ควบคุมกล้องวิดีโอผ่านเครือข่ายอินเตอร์เน็ต

Camera Control via Internet

โดย

นาย ชัชชัย สืบกลาง

นาย สุชาติ เหมือนจีน

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ควบคุมกล้องวิดีโอผ่านเครือข่ายอินเตอร์เน็ต

Camera Control via Internet

โดย

นาย ชัชชัย สืบกลาง

นาย สุชาติ เหมือนจีน

อาจารย์ที่ปรึกษา

ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

รายงาน ประจำปีการศึกษา 2541

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ควบคุมกล้องผ่านเครือข่ายอินเตอร์เน็ต

ผู้จัดทำ

นาย ชัชชัย สืบกลาง

นาย สุชาติ เหมือนจีน

.....

(ผศ.ดร.สุรพันธุ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

ปริญญานิพนธ์เรื่อง ควบคุมกล้องผ่านเครือข่ายอินเตอร์เน็ต

Camera control via Internet

จัดทำโดย นาย ชัชชัย สืบกลาง เลขประจำตัว 39013153

นาย สุชาติ เหมือนจีน เลขประจำตัว 39013182

อาจารย์ที่ปรึกษา ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

รายงานฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ .....อาจารย์ที่ปรึกษา

(ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์)

วันที่ ..10../..พ.ค./..42....

ควบคุมกล้องผ่านเครือข่ายอินเตอร์เน็ต

นาย ชัชชัย สืบกลาง รหัส 39013153
นาย สุชาติ เหมือนจีน รหัส 39013182
ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ (อาจารย์ที่ปรึกษา)
ภาคการศึกษาที่ 2 ปีการศึกษา 2541

บทคัดย่อ

ปัจจุบันการสื่อสารติดต่อและประยุกต์ใช้งานบนเครือข่ายอินเตอร์เน็ตมีความหลากหลายและใช้งานครอบคลุมได้กว้างขวางที่เห็นได้ชัดก็คงเป็นการแลกเปลี่ยนข้อมูลกัน มันเป็นการดีที่ผู้คนจากมุมโลกหนึ่งหรือที่อยู่ห่างไกลกัน สามารถที่รับรู้ข้อมูลข่าวสารจากอีกที่ที่หนึ่งได้อย่างรวดเร็ว การควบคุมระยะไกลก็ได้เข้ามามีบทบาททางด้านงานการควบคุมต่างๆ เพราะไม่จำเป็นที่ผู้ควบคุมต้องทำงานในสถานที่ที่เครื่องมือตั้งอยู่ ถ้าเราสามารถนำคุณสมบัติของอินเตอร์เน็ตมาประยุกต์ใช้งาน เราก็สามารถที่จะควบคุมเครื่องจักรนั้น จากสถานที่ที่อยู่ห่างไกลได้โดยผ่านเครือข่ายอินเตอร์เน็ต นี่ก็เป็นที่มาในการสร้างโครงงานชิ้นนี้ คือจะเป็นการสร้างกล้องวิดีโอควบคุมระยะไกลโดยผ่านเครือข่ายอินเตอร์เน็ต แนวความคิดเบื้องต้นคือ ต้องการที่จะควบคุมการหมุนของกล้องที่อยู่ไกลออกไปโดยใช้อินเตอร์เน็ตเป็นสื่อกลาง และทำการรับข้อมูลภาพที่ได้จากกล้องตัวนั้น นั่นก็หมายความว่าเราสามารถที่จะตรวจสอบดูเหตุการณ์ต่างๆที่อยู่ในบริเวณนั้นโดยที่ผู้ควบคุมอยู่ที่อื่น ซึ่งเราสามารถนำเอาหลักการเบื้องต้นตรงนี้ไปประยุกต์ใช้งานในด้านการรักษาความปลอดภัยหรืองานที่ต้องการตรวจสอบเหตุการณ์ที่อยู่ห่างไกลออกไป

Camera Control via Internet

Mr.Chutchai Suebklang

Mr.Suchart Muenjeen

Assistant.Prof.Dr.Surapan Airphaiboon

(Advisor)

2nd Semester , education Year 1998

Abstract

Nowaday communication via internet working has several and high performance usage of wide application such as the data exchange. It has an advantage when many people far away use internet to communicate with the other people. The Information can be received and sent faster than the old communication method. The remote control system has a role in the new control technology. Form this reason , we build control video project controlled via internet and observe the picture from the controlled camera .

สารบัญ

บทที่ 1
บทนำ

เนื่องด้วยในปัจจุบันอินเตอร์เน็ตได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น ดังนั้นการทำให้อุปกรณ์หรือเครื่องมือใดๆที่ใช้ประโยชน์จากการติดต่อผ่านอินเตอร์เน็ต ซึ่งจะได้รับประโยชน์อย่างมากมาย ที่เห็นได้อย่างชัดเจนคือก็คงเป็น ความครอบคลุมการใช้งาน และการควบคุมระยะไกล (Remote Control) ซึ่งเครือข่ายอินเตอร์เน็ตนั้นได้ครอบคลุมไปทุกมุมโลกแล้วในขณะนี้ และจำนวนผู้ใช้งานก็มีอัตราเพิ่มขึ้นอย่างรวดเร็ว เราจึงใช้คุณสมบัติตรงนี้ของอินเตอร์เน็ตเข้ามาประยุกต์ใช้ในชีวิตประจำวันได้

ในงานที่ต้องการความสะดวกในการติดต่อหรือควบคุมระยะไกล (Remote Control)ไม่มีการสื่อสารชนิดใดที่จะทำได้ง่ายและรวดเร็วกว่าการติดต่อโดยสื่ออื่นๆ ซึ่งเราไม่จำเป็นต้องอยู่ในสถานที่อันตรายอันเนื่องมาจาก ความไม่ปลอดภัยจากสถานที่ทำงานหรือความไม่สะดวกจากการที่ต้องเดินทางไปปฏิบัติงาน การควบควบคุมสามารถที่จะทำได้ในสถานที่ที่เหมาะสมได้ การควบคุมตรวจสอบหรือดูแล ก็ทำได้ง่ายขึ้น เพียงแต่เพิ่มในส่วนของอินเตอร์เน็ตเข้ามาทำงานร่วมกับชุดควบคุมหรือที่เรียกกันว่า Hardware และเพิ่มชุดของโปรแกรมควบคุมการทำงาน(Software)อีก เพียงเท่านี้เครื่องมือเหล่านี้ก็สามารถที่จะสามารถควบคุมการทำงานผ่านเครือข่ายอินเตอร์เน็ต การควบคุมสามารถที่จะทำได้ทุกมุมโลกที่อินเตอร์เน็ตเข้าถึง ซึ่งมีความสะดวกมาก ยกตัวอย่างเช่น บริษัทรักษาความปลอดภัยแห่งหนึ่งได้รับมอบหมายในการจัดเวรยามคอยเฝ้าดูสถานที่ต่างๆตลอด 24 ชั่วโมง ทั้งหมด 10 สถานที่ที่ตั้งซึ่งอยู่ไกลกันมาก ดังนั้นบริษัทนี้จึงจำเป็นต้องจ้างเวรยามจำนวนมากที่คอยเฝ้าดูแลสถานที่ทั้งหมด สถานที่ละ 3 คนต่อวัน รวมหนึ่งวันก็ 30 คน และต้องเสียค่าใช้จ่ายจำนวนมากต่อวัน และถ้าบริษัทนี้ได้ใช้กล้องวิดีโอควบคุมผ่านเครือข่ายอินเตอร์เน็ตติดตั้งไว้ทั้ง 10 ที่ และใช้เพียงยามคนเดียวต่อการเฝ้าดูการเคลื่อนไหวพร้อมๆกันทั้ง10 ที่ จึงเป็นการลดจำนวนบุคลากรตรงนี้ได้มากเลยทีเดียว

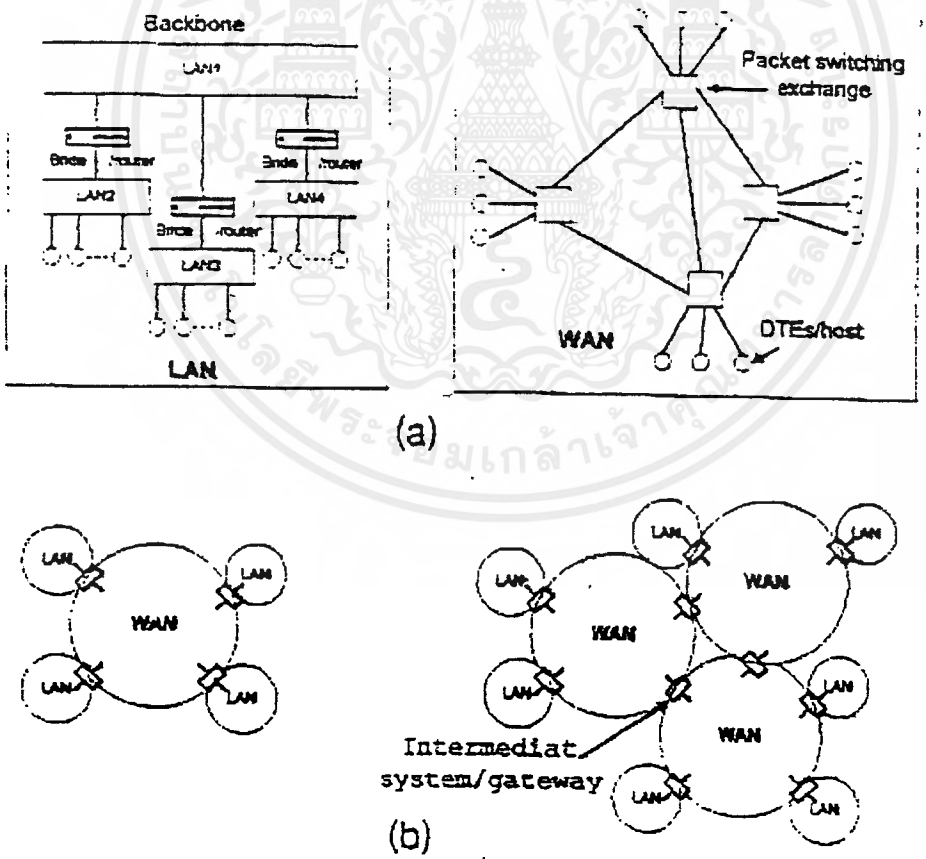
โครงงานนี้ได้นำเอาแนวคิดจากตัวอย่างข้างบนมาทำการศึกษาและสร้างเครื่องมือชุดนี้ขึ้นมาเพื่อให้ได้มาซึ่งวัตถุประสงค์ที่ต้องการ โดยหลักการทำงานอย่างคราวๆ สามารถดูได้จากบล็อกรูปที่ 1.1 ดังต่อไปนี้

บทที่ 2
เครือข่ายอินเตอร์เน็ต

Internet คือ การที่เครือข่าย 2 หรือมากกว่า เชื่อมต่อเข้าด้วยกัน และทำงานเสมือนเป็นเครือข่ายเดียวกัน โดย network ที่เป็นส่วนประกอบของ internet คือ Subnetwork (Subnet) ซึ่งอาจจะเป็นเครือข่าย Local area network (LAN) หรือ Wide area network (WAN) อุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่าย เข้าด้วยกันก็คือ intermediate system (IS) หรือ internetworking unit (IWU) การเชื่อมโยงระหว่างระบบที่แตกต่างกัน จำเป็นต้องมีมาตรฐานในการติดต่อกัน ซึ่งเรียกเป็นศัพท์เฉพาะว่า โปรโตคอล (Protocol)

2.1 สถาปัตยกรรม อินเตอร์เน็ต (Internet Architectures)

สถาปัตยกรรมพื้นฐานของ Internet แสดงดังรูปที่ 2.1

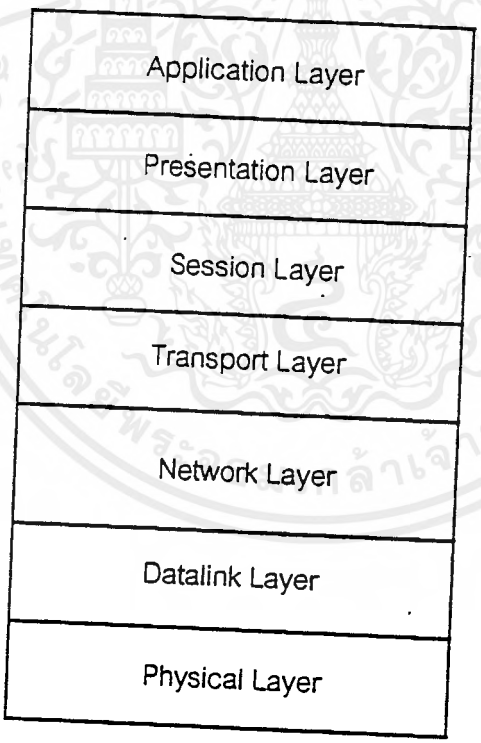


รูปที่ 2.1 แสดงสถาปัตยกรรมของ Internet (a) Single LAN and WAN (b) Interconnected LAN/WAN

ในรูป (a) แสดงตัวอย่าง 2 ตัวอย่างของเครือข่ายเดี่ยว (Single network) ซึ่งอย่างแรกเป็น site-wide LAN ซึ่งประกอบขึ้นมาจากชุดของ LANs ซึ่งถูกต่อเข้ากับเครือข่ายหลัก (backbone) ซึ่งอุปกรณ์ที่ใช้ต่อ LAN เข้ากับเครือข่ายหลัก ถ้า LAN ทุกเครือข่ายมีระบบเดียวกันก็จะใช้ bridge ถ้าเป็น LAN ที่แตกต่างกันก็จะใช้ router ตัวอย่างที่ 2 เป็นตัวอย่างของ WAN เดี่ยว ๆ ในรูป (b) แสดงถึงเครือข่ายอินเตอร์เน็ต ซึ่งประกอบด้วย network ทั้ง 2 ชนิด ข้างต้น

2.2 OSI โมเดล

องค์กรมาตรฐานสากล ISO (International Organization for Standardization) ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อย ๆ และกำหนดโมเดลแบ่งเป็นชั้น ๆ ตามลำดับเรียกว่ามาตรฐาน OSI (Open System Interconnection) โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อย ๆ ก็จะช่วยในการออกแบบ และการใช้งานเครือข่าย รวมถึงการเชื่อมโยงกัน เป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังในรูปที่ 2.2



รูปที่ 2.2 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

ในแต่ละชั้นของ OSI model จะมีการติดต่อสื่อสารกันเป็นชั้น ๆ ตามลำดับลงมาเช่น Application Layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงชั้นแรกสุดคือ Physical Layer

Application Layer เป็นชั้นบนสุดของ โมเดลเป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้ เป็นไปได้ตามต้องการ ตัวอย่างแอปพลิเคชั่นของเครือข่าย เช่น ระบบ e-mail, การโอนถ่ายข้อมูล (File Transfer), การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่ายเป็นต้น

Presentation Layer มีการกำหนดหน้าที่ไม่ชัดเจนนักและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือเป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมไปถึงการจัดแปลงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC, การลดขนาดข้อมูล (data compression) การเข้ารหัส หรือถอดรหัสของข้อมูล แต่ส่วนใหญ่แล้ว แอปพลิเคชั่นจะจัดการแทนได้

Session Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณ และรายละเอียดวิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้าย ที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP (Transmission Control Protocol) ในโพรโตคอล TCP/IP ทำงานที่ระดับนี้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน packet ข้อมูล ผ่านอุปกรณ์ต่าง ๆ ไปยังเครือข่ายย่อยได้อย่างถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูล (Routing table) และกั้นหรือกรอง packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP, TCP/IP และ IPx เป็นโพรโตคอลที่ทำงานอยู่ใน layer นี้

Data link Layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Tokenring หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ ที่จะส่งข้อมูลไป (address) ทั้งนี้ Data link layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก bit ให้เป็น packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลที่เข้ามา ก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งาน และจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

Physical Layer รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณผ่านสายสัญญาณแบบต่าง ๆ การเชื่อมต่อเข้ากับ

เครือข่ายแบบต่าง ๆ โดยใช้ Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือสัญญาณที่จำเป็นในการสื่อสารโดยตรง

เนื่องจาก Network Layer เป็นชั้นที่โพรโตคอล IP, TCP/IP ทำงานอยู่จะกล่าวโดยละเอียดต่อไป

2.3 โครงสร้างชั้นเน็ตเวิร์ค (Network layer structure)

หน้าที่ของ Network Layer ในแต่ละ End System (ES) จะเป็นตัวจัดการการติดต่อแบบ end-to-end ของการบริการ internetwide ไปยังผู้ใช้บริการ (NS-User)

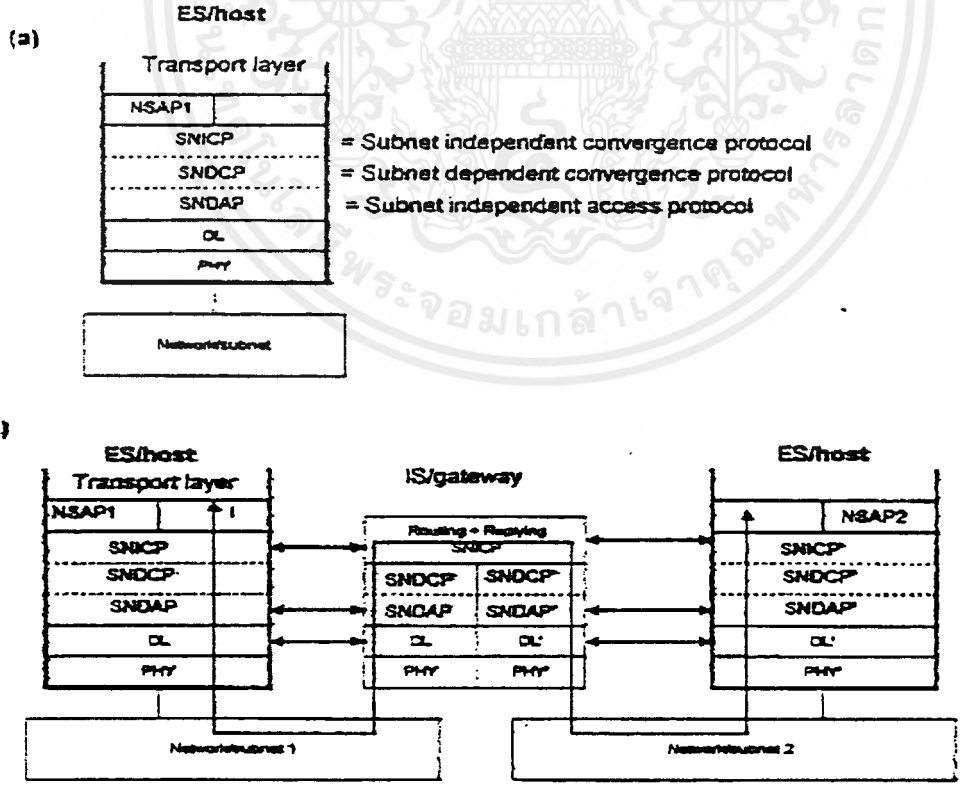
โดย ISO ได้จัด network layer เป็น 3 (sublayer) protocol ซึ่งจะทำงานร่วมกัน เพื่อให้บริการใน network layer ได้แก่

Subnetwork independent convergence protocol (SNICP)

Subnetwork dependent convergence protocol (SNDCP)

Subnetwork dependent access protocol (SNDAP)

โดยที่ โครงสร้างแสดงดังรูปที่ 2.3



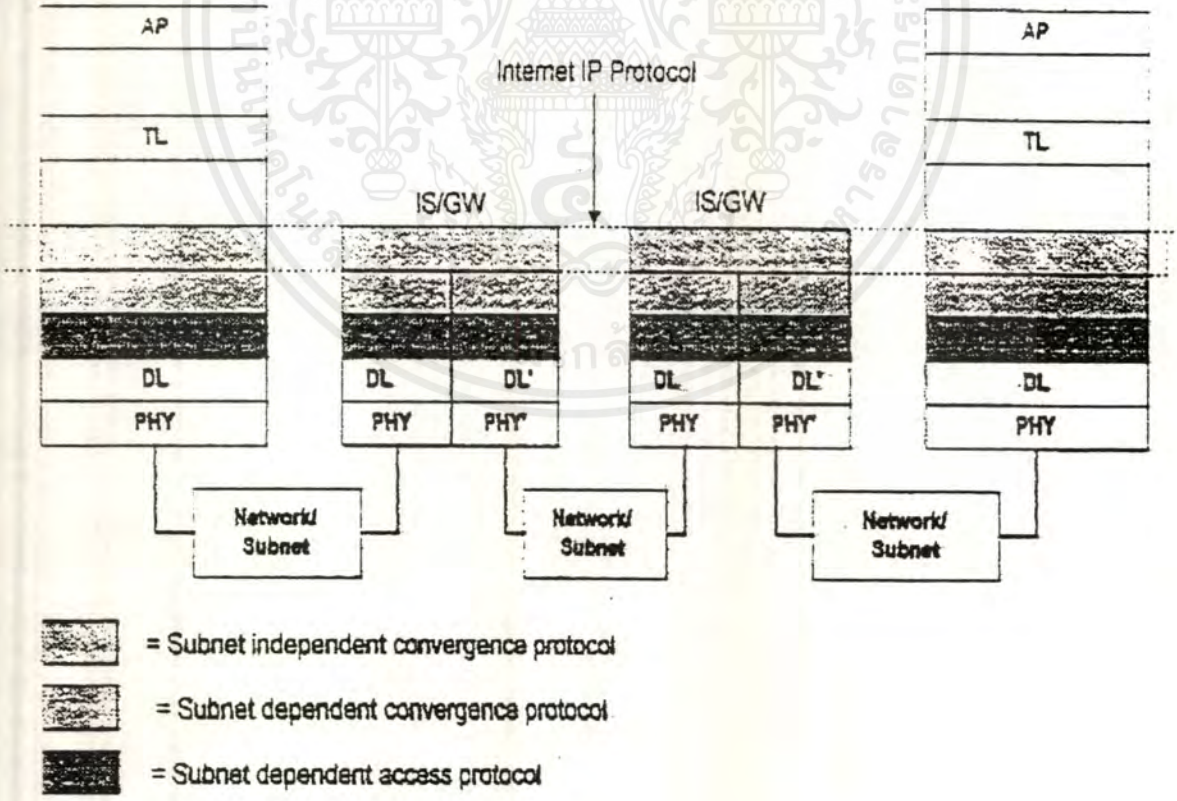
รูปที่ 2.3 Network layer structure (a) Sublayer Protocol ; (b) IS structure

โดยที่ SNICP จะเป็นตัวสนับสนุนจัดการให้ผู้ใช้บริการ (NS-user) สามารถ interface กับ Internet ซึ่งมันจะมีหน้าที่ เป็นตัวประสานฟังก์ชันต่าง ๆ ที่จำเป็นในการเลือกเส้นทางและถ่ายทอดข้อมูลของผู้ใช้ข้าม Internet ซึ่งการทำงานของมันไม่ขึ้นอยู่กับคุณสมบัติเฉพาะของ เครือข่ายย่อย (subnet)

SNDAP จะเป็นตัวโพรโตคอลที่ติดต่อกับเครือข่ายย่อย (subnet) ที่มีลักษณะเฉพาะใน Internet เช่น X.25 packet layer protocol สำหรับเครือข่าย X.25 ซึ่งใช้บ่อยใน LAN เพราะว่าการบริการและการทำงานของ SNDAP แตกต่างจาก network แบบอื่น ๆ sublayer ที่อยู่ตรงกลางคือ SNDCP จะเป็นตัวจัดการระหว่าง SNICP และ SNDAP

2.4 รูปแบบมาตรฐานโปรโตคอลของอินเตอร์เน็ต (Internet protocol standards)

อินเตอร์เน็ตโพรโตคอลซึ่งถูกใช้ในอินเตอร์เน็ตมี โพรโตคอลก็คือ TCP/IP (Transfer Control Protocol / Internet Protocol) ซึ่งรวมถึง transport และ application โพรโตคอล ซึ่งทั้งหมดของ TCP/IP จะกำหนด ให้เหมาะกับการใช้ในเชิงสาธารณะ ซึ่งรูปแบบโดยทั่วไปแสดงดังรูปที่ 2.4



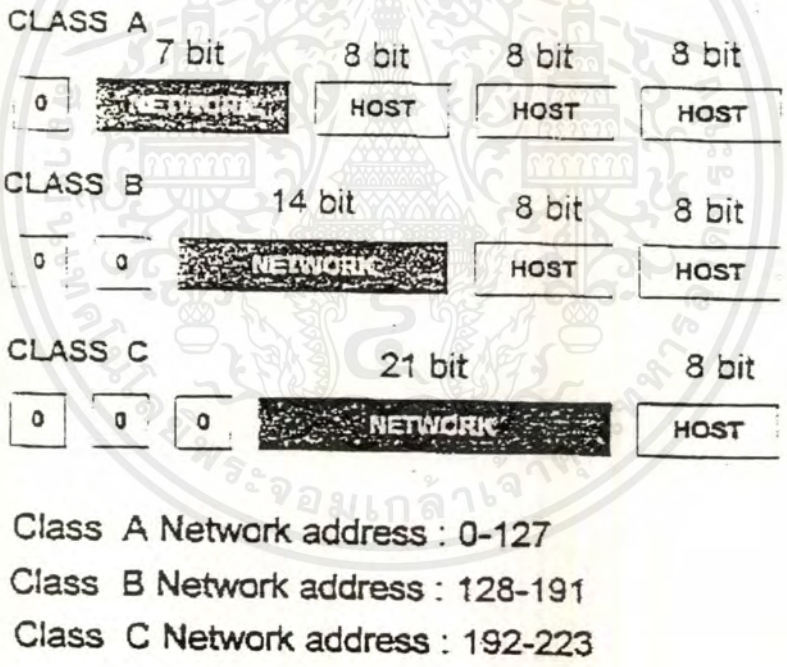
รูปที่ 2.4 Internetwide IP schematic

IP เป็น internetwide protocol ซึ่งทำให้สอง transport protocol ที่ต่างสถานที่และต่าง ESs / hosts กันสามารถแลกเปลี่ยนหน่วยข้อมูล (NSDUS) กันได้ ซึ่งหมายถึงว่า หลาย ๆ network / subnet และ ISs / gateways ที่แตกต่างกันสามารถ ติดต่อสื่อสารกันได้อย่างสมบูรณ์

2.5 Internet IP

2.5.1 Address structure

ในศัพท์ของ ISO เมื่อ 2 network ติดต่อกันด้วย host/ES ที่ต่อกับอินเตอร์เน็ต network เหล่านี้ ติดต่อกันได้โดยใช้ network service access point (NSAP) address และ subnet point of attachment (SNPA) สำหรับใน TCP/IP ก็จะมี IP address และ NPA address ตามลำดับ โดย NPA address จะแตกต่างกันในแต่ละชนิดของ network/subnet ขณะที่ IP address จะเป็นรูปแบบเดียวกัน โครงสร้างของ IP address แสดงดังรูปที่ 2.5



รูปที่ 2.5 โครงสร้างของ Address ที่ใช้ใน class ต่าง ๆ ของเครือข่ายทั้งหมดความยาว 32 bit

IP address นี้มีการจัดแบ่งออกเป็นทั้งหมด 5 ระดับ (class) แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ระดับคือ Class A, Class B และ Class C ซึ่งจะแบ่งตามขนาดความใหญ่ของเครือข่าย ถ้าเครือข่ายใดมีจำนวนเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ใน Class A และลดหลั่นกันมาใน Class B และ Class C ตามลำดับ

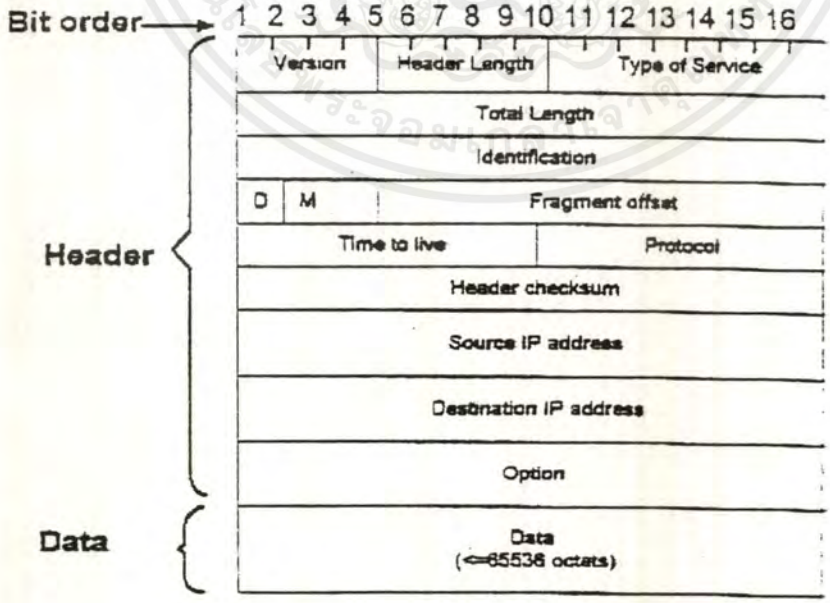
จากรูปจะเห็นว่าหมายเลข IP ของ Class A มีตัวแรกเป็น 0 และหมายเลขของเครือข่าย (network number) ขนาด 7 bit และมีหมายเลขเครื่องคอมพิวเตอร์ (Host number) ขนาด 24 bit ทำให้ในหนึ่งเครือข่ายของ Class A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ในเครือข่ายได้ถึง 2^{24} =16 ล้านเครื่อง แต่ใน Class A นี้ จะมีหมายเลขเครือข่ายได้ 128 ตัวเท่านั้นทั่วโลก ซึ่งก็คือจะมีเครือข่ายใหญ่แบบนี้เพียง 128 เครือข่ายเท่านั้น

สำหรับ Class B จะมีหมายเลขเครือข่ายแบบ 14 bit และหมายเลขเครื่องคอมพิวเตอร์แบบ 16 bit (ส่วนอีก 2 bit ที่เหลือบังคับว่าต้องขึ้นต้นด้วย 10_2) ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่าย Class B แต่ละเครือข่ายได้ถึง 2^{16} = กว่า 65,000 เครื่อง และสุดท้ายคือ Class C ซึ่งมีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit ส่วน 3 bit แรกบังคับว่าต้องเป็น 110_2 ดังนั้นในแต่ละเครือข่าย Class C จะมีจำนวนเครื่องต่อเชื่อมได้เพียงไม่เกิน 254 เครื่องในแต่ละเครือข่าย (2^3=256 แต่หมายเลข 0 และ 255 จะไม่ถูกใช้งาน จึงเหลือเพียง 254)

จะเห็นได้ว่าเมื่อเครือข่ายและเครื่องคอมพิวเตอร์ ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกัน และมีความหมายให้ทราบถึงขนาดเครือข่ายแล้ว การติดต่อส่งผ่านข้อมูล จึงกระทำได้ไม่สับสน

2.5.2 รูปแบบของข้อมูล (Datagrams)

รูปแบบของ IP data unit ก็คือ datagrams ซึ่งโครงสร้างของ datagrams เป็นดังรูปที่ 2.6



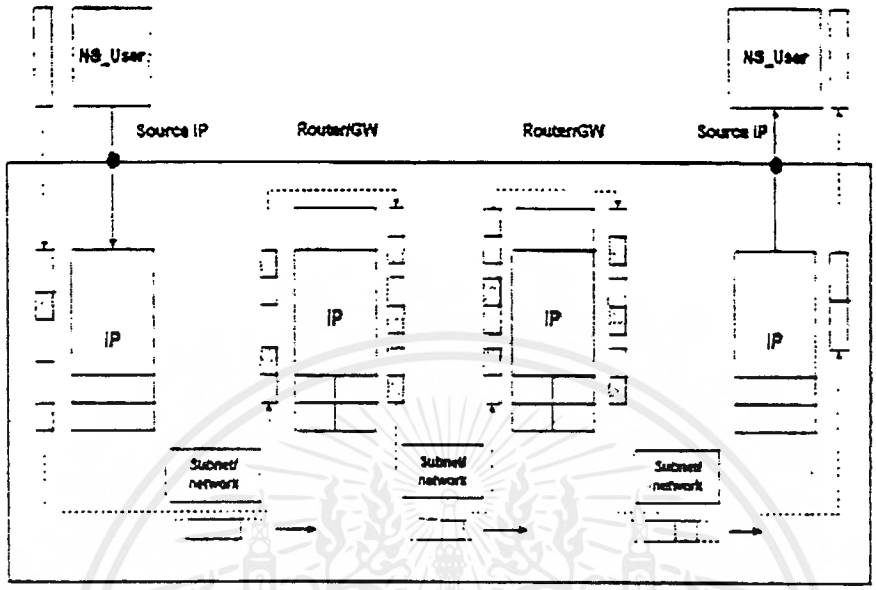
รูปที่ 2.6 Internet datagrams format and contents

หน่วยข้อมูล IP (IP datagrams) แต่ละหน่วยจะประกอบด้วย ส่วนของข้อมูลที่รับมาจากส่วนของงาน TCP หรือ UDP และส่วนของข้อมูลนำทาง (Header) ซึ่งมีรายละเอียดดังนี้

- Version หมายเลขรุ่นของข้อกำหนด IP
- Header Length ความยาวของข้อมูลนำทาง
- Type of service วิธีการจัดการกับข้อมูล
- Total Length ความยาวของหน่วยข้อมูล
- Identification, Flags และ Fragment offset รายละเอียดที่เกี่ยวกับการแบ่งย่อยข้อมูล ซึ่งจะถูกนำมาใช้ในการรวบรวมข้อมูล
- Time to live เวลาสูงสุดที่ใช้ในการเดินทาง ซึ่งกำหนดมาจากต้นทาง เวลานี้จะลดลงเรื่อย ๆ ในระหว่างทาง ถ้าลดลงไปถึงศูนย์ หน่วยข้อมูลนั้นจะถูกกำจัดไป
- Protocol ชนิดของข้อมูลเป็น UDP หรือ TCP
- Header Checksum ค่าตรวจสอบข้อมูลนำทาง
- IP address หมายเลข internetwide IP (NSAP) ของเครื่องต้นทางและปลายทาง
- Option ข้อมูลอื่น ๆ เช่น ข้อมูลเกี่ยวกับการรักษาความปลอดภัย บันทึกเส้นทางเดินของข้อมูล และเวลาที่ข้อมูลเดินทางมาถึง เป็นต้น

2.5.3 การแบ่งส่วนย่อยของข้อมูลและการประกอบขึ้นใหม่ (Fragmentation and Reassembly)

ขนาดข้อมูลของผู้ใช้ซึ่งอ้างอิงกับ NSDU มีความจุได้ถึง 64k หรือ 65,536 bytes แต่ขนาดของหน่วยข้อมูล (packet size) ที่สามารถติดต่อกันในระบบที่ต่างกัน สามารถมีได้ตั้งแต่ 128 byte สำหรับระบบ X.25 packet switching จนถึง 8000 byte สำหรับบาง LAN ดังนั้นกระบวนการ Fragmentation และ Reassembly จึงถูกนำมาใช้เพื่อ ทำให้ขนาดของข้อมูลเล็กลง และสามารถส่งไปในระบบได้ และเมื่อถึงปลายทาง IP ก็จะทำการประกอบข้อมูล (reassembly) ขึ้นมาใหม่ก่อนที่จะส่งผ่านไปยังผู้ใช้ปลายทาง ดังรูปที่ 2.7



รูปที่ 2.7 internet fragmentation and reassembly

อันดับแรก IP ใน Host ต้นทางจะแยกข้อมูลของผู้ใช้ (NS -User), NSDU เป็น Datagram ซึ่งมี Address กำกับเป็นเฉพาะส่วน ๆ ไปซึ่งจะถูกออกคำสั่งโดย Network ที่มันติดต่ออยู่ด้วยและส่ง Datagram ไปยัง IP ใน Gateway ตัวแรก โดยที่ IP ใน Gateway จะไม่ Reassemble NSDU แต่จะปรับปรุงในขอบเขตที่เหมาะสม และส่ง Datagram ที่ได้รับตรงไปยัง Network ที่สอง (ถ้า Network ที่สองสามารถรองรับขนาด Datagram นี้) หรือทำการ Fragment datagram ให้มีขนาดเล็กลง ซึ่งขั้นตอนนี้จะถูกทำซ้ำที่ Gateway ตัวต่อไป โดยในรูปที่ 7 Network หลังสุดสามารถรองรับขนาดของ Packet ได้มากกว่า Packet ที่มันได้รับข้อมูลจึงถูกส่งได้โดยตรง โดยที่จะมีการปรับปรุงในบางส่วน Header ของ Datagram เท่านั้น จากนั้น IP ใน Host ปลายทางจะทำการประกอบข้อมูล (Reassemble) ที่มันได้รับขึ้นมาใหม่ และส่งผลที่ได้รับก็คือ NSDU ไปยังผู้ใช้ (NS-user)

ในการคิดค่าเวลาสูงสุดที่ Host ต้นทางกำหนดให้ Gateway รอ Datagram (NSDU) ระหว่างแต่ละการ Assembly ซึ่งก็คือ time-to-live ซึ่งจะถูกนำติดไปที่ส่วน Header ของ Datagram ซึ่งจะถูกตั้งค่าโดย IP ใน Host ต้นทาง ซึ่งจะมีค่าลดลงเรื่อยๆในแต่ละขั้นตอนของการ Process datagram ถ้า Datagram ถูก Fragment ค่าปัจจุบันจะถูกนำไปใส่ในส่วน Header ของ Datagram ตัวใหม่ ถ้ามันถึงค่า 0 ที่จุดใด ๆ ระหว่างการ Process ใน Gateway (หรือ Host) การ Reassembly ก็จะล้มเหลวและทุกๆการ Fragment ที่เกี่ยวกับ NSDU ก็จะถูกดัดทิ้ง

ค่า Time-to-live ในแต่ละ Datagram จะเป็นจำนวนเท่าของ 1 วินาที โดยที่จำนวนของมันจะถูกลดลงโดยแต่ละ IP ซึ่งจะเปลี่ยนแปลงไปตามค่าเวลาจริงในการส่งถ่ายข้อมูลของ Network ที่ติดต่อด้วย

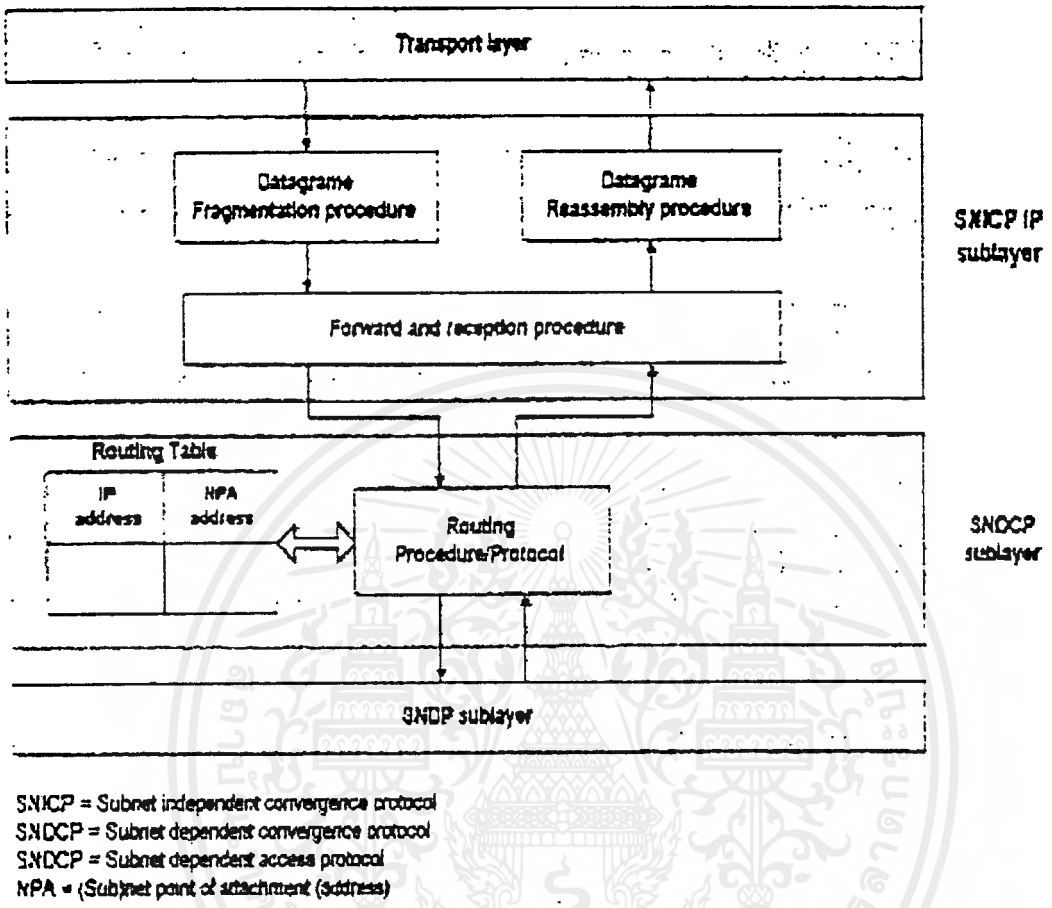
2.5.4 การเลือกเส้นทาง (Routing)

ในแต่ละ Network (หรือ subnet) ใน Internet จะมีชนิดของ PA address ที่แตกต่างกัน ซึ่งระบบ (system)-host หรือ gateway ที่ถูกต่อเข้ากับ network จะสามารถส่ง datagram ไปยังระบบอื่นได้โดยตรงเฉพาะ network ที่เหมือนกันเท่านั้น ในการเลือกเส้นทาง (routing) ให้ datagram ข้ามไปยังหลาย ๆ network IP ในแต่ละ internetwork gateway ต้องรู้ PA address ของ host ปลายทาง

ซึ่งมี 2 วิธีการพื้นฐานที่ถูกใช้ในการหาเส้นทางภายใน Internet คือ centralized และ distributed ด้วยวิธีการ centralized routing ข้อมูลเกี่ยวกับการเลือกเส้นทาง ที่เกี่ยวข้องกับแต่ละ gateway จะถูก download จาก site ส่วนกลางโดยใช้ข้อมูล network และ special network management โดย network management จะพยายามตรวจสอบ network และ host ที่ถูกเพิ่มเข้าและถอดออก และข้อบกพร่องที่จะถูกวินิจฉัยและตรวจซ่อม

ด้วยวิธีการ Distributed routing ทุก ๆ host และ gateway จะร่วมกันในการแบ่งปัน วิธีการในการรับประกันว่า ข้อมูลเกี่ยวกับการเลือกเส้นทางในแต่ละ system, host และ gateway จะถูกทำให้ทันสมัย และสอดคล้องกัน ข้อมูลเกี่ยวกับการเลือกเส้นทางจะถูกจดจำไว้โดยแต่ละระบบ ในรูปของ routing table ซึ่งจะมี NPA address ไว้ใช้ในการส่งแต่ละ datagram ซึ่ง Internet จะใช้วิธีการแบบนี้

ขั้นตอนการ Routing ที่เกี่ยวกับ IP ขั้นตอนแรกจะอ่าน IP address (NSAP) ปลายทางจากภายใน datagram และใช้มันในการหาการตอบสนอง PA address ของ host หรือ gateway จาก routing table ในส่วนที่เพิ่มเติมชุดของ routing protocol จะถูกใช้เพิ่มและรักษาส่วนที่อยู่ในแต่ละ routing table ในแบบของ distributed ซึ่งรูปแบบทั่วไปที่ถูกใช้ภายใน host IP แสดงดังรูปที่ 2.8



รูปที่ 2.8 รูปแบบทั่วไปของการเลือกเส้นทางภายใน Host

บทที่ 3
หลักการเขียนโปรแกรม Winsock ที่ใช้กับ TCP/IP

ที่ผ่านมาเร็วๆ นี้ การสื่อสารข้อมูลได้แพร่หลายอย่างมาก เมื่อประมาณ 5 ปี ที่ผ่านมามี Programmer จำนวนไม่มากที่ได้รับข่าวของ TCP/IP ซึ่งเป็นชุด Protocol ไว้ใช้ในการส่งผ่านเครือข่าย Internet สำหรับงาน Internetwork Communication

ในช่วง 5 ปีที่ผ่านมาไม่มีระบบปฏิบัติการใดที่จะสามารถเอาใจใส่ทำให้สมบูรณ์เหมือนดังทุกวันนี้ ซึ่งระบบปฏิบัติการ Windows ก็ยังไม่ถูกยกเว้น

Application Programming ที่ส่วนมากใช้กันอย่างกว้างขวางสำหรับ TCP/IP Programming คือ Berkeley Sockets Library ซึ่ง ได้ถูกใช้ในการส่งออกที่ Internet สำหรับการให้สิ่งแวดล้อม TCP/IP Application และเป็นธรรมดาอยู่แล้วเมื่อเวลาผ่านมา Microsoft ได้คิดค้น API สำหรับ TCP/IP ภายใต้ระบบปฏิบัติการของ Windows

ในการใช้งานของ Berkeley Sockets จะไม่ถูกจำกัดในเรื่องของ TCP/IP Programming Berkeley Sockets จะแสดงให้เห็นว่าแม้การติดต่อที่เป็นงานหนัก เมื่อใช้ TCP/IP หรือ Protocol อื่นๆ สำหรับการส่งค่าจริงๆ ของข้อมูลอย่างไรก็ตามในปัจจุบัน Winsock Implementation จะจำกัดการใช้งานของ Berkeley Sockets ในการติดต่อกับ TCP/IP

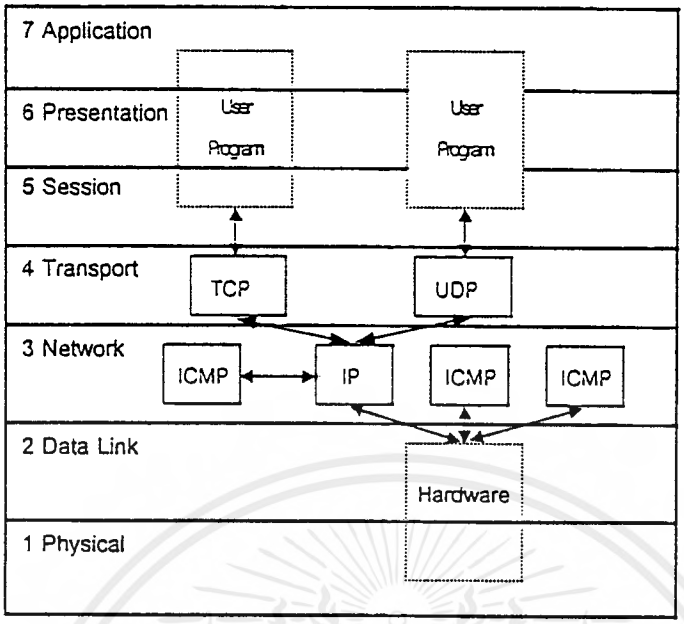
3.1 TCP/IP Networks and OSI

เพื่อเป็นการประหยัดเวลาจะอธิบายเรื่องราวของ TCP/IP อย่างสั้นๆ คือ เกิดจากการคิดค้นหาทางพัฒนา Protocol Suite ของ ARPANET หลังจากนั้น NSFNET ก็ได้ปรับปรุงพัฒนาอีกในเวลาต่อมา

3.2 The Internet Protocol Suite

Internet Protocol Suite ส่วนมากจะประกอบไปด้วย Components หลาย ๆ อัน ซึ่งมากกว่า TCP/IP ดังรูปที่ 2.1 จะแสดงรูปแบบง่ายๆ พร้อมคำอธิบายทั้งหมด 7 Layer ของ OSI (Open Systems Interconnect)

3.2.1 TCP (Transmission Control Protocol) จะให้ Byte Stream ที่เชื่อถือได้ระหว่าง 2 กระบวนการรับและส่ง ความเชื่อถือได้นี้ในคำนี้จะหมายถึง Applications ไม่ต้องการที่จะตักเตือนสำหรับความผิดพลาดเมื่อเกิด Corrupt Packets TCP ส่วนมากจะเป็น Connection Oriented Protocol Application ที่สื่อสารผ่าน TCP จะอยู่ในรูป Logical Connection ก่อนที่จะเปลี่ยนแปลงข้อมูลโยกย้ายและติดต่อ เมื่อข้อมูลได้ทำการแปลงแล้วอย่างสมบูรณ์



รูปที่ 3.1 The Internet Protocol Suite and OSI Layer

3.2.2 UDP (User Datagram Protocol) จะเป็นโปรโตคอลแบบ Connectionless Protocol ชุดของข้อมูลจะถูกกำหนดที่อยู่และถูกส่งไปยังปลายทางที่เฉพาะเจาะจงเท่านั้น UDP เป็นโปรโตคอลที่ไว้วางใจไม่ได้ ดังนั้น Application ต้องเตรียมการที่จะเลือกข้อมูลชุดที่เสียออกไป

3.2.3 ICMP (Internet Control Message Protocol) โปรโตคอลส่วนมากจะถูกใช้ในการส่ง Error และ Control information บน TCP/IP Networks ชุดของ ICMP ไม่บ่อยนักที่จะถูกใช้โดยกระบวนการของผู้ใช้ และจุดเด่นอีกอย่างหนึ่งก็คือ Ping Utility ซึ่งใช้ในการตรวจสอบการเข้าดึงข้อมูลของ Host ที่อยู่ไกลออกไป โดยจะส่ง ICMP "Echo" Packet และจะแสดงผลของมันกลับมา

3.2.4 ARP คือ Address Resolution Protocol จะถูกใช้เพื่อแปลความหมายของ Network Address ไปยัง Hardware Address

3.2.5 RARP คือ Reverse Address Resolution Protocol จะใช้เพื่อถ่ายโอน Hardware Address เข้าไปใน Network Address

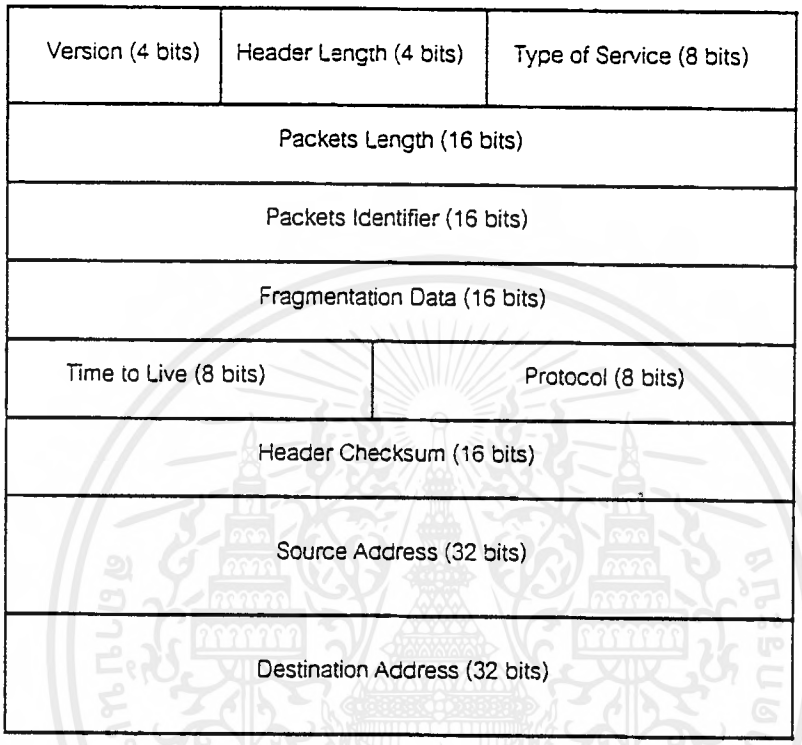
3.3 IP Datagram

หัวใจหลักของการส่งข้อมูล TCP/IP คือ IP Datagram ซึ่งจะเป็นชุดของข้อมูลซึ่งประกอบด้วย Source , Address ปลายทาง , Type of Service Information , User data และ Error Correction information

IP Datagram จะประกอบไปด้วยส่วนของ Header และ Block ของข้อมูล ซึ่งข้อมูลสามารถที่จะกำหนดขึ้นอยู่กับชนิดของ Service และความต้องการของผู้ใช้ ในส่วนของ Header จะประกอบด้วยชุดของ Well-Defined Field

3.3.1 IP Headder

ในส่วนหัวของ IP Datagram หรือ IP Header โดยส่วนมากจะประกอบไปด้วย 20 bytes ตามรูปที่ 2.2 จะแสดง IP Header



รูปที่ 3.2 แสดง IP Header

ตามรูปที่ 1.2 ครั้งแรกจะพิจารณาสิ่งที่น่าสนใจในส่วนของ Protocol, Source, Address และ Destination Address Field ในส่วนของ Protocol จะถูกกำหนดวิธีของ IP Packet จะถูกแปลทั่วๆไป ค่าต่างๆ จะถูกกำหนดสำหรับส่วนนี้ ในส่วนของ Address จะแสดงให้เห็นส่วนของ Host Address ซึ่ง Address นี้จะเพียงหมายเลขเดียวเท่านั้นที่เหมือนกันในระบบ Internet

3.4 IP Host Address and Routing

IP host address จะเป็นข้อมูล 32 bits ซึ่งมีเพียงหมายเลขเดียวเท่านั้นที่เหมือนกันซึ่งใช้ในการติดต่อกับ Internet host ในส่วนของ Gateway (เป็นตัวทำหน้าที่ Interface Network ที่มีมากกว่า 1 Network) จะมี Host อยู่มากมาย

ตามหลักที่ถูกต้องคือ 4 bytes ของ Internet host address มักจะเขียนให้อยู่ในรูปของเลขฐานสิบ ตัวอย่างเช่น 127.0.0.1

Internet host address ส่วนมากจะถูกแบ่งออกเป็น 2 ส่วน คือ Network address และ Actual host address ถ้าเรียงตามความยาวทั้งสองตาม address โดยมากจะขึ้นอยู่กับการกำหนด bytes ใน address เสียเป็นส่วนใหญ่

Class A Network address จะสังเกตุได้ว่า byte แรกจะอยู่ระหว่าง 0 ถึง 127 และจะประกอบไปด้วย 8 bit network address และ 24 bit host address เนื่องจาก Address จะเริ่มต้นที่ 0 และ 127 จะถูกสงวนเอาไว้ ซึ่งสามารถมีค่าสูงสุดคือ 126 Class A Subnets ใน Class A นี้สามารถมได้ทั้งหมด 16,777,214 hosts (address อยู่ในรูป nnn.0.0.0 และ nnn.255.255.255 จะถูกสงวนเอาไว้

Class B Network address โดยส่วนมากจะสังเกตุได้จากตัวเลขหลักแรกสุดอยู่ระหว่าง 128 และ 191 , Class B Network จะประกอบไปด้วย 16 bits network Address และ 16 bits host Address และสามารถมี 16,383 Subnets Class B Subnets สามารถที่จะบรรจุได้ถึง 65,534 hosts (address อยู่ในรูป nnn.mmm.0.0 และ nnn.mmm.255.255 จะถูกสงวนไว้)

Class C Network address จะสังเกตุได้จากตัวเลขหน้าสุดอยู่ระหว่าง 192 และ 223 ซึ่งจะมีได้ 2,097,152 Subnets Class C Subnets สามารถที่จะบรรจุได้ 254 Hosts (โดยจะเริ่มที่ address nnn.mmm.kkk.0 และ nnn.mmm.kkk.255 จะถูกสงวนเอาไว้)

Class D Network address (โดยทั่วไปตัวเลข Byte แรกจะอยู่ระหว่าง 224 ถึง 255) จะถูกสงวนเอาไว้เพื่อ IP Multitasking ที่ถูกจำกัดจากในส่วนของ IP broadcasting คือจะไม่ให้ไปยุ่งเกี่ยวกับ Winsock programmer

3.5 Host Names

ในระหว่างการคิดประดิษฐ์ Internet ซึ่งตัวเลขที่จะนำมาแทน host address ซึ่งจะมีไม่เพียงพอและอีกอย่างหนึ่งคือตัวเลขเหล่านี้เป็นการยากที่จะจดจำ และอีกอย่างถ้าเกิด host address มีการเปลี่ยนแปลงอันเนื่องจากหลายๆ สาเหตุ จึงทำให้เกิดความยุ่งยาก ดังนั้นระบบ Naming System จึงได้ถูกสร้างขึ้นมา ซึ่งใช้ในการ Map ตัวเลขของ IP address ไปเป็น Memories host Names

ในทุกวันนี้จะมีอยู่ 5 hosts บน Internet ทุกๆ host จะถูกจับเปลี่ยนให้อยู่ในรูปของ Files กับข้อมูลของ Internet host ทุกตัว และ address อย่างไรก็ตามอินเตอร์เนตก็ได้เจริญเติบโตอย่างต่อเนื่องทำให้ดูว่าจะไม่เพียงพอ อย่างแรกคือการตั้งชื่อ Internet host จะทำได้มีมาตรฐาน อย่างที่ 2 คือทางด้าน Hardware สามารถที่จะแยกแยะความต้องการที่จะติดต่อกับทุกเครื่องคอมพิวเตอร์ทุกตัวที่ต่ออินเตอร์เนตอยู่

คำตอบของปัญหานี้คือการกำหนด Domain Name System (DNS) ซึ่งจะเป็นตัวจัดระเบียบของการตั้งชื่อ (Naming System) รูปแบบของ Domain Names คือจะคล้ายกันทุกคนที่ใช้อินเตอร์เนต โดยปกติการตั้งชื่อจะอยู่ในรูปแบบคือ Host.Subdomain.Domain

ในส่วนที่สูงสุดคือ Root Domain แสดงว่ามีเพียงชุดเดียว ลำดับต่อไปคือ Top-Level Domain ในส่วนนี้จะกำหนดโดยหน่วยงานหนึ่ง (ส่วนมากเป็นหน่วยงานในสหรัฐ) หรือโดยแต่ละประเทศ โดยจะมีดังต่อไปนี้

GOV : Government Bodies

EDU : Education Institutions

COM : Commercial enterprises

MIL : Military organization

ORG : Other organization

Top-level Domain โดยใช้ชื่อระเทศ มักจะใช้ตามมาตรฐาน ISO 3166 โดยจะใช้อักษรย่อ 2 ตัวของชื่อประเทศ

3.6 TCP และ UDP Packets, Port, Port numbers and Sockets

เพื่อความกระจ่างการที่เราทราบ host address ของปลายทางของ IP Packets คงไม่เพียงพอสำหรับ application ส่วนใหญ่เบื้องหลังทุกอย่าง host สามารถที่จะเข้าไป open Connection โดยเข้าไปใช้ใน TCP/IP ต่างๆ เพื่อจะเปลี่ยนแปลงข้อมูลและเมื่อ IP Packets ถูกรับไป host จะมีวิธีการอย่างไรในการที่จะเปลี่ยนแปลง Packet ตัวนี้

ในสาเหตุนี้ของ TCP และ UDP Packets ในการที่จะเพิ่มในส่วนของ IP Header, Packet นี้มักจะบรรจุ header Information ที่ถูกเพิ่มเข้ามา สิ่งแรกคือ 4 ไบต์ของ TCP และ UDP Header จะบรรจุด้วย 2 ไบต์สำหรับ Source และ 2 ไบต์สำหรับ Port number ของปลายทาง

Port number จะถูกรวมเข้ากับ IP number ในรูปของ Sockets ซึ่งมีเพียงหนึ่งเดียวที่ถูกส่งออกมาจาก Internet

หมายเหตุ TCP และ UDP Ports จะไม่คล้ายกัน เช่นตัวอย่าง TCP ใช้ Port 25 และ UDP ใช้ Port 25 จะแสดงให้เห็นถึงการเข้าถึงข้อมูลที่ต่างกัน

3.7 Internet Services

มีบริการมากมายหลายชนิดบน IP Networks ที่ใช้กันอย่างแพร่หลาย เช่น FTP, Telnet, Gopher,The WWW, Archive, DNS name, Service, Whois, Finger และอื่นๆ อีกมากมาย

Protocol จะถูกใช้สำหรับบริการเหล่านี้โดยจะถูกกำหนดโดย Internet Requests For Comment Document หรือ RFCs ซึ่งบริการเหล่านี้จะเปลี่ยนแปลงตาม Well-Known Port

Number ยกตัวอย่างเช่นการจะติดต่อโดย Telnet Server บน Host ต่างๆ บน Internet คุณจะต้องทำการติดต่อไปที่ Port 23 บนเครื่องนั้น

โดยระบบส่วนมาก TCP และ UDP Port หมายเลข 0 – 1023 ต่างถูกสงวนเอาไว้สำหรับบริการพิเศษต่างๆ

3.7.1 The Winsock API

ในการอินเตอร์เฟสด้วย Berkeley Sockets สามารถที่จะใช้ในการติดต่อสื่อสารโดยใช้ Connection-Orientd Protocol (TCP) และ Connectionless Protocol (UDP) การเขียนโปรแกรมจะเป็นแบบ Client Server ; Server จะรอเมื่อมีการร้องขอเข้ามา ระหว่างนั้น Client จะเริ่มทำการพิจารณาการทำงาน

จะกล่าวถึงความแตกต่างระหว่างเครื่องมือในการทำงานระหว่าง Winsock และ Unix เวอร์ชันของ Berkeley Sockets ส่วนมากจะแสดงให้เห็นว่าแท้จริงแล้วลักษณะของ Sockets และ Files ไม่สามารถที่จะใช้คุณสมบัติแทนกันได้ ในที่นี้จะทำให้เห็นผลเมื่อมีการกำหนด Port ของ application ของ equivalence นี้

ความแตกต่างอีกอย่างคือ Winsock Library ต้องการการกำหนดค่าเริ่มต้น Application ใดที่จะใช้ Winsock function ต้องทำการเรียก WSAstartup function และก็จะทำงานที่ Winsock Library และเมื่อสิ้นสุดการทำงานจะทำการเรียก WSAcleanup สำหรับการธtermination อย่างสมบูรณ์

3.8 Winsock Initialization

เมื่อ Winsock Library ถูกเริ่มต้นการทำงานด้วยการเรียก WSAstartup Application จะทำการเรียก function นี้แล้วให้ address ไปที่ WSAdata Structure ซึ่งจะเป็นตัวแปรสำหรับเก็บค่า Initialization information

ตลอดเวลาที่ผ่านมาเราเรียก WSAstartup การประยุกต์ใช้งานต่างๆ จะไม่สนใจ Version ของ Winsock Library ในการ Initialization จะไม่เกิดความผิดพลาด ถ้าไม่เกิดการ Overlap ระหว่าง Version number ที่ถูก Supported โดย Application และ Version ที่ Supported โดย Winsock Library

ถ้ามีความผิดพลาดเกิดขึ้น WSAstartup จะส่งค่ากลับที่ไม่ใช่ศูนย์ ซึ่ง Application แต่ละตัวสามารถที่จะดู Information ของ Error ที่เกิดขึ้นได้จาก WSAgetlastError

3.9 การเพิ่มและการใช้ Sockets

Sockets จะถูกเพิ่มขึ้นมาโดยการเรียกใช้ฟังก์ชั่น Sockets พารามิเตอร์ของฟังก์ชั่นนี้คือ ชนิดของ Sockets ชนิดของ Network Address และ Protocol ตัวอย่างเช่น Sockets (AF_INET,

SOCK_STREAM, IPPROTO_TCP)

ในระบบมาตรฐานของ Sockets จะต้องประกอบไปด้วย host address และ port number ซึ่ง application นี้สามารถที่จะเรียกฟังก์ชั่น Bind ในการเพิ่ม Sockets Description ฟังก์ชั่น Bind จะนำพารามิเตอร์ นั้นก็คือชุดของ Structure descriping ของ Sockets address ซึ่งสามารถดูได้จาก

```
Struct Sockaddr
{
    u_short  Sa_family;
    char  Sa_data[14];
};
```

application สามารถที่จะเข้าถึงได้ง่ายด้วย Component มากมายโดยการส่งผ่าน Sockaddr_in Structure ซึ่งกำหนดได้ดังนี้

```
Sturct Sockaddr_in
{
    Short  Sin_family;
    U_short  Sin_port;
    Sturct  in_addr  Sin_addr;
    Char  Sin_Zero[8];
};
```

ในส่วนของ Members คือ Sin_port จะเป็น 16 bits Port number และ Sin_addr คือ 32 bits host address

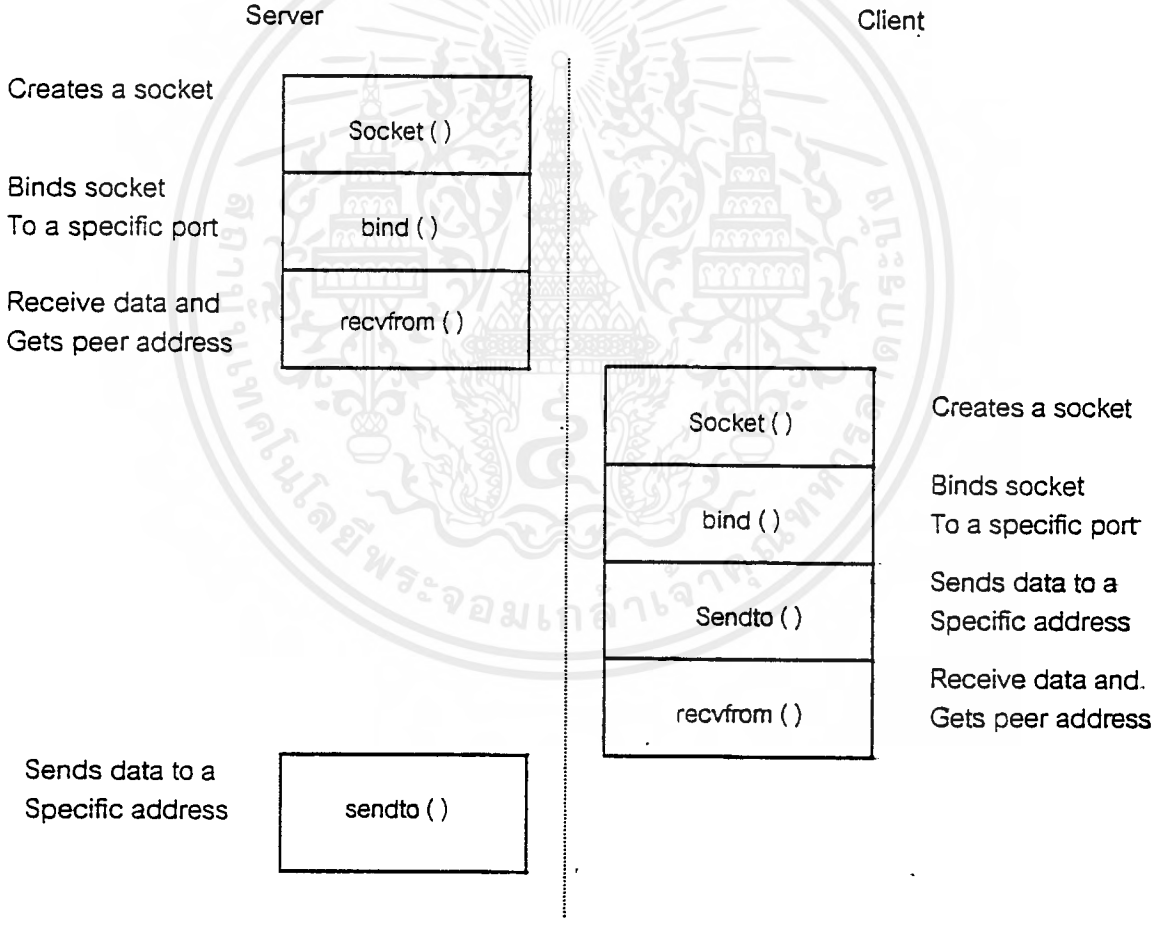
3.10 Name Services

ในการกำหนดค่าที่จะส่งให้กับ Sockaddr_in Structure จะต้องประกอบไปด้วย 32 bits host address ในการที่จะบรรจุเมื่อเราได้เขียน host ที่เรารู้ จะใช้ Gethostbyname function

เมื่อเรียกใช้ Gethostbyname application จะส่งค่า Symbolic name ของ hostและรับ Pointer จาก hostent structure กลับมา hostent Structure จะถูกกำหนดได้ดังนี้

```
struct  hostent
{
    char FAR * h_name;
    char FAR * FAR * h_aliases;
```

ในการติดต่อกันแต่ละครั้งเมื่อสำเร็จแล้ว ทั้ง Client และ Server สามารถที่จะเรียกใช้ Send ในการส่งข้อมูลหรือเรียกใช้ Recv ในการรับข้อมูล และในการหยุดการติดต่อจะใช้ Close Socket function และถ้าเป็นกรณีของ Connectionless UDP Protocol ลำดับเหตุการณ์ที่เกิดขึ้นของ Client และ Server จะไม่เหมือนกันทุกอย่าง โดยจะเพิ่ม Respective Sockets และ bind กำหนด Port number ที่แตกต่างกันคือ Server หลังจากนั้นจะเรียกไปที่ recvfrom function ซึ่งไว้สำหรับรอข้อมูลที่จะเข้ามา Client จะใช้ Sendto เพื่อที่จะ Senddata ไปที่ Specific address และเมื่อ data ถูกรับแล้วโดย Server และ recvform จะถูก return และ Server มักจะประกอบไปด้วยรูปแบบของ address ซึ่งจะเป็น data ที่รับเข้ามา สามารถที่จะใช้ address อันนี้ในการ Subsequent โดยการเรียก Sendto ซึ่งดูได้จากรูปที่ 3.4



รูปที่ 3.4 แสดง Socket connection for UDP

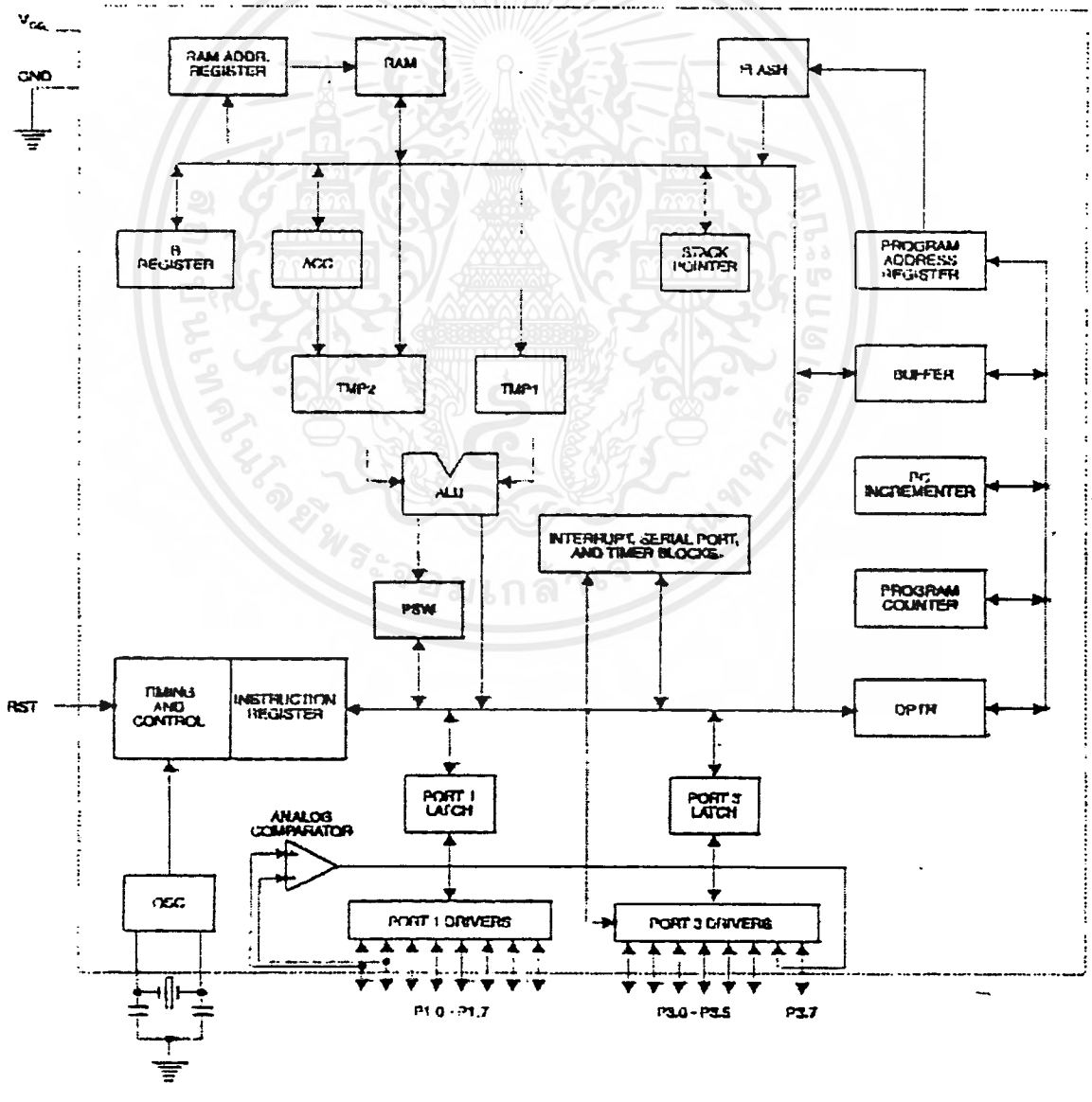
บทที่ 4
พื้นฐานไมโครคอนโทรลเลอร์ และ RS-232

4.1 ไมโครคอนโทรลเลอร์

เป็นส่วนควบคุมการทำงานของ ชุดควบคุมมอเตอร์ ซึ่งหน้าที่ของไมโครคอนโทรลเลอร์

1. รับคำสั่งควบคุมจาก Computer มาประมวลผลเพื่อที่จะทำงานต่อไป
2. จัดเก็บข้อมูลตำแหน่งของมอเตอร์ทั้งสองตัวเพื่อที่จะส่งไปให้ Computer

Block Diagram



รูปที่ 4.1 แสดงสถาปัตยกรรมของ 89C2051

89C2051 ไมโครคอนโทรลเลอร์แต่ละข้างจะมีขาอยู่ข้างละ 10 ขา รวมกันทั้งหมด 20 ขา โดย V_{cc} ขาที่ 20 เป็นขาที่ป้อนไฟเลี้ยง +5 โวลท์ เข้าไปเพื่อให้วงจรรวมทำงานได้ ระดับแรงดันของลอจิก 0 และ 1 ของ 89C2051 จึงต่อเข้ากับอุปกรณ์ลอจิกแบบ TTL ได้โดยตรง Port 1 เป็นพอร์ตขนานขนาด 8 บิต อยู่ที่ขา 12 ถึง 19 เริ่มจากบิต 0 ถึงบิต 7 ตามลำดับ แต่ละขาจะเขียนว่า P1.0,P1.1,....P1.7 ดังนั้น P1.7 หมายถึงบิต 7 ของพอร์ต 1 ซึ่งเป็นบิตที่มีนัยสำคัญสูงสุด (Most Significant) และ P1.0 คือ บิต 0 ของพอร์ต 1 เป็นบิดที่มีนัยสำคัญน้อยที่สุด (Least Significant) พอร์ต 0 นี้ใช้ได้ทั้งการรับ-ส่ง ตำแหน่งของข้อมูลกับหน่วยความจำหรือใช้เป็นพอร์ตรับ-ส่งข้อมูลก็ได้ ข้อมูลที่ส่งออกทางพอร์ต 1 จะถูกค้างค่า (Latch) ไว้ที่ขาพอร์ต

วงจรภายในส่วน Timing and Control จะเป็นตัวสร้างสัญญาณมาควบคุมวงจร เพื่อให้การทำงานแต่ละอย่างข้างต้น เมื่อแต่ละบิตของพอร์ต 1 มีการทำงานคือ รับ-ส่งข้อมูลกับ Data Memory หรือใช้รับข้อมูลจาก Program Memory และ ใช้รับ-ส่งข้อมูลผ่านทางพอร์ตโดยตรง ในกรณีที่ไม่มีการใช้หน่วยความจำ Program Memory หรือ Data Memory ภายนอกแล้ว วงจร Timing and Control จะทำให้สภาวะลอจิกของขา Control เป็น 1 ซึ่งทำให้สวิทช์ MUX อยู่ในตำแหน่งข้างบน เมื่อพอร์ต 1 จะส่งข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำหรือข้อมูลที่จะเขียนออกไปยังหน่วยความจำภายนอกก็จะส่งค่าดังกล่าวมายัง ADDR/DATA ถ้าข้อมูลที่ส่งมาเป็น 1 ก็จะทำให้สัญญาณออกจาก AND GATE เป็น 1 และสัญญาณที่ออกจาก Inverter ดังนั้น FET ตัวบน ON (สภาวะ ON ของ FET คือความต้านทานระหว่างขา D กับ S มีค่าต่ำมากเสมือนกับวงจรปิด) ส่วน FET ตัวล่าง OFF (สภาวะ OFF ของ FET คือความต้านทานระหว่างขา D กับมีค่าสูงมากเสมือนกับเป็นวงจรปิด) สภาวะลอจิกที่ขา P1.X PIN จะเป็น 1 แต่ถ้าข้อมูลที่ส่งออกมายัง ADDR/DATA เป็น 1 ก็จะทำให้สัญญาณจาก AND GATE เป็น 0 และสัญญาณที่ออกจาก Inverter เป็น 1 ดังนั้น FET ตัวบนจะ OFF ส่วน FET ตัวล่างจะ ON ทำให้สภาวะลอจิกที่ขา P0.X PIN เป็น 0 เมื่อ 89C2051 ต้องการใช้พอร์ต 0 สำหรับอ่านข้อมูลจากหน่วยความจำภายนอกก็จะทำได้โดยวงจร Timing and Control ทำให้สภาวะลอจิกของสัญญาณ Control ในรูปเป็น 0 ทำให้เอาท์พุทจาก AND GATE เป็น 0 FET ตัวบนจะ OFF และสวิตช์ MUX จะอยู่ในตำแหน่งข้างล่างดังนั้น FET ตัวล่างจะ ON หรือ OFF ก็แล้วแต่ข้อมูลที่ขา Q มีสภาวะลอจิก 0 ทำให้ FET ตัวล่าง OFF ดังนั้นขา P0.X จะอยู่ในสภาวะอิมพีแดนซ์สูงเพราะ FET ทั้ง 2 ตัว OFF

Port 1 เป็นพอร์ตขนานขนาด 8 บิตคือขา P1.0 ถึง P1.7 โดย P1.0 หมายถึง บิต 0 ของพอร์ตเป็นบิตนัยสำคัญต่ำสุด (Least Significant Bit) และบิต P1.7 หมายถึง บิตที่ 7 ของพอร์ต 1 ซึ่งเป็นบิตนัยสำคัญสูงสุด (Most Significant Bit)

ส่วนที่ 1 Port 1 Latch ซึ่งจะมีการทำงานเหมือนส่วนที่ 1 ของพอร์ต 0 ส่วนที่ 2 คือ Port 1 Driver นี้จะมีตัวต้านทานต่ออยู่เป็น Internal Pull Up พอร์ต 1 นี้จะใช้ทำหน้าที่เป็นตัวรับ-ส่งข้อมูลเท่านั้น ข้อมูลที่ส่งออกมาทางพอร์ต 1 จะถูก Latch ไว้แล้วส่งออกไปทางแต่ละขาก่อนที่จะอ่านข้อมูลเข้าไปทางพอร์ต 1 จะต้องเขียน 1 ไปยังทุกบิตของพอร์ต 1 เสียก่อนเพื่อให้ FET อยู่ในสภาวะ OFF ก่อน มิฉะนั้นแล้วถ้ามีข้อมูล 0 ส่งมาค้างอยู่ก็จะทำให้ FET อยู่ในสภาวะ ON ดังนั้นถ้าสัญญาณภายนอกส่งเข้ามาที่ขานี้ก็จะถูกลัดลงกราวน์ โดยไม่สนใจว่าสภาวะลอจิกของสัญญาณที่เข้ามาจะเป็นอะไร ข้อมูลที่อ่านเข้าไปจะเป็น 0 เสมอ

Port 3 คือ ขา P3.0 ถึง P3.5 และ P3.7 ตามลำดับ โดยข้อมูลที่เขียนมายังพอร์ต 3 ทาง Internal Bus เหมือนกับพอร์ตอื่น ๆ และ พอร์ต 3 จะมี Internal Pull Up อยู่ทุกบิตแต่พอร์ต 3 นี้แต่ละบิตจะใช้ในการทำงานอื่นได้โดยใช้คำสั่งควบคุมการทำงาน และจะมีส่วนที่มีสัญญาณ Alternative Output Function ที่สร้างมาจากส่วน Timing and Control สัญญาณ Alternative Output Function เป็นสัญญาณที่ส่งออกในกรณีที่ใช้พอร์ต 3 ทำงานในฟังก์ชั่นอื่นและจุด Alternative Input Function เป็นจุดที่จะเอาสัญญาณไปเข้ากับส่วนอื่นตามการทำงานของบิตนั้น แต่ละบิตของพอร์ต 3 จะมีฟังก์ชั่นอื่นดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับส่งข้อมูลแบบอนุกรม

P3.1/TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

P3.2/INTO(External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.4/TO (Timer/Counter 0 External Input) ขารับสัญญาณเข้าไปในวงจร Timer/Counter 0 ที่ทำหน้าที่นับจำนวนไซเคิลของสัญญาณ T0 นี้หรือสัญญาณนาฬิกาก็ได้

P3.5/T1 (Timer/Counter 1 External Input) ขารับสัญญาณเข้าไปยัง Timer/Counter 1 ซึ่งมีการทำงานเหมือนกับ T0

P3.7/RD ขาสัญญาณข้อมูล

RST ขารีเซ็ทขานี้จะใช้ทำการรีเซ็ทการทำงานของ 89c2051 ที่ขา RST ภายใน 89c2051 จะมีตัวต้านทานต่อระหว่างขานี้กับกราวด์ (Ground) ถ้าป้อนสัญญาณที่สภาวะลอจิก 1 เข้าไปที่ขานี้จะเป็นการรีเซ็ทการทำงานของไมโครคอนโทรลเลอร์ดังนั้นจึงสามารถต่อตัวเก็บประจุภายนอกระหว่างขา RST กับไฟเลี้ยง +5 โวลท์เพื่อให้เกิดการรีเซ็ทเมื่อเริ่มป้อนไฟเลี้ยงให้กับ 89c2051 ซึ่งเรียกว่า "Power On Reset" การรีเซ็ทจะทำให้ค่าในรีจิสเตอร์ต่าง ๆ เปลี่ยนไปเป็นค่าดังในตารางที่ 4.1

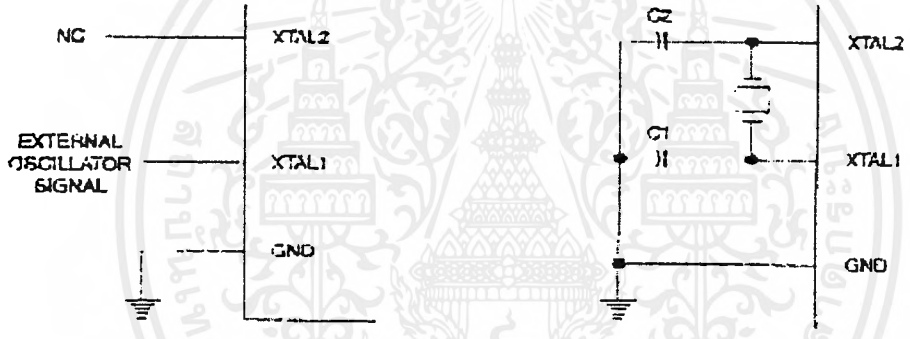
Table 1. AT89C2051 SFR Map and Reset Values

0F8H							0FFH
0F0H	B 00000000						0F7H
0E8H							0EFH
0E0H	ACC 00000000						0E7H
0D8H							0DFH
0D0H	PSW 00000000						0D7H
0C8H							0CFH
0C0H							0C7H
0B8H	IP XXX00000						0BFH
0B0H	P3 11111111						0B7H
0A8H	IE 0XX00000						0AFH
0A0H							0A7H
98H	SCON 00000000	SBUF XXXXXXXX					9FH
90H	P1 11111111						97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	8FH
80H		SP 00000111	DPL 00000000	DPH 00000000			PCON 0XXX0000	87H

ตารางที่ 4.1 ค่ารีจิสเตอร์เมื่อเกิดการรีเซ็ท

XTAL 1 ขาที่ 5 ขานี้จะต่อเข้ากับขาของ Inverting Amolifier (วงจรขยายแบบป้อนกลับเฟสสัญญาณ) ที่ประกอบเป็นวงจรออสซิลเลเตอร์ ซึ่งจะเห็นว่าวงจรภายในออสซิเลเตอร์ NAND Gate จะทำหน้าที่เป็นวงจรขยายแบบกลับเฟสของสัญญาณที่จะควบคุมให้มีการออสซิลเลตหรือไม่ ก็ขึ้นกับสัญญาณ PD ซึ่งต่อมาจากบิต PD ของรีจีสเตอร์ POON ถ้าต้องการใช้สัญญาณนาฬิกา (Clock Signal) จากภายนอกมาเป็นสัญญาณนาฬิกา ควบคุมการทำงานของ 89c2051 ก็ให้ป้อนสัญญาณเข้ามาที่จุดนี้แต่ถ้าต้องการใช้วงจรออสซิเลเตอร์ภายในก็ให้ต่อ Crytal

XTAL 2 ขาที่ 4 ขานี้เป็นจุดเอาท์พุทของวงจรขยายแบบกลับเฟสสัญญาณที่ประกอบเป็นวงจรออสซิเลเตอร์ (อินพุทคือขา (XTAL1) ถ้าจะใช้สัญญาณนาฬิกาที่สร้างมาจากภายนอกมาเป็นสัญญาณนาฬิกาของ 89C2051 แล้ว ให้ปล่อยขานี้ลอยไว้แล้วป้อนสัญญาณนาฬิกาจากภายนอกเข้ามาที่ขา XTAL1 ดังรูปที่ 4.2



รูปที่ 4.2 การต่อ Oscillator

4.2 การสื่อสารพอร์ตอนุกรม RS-232

ลักษณะของการส่งข้อมูลแบบอนุกรมนั้น ข้อมูลจะถูกส่งออกมาทีละบิต จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ช่องสัญญาณในการส่งข้อมูลอาจจะใช้เพียง 1 หรือ 2 ช่องสัญญาณเท่านั้น ทำให้ค่าใช้จ่ายในการสื่อสารถูกกว่าแบบขนาน แต่อัตราการส่งข้อมูลจะล่าช้ากว่าการส่งแบบขนาน ในการส่งข้อมูลแบบอนุกรม ข้อมูลที่ต้องการส่งจะอยู่ในลักษณะเป็นไบต์จะถูกทยอยส่งทีละบิต และทางอุปกรณ์รับจะต้องรับข้อมูลเข้ามาทีละบิต แล้วมารวมกันเป็นไบท์ซึ่งทางด้านอุปกรณ์รับจะต้องคอยตรวจสอบว่า บิตใดเป็นบิตเริ่มแรกของไบท์นั้น การตรวจสอบขึ้นอยู่กับรูปแบบของรหัสของบิตที่ใช้ ซึ่งในการรับส่งข้อมูลแบบอนุกรมระหว่างไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกนั้น จำเป็นต้องมีมาตรฐานในการรับส่ง ซึ่งมาตรฐานที่นิยมมากที่สุดก็คือ มาตรฐาน RS-232

4.2.1 มาตรฐาน RS-232

เพื่อที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกันทำงานร่วมกันได้ มาตรฐานหลายชนิดจึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันอย่างกว้างขวางที่สุด คือ RS-232C ซึ่งโดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรมอยู่ในตัวอยู่แล้ว และจะทำหน้าที่รับส่งข้อมูลในแบบอนุกรม

ตามจุดประสงค์ของมาตรฐาน RS-232C นั้นเพื่อจะสามารถเชื่อมต่อกันระหว่างอุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment : DTE) เช่น พอร์ตของคอมพิวเตอร์หลักหรืออุปกรณ์ปลายทาง กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment : DCE) หมายถึงอุปกรณ์ที่สามารถแปลงรูปคลื่นดิจิตอลไปเป็นสัญญาณที่เหมาะสมสำหรับการส่งผ่านสายโทรศัพท์หรือ ตัวกลางการสื่อสารอย่างอื่น โดยกระบวนการที่เรียกว่า มอดูเลชั่น (Modulation)

4.2.2 การอินเตอร์เฟสตามมาตรฐาน RS-232C

มาตรฐาน RS-232C ใช้สายสัญญาณเพียงเส้นเดียวในการส่งสัญญาณ โดยสัญญาณที่ส่ง ๆ ไปได้ทิศทางเดียว สำหรับความเร็วและระยะทางของการเชื่อมต่อ RS-232C สามารถเชื่อต่อการถ่ายโอนข้อมูลได้จาก 0 – 20K bps(บิตต่อวินาที) ซึ่งเพียงต่อสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบอด (Baud Rate) 110 ถึง 9600 ความยาวของสายเชื่อมสัญญาณระหว่าง DTC และ DCEโดยสัญญาณตามมาตรฐานของ RS-232 จำกัดอยู่แค่ 50 ฟุตหรือประมาณ 15 เมตร ซึ่งเพียงพอสำหรับการสื่อสารไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอก สำหรับแรงดันของระดับสัญญาณจะถูกกำหนดลงในสองบริเวณคือ

- แรงดันไฟบวก (สถานะ SPACE) อยู่ระหว่าง +5 ถึง +15 โวลท์สำหรับเอาท์พุต และระหว่าง +3 และ +15 โวลท์สำหรับอินพุท ความแตกต่างมีไว้เผื่อกรณีที่แรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณ ในทำนองเดียวกัน
- แรงดันไฟลบ (สถานะ MARK) ถูกกำหนดไว้ระหว่าง –5 ถึง –15 โวลท์สำหรับเอาท์พุต และ –3 ถึง –15 โวลท์สำหรับอินพุท

ซึ่งจะเห็นได้ว่า ถ้าให้สายสัญญาณยาวเกินไป ระดับแรงดันไฟฟ้าจะตกลงเกินขอบเขตที่กำหนด นอกจากนี้ ความจุไฟฟ้าที่เกิดขึ้นจะมีผลกับคุณภาพของสัญญาณ โดยทำให้การเปลี่ยนสถานะจากแรงดันไฟบวกไปเป็นแรงดันไฟลบนั้นไม่ชัดเจน เนื่องจาก RS-232C ไม่ได้ออกแบบให้นำไปใช้กับระยะทางไกล ดังนั้นถ้าอุปกรณ์อยู่ห่างกันมาก อาจจำเป็นต้องใช้โมเด็มหรือวิธีการอื่น ๆ แทน

บทที่ 5

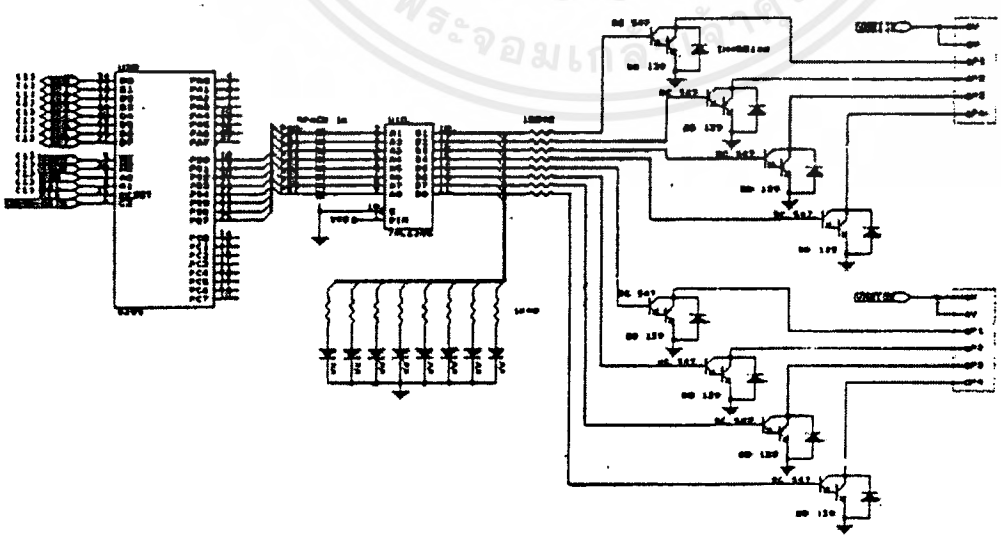
หลักการออกแบบและสร้าง

ในส่วนของเทอมนี้เราได้ทำการออกแบบและสร้างในส่วนของ ชุดควบคุมกล้องและส่วนของประมวล(Server Process) และจะได้อธิบายดังต่อไปนี้

5.1 ชุดควบคุมกล้อง (Camera Control) โดยจะแบ่งเป็นสองส่วนคือ

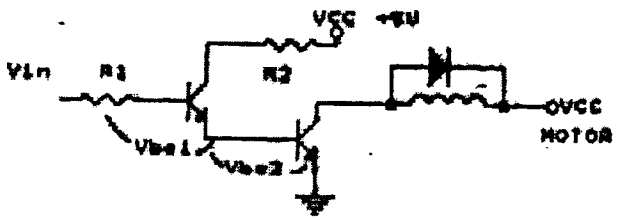
5.1.1 ส่วนฐานจับกล้อง ได้ออกแบบตามลักษณะกล้องที่ใช้งานคือจะเป็นกล้อง Color QuickCam ซึ่งเป็นกล้องดิจิตอลที่นิยมใช้ในปัจจุบัน มีการอินเตอร์เฟสกับคอมพิวเตอร์โดยผ่านทางพอร์ตขนาน คุณภาพการแสดงภาพ จะเป็นภาพสีคุณภาพจัดว่าใช้ได้ เมื่อมาใช้กับโครงงานนี้ และในส่วนของ Stepping Motor ที่ใช้จะมีขนาดเล็กและหาซื้อได้ง่ายราคาถูก(หาได้ตามร้านที่บ้านหม้อ) ซึ่งมีคุณสมบัติคือ ใช้ไฟเลี้ยง 12 Volt และมี 1.8 ดีกรีต่อ1 สเต็ป จำนวนสองตัว เพื่อทำงานในการหมุนทางด้านแนวซ้ายขวา กับ แนวบนล่าง

5.1.2 ส่วนไมโครคอนโทรเลอร์ โดยจะมีอยู่สองส่วนด้วยกันคือ ส่วนที่เป็นไมโครคอนโทรเลอร์ซึ่งโครงงานนี้ได้เลือกใช้ Single Board Version 4 ซึ่งมี Z-80 เป็นหน่วยประมวลผลกลาง (CPU) ซึ่งทางผู้จัดทำได้มีอยู่แล้วจึงได้นำมาใช้ในการประมวลผลและควบคุมการทำงานในการควบคุมการหมุนของชุดจับกล้อง และอีกส่วนคือ ชุดขับStepping Motor เพื่อไปขับStepping Motor ทั้งสองตัว ซึ่งรูปวงจรก็ได้แสดงไว้ในรูปข้างล่างดังต่อไปนี้



รูปที่ 5.1 แสดงส่วนของวงจรขับ Stepping Motor

การคำนวณ การ Bias ภาคขยายกระแสไฟฟ้าให้ได้กระแสแต่ละเฟสตามต้องการ



รูปที่ 5.2 แสดงส่วนของวงจรขับ Stepping Motor ในหนึ่งเฟส

สมมุติให้ B2 คือค่าอัตราการขยายต่ำสุดของ Q1 และ IC2 คือ กระแสขับสูงสุดต่อ จะได้

$$R2 < \frac{(5\text{-}VBE2)}{Ib2} = \frac{(5\text{-}VBE2)\times B2}{Ic2}$$

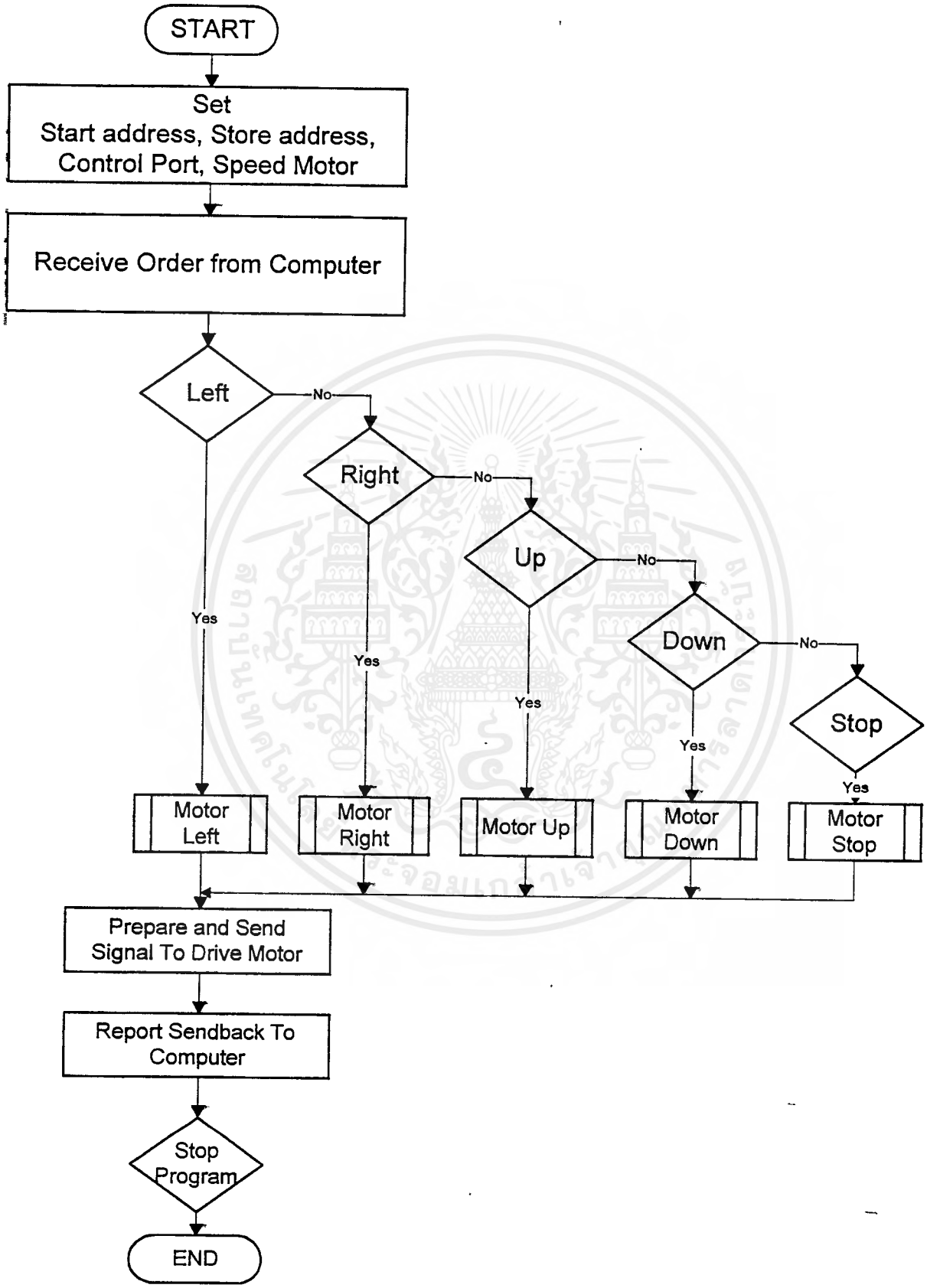
สำหรับ R1 สมมุติให้ B คือ ค่าอัตราการขยายกระแสต่ำสุดของ Q1 และ VIN มีค่าประมาณ 4V สำหรับ LOGIC "1" จะได้

$$R1 < \frac{(4 - VBE1 - VBE2)}{Ib1}$$

$$= \frac{(4 - VBE1 - VBE2)\times B1}{Ic1}$$

$$= \frac{2.8\ B1}{Ib2}$$

สำหรับโปรแกรมที่เอาไว้ควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ทั้งสองตัว ซึ่งโปรแกรมนี้จะถูกเก็บเอาไว้ที่บอร์ดควบคุมเป็นของ Single Board ซึ่งมีหน่วยประมวลผลกลางเป็น 89C2051 โปรแกรมแอสเซมบลี้ หลักการออกแบบของเขียนโปรแกรมนี้ก็เริ่มจาก เนื่องจาก Single Board ที่ใช้มีConnector User Poet ที่เอาไว้ใช้อยู่หนึ่งพอร์ต(พอร์ต 1) และเราได้นำพอร์ต 1 มาใช้งาน(หมาก็หมายความว่าโปรแกรมก็ส่งค่าเอาท์พุท มาที่พอร์ตหมายเลข 1 เท่านั้น และสามารถดูได้จาก Flowchart ด้านล่างนี้



รูปที่ 5.3 Flowchart แสดงการทำงานของโปรแกรม

เริ่มจากการที่เราต้องกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างที่จำเป็นในการกำหนดรูปแบบการทำงาน โดยเราจะใช้งานของ Single Board ที่พอร์ต 1 และค่าอื่นๆอีกเช่น ความเร็วของสเต็ปปิ้งมอเตอร์ ตำแหน่งสูงสุดและต่ำสุดของการหมุน ตำแหน่งแอดเดรสต่างๆที่ใช้ในการเก็บข้อมูลของโปรแกรม เมื่อได้กำหนดค่าพารามิเตอร์ต่างๆแล้ว ก็จะเป็นส่วนของการรอรับค่าจากคอมพิวเตอร์โดยผ่านทาง RS 232 ซึ่งเป็นพอร์ตอนุกรม การรับค่าจาก คอมพิวเตอร์จะรอรับเพียงแค่Byteเดียว ซึ่งจะเป็นโค้ดที่ใช้ในการกำหนดให้Single Board ทำงานตามโค้ดที่ส่งมา ซึ่งจะมีทั้งหมด 5 โค้ดด้วยกัน คือ 36h สำหรับการหมุนขวา 34h สำหรับการหมุนซ้าย 38h สำหรับการหมุนบน 32h สำหรับการหมุนลงล่าง 35h สำหรับสั่งให้หยุด และสุดท้ายคือ 37h สำหรับ ปรับค่าการหมุนทุกตัวให้อยู่ที่ตำแหน่งเริ่มต้น

เมื่อได้รับโค้ดที่ส่งมาให้แล้วก็จะทำการตรวจสอบว่าเป็นโค้ดสำหรับอะไร ถ้าโค้ดที่ได้รับมาถูกต้องตามที่ได้กำหนดไว้หรือเปล่า ถ้าถูกต้องก็จะทำการเข้าไปทำโปรแกรมย่อยที่ใช้ในการเคลื่อนที่แต่ล่ะส่วนของมอเตอร์ และในการทำงานในส่วนของโปรแกรมย่อยนี้จะมีการตรวจสอบตำแหน่งของมอเตอร์ว่าอยู่ในขอบเขตที่กำหนดไว้ในตอนแรกหรือเปล่า ถ้ามากไปหรือน้อยไปก็จะส่งข้อมูลมาเตือนที่คอมพวเตอร์ว่าได้หมุนเกินพิกัดแล้วก็จะทำการหยุดหมุนทันทีและในระหว่างการหมุนก็จะทำการส่งตำแหน่งของมอเตอร์กลับไปยังคอมพิวเตอร์ตลอดการหมุน Single Board ก็จะทำการหมุนไปเรื่อยจนกว่าจะได้รับคำสั่งให้หยุดหมุนจากคอมพิวเตอร์ ลักษณะโปรแกรมนี้จะเป็นโปรแกรมวนลูปไม่รู้จบทำไปเรื่อยๆจนกว่าจะมีการรีเซ็ทการทำงาน และในส่วนโปรแกรมย่อยที่ทำหน้าส่งสัญญาณออกไปควบคุมการหมุนมอเตอร์จะประกอบด้วยการทำงานสองส่วนคือการนำรูปแบบการหมุนของมอเตอร์แต่ละตัว(รูปแบบของการหมุนแตกต่างกันเนื่องจากการใช้งานที่แตกต่างกัน)มาทำการประมวลผลเพื่อให้ได้สัญญาณที่จะส่งออกไปขับมอเตอร์ทั้งสองตัวและอีกส่วนคือส่วนที่คอยหน่วงเวลาสำหรับการขับมอเตอร์ออกไปแต่ล่ะสเต็ปของการหมุน ถ้าพูดง่ายๆก็จะเป็นการควบคุมความเร็วของการหมุนของมอเตอร์ทั้งสองตัว เมื่อได้สัญญาณและความเร็วในการหมุนที่แน่นอนก็จะส่งข้อมูลออกไปที่พอร์ตของ Single Board ที่ได้กำหนดไว้ ก็คือพอร์ต 1 เบอร์ 20h เมื่อได้ส่สัญญาณไปแล้วก็จะทำการรายงานผลของการหมุนไปที่คอมพิวเตอร์อีกครั้งหนึ่ง โดยจะนำเอาข้อมูลของการหมุนที่เก็บเอาไว้ที่ตำแหน่งที่กำหนดในตอนต้นโปรแกรมส่งออกไป โดยข้อมูลที่ส่งออกไปก็มี ตำแหน่งซ้ายขวา ตำแหน่งบนล่าง ความเร็วมอเตอร์ และ ข้อมูลเตือนข้อผิดพลาด(เกิดจากการที่เราสั่งหมุนเกิดพิกัดตำแหน่งที่กำหนดไว้)รวมทั้งหมดก็ 4 Byte โดยผ่านทางพอร์ตอนุกรม RS-232 ไปยังคอมพิวเตอร์ และโปรแกรมที่คอมพิวเตอร์ก็จะนำข้อมูลเหล่านี้มาวิเคราะห์และแสดงผลต่อไป

สำหรับในส่วนของคอมพิวเตอร์ที่เป็น Server Control จะใช้ PC ธรรมดาที่มีคุณสมบัติพอใช้ได้ (โครงงานนี้ใช้ PentuimMMx 233 Ram 32M)เพื่อที่ความสามารถที่จะรันโปรแกรม Delphi3 ได้อย่างไม่มีปัญหา ซึ่งในส่วนของเทอมนี้จะทำในส่วนแสดงผลและควบคุมการทำงานของการหมุนStepping Motor และอีกส่วนหนึ่งคือ การแสดงสัญญาณภาพจากกล้องวิดีโอดิจิตอลมาแสดงบนจอภาพ เริ่มจากการหาโปรแกรมที่สามารถเขียนและทำงานได้บนWindows95 ทางผู้จัดทำได้พิจารณาแล้วโปรแกรม Delphi3 ตรงกับความต้องการเบื้องต้นที่ต้องการ และได้ทำการเขียนโปรแกรมที่ใช้สำหรับในเทอมนี้

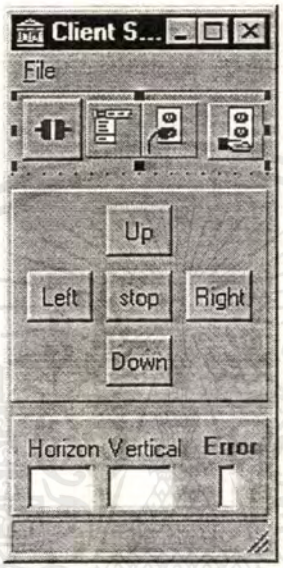
ในการทำงานของโปรแกรมนี้หลักที่สำคัญคือรับข้อมูลจาก Single Board มาทำการประมวลผลและแสดงผล และรับคำส่งจากผู้ใช้งาน (User)ไปควบคุมการหมุนของมอเตอร์อีกที และที่สำคัญคือการแสดงผลภาพจากกล้องวิดีโอ ซึ่งรายละเอียดปลีกย่อยของโปรแกรมได้อยู่ในส่วนของภาคผนวกแล้ว

5.2 โปรแกรมการติดต่อใช้งานระหว่างผู้ควบคุมและผู้ให้บริการ

จากที่ได้ได้อธิบายมาจะเป็นส่วนของการติดต่อระหว่างเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Server กับ ตัวชุดจับกล้องโดยจะผ่านทางพอร์ตอนุกรม RS232 ต่อไปก็จะเป็นการออกแแบบในส่วนของการติดต่อระหว่างผู้ใช้กับส่วนที่เป็น Server โปรแกรมที่ใช้ในโครงงานนี้จะเป็นโปรแกรมในส่วนที่ติดต่อกับผู้ใช้งานมีอยู่สองโปรแกรมคือ โปรแกรมทางด้านผู้ควบคุม(ผู้ใช้งาน) และโปรแกรมทางด้านผู้ให้บริการ(Server) โดยจะเริ่มจากออกแบบโปรแกรมในส่วนที่เป็นผู้ใช้ก่อน โดยจะมีรายละเอียดดังต่อไปนี้

5.2.1 โปรแกรมควบคุมด้านผู้ใช้ (Client Side Program) เนื่องจากโปรแกรมที่ทางผู้จัดทำได้เขียนบนระบบปฏิบัติการ Windows95 ดังนั้นโปรแกรมที่เรานำมาสร้างโปรแกรมควบคุมจะใช้โปรแกรม Delphi3 ซึ่งสามารถทำงานบนระบบนี้ได้ เริ่มจากได้สร้างโปรแกรม โดยจะทำการออกแบบโครงสร้างของโปรแกรมนี้ก่อน โดยคิดจากฟังก์ชั่นการทำงานทั้งหมดที่จำเป็นในการทำงานในการควบคุมการหมุนของกล้องที่อยู่ด้านปลายทางอีกที โดยมีฟังก์ชั่นการทำงานที่เห็นคือ ส่วนของการควบคุมทิศทางการหมุน การแสดงผลตำแหน่งของกล้องและจะบอกความผิดพลาดที่เกิดบนตัวชุดจับกล้องอันเนื่องมาจากสาเหตุ การที่ชุดจับกล้องหมุนเกินตำแหน่งที่ตั้งไว้ที่โปรแกรม โดยโปรแกรมได้ระยะตำแหน่งของการหมุนไว้เป็นดังนี้คือ หมุนได้ 90 องศา จากตำแหน่งจุดศูนย์กลางทั้งสองตัว เมื่อการแจ้งการเกิดข้อผิดพลาดขึ้นมา ผู้ใช้ไม่สามารถที่จะสั่งไปในทิศทางนั้นได้ต้องเปลี่ยนทิศทางการควบคุมให้ให้มีทิศตรงกันข้ามจากเดิม และส่วนต่อมาจะเป็นส่วนของโปรแกรมที่ใช้สำหรับติดต่อกับชุดจับกล้อง โดยชุดจับกล้องตัวนี้จะอยู่คนละที่กับด้านของผู้ใช้นี้ ซึ่งการติดต่อนี้จะควบคุมผ่านเครือข่ายอินเตอร์เน็ตโดยการโปรแกรมก็จะเป็นการส่งรหัสควบคุม

การหมุนไปยังชุดควบคุมกล้องซึ่งอยู่ไกลออกไป และจะคอยรับข้อมูลที่จะบอกตำแหน่งปัจจุบันของมอเตอร์ที่เราได้สั่งไป และถ้าเป็นการสั่งหมุนเกินขนาดที่โปรแกรมได้ตั้งเอาไว้ จะแจ้งให้ผู้ใช้ทราบทันที เมื่อได้กำหนดการทำงานทั้งหมดแล้วก็จะเป็นการเริ่มสร้างส่วนของโปรแกรมชุดนี้โดยจะมีรูปแบบตามรูปข้างนี้



รูปที่ 5.4 รูปแสดงการวางตำแหน่ง Component ต่างๆ

โดยทำการจัดรูปแบบของโปรแกรมตามรูปที่ 5.4 แล้วเขียนโปรแกรมต่างๆที่ได้กำหนดไว้ตามหน้าที่การใช้งาน โดยComponent ที่สำคัญในส่วนนี้ก็คงเป็นตัว TClientSocket และ TServerSocket จะเป็นตัวทำงานในด้านอินเตอร์เน็ตทั้งหมด ส่วนชุดแป้นกดจะแบบชุดควบคุมทิศทางการหมุนของมอเตอร์ และสุดท้ายจะเป็นชุดแสดงผลตำแหน่งของมอเตอร์ที่เราได้สั่งไป และตัวแสดงความผิดพลาดอันเนื่องมาจากการหมุน และได้ทำการเขียนโปรแกรมการทำงาน ซึ่งได้แสดงเอาไว้ที่ภาคผนวกแล้ว

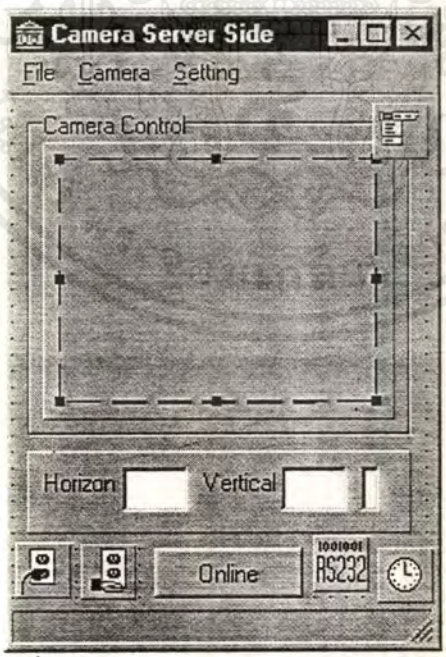
5.2.2 โปรแกรมด้านให้บริการ(Server Side Program) โดยในส่วนนี้จะเป็นส่วนที่เป็นตัวจัดการเกี่ยวกับการควบคุม การรับส่งข้อมูล การจัดการเกี่ยวกับภาพที่จะส่งไปยังจุดหมายปลายทาง โดยจะมีชุดจับกล้องรวมอยู่ที่ส่วนนี้ด้วย แต่เราจะไม่พูดถึงในส่วนนี้เพราะได้กล่าวมาแล้วในตอนต้น โดยจะกล่าวถึงโปรแกรมที่ใช้ควบคุมในส่วนนี้ จะเป็นตัวคอยควบคุมติดต่อกับชุดจับกล้องโดยผ่านทางพอร์ตอนุกรมRS232 และรับภาพจากกล้องวิดีโอโดยผ่านทางพอร์ตขนาน และสุดท้ายจะเป็นการรับคำสั่งควบคุมจากผู้ใช้แล้วรายนงานผลของตำแหน่งของมอเตอร์และข้อผิดพลาดที่เกิดขึ้น รวมทั้งการส่งภาพไปยังปลายทางด้วย การโปรแกรมจะเริ่มจากการกำหนดลักษณะของหน้าที่การทำงานทั้งหมด อาจจะแบ่งหน้าที่การทำงานภายในโปรแกรมนี้ออกได้เป็น 3 ส่วนใหญ่ได้แก่

1) ส่วนการติดต่อกับชุดจับกล้อง โดยที่ชุดจับกล้องที่ได้ออกแบบจะติดต่อโดยผ่านทางพอร์ตอนุกรมRS232 แต่ในการเขียนโปรแกรมโดยใช้ Delphi3 นั้น Component ที่เกี่ยวกับRs232นี้ไม่มีมาให้ในส่วนของโปรแกรมที่เป็นสแตนดาร์ดทั่วไป จึงได้ใช้ Component ที่ได้Downloadมาและเมื่อนำมาทดลองใช้ก็สามารถทำงานในส่วนนี้ได้อย่างดี

2) ส่วนในการรับภาพและบันทึกภาพจากกล้องวิดีโอ ในส่วนนี้ถือว่าเป็นส่วนที่สำคัญมาก เพราะจุดประสงค์หลักของโครงงานนี้จะเกี่ยวกับการส่งภาพ และนี่ก็เป็นปัญหาที่คล้ายกับการโปรแกรมในส่วนแรกที่ได้กล่าวมาในข้อ 6.4.1 คือ Component ที่ทำการรับภาพไม่มี สรุปก็คือต้องไปหาDownload จากอินเตอร์เน็ตเหมือนที่ผ่านมา และเมื่อได้componentนี้มาก็ทำการทดสอบการทำงาน ทั้งการรับภาพจากกล้องวิดีโอ และการบันทึกภาพเพื่อที่จะส่งต่อไป ปรากฏว่าทำงานได้ดังที่ตั้งใจเอาไว้

3) ส่วนในการติดต่อผ่านอินเตอร์เน็ต ก็จะใช้Component ที่ได้มีให้แล้วในDelphi3 โดยที่ใช้จะเป็นตัว TClientSocket กับ TServerSocket ทั้งสองตัวนี้จะเป็นตัวทำหน้าที่ในการรับคำสั่งควบคุม และส่งข้อมูลตำแหน่งของมอเตอร์ไปยังผู้สั่งงาน การทำงานในส่วนนี้ก็สามารถทำงานได้ถูกต้อง

เมื่อได้เขียนโปรแกรมทั้งหมดแล้วในส่วนนี้ก็จะมีการวางรูปแบบดังรูปต่อไปนี้



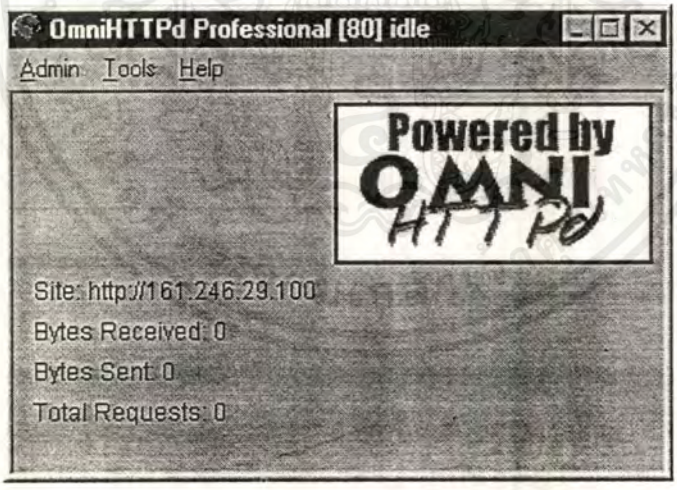
รูปที่ 5.5 แสดงการวางรูปแบบของโปรแกรมด้านServer

จากรูปที่ 5.5 จะเห็นได้ว่าจะมี Component ที่ถูกเพิ่มเข้ามาอยู่สองตัว คือ RS232 และ Video Capture ซึ่งทั้งสองตัวนี้ไม่มีอยู่ในโปรแกรมที่เป็นมาตราฐานของDelphi3 จึงต้องทำการไปDownload มาซึ่ง

ก็ได้แสดงดังรูปข้างบนนี้แล้ว ในส่วนของตัวซอสโค้ดโปรแกรมได้แสดงไว้ในส่วนของภาคผนวกแล้ว

4)ส่วนของการส่งภาพไปยังผู้ควบคุม(ผู้ใช้งาน) จะใช้การส่งภาพผ่านอินเตอร์เน็ต ใช้วิธีการส่งภาพแบบเอกสารHTML โดยจะใช้โปรแกรมที่เป็น Browser จำพวก Internet Explorer Netscape หรือ Opera เป็นตัวแสดงเอกสารที่ได้ส่งไป(รวมทั้งรูปภาพที่ได้จากกล้องวิดีโอด้วย) สาเหตุที่ต้องใช้การส่งรูปภาพโดยวิธีนี้ก็เพราะว่า ตอนแรกได้วางแผนที่จะส่งภาพและรับภาพโดยการเขียนโปรแกรมขึ้นมาเอง แต่เนื่องจากข้อจำกัดทางเวลาและความซับซ้อนของการเขียนโปรแกรมในส่วนการส่งและรับภาพผ่านอินเตอร์เน็ตนั้นมีความลำบากมาก ทางผู้จัดทำได้เขียนและพยายามหาข้อผิดพลาดที่เกิดขึ้นแต่ก็ไม่ประสบความสำเร็จ จึงได้หันมาใช้วิธีที่ใช้อยู่ในตอนนี้ ซึ่งอาจดูว่าไม่สะดวกหรือถูกจุดประสงค์ที่ได้วางไว้ในตอนแรกเท่าไหร่นัก แต่ผลที่ได้รับก็อยู่ในระดับที่ยอมรับได้ จึงหันมาใช้วิธีน่าจะสะดวกและง่ายกว่าเดิม

เมื่อได้ข้อตกลงที่จะใช้วิธีที่จะส่งภาพที่ได้กล่าวในขั้นต้น โดยจะเริ่มจากการจะต้องหาตัวที่ทำหน้าที่เป็น Web Server โดยทางผู้จัดทำได้เลือกใช้เครื่องที่ใช้ Windows95 จึงจำเป็นต้องหาโปรแกรมที่เป็นตัว Web server โดยได้เลือกใช้โปรแกรม OmniHttpd 2.0 Alphaเป็นตัว Web server



รูปที่ 5.6 แสดงโปรแกรม OmniHttpd 2.0 ซึ่งนำมาใช้เป็น Web Server

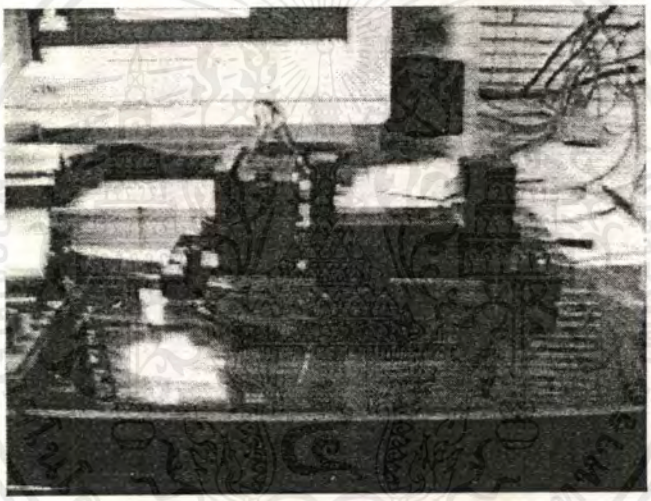
และได้ทำการกำหนดค่าพารามิเตอร์ต่างๆที่โปรแกรมต้องการในการทำงานก็พร้อมที่จะทำงานได้แล้ว ต่อไปก็ต้องเขียนโปรแกรมเอกสารHTMLที่จะทำการส่งภาพไปให้ผู้ใช้ โดยต้องเขียนโปรแกรมเพื่อที่จะให้พวกBrowserที่เปิดจะต้องทำการReloadภาพที่ส่งไป โดยสามารถกำหนดได้จากโปรแกรม(ขั้นแรกกำหนดไว้ที่5วินาที) เนื่องจากภาพที่ได้ทำการบันทึกโดยในส่วนของโปรแกรมด้านServer ฟอร์แมทภาพจะเป็นลักษณะบิตแมท (Bitmap,BMP file) ภาพที่บันทึกได้จึงมี

ขนาดใหญ่ ทำให้ในการส่งภาพจะใช้เวลานานเพราะข้อจำกัดทางด้านความเร็วและระยะทางระหว่างผู้ใช้และServer จากการทดสอบถ้ากำหนดไว้ในการReloadแต่ละครั้งประมาณ5วินาทีสามารถแสดงผลได้อย่างมีประสิทธิภาพ และในส่วนของเอกสารHTML ได้แสดงไว้ภาคผนวกแล้ว

บทที่ 6
สรุปผลการทดลอง

จากการที่ได้ลงมือสร้างโครงงานนี้ก็ได้ทดสอบคุณสมบัติต่างๆของโครงงานนี้ว่าได้ตามที่ออกแบบหรือคิดไว้หรือเปล่าโดยจะแยกผลการทดลองแต่ละส่วนเพื่อที่ให้เห็นคุณสมบัติกันอย่างชัดเจน เริ่มที่ชุดควบคุมกล้องก่อนผลการทดลองมีดังนี้

6.1ชุดจับกล้อง ซึ่งเป็นคล้ายๆแขนกลทางผู้จัดทำได้ทำการออกแบบมาเฉพาะกล้องเท่านั้น (Color Quick Cam) โดยโครงสร้างทำมาจากพลาสติกเพราะมีราคาถูกและง่ายต่อการสร้างมาก ได้ทำตามแบบที่ได้ออกแบบไว้ทุกอย่างและชุดนี้



รูปที่ 6.1 แสดงโครงสร้างของชุดจับกล้อง

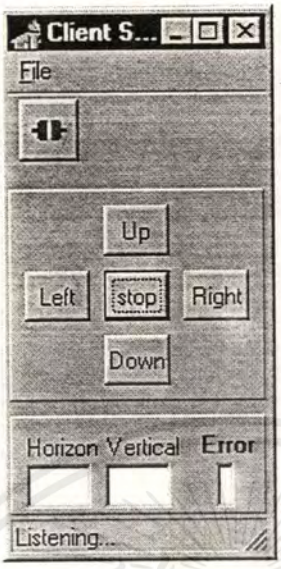
จะใช้น็อตทั้งหมดในการจับยึดของแต่และหลีกเลี่ยงการใช้กาวเพราะไม่สะดวกต่อการปรับปรุงโครงสร้างใหม่ ในกรณีที่ต้องการเปลี่ยนโครงสร้างและปรับปรุงให้ดีขึ้น ดังนั้นการสร้างส่วนของชุดจับกล้องนี้จึงใช้เวลานานพอสมควร หลังจากการประกอบมอเตอร์เข้าไปเรียบร้อย เราได้ทำการทดสอบการเคลื่อนที่ของชุดจับกล้องในแต่ล่ะส่วน(โดยใช้ชุดขับที่โปรแกรมทดสอบเบื้องต้นเอาไว้แล้ว) เมื่อทดสอบก็ได้พบว่า จะได้ความเร็วที่เหมาะสมที่ความเร็วหนึ่ง เพราะถ้าเร็วไปมอเตอร์จะมีแรงน้อยมาก(ไม่พอที่จะรับภาระโหลดได้) และได้ทดสอบรูปแบบของการขับสเต็ปปิ้งมอเตอร์ จะได้รูปแบบการขับคือ มอเตอร์ที่ทำหน้าที่ในการหมุนซ้าย-ขวาจะขับแบบเฟสเดียว และมอเตอร์ที่ทำหน้าที่ในการหมุนบน-ล่างจะขับแบบสองเฟส (เฟสคู่)คือทั้งสองแบบจะให้แรงทอร์คที่ต่างกัน(แบบที่สองจะให้มากกว่า)เพราะว่าในแบบที่สองต้องรับน้ำหนักของกล้องด้วยจึงจำเป็นต้องจัดรูปแบบเพื่อให้แรงเพียงพอ เมื่อทุกอย่างถูกจัดเอาไว้อย่างลงตัว ก็จะเป็นการทดสอบการหมุน ผลคือ การ

หมุนของชุดจับกล้องมีการเคลื่อนไหวที่สมบูรณ์อยู่ในขั้นใช้ได้ ทั้งสองแกน จะติดก็ตรงที่ในจังหวะในการเริ่มหมุนครั้งแรกจะมีอาการกระตุกนิดหน่อยแต่เพื่อดูภาพรวมๆของการเคลื่อนไหวก็จัดว่าใช้ได้

6.2 ชุดควบคุมการหมุน ซึ่งประกอบไปด้วย Single Board ต่ออินเตอร์เฟซกับชุดขับมอเตอร์ จะเริ่มจากชุดขับมอเตอร์ก่อนโดยจะมีอยู่ 2 ชุด(ขับมอเตอร์สองตัว)แต่ตัวต้องใช้ชุดขับทั้งหมดอีก 4 ชุด เพราะมอเตอร์จะมีตัวละ 4 เฟส เมื่อเราได้ทำการทดสอบการทำการโดยการป้อนสัญญาณควบคุมไปที่ภาคอินพุทเมื่อชุดขับมอเตอร์นี้ทำงานจะกินกระแสประมาณ 700mA/เฟส เมื่อได้ทดสอบป้อนสัญญาณไปเรื่อยๆได้ระยะเวลาหนึ่ง จะไม่มีความร้อนที่เกิดบนทรานซิสเตอร์เพราะได้คำนวณปริมาณกระแสที่ต้องการให้ไหลผ่านทรานซิสเตอร์ให้อยู่ในย่านที่ปลอดภัยไว้แล้วจึงไม่มีปัญหาเนื่องจากความร้อนและการทดสอบโปรแกรมที่ติดตั้งบน Single Board เพื่อที่จะทำหน้าที่ขับมอเตอร์ เราได้ทำการทดสอบเบื้องต้นกับโปรแกรม pcplus ก่อนเพื่อจะตรวจสอบการทำงานเบื้องต้นเพื่อให้มั่นใจว่าทำงานได้ถูกต้อง จากการทดสอบโปรแกรมพบว่าการควบคุมสามารถทำได้ดีโดยควบคุมที่คอมพิวเตอร์ ได้สอบสอบฟังก์ชั่นการทำงานต่างๆที่ได้ออกแบบไว้ได้อย่างครบถ้วน การทำงานของชุดควบคุมการหมุนนั้นจึงทำงานได้ตามที่ออกแบบไว้และพร้อมที่จะทำงานร่วมกับโปรแกรมอื่นต่อไป โดยจะพูดถึงต่อไป

6.3 โปรแกรมควบคุมการทำงาน การทดสอบการทำงานส่วนของโปรแกรมใช้งานสำหรับผู้ใช้งาน(User Interface)ตัวโปรแกรมที่จะสร้างโปรแกรมชุดนี้ โดยเลือกใช้โปรแกรม Delphi3 ซึ่งเป็นโปรแกรมที่รันบนWindow95 ได้ โดยจะแบ่งเป็นสองส่วนคือ โปรแกรมส่วนของผู้ใช้ และส่วนโปรแกรมที่ทำหน้าที่เป็น Server โดยมีผลการทดสอบดังนี้

6.3.1 โปรแกรมส่วนของผู้ใช้ ได้ทำการทดสอบการทำงานของโปรแกรม โดยทำการติดต่อไปยังตำแหน่งของคอมพิวเตอร์ที่ได้ติดตั้งกล้องเอาไว้ ก็คือชุดทางด้านserverที่ได้เตรียมเอาไว้แล้ว ผลจากการรันโปรแกรมจะแสดงได้ดังรูปที่ 6.2 ซึ่งเมื่อได้ทำการทดสอบโดยการติดต่อเข้าไปกับชุดด้านServer เมื่อได้ทำการติดต่อและควบคุมการหมุนของมอเตอร์ ผลที่ได้คือสามารถที่จะสั่งการหมุนของมอเตอร์ได้แล้วการรับข้อมูลของตำแหน่งของมอเตอร์ที่ถูกส่งมาจากชุดServerก็ถูกต้องตรงกับตำแหน่งจริงๆที่ได้สั่งงานไป และเมื่อได้ทดสอบสั่งให้มอเตอร์หมุนเกินมุมที่ได้กำหนดเอาไว้ก็จะฟ้องผิดพลาดเนื่องจากการสั่งงานหมุนที่มุมมากไปหรือน้อยไป(คือต้องให้อยู่ในพิกัดที่ได้ตั้งเอาไว้) นั้นก็หมายความว่าการทำงานในส่วนนี้สามารถที่จะทำงานได้ตามที่ได้ออกแบบไว้ดังที่กล่าวมาข้างต้น



รูปที่ 6.2 แสดงผลการทำงานของโปรแกรมควบคุมมอเตอร์

6.3.2 โปรแกรมด้านServer เมื่อได้ทำการทดสอบโดยการให้โปรแกรมทำงาน และผลของการทำงานก็จะเป็นดังรูปต่อไป

รูปที่ 6.3 แสดงผลการทำงานของโปรแกรมด้านServer

จากการทดสอบโปรแกรมนี้ก็สามารถทำงานได้ตามที่ต้องการ ซึ่งเราได้ออกแบบไว้ข้างต้น แต่การทำงานในส่วนนี้ได้ปรับปรุงจากโปรแกรมในเทอมแรกที่ได้มีปัญหาเกิดขึ้นจากการทำงานของComponentที่ทำงานเกี่ยวกับ RS232 และ การรับภาพจากกล้องวิดีโอ โดยการจัดลักษณะการ

ทำงานใหม่ซึ่งเกี่ยวกับฐานเวลาทำให้เมื่อทำงานทั้งสองตัวจึงไม่เกิดปัญหาเกิดขึ้น เราได้ใช้ Component ของvideo ใหม่ซึ่งเมื่อได้ทดสอบแล้วก็ได้คุณสมบัติที่ถูกต้องกับงานมากที่สุดและไม่มีผลต่ออุปกรณ์ตัวอื่นบนโปรแกรมนี้ และเมื่อได้กำหนดเวลาของการบันทึกภาพสำหรับที่จะส่งไปยังปลายทางก็จะได้ผลดังรูป

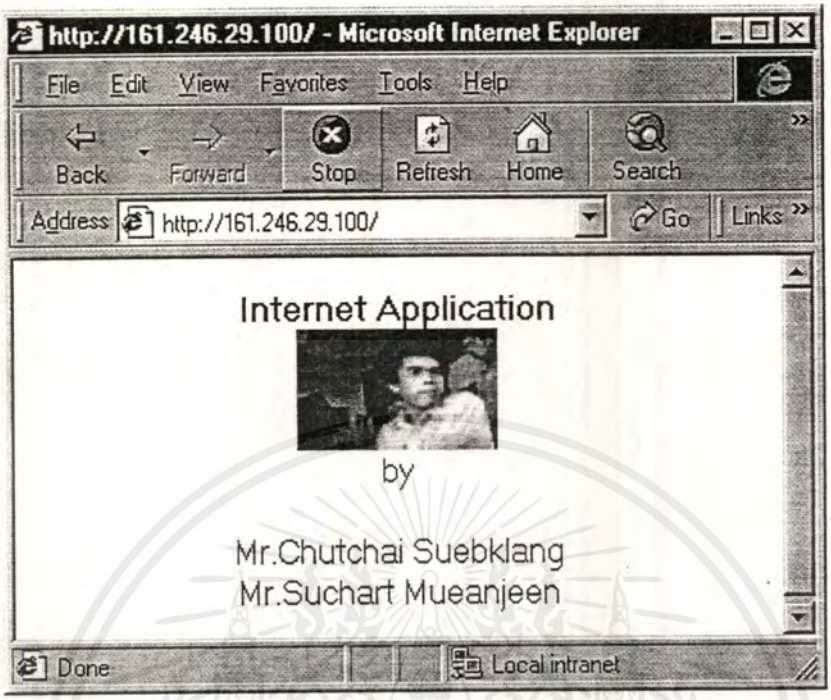
รูปที่ 6.4 แสดงการกำหนดเวลาและตำแหน่งของข้อมูล

เมื่อตรวจสอบฟังก์ชันการทำงานต่างๆของโปรแกรมนี้ก็สามารถที่จะทำงานได้ตามที่ได้ออกแบบไว้ได้ดี

6.3.3 ส่วนของการแสดงภาพ เมื่อทุกอย่างถูกกำหนดไว้เรียบร้อยก็ได้ลองทดสอบการรับภาพจากกล้องวิดีโอจริงๆ โดยภาพที่ได้จะถูกส่งโดย Web server โดยจะใช้โปรแกรม Internet Explorer 5.0 เป็นตัวแสดงภาพ ซึ่งผลที่ได้จะแสดงดังรูปที่ 6.5

จากรูปที่ 6.5 จะเห็นภาพมีขนาดเล็กและแสดงผลเป็นขาวดำ เพราะเนื่องจากต้องการที่จะให้การแสดงภาพเป็นไปด้วยความรวดเร็ว(เพราะภาพต้องผ่านเครือข่ายอินเตอร์เน็ตต้องใช้เวลามาก) จึงจำเป็นลดขนาดของไฟล์ให้มีขนาดเล็กที่สุดเท่าที่จะทำได้ แต่คุณภาพของภาพก็ต้องอยู่ในเกณฑ์ที่ดีด้วย และเมื่อได้ทดสอบหาขนาดและรูปแบบต่างๆของภาพก็ได้ดังที่แสดงข้างบนนี้

สรุปผลการทดลอง จากการออกแบบและสร้างตลอดจนการทดสอบการทำงานในส่วนต่างๆ ก็ได้พบกับปัญหาต่างๆที่เกิดขึ้นในขั้นตอนการสร้าง เนื่องจากโครงงานชุดนี้ค่อนข้างจะเป็นโครงงานที่ได้รวมการประยุกต์ใช้งานด้านต่างๆมารวมกันมาก อาทิเช่น MicroController , Assembly , Delphi3 , Internet ฯลฯ แต่ที่สำคัญที่สุดคงเป็นการที่ได้นำความรู้หลายๆ ด้าน มารวมกันเป็นโครงงานนี้ทำให้เกิดการเรียนรู้ใหม่ที่ไม่ได้เกิดที่ห้องเรียนเพียงอย่างเดียว และเป็นการฝึกการแก้ปัญหาด้วย



รูปที่ 6.5 แสดงผลที่ได้จากการรับภาพจาก Server

การทำงานโดยรวมของโครงงานนี้ยังอยู่ในระดับที่พอยอมรับได้ ยังมีบ้างจุดที่ยังต้องปรับปรุงอีกมาก แต่โครงงานนี้ก็สามารถที่จะทำงานได้ดีในระดับหนึ่ง การส่งภาพไปยังผู้รับวิธีการที่ได้เสนอไปนั้นเป็นวิธีการที่ไม่ค่อยดีเท่าไหร่นัก ในแง่การใช้งานเพราะผู้ใช้ต้องจัดหาโปรแกรมที่เป็น Web Server มาทำงานด้วย แต่ก็เป็นวิธีที่ดีที่สุดในขั้นตอนนี้ ซึ่งในส่วนนี้อาจจะมีการนำไปพัฒนาต่อไปอีก

บทที่ 7
สรุปและวิจารณ์การทดลอง

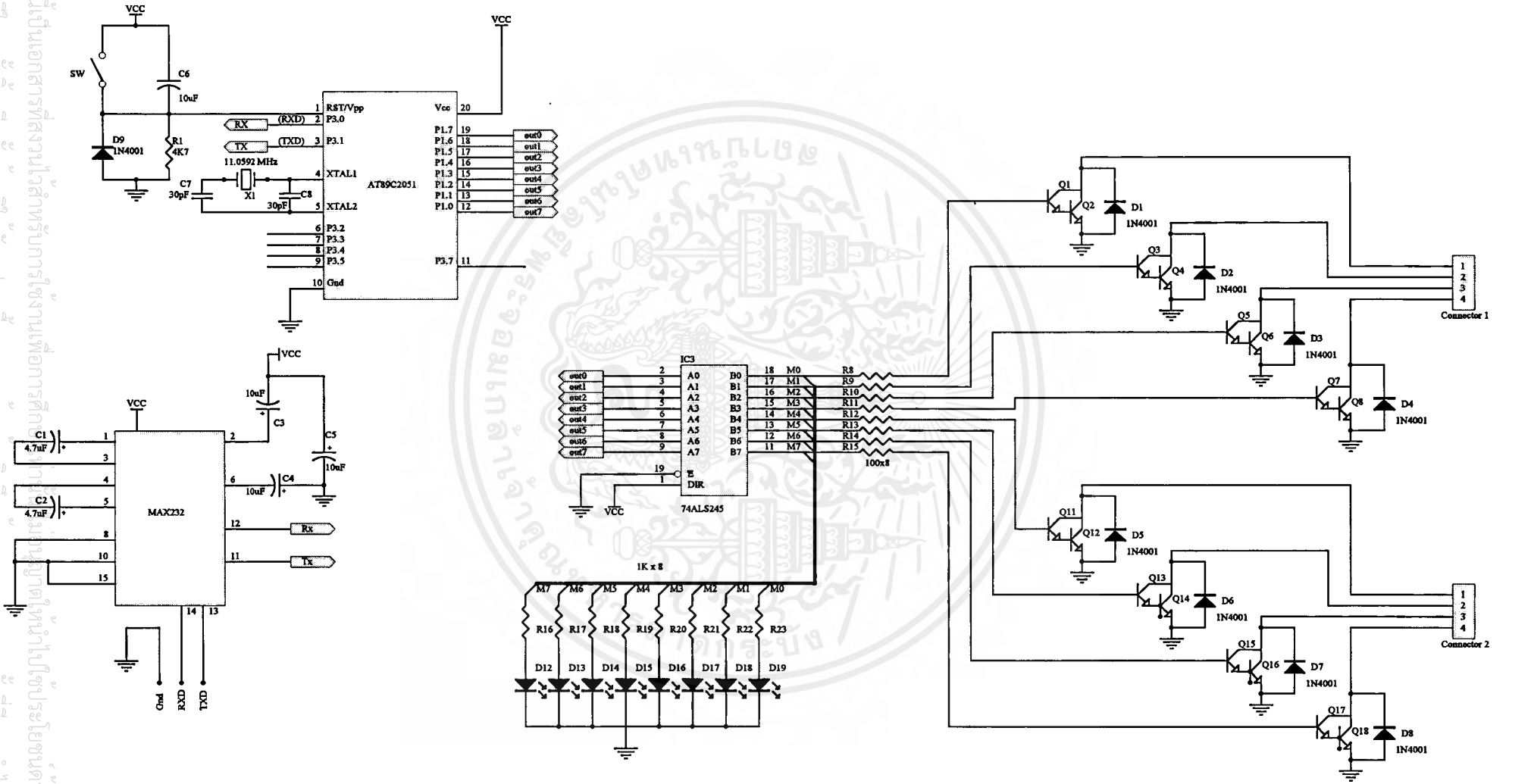
จากการที่ได้ออกแบบ สร้างและทดสอบตามกระบวนการทดลองแล้วทุกๆขั้นตอนที่ได้ปฏิบัติไปจะพบกับปัญหามากมายเกินความตั้งใจเอาไว้ อันเนื่องมาจากโครงงานนี้เป็นโครงงานที่ใหม่(ตามความคิดของผู้จัดทำ)และยังไม่มีใครทำมาก่อนในลักษณะโครงงานแบบนี้ ดังนั้นการค้นหาข้อมูลทุกอย่างที่ใช้ในโครงงานนี้จึงเปรียบได้กับการเริ่มต้นใหม่ทั้งสิ้น ถ้าจะมองในแง่ของผลงานของเทอมนี้อาจจะดูว่าน้อยมากเมื่อเทียบกับระยะเวลาและความตั้งใจที่ได้ตั้งใจเอาไว้ในแผนงานเบื้องต้น แต่จุดๆหนึ่งที่ทางผู้จัดทำรู้สึกภาคภูมิใจกับโครงงานชิ้นนี้ก็คือ การค้นหาข้อมูลต่างๆที่ใช้ในโครงงานนี้ ซึ่งได้มาหลายวิธีการมาก อาทิเช่น ที่สำคัญการค้นหาผ่านทางอินเตอร์เน็ตซึ่งทางผู้จัดทำคิดว่าข้อมูลส่วนมากนี้มาจากการค้นหาโดยวิธีนี้ ปรึกษาอาจารย์ที่ปรึกษาซึ่งได้ความกรุณาข้อมูลต่างๆที่จำเป็นมา ข้อมูลจากห้องสมุด และอื่นๆอีกหลายวิธี ข้อมูลเหล่านี้ก็ได้ถูกนำมารวบรวมเนื้อหาที่สำคัญมาเป็นฐานข้อมูลที่เชื่อได้สำหรับการสร้างออกแบบโครงงานนี้ และการสร้างและทดสอบก็ทำเป็นขั้นตอนเพื่อให้แน่ใจว่าแต่ละขั้นตอนถูกต้องและสมบูรณ์ที่สุดตามที่ได้ออกแบบไว้ ถึงกระนั่นก็ยังพบกับปัญหาทุกขั้นตอนอย่างหลีกเลี่ยงไม่ได้ แต่ก็สามารถแก้ปัญหาได้ลุล่วงมาได้ สุดท้ายก็ได้พบกับปัญหาที่หาทางแก้ไขไม่ได้ในเทอมนี้แต่จะต้องหาวิธีแก้กันต่อไปในเทอมต่อไป สำหรับการแก้ปัญหาต่างๆที่เกิดขึ้นมานั้น ก็ถือว่าเป็นจุดที่สำคัญและเป็นจุดเด่นอีกอย่างหนึ่งของการสร้างโครงงานนี้ ซึ่งมีทุกขั้นตอนของกระบวนการสร้างเริ่มจากส่วนล่างสุดคือชุดจับกล้องซึ่งต้องอาศัยทักษะทางด้านเครื่องมือช่างพื้นฐานมากพอสมควร เริ่มจากการซื้อวัสดุที่ใช้สร้างซึ่งหายากมากและราคาแพง ต่อมาก็การสร้างซึ่งใช้เวลานานพอสมควรสำหรับชุดจับกล้องแต่ก็ผ่านมาได้(ช่วงเวลานี้เป็นช่วงเวลาแห่งปัญหาซึ่งมาพร้อมการปรึกษาซึ่งกันและกัน)และต่อมาก็เป็นส่วนการ Interface Hardware และการเขียนโปรแกรมต่างๆที่ใช้ในโครงงานนี้ ก็จะมีที่ใช้จริงจะมีอยู่ 3 โปรแกรม แต่โปรแกรมที่เขียนขึ้นมาก่อนที่จะเป็นโปรแกรมที่สมบูรณ์ที่ได้เห็นในข้างต้นที่ได้กล่าวมาแล้ว มีโปรแกรมที่เขียนขึ้นมาเพื่อทดสอบการทำงานในช่วงการทำงานต่างๆซึ่งเราจำเป็นต้องมีโปรแกรมเหล่านี้เพื่อทดสอบการทำงานเบื้องต้นเอาไว้ เพราะเราไม่สามารถที่จะเขียนโปรแกรมเพียงครั้งเดียวแล้วสามารถที่จะทำงานได้อย่างสมบูรณ์และถูกต้อง จึงจำเป็นต้องมีการสร้างโปรแกรมขึ้นมาเพื่อประโยชน์หลายๆด้านในการทำงาน ส่วนในรายละเอียดปลีกย่อยสำหรับปัญหาในการเขียนโปรแกรมก็ได้อธิบายไปแล้วในส่วนของการสร้างและทดสอบ

<u>สรุป</u> การสร้างโครงงานนี้ ได้ทำให้เกิดทักษะมากมายอาทิเช่น การค้นหาข้อมูลจากแหล่งต่างๆ การทำงานร่วมกัน และการแก้ปัญหาต่างๆที่เกิดขึ้น การที่ได้ศึกษาความรู้เรื่องใหม่ๆซึ่งไม่มีในบทเรียน และเป็นการศึกษาหลายๆด้าน ที่สามารถนำไปประยุกต์ในงานด้านอื่นอีกต่อไป ทำให้มีความรู้เพิ่มขึ้นมากมายจากเดิมที่มีอยู่

บรรณานุกรม

1. Pat Bonner, " Network Programming With Windows Sockets ", Prentice-Hall INC., ISBN 0-13-230152-0, PP.49-78
2. Markus Pope, " Programming Internet Control ", Prima Publishing, ISBN 0-7615-0773-6 PP.167-184

ภาคผนวก



Title		
Size: B	Number	Revision
Date: 15-Mar-1999		Sheet of
File: C:\TRACK\SHEET_1.-CH		Drawn By:

```
// Program Internet Camera
// Design & Develop By
// Chutchai  Suebklang  &  Suchart Muenjeen
unit tcpcam;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Buttons, ScktComp, ExtCtrls, ComCtrls, ComDrv32,
  EnhAVICap, Clipbrd ,CDIBs;
type
  TChatForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    FileConnectItem: TMenuItem;
    FileListenItem: TMenuItem;
    StatusBar1: TStatusBar;
    Bevel1: TBevel;
    N1: TMenuItem;
    Disconnect1: TMenuItem;
    ServerSocket: TServerSocket;
    ClientSocket: TClientSocket;
    CommPortDriver1: TCommPortDriver;
    Connect: TButton;
    Panel3: TPanel;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    GroupBox1: TGroupBox;
    Panel1: TPanel;
    video: TEnhAVICap;
    Timer1: TTimer;
```

```
    private
    procedure ApplyCommSettings;
  protected
    IsServer: Boolean;
  end;
var
  ChatForm: TChatForm;
  Server: String;
  horizon: boolean;
  vertical: boolean;
  horover: integer;
  vertover: integer;
  stime : boolean;
implementation
 uses camset;
{$R *.DFM}
 procedure TchatForm.ApplyCommSettings;
var wasConnected: boolean;
begin
  wasConnected := CommPortDriver1.Connected;
end;
procedure TChatForm.FileListenItemClick(Sender: TObject);
begin
  FileListenItem.Checked := not FileListenItem.Checked;
  if FileListenItem.Checked then
  begin
    ClientSocket.Active := False;
    ServerSocket.Active := True;
    Statusbar1.Panels[0].Text := 'Listening...'
  end
  else
  begin
    if ServerSocket.Active then
      ServerSocket.Active := False;
```

```
Image1: TImage;
setting1: TMenuItem;
Format1: TMenuItem;
Source1: TMenuItem;
Setting: TMenuItem;
procedure FileListenItemClick(Sender: TObject);
procedure FileConnectItemClick(Sender: TObject);
procedure Exit1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure ServerSocketError(Sender: TObject; Number: Smallint;
  var Description: string; Scode: Integer; const Source,
  HelpFile: string; HelpContext: Integer; var CancelDisplay: Wordbool);
procedure Disconnect1Click(Sender: TObject);
procedure ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ServerSocketClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
  ErrorEvent: TErrorEvent; var ErrorCode: Integer);
procedure ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure ConnectClick(Sender: TObject);
procedure CommPortDriver1ReceiveData(Sender: TObject; DataPtr: Pointer;
  DataSize: Integer);
procedure Format1Click(Sender: TObject);
procedure Source1Click(Sender: TObject);
procedure SettingClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
```

```
    Statusbar1.Panels[0].Text := '';
  end;
end;
procedure TChatForm.FileConnectItemClick(Sender: TObject);
begin
  if ClientSocket.Active then ClientSocket.Active := False;
  if InputQuery('Computer to connect to', 'Address Name:', Server) then
    if Length(Server) > 0 then
      with ClientSocket do
      begin
        Host := Server;
        Active := True;
      end;
end;
procedure TChatForm.Exit1Click(Sender: TObject);
begin
  ServerSocket.Close;
  ClientSocket.Close;
  Close;
end;
procedure TChatForm.FormCreate(Sender: TObject);
begin
  FileListenItemClick(nil);
  video.OpenVideo(0);
  end;
procedure TChatForm.ServerSocketError(Sender: TObject; Number: Smallint;
  var Description: string; Scode: Integer; const Source, HelpFile: string;
  HelpContext: Integer; var CancelDisplay: Wordbool);
begin
  ShowMessage(Description);
end;
procedure TChatForm.Disconnect1Click(Sender: TObject);
begin
  ClientSocket.Close;
```

```
  FileListenItemClick(nil);
end;
procedure TChatForm.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteHost;
end;
procedure TChatForm.ServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
  var
  datain :string;
  begin
 datain:=Socket.ReceiveText;
 CommPortDriver1.SendString(datain);
  if (datain='8') or (datain = '2') then
  begin
   vertical:= true;
   horizon:=false;
   end;
  if (datain='4') or (datain='6') then
  begin
   horizon:=true;
   vertical:=false;
   end;
end;
procedure TChatForm.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  IsServer := True;
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteAddress;
end;
procedure TChatForm.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
```

```
  Statusbar1.Panels[0].Text := 'Now Client DisConnect';
  FileListenItemClick(nil);
end;
procedure TChatForm.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  Statusbar1.Panels[0].Text := 'Error connecting to : ' + Server;
  ErrorCode := 0;
end;
procedure TChatForm.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  ServerSocket.Active := False;
  FileListenItem.Checked := not FileListenItem.Checked;
  FileListenItemClick(nil);
end;
procedure TChatForm.ConnectClick(Sender: TObject);
begin
// Apply settings
  ApplyCommSettings;
  // Connect
  if CommPortDriver1.Connect then
  begin
    Connect.Enabled := false;
    //Disconnect.Enabled := true;
  end
  else // Error !
    begin
      MessageBeep( 0 );
    end;
  horover:=0;
    vertover:=0;
  end;
```

```
procedure TChatForm.CommPortDriver1ReceiveData(Sender: TObject;
  DataPtr: Pointer; DataSize: Integer);
var   p: pbyte;
      c: integer;
  begin
      p:=dataPtr;
      c:=p^;
      if horizon then
       begin
       if (c<>66)and (c<>65) then
       begin
       horover:=0;
       edit3.text:='';
       edit3.text:= floattostr((c-127)*1.8);
       ServerSocket.Socket.Connections[0].SendText(edit3.text);
       edit5.Color:=clgreen;
       end
       else
      begin
       case c of
        65 : begin
        horover:=2;
        ServerSocket.Socket.Connections[0].SendText('100');
        edit5.Color:=clred;
        end;
        66 :begin
         horover:=1;
         ServerSocket.Socket.Connections[0].SendText('101');
         edit5.Color:=clred;
         end;
         end;
      end
      end;
```

```
    if vertical then
     if (c<>66)and (c<>65) then
     begin
     vertover:=0;
     edit4.text:='';
     edit4.text:= floattostr((c-127)*1.8);
     ServerSocket.Socket.Connections[0].SendText(edit4.text);
     edit5.Color:=clgreen;
     end
     else
     begin
     case c of
      65 : begin
      vertover:=2;
      ServerSocket.Socket.Connections[0].SendText('100');
      edit5.Color:=clred;
      end;
      66 :begin
       vertover:=1;
       ServerSocket.Socket.Connections[0].SendText('101');
       edit5.Color:=clred;
       end;
       end;
    end
end;
procedure TChatForm.Format1Click(Sender: TObject);
begin
 video.DlgFormat;
end;
procedure TChatForm.Source1Click(Sender: TObject);
begin
 video.DlgSource;
end;
```

```
procedure TChatForm.SettingClick(Sender: TObject);
begin
form1.Show;
end;
procedure TChatForm.Timer1Timer(Sender: TObject);
var
  MyFormat : Word;
  dib : Tdib;
  AData,APalette : Thandle;
begin
     if stime then
     begin
      try
    ClipBoard.SetAsHandle(MyFormat,AData);
    dib := tdib.Create;
    video.ToClipboard;
    dib.Assign(clipboard);
    dib.SaveToClipboardFormat(MyFormat,AData,APalette);
    dib.SaveToFile(form1.Edit2.Text);
  finally
    dib.Free;
  end;
     stime:=false;
     end
     else
     begin
     stime:=true;
     end;
end;
end.
```

```
// Program Internet Camera Client Unit  Testing
// Design & Develop By
// Chutchai  Suebklang   &  Suchart Muenjeen
unit test_cli;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Buttons, ScktComp, ExtCtrls, ComCtrls, ComDrv32;
type
  TChatForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    FileConnectItem: TMenuItem;
    FileListenItem: TMenuItem;
    StatusBar1: TStatusBar;
    Bevel1: TBevel;
    Panel1: TPanel;
    N1: TMenuItem;
    SpeedButton1: TSpeedButton;
    Disconnect1: TMenuItem;
    ServerSocket: TServerSocket;
    ClientSocket: TClientSocket;
    Panel2: TPanel;
    stop: TButton;
    Left: TButton;
    Up: TButton;
    Down: TButton;
    Right: TButton;
    Panel3: TPanel;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label3: TLabel;
```

```
    Label4: TLabel;
    procedure FileListenItemClick(Sender: TObject);
    procedure FileConnectItemClick(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
   procedure FormCreate(Sender: TObject);
    procedure ServerSocketError(Sender: TObject; Number: Smallint;
      var Description: string; Scode: Integer; const Source,
      HelpFile: string; HelpContext: Integer; var CancelDisplay: Wordbool);
    procedure Disconnect1Click(Sender: TObject);
    procedure ClientSocketConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketRead(Sender: TObject; Socket: TCustomWinSocket);
  procedure ServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure ServerSocketClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure UpClick(Sender: TObject);
    procedure DownClick(Sender: TObject);
    procedure RightClick(Sender: TObject);
    procedure stopClick(Sender: TObject);
    procedure LeftClick(Sender: TObject);
    private
   protected
    IsServer: Boolean;
   end;
   var
   ChatForm: TChatForm;
    Server: String;
```

```
  horizon: boolean;
  vertical: boolean;
  horover: integer;
  vertover: integer;
implementation
{$R *.DFM}
procedure TChatForm.FileListenItemClick(Sender: TObject);
begin
  FileListenItem.Checked := not FileListenItem.Checked;
  if FileListenItem.Checked then
  begin
    ClientSocket.Active := False;
    ServerSocket.Active := True;
    Statusbar1.Panels[0].Text := 'Listening...'
  end
  else
  begin
    if ServerSocket.Active then
      ServerSocket.Active := False;
    Statusbar1.Panels[0].Text := '';
  end;
end;
procedure TChatForm.FileConnectItemClick(Sender: TObject);
begin
  if ClientSocket.Active then ClientSocket.Active := False;
  if InputQuery('Computer to connect to', 'Address Name:', Server) then
    if Length(Server) > 0 then
      with ClientSocket do
      begin
        Host := Server;
        Active := True;
      end;
end;
procedure TChatForm.Exit1Click(Sender: TObject);
```

```
begin
  ServerSocket.Close;
  ClientSocket.Close;
  Close;
end;
procedure TChatForm.FormCreate(Sender: TObject);
begin
  FileListenItemClick(nil);
end;
procedure TChatForm.ServerSocketError(Sender: TObject; Number: Smallint;
  var Description: string; Scode: Integer; const Source, HelpFile: string;
  HelpContext: Integer; var CancelDisplay: Wordbool);
begin
  ShowMessage(Description);
end;
procedure TChatForm.Disconnect1Click(Sender: TObject);
begin
  ClientSocket.Close;
  FileListenItemClick(nil);
end;
procedure TChatForm.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteHost;
end;
procedure TChatForm.ClientSocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
  var number :integer;
  begin
  if horizon then
  begin
      edit3.text:='';
      edit3.text:=Socket.ReceiveText;
      edit5.Color:=clgreen;
```

```
  end;
  if vertical then
  begin
  edit4.text:='';
  edit4.text:=Socket.ReceiveText;
  edit5.Color:=clgreen;
   end;
end;
procedure TChatForm.ServerSocketAccept(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 IsServer := True;
 Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteAddress;
end;
procedure TChatForm.ClientSocketDisconnect(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 Statusbar1.Panels[0].Text := 'Now Client DisConnect';
 FileListenItemClick(nil);
end;
procedure TChatForm.ClientSocketError(Sender: TObject;
 Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
 var ErrorCode: Integer);
begin
 Statusbar1.Panels[0].Text := 'Error connecting to : ' + Server;
 ErrorCode := 0;
end;
procedure TChatForm.ServerSocketClientDisconnect(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 ServerSocket.Active := False;
 FileListenItem.Checked := not FileListenItem.Checked;
 FileListenItemClick(nil);
end;
```

```
procedure TChatForm.UpClick(Sender: TObject);
begin
    ClientSocket.Socket.SendText('8');
    horizon:= false;
    vertical:= true;
  end;
procedure TChatForm.DownClick(Sender: TObject);
begin
    ClientSocket.Socket.SendText('2');
    horizon:= false;
    vertical:= true;
end;
procedure TChatForm.RightClick(Sender: TObject);
begin
    ClientSocket.Socket.SendText('6');
    horizon:= true;
    vertical:= false;
  end;
procedure TChatForm.stopClick(Sender: TObject);
begin
    ClientSocket.Socket.SendText('5');
  end;
procedure TChatForm.LeftClick(Sender: TObject);
begin
    ClientSocket.Socket.SendText('4');
    horizon:= true;
    vertical:= false;
end;
end.
```

```
        ; Program Control Interfacing
        ; Design & Develop By
        ; Chutchai  Suebklang   &   Suchart Muenjeen
        CPU "8051.TBL"
        HOF "INT8"
        TEMP:    EQU 100
        TEMP1:   EQU 101
        ORG 0000H
        LJMP MAIN
        ORG 23H
        LJMP INTER
        ORG 40H
MAIN:   MOV A,#11H
        MOV R0,#CON_M
        MOV @R0,A
        INC R0
        MOV A,#33H
        MOV @R0,A
        MOV A,#07FH
        MOV R0,#POS_M
        MOV @R0,A
        INC R0
        MOV @R0,A
        MOV ERROR,#41H
        MOV R3,#0H
        MOV TEMP,#0H
        MOV TEMP1,#0H
        MOV SCON,#50H
        MOV TMOD,#20H
        MOV TH1,#0FDH
        CLR PCON.7
        SETB TR1
        SETB EA
        SETB ES
```

```
        SETB REN
        MOV R4,#04H
RRR:    MOV P1,#0FFH
        LCALL DELAY
        LCALL DELAY
        MOV P1,#0H
        LCALL DELAY
        LCALL DELAY
        DJNZ R4,RRR
MAIN2:  CJNE R3,#36H,MAIN3
        LCALL R_L
MAIN3:  CJNE R3,#34H,MAIN4
        LCALL R_L
MAIN4:  CJNE R3,#38H,MAIN5
        LCALL U_D
MAIN5:  CJNE R3,#32H,MAIN6
        LCALL U_D
MAIN6:  CJNE R3,#37H,MAIN7
        NOP
MAIN7:  CJNE R3,#00H,MAIN2
        LJMP MAIN7
R_L:    CLR ES
        MOV R1,POS_M
        MOV A,CON_M
        CJNE R3,#36H,SUBT
        ADDR: RL A
        INC R1
        LJMP STORE
SUBT:   RR A
        DEC R1
STORE:  MOV ERROR,#0H
        MOV CON_M,A
        MOV POS_M,R1
        LCALL DELAY
```

```
        LCALL MOTOR
        SETB ES
        MOV A,POS_M
        MOV SBUF,A
        PUSH A
        LCALL CHK1
        POP A
        MOV R3,#0H
        RET
U_D:    CLR ES
        MOV R1,POS_M+1
        MOV A,CON_M+1
        CJNE R3,#38H,SUBT1
ADDR1:  RL A
        INC R1
        LJMP STORE1
SUBT1:  RR A
        DEC R1
STORE1:MOV ERROR,#0H
        CHK1: CJNE R1,#0B2H,CHK2
        MOV A,#41H
        MOV SBUF,A
        MOV R3,#0H
        LJMP ENT
CHK2:   CJNE R1,#4CH,ENT
        MOV A,#42H
        MOV SBUF,A
        MOV R3,#0H
ENT:    RET
MOTOR:  PUSH A
        PUSH PSW
        MOV R0,#CON_M
        MOV OUT_M,@R0
        ANL OUT_M,#0FH
```

```
        INC R0
        MOV A,#0F0H
        ANL A,@R0
        ORL A,OUT_M
        MOV P1,A
        POP PSW
        POP A
        RET
INTER:  CLR ES
        PUSH ACC
        PUSH PSW
        JB TI,TRANSMIT
RECEIV: MOV A,SBUF
        MOV R3,A
        CLR RI
        LJMP ENDLESS
TRANSMIT: CLR TI
ENDLESS: POP PSW
        POP ACC
        SETB ES
        RETI
DELAY:  MOV R7,#0FFH
DELAY1: MOV R6,#0FFH
        DJNZ R6,$
        DJNZ R7,DELAY1
        RET
    ORG 20H
    OUT_M: DFS 1
    CON_M: DFS 2
    POS_M: DFS 2
    ERROR: DFS 1
    END
```