน
- Connect เป็นการส่งสัญญาณว่าตอนนี้ทำการติดต่อเรียบร้อยแล้ว

- Getdata เป็นการรับข้อมูลเมื่อมีการส่งมาโดยคำสั่งนี้จะอยู่ในส่วนของ Procedure DataArrival
- Listen เป็นการกระทำการตรวจรับสัญญาณที่ส่งไปฝ่ายตรงข้ามตอบรับการร้องขอการติดต่อ
- LocalHostName เป็นคำสั่งที่ส่งชื่อของคอมพิวเตอร์เครื่องนั้นๆ
  Debug.Print Winsock1.LocalHostName
- LocalIP เป็นคำสั่งที่ใช้ในการส่งหมายเลข IP Address
  Debug.Print Winsock1.LocalIP
- LocalPort เป็นคำสั่งที่ใช้ส่งหมายเลขของการติดต่อ TCP/IP ของเครื่องนั้น
  Debug.Print Winsock1.LocalPort
- RemoteHost เป็นการกำหนดหรือคืนค่าชื่อ Computer Name ของเครื่องที่จะทำการติดต่อ
  Winsock1.RemoteHost = Myserver
- RemoteHostIP กำหนดหมายเลข IP Address ของเครื่องที่จะทำการติดต่อ
  Winsock1.RemoteHostIP = 10.10.0.0
- RemoteHostPort กำหนดหมายเลขพอร์ตที่ใช้ติดต่อระหว่างกัน
  Winsock1.RemoteHostPort = 5000
- SocketHandle ใช้คืนค่าของช่องทางที่ใช้ในการติดต่อระหว่างกัน ซึ่งสามารถเรียกดูได้จาก
  Debug.Print Winsock1.SocketHandle
- State ใช้คืนค่าของสถานะซ็อกเก็ตขณะที่อยู่ระหว่างการติดต่อ โดยอาจใช้การตรวจสถานะโดยค่าคงที่เหล่านี้เช่น sckClosed (มีค่า=0) Socket ปิดการใช้งาน, sckOpen (มีค่า = 1) Socket เปิดการใช้งาน หรือ sckError (มีค่า = 9) Socket เกิดความผิดพลาด

2.5 ชุดของฟังก์ชันAPI

API ย่อมาจาก Application Program Interface เป็นอีกทางเลือกหนึ่งที่ยอมให้ฟังก์ชันAPI มองหาเครื่องมือต่างๆ ในระบบปฏิบัติการที่เรามีอยู่ เนื่องจากเครื่องมือทั่วไปใน Visual Basic (เช่น OCXs หรือ ActiveX) อาจยังไม่เพียงพอต่อความต้องการและเพื่อให้ง่ายในการเขียนโปรแกรม เราสามารถเรียกใช้ฟังก์ชันต่างๆ ของระบบปฏิบัติการที่เรามีอยู่ โดยผ่านทางฟังก์ชัน API เนื่องจากว่ากลุ่มฟังก์ชัน API ถูกเขียนขึ้นมาด้วยภาษา C ดังนั้น ทำให้กลุ่มฟังก์ชันAPI ส่วนหนึ่งไม่สามารถถูกเรียกใช้งานโดย Visual Basic ได้ทั้งหมด เนื่องจากว่าชนิดของข้อมูลที่มีโครงสร้างที่แตกต่างกัน

ประโยชน์ของการเรียกใช้ฟังก์ชันAPI

1. ทำให้ขีดความสามารถของ Visual Basic เพิ่มขึ้น เนื่องจากว่าการเรียกใช้งานฟังก์ชัน API เป็นการเรียกใช้งานของระบบปฏิบัติการ Windows ส่งผลให้สามารถขยายขีด

ความสามารถของ Visual Basic ได้ในระดับหนึ่ง กล่าวคือ เมื่อระบบปฏิบัติการ Windows มีการพัฒนาเพิ่มขึ้น ทำให้ฟังก์ชัน API มีความสามารถเพิ่มขึ้นตาม Windows

2. ทำให้ขนาดของแอพพลิเคชันลดลง จุดด้อยข้อหนึ่งของแอพพลิเคชันที่พัฒนาด้วย Visual Basic ก็คือ ขนาดของโปรแกรมมักจะมีขนาดใหญ่เกินความจำเป็น การเรียกใช้งานกลุ่มฟังก์ชันAPI จึงเป็นอีกหนทางหนึ่งที่จะช่วยลดขนาดของแอพพลิเคชันลง
3. ทำให้การประมวลผลโดยรวมเร็วขึ้น เหตุที่ทำให้ภาษา C มีการประมวลผลที่รวดเร็วกว่าภาษาอื่นๆ ส่วนหนึ่งมาจากการที่ภาษา C มีการเรียกใช้งานไลบรารี่โดยตรง หรือพูดได้อีกอย่างหนึ่งว่าเรากำลังใช้งาน Visual Basic ในลักษณะเดียวกับที่ทำในภาษา C โดยมีข้อแตกต่างก็คือ ภาษา C สามารถเรียกใช้ได้ทุกฟังก์ชัน แต่ Visual Basic เรียกใช้ได้เฉพาะบางส่วน

2.5.1 การเรียกใช้ฟังก์ชันAPI

เรารู้มาว่าใน Windows มีไฟล์ DLL อยู่มากมาย บางไฟล์เล็กบางไฟล์ใหญ่ เพื่อที่จะเข้าถึงไฟล์ DLLเหล่านี้ เราสามารถเรียกใช้งาน Windows โดยผ่านทางฟังก์ชัน API ซึ่งสามารถเรียกใช้ได้กว่า 1,000 ฟังก์ชัน และก็เป็นจุดแข็งจุดหนึ่งสำหรับการทำงานกับฟังก์ชัน API ที่เพียงรู้เทคนิคการเรียกใช้งานเพียงเล็กน้อยก็สามารถดึงเอาความสามารถของ Windows มาใช้ประโยชน์ได้อย่างมากมาย และจะช่วยได้มากถ้าเรามีหนังสือที่เอาไว้อ้างอิงเพื่อเรียนรู้เกี่ยวกับว่าไฟล์ DLL แต่ละตัวใช้ทำอะไรบ้าง เพื่อหาการประกาศฟังก์ชัน API ที่ถูกต้อง, ค่าที่คาดหวังเอาไว้ และค่าที่ต้องการให้มีการส่งค่ากลับ

อย่างไรก็ตามใน Visual Basic ได้ช่วยในเรื่องของการประกาศ (declaration) ฟังก์ชันที่ถูกต้อง รวมถึงชนิด (types) และค่าคงที่ (constants) ที่ต้องการ โดยผ่านทาง Text API Viewer โดยเราสามารถที่จะทำการก๊อปปี้จากคลิปบอร์ดและวางมันลงในโค้ดของเราได้โดยตรง

ฟังก์ชันของWindows DLLs

มันจะช่วยได้มาก ถ้าเราคำนึงถึงว่าแต่ละการเรียกใช้ฟังก์ชัน API เป็นโปรแกรมย่อยที่เป็นซับรูทีน (subroutine) หรือเป็นฟังก์ชั่น (function) ไมโครซอฟร์ฉลาดที่จะรวมกลุ่มไฟล์ DLL หลักๆเข้าไว้ด้วยกัน โดยจะแบ่งเป็น 4 กลุ่มหลัก ได้แก่

KERNEL32 เป็นไฟล์ DLL หลัก ที่ทำหน้าที่จัดการเกี่ยวกับหน่วยความจำ โปรแกรมงานหนักๆ หลายอย่างที่ทำงานอยู่ และฟังก์ชันส่วนใหญ่จะมีผลกระทบโดยตรงกับการทำงานของ Windows

USER32 เก็บฟังก์ชันที่เกี่ยวกับการติดต่อกับผู้ใช้งาน ประกอบด้วยฟังก์ชันที่ตกลงเกี่ยวกับ เมนู, เวลา, การติดต่อสื่อสาร, ไฟล์ และอื่นๆอีกมากมายที่ไม่ได้แสดงอยู่บนWindows

GDI32	GDI (Graphic Device Interface) เก็บฟังก์ชันเกี่ยวกับการแสดงผล และฟังก์ชันที่เกี่ยวกับการแสดงกราฟิก
WINMM	ประกอบไปด้วยฟังก์ชันมัลติมีเดียสำหรับการตกลงเกี่ยวกับเสียง, ภาพวีดีโอ และอื่นๆ ซึ่งเป็นไฟล์ DLLขนาด32 บิทเท่านั้น ส่วนถ้าเป็นไฟล์ขนาด 16 บิท จะเท่ากับการเรียกใช้ MMSYSTEM

คุณสามารถพบไฟล์เหล่านี้ได้ในไดเรคทอรี่ของระบบ Windows มีอีกหลายไฟล์ที่มีขนาดเล็กกว่าที่กล่าวมาแล้ว ถูกใช้น้อยกว่าไฟล์ DLL ซึ่งถูกจัดเตรียมไว้เพื่อให้บริการแก่โปรแกรม

ไฟล์ DLL ตามที่ได้กล่าวมาเป็นไฟล์ขนาด 32 บิท ในอดีตไมโครซอฟได้ทำการลบรุ่นเก่าๆ ของVisual Basic ที่ใช้งานบนเพลทฟอร์ม (Platform) ของWindows ขนาด 16 บิท เช่น Window for Workgroup หรือ Window3.1 อย่างไรก็ตาม Visual Basic5.0 เป็นรุ่นแรกที่ใช้ 32 บิท ส่วน Visual Basic4.0 ยังคงเป็นรุ่นสุดท้ายที่ใช้ขนาด 16 บิท ในการพัฒนาเครื่องมือต่างๆจากไมโครซอฟ

สิ่งที่เราต้องทำต่อไปก็คือ เราต้องพยายามค้นหาการเรียกฟังก์ชัน API เราต้องแน่ใจว่าเรารู้เกี่ยวกับทุกสิ่งที่เราต้องการจะรู้เกี่ยวกับการเรียกใช้ฟังก์ชัน API หลังจากนั้นก็ขึ้นอยู่กับคุณจะใช้มันทำอะไร

การเรียกใช้ API จากตัวโปรแกรม

การเรียกใช้โพรซีเยอร์ใน API นั้นไม่แตกต่างกับการเรียกใช้ฟังก์ชันหรือซับรูทีนเลย ซึ่งสิ่งเหล่านี้คุณได้กระทำขึ้นมาเองและเพิ่มมันเข้าไปที่โมดูล (module) ในโปรเจคของคุณ ดังตัวอย่าง

```
Public Sub FindText(objDataControl As Control, sFieldName As String)
'ส่วนของโค้ด
End Sub
```

ร้องขอการกระทำในโพรซีเยอร์ โดยใช้โค้ดนี้

```
FindText datTitles, "Titles"
```

การเรียกใช้ฟังก์ชัน API ก็เหมือนกับตัวอย่างที่กล่าวมา ซึ่งเป็นการเรียกซับโพรซีเยอร์ที่ไม่เพียงแต่อยู่เฉพาะภายนอกโมดูลแต่อยู่ภายนอกโปรแกรม Visual Basic ด้วย

การประกาศฟังก์ชัน API

ก่อนที่รูทีน DLL จะถูกใช้ได้ มันต้องการการประกาศ (declaration) ก่อน Visual Basic ต้องการที่จะบอกถึง

- ชื่อของซับรูทีนและชื่อฟังก์ชันที่ใช้
- สถานที่ที่สามารถพบไฟล์ DLL ได้
- ค่าพารามิเตอร์ที่คาดหวังจะได้รับ
- ชนิดของค่า (value) ที่สามารถส่งกลับในกรณีที่ฟังก์ชันมีลักษณะเป็นรอบ

เรายังสามารถใช้คำว่า “Sub” หรือ “Function” เพื่อเริ่มต้นโค้ด แต่เราจะต้องนำหน้ามันด้วยคำว่า “Declare” เพราะว่าเรากำลังทำการเรียกใช้ฟังก์ชัน API และหลังจากการประกาศฟังก์ชันโค้ดจะไม่ขึ้นตรงกับโปรแกรม Visual Basic ของเรา มันจะขึ้นตรงกับไฟล์ DLL ที่เราบ่งชี้ การประกาศก็เหมือนกับฟังก์ชันหนึ่งที่เราทำการเขียนขึ้นมา ครั้งหนึ่งที่ฟังก์ชันถูกประกาศขึ้นมันจะถูกเรียกอย่างโดยตรง ดังตัวอย่างการเรียกใช้ฟังก์ชัน API

การกระพริบของหน้าต่างด้วยการเรียกใช้ฟังก์ชัน API

1. ทำการสร้างโปรเจคขึ้นมาใหม่จากโปรแกรม Visual Basic
2. เขียนตัวควบคุมเวลาลงบนฟอร์มและทำการตั้งค่าพร็อพเพอตี้ Interval ไปที่ 10 ซึ่งหมายความว่าเหตุการณ์ของเวลาจะปรากฏขึ้นทุกๆ 10 มิลลิเซคคัน
3. ทำการดับเบิ้ลคลิกที่ตัวควบคุมเวลา จะประกฏหน้าต่างที่ไว้สำหรับเขียนโค้ดกำกับ จากนั้นเพิ่มโค้ดข้างล่างนี้ลงไป

```
Private Sub Timer1_Timer ()
Dim nReturnValue As Integer
    nReturnValue = FlashWindow(Form1.hWnd, Ture)
End Sub
```

4. ทำการประกาศ FlashWindow ฟังก์ชัน ดังนี้

```
Private declare Function FlashWindow Lib “user32” Alise “FlashWindow”
(ByVal hWnd As Long, ByVal bInvert As Long) As Long
```

5. ทำการรันโปรแกรมเมื่อฟอร์มปรากฏขึ้น ตัว Caption ของมันจะกระพริบ นี่ก็เป็นตัวอย่างง่ายๆ ในการเรียกใช้ฟังก์ชัน API โดยถ้าเราทำการเขียนโค้ดเหล่านี้จาก Visual Basic จริงๆ จะทำได้ยากมาก และจำนวนโค้ดที่ต้องทำการเขียนก็จะมีมากด้วย

2.5.2 การทำงานของการประกาศฟังก์ชัน API

การประกาศฟังก์ชันจะเป็นไปอย่างตรงไปตรงมาถ้าคุณเข้าใจส่วนสำคัญของมัน มันจะช่วยได้มากถ้าคุณมีหนังสือคู่มือไว้อ้างอิงเกี่ยวกับ Windows API เพื่อระบุถึงการเรียกที่ใช้ที่เป็นซับรูทีนหรือฟังก์ชัน ซึ่งประกอบไปด้วยไฟล์ DLL อยู่ภายใน และพารามิเตอร์อะไรที่ควรจะทำการส่งผ่านค่า

คำว่า “Declare” เพื่อบอกกับ Visual Basic ว่าเรากำลังประกาศใช้ไฟล์ DLL ตามติดกับคำว่า “Declare” ก็เป็นคำว่า “Sub” หรือ “Function” ซึ่งจะประกาศเป็นซับรูทีนหรือฟังก์ชันก็ได้ และแน่นอนเราไม่สามารถประกาศทั้งซับรูทีนและฟังก์ชันปนกันได้ คำว่า “Lib” เป็นตัวบอกกับ Visual Basic ว่าไฟล์ DLL กลุ่มใดที่เราต้องการจะใช้ (มาจากไลบรารี่ใด) ในกรณีตัวอย่าง เราใช้กลุ่มไฟล์ DLL ของ User32 คำว่า “Alias” บอกกับ VB ถึงชื่อที่เราต้องการกระทำของฟังก์ชันชื่ออะไรซึ่งมันควรจะต่างกับชื่อที่เราให้ใน Lib ดังตัวอย่าง

Private declare Function FlashWindow Lib “user32” Alise “FlashWindow”
(ByVal hWnd As Long, ByVal bInvert As Long) As Long

ท้ายที่สุดก็คือค่าพารามิเตอร์ที่เราต้องการจะผ่านค่าไปยังฟังก์ชันที่เราได้ทำการประกาศไว้ ซึ่งเป็นไปตามชนิดของค่าที่เราจะทำการส่งกลับ
ค่าพารามิเตอร์ที่เราต้องทำการส่งผ่านคือ

(ByVal hWnd As Long, ByVal bInvert As Long) As Long

ค่าพารามิเตอร์ตัวแรกคือ hWnd เป็นการจัดการ โดยระบุถึงหน้าต่างที่เราต้องการทำให้เกิดการกระพริบขึ้น มันสำคัญถ้าคุณเข้าใจถึงแนวคิดของการจัดการหน้าต่างเพื่อจะได้ผ่านกลับมาให้มันได้ พารามิเตอร์ตัวที่สอง bInvert เพื่อเป็นเหมือนกับสวิทให้เกิดการกระพริบปิดเปิด โดยถ้าค่านี้ถูกตั้งเป็น Ture ที่ได้จากการเรียกภายในโค้ด ก็จะทำให้เกิดการกระพริบขึ้น และเพื่อจะให้มันกลับไปสู่สถานะเริ่มต้น เราจะต้องทำการเรียกฟังก์ชันอีกครั้ง จะทำให้ค่าเปลี่ยนกลับเป็น False

มีการเรียกฟังก์ชัน API มากมายที่ชื่อของ Alias ไปเหมือนกับชื่อของการกระทำที่มีอยู่จริงในรูทีน เช่น FlashWindow ในกรณีนี้ เราสามารถรวมส่วนของ Alias ไว้กับส่วนของฟังก์ชั่นได้เลย

Private declare Function FlashWindow Lib “user32”
(ByVal hWnd As Long, ByVal bInvert As Long) As Long

อย่างไรก็ตาม บางชื่อก็ผิดกฎใน VB เช่น _lopen และอื่นๆ อีกมากที่มาพร้อมความแตกต่างของรุ่นของโปรแกรม VB ดังนั้นจึงเป็นการปลอดภัยกว่าถ้าเราทำการระบุ Alias ลงไปด้วย

รูปแบบการส่งผ่านค่าของอาร์กิวเมนต์ (Passing Argument)

สำหรับรูปแบบของการส่งค่าอาร์กิวเมนต์ (Argument) ในฟังก์ชัน API สามารถแบ่งได้ 2 ลักษณะ ดังนี้

1. การส่งผ่านค่าแบบ ByVal หมายถึง เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ในฟังก์ชันเกิดขึ้น การเปลี่ยนแปลงดังกล่าวจะไม่มีผลกระทบกับค่าของอาร์กิวเมนต์ต้นฉบับที่ส่งเข้ามา หรืออาจกล่าวได้อีกอย่างหนึ่งว่า ค่าของอาร์กิวเมนต์ต้นฉบับ ไม่เกิดการเปลี่ยนแปลงใดๆเกิดขึ้น ซึ่งก็คือ การที่อาร์กิวเมนต์ต้นฉบับจะก๊อบปี้ค่าตัวมันเองก่อน แล้วจึงส่งค่าก๊อบปี้นั้นเข้าไปในฟังก์ชัน ส่งผลให้เมื่อมีการเปลี่ยนแปลงค่าอาร์กิวเมนต์ในฟังก์ชันภายหลัง ค่าของอาร์กิวเมนต์ต้นฉบับจึงไม่มีการเปลี่ยนแปลง การส่งค่าแบบนี้จะเป็นวิธี

ค่าที่ถูกตั้งไว้เป็นdefault และในฟังก์ชัน API ส่วนใหญ่ จึงใช้คำ ByVal นำหน้าอาร์กิวเมนต์ตัวนั้น

2. ส่งค่าแบบ ByRef หมายถึง เมื่อมีการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ที่ส่งเข้ามา ค่าของอาร์กิวเมนต์ต้นฉบับ จะถูกเปลี่ยนแปลงไปด้วย สามารถByRef นำหน้าอาร์กิวเมนต์ตัวนั้น

ทำไมถึงใช้Visual Basic

เพราะ Visual Basic สนับสนุนการเรียกกลับของฟังก์ชัน (callback) ถ้าปราศจากสิ่งเหล่านี้ โปรแกรม VFW จะต้องการโปรแกรม 3 กลุ่มเพื่อใช้ในการติดต่อสื่อสารกับคราส การเรียกใช้ฟังก์ชันการเรียกกลับทำให้ขั้นตอนการทำงานทางคณิตศาสตร์เป็นไปได้เร็วมาก ซึ่งมันเป็นพื้นฐานของการจัดการทางด้านกราฟิก เครื่องมือการพัฒนาที่ทำได้รวดเร็วอย่าง Visual Basic

ขั้นตอนการสร้างแอพพลิเคชั่นในการ Capture ภาพ

ขั้นตอนที่ 1 สร้างหน้าต่างที่ใช้ในการจับภาพ

ขบวนการการจับภาพวีดีโอจะเริ่มต้นจากการใช้ฟังก์ชัน capCreateCaptureWindowA ซึ่งเป็นการสร้างหน้าต่างขึ้นใหม่และฟังก์ชันนี้ก็สามารถที่จะจัดการคืนค่ากลับ ฟังก์ชันการจัดการนี้มีขนาด 32 บิท ซึ่งใช้ในพอยเตอร์ หรือใช้ในการอ้างอิงของหน้าต่าง การจัดการนี้เป็นตัวหลักที่อยู่ในตัวโปรแกรมของเรา จึงควรถูกเก็บไว้ในที่ที่ปลอดภัย เราสามารถที่จะปรับขนาดมุมมองของหน้าต่างนี้โดยการเปลี่ยนค่าพารามิเตอร์ต่างๆได้ตามต้องการ

```
lwndC = capCreateCaptureWindowA ("My Capture Window", WS_CHILD Or
WS_VISIBLE, 0, 0, 160, 120, Me.hwnd, 0)
```

ทุกคำสั่งจะถูกส่งโดยการส่งข้อความ (Message) ไปที่หน้าต่างที่เราสร้างขึ้น ข้อความดังกล่าวจะถูกส่งด้วยฟังก์ชัน SendMessage ซึ่งสร้างขึ้นจากWIN32 จากการเรียกใช้ฟังก์ชัน API คุณสามารถผ่านฟังก์ชันเหล่านี้เพื่อจัดการหน้าต่างที่คุณต้องการส่งคำสั่งไปเป็นข้อความขนาด 32 บิท, ค่าพารามิเตอร์ 16 บิท, ค่าพารามิเตอร์ 32 บิท ข้อความขนาด 32 บิท จะถูกแทนที่ด้วยค่าคงที่ (Constant) ที่ง่ายต่อการเข้าใจ

ขั้นตอนที่ 2 การเชื่อมต่อกับWindow

การสร้างหน้าต่างของการจับภาพขึ้นครั้งหนึ่ง เราสามารถทำการติดต่อหน้าต่างนี้กับวีดีโอไดรเวอร์ที่ได้ทำการติดตั้งด้วยกล้องวีดีโอของเรา การเชื่อมต่อจะถูกกระทำขึ้นโดยข้อความ WM_CAP_DRIVER_CONNECT ซึ่งจะทำการผูกตัวหน้าต่างที่เราได้ทำการสร้างเข้ากับไดลเวอร์วีดีโอตัวแรกที่ถูกค้นพบในเครื่องของเรา ระลึกไว้ในใจเสมอว่าเราไม่สามารถส่งข้อความนี้ให้กับหน้าต่างได้ทุกหน้าต่าง แต่จะเกิดขึ้นเฉพาะกับหน้าต่างที่เราทำการสร้างด้วย capCreateCaptureWindow เท่านั้น

```
SendMessage lwnd, WM_CAP_DRIVER_CONNECT, 0, 0
```

CapDriverConnect lwnd, 0

การใช้ฟังก์ชัน SendMessage จะช่วยลดคำสั่งที่อาจเกิดความสับสนในการผ่านค่าพารามิเตอร์ เราควรใช้ฟังก์ชันนี้เมื่อไหร่ก็ตามที่เป็นไปได้

การผ่านค่า String ไปยังฟังก์ชัน SendMessage

การส่งข้อความโดยใช้ฟังก์ชั่นAPI เป็นการผ่านค่าพารามิเตอร์อยู่ 4 ค่าคือ พารามิเตอร์ที่เกี่ยวกับการจัดการหน้าต่าง (hwnd), พารามิเตอร์ที่เกี่ยวกับข้อความที่ส่งไปยังหน้าต่าง (wMsg), พารามิเตอร์ที่มีขนาดสั้น (wParam), พารามิเตอร์ที่มีขนาดยาว (lParam) โดยตัว "w" ใน wParam หมายถึงค่าพารามิเตอร์นี้เป็นข้อมูลชนิด "WORD" และตัว "l" ใน lParam หมายถึงค่าพารามิเตอร์นี้เป็นข้อมูลชนิด "LONG" ซึ่งเป็นชนิดข้อมูลของโปรแกรม C ข้อมูลชนิด WORD นี้เป็นข้อมูลขนาด 16 บิท และ ข้อมูลชนิด LONG เป็นข้อมูลชนิด 32 บิท ใน VB เราเรียกข้อมูลชนิดเหล่านี้ว่าข้อมูลชนิด INTEGER และ LONG การประกาศฟังก์ชันจะมีลักษณะเป็นดังตัวอย่างข้างล่าง

```
Declare Function SendMessage Lib "user32" Alise "SendMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Integer,
ByVal lParam As Long) As Long
```

คำสั่งนี้เพื่อเป็นการผ่านค่าข้อมูลชนิด String ไปให้ฟังก์ชัน เราต้องการที่จะระบุ lParam ใหม่เพื่อให้เป็นชนิดของ String แทน Long สิ่งหนึ่งที่เราต้องคิดเมื่อทำการส่งค่าชนิดที่ผิดไปให้ยังไฟล์ DLL ซึ่งจะทำให้เกิดปัญหาใหญ่ได้ เราจะพบใน Visual Basic เมื่อเราทำการผ่านค่าที่เป็น STRING โดยค่าของมันถูกส่งไปยังภายนอกฟังก์ชัน มันจะทำการส่งพอยเตอร์ไปยัง STRING แทน โดยพอยเตอร์นี้เป็นขนาด 32 บิท ซึ่งเป็นข้อมูลชนิด LONG ซึ่งเป็นค่าที่คาดหวังว่าจะคืนกลับมา อีกอย่างหนึ่งก็คือเราไม่สามารถเปลี่ยนการประกาศฟังก์ชันของเราได้ เพราะการเรียกบางอย่างต้องการ lParam ที่เป็นชนิด LONG ดังนั้นเราสามารถที่จะทำการประกาศฟังก์ชันขึ้นใหม่และเรียกใช้บางสิ่งที่แตกต่างออกไป เช่น

```
Declare Function SendMessage Lib "user32" Alise "SendMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Integer,
ByVal lParam As String) As Long
```

ภาพวีดีโอบนวินโดว์ (Video For Windows: VFW)

ไมโครซอฟได้สร้างฟังก์ชันการทำงานของระบบเพื่อใช้ในการเล่นภาพ และจับภาพ (capture) วีดีโอจากกล้องวีดีโอดิจิตอล กล้องโดยทั่วไปจะสามารถทำการติดต่อกับเครื่องคอมพิวเตอร์โดยผ่านทางพอร์ทขนาน (parallel port), USB port หรือการ์ดชนิดพิเศษต่างๆ กล้องโดยส่วนใหญ่จะมาพร้อมกับ

ซอร์ฟแวร์ที่ใช้กับตัวของมันเอง กล้องที่ทันสมัยจะมีโปรแกรมสำหรับการจับภาพมาในระบบของมันด้วย ซึ่งทำให้โปรแกรมตัวอื่นสามารถใช้ประโยชน์จากมันได้ เราสามารถที่จะดูว่าในคอมพิวเตอร์ของเราว่ามีการลงไดรเวอร์ของการจับภาพไว้บ้างหรือเปล่าโดยดูใน Control Panel, Multimedia, บนแท็บ Device, ดูส่วนของ Video Capture Device เราสามารถเลือกเพื่อทำการติดต่อกับ Driver ที่เราต้องการ

ไฟล์ AVI คืออะไร

AVI (Audio Video Interleave) ถูกออกแบบโดยไมโครซอฟ ไฟล์ AVI เป็นไฟล์ที่มีรูปแบบธรรมดามากสำหรับข้อมูลที่เป็นเสียง และภาพวีดีโอบนเครื่องคอมพิวเตอร์ส่วนบุคคล เหมาะสำหรับรูปของภาพวีดีโอที่มีขนาดเล็กๆ เป็นรูปแบบการเข้าโค้ดของภาพและเสียงที่เป็นดิจิตอลเพื่อใช้ในการส่งผ่าน

เฟรมหลอก (Phantom Frame)

ในการเล่นไฟล์ AVI ด้วยโปรแกรม Microsoft Window คุณจะพบกับเฟรมหลอกที่อยู่ที่ตำแหน่งเฟรมสุดท้าย เฟรมหลอกจะปรากฏเป็นเฟรมเปล่าหรือถูกถ่ายสำเนาจากเฟรมท้ายสุดหรือไม่ก็ขึ้นอยู่กับโปรแกรม Windows ที่เรานำมาใช้ เช่นถ้าเรามีไฟล์ AVI ซึ่งประกอบด้วย 30 เฟรม และเราต้องการจะเล่นมันผ่าน Microsoft VidEdit 1.1 ตัว VidEdit จะทำการเล่นบนตำแหน่งหมายเลข 0-30 เพราะฉะนั้นจึงทำให้เกิดเฟรมว่างที่ตำแหน่ง 30 (0-30 หมายถึงรวมแล้วมี 31 ตำแหน่ง) เราจึงเรียกเฟรมสุดท้ายเป็นเฟรมหลอก แต่ถ้าเราเล่นไฟล์ AVI บนโปรแกรม Microsoft Media Player มันก็จะปรากฏเป็นตำแหน่ง 0-30 เช่นเดียวกัน โดยตำแหน่งเฟรมที่30 จะถูกถ่ายสำเนามาจากเฟรมที่29

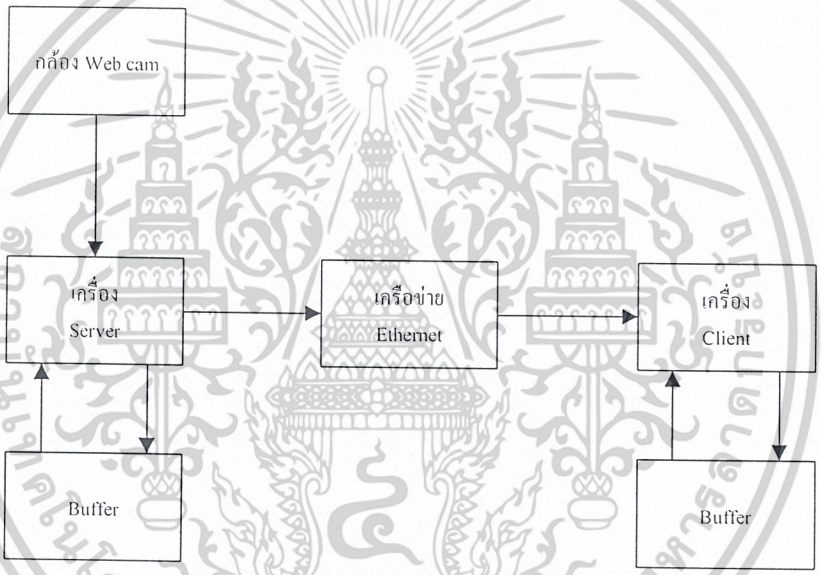
บทที่ 3
การคำนวณและการสร้าง

การคำนวณและการสร้าง

3.1 ส่วนประกอบของระบบ

ระบบของเราจะประกอบไปด้วย 5 ส่วนต่างๆ ที่สำคัญดังนี้

1. กล้อง Web camera ซึ่งเป็นแหล่งกำเนิดข้อมูลในการนำข้อมูลเข้า เป็นกล้องยี่ห้อ Logictech
2. เครื่องคอมพิวเตอร์ที่เราจะให้เป็นตัวเซิร์ฟเวอร์ เป็นเครื่องที่ใช้ระบบปฏิบัติ Windows XP
3. เครื่องคอมพิวเตอร์ที่เราจะให้ทำหน้าที่เป็นไคลเอน ใช้ระบบปฏิบัติการ Windows Me
4. เครือข่าย Ethernet ที่ใช้มีแบนวิด 10 Mbps
5. Buffer ของทั้งทางด้านเครื่องส่งและเครื่องรับ

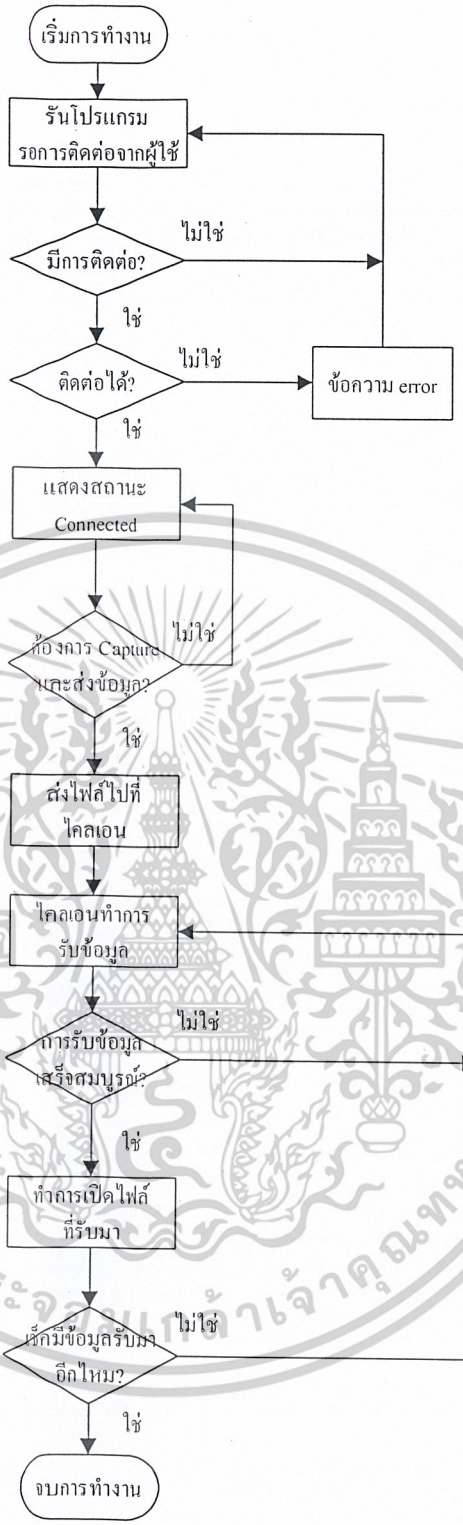


รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

โดยเราจะทำการนำภาพจากกล้อง Web cam มาแปลงให้เป็นข้อมูลที่มีลักษณะที่เหมาะสม แล้วจึงทำการส่งข้อมูลนี้ผ่านเครือข่าย Ethernet เราทำการเขียนแอพพลิเคชั่นเพื่อใช้ในการติดต่อกับกล้อง Web cam จากนั้นทำการโหลดภาพจากกล้อง Web cam มาแสดงบนแอพพลิเคชั่นของเรา เครื่องที่ต่ออยู่กับกล้อง Web cam นี้ เราจะให้มันทำหน้าที่เป็นเซิร์ฟเวอร์ ข้อมูลภาพที่ได้จากกล้อง web cam จะได้จากการ Capture ภาพมาเป็นลักษณะของไฟล์ ไฟล์ที่เราทำการ Capture จะมีขนาดเล็กหรือใหญ่ขึ้นกับว่าเราได้ทำการ Capture แบบใด ถ้าเราทำการ Capture แบบเฟรมเดี่ยว ขนาดของไฟล์ก็จะมีขนาดเล็ก แต่ถ้าทำการ Capture แบบต่อเนื่อง หรือ เป็นวีดีโอ ขนาดของไฟล์ก็จะมีขนาดใหญ่มากๆ ที่กล่าวในลักษณะนี้ก็เพื่อจะ

ใกล้เคียงกับภาพที่เกิดขึ้นจริง ณ เวลาขณะนั้น ด้วยเหตุนี้จึงเป็นสาเหตุสำคัญประการหนึ่งที่ทำให้เราเลือกใช้การ Capture ที่เป็นภาพเฟรมเดียว แล้วจึงทำการส่ง เพราะภาพเฟรมๆ เดียวนั้นมีขนาดข้อมูลที่เล็ก ทำให้การส่งข้อมูลไปยังปลายทางมีความรวดเร็ว มีความเป็นไปได้ที่จะให้ภาพที่ปรากฏที่ปลายทางนั้นมีลักษณะเหมือนกับภาพที่ต้นทางแสดง

ทางเครื่องที่ทำหน้าที่เป็นไคลเอนเราจะทำให้มันมีลักษณะที่เป็นการรอคอยการติดต่อจากทางเครื่องเซิร์ฟเวอร์เพียงฝ่ายเดียว เมื่อทางเครื่องเซิร์ฟเวอร์ติดต่อมาจะแสดงสถานะของการติดต่อออกไปเพื่อบอกให้รู้ว่าพร้อมที่จะทำการรับข้อมูล กรณีเกิดผิดพลาดใดๆ เครื่องไคลเอนจะทำการส่งข้อความสถานะของการติดต่อไปให้ฝั่งเซิร์ฟเวอร์ ข้อความนี้เป็นตัวบอกให้รู้ถึงสาเหตุของความผิดพลาดนั้นๆ เพื่อฝั่งเซิร์ฟเวอร์จะได้ทำการแก้ไขหรือรอเพื่อจะทำการใดๆ ต่อไป



รูปที่ 3.2 ขั้นตอนการทำงานของระบบ

3.2 โปรโตคอลที่ใช้ในการทดลอง

สิ่งที่ต้องคำนึงถึงอีกอย่างหนึ่งก็คือ การเลือกใช้โปรโตคอลในการส่งผ่านข้อมูลซึ่งแต่ละโปรโตคอลก็มีข้อดีข้อเสียที่แตกต่างกัน ดังนั้นเราจึงต้องเลือกใช้โปรโตคอลให้เหมาะสมกับชนิดของการใช้งาน ในที่นี้โปรโตคอลที่เราทำการเลือกใช้มีอยู่สองชนิดด้วยกันคือ โปรโตคอล TCP และโปรโตคอล UDP โดยเลือกจากข้อแตกต่างดังต่อไปนี้

1. โปรโตคอล TCP จะพิจารณาขนาดของข้อมูลที่จะทำการส่ง แล้วทำการแบ่งออกเป็นเซ็คเมนต์เล็กๆ เพื่อทำการส่งต่อไป ส่วนโปรโตคอล UDP เราจะต้องทำการกำหนดขนาดของข้อมูลในการส่งเอง เช่น ถ้าเรากำหนดขนาดเล็กไปก็จะทำใช้ประสิทธิภาพการส่งได้ไม่เต็มที่ ซึ่งขนาดของข้อมูลจะมีขนาดไม่เกิน 64 Kbit ทำให้ไม่เหมาะสมกับการส่งข้อมูลที่มีขนาดใหญ่
2. โปรโตคอล TCP มีความเชื่อถือสูง ถ้าในเวลาที่กำหนดไม่มีการตอบรับกลับมา ก็จะทำการส่งกลับไปใหม่ หรืออาจจะยกเลิกการติดต่อก็ได้ แล้วแต่กรณี
3. โปรโตคอล TCP มีการตอบรับข้อมูลที่ส่งมาเสมอ
4. โปรโตคอล TCP มีการตรวจสอบข้อมูล เพื่อที่จะแน่ใจได้ว่าข้อมูลที่ถูกส่งไปจะไม่มีการเปลี่ยนแปลงระหว่างทาง ถ้าเกิดความผิดพลาดเกิดขึ้นทางฝั่งไคลเอนก็จะทำเสมือนกับว่าไม่ได้รับข้อมูลนั้น เพื่อให้ทางเซิร์ฟเวอร์ทำการส่งใหม่ ต่างกับโปรโตคอล UDP ที่ไม่มีการตรวจสอบความถูกต้องของข้อมูล
5. จากที่โปรโตคอล TCP อาศัย IP แยกข้อมูลออกเป็นส่วนย่อยๆ โปรโตคอล TCP จะทำการลำดับของข้อมูลที่แยกมานั้นประกอบกับให้ถูกต้องสมบูรณ์ ซึ่งต่างจากโปรโตคอล UDP ไม่มีการลำดับของข้อมูล
6. โปรโตคอล TCP มีการควบคุมการไหลของข้อมูลเพื่อให้ทั้งฝั่งเซิร์ฟเวอร์และฝั่งไคลเอนทำงานสัมพันธ์กัน ทั้งนี้ขึ้นอยู่กับบัฟเฟอร์ที่ใช้ การส่งข้อมูลจะอยู่ภายใต้ขีดจำกัดของบัฟเฟอร์ เพื่อป้องกันการส่งข้อมูลที่มีขนาดใหญ่ หรือการส่งข้อมูลด้วยความเร็วเกินไป

จากข้อแตกต่างดังกล่าวทำให้เราเลือกใช้โปรโตคอล TCP ซึ่งเหมาะสมกับการใช้งานบนแอพพลิเคชั่นของเรา เนื่องจากข้อมูลที่เราต้องการจะส่งเป็นข้อมูลที่มีขนาดค่อนข้างใหญ่, ต้องการความถูกต้องของข้อมูล และต้องการตอบรับการรับข้อมูลเสมอเมื่อถึงมือผู้รับ ด้วยเหตุผลเหล่านี้จึงทำให้เราเลือกโปรโตคอล TCP ในการใช้งาน แทนที่จะเลือกโปรโตคอล UDP ที่เหมาะสำหรับการส่งข้อมูลที่มีขนาดเล็กๆ ส่งอยู่ในช่วงระยะเวลาหนึ่ง และไม่ต้องการความถูกต้องของข้อมูลมากนัก

3.3 รูปแบบของภาพที่ใช้

รูปแบบ (Format) ของภาพที่ใช้ในการทดลองจะเป็นไฟล์ .avi ซึ่งได้จากการเรียกใช้ไดรเวอร์ของกล้อง Web cam เราสามารถกำหนดรูปแบบต่างๆ ที่ต้องการผ่านทางไดรเวอร์ของกล้อง Web cam เช่นค่าความสว่าง(Brightness), กำหนดขนาดของภาพ, ค่าขนาดของสีที่ใช้ และอื่นๆ อีกมาก ซึ่งขึ้นอยู่กับความสามารถของกล้อง เช่น ถ้าเราทำการกำหนดภาพให้มีรูปแบบเป็นภาพขาวดำก็จะทำให้ขนาดของ

รูปภาพมีขนาดเล็กลงกว่าตอนที่ภาพมีรูปแบบเป็นสี ขนาดของภาพที่เราทำการส่งในการแคปเจอร์ภาพเฟรมเดียวจะมีขนาดประมาณ 200 Kbit ซึ่งค่านี้เป็นค่าที่ใช้ในการทดลอง โดยเรากำหนดให้ภาพมีขนาที่เล็กที่สุดประมาณ 160*120

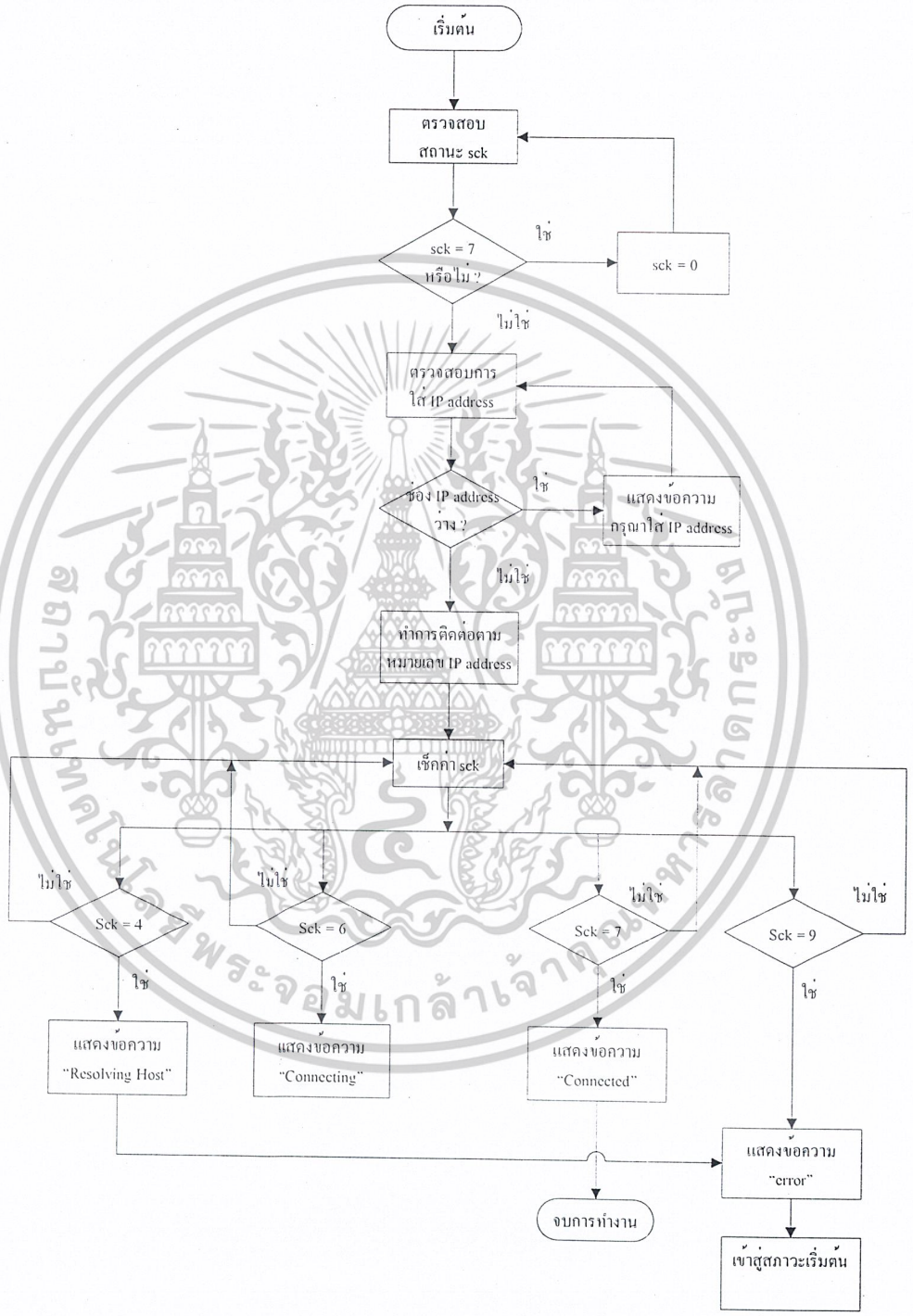
3.4 การทำงานด้านฝั่งเซิร์ฟเวอร์

ทางด้านฝั่งเซิร์ฟเวอร์เราจะทำการสร้างแอพพลิเคชั่นของเราให้มีเหตุการณ์ที่สำคัญๆ อยู่ด้วยกัน 4 เหตุการณ์ ดังนี้คือ

- เหตุการณ์ของการร้องขอการติดต่อ (Connect)
- เหตุการณ์ของการแคปเจอร์ (Capture)
- เหตุการณ์ในการออกจากแอพพลิเคชั่น (Exit)
- เหตุการณ์ในการเลือกไฟล์ที่จะทำการส่ง (Browse)

รูปที่ 3.3 ขั้นตอนการทำงานของฝั่งเซิร์ฟเวอร์

จากรูปที่ 3.3 แสดงขั้นตอนการทำงานของฝั่งเซิร์ฟเวอร์ ซึ่งในช่วงแรกจะต้องรอการติดต่อจากผู้ใช้โดยทำการคลิกที่เมนู Connect เมื่อมีการติดต่อเราจะได้ข้อความบอกถึงสถานะของการติดต่อ ถ้าการติดต่อประสบผลสำเร็จ จะทำการแสดงข้อความ “Connected” ถ้าไม่สำเร็จก็จะแสดงสถานะผิดพลาดต่างๆ ออกมา



รูปที่ 3.4 ขั้นตอนการทำงานเมื่อทำการติดต่อไปที่ฝั่งไคลเอน

จากรูปที่ 3.4 เป็นเหตุการณ์ของการทำการติดต่อ เพื่อทำการติดต่อไปที่ฝั่งไคลเอน โดยเราจะกระทำเหตุการณ์นี้ผ่านทางเมนูไฟล์ ซึ่งเหตุการณ์นี้จะเกิดขึ้นเมื่อเราทำการคลิก Connect ที่เมนูไฟล์ เหตุการณ์นี้เสมือนกับการเชื่อมเส้นทางของการส่งผ่านข้อมูลรอไว้ก่อน เพื่อใช้ในการส่งข้อมูลเมื่อต้องการ และจะมีช่องแสดงสถานะในการติดต่อสื่อสาร โดยการอ้างอิงสถานะของซ็อคเก็ตที่เกิดจากการใช้ "Winsock Control" โดยสถานะต่างๆจะมีความหมายต่างกัน เช่น sck = 0 หมายถึง ซ็อคเก็ตมีการปิดอยู่ไม่สามารถใช้งานได้ sck = 4 หมายถึง มีการใส่ชื่อ IP address ผิด ทำให้ฝั่งเซิร์ฟเวอร์ทำการหาเครื่องที่มีหมายเลข IP address ดังกล่าวซึ่งไม่มีในเครือข่าย sck = 6 หมายถึง กำลังทำการเชื่อมต่อ sck = 7 หมายถึง การเชื่อมต่อได้ทำเสร็จแล้ว sck = 9 หมายถึงเกิดการผิดพลาดในการเชื่อมต่อ

รูปที่ 3.5 ขั้นตอนของการ Capture

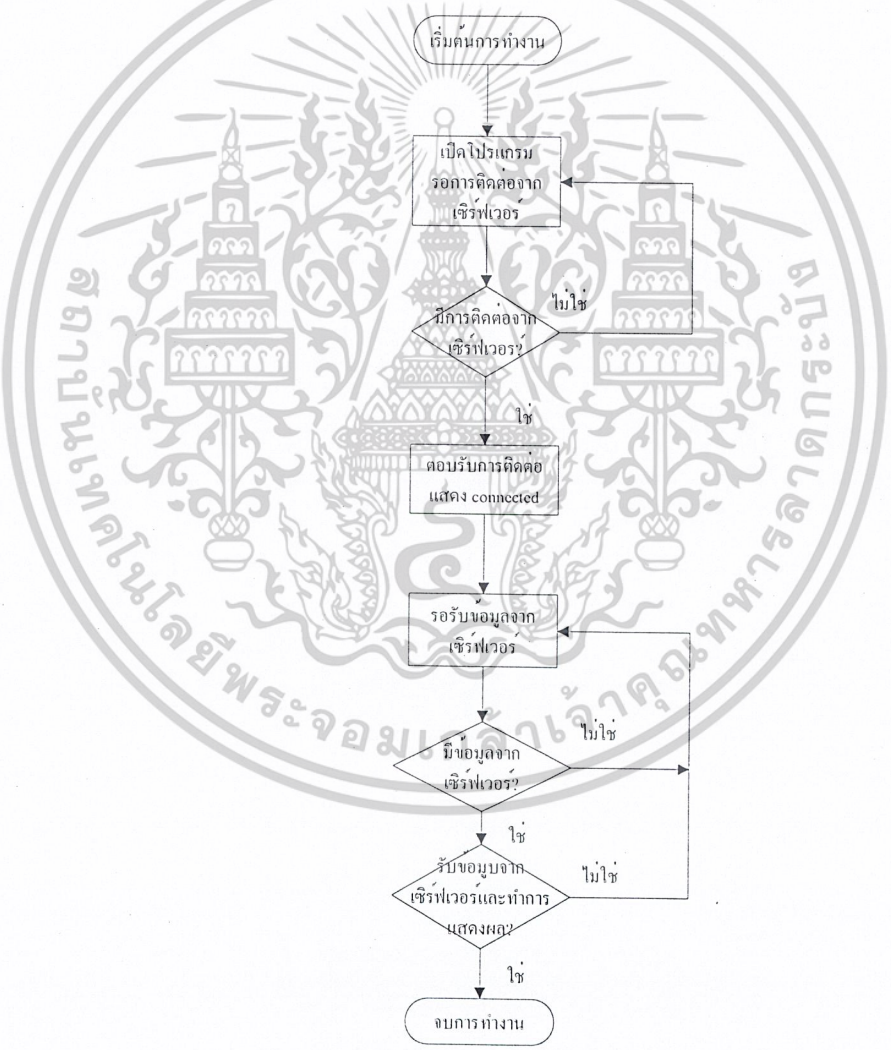
รูปที่ 3.5 เป็นเหตุการณ์ของการแคบเจอร์ซึ่งจะทำการแคบเจอร์และทำการส่งไฟล์ไปพร้อมๆกัน เหตุการณ์จะเกิดขึ้นเมื่อทำการคลิกที่ปุ่มแคบเจอร์ โดยจะมีการตรวจสอบภายในตัวว่ามีการเลือกไฟล์

ที่ต้องจะส่งหรือยังเสมือนกับการคลิกปุ่ม Browse โดยเมื่อเราทำการคลิกปุ่มนี้ครั้งแรกมันจะทำการแคบเจอร์ภาพที่ได้กล้อง (Web camera) แล้วจึงทำการตรวจสภาวะต่างๆ เช่น ตรวจว่ามีการเลือกไฟล์ที่จะทำการส่ง ตรวจสอบสถานะของซ็อคเก็ตว่ามีการติดต่อแล้วหรือยัง เมื่อตรวจสอบสภาวะเรียบร้อยแล้วจึงทำการส่งไฟล์ที่ได้จากการแคบเจอร์นั้นไปที่ฝั่งไคลเอน

3.5 การทำงานทางด้านฝั่งไคลเอน

มีเหตุการณ์ที่สำคัญอยู่ 3 เหตุการณ์คือ

- เหตุการณ์ในการร้องขอการติดต่อ (Connect) และยกเลิกการติดต่อ (Disconnect)
- เหตุการณ์ในการรับข้อมูลและการเล่นภาพ
- เหตุการณ์ในการออกจากโปรแกรม (Quit)



รูปที่ 3.6 ขั้นตอนการทำงานด้านฝั่งไคลเอน

จากรูปที่ 3.6 แสดงขั้นตอนการทำงานทางด้านฝั่งไคลเอน โดยเราจะทำการสร้างให้มีเพียงแต่ช่องแสดงสถานะในการติดต่อ การรับข้อมูล และหน้าต่างแสดงการเล่นไฟล์ที่ได้รับมาจากฝั่งเซิร์ฟเวอร์ โดยทำการตั้งค่าพอร์ตเริ่มต้นให้ตรงกับทางเซิร์ฟเวอร์ และรอแต่เพียงการร้องขอการติดต่อมาเท่านั้น โดยเมื่อทางฝั่งเซิร์ฟเวอร์ทำการร้องขอการติดต่อ ฝั่งไคลเอนจะทำการยอมรับเพื่อตกลงการร้องของฝั่งเซิร์ฟเวอร์ แล้วจึงทำการติดต่อเพื่อรอทางฝั่งเซิร์ฟเวอร์ส่งข้อมูลมา เมื่อฝั่งเซิร์ฟเวอร์ทำการส่งข้อมูลมา ฝั่งไคลเอนจะทำการหยิบไฟล์ที่ได้จากฝั่งเซิร์ฟเวอร์จากพาร์ทที่เก็บไฟล์นั้น แล้วทำการเล่นไฟล์ๆ นั้นแสดงออกเป็นภาพออกมาทางหน้าต่างแอพพลิเคชั่นที่ได้ทำการออกแบบไว้

บทที่ 4
การทดลอง และ ผลการทดลอง

4.1 อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่อง Computer	2 เครื่อง
2.	การ์ด LAN Ethernet 10 Mbps	2 อัน
3.	กล้อง Web cam ยี่ห้อ Logitech	1 อัน
4.	Switch Hub 10 Mbps	1 อัน

4.2 การทดลอง

4.2.1 ด้านส่งหรือ Server

4.2.1.1 ในการทดลอง การส่งข้อมูลภาพผ่านเครือข่ายเราจะทำการทดลองการรันโปรแกรมเพื่อให้เห็นสถานะของการรันโปรแกรมตอนแรกตลอดจนการกดปุ่ม Connect เพื่อติดต่อทางด้านไคลเอนแสดงสถานะการติดต่อได้สำเร็จและสถานะการติดต่อไม่สำเร็จ การกดปุ่ม Capture โดยที่ยังไม่มีการกดปุ่ม Brows ก่อนการกดปุ่ม Captureจะมีการแสดงข้อความเตือน "ไม่มีข้อมูลที่จะส่งครับ" ทำการกดปุ่ม Ok แล้วจะแสดง Common Dialog Box ให้ทำการเลือกข้อมูลคือ C:\Capture.avi กดปุ่ม Open ทำการกดปุ่ม Capture อีกครั้งก็จะมีการส่งข้อมูลภาพไปยังฝั่งไคลเอน

4.2.1.2 เป็นการแสดงสถานะต่าง ๆ ของ Winsock Control ที่แท็บแคปชั่นของForm ของโปรแกรมซึ่งมีสภาวะต่าง ๆ ดังต่อไปนี้ 1.สภาวะเริ่มต้น (Default Closed) คือสภาวะที่ยังไม่มีการติดต่อของโปรแกรม 2.สภาวะ Connected คือสภาวะการติดต่อสำเร็จพร้อมที่จะรับส่งข้อมูลผ่านเครือข่ายได้ 3. สถานะเป็นกำลังติดต่อ (Connecting) 4.สถานะความผิดพลาด (Error) 5.สถานะ (Resolving Host) กำลังทำการหาเครื่องคอมพิวเตอร์ตามที่หมายเลข IP address ทำการระบุไว้ 6.สถานะPeer is closing the connection คือเป็นความผิดพลาดที่เกิดขึ้นเมื่อทั้งสองฝั่งได้ทำการติดต่อกันได้แล้ว แต่ทางฝั่งไคลเอนได้ทำการปิดตัวแอพพลิเคชั่นลงไปก่อน

4.2.1.3 เป็นการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อและขณะที่เครื่องรับไม่ได้เปิดโปรแกรมรอการติดต่อ

4.2.1.4 เป็นการแสดงการกดปุ่ม Brows ก่อนการกดปุ่ม Capture ในสถานะการฝั่ง Server และฝั่ง Client ติดต่อเรียบร้อยแล้ว

4.2.1.5 เป็นการแสดงการกดปุ่ม Brows ก่อนการกดปุ่ม Capture ในสถานะการฝั่ง Server และฝั่ง Client ยังไม่ได้ติดต่อซึ่งกันและกัน

4.2.1.6 เป็นการแสดงทำการกดปุ่ม Connect ในขณะที่ช่อง IP ADDRESS ว่างอยู่

4.2.1.7 เป็นการแสดงเมื่อเปิดโปรแกรมแล้ว โปรแกรมไม่สามารถติดต่อกับ Diver ของกล้องได้

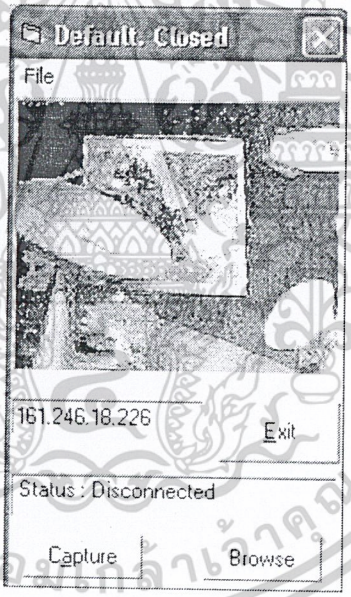
4.2.1.8 เป็นการแสดงให้เห็นเมื่อที่แท็บแคปชั่นของForm แสดงข้อความ"Peer is closing the connection" ทำการกดปุ่ม Connect ที่ฝั่งเซิร์พเวอร์ขณะที่ฝั่งไคลเอนทำการเปิดโปรแกรมรอการติดต่อ

4.2.1.9 เป็นการแสดงให้เห็นเมื่อมีการติดต่อกันสำเร็จแล้วแต่ปิดโปรแกรมฝั่งเซิร์พเวอร์ขณะที่ฝั่งไคลเอนยังไม่ยกเลิกการติดต่อแล้วเปิดโปรแกรมฝั่งเซิร์พเวอร์ขึ้นมาอีกครั้งและทำการติดต่ออีกครั้ง

ในการทดลอง การส่งข้อมูลภาพผ่านเครือข่ายถ้าเราทำการแบ่งการส่งข้อมูลไม่สัมพันธ์กับขนาดของข้อมูลที่จะทำการส่งจะทำให้ด้านรับไม่สามารถรับข้อมูลได้ทั้งหมด ทำให้ไม่สามารถแสดงผลทางด้านรับได้และถ้าระหว่างการส่งข้อมูลที่ยังไม่เสร็จสิ้นจากด้านส่งไปยังด้านรับ และเราทำการส่งข้อมูลเดิมไปอีกจะเกิดการทับซ้อนของข้อมูลทางด้านรับ จะทำให้เกิด Error ทางด้านรับทำให้ไม่สามารถทำงานต่อไปได้

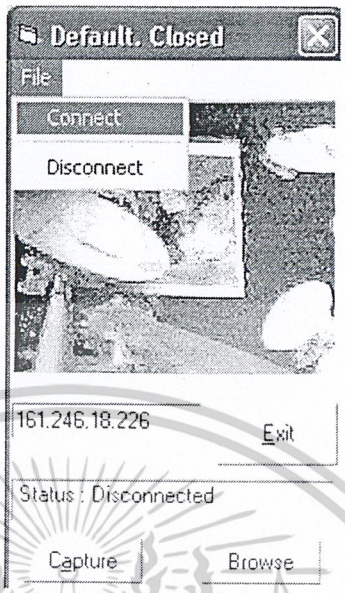
4.2.1.1 การรันแอพพลิเคชั่น

4.2.1.1.1ในการทดลองเราทำการเปิดโปรแกรมที่เขียนไว้เพื่อทดลองการส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่าย



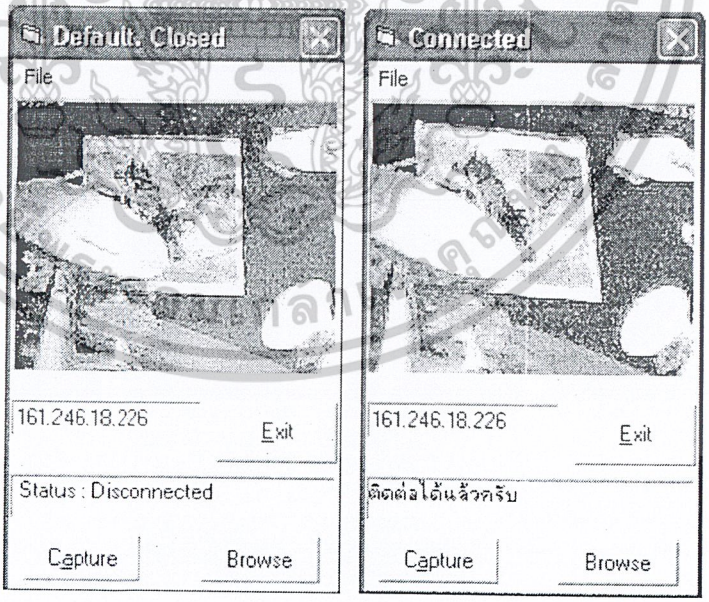
รูปที่ 4.1 สภาวะเริ่มต้นเมื่อทำการรันโปรแกรม

4.2.1.1.2 ทำการกดปุ่ม Connect ที่ Manu File



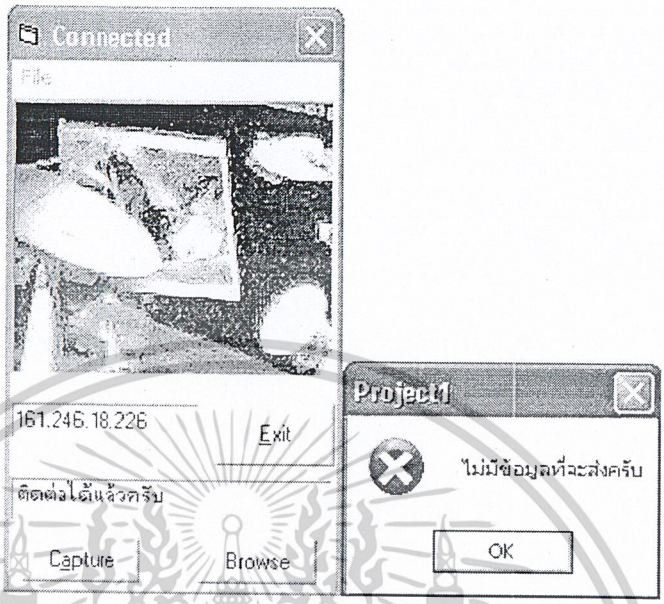
รูปที่ 4.2 การกดปุ่ม Connect ที่ Manu File

4.2.1.1.3 สังเกตที่ช่องStatus แสดงสถานะ Status: Disconnect ที่ช่อง Status แสดงว่าไม่สามารถติดต่อได้ แต่ถ้าที่ช่องStatus แสดงสถานะ "ติดต่อได้แล้วครับ" แสดงว่าสามารถติดต่อได้แล้ว

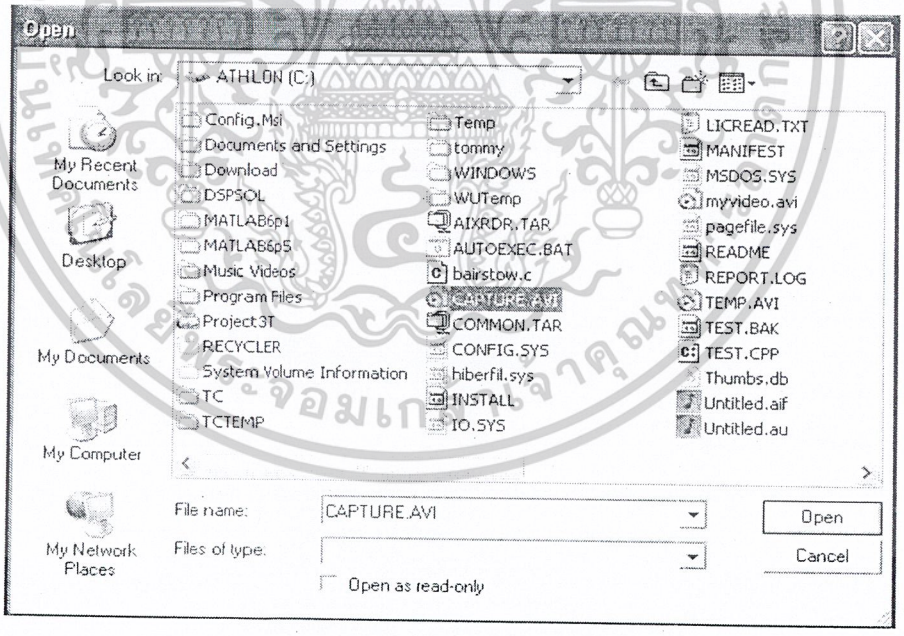


รูปที่ 4.3 สภาวะลักษณะของการติดต่อ

4.2.1.1.4 ทำการกดปุ่ม Capture จะแสดงการเตือนว่า “ไม่มีข้อมูลจะส่งครับ” เมื่อกดปุ่ม OK จะแสดง Common Dialog Box เพื่อให้เลือกข้อมูลที่จะส่ง ทำการเลือก C:\Capture.avi ให้เลือกข้อมูลที่จะส่ง



รูปที่ 4.4 สภาวะการเตือนไม่มีข้อมูลที่จะส่ง



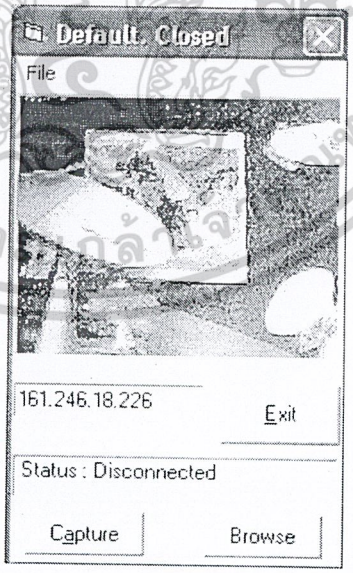
รูปที่ 4.5 สภาวะการการเลือกข้อมูลที่จะส่ง

4.2.1.1.5 ทำการ กดปุ่ม Capture อีกครั้งจะมีการส่งข้อมูลไปยังด้านไคลเอน

รูปที่ 4.6 สภาวะการส่งข้อมูลไปทางไคลเอน

4.2.1.2 สถานะต่างของ Winsock Control

4.2.1.2.1สถานะเริ่มแรกเมื่อทำการรันโปรแกรม จะแสดงเป็นสภาวะเริ่มต้น (Default Closed) ที่ชื่อของแท็บแคปชั่นที่อยู่ข้างบนซึ่งใช้เป็นที่แสดงสถานะของ Winsock Control ส่วนเลเบลข้างล่างจะแสดงสถานะที่เราทำการเขียนกำกับไว้ ในที่นี้แสดงว่าซ็อคเก็ตปิดไม่พร้อมทำการรับส่งข้อมูล

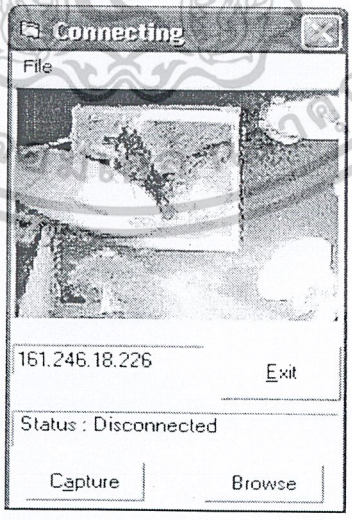


รูปที่ 4.7 สภาวะเริ่มต้นเมื่อทำการรันแอพพลิเคชั่น

4.2.1.2.2 เมื่อทำการกดปุ่ม Connect ในลักษณะที่ทำการติดต่อได้ ที่ช่องที่ใช้แสดงสถานะของ Winsock Control จะแสดงว่า Connected ส่วนในช่องเลเบลจะแสดงว่า "ติดต่อได้แล้วครับ" แสดงว่าได้ทำการเชื่อมต่อเส้นทาง พร้อมกับการรับส่งข้อมูล

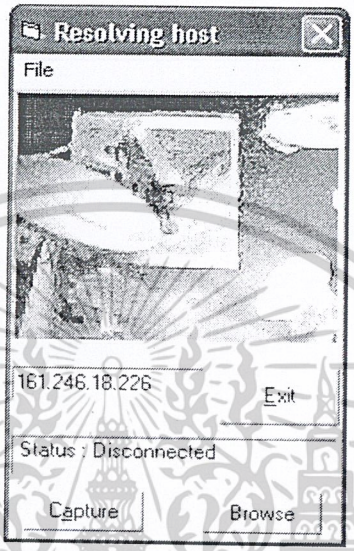
รูปที่ 4.8 สภาวะเมื่อทำการติดต่อได้

4.2.1.2.3 เมื่อการติดต่อไม่สามารถทำได้ ซึ่งมีอยู่หลายกรณีด้วยกัน โดยกรณีดังรูปที่4.9นี้ เป็นการติดต่อที่เกิดความผิดพลาดโดยในตอนแรกมันจะแสดงสถานะเป็นกำลังติดต่อ (Connecting) แล้วจึงแสดงความผิดพลาดในการเชื่อมต่อออกมา (Error) ที่มันแสดงเช่นนี้เพราะที่เครื่องโคลเอนไม่ได้ทำการเปิดเครื่อง

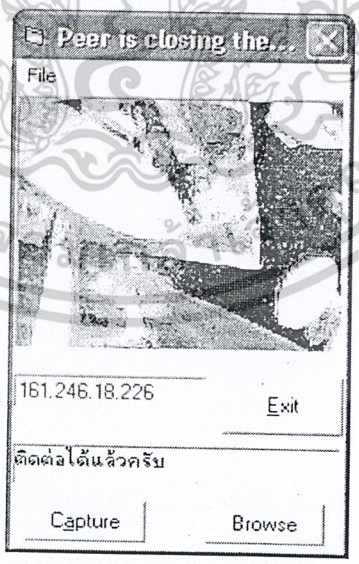


รูปที่ 4.9 สภาวะเมื่อกำลังทำการติดต่อ

4.2.1.2.4 เป็นความผิดพลาดอีกอย่างหนึ่งของการติดต่อ โดยที่แท็บแสดงสถานะข้างบนจะแสดงคำว่า “Resolving Host” หมายถึง กำลังทำการหาเครื่องตามที่หมายเลข IP address ทำการระบุไว้ แต่เป็นเพราะว่าในเครือข่ายของเราไม่มีเครื่องที่มีหมายเลข IP address ตามที่ระบุไว้ ดังนั้นเพื่อแก้ปัญหานี้เราต้องทำการระบุหมายเลข IP address ให้ตรงกับเครื่องที่เราต้องทำการติดต่อ



รูปที่ 4.10 สภาวะเมื่อกำลังทำการหาเครื่องตามหมายเลข IP address



รูปที่ 4.11 สภาวะเมื่อฝ่ายไคลเอนทำการปิดแอพพลิเคชั่น

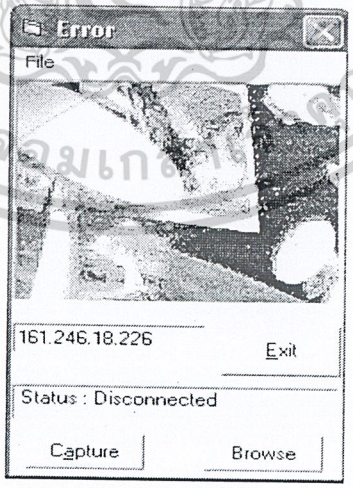
4.2.1.2.5 จากรูปที่ 4.11 เป็นความผิดพลาดที่เกิดขึ้นเมื่อทั้งสองฝั่งได้ทำการติดต่อกันได้แล้ว แต่ทางฝั่งไคลเอนได้ทำการปิดตัวแอพพลิเคชั่นลงไปก่อน ส่งผลให้ทางฝั่งเซิร์ฟเวอร์ฟ้องสถานะขึ้นมาว่า "Peer is closing the connection" เมื่อเกิดเหตุการณ์เช่นนี้เกิดขึ้นทำให้ฝั่งเซิร์ฟเวอร์เปลี่ยนสถานะในการติดต่อจากเดิม "Peer is closing the connection"

4.2.1.3 เป็นการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อและขณะที่เครื่องรับไม่ได้เปิดโปรแกรมรอการติดต่อ

4.2.1.3.1 ทำการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อจะเห็นที่ช่อง Status จะแสดงสถานะ "ติดต่อได้แล้วครับ"

รูปที่ 4.12 สถานะได้รับการติดต่อ

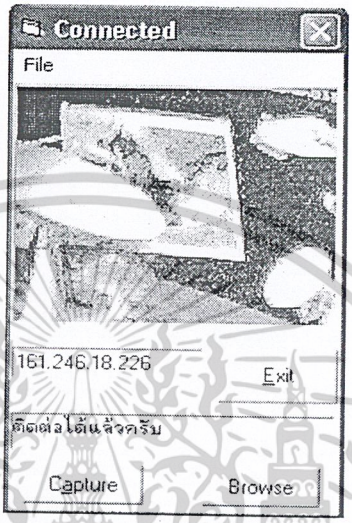
4.2.1.3.2 ทำการกดปุ่ม Connect ขณะที่เครื่องรับไม่ได้เปิดโปรแกรมรอการติดต่อ จะแสดงสถานะ "Status: Disconnected"



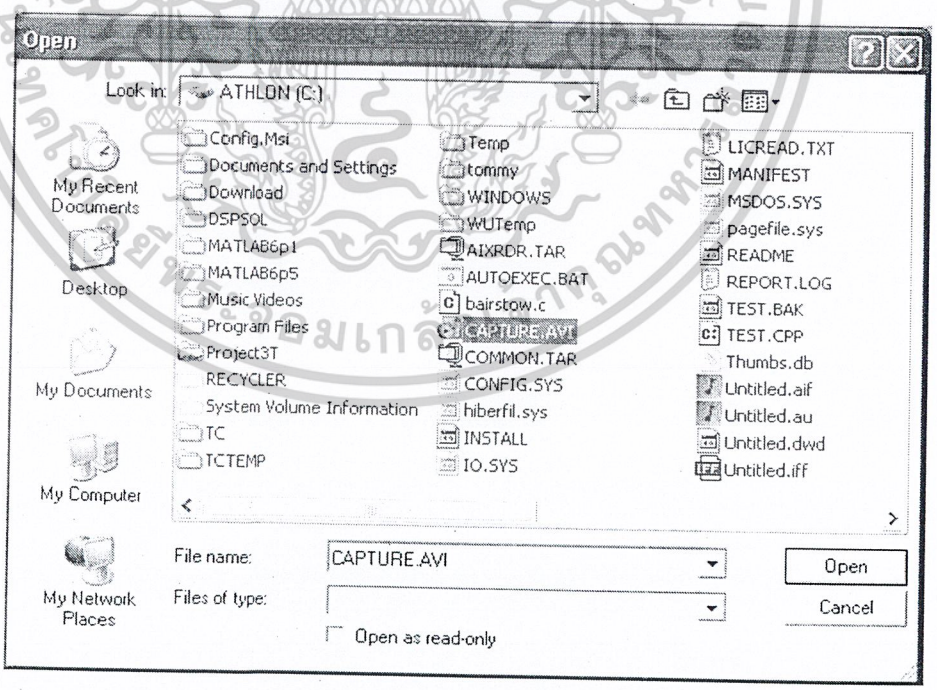
รูปที่ 4.13 สถานะไม่ได้รับการติดต่อ

4.2.1.4 เป็นการแสดงการกดปุ่ม Brows ก่อนการกดปุ่ม Capture ในสถานะการฝั่ง Server และฝั่ง Client ติดต่อเรียบร้อยแล้ว

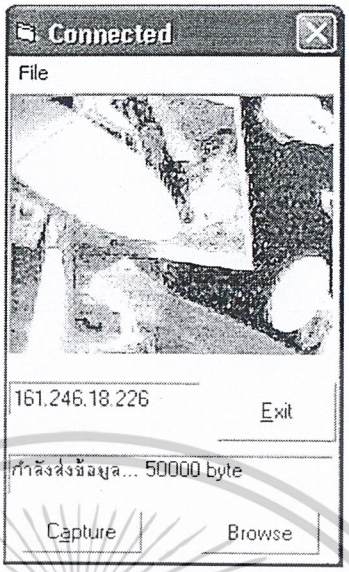
ทำการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อจะเห็นที่ช่อง Status จะแสดงสถานะ "ติดต่อได้แล้วครับ"กดปุ่ม Brows ทำการเลือก C:\capture.avi กดปุ่ม Open และทำการกดปุ่ม Capture จะไม่มีการเตือนต่าง ๆ ทำการกดปุ่ม Capture อีกครั้งจะมีการส่งข้อมูลไปฝั่งไคลเอนจนเสร็จ



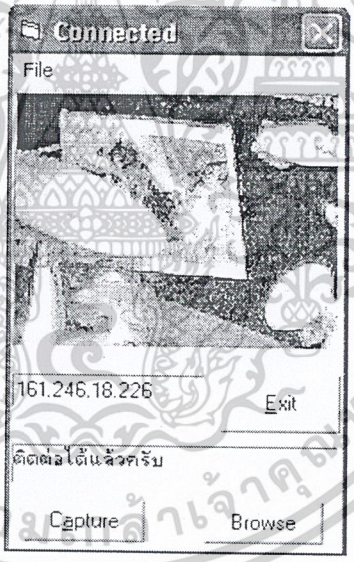
รูปที่ 4.14 สถานะได้รับการติดต่อ



รูปที่ 4.15 การทำการเลือกไฟล์ที่ต้องการจะส่ง



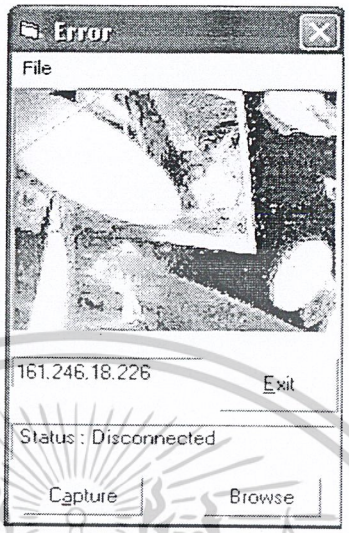
รูปที่ 4.16 การทำงานเมื่อมีการส่งข้อมูล



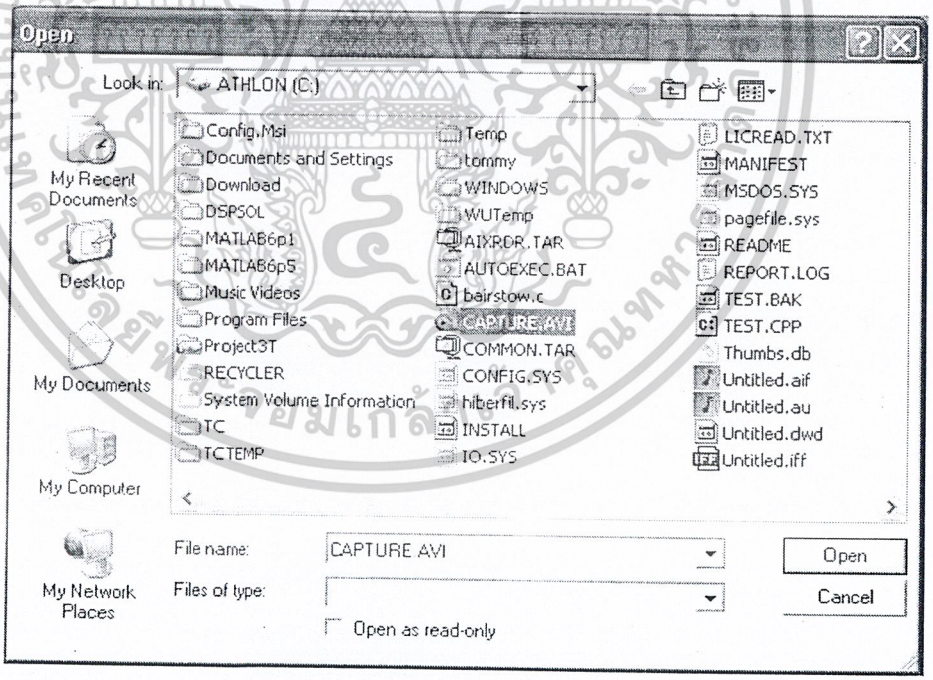
รูปที่ 4.17 การทำงานเมื่อมีการส่งข้อมูลเสร็จสิ้น

4.2.1.5 เป็นการแสดงการกดปุ่ม Brows ก่อนการกดปุ่ม Capture ในสถานะการฝั่ง Server และฝั่ง Client ยังไม่ได้ติดต่อซึ่งกันและกัน

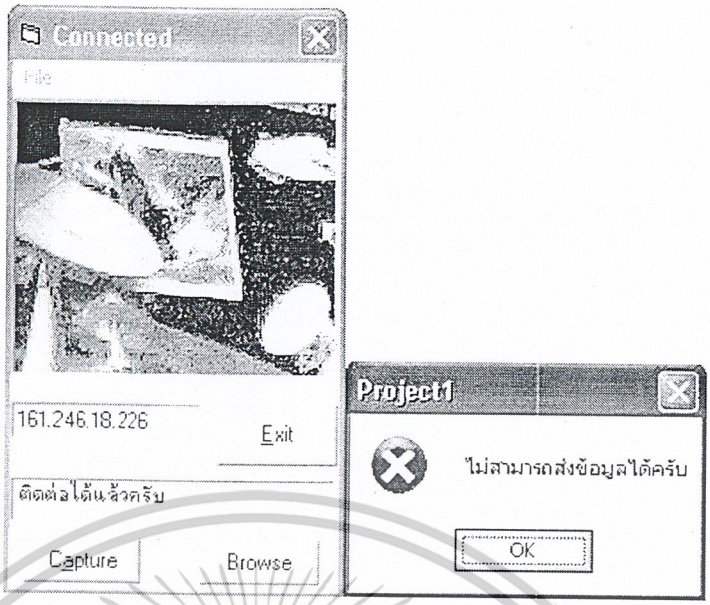
ทำการกดปุ่ม Connect ขณะที่เครื่องรับไม่ได้เปิดโปรแกรมรอการติดต่อจะแสดงสถานะ Status: Disconnected กดปุ่ม Brows ทำการเลือก C:\capture.avi กดปุ่ม Open และทำการกดปุ่ม Capture มีการเตือนว่า "ไม่สามารถส่งข้อมูลได้ครับ"



รูปที่ 4.18 สถานะไม่ได้รับการติดต่อ



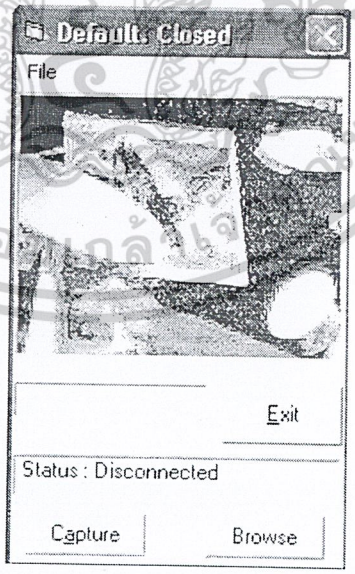
รูปที่ 4.19 การทำการเลือกไฟล์ที่ต้องการจะส่ง



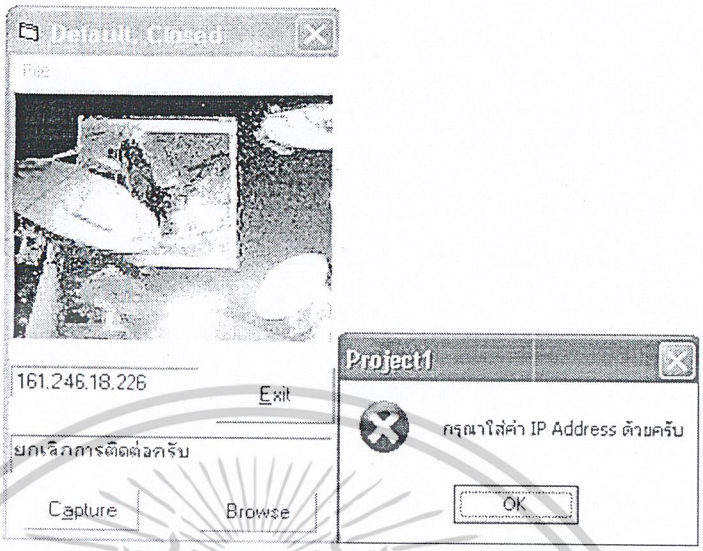
รูปที่ 4.20 สถานะไม่สามารถส่งข้อมูลได้

4.2.1.6 เป็นการแสดงทำการกดปุ่ม Connect ในขณะที่ช่อง IP ADDRESS ว่างอยู่

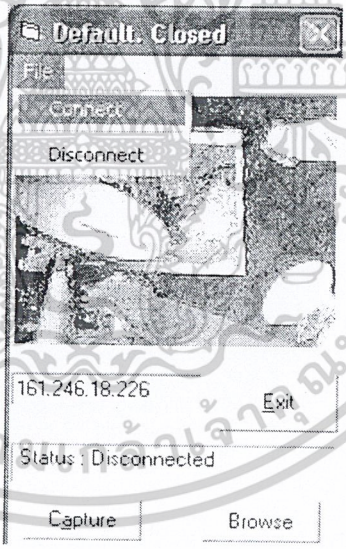
ทำการกดปุ่ม Connect ในขณะที่ช่อง IP ADDRESS ว่างอยู่ จะมีการเตือนว่า กรุณาใส่ค่า "IP Address ด้วยครับ" กดปุ่ม Ok และทำการใส่ค่า IP ADDRESS ที่ถูกต้องทำการกดปุ่ม Connect อีกครั้งจะสามารถติดต่อได้



รูปที่ 4.21 สถานะไม่ใส่ค่า IP ADDRESS



รูปที่ 4.22 สถานะเตือนให้ใส่ค่า IP ADDRESS

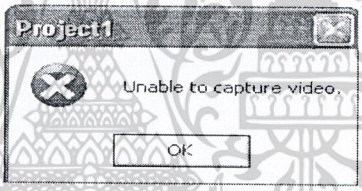


รูปที่ 4.23 การทำการกดปุ่ม Connect

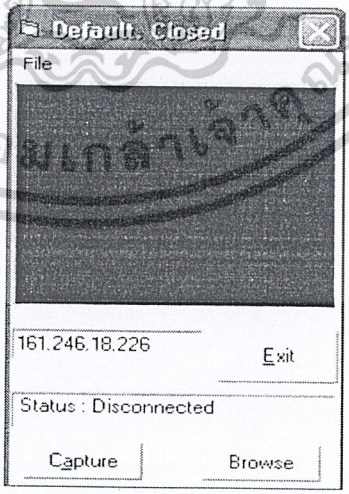
รูปที่ 4.24 สถานะเมื่อมีการติดต่อได้แล้ว

4.2.1.7 เป็นการแสดงเมื่อเปิดโปรแกรมแล้ว โปรแกรมไม่สามารถติดต่อกับ Diver ของกล้องได้

เมื่อเปิดโปรแกรมแล้ว โปรแกรมไม่สามารถติดต่อกับ Diver ของกล้องได้จะแสดงการเตือนว่า "Unable to capture Video"



รูปที่ 4.25 สถานการเตือนเมื่อติดต่อกับไดร์เวอร์กล้องไม่ได้

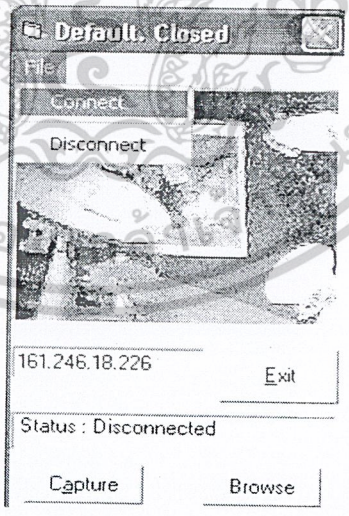


รูปที่ 4.26 สถานะเมื่อติดต่อกับไดร์เวอร์กล้องไม่ได้

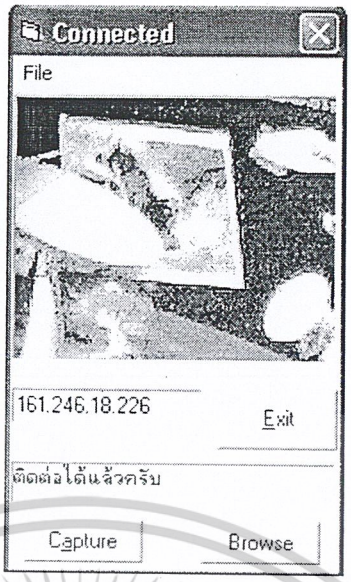
4.2.1.8 เป็นการแสดงให้เห็นเมื่อที่แท็บแคปชั่นของForm แสดงข้อความ"Peer is closing the connection" ทำการกดปุ่ม Connect ที่ฝั่งเซิร์พเวอร์ขณะที่ฝั่งไคลเอนทำการเปิดโปรแกรมรอการติดต่อ

ทำการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อจะเห็นที่ช่อง Status จะแสดงสถานะ "ติดต่อได้แล้วครับ" ถ้าเราทำการปิดโปรแกรมทางด้านรับ และโปรแกรมทางด้านส่งยังทำงานอยู่ ที่ด้านบนของโปรแกรมจะแสดงข้อความ "Peer is closing the connection"เราทำการเปิดโปรแกรมทางด้านรับอีกครั้ง และ ทำการกดปุ่ม Connect อีกครั้งจะสามารถติดต่อกันได้อีกครั้ง

รูปที่ 4.27 สภาวะเมื่อฝ่ายไคลเอนทำการปิดแอพพลิเคชั่น



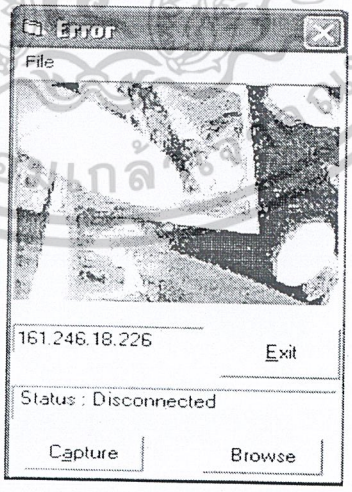
รูปที่ 4.28 การทำการกดปุ่ม Connect



รูปที่ 4.29 สถานะเมื่อมีการติดต่อได้แล้ว

4.2.1.9 เป็นการแสดงให้เห็นเมื่อมีการติดต่อกันสำเร็จแล้วแต่ปิดโปรแกรมฝั่งเซิร์พเวอร์ ขณะที่ฝั่งไคลเอนยังไม่ยกเลิกการติดต่อแล้วเปิดโปรแกรมฝั่งเซิร์พเวอร์ขึ้นมาอีกครั้ง และทำการติดต่ออีกครั้ง

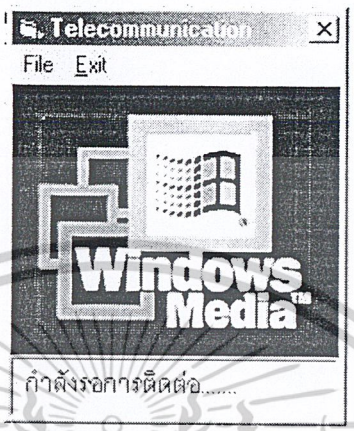
ทำการกดปุ่ม Connect ขณะที่เครื่องรับทำการเปิดโปรแกรมรอการติดต่อจะเห็นที่ช่อง Status จะแสดงสถานะ "ติดต่อได้แล้วครับ" ถ้าเราทำการปิดโปรแกรมทางด้านส่ง และโปรแกรมทางด้านรับยังคงทำงานอยู่ เราทำการเปิดโปรแกรมทางด้านส่งอีกครั้ง และ ทำการกดปุ่มConnect อีกครั้งจะไม่สามารถทำการติดต่อได้ ด้านบนของโปรแกรมจะแสดง Error



รูปที่ 4.30 สถานะไม่ได้รับการติดต่อ

4.2.2 ด้านรับ หรือ Client

4.2.2.1. ในตอนแรกทำการเปิดแอพพลิเคชั่นเพื่อรอการติดต่อ โดยในสภาวะแรกเป็นสภาวะโหลดซึ่งมีรูปร่างหน้าตาดังรูปที่ 4.31



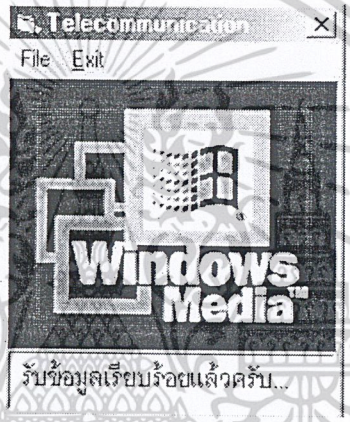
รูปที่ 4.31 สภาวะรอการติดต่อจากเครื่องเซิร์ฟเวอร์

4.2.2.2 เมื่อมีการติดต่อมาจากเครื่องเซิร์ฟเวอร์จะได้สภาวะ "ติดต่อได้แล้ว... (Connected)" ซึ่งแสดงได้ดังรูปที่ 4.32

รูปที่ 4.32 สภาวะเมื่อมีการติดต่อมาจากเครื่องเซิร์ฟเวอร์

4.2.2.3 มีการส่งข้อมูลจากฝั่งเซิร์ฟเวอร์ ฝั่งไคลเอนจะทำการรับข้อมูลจนเสร็จในหนึ่งไฟล์แล้วจึงทำการเล่นไฟล์ที่รับมาได้โดยแสดงให้เห็นที่จอแสดงผลดังรูปที่ 4.34

รูปที่ 4.33 สภาวะกำลังรับข้อมูล



รูปที่ 4.34 สภาวะเมื่อทำการรับข้อมูลเสร็จ

รูปที่ 4.35 สภาวะเมื่อทำการเล่นภาพที่รับมาได้

บทที่5
บทสรุปและบทวิจารณ์

บทสรุปและบทวิจารณ์

ในระบบการส่งภาพเคลื่อนไหวผ่านเครือข่ายนั้นมีความต้องการมีความต้องการทรัพยากรที่จะนำมาใช้งานเป็นจำนวนมาก ไม่ว่าจะเป็นหน่วยประมวลผลหรือซีพียูที่มีความสามารถในการประมวลผลสูง หน่วยความจำหรือบัฟเฟอร์ที่ใช้เป็นพื้นที่ในการจัดเก็บข้อมูล รวมทั้งกล้อง Web cam ที่มีประสิทธิภาพในการทำงานในการจับภาพ (Capture)

นอกจากความจำเป็นทางด้านอุปกรณ์ฮาร์ดแวร์แล้ว ตัวซอฟต์แวร์เองก็เป็นส่วนสำคัญอีกส่วนหนึ่ง โดยเราควรมีซอฟต์แวร์ที่สนับสนุนงานทางด้านการพัฒนาระบบดังกล่าวนี้ด้วย ซอฟต์แวร์ที่ว่านี้มีทั้งซอฟต์แวร์ที่มาพร้อมกับกล้อง Web cam และซอฟต์แวร์ที่ผลิตออกมาต่างหาก เช่นซอฟต์แวร์ที่มาพร้อมกับกล้องก็ควรมีซอฟต์แวร์ที่ใช้ในการจับภาพ บันทึกภาพ เล่นภาพเคลื่อนไหว นอกจากนี้ก็เป็นซอฟต์แวร์ที่ใช้ในการพัฒนาแอพพลิเคชั่น ซึ่งมีความสามารถในการสร้างเครื่องมือหน้าตาได้อย่างรวดเร็ว

จะเห็นว่าการเลือกใช้อุปกรณ์ก็เป็นข้อสำคัญอีกอย่างหนึ่ง อุปกรณ์ที่ใช้เป็นแหล่งในการนำภาพเข้ามานอกจากเราจะต้องทำการพิจารณาควบคู่ไปกับประสิทธิภาพและการทำงานของตัวกล้องแล้ว เรายังต้องพิจารณาถึงตัวซอฟต์แวร์ที่มาพร้อมกับกล้องด้วย เพื่อให้การทำงานของเราเป็นไปอย่างมีประสิทธิภาพ แต่ถึงแม้ว่าจะมีอุปกรณ์ทางด้านฮาร์ดแวร์และซอฟต์แวร์ที่พร้อมแล้ว ก็ยังพบปัญหาอื่นๆ อีกหลายประการ โดยปัญหาที่เกิดขึ้นคือ

- ข้อมูลที่เราใช้ในการทำการส่งผ่านเครือข่าย เป็นข้อมูลภาพที่มีขนาดค่อนข้างใหญ่ การที่เราจะทำให้การส่งและการรับระหว่างเครื่องเซิร์ฟเวอร์และไคลเอนนั้นสัมพันธ์กันขึ้นอยู่กับการประมาณชิ้นของข้อมูลที่จะทำการส่ง เพื่อให้ทางด้านรับมีความเร็วพอที่จะรับได้อย่างสัมพันธ์กับทางด้านส่ง
- เครื่องที่ใช้ในการทดลองมีความสามารถในการประมวลผลที่แตกต่างกัน ให้ประสิทธิภาพในการดำเนินงานไม่เท่ากัน ในการทดลองจึงต้องทำการเผื่อเวลาสำหรับการประมวลผลของเครื่องที่ทำงานช้ากว่าไว้ด้วย
- ปัญหาที่เกิดจากข้อจำกัดในการเขียนโปรแกรม กล่าวคือในการแสดงภาพแต่ละภาพนั้นจำเป็นต้องมีการเขียนช่วงของการหน่วงเวลา เพื่อให้การโหลดภาพเสร็จก่อนที่ภาพต่อไปจะมาถึง ปัญหาการหน่วงเวลานี้คือต้องใช้คำสั่ง DoEvent() ซึ่งเป็นฟังก์ชันที่อยู่ใน Visual Basic ซึ่งในการทดลองโดยการลองใช้ลูปเพียงอย่างเดียวก็ไม่สามารถทำงานได้ดีกว่าในวิธีแรก
- ปัญหาในการใช้เครือข่าย สามารถทำการส่งข้อมูลที่ด้วยขนาดที่จำกัดตามความเร็วของการส่งผ่านข้อมูลของระบบเครือข่าย

โครงงานฉบับนี้ เป็นการศึกษาเกี่ยวกับการส่งผ่านข้อมูลที่เป็นภาพเคลื่อนไหวผ่านระบบเครือข่าย โดยอาศัยโปรแกรม Visual Basic ในการพัฒนาแอพพลิเคชั่นของเรา โดยการทำงานของ

แอพพลิเคชั่นจะเป็นลักษณะการแคบเจอร์ข้อมูลจากกล้อง web cam เพื่อให้ได้เป็นไฟล์หนึ่งไฟล์แล้วทำการส่งไป จากนั้นทางด้านรับทำการรับไฟล์นี้แล้วทำการเล่นต่อไป และแสดงผลออกที่หน้าจอแสดงผล โดยการส่งผ่านข้อมูลนี้ต้องระวังในเรื่องของเวลาที่ใช้ในการส่งไฟล์ และเวลาที่ใช้ในการเล่นไฟล์ หากเวลาไม่สัมพันธ์ก็จะทำให้เครื่องแฮงค์ได้ ส่วนที่สำคัญอีกส่วนหนึ่งคือการเรียกใช้ฟังก์ชัน API ด้วยความระมัดระวังการกำหนดชนิดของอากิวเมนต์ที่ใช้ในการผ่านค่าต่างๆ ควรจะแน่ใจว่าเป็นที่ถูกต้องแล้ว เพราะการเรียกใช้ฟังก์ชั่น API คือการยืมเครื่องมือต่างๆ ในระบบปฏิบัติการใน Windows มาใช้ ซึ่งไฟล์บางตัวที่ยืมมานั้นอาจมีผลกระทบโดยตรงกับการทำงานของ Windows ทำให้ระบบปฏิบัติการของเราทำงานขัดข้องได้

แนวทางในการพัฒนา อาจทำได้โดยการส่งแคปเจอร์เป็นคลิปวีดีโอเล็กๆ แล้วจึงทำการส่งไป เพื่อให้ภาพที่ปรากฏที่หน้าจอแสดงผลของฝั่งไคลเอนมีความต่อเนื่องกันมากขึ้น หรืออาจทำการแคบเจอร์ภาพแบบต่อเนื่อง ซึ่งการแคบเจอร์ในแต่ละครั้งจะต้องออกมาเป็นข้อมูลที่พร้อมจะทำการส่งได้เลย เพื่อให้การส่งเป็นไปแบบต่อเนื่องเหมือนการแคบเจอร์ โดยการนำการบีบอัดข้อมูลมาช่วย เมื่อเราทำได้เช่นนี้เราก็สามารถกำหนด frame rate ได้ และสามารถได้ภาพที่มีความต่อเนื่องกันมากขึ้น

หนังสืออ้างอิง

1. สัจจะ จรัสรุ่งรวีวร คู่มือการเขียนโปรแกรมและการใช้งาน Visual Basic 6.0 กรุงเทพฯ : อินโฟเพรส 2544
2. ฉันทวุฒิ พืชผลม, พิชิต สันติกุลานนท์ คู่มือเรียน Visual Basic 6.0 : โปรวิชั่น 2544

ภาคผนวก ก

ด้านส่ง (Server)

```
Form1
Option Explicit
Const WS_VISIBLE = &H10000000
Const WS_CHILD = &H40000000
Const WM_USER = 1024
Const WM_CAP_EDIT_COPY = WM_USER + 30
Const WM_CAP_DRIVER_CONNECT = WM_USER + 10
Const WM_CAP_SET_PREVIEWRATE = WM_USER + 52
Const WM_CAP_SET_PREVIEW = WM_USER + 50
Const PREVIEWRATE = 30
Private Declare Function SendMessage Lib "user32"Alias "SendMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal
lParam As Long) As Long
Private Declare Function capCreateCaptureWindow Lib "avicap32.dll"
Alias "capCreateCaptureWindowA" (ByVal a As String, ByVal b As Long,
ByVal c As Integer, ByVal d As Integer, ByVal e As Integer, ByVal f
As Integer, ByVal g As Long, ByVal h As Integer) As Long
Dim hwndc As Long
Dim temp As String
Private m_LocalFile As String
Private m_RemoteFile As String
Public Property Get capwnd() As Long
  capwnd = hwndc
End Property

Private Sub cmdBrowse_Click()
  cdopen.ShowOpen
  If vbOK Then txtFileName = cdopen.FileName
End Sub

Private Sub line()
Dim FName_Only As String
  capCaptureSingleFrame (Form1.capwnd)
  Call capCaptureSingleFrameClose(Form1.capwnd)
     FName_Only = GetFileName(txtFileName)
     SendFile1 FName_Only
  Call capCaptureSingleFrameOpen(Form1.capwnd)
End Sub

Private Sub cmdCapture_Click()
Dim FName_Only As String
Dim i As Integer
capCaptureSingleFrame (Form1.capwnd)
Call capCaptureSingleFrameClose(Form1.capwnd)
  If txtFileName = "" Then
    MsgBox "ไม่มีข้อมูลที่จะส่งครับ", vbCritical
    cdopen.ShowOpen
    If vbOK Then txtFileName = cdopen.FileName
     Else
       If Winsock1.State = sckClosed Then
       MsgBox "ไม่สามารถส่งข้อมูลได้ครับ", vbCritical
       Else
         If lblStatus.Caption = "ติดต่อได้แล้วครับ" Then
```

```
        FName_Only = GetFileName(txtFileName)
        SendFile FName_Only
       Else
        MsgBox "ไม่สามารถส่งข้อมูลได้ครับ", vbCritical
       End If
      End If
  End If
Call capCaptureSingleFrameOpen(Form1.capwnd)
End Sub

Private Sub cmdexit_Click()
  Winsock1.Close
  Winsock2.Close
   bReplied = False
  Timer1.Enabled = False
  Unload Me
End Sub

Private Sub Form_Load()
On Error GoTo handler:
    hwndc = capCreateCaptureWindow("CaptureWindow", ws_child Or
ws_visible, 0, 0, PichWnd.Width, PichWnd.Height, PichWnd.hwnd, 0)
    If (hwndc <> 0)Then
        temp = SendMessage(hwndc, wm_cap_driver_connect, 0, 0)
        temp = SendMessage(hwndc, wm_cap_set_preview, 1, 0)
        temp = SendMessage(hwndc, WM_CAP_SET_PREVIEWRATE,
               PREVIEWRATE, 0)
        temp = SendMessage(Me.hwnd, WM_CAP_EDIT_COPY, 1, 0)
    Else
        MsgBox "Unable to capture video.", vbCritical
    End If
capFileGetCaptureFile (Form1.capwnd)
Call capCaptureSingleFrameOpen(Form1.capwnd)
Exit Sub
  Winsock1.Close
  Winsock2.Close
handler:
End Sub

Private Sub mnuconnect_Click()
Dim Tekst As String
  If Winsock1.State = sckConnected Then
  Winsock1.Close
  Exit Sub
  Else
   If txt1.Text = "" Then
   MsgBox "กรุณาใส่ค่า IP Address ด้วยครับ"
   Exit Sub
   Else
   With Winsock1
     .Close
     .LocalPort = 610
     .RemotePort = 600
     .Connect txt1.Text
```

```
      bReplied = True
    End With
     Timer1.Enabled = True
   End If
End If
   If Winsock2.State = sckConnected Then
    Winsock2.Close
  Exit Sub
  Else
    With Winsock2
      .Close
      .LocalPort = 611
      .RemotePort = 601
      .Connect txt1.Text
      bReplied = True
    End With
    Timer1.Enabled = True
 End If
End Sub

Private Sub mnudisconnect_Click()
  Winsock1.Close
  Winsock2.Close
   bReplied = False
  Timer1.Enabled = True
  lblStatus = "ยกเลิกการติดต่อครับ"
End Sub

Private Sub Timer1_Timer()
  Select Case Winsock1.State
    Case sckClosed: Me.Caption = "Default. Closed"
    Case sckOpen: Me.Caption = "Open"
    Case sckListening: Me.Caption = "Listening"
    Case sckConnectionPending: Me.Caption = "Connection pending"
    Case sckResolvingHost: Me.Caption = "Resolving host"
    Case sckHostResolved: Me.Caption = "Host resolved"
    Case sckConnecting: Me.Caption = "Connecting"
    Case sckConnected: Me.Caption = "Connected"
    Case sckClosing: Me.Caption = "Peer is closing the connection"
    Case sckError: Me.Caption = "Error"
   End Select
End Sub

Private Sub winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Command As String, NewArrival As String, Data As String
Static DataCnt As Long
Dim i As Integer
  Winsock1.GetData NewArrival, vbString
  If Len(NewArrival) < MAX_CHUNK Then
      For i = 0 To Len(NewArrival)
          If Mid(NewArrival, i + 1, 1)="," Then
           Command = Left(NewArrival, i)
           Data = Right(NewArrival, Len(NewArrival) - (i + 1))
           Exit For
        End If
```

```
    Next
  End If
  Select Case Command
    Case "Accepted"
      bReplied = True
      lblStatus = "ติดต่อได้แล้วครับ"
    Case "ServerClosed"
      Form_Load
      Winsock1.Close
    Case "OpenFile"
      Call cmdCapture_Click
    Case Else
  Exit Sub
  End Select
End Sub

Private Sub winsock2_DataArrival(ByVal bytesTotal As Long)
Dim Command As String, NewArrival As String, Data As String
Static DataCnt As Long
Dim i As Integer
  Winsock2.GetData NewArrival, vbString
  If Len(NewArrival) < MAX_CHUNK Then
    For i = 0 To Len(NewArrival)
      If Mid(NewArrival, i + 1, 1)="," Then
        Command = Left(NewArrival, i)
        Data = Right(NewArrival, Len(NewArrival) - (i + 1))
        Exit For
      End If
    Next
  End If
  Select Case Command
    Case "Accepted"
      bReplied = True
    Case "ServerClosed"
      Form_Load
      Winsock2.Close
    Case "OpenFile"
      Call line1
    Case Else
      Exit Sub
  End Select
End Sub
```

```
Module1
Option Explicit
Private Declare Function SendMessageAsLong Lib "user32" Alias _
"SendMessageA" _(ByVal hwnd As Long, _
                        ByVal wMsg As Long, _
                        ByVal wParam As Long, _
                        ByVal lParam As Long) As Long
Declare Function SendMessage Lib "user32" Alias "SendMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Integer,
ByVal lParam As Long) As Long
Private Declare Function SendMessageAsString Lib "user32" Alias
"SendMessageA" _(ByVal hwnd As Long, _
                        ByVal wMsg As Long, _
                        ByVal wParam As Long, _
                        ByVal lParam As String) As Long
Public Const WM_USER = &H400
Public Const WM_CAP_START = WM_USER
Public Const WM_CAP_DLG_VIDEOFORMAT = WM_CAP_START + 41
Public Const WM_CAP_SINGLE_FRAME_OPEN As Long = WM_CAP_START + 70
Public Const WM_CAP_SINGLE_FRAME As Long = WM_CAP_START + 72
Public Const WM_CAP_SINGLE_FRAME_CLOSE = WM_CAP_START + 71
Public Const WM_CAP_FILE_GET_CAPTURE_FILE = WM_CAP_START + 21
Function capDlgVideoFormat(ByVal hwndc As Long) As Boolean
   capDlgVideoFormat = SendMessage(hwndc, WM_CAP_DLG_VIDEOFORMAT, 0,
0)
End Function

Function capCaptureSingleFrame(ByVal hwndc As Long) As Boolean
   capCaptureSingleFrame = SendMessageAsLong(hwndc,
WM_CAP_SINGLE_FRAME, 0&, 0&)
End Function

Function capCaptureSingleFrameOpen(ByVal hwndc As Long) As Boolean
   capCaptureSingleFrameOpen = SendMessageAsLong(hwndc,
WM_CAP_SINGLE_FRAME_OPEN, 0&, 0&)
End Function

Function capFileGetCaptureFile(ByVal hwndc As Long) As String
Dim szBuffer As String
Dim retVal As Boolean
   szBuffer = String$(128, 0)
   retVal = SendMessageAsString(hwndc, WM_CAP_FILE_GET_CAPTURE_FILE,
128, szBuffer)
   If retVal Then
        capFileGetCaptureFile = Left$(szBuffer, InStr(szBuffer,
vbNullChar) - 1)
    End If
End Function

Function capCaptureSingleFrameClose(ByVal hwndc As Long) As Boolean
   capCaptureSingleFrameClose = SendMessage(hwndc,
WM_CAP_SINGLE_FRAME_CLOSE, 0, 0)
End Function
```

Module2

```
Option Explicit
Public Declare Function GetTickCount Lib "kernel32"()As Long
Public Const MAX_CHUNK = 50000
Public bReplied As Boolean
Sub SendFile(Fname As String)
Dim DataChunk As String
Dim Passes As Long
Dim IRead As Long, ILen As Long, IThisRead As Long, ILastRead As
Long
SendData "OpenFile," & Fname '
Pause 200
Open Fname For Binary As #1
   Do While Not EOF(1)
      Passes = Passes + 1
      DataChunk = Input(MAX_CHUNK, #1)
      SendData DataChunk
      Form1.lblStatus = "กำลังส่งข้อมูล... " & (MAX_CHUNK * Passes) & " bytes"
      Pause 200
   Loop
   SendData "CloseFile,"
   Form1.lblStatus = "ติดต่อได้แล้วครับ"
 Close #1
 Passes = 0
End Sub

Function GetFileName(Fname As String) As String
Dim i As Integer
Dim TempStr As String
 For i = 1 To Len(Fname)
    TempStr = Right(Fname, i)
    If Left(TempStr, 1)="\" Then
       GetFileName = Mid(TempStr, 2, Len(TempStr))
       Exit Function
    End If
 Next i
End Function

Function SendData(sData As String) As Boolean
Dim TimeOut As Long
On Error GoTo ErrH
bReplied = False
Form1.Winsock1.SendData sData
 Do Until (Form1.Winsock1.State = 0)Or (TimeOut < 100000)
    DoEvents
    TimeOut = TimeOut + 1
    If TimeOut > 100000 Then Exit Do
 Loop
 SendData = True
Exit Function
ErrH:
    SendData = False
    MsgBox Err.Description, 16, "Error #" & Err.Number
    Form1.lblStatus = "Disconnected."
```

```
End Function

Sub Pause(HowLong As Long)
Dim Tick As Long
 Tick = GetTickCount()
 Do
    DoEvents
 Loop Until Tick + HowLong < GetTickCount
End Sub

Sub SendFile1(Fname As String)
Dim DataChunk As String
Dim Passes As Long
Dim IRead As Long, ILen As Long, IThisRead As Long, ILastRead As
Long
 SendData "OpenFile," & Fname '
 Pause 200
 Open Fname For Binary As #1
  Do While Not EOF(1)
     Passes = Passes + 1
     DataChunk = Input(MAX_CHUNK, #1)
     SendData DataChunk
     Form1.lblStatus = "กำลังส่งข้อมูล..." & (MAX_CHUNK * Passes) & " bytes"
     Pause 200
  Loop
  SendData1"CloseFile,"
  Form1.lblStatus = "ดิสต่อได้แล้วครับ"
 Close #1
Passes = 0
End Sub

Function SendData1(sData As String) As Boolean
Dim TimeOut As Long
On Error GoTo ErrH
bReplied = False
Form1.Winsock2.SendData sData
Do Until (Form1.Winsock2.State = 0)Or (TimeOut < 100000)
  DoEvents
  TimeOut = TimeOut + 1
  If TimeOut > 100000 Then Exit Do
Loop
SendData1=True
Exit Function

ErrH:
    SendData1=False
    MsgBox Err.Description, 16, "Error #" & Err.Number
    Form1.lblStatus = "Disconnected."
End Function
```

ภาคผนวก ข

ภาครับ (Client)

```
Form 1
Option Explicit
Private Sub cmdquit_Click()
  tcpServer.Close
  Winsock1.Close
  Unload Me
End
End Sub

Private Sub Form_Load()
  With tcpServer
    .LocalPort = 600
    .RemotePort = 610
    .Listen
  End With
  With Winsock1
    .LocalPort = 601
    .RemotePort = 611
    .Listen
  End With
  bInConnection = False
  lblStatus = "   กำลังรอการติดต่อ....."
End Sub

Private Sub mnudisconnect_Click()
  tcpServer.Close
  Winsock1.Close
  With tcpServer
    .LocalPort = 600
    .RemotePort = 610
    .Listen
  End With
  With Winsock1
    .LocalPort = 601
    .RemotePort = 611
    .Listen
  End With
   bInConnection = False
   lblStatus = "   กำลังรอการติดต่อ....."
End Sub

Private Sub tcpServer_Close()
  If tcpServer.State <> sckClosed Then tcpServer.Close
  Form_Load
End Sub

Private Sub winsock1_Close()
  If Winsock1.State <> sckClosed Then Winsock1.Close
  Form_Load
End Sub

Private Sub tcpServer_ConnectionRequest(ByVal requestID As Long)
On Error GoTo IDERROR
  If tcpServer.State <> sckClosed Then tcpServer.Close
  tcpServer.Accept requestID
  bInConnection = True
```

```
 lblStatus = "ติดต่อได้แล้ว...(Connected.)"
 SendData "Accepted,"
Exit Sub
IDERROR:
  MsgBox Err.Description, vbCritical
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
Dim Command As String, NewArrival As String, Data As String
Static DataCnt As Long
Dim i As Integer
tcpServer.GetData NewArrival, vbString
 If Len(NewArrival) < MAX_CHUNK Then
   For i = 0 To Len(NewArrival)
     If Mid(NewArrival$, i + 1, 1) = "," Then
       Command = Left(NewArrival, i)
       Data$ = Right(NewArrival, Len(NewArrival) - (i + 1))
       Exit For
    End If
   Next
 End If
 Select Case Command$
   Case "OpenFile"
     Dim Fname As String
     Fname$ = App.Path & "\" & Data$
     Open Fname$ For Binary As #1
     lblStatus = "เปิด File .... " & Data$
   Case "CloseFile"
     Close #1
     lblStatus = "รับข้อมูลเรียบร้อยแล้วครับ.."
     lblStatus = "ติดต่อได้แล้ว...(Connected)"
     DataCnt = 0
     SendData1 "OpenFile,"
   Dim Myfso As New FileSystemObject, Tmpfil1
   Set Tmpfil1 = Myfso.GetFile("C:\1\New Folder\capture.AVI")
   Tmpfil1.Copy ("C:\1\New Folder\2\TEMP2.AVI")
   MPlayer1.FileName = "C:\1\New Folder\2\TEMP2.AVI"
   MPlayer1.Play
   Case Else
       Put #1, , NewArrival
       DataCnt = DataCnt + 1
       lblStatus = "กำลังรับข้อมูล..." & MAX_CHUNK * DataCnt & " Bytes"
 End Select
End Sub

Sub PlayAVIPictureBox(FileName As String, ByVal Window As
PictureBox)
Dim RetVal As Long
Dim CommandString As String
Dim ShortFileName As String * 260
Dim deviceIsOpen As Boolean
 RetVal = GetShortPathName(FileName, ShortFileName,
Len(ShortFileName))
 FileName = Left$(ShortFileName, RetVal)
 CommandString = "Open " & FileName & " type AVIVideo alias AVIFile
parent " _
& CStr(Window.hWnd) & " style " & CStr(WS_CHILD)
```

```
 RetVal = mciSendString(CommandString, vbNullString, 0, 0&)
 If RetVal Then GoTo error
 deviceIsOpen = True
 CommandString = "Play AVIFile wait"
 RetVal = mciSendString(CommandString, vbNullString, 0, 0&)
 If RetVal <> 0 Then GoTo error
 CommandString = "Close AVIFile"
 RetVal = mciSendString(CommandString, vbNullString, 0, 0&)
 If RetVal <> 0 Then GoTo error
Exit Sub
error:
Dim ErrorString As String
 ErrorString = Space$(256)
 mciGetErrorString RetVal, ErrorString, Len(ErrorString)
 ErrorString = Left$(ErrorString, InStr(ErrorString, vbNullChar) -
1)
 If deviceIsOpen Then
  CommandString = "Close AVIFile"
  mciSendString CommandString, vbNullString, 0, 0&
 End If
Err.Raise 999, , ErrorString
End Sub

Private Sub winsock1_ConnectionRequest(ByVal requestID As Long)
On Error GoTo IDERROR
 If Winsock1.State <> sckClosed Then Winsock1.Close
  Winsock1.Accept requestID
  bInConnection = True
Exit Sub
IDERROR:
   MsgBox Err.Description, vbCritical
End Sub

Private Sub winsock1_DataArrival(ByVal bytesTotal As Long)
Dim Command As String, NewArrival As String, Data As String
Static DataCnt As Long
Dim i As Integer
 Winsock1.GetData NewArrival, vbString
 If Len(NewArrival) < MAX_CHUNK Then
    For i = 0 To Len(NewArrival)
       If Mid(NewArrival$, i + 1, 1) = "," Then
          Command = Left(NewArrival, i)
          Data$ = Right(NewArrival, Len(NewArrival) - (i + 1))
          Exit For
       End If
    Next
 End If
 Select Case Command$
    Case "OpenFile"
      Dim Fname As String
      Fname$ = App.Path & "\" & Data$
      Open Fname$ For Binary As #1
      lblStatus = "เปิดFile .... " & Data$
    Case "CloseFile"
      Close #1
      lblStatus = "รับข้อมูลเรียบร้อยแล้วครับ.."
      lblStatus = "ติดต่อได้แล้ว...(Connected)"
```

```
        DataCnt = 0
        SendData "OpenFile,"
        Dim Myfso As New FileSystemObject, Tmpfil1
        Set Tmpfil1 = Myfso.GetFile("C:\1\New Folder\capture.AVI")
        Tmpfil1.Copy ("C:\1\New Folder\2\TEMP1.AVI")
        MPlayer1.FileName = "C:\1\New Folder\2\TEMP1.AVI"
        MPlayer1.Play
      Case Else
        Put #1, , NewArrival
        DataCnt = DataCnt + 1
        lblStatus = "กำลังรับข้อมูล..." & MAX_CHUNK * DataCnt & " Bytes"
  End Select
End Sub
```

Module1

```
Option Explicit
Declare Function mciSendString Lib "winmm" Alias "mciSendStringA"
(ByVal lpstrCommand As String, ByVal lpstrReturnString As String,
ByVal uReturnLength As Long, ByVal hwndCallback As Long) As Long
Declare Function mciGetErrorString Lib "winmm" Alias
"mciGetErrorStringA" (ByVal dwError As Long, ByVal lpstrBuffer As
String, ByVal uLength As Long) As Long
Declare Function GetShortPathName Lib "kernel32" Alias
"GetShortPathNameA" (ByVal lpszLongPath As String, ByVal
lpszShortPath As String, ByVal cchBuffer As Long) As Long
Public Const WS_CHILD = &H40000000
Public Declare Function GetTickCount Lib "kernel32" () As Long
Public Const MAX_CHUNK = 50000
Public bInConnection As Boolean '
Function GetFileName(Fname As String) As String
Dim i As Integer
Dim TempStr As String
 For i = 1 To Len(Fname)
    TempStr = Right(Fname, i)
    If Left(TempStr, 1) = "\" Then
      GetFileName = Mid(TempStr, 2, Len(TempStr))
      Exit Function
    End If
 Next i
End Function

Sub Pause(HowLong As Long)
Dim Tick As Long
 Tick = GetTickCount()
 Do
   DoEvents
 Loop Until Tick + HowLong < GetTickCount
End Sub

Function SendData(sData As String) As Boolean
Dim TimeOut As Long
On Error GoTo ErrH
 bInConnection = False
 frmSlave.tcpServer.SendData sData
 Do Until (frmSlave.tcpServer.State = 0) Or (TimeOut < 10000)
    DoEvents
    TimeOut = TimeOut + 1
    If TimeOut > 10000 Then Exit Do
 Loop
  SendData = True
  Exit Function
ErrH:
   SendData = False
   MsgBox Err.Description, 16, "Error #" & Err.Number
   frmSlave.lblStatus = "หยุดการติดต่อ"
End Function

Function SendData1(sData As String) As Boolean
Dim TimeOut As Long
On Error GoTo ErrH
 bInConnection = False
 frmSlave.Winsock1.SendData sData
```

```
 Do Until (frmSlave.Winsock1.State = 0) Or (TimeOut < 10000)
    DoEvents
    TimeOut = TimeOut + 1
    If TimeOut > 10000 Then Exit Do
 Loop
 SendData1 = True
 Exit Function
ErrH:
   SendData1 = False
   MsgBox Err.Description, 16, "Error #" & Err.Number
   frmSlave.lblStatus = "หยุดการติดต่อ"
End Function
```