ควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

REMOTE COMPUTER- CONTROLLED RC CAR

โดย

นายฐานันดร เชื้อศิริโรจน์

นายณัฐ ปิ่นเงิน

นายณัฐวิทย์ ประภากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

REMOTE COMPUTER-CONTROLLED RC CAR

โดย

นาย ฐานันดร เชื้อศิริโรจน์ 45010208

นายณัฐ ปิ่นเงิน 45010223

นาย ณัฐวิทย์ ประภากร 45010249

อาจารย์ที่ปรึกษา

ดร.พิพัฒน์ พรหมมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(.....)

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

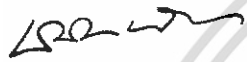
Remote computer-controlled RC car

ผู้จัดทำ

1.นาย ฐานันดร เชื้อศิริโรจน์ รหัสประจำตัว 45010208

2.นายณัฐ ปิ่นเงิน รหัสประจำตัว 45010223

3.นาย ณัฐวิทย์ ประภากร รหัสประจำตัว 45010249



.....อาจารย์ที่ปรึกษา

(ดร.พิพัฒน์ พรหมมี)

ควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

Remote computer-controlled RC car

โดย นาย ฐานันดร เชื้อศิริโรจน์ 45010208

นาย ณัฐ ปิ่นเงิน 45010223

นาย ณัฐวิทย์ ประภากร 45010249

อาจารย์ที่ปรึกษา ดร.พิพัฒน์ พรหมมี

บทคัดย่อ

โครงงานนี้เสนอการใช้คอมพิวเตอร์และไมโครคอนโทรลเลอร์ MCS-51ควบคุมรถบังคับวิทยุผ่านเครือข่ายอินเตอร์เน็ต โดยโครงงานนี้จะประกอบด้วยส่วนติดต่อกับผู้ใช้ผ่านโปรแกรมที่พัฒนาด้วย Visual C++ โดยผู้ใช้จะสามารถเห็นภาพจากกล้องที่ติดอยู่ที่รถบังคับ และสามารถควบคุมการเคลื่อนที่ของรถได้ และ สามารถโปรแกรมเส้นทางการเคลื่อนที่ของรถได้ ในส่วนของการควบคุมรถ โดยที่ตัวรถจะมีอุปกรณ์ตรวจจับสภาพควันไฟ และ จะส่งสัญญาณกลับมายังเครื่องแม่ข่าย โดยคอมพิวเตอร์ปลายทางจะรับข้อมูลจากต้นทางผ่านอินเตอร์เน็ตและส่งไปยังไมโครคอนโทรลเลอร์เพื่อแปลงเป็นสัญญาณควบคุมการเคลื่อนที่ทางความถี่วิทยุ

Abstract

This project presents a RC car via internet. It consists of web based user interface and far-end controlling unit. The user can see the far-end picture from a camera on RC car and able to control the moving of RC car by Visual C++. The programmable routing is also available to operate by user. The controlling unit is based on microcontroller MCS-51 that connected to the far-end computer. The smoke alarm is able to detect by a smoke sensor. The controlling data from far-end computer is transferred to microcontroller and converts to RF control signal in order to control the RC car.

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

ปัจจุบันเครือข่ายอินเตอร์เน็ตมีส่วนสำคัญมากในการสื่อสาร เพราะเป็นการสื่อสารที่เชื่อมต่อกันได้จากทุกส่วนของโลก ด้วยเหตุนี้เองโครงงานนี้จึงได้นำเอาประโยชน์ของอินเตอร์เน็ตมาประยุกต์เข้ากับการรักษาความปลอดภัย เพื่อไม่ว่าจะอยู่ที่ใดในโลกถ้าติดต่อกับเครือข่ายอินเตอร์เน็ตก็สามารถ ควบคุมรถบังคับวิทยุผ่านเครือข่ายอินเตอร์เน็ต โดยโครงงานนี้จะประกอบด้วยส่วนติดต่อกับผู้ใช้ผ่านตัวโปรแกรมที่ถูกพัฒนาขึ้นมา โดยผู้ใช้จะสามารถเห็นภาพจากกล้องที่ติดอยู่ที่รถบังคับ และสามารถควบคุมการเคลื่อนที่ของรถได้ตามต้องการ ในส่วนของการควบคุมรถ คอมพิวเตอร์ปลายทางจะรับข้อมูลจากต้นทางผ่านอินเตอร์เน็ตและส่งไปยังไมโครคอนโทรลเลอร์เพื่อแปลงเป็นสัญญาณควบคุมการเคลื่อนที่ทางความถี่วิทยุโดยที่ตัวรถจะมีอุปกรณ์ตรวจจับสภาพควันไฟ และ จะส่งสัญญาณกลับมายังเครื่องแม่ข่าย ซึ่งผู้ใช้จะสามารถรับรู้การเตือนควันไฟได้

1.2 วัตถุประสงค์ของการทำโครงงาน

1. เพื่อศึกษาการสื่อสารผ่าน IP (Internet Protocol)
2. เพื่อศึกษาการควบคุมรีโมทคอนโทรลของรถวิทยุบังคับโดยอาศัยไมโครคอนโทรลเลอร์
3. เพื่อศึกษาการรับส่งข้อมูลผ่านพอร์ตอนุกรมของคอมพิวเตอร์

1.3 โครงสร้างพื้นฐานของระบบ

1. ส่วนประมวลผล (Server Process) จะใช้คอมพิวเตอร์ส่วนบุคคล (PC) ที่บรรจุโปรแกรมเพื่อทำหน้าที่เป็น Server และสามารถตีความคำร้องขอของ Client ได้ และส่งสัญญาณไปควบคุมการทำงานของส่วนต่างๆ เช่น รถบังคับวิทยุที่ติดกล้องไว้ การส่งภาพให้ผู้ใช้งานรวมถึงการจับภาพของกล้อง เป็นต้น

2. ส่วนผู้ใช้งาน (Client Control) จะใช้คอมพิวเตอร์ส่วนบุคคล (PC) ทำหน้าที่ในการเชื่อมต่อเข้าสู่ Server จากนั้นรับภาพแล้วทำการแสดงผล รวมถึงทำหน้าที่ในการป้อนสัญญาณไปควบคุมส่วนต่างๆ ของรถวิทยุบังคับ

3. ส่วนควบคุมรถ (RC Car Control Unit) จะเป็นส่วนที่รับคำสั่งควบคุมการทำงานของรถจาก Server แล้วปฏิบัติ ตามคำสั่งนั้นๆ

4. ส่วนวงจรตรวจจับแก๊ส (Gas sensor) จะคอยส่งสัญญาณเตือนเมื่อสามารถตรวจจับแก๊สได้ ไปยังไมโครคอนโทรลเลอร์

5. กล้องถ่ายภาพ (Camera) ทำหน้าที่รับภาพแล้วทำการส่งสัญญาณภาพเข้าสู่เครื่องคอมพิวเตอร์

1.4 โครงสร้างของปริญญานิพนธ์

รายงานฉบับนี้จัดทำขึ้นเพื่อประกอบการนำเสนอการสร้างระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์โดย

บทที่ 1 จะกล่าวถึงความสำคัญ โครงสร้างพื้นฐานในการสร้างระบบควบคุมรถผ่านอินเตอร์เน็ตพร้อมด้วยขอบเขตของการศึกษาและออกแบบสร้างชิ้นงาน และโครงสร้างทั้งหมดของรายงาน

บทที่ 2 ได้อธิบายถึงรายละเอียดของทฤษฎี และหลักการที่สำคัญของระบบควบคุมรถวิทยุบังคับผ่านอินเตอร์เน็ตไว้โดยละเอียด

บทที่ 3 ได้นำเสนอขั้นตอนการออกแบบในแต่ละส่วนของระบบควบคุมรถวิทยุบังคับผ่านอินเตอร์เน็ตที่ใช้ในการทดลองจริง

บทที่ 4 ได้ทำการแสดงผลการทดลอง ระบบควบคุมรถวิทยุบังคับผ่านอินเตอร์เน็ตที่สร้างขึ้นว่ามีประสิทธิภาพตรงกับความต้องการเริ่มต้นหรือไม่

บทที่ 5 ได้แสดงการสรุปและวิจารณ์ถึงผลการทดลองที่เกิดขึ้นในบทที่ 4 และแจกแจงปัญหาหรือจุดบกพร่องต่างๆ ที่พบขณะทำการออกแบบและทดสอบในประเด็นต่างๆ

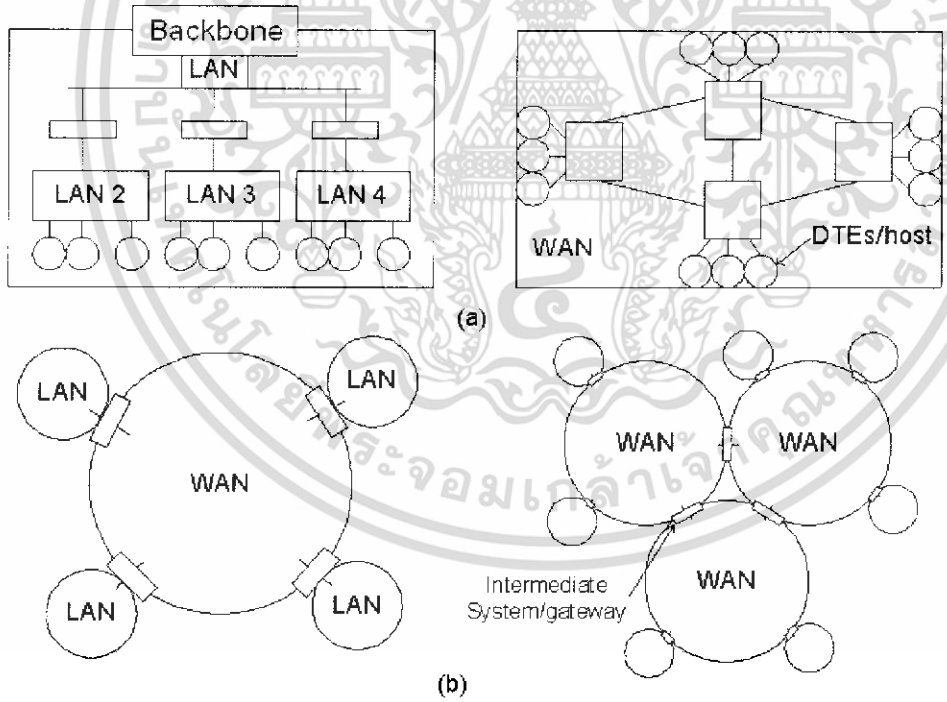
บทที่ 2
ทฤษฎีและหลักการ

ในส่วนของบทนี้จะกล่าวถึงทฤษฎีและหลักการที่ได้นำมาประยุกต์ใช้ในระบบควบคุมกล้องโทรทัศน์วงจรปิดผ่านอินเตอร์เน็ต ซึ่งเราสามารถแบ่งเป็นหัวข้อได้ดังนี้

2.1 เครือข่ายอินเทอร์เน็ต

อินเตอร์เน็ต (Internet) คือการที่เครือข่าย 2 เครือข่ายหรือมากกว่า เชื่อมต่อเข้าด้วยกัน และทำงานเสมือนเป็นเครือข่ายเดียวกัน โดย Network ที่เป็นส่วนประกอบของ Internet คือ Subnetwork (subnet) จะเป็นเครือข่าย Local Area Network (LAN) หรือ Wide Area Network (WAN) ก็ได้ โดยอุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่ายเข้าด้วยกันก็คือ Intermediate system (IS) หรือ Internetworking unit (IWU) การเชื่อมโยงระหว่างระบบที่แตกต่างกัน จำเป็นต้องมีมาตรฐานการติดต่อกัน ซึ่งเรียกเป็นศัพท์เฉพาะว่า โปรโตคอล (Protocol)

2.1.1 สถาปัตยกรรมอินเตอร์เน็ต (Internet Architectures)



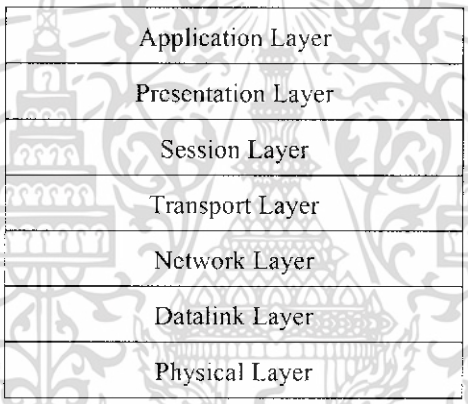
รูปที่ 2.1 สถาปัตยกรรมของ Internet (a) Single LAN and WAN

(b) Internetconnected LAN/WAN

ในรูป (a) แสดงตัวอย่าง 2 ตัวอย่างของเครือข่ายเดี่ยว (Single network) ซึ่งรูปทางด้านซ้ายเป็น Site-wide LAN ซึ่งประกอบขึ้นมาจากชุดของ LANs ซึ่งถูกต่อเข้ากับเครือข่ายหลัก (Backbone) ซึ่งอุปกรณ์ที่ใช้ต่อ LAN เข้ากับเครือข่ายหลัก ถ้า LAN ทุกเครือข่ายมีระบบเดียวกันก็จะใช้ Bridge ถ้าเป็น LAN ที่แตกต่างกันก็จะใช้ Router ตัวอย่างที่ 2 เป็นตัวอย่างของ WAN เดี่ยวๆ ในรูป (b) แสดงถึงเครือข่ายอินเตอร์เน็ต ซึ่งประกอบด้วย Network ทั้ง 2 ชนิดข้างต้น

2.1.2 OSI โมเดล

องค์ประกอบมาตรฐานสากล ISO (International Organization for Standardization) ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อยๆ และกำหนดโมเดลแบ่งเป็นชั้นๆ ตามลำดับ เรียกว่ามาตรฐาน OSI (Open System Interconnection) โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อยๆ ซึ่งจะช่วยในการออกแบบ และการใช้งานเครือข่ายรวมถึงการเชื่อมโยงกันเป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังรูปที่ 2.2



รูปที่ 2.2 การแบ่งการทำงานของเครือข่ายออกเป็น OSI model

2.1.3 โครงสร้างชั้น Network

ในแต่ละชั้นของ OSI model จะมีการติดต่อสื่อสารกันเป็นชั้น ตามลำดับลงมา เช่น Application Layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงชั้นแรกสุด คือ Physical Layer

- **Application Layer** เป็นชั้นบนสุดของโมเดล เป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้เป็นไปตามต้องการ ตัวอย่าง แอปพลิเคชั่นของเครือข่าย เช่น ระบบ E-mail, การโอนถ่ายข้อมูล (File transfer), การขอใช้ระบบคอมพิวเตอร์ในเครือข่าย เป็นต้น

- **Presentation Layer** มีการกำหนดหน้าที่ไม่ชัดเจนและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือเป็นส่วนที่จัดรูปและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมไปถึงการจัดแจงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC การลดขนาดข้อมูล (data compression) การเข้ารหัส หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่งโดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูลในการติดต่อครั้งนั้นๆ ให้เป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

- **Transport Layer** ทำหน้าที่ควบคุมปริมาณ และรายละเอียดการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้ายที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูลซึ่งส่วนของ TCP (Transmission Control Protocol) ในโปรโตคอล TCP/IP ทำงานที่ระดับนี้

- **Network Layer** ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน Packet ข้อมูล ผ่านอุปกรณ์ต่างๆ ไปยังเครือข่ายย่อยได้อย่างถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูล (Routing table) และกลั่นกรอง Packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกัน ไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โปรโตคอล IP, TCP/IP และ Ipx เป็นโปรโตคอลที่ทำงานอยู่ใน layer นี้

- **Datalink Layer** ทำหน้าที่เรียกใช้หรือกำหนดช่องทางในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Token ring หรือ FDDI เป็นต้น รวมถึงการกำหนดลำดับและอัตราการรับส่งข้อมูลหรือ Flow control และสถานที่ที่จะส่งข้อมูลไป (Address) ทั้งนี้ Datalink layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก Bit ให้เป็น Packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบผ่าน Checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ Packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนี้ไปใช้งานและบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

- **Physical Layer** รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน Hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณแบบต่างๆ การเชื่อมต่อเข้ากับเครือข่ายต่างๆ โดย Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือสัญญาณที่จำเป็นในการสื่อสารโดยตรง

เนื่องจาก Network Layer ในแต่ละ End System (ES) จะเป็นตัวจัดการการติดต่อแบบ End to End ของการให้บริการ Internetwide ไปยังผู้ใช้บริการ (NS-User) โดย ISO ได้จัด Network Layer เป็น 3 Protocol ซึ่งจะทำงานร่วมกัน เพื่อให้บริการใน network layer ได้แก่

SubNetwork Independent Convergence Protocol (SNICP)

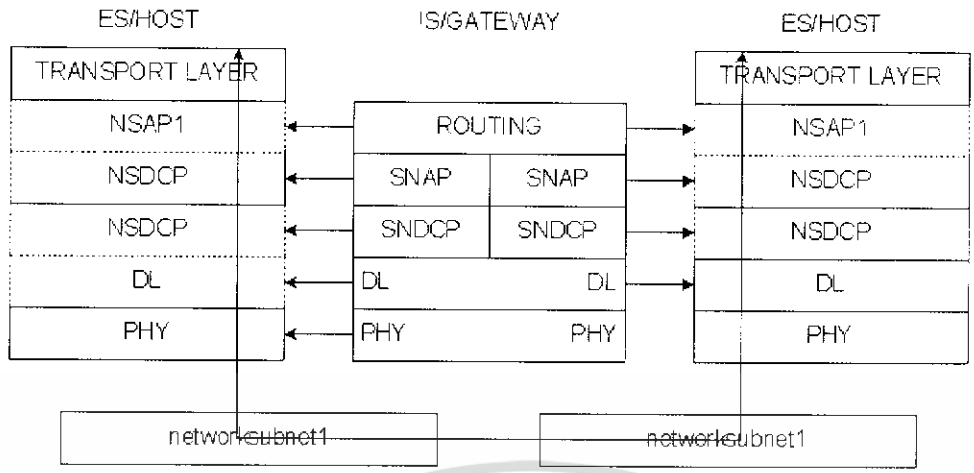
SubNetwork Dependent Convergence Protocol (SNDCP)

SubNetwork Dependent Access Protocol (SNDAP)

Es/Host

Transport Layer	
SNICP	= Subnet independent convergence protocol
SNDCP	= Subnet dependent convergence protocol
SNDAP	= Subnet dependent access protocol
DL	
PHY	

(a)



(b)

รูปที่ 2.3 Network Layer Structure (a) Sub Layer Protocol

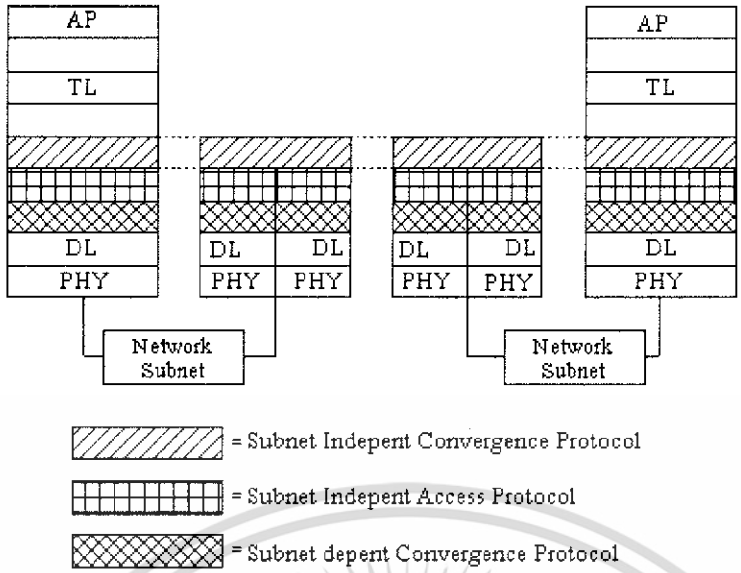
(b) IS Structure

โดยที่ SNICP จะเป็นตัวสนับสนุนจัดการให้ผู้ใช้บริการ (NS-user) สามารถ Interface กับ Internet ซึ่งมันจะมีหน้าที่เป็นตัวประสานฟังก์ชันต่างๆ ที่จำเป็นในการเลือกเส้นทางและถ่ายทอดข้อมูลของผู้ใช้งาน Internet ซึ่งการทำงานของมันไม่ขึ้นอยู่กับคุณสมบัติเฉพาะของเครือข่าย (Subnet)

SNDAP จะเป็นตัวโปรโตคอลที่ติดต่อกับเครือข่ายย่อย (Subnet) ที่มีลักษณะเฉพาะใน Internet เช่น X.25 Packet Layer Protocol

2.1.4 รูปแบบมาตรฐานโปรโตคอลของอินเตอร์เน็ต (Internet Protocol Standards)

อินเตอร์เน็ตโปรโตคอลซึ่งถูกใช้ในอินเตอร์เน็ต TCP/IP (Transmission Control Protocol/Internet Protocol) ซึ่งรวมถึง Transport และ Application โปรโตคอล ซึ่งโปรโตคอลทั้งหมดของ TCP/IP จะกำหนดให้เหมาะสมกับการใช้ในเชิงสาธารณะ ซึ่งรูปแบบโดยทั่วไปแสดงดังรูปที่ 2.4 IP เป็น Internetwide protocol ซึ่งทำให้สอง Transport protocol ที่ต่างสถานที่และต่าง Ess/hosts กันสามารถแลกเปลี่ยนหน่วยข้อมูลกันได้ ซึ่งหมายถึงว่า หลายๆ network / subnet และ ISO/Gateway ที่แตกต่างกันสามารถติดต่อสื่อสารได้อย่างสมบูรณ์

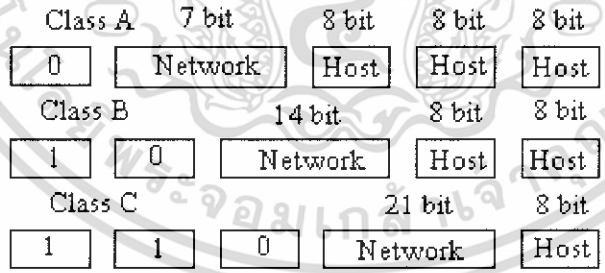


รูปที่ 2.4 Internetwide IP schematic

2.1.5 Internet IP สามารถอธิบายแยกย่อยได้ดังนี้

2.1.5.1 Address Structure

ในศัพท์ของ ISO เมื่อ 2 Network ติดต่อกันด้วย Es/host ที่ต่อกับอินเตอร์เน็ต Networkเหล่านี้ติดต่อกันได้โดยใช้ Network Service Access Point (NSAP) address และ Subnet Point of Attachment (SNPA) สำหรับใน TCP/IP ก็จะมี IP address และ NPhysical Address ตามลำดับ โดย NPhysical Address จะเป็นรูปแบบเดียวกัน กับโครงสร้างของ IP address แสดงดังรูปที่ 2.5



รูปที่ 2.5 โครงสร้างของ address ที่ใช้ใน class ต่าง ๆ ของเครือข่ายทั้งหมดความยาว 32 bits

IP address นี้มีการจัดแบ่งออกเป็นทั้งหมด 5 ระดับ (Class) แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ระดับ คือ คอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ใน Class A และลดหลั่นกันมาใน Class B และ Class C ตามลำดับ

จากรูปจะเห็นว่าหมายเลข IP ของ Class A มีตัวแรกเป็น 0 และหมายเลขเครือข่าย (Network number) ขนาด 7 Bit และมีหมายเลขเครื่องคอมพิวเตอร์ (Host Number) ขนาด 24 bit ทำให้ในหนึ่งเครือข่าย

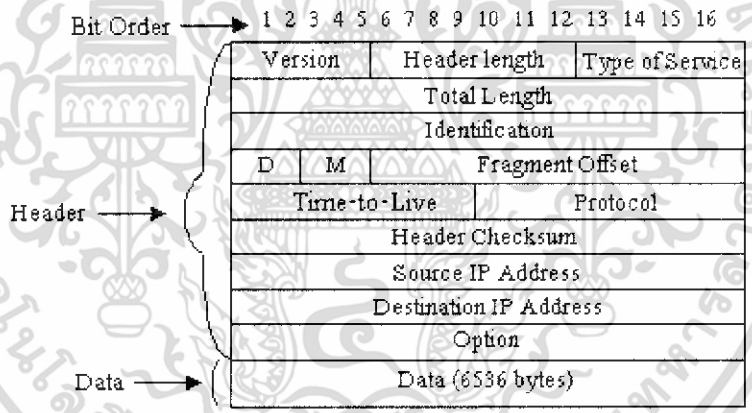
ของ Class A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ในเครือข่ายได้ถึง 2^{24} เครื่อง หรือประมาณ 16 ล้านเครื่อง

สำหรับ Class B จะมีหมายเลขเครือข่ายของเครื่องคอมพิวเตอร์แบบ 14 bits (ส่วนอีก 2 bit ที่เหลือบังคับว่าต้องขึ้นต้นด้วย 10_2) ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่าย Class B แต่ละเครือข่ายได้ถึง 2^{16} เครื่อง หรือประมาณ 65000 เครื่อง และสุดท้ายคือ Class C ซึ่งมีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit (ส่วน 3 bit แรกบังคับว่าต้องเป็น 110_2) ดังนั้นในแต่ละเครือข่ายจึงมีคอมพิวเตอร์เชื่อมต่อได้ 2^8 = 256 เครื่อง แต่หมายเลข 0 และ 255 จะไม่ถูกใช้งานจึงเหลือเพียง 254 เครื่อง

จะเห็นได้ว่าเมื่อเครือข่ายและเครื่องคอมพิวเตอร์ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกัน และมีความหมายให้ทราบถึงขนาดเครือข่ายแล้วการติดต่อผ่านข้อมูลจึงกระทำได้ไม่สับสน

2.1.5.2 รูปแบบของข้อมูล (Datagram)

รูปแบบของ IP data unit ก็คือ datagram ซึ่งโครงสร้างของ datagram เป็นดังรูปที่ 2.6



รูปที่ 2.6 Internet Datagram Format and Content

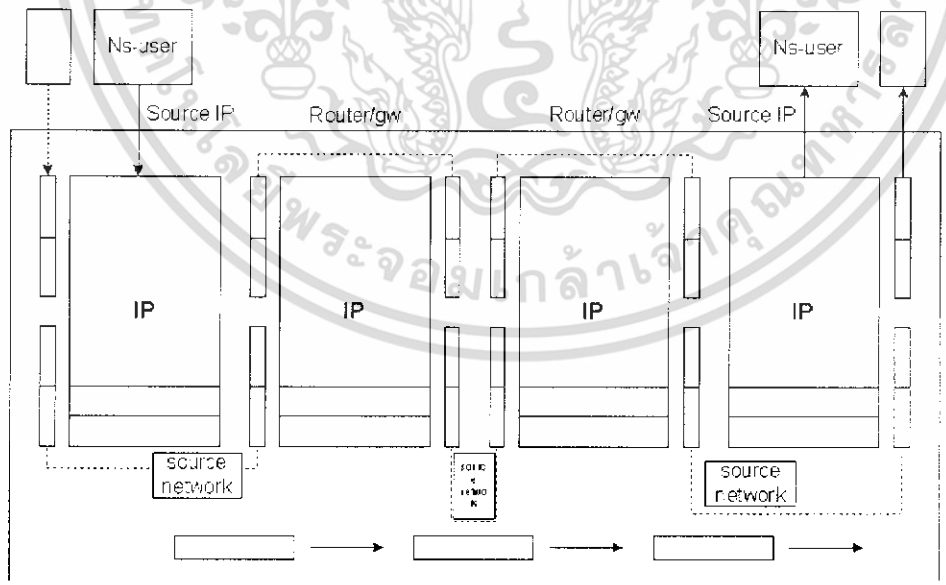
หน่วยข้อมูล IP (IP datagram) แต่ละหน่วยประกอบด้วยส่วนของข้อมูลที่รับมาจากส่วนงาน TCP หรือ UDP และส่วนของข้อมูลนำทาง (Header) ซึ่งมีรายละเอียดดังนี้

- Version หมายถึง รุ่นของข้อกำหนด IP
- Header Length ความยาวของข้อมูลนำทาง
- Type of Service วิธีการจัดการกับข้อมูล
- Total Length ความยาวของหน่วยข้อมูล

- Identification, Flags และ Fragment offset รายละเอียดที่เกี่ยวกับการแบ่งย่อยข้อมูลซึ่งจะถูกนำมาใช้ในการรวบรวมข้อมูล
- Time to live เวลาสูงสุดที่ใช้ในการเดินทาง ซึ่งกำหนดมาจากต้นทาง เวลานี้จะลดลงเรื่อยในระหว่างทางถ้าลดลงไปถึงศูนย์ หน่วยข้อมูลนั้นจะถูกกำจัดไป
- Protocol ชนิดของข้อมูลเป็น UDP หรือ TCP
- Header Checksum ค่าตรวจสอบข้อมูลนำทาง
- Source IP address และDestination IP หมายเลข Internetwide IP (NSP) ของเครื่องต้นทางและปลายทาง
- Option ข้อมูลอื่น ๆ เช่น ข้อมูลที่เกี่ยวกับการรักษาความปลอดภัย บันทึกเส้นทางเดินของข้อมูลและเวลาที่ข้อมูลเดินทางมาถึง เป็นต้น

2.1.5.3 การแบ่งส่วนข้อมูลและการประกอบขึ้นใหม่ (Fragmentation and Reassembly)

ขนาดข้อมูลของผู้ใช้ซึ่งอ้างอิง NSDU มีความจุ ได้ถึง 64 k หรือ 65,536 byte ต่อขนาดของหน่วยข้อมูล (packet size) ที่สามารถติดต่อกันในระบบที่ต่างกัน สามารถมีได้ตั้งแต่ 128 byte สำหรับระบบ x.25 packet switching จนถึง 8,000 bytes 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 ไป ซึ่งจะถูกออกคำสั่งโดย network ที่ติดต่ออยู่ด้วยและการส่ง Datagram ไปยัง IP ใน Gateway ตัวแรก โดยที่ IP ใน Gateway จะไม่ reassemble แต่จะปรับปรุงในขอบเขตที่เหมาะสม และส่ง Datagram ที่ได้รับตรงไปยัง Network ที่สอง (ถ้า Network ที่สองสามารถรองรับ ขนาด Datagram นี้ได้) หรือทำการ Fragment Datagram ให้มีขนาดเล็กลง ซึ่งขั้นตอนนี้จะถูกทำซ้ำที่ Gateway ตัวต่อไป Network ตัวสุดท้ายสามารถรองรับขนาดของ packet ได้มากกว่า packet ที่มันได้รับข้อมูลจึงถูกส่งได้โดยตรง โดยที่จะมีการปรับปรุงในส่วน Header ของ Datagram เท่านั้น จากนั้น IP ใน Host ปลายทางจะทำการประกอบข้อมูล (reassemble) ที่มันได้รับขึ้นมาใหม่และส่งผลที่ได้รับก็ คือ NSDU ไปยังผู้ใช้ (NS – User)

ในการคิดค่าเวลาสูงสุดที่ Host ต้นทางกำหนดให้ Gateway รอ datagram ระหว่างแต่ละการ Assembly ก็คือ Time – to – Live ซึ่งจะถูกตั้งค่าโดย IP ใน Host ต้นทาง ซึ่งจะมีค่าลดลงเรื่อย ๆ ในแต่ละขั้นตอนของการ Process datagram ถ้า Datagram ถูก Fragment ค่าปัจจุบันจะถูกนำไปใส่ในส่วน Header ของ Datagram ตัวใหม่ ถ้ามันถึงค่า 0 ที่จุดใด ๆ ระหว่างการ Process ใน Gateway การ Reassembly ก็จะล้มเหลวและทุก ๆ การ Fragment ที่เกี่ยวกับ NSDU ก็จะถูกตัดทิ้ง

ค่า Time – to – Live ในแต่ละ datagram จะเป็นจำนวนเท่าของวินาที โดยที่จำนวนของมันจะถูกลดลงโดยแต่ละ IP ซึ่งจะเปลี่ยนแปลงไปตามค่าเวลาจริงในการส่งถ่ายข้อมูลของ network ที่ติดต่อด้วย

2.1.5.4 การเลือกเส้นทาง (Routing)

ในแต่ละ Network (หรือ subnet) ใน Internet จะมีชนิดของ Physical Address ที่แตกต่างกัน ซึ่งระบบ (System Host) หรือ Gateway ที่ต่อเข้ากับ Network จะสามารถส่ง Datagram ไปยังระบบอื่นได้โดยตรงเฉพาะ Network ที่เหมือนกันเท่านั้น ในการเลือกเส้นทาง (Routing) ให้ Datagram ข้ามไปยังหลาย ๆ Network IP ในแต่ละ Internetwork gateway ต้องรู้ Physical Address ของ Host ปลายทาง

ซึ่งมี 2 วิธีการพื้นฐานที่ถูกใช้ในการหาเส้นทางภายใน Internet คือ Centralized และ Distributed ด้วยวิธีการ Centralized routing ข้อมูลเกี่ยวกับการเลือกเส้นทางที่เกี่ยวข้องกับแต่ละ Gateway จะถูก Download จาก Site ส่วนกลาง โดยใช้ข้อมูล Network และ Special Network Management จะพยายามตรวจสอบ Network และ Host ที่ถูกเพิ่มเข้า และถอดออก และข้อบกพร่องที่จะถูกวิจัยและตรวจสอบด้วยวิธีการ Distributed routing ทุก ๆ Gateway จะร่วมกันในการแบ่งปัน วิธีการในการรับประกันว่า ข้อมูลที่เกี่ยวกับการเลือกเส้นทางจะถูกจดจำไว้โดยแต่ละระบบในรูปของ Routing table ซึ่งจะมี Physical Address ไว้ให้ในการส่งแต่ละ Datagram ซึ่ง Internet จะใช้วิธีการแบบนี้

ขั้นตอนการ Routing ที่เกี่ยวกับ IP ขั้นตอนแรกจะอ่าน IP Address ปลายทางจากภายใน Datagram และใช้มันในการหาการตอบสนอง Physical Address ของ Host หรือ Gateway จาก Routing table ในส่วนที่เพิ่มเติมชุดของ Distributed protocol จะถูกใช้เพิ่มและรักษาส่วนที่อยู่ในแต่ละ Routing table ในรูปแบบของ Distributed ซึ่งรูปแบบทั่วไปถูกใช้ภายใน Host IP

2.2 หลักการใช้โปรโตคอลของ TCP/IP

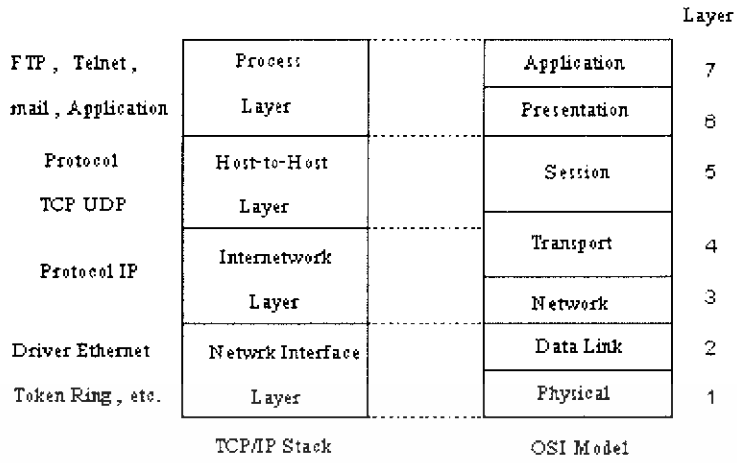
Application Programming ที่ส่วนมากใช้กันอย่างกว้างขวางสำหรับ TCP/IP Programmer คือ Windows Sockets Library ซึ่งได้ถูกใช้ในการรับส่งข้อมูลออกสู่ระบบ Internet สำหรับการใช้สิ่งแวดล้อม TCP/IP Application และเป็นธรรมดาอยู่แล้วเมื่อเวลาผ่านมา Microsoft ได้คิดค้น API สำหรับ TCP/IP ภายใต้ระบบปฏิบัติการของ Windows

ในการใช้งาน Windows Sockets จะไม่ถูกจำกัดในเรื่องของ TCP/IP Programming Windows Sockets ให้เห็นว่าแม้การติดต่อ ที่เป็นงานหนัก เมื่อใช้ TCP/IP หรือ Protocol อื่น ๆ สำหรับการส่งค่าจริง ๆ ของข้อมูล อย่างไรก็ตามในปัจจุบัน Winsock Implementation จะจำกัดการใช้งานของ Windows Sockets ในการติดต่อกับ TCP/IP

โปรโตคอล TCP/IP มีกลไกการทำงานเป็นชั้นหรือ Layer เรียงต่อกัน โดยในแต่ละ Layer จะมีการทำงานเทียบได้กับ OSI Model มาตรฐาน แต่บาง Layer ของโปรโตคอล TCP/IP จะทำงานเทียบกับ OSI หลาย Layer ปนกันซึ่งในแต่ละ Layer ของโปรโตคอล TCP/IP จะประกอบด้วย

- Process layer
- Host – to – Host layer
- Internetwork layer
- Network Interface layer

โดยเมื่อเทียบกับมาตรฐาน OSI model แล้วจะเป็นดังรูปที่ 2.8 ซึ่งเราจะเห็นว่าบางกลไกของโปรโตคอล TCP/IP เทียบได้กับมาตรฐาน OSI model สองชั้น หรือบางกลไกก็จะทำงานคาบเกี่ยวกันระหว่างบางชั้นของ OSI model ตัวอย่างเช่น กลไกก็จะทำงาน TCP/IP ในส่วน Network Interface layer เมื่อเทียบกับมาตรฐาน OSI model จะเทียบได้กับ Data Link layer และ Physical layer 2 ชั้นรวมกัน เป็นต้น ในแต่ละกลไกของโปรโตคอล TCP/IP จะมีโปรโตคอลอื่น ๆ ชุดของ TCP/IP ร่วมทำงานอยู่ด้วย ซึ่งจะกล่าวโดยละเอียดต่อไป



รูปที่ 2.8 TCP/IP stack เปรียบเทียบกับมาตรฐาน OSI

2.2.1 Process layer

จากรูปแสดงลำดับชั้นการทำงาน ของโปรโตคอล TCP/IP เทียบกับมาตรฐาน OSI model นั้น ในชั้นบนสุดเรียกว่า Process layer ทำงาน 2 หน้าที่ เทียบได้กับ Application layer และ Presentation layer ในชั้นนี้จะรองรับการทำงานของ Application ต่าง ๆ ที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ (Server) ให้บริการและเครื่องที่ขอใช้บริการ หรือ ไคลเอนต์ (Client) ซึ่งจะติดต่อกันผ่านโปรโตคอลเฉพาะแอพพลิเคชั่นอีกทีหนึ่ง ตัวอย่างเช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการโอนถ่ายไฟล์หรือ Download ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกใช้โปรแกรม FTP client ทั่วไป เช่น โปรแกรม WS FTP ติดต่อกับโปรเซส FTP ที่กำลังให้บริการอยู่ที่เครื่องเซิร์ฟเวอร์จากนั้นตัวโปรเซส FTP ก็จะเรียกใช้โปรโตคอล FTP (File Transfer Protocol) เพื่อทำการโอนถ่ายไฟล์นี้ หรือถ้าผู้ใช้ต้องการเรียกใช้งานคอมพิวเตอร์เครื่องที่อยู่ห่างไกลออกไปด้วยการใช้โปรแกรม Telnet ที่เครื่องเซิร์ฟเวอร์ ใช้บริการ ตัวโปรเซส telnet ที่อยู่ก็จะเรียกใช้โปรโตคอล Telnet เพื่อติดต่อกัน หรือในกรณีที่มีการเรียกใช้โปรแกรม web browser เช่น Netscape Navigator เพื่อเรียกดูเว็บเพจในเว็บไซต์ CNN ที่เครื่องซึ่งให้บริการเว็บของ CNN ก็จะมีโปรเซส HTTP (Hyper Text Transfer Protocol) ทำงานอยู่และจะติดต่อกับผู้ใช้ผ่านโปรโตคอล HTTP เป็นต้น

การทำงานของแอพพลิเคชั่นต่าง ๆ จะอยู่ที่ Process layer นี้ และมีการติดต่อกันตามแต่โปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งาน จากการที่ Process layer ของ TCP/IP รองรับให้โปรโตคอลอื่นทำงานได้หลายโปรเซส และหลายโปรโตคอลได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถ เปิดโปรแกรมใช้งานได้หลาย ๆ อย่างพร้อมกัน เช่น เปิดโปรแกรม Internet Explorer เพื่อเรียกดูเว็บเพจพร้อมกับใช้งานโปรแกรม Outlook Express เพื่อรับส่งอีเมล์ไปพร้อมกันได้โดยไม่ต้องรอให้ทำงานอย่างหนึ่งอย่างใดให้เสร็จก่อน หรือในปัจจุบันมีการพัฒนาโปรแกรม web browser ให้สามารถเรียกใช้งานโปรโตคอล FTP ได้โดยไม่ต้องไปหาโปรแกรมอื่นมาใช้

โปรโตคอลหลัก ๆ ที่ทำงานใน Process layer ซึ่งผู้ใช้มักจะคุ้นเคยกันดี ได้แก่ FTP (File Transfer Protocol), Telnet, HTTP (Hyper Text Transfer Protocol) และ SMTP (Simple Mail Transfer Protocol)

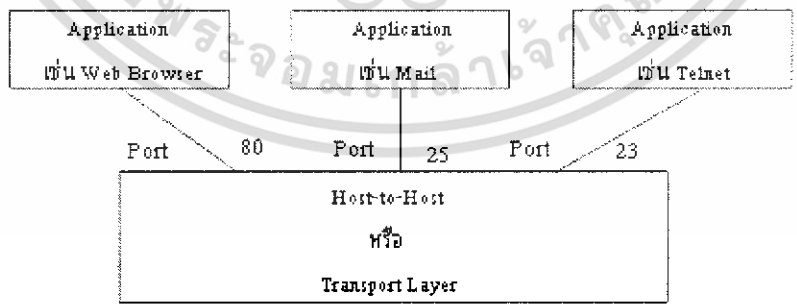
นอกจากนี้ยังมีโปรโตคอลอื่นอยู่เบื้องหลัง ซึ่งทำงานโดยที่ผู้ใช้ไม่สามารถมองเห็นได้จากโปรแกรมหรือไม่ได้มีการใช้งานโดยตรง เช่น

- โปรโตคอล DNS (Domain Name System) ที่ทำหน้าที่แปลงข้อมูลชื่อ domain name หรือชื่อเว็บไซต์ทั้งหลายให้เป็นหมายเลข IP address
- โปรโตคอล SNMP (Simple Network Management Protocol) ใช้ในการควบคุมและตรวจสอบอุปกรณ์ที่อยู่ในเครือข่าย
- โปรโตคอล DHCP (Dynamic Host Configuration Protocol) ทำหน้าที่แจกจ่ายข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายที่เชื่อมต่ออยู่

2.2.2 Host - to - Host layer

เครื่องเซิร์ฟเวอร์ที่ให้บริการต่าง ๆ เช่น เว็บเซิร์ฟเวอร์นั้น เมื่อมีผู้เข้ามาเรียกใช้บริการพร้อมกันหลายคน จะมีวิธีการส่งข้อมูลกลับไปยังต้นทางได้อย่างไรโดยไม่ผิดพลาด ซึ่งบางครั้งผู้ใช้รายหนึ่งอาจจะเปิดโปรแกรม Web Browser ซ้อนกันเพื่ออ่านข้อมูลจากเว็บเพจอื่น ๆ พร้อมกันไปได้ ดังนั้นระบบจะจัดการให้การจัดส่งข้อมูลเป็นไปอย่างถูกต้อง

การทำงานที่ชั้นของ Host - to - Host layer นี้จะมีบทบาทในการจัดการต่อจาก Process layer บางครั้งเรามีเรียกชั้น Host - to - Host ว่าเป็น Transport layer ซึ่งไม่ใช่ชั้นของ Transport layer ในมาตรฐาน OSI model การทำงานของ Host - to - Host layer นี้จะมีการสร้าง connection หรือการเชื่อมต่อกันระหว่างแอพพลิเคชั่น กับ Host - to - Host layer โดยจุดที่เชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า port (คำว่า port ในที่นี้ไม่ได้หมายถึง port ทางฮาร์ดแวร์) และในแต่ละแอพพลิเคชั่น ก็จะสร้างการเชื่อมต่อผ่าน port ได้พร้อมกันหลายแอพพลิเคชั่น ซึ่งการใช้งาน port ของแต่ละแอพพลิเคชั่นที่อยู่ในชั้น Process layer จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละโปรโตคอลจะมีการใช้งาน port หมายเลขต่าง ๆ ไม่ซ้ำกัน ตามรูปที่ 2.9

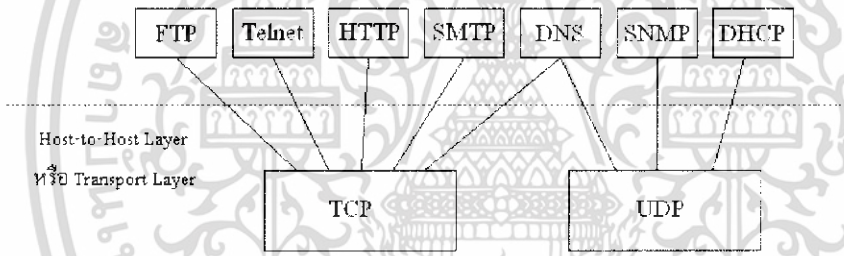


รูปที่ 2.9 แอพพลิเคชั่นหรือโปรเซสต่าง ๆ สื่อสารกับ Host - to - Host layer ผ่านจุดเชื่อมต่อ หรือ port

ส่วนหมายเลขในรูป คือ หมายเลข port ที่โปรเซสใช้งาน เช่น เว็บ หรือ โปรเซส http ใช้งาน port 80 ในการส่งผ่านข้อมูล เป็นต้น

เมื่อแอพพลิเคชั่นทำงานผ่านโปรโตคอลในชั้น Process layer จะมีการส่งผ่านข้อมูลไปยัง Host - to - Host layer ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน port ที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละโปรโตคอลทำได้ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซสที่แตกต่างกันก็ตาม หรือมีผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมาก และหลายแอพพลิเคชั่นในเวลาเดียวกัน ในชั้น Host - to - Host layer หรือ Transport layer ของ TCP/IP นี้จะมีโปรโตคอลทำงานอยู่ 2 โปรโตคอลที่แตกต่างกัน คือ โปรโตคอล TCP และโปรโตคอล UDP (User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปที่ชั้นถัด ๆ ไป เราจะเห็นว่าโปรโตคอล TCP และ UDP จะถูกผนึกเข้าไปในโปรโตคอล IP อีกทีหนึ่งและส่งต่อไปยังเครือข่ายอินเตอร์เน็ตต่อไป

ตัวโปรโตคอล TCP และโปรโตคอล UDP จะมีแอพพลิเคชั่น เฉพาะเพื่อเรียกใช้งานแยกกัน คือ แอพพลิเคชั่นที่ใช้โปรโตคอล FTP, Telnet, HTTP และ SMTP จะมีการส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอล TCP ส่วนแอพพลิเคชั่นที่ใช้โปรโตคอล SNMP และ DHCP จะมีการส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอล UDP และสำหรับโปรโตคอล DNS นั้นจะสามารถเรียกใช้งานได้ทั้ง TCP และ UDP ดังรูป ซึ่งเหตุผลที่มีการเรียกใช้โปรโตคอล TCP และ UDP แตกต่างกัน ก็เนื่องจากวิธีการทำงานของทั้งสองโปรโตคอลต่างกันนั่นเอง



รูปที่ 2.10 โปรเซสต่าง ๆ ที่เรียกใช้ Transport layer เพื่อส่งผ่านข้อมูลโดยอาศัย port

ซึ่งในแต่ละโปรเซสจะเรียกใช้งาน port เฉพาะแตกต่างกันยกเว้น DNS ที่สามารถใช้งานได้ทั้ง TCP และ UDP

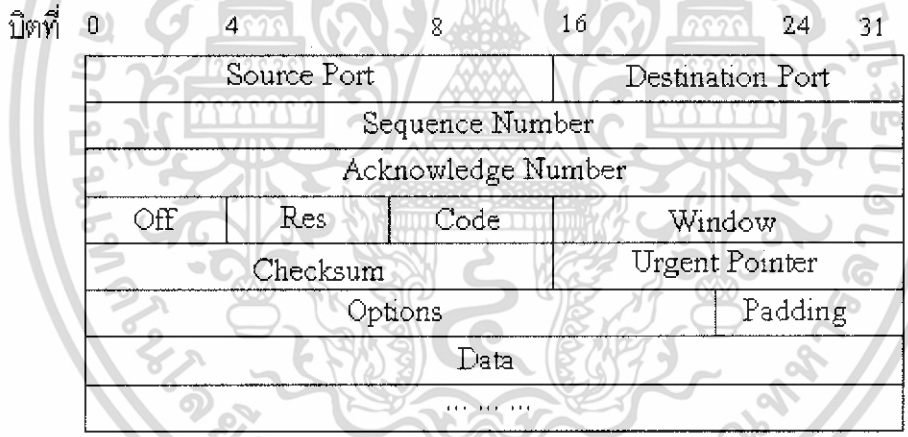
2.2.3 โปรโตคอล TCP

โปรโตคอล TCP (Transmission Control Protocol) เป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไปก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล Datagram ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชั่นหรือโปรเซสเซอร์ใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอล TCP จะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (bandwidth) มากกว่า UDP

การติดต่อระหว่างกันจะต้องเป็นแบบ Connection - oriented คือ ต้องมีการสร้างการติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้ว จึงเริ่มสนทนา เช่น พูดคำว่า สวัสดี หรือ ฮัลโหล กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า สวัสดี, ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือ ไม่พูดอะไรเป็นเวลานาน ๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไกโปรโตคอล TCP เมื่อแอพพลิเคชั่นต้องการส่งผ่านข้อมูลจะใช้โปรโตคอลที่เหมาะสมในชั้น Process layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่าน port ที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโปรโตคอล TCP

ในระหว่างการรับส่งข้อมูลนี้ โปรโตคอล TCP จะเพิ่มกระบวนการสอบทานข้อมูล (acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโปรโตคอล TCP จะมีมากกว่า แต่ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน



รูปที่ 2.11 รูปแบบของ TCP Packet

จะเห็นว่ามีฟิลด์ Acknowledgement Number และข้อมูล Checksum เพื่อใช้ตรวจสอบการเดินทางของข้อมูล ส่วน Header มีข้อมูลมากทำให้ต้องอาศัยทรัพยากรของระบบทำงานมาก

2.2.4 โปรโตคอล UDP

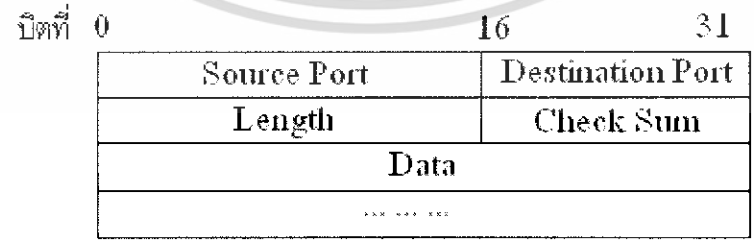
ใน Host - to - Host layer นอกจากจะมีโปรโตคอล TCP ทำงานแล้ว ก็ยังมีโปรโตคอล UDP (User Datagram Protocol) ที่มีคุณสมบัติแตกต่างกันอยู่ด้วยในการรับส่งข้อมูลผ่าน โปรโตคอล UDP จะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอล TCP และไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้น ๆ ด้วย เนื่องจากโปรโตคอล UDP ไม่มีสัญญาณ Acknowledgement ในการส่งข้อมูลแต่ละครั้งและไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชั่นหรือโปรเซสใดที่ต้องอาศัยโปรโตคอล UDP ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างกระบวนการตรวจสอบข้อมูลขึ้นมาเอง

โปรโตคอลชั้นบนขึ้นไปที่ใช้การส่งผ่านข้อมูลโดยโปรโตคอล UDP เช่น โปรโตคอล SNMP (ใช้ควบคุมและจัดการอุปกรณ์ในเครือข่าย) หรือโปรโตคอล DHCP (ใช้ส่งข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายได้ใช้งาน) การส่งข้อมูลเหล่านั้นที่มีการรับส่งจะไปทำในขั้นตอนของโปรโตคอลชั้นที่สูงกว่าแทน

ตัวอย่างขั้นตอนกลไกการทำงานโดยใช้โปรโตคอล UDP มีดังต่อไปนี้

1. ในชั้นของ Process layer เมื่อโปรแกรมควบคุมอุปกรณ์เครือข่าย เช่น โปรแกรม Network management ต้องการส่งข้อมูลไปยังอุปกรณ์ที่ต้องการ แอพพลิเคชั่นนั้นจะติดต่อผ่านโปรโตคอล SNMP ในชั้น Process layer
2. โปรโตคอล SNMP จะติดต่อกับโปรโตคอล UDP ในชั้นถัดไป เพื่อขอติดต่อผ่าน port ที่กำหนด
3. โปรโตคอล SNMP เตรียมข้อมูลที่จะส่ง รวมทั้งที่อยู่ปลายทาง
4. โปรโตคอล SNMP ส่งผ่านข้อมูลให้โปรโตคอล UDP ที่อยู่ในชั้น Host - to - Host layer
5. โปรโตคอล UDP ทำหน้าที่ผนึกข้อมูลหรือ datagram นั้น ไปกับโปรโตคอล IP ในชั้นถัดลงไป เพื่อส่งข้อมูลออกจากเครื่อง

ซึ่งจะเห็นว่ากลไกที่ต่างจากการส่งข้อมูลด้วยโปรโตคอล TCP ซึ่งจะต้องติดต่อกันก่อน และทั้งสองฝ่ายรับทราบการรับส่งข้อมูลของช่องการส่งข้อมูลนั้น



รูปที่ 2.12 รูปแบบของ UDP packet

จะมีฟิลด์ข้อมูลส่วน Header น้อยมากและไม่มีข้อมูลส่วนการตรวจสอบข้อมูล ทำให้ UDP packet มีขนาดเล็กและใช้หน่วยความจำหรือทรัพยากรของระบบน้อย

2.2.5 Internetwork Layer

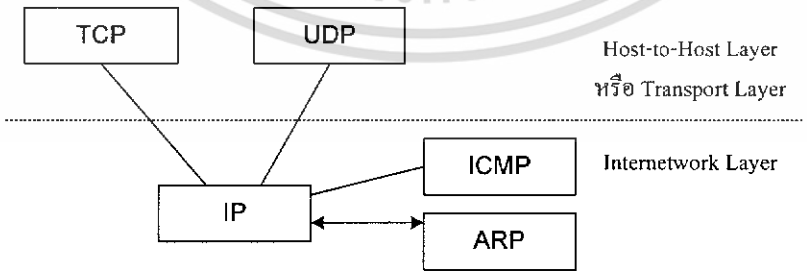
ในระดับล่างต่อมาในชั้น Internetwork layer มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใด ๆ บนอินเตอร์เน็ต คือ โปรโตคอล IP (Internet Protocol) นอกจากนี้ในชั้น Internetwork layer ยังมีโปรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ โปรโตคอล Internet Control Message Protocol (ICMP) และโปรโตคอล Address Resolution Protocol (ARP)

2.2.6 โปรโตคอล IP

โปรโตคอล IP ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจาก Host - to - Host layer เพื่อส่งข้ามไปยังเครือข่ายใด ๆ ได้อย่างถูกต้อง แม้ว่าจะมีเครือข่ายเชื่อมต่อกันอยู่ในอินเตอร์เน็ตเป็นล้าน ๆ เครือข่ายก็ตาม เนื่องจากโปรโตคอล IP มีข้อมูลตำแหน่ง IP ปลายทางที่จะส่งข้อมูลไป โดยทำงานร่วมกับอุปกรณ์ Router เพื่อส่งข้อมูลผ่านสวิตช์ (Switch) ไปยังปลายทาง ตัววงจรผ่าน หรือ switch นี้อาจเป็น Gateway หรือ Router ในระบบเครือข่ายก็ได้ ซึ่งในข้อมูลของโปรโตคอล IP จะมีข้อมูลของหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์ และจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วยโปรโตคอล ARP ตามรูป 2.13 ที่จะแสดงการติดต่อกันระหว่างโปรโตคอลในชั้นของ Host - to - Host layer และ Internetwork layer

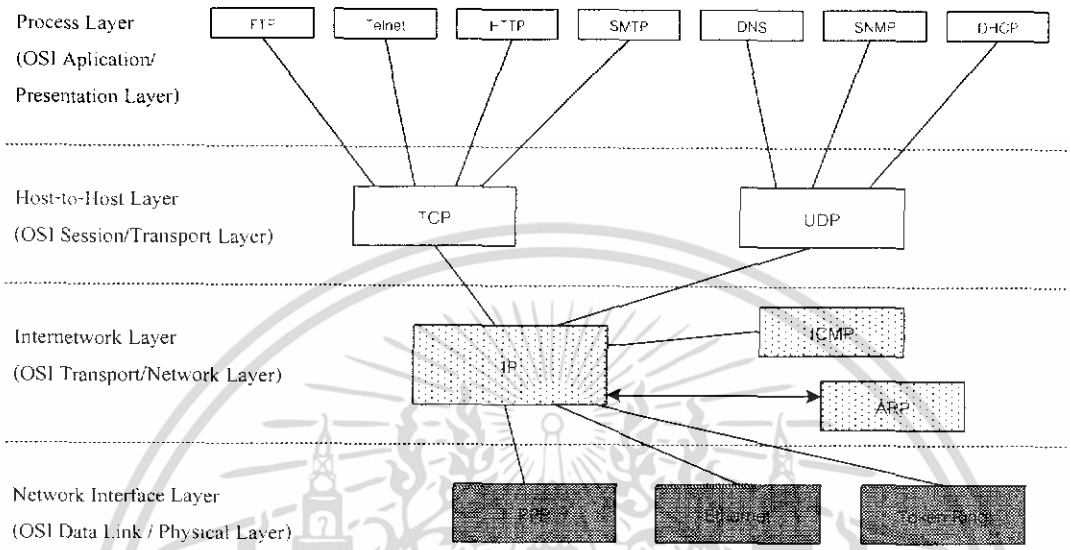
2.2.7 โปรโตคอล ICMP

หน้าที่หลักของโปรโตคอล ICMP (Internet Control Message Protocol) คือ การแจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากที่พบคือ ส่งไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ เป็นต้น นอกจากนี้โปรโตคอล ICMP ยังถูกเรียกใช้งานจากเครื่องเซิร์ฟเวอร์และ Router อีกด้วยเพื่อและเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานของโปรโตคอล ICMP นั้นจะทำงานควบคู่กับโปรโตคอล IP ในระดับเดียวกัน และข้อความต่าง ๆ ที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของ IP (IP datagram) อีกทีหนึ่ง



รูปที่ 2.13 โปรโตคอล TCP และ UDP

จากรูปจะเห็นว่าโปรโตคอล TCP และ UDP อาศัยโปรโตคอล IP ที่อยู่ชั้นล่างเพื่อส่งผ่านข้อมูลระหว่างเครือข่าย และในชั้น Internetwork Protocol ยังมีโปรโตคอล ICMP ทำหน้าที่ส่งข้อความแจ้งเตือน และโปรโตคอล ARP ทำหน้าที่แปลงเลขหมาย IP ไปเป็นหมายเลขของฮาร์ดแวร์จริง



รูปที่ 2.14 โครงสร้างของโปรโตคอล TCP/IP

ในแต่ละชั้นหรือ layer จะมีโปรโตคอลหลักทำหน้าที่ต่าง ๆ และส่งผ่านข้อมูลไปยังเครือข่าย และ ออกสู่อินเตอร์เน็ต

กล่าวโดยสรุปก็คือ โปรโตคอล TCP/IP ทำงานโดยแบ่งเป็นชั้นเทียบกับ OSI model ได้กลไกในการทำงานของโปรโตคอล TCP/IP มี 4 ชั้น

ซึ่งในชั้นแรก คือ Process layer ทำหน้าที่ติดต่อกับแอพพลิเคชั่นและโปรโตคอลที่แอพพลิเคชั่นนั้น ๆ ใช้งานและ ส่งต่อมาให้ชั้น Host - to - Host layer เพื่อติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องผู้ขอใช้ บริการ ในชั้นนี้จะมีการสร้าง Session หรือการเชื่อมต่อระหว่างระบบขึ้นตามแต่ละโปรโตคอลที่ต้องการ

ต่อมาเป็นการผนึกข้อมูลไปเป็น IP datagram ที่ชั้น Internetwork layer โดยอาศัยโปรโตคอล IP เพื่อให้สามารถติดต่อส่งข้อมูลข้ามเครือข่ายไปยังเครือข่ายและเครื่องที่ถูกต้องได้ และสุดท้ายการส่งข้อมูลออกสู่โลกภายนอก ต้องอาศัยกลไกในชั้น Network Interface layer เพื่อแปลงข้อมูลใหม่ เพิ่มข้อมูลใหม่ที่จำเป็นในการอ้างอิงตำแหน่งและแปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งออกไปเครือข่าย และอาจจะออกไปยัง Gateway หรือ Router เพื่อข้ามเครือข่ายออกไปยังเส้นที่กำหนดไว้เพื่อผ่านข้อมูลไปยังระดับล่างและออกสู่เครือข่ายอินเตอร์เน็ตในที่สุด

2.2.8 กลไกของโปรโตคอล IP

ในการส่งผ่านข้อมูลหรือ IP datagram ไปยังเครือข่ายอินเตอร์เน็ตนั้นโปรโตคอล IP จะทำหน้าที่พิจารณาว่าปลายทางในการส่ง IP datagram นั้นจะเป็นภายในเครือข่ายของตนเองหรือจะต้องส่งข้อมูลข้ามเครือข่ายไปอีก โดยการพิจารณานี้โปรโตคอล IP address ปลายทางว่าส่วนที่เป็นค่าหมายเลขเครือข่าย (Network Address) จะเหมือนกับค่าหมายเลขเครือข่ายของ IP address ต้นทางหรือไม่ ถ้าตรงกันแสดงว่าการส่งข้อมูลอยู่ภายในเครือข่ายเดียวกัน แต่ถ้าค่าต่างกันแสดงว่าต้องส่งข้อมูลไปยังปลายทางที่อยู่คนละเครือข่ายกัน

- การส่งข้อมูลภายในเครือขายเดียวกัน มีกลไกดังนี้

1. โปรโตคอล IP จะเรียกใช้บริการโปรโตคอล ARP (Address Resolution Protocol) เพื่อแปลงหมายเลข IP ปลายทาง ให้เป็นค่าหมายเลขฮาร์ดแวร์ เช่น MAC Address
2. เมื่อโปรโตคอล IP รับค่าหมายเลขฮาร์ดแวร์แล้ว ก็จะส่งข้อมูลนั้นไปยังฮาร์ดแวร์ที่ระบุไว้

- การส่งข้อมูลข้ามเครือข่าย มีกลไกดังนี้

1. โปรโตคอล IP จะตรวจสอบพบว่า หมายเลข IP Address ปลายทางอยู่คนละเครือข่ายกัน โดยโปรโตคอล IP จะอ่านค่า IP Address ของ Router เพื่อเตรียมส่งข้อมูลไปที่ Router แทนซึ่งในที่นี้จะมีการกำหนดเป็น default router
2. โปรโตคอล IP จะเรียกใช้บริการโปรโตคอล ARP เพื่อแปลงค่า IP Address ของ Router ให้เป็นค่าหมายเลขฮาร์ดแวร์
3. โปรโตคอล IP ส่งข้อมูล IP datagram ไปยัง Router ที่กำหนดไว้ จากนั้น Router ส่งข้อมูลข้ามเครือข่ายไปตามขั้นตอน

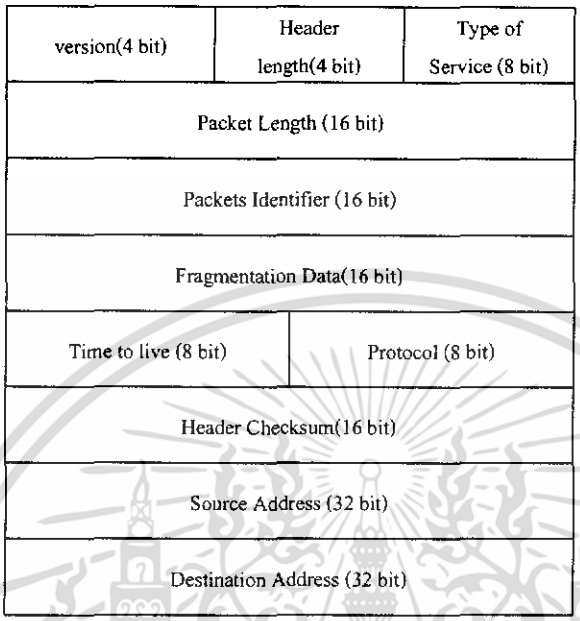
2.2.8.1 IP Datagram

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

IP datagram จะประกอบด้วยส่วนของ Header Block ของข้อมูลซึ่งข้อมูลสามารถที่จะกำหนดขึ้นอยู่กับชนิดของ Service และความต้องการของผู้ใช้ ในส่วน Header จะประกอบด้วยชุดของ Well - Defined Field

2.2.8.2 Header

ในส่วนหัวของ IP datagram หรือ IP Header โดยส่วนมากจะประกอบด้วย 20 bytes ตามรูปที่ 2.15 จะแสดง IP Header



รูปที่ 2.15 IP Header

ตามรูปที่ 2.15 ครั้งแรกจะพิจารณาสิ่งที่น่าสนใจในส่วนของ Protocol,Source,Address และ Destination Address Field ในส่วนของ Protocol จะถูกกำหนดวิธีของ IP Packet จะถูกแปลทั่ว ๆ ไป ค่าต่าง ๆ จะถูกกำหนดสำหรับส่วนนี้ ในส่วนของ Address จะแสดงให้เห็นส่วนของ Host Address ซึ่ง Address นี้จะมีเพียงหมายเลขเดียวเท่านั้นที่เหมือนกันในระบบ Internet

2.2.8.3 IP Host Address and Routing

IP Host Address จะเป็นส่วนข้อมูล 32 bits ซึ่งมีเพียงตัวเลขเดียวเท่านั้นที่เหมือนกันซึ่งใช้ในการติดต่อกับ Internet Host ส่วนของ Gateway (เป็นตัวทำหน้าที่ Interface Network ที่มีมากกว่า 1 Network) จะมี Host อยู่มากมาย

ตามหลักที่ถูกต้องคือ 4 bytes ของ Internet host address มักจะเขียนอยู่ในรูปของเลขฐานสิบ ตัวอย่างเช่น 121.0.0.1

Internet Host address ส่วนมากจะถูกแบ่งออกเป็น 2 ส่วน คือ Network address และActual host address ถ้าเรียงตามความยาวทั้งสองตาม address โดยมากจะขึ้นอยู่กับการกำหนด bytes ใน address เสียเป็นส่วนใหญ่

- **Class A Network address** จะสังเกตได้ว่า byte แรกจะอยู่ระหว่าง 0 ถึง 127 และจะประกอบไปด้วย 8 bit Network address และ 24 bit Host address เนื่องจาก address เริ่มต้นที่ 0 และ 127 จะถูกสงวนไว้ ซึ่งสามารถมีค่าสูงสุด คือ 126 Class A Subnets ใน Class A นี้สามารถมีได้ทั้งหมด 16,777,214 Hosts (Address อยู่ในรูป nnn.0.0.0 และ nnn.255.255.255 จะถูกสงวนเอาไว้)

- **Class B Network address** จะสังเกตได้จากตัวเลขหลักแรกสุดอยู่ระหว่าง 128 และ191 Class B Network จะประกอบไปด้วย 16 bit Network address และ 16 bit 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** จะสังเกตได้จากตัวเลขหน้าสุดอยู่ระหว่าง 192 และ 223 ซึ่งจะมีได้ 2,097,152 Subnets Class C Subnets สามารถที่จะบรรจุได้ถึง 254 hosts (โดยเริ่มที่Address nnn.mmm.bbb.0 และ nnn.mmm.bbb.255 จะถูกสงวนไว้)

- **Class D Network address** (โดยทั่วไปตัวเลข byte แรกจะอยู่ระหว่าง 224 ถึง 255) จะถูกสงวนเขาไว้เพื่อ IP Multitasking ที่ถูกจำกัดจากในส่วนของ IP Broadcasting คือจะไม่ไปยุ่งเกี่ยวกับ Winsock Programmer

2.2.8.4 Host Names

ในระหว่างการคิดค้นประดิษฐ์ Internet ซึ่งตัวเลขที่จะนำมาแทน Host address ซึ่งจะมีไม่เพียงพอและอีกอย่างหนึ่งคือ ตัวเลขเหล่านี้เป็นการยากที่จะจดจำ และอีกอย่างถ้าเกิด Host address มีการเปลี่ยนแปลงอันเนื่องมาจาก หลาย ๆ สาเหตุ จึงทำให้เกิดความยุ่งยาก ดังนั้นระบบ Naming System จึงได้ถูกสร้างขึ้นมาซึ่งใช้ในการ Map ตัวเลขของ IP address ไปเป็น Memories host names ในเครือข่ายอินเตอร์เน็ต host จะถูกปรับเปลี่ยนให้อยู่ในรูปของ Files กับข้อมูลของ Internet host ทุกตัวและ address อย่างไรก็ตามอินเตอร์เน็ตก็ได้เจริญเติบโตอย่างต่อเนื่อง ทำให้ดูว่าจะไม่เพียงพอ อย่างแรกคือการตั้งชื่อ internet host จะทำได้มีมาตรฐาน อย่างที่ 2 คือ ทางด้าน Hardware สามารถที่จะแยกแยะความต้องการที่จะติดต่อกับทุกเครื่องคอมพิวเตอร์ทุกตัวที่ต่ออินเตอร์เน็ตอยู่

คำตอบของปัญหานี้คือ การกำหนด Domain Name System (DNS) ซึ่งจะเป็นตัวจัดระเบียบของการตั้งชื่อ (Naming System) รูปแบบของ Domain Names คือ จะคล้ายกันทุกคนที่ใช้อินเตอร์เน็ต โดยปกติการตั้งชื่อจะอยู่ในรูปแบบ คือ Host Subdomain Domain

ในส่วนนี้จะกำหนดโดยหน่วยงานหนึ่ง (ส่วนมากเป็นหน่วยงานในสหรัฐ) หรือ โดยแต่ละประเทศ โดยจะมีดังต่อไปนี้

- GOV : Government Bodies
- EDU : Education Institutions
- COM : Commercial enterprises

- MIL : Military organization
- ORG : Other organization

Top – level Domain โดยใช้ชื่อประเทศ มักจะใช้ตามมาตรฐาน ISO 3166 D โดยจะใช้อักษรย่อ 2 ตัวของชื่อประเทศ

การทำงานเกี่ยวกับเครือข่ายคอมพิวเตอร์นั้นเป็นเรื่องค่อนข้างซับซ้อน จนทำให้เราสร้างข้อกำหนดต่าง ๆ ขึ้นมาเพื่อให้วิธีการสื่อสารระหว่างกันเป็นไปได้โดยราบรื่น แล้วก็ทำให้เกิดเป็นโปรโตคอล (Protocol) มากมายหลายชนิดมาใช้งานร่วมกับเครือข่าย ซึ่งเรามีการแบ่งโปรโตคอลต่าง ๆ ออกเป็นระดับชั้น เพื่อให้งานต่อการใช้งานและง่ายต่อการทำความเข้าใจ นานวันเข้าโปรโตคอลที่เราคิดว่ามันทำให้เราทำงานสะดวก นั้นก็เพิ่มมากขึ้นจนกลายเป็นความยุ่งยาก ต่างบริษัทก็ต่างมีผลิตภัณฑ์ที่สนับสนุนโปรโตคอลต่าง ๆ ไม่เหมือนกัน ทำให้การใช้งานร่วมกันทำได้ยากขึ้น ทำให้เราต้องมาหาวิธีใช้งานโปรโตคอลมาตรฐานเหล่านั้นมาเอง

2.3 ภาษา C++

Visual C++ ได้รับการพัฒนาขึ้นมาจาก Microsoft C/C++ ให้เป็น IDE ที่ทำงานบนระบบปฏิบัติงานวินโดวส์ได้อย่างเต็มที่ รองรับการพัฒนาโปรแกรมบนวินโดวส์โดยมี MFC (Microsoft Foundation Class) เป็นไลบรารีที่จะช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดวส์

ในตอนนี้ถือได้ว่า MFC นั้นเป็นส่วนหนึ่งของ Visual C++ เลยทีเดียวเพราะ Visual C++ เป็นเครื่องมือในการพัฒนาโปรแกรมบนวินโดวส์ที่มี MFC เป็นไลบรารีที่ช่วยอำนวยความสะดวกแทนการใช้ SDK ซึ่งทำให้ MFC ก็ได้รับการพัฒนาให้มีประสิทธิภาพสูงขึ้นไปพร้อมกับ Visual C++ จนถึงทุกวันนี้

ข้อดีของการเขียนโปรแกรมด้วย Visual C++ และ MFC อีกข้อหนึ่งคือ ความสามารถในการพอร์ตเทเบิล (Portability) หมายความว่า หากเรามีซอร์สโค้ดที่เขียนด้วย Visual C++ ในเวอร์ชั่นต่ำกว่าได้ เราสามารถนำมาคอมไพล์และลิงค์ใหม่ได้ โดยใช้ Visual C++ หรือที่สูงกว่านี้ได้ และการเขียนโปรแกรมด้วยภาษา C++ นี้จะเป็นการเขียนโปรแกรมแบบโครงสร้าง (Structure) เป็นการรวมข้อมูลที่แตกต่างกันรวมเข้าไว้ด้วยกัน ภายใต้ชื่อเดียวกัน ซึ่งจะช่วยอำนวยความสะดวกในการเขียนโปรแกรมได้ง่ายขึ้นและความถูกต้องมากยิ่งขึ้น ลักษณะการจัดข้อมูลแบบโครงสร้างนี้ มีลักษณะเหมือนกับการจัดข้อมูลแบบ Record ในภาษาปาสคาล และแนวคิดของโครงสร้างข้อมูลนี้ก็เป็นพื้นฐานสำคัญในการพัฒนาแนวคิดการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ซึ่งเป็นแนวคิดในการขียนโปรแกรมแบบหนึ่ง ที่ผู้รู้หลายๆท่านได้สรุปหรือได้ให้คำนิยามไว้ว่า เป็นการเขียนโปรแกรมเชิงวัตถุที่มีลักษณะที่คล้ายกับธรรมชาติของสิ่งของสิ่งหนึ่งซึ่งเราสามารถแบ่งแยกสิ่งต่างๆ ออกเป็นประเภทๆได้ ถ้าเราได้นำเอาแนวคิดของ OOP มาใช้ในการเขียนโปรแกรมและการจัดการข้อมูล เราจะพบว่าโปรแกรมหรือฟังก์ชันจะมีความเป็นอิสระแก่กันอย่างเห็นได้ชัด อธิบายง่ายๆก็คือโปรแกรมหรือฟังก์ชันแต่ละตัวถึงแม้จะมาจากที่เดียวกันแต่มันสามารถทำงานในคนละหน้าที่ เก็บข้อมูลคนละค่าได้ โดยจะไม่มายุ่งเกี่ยวกันแต่อย่างใด

Visual C++ 6.0 มีรูปแบบการอินเตอร์เฟสคล้ายกับ Visual C++ เวอร์ชั่นก่อน แต่จะมีจุดหนึ่งที่แตกต่างกันคือ ระบบช่วยเหลือของ Visual C++ 6.0 นั้นจะแยกออกจากตัวโปรแกรม (IDE) ซึ่งระบบ

ช่วยเหลือจะอยู่ในรูปของ HTML Help และในการที่เราจะใช้ Visual C++ 6.0 ได้นั้นเราจะต้องติดตั้งโปรแกรม Internet Explorer 4.0 (IE4) ลงไปก่อนด้วย หากในระบบของเราไม่มี IE4 โปรแกรมติดตั้ง Visual C++ จะทำการติดตั้งให้เราก่อน

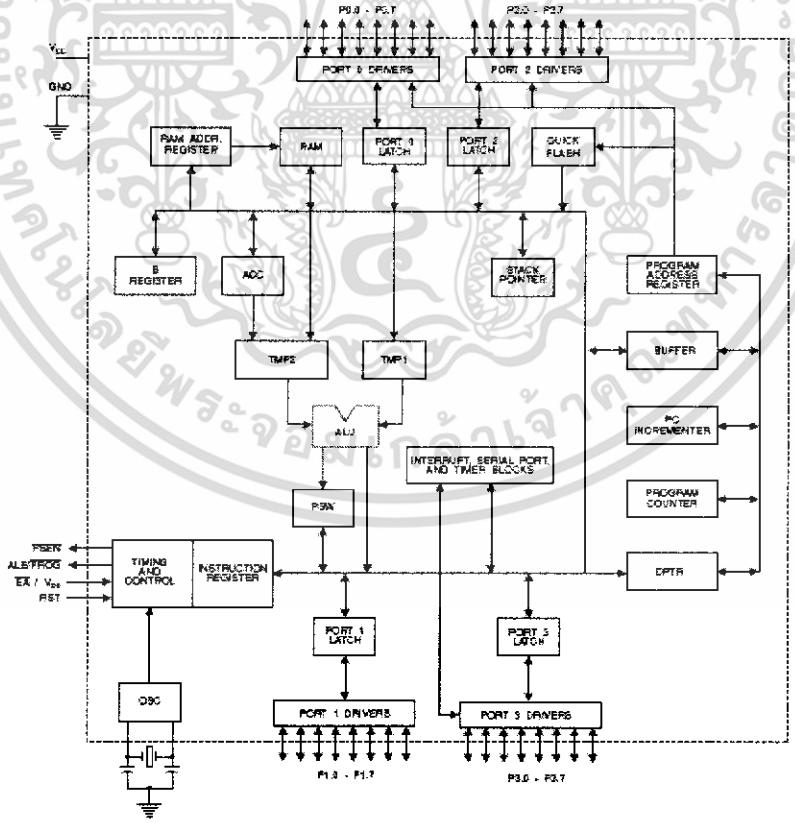
2.4 พื้นฐานไมโครคอนโทรลเลอร์ MCS - 51 สามารถอธิบายแยกย่อยได้ดังนี้

2.4.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาต์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ

2.4.2 โครงสร้างของ MCS - 51

ไมโครคอนโทรลเลอร์ที่เลือกใช้ในครั้งนี้เป็นไมโครคอนโทรลเลอร์ในตระกูล MCS – 51 ซึ่งมีอยู่ด้วยกันหลายเบอร์ขึ้นกับโครงสร้างภายในของมันโดยใช้เป็นส่วนควบคุมการทำงานของชุดควบคุมมอเตอร์ ซึ่งไมโครคอนโทรลเลอร์มีหน้าที่รับคำสั่งควบคุมจากคอมพิวเตอร์มาประมวลผลเพื่อกำหนดตำแหน่งให้กับมอเตอร์รวมทั้งส่งค่าตำแหน่งปัจจุบันของมอเตอร์ให้กับคอมพิวเตอร์



รูปที่ 2.16 โครงสร้างภายในของ MCS - 51

2.4.3 คุณสมบัติที่สำคัญของ MCS – 51 มีดังนี้

- มีหน่วยความจำ ROM 4 k byte
- มีหน่วยความจำ RAM 128 byte
- มี Port I/O ขนาด 8 บิต 4 Port
- มี Timer 16 บิต 2 ตัว
- สามารถ Interrupt ได้ 5 แหล่ง
- มีวงจรออสซิลเลเตอร์และวงจรนาฬิกาบนชิพ
- มี Port อนุกรมที่สามารถรับส่งข้อมูลแบบ Full Duplex ความเร็วสูง
- อ้างหน่วยความจำโปรแกรมภายนอกได้ 64 k
- อ้างหน่วยความจำข้อมูลภายนอกได้ 64 k
- สามารถประมวลผลทีละบิตได้
- สามารถอ้างหน่วยความจำแต่ละบิตได้ 210 ตำแหน่ง
- หนึ่งวัฏจักรคำสั่งกินเวลาประมาณ 1 ไมโครวินาที ขณะทำงานด้วย Clock 12 MHz

เบอร์พื้นฐานของไมโครคอนโทรลเลอร์ MCS – 51 ประกอบด้วย 8051, 8031, 8751 ซึ่งแตกต่างกันที่ชนิดและหน่วยความจำภายใน

รูปที่ 2.17 ตำแหน่งของของไมโครคอนโทรลเลอร์ตระกูล MCS - 51

2.4.4 หน้าที่การใช้งานของแต่ละขาของ MCS - 51

- ขา Vss (ขา 20) สำหรับต่อลง Ground
- ขา Vcc (ขา 40) สำหรับต่อแหล่งจ่ายไฟตรง +5 V
- ขาพอร์ต 0 (ขา 32 – 39) มี 8 ขา ใช้เป็นพอร์ต 0 (P0.0 – P0.7) เป็นได้ทั้งเอาต์พุตและอินพุตพอร์ตและใช้ในการติดต่อหน่วยความจำเป็นไบต์ต่ำ (A0 – A7)

- ขาพอร์ต 1 (ขา 1 – 8) มี 8 ขา ใช้เป็นพอร์ต 1 (P1.0 – P0.7) เป็นได้ทั้งเอาต์พุตและอินพุตพอร์ต โดยทั่วไปมีวงจรพูลอัพภายใน
- ขาพอร์ต 2 (ขา 21 – 28) มี 8 ขา ใช้เป็นพอร์ต 2 (P2.0 – P2.7) เป็นได้ทั้งเอาต์พุตและอินพุตพอร์ตและใช้ในการติดต่อหน่วยความจำเป็นไบต์สูง
- ขาพอร์ต 3 (ขา 10 – 17) มี 8 ขา ใช้เป็นพอร์ต 3 (P3.0 – P3.7) เป็นได้ทั้งเอาต์พุตและอินพุตพอร์ตโดยมีวงจรพูลอัพภายในและยังใช้ในหน้าที่พิเศษต่าง ๆ ดังนี้
  - ขา P3.0 ใช้รับข้อมูลภายนอกแบบอนุกรม
  - ขา P3.1 ใช้ส่งข้อมูลออกภายนอกแบบอนุกรม
  - ขา P3.2 ใช้เป็นอินพุตอินเตอร์รัปต์ชนิดที่ 0
  - ขา P3.3 ใช้เป็นอินพุตอินเตอร์รัปต์ชนิดที่ 1
  - ขา P3.4 สัญญาณอินพุตให้เคาน์เตอร์ของไทเมอร์ 0
  - ขา P3.5 สัญญาณอินพุตให้เคาน์เตอร์ของไทเมอร์ 1
  - ขา P3.6 ใช้เป็นสัญญาณการเขียนข้อมูลไปยังหน่วยความจำข้อมูลภายนอก
  - ขา P3.7 ใช้เป็นสัญญาณการอ่านข้อมูลไปยังหน่วยความจำข้อมูลภายนอก

การใช้งานพอร์ต 3 ในหน้าที่พิเศษนี้จะต้องโหลดค่า 1 ไปยังแต่ละบิตที่ต้องการใช้งานก่อนทุกครั้ง

- ขา RST (ขา 9) ใช้ในการรีเซตค่าภายในทั้งหมด
- ขา ALE/PROG (ขา 30) เป็นขาที่ใช้ในการส่งสัญญาณควบคุมการแลตช์ค่าแอดเดรสไบต์ต่ำ จาก P0 ในการติดต่อกับหน่วยความจำภายนอก
- ขา PSEN (ขา 29) ใช้ส่งสัญญาณสโตรบเพื่ออ่านคำสั่งจากโปรแกรม
- ขา EA/Vpp (ขา 31) เป็นขาสำหรับใช้เลือกให้ MCS – 51 ทำงานจากโปรแกรมภายในหรือภายนอก
- ขา XTAL (ขา 19) ใช้ต่อคริสตัลภายนอก โดยเป็นขาอินพุต
- ขา XTAL (ขา 18) ใช้ต่อคริสตัลภายนอก โดยเป็นขาเอาต์พุต

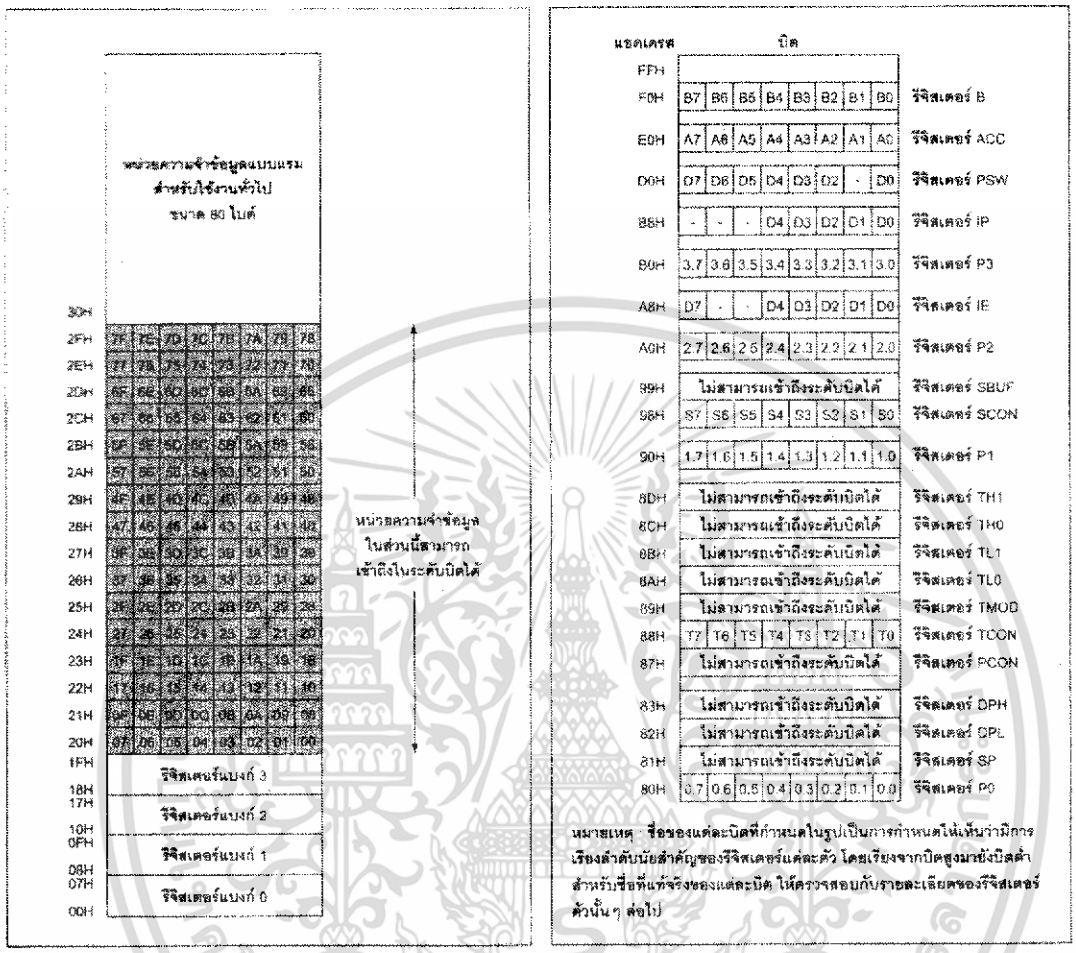
2.4.5 โครงสร้างหน่วยความจำใน MCS – 51

แบ่งออกเป็น 2 ส่วน คือ

1. หน่วยความจำโปรแกรม
2. หน่วยความจำข้อมูล

MCS – 51 ทุกเบอร์จะมีหน่วยความจำสำหรับเก็บข้อมูลภายในอย่างน้อย 128 ไบต์ไปจนถึง 256 ไบต์ หน่วยความจำสำหรับเก็บข้อมูลภายในบริเวณ 128 ไบต์แรกเรียกว่า Lower 128 และในบริเวณ 128 ไบต์หลังที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า Upper 128

หน่วยความจำสำหรับเก็บข้อมูลภายในบริเวณ 128 ไบต์แรกและหลังจะมีวิธีการเข้าถึงข้อมูลในหน่วยความจำทั้งสองส่วนไม่เหมือนกัน



รูปที่ 2.18 ตำแหน่งหน่วยความจำของ MCS – 51

2.4.6 รีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Register: SFR)

ใน MCS – 51 รีจิสเตอร์จะใช้หน่วยความจำ RAM ภายในชิพ โดยส่วนหนึ่งรีจิสเตอร์พิเศษต่าง ๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80 H ถึง FFH ซึ่งหน่วยความจำมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังก์ชันพิเศษเพียง 2 ตำแหน่ง แต่ถ้าเป็น 8032/8051 จะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

สำหรับรายละเอียดเบื้องต้นของ SFR มีดังนี้

- **รีจิสเตอร์แสดงสถานะของโปรแกรม (Program Status Word)** รีจิสเตอร์นี้เรียกย่อ ๆ ว่า PSW จะอยู่ที่ตำแหน่ง D0H ซึ่งสามารถเข้าถึงข้อมูลระดับบิตได้ โดยรีจิสเตอร์ตัวนี้จะเป็นตัวบอกสภาวะต่าง ๆ ของไมโครคอนโทรลเลอร์ MCS – 51 แยกเป็นบิต คือ

1. แฟลกตัวทด (Carry Flag) เป็นบิตที่ 7 ของ PSW บิตนี้จะมีความสำคัญหากมีการกระทำทางคณิตศาสตร์ โดยบิตนี้จะเซตเมื่อเกิดการทดของบิตที่ 7 ขณะทำการบวกเลขหรือเซตเมื่อเกิดการยืมของบิตที่ 7 เมื่อเกิดการลบเลข
2. แฟลกตัวทดช่วย (Auxiliary Carry Flag) เป็นบิตที่ 6 เมื่อมีการบวกแบบ Binary Code Decimal (BCD) บิต AC หรือบิตตัวทดช่วยจะถูกเซตเมื่อมีการทดจากบิตที่ 3 ไปบิตที่ 4 หรือถ้าใน Low Nibble มีค่าระหว่าง 0AH – 0FH BCD BCD DAA BCD
3. แฟลกศูนย์ เป็นบิตที่ 5 ของ PSW เป็นแฟลกใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่าจะกระทำคำสั่งใด ๆ ที่ บิตนี้จะไม่มีการเปลี่ยนแปลง
4. บิตเลือกรีจิสเตอร์แบงค์ (Register Select 1) เป็นบิตที่ 4 และ 3 ทำให้เลือกใช้รีจิสเตอร์ R0 – R7 ได้ 4 ชุด
5. แฟลกโอเวอร์โฟลว บิตที่ 2 จะถูกเซตเมื่อกระทำทางคณิตศาสตร์แล้วเกิดโอเวอร์โฟลว คือ จำนวนที่ได้จากการกระทำมีค่าเกินกว่าจำนวนไบต์ที่จะเป็นไปได้ คือ มากกว่า 128 หรือน้อยกว่า – 128
6. บิตพาริตี้ (Parity Bit) เป็นบิตที่บอกค่าพาริตี้ของรีจิสเตอร์ A โดยเป็นบิตที่ 0 ของ PSW

- **แอกคิวมูเลเตอร์ (Accumulator : ACC)** มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์ รีจิสเตอร์นี้สามารถเข้าถึงได้ในระดับบิต
- **รีจิสเตอร์ B** อยู่ที่ตำแหน่ง F0H เป็นรีจิสเตอร์ที่สามารถใช้งานได้ทั่วไปได้ แต่ส่วนใหญ่จะใช้รีจิสเตอร์นี้คูณหรือหารกับรีจิสเตอร์ A
- **สแต็กพอยน์เตอร์ (Stack Pointer : SP)** เป็นรีจิสเตอร์ขนาด 8 บิต อยู่ที่ตำแหน่ง81 H รีจิสเตอร์นี้ชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data Pointer : DPTR) เป็นรีจิสเตอร์ที่ใช้สำหรับชี้ตำแหน่งรหัสโปรแกรมหรือข้อมูลในหน่วยความจำ โดยเป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งประกอบด้วยรีจิสเตอร์ 2 ตัว คือ DPL อยู่ตำแหน่งที่ 82H โดยจะเก็บเป็น 8 บิตต่ำและ DPH ตำแหน่งที่ 83H โดยจะเก็บค่า 8 บิตสูง
- **รีจิสเตอร์พอร์ต (Port Register)** มีขนาด 8 บิตที่ใช้เก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS – 51 มี 4 ตัว คือ รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิสเตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H และรีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์ทุกตัวสามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหรือเขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องกระทำผ่านรีจิสเตอร์นี้ทุกครั้ง
- **รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)**
- **รีจิสเตอร์ไทเมอร์ (Timer Register)** ใน MCS – 51 เบอร์ 8051 จะมีรีจิสเตอร์ที่ใช้นับ

เวลาขนาด 16 บิตอยู่ 2 ตัว คือ Timer 0 อยู่ที่ตำแหน่ง 8AH และ 8CH โดยหมายถึง TL0 และ TH0 ตามลำดับ รีจิสเตอร์อีกตัวคือ Timer 1 อยู่ที่ตำแหน่ง 8BH เป็น TL1 และ 8DH เป็น TH1 การใช้งาน Timer จะต้องกำหนดการทำงานในรีจิสเตอร์ TMOD ซึ่งอยู่ตำแหน่งที่ 88H ก่อน

- **รีจิสเตอร์ควบคุม (Control Register)** แบ่งเป็น

1. รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการกำหนดอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์ MCS – 51
2. รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS – 51
3. รีจิสเตอร์ TCON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS – 51
4. รีจิสเตอร์ TMOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะในการทำงานของไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS – 51
5. รีจิสเตอร์ IE หรือ IP เป็นรีจิสเตอร์ที่เกี่ยวข้องกับการสนองการอินเตอร์รัปต์ โดย IE เป็นรีจิสเตอร์สำหรับเอ็นเอเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะให้ซีพียูตอบสนองการเกิด อินเตอร์รัปต์ในลักษณะใดก่อนหลัง

2.5 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS – 51

ไมโครคอนโทรลเลอร์ตระกูล MCS – 51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (สื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS – 51 แบบแฟลชเป็นอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมไมโครคอนโทรลเลอร์ MCS – 51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS – 232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐานของ RS – 422 หรือ RS – 485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

2.5.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

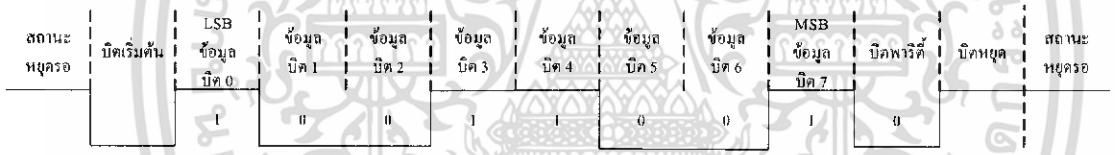
การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การรับและส่งข้อมูลโดยไม่ต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดค่าอัตราความเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (Baud Rate) มีหน่วยเป็น บิตต่อวินาที (bit per second: bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (Start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิต หรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (Stop bit) มีขนาด 1 บิต

รูปที่ 2.19 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัสเมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะหยุดรอ (Waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (Start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อนซึ่งข้อมูลที่ต้องการส่งมีทั้งหมด 8 บิต จากนั้นตามด้วยบิตพาริตี้ (Parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่ง คือ บิตปิดท้ายหรือบิตหยุด (Stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิตหรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS – 232 มีด้วยกันหลายค่า ตั้งแต่ 110 ถึง 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์



รูปที่ 2.19 รูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก "1" ทั้งหมดภายในข้อมูลที่ส่งไป 1 ไบต์รวมพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter : เป็นอุปกรณ์ที่ใช้ในการรับส่งข้อมูลอนุกรม) ซึ่งทางภาครับต้องกำหนดการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าจะทำการตรวจสอบพาริตี้คี่หรือคู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้ว่าเป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้เป็น NONE นั้น ทั้งภาครับและส่งจะไม่มีการตรวจสอบพาริตี้

2.5.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS – 51

ในการทำงานของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS – 51 มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว ดังนี้

- **รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial Data Buffer Register)**

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลและรับข้อมูล เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับข้อมูล เพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลช

- **รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial Port Control Register)** เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 - SM1 (Serial port mode bit 0 – 1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS – 51 ดังมีรายละเอียดต่อไปนี้

SM2 : ใช้ในการเอ็นเอเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS – 51 ถ้าบิตนี้เป็น "1" บิต RI จะไม่แอคตีฟ ถ้าบิตที่ 9 ที่รับเข้ามาเป็น "0" ในการทำงานโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอคตีฟถ้ายังไม่ได้รับบิตหยุดส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN (Enable serial reception) : ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมใน MCS – 51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS – 51 แต่ถ้าหากพอร์ตอนุกรมทำงานในโหมด 1 และ บิต SM2 เป็น "0" ข้อมูลที่บิตRB8 คือ ข้อมูลของบิตหยุด สำหรับในการทำงานในโหมด 0 บิตนี้ไม่ใช้งาน บิต RB8 นี้สามารถเซตด้วยกระบวนการทางซอฟต์แวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS – 51 สามารถเซตได้ด้วยกระบวนการฮาร์ดแวร์ เมื่อมีการส่ง

ข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรม สามารถเซตได้ด้วยกระบวนการฮาร์ดแวร์ เมื่อมีการรับข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นกรณีที่บิต SM2 มีการเซต บิตนี้จะเซต ได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

2.5.3 โหมดการทำงานของพอร์ตอนุกรมใน MCS – 51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS – 51 สามารถเลือกการทำงานได้ถึง 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์ การทำงานในโหมดนี้ของไมโครคอนโทรลเลอร์ MCS – 51 จะใช้ในการเชื่อมต่อกับไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายพอร์ตอินพุตหรือเอาต์พุต แต่ไม่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรลเลอร์ MCS – 51 เองมีพอร์ตอยู่ค่อนข้างมาก และติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก
2. กำหนดจากอัตราการเกิดโอเวอร์โฟลว (Overflow) ของไทเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์ เบอร์ AT98C52 และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 หรือไทเมอร์ 2 ในการกำหนดอัตราบอดได้ การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดทำได้ด้วยการกำหนดข้อมูลให้แก่บิต SM0 และ SM1 ในรีจิสเตอร์ SCON

2.5.4 อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS – 51

- **โหมด 0**

อัตราบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด = ความถี่ของสัญญาณนาฬิกา/2 หน่วยเป็น บิตต่อวินาที

- **โหมด 1 และ 3**

เนื่องจากทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่ง คือ จากอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลวของไทเมอร์ 1 จะต้องใช้ค่าบิต SMOD ในรีจิสเตอร์ PCON มาพิจารณาประกอบด้วย สามารถคำนวณค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าของบิต SMOD}}/32) * \text{อัตราโอเวอร์โฟลวของไทเมอร์ 1}$$

ถ้าหากในไทเมอร์ 1 ไม่ได้เอ็นเอเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณค่าอัตราบอดได้จาก

$$\text{อัตราบอด} = (2^{\text{ค่าของบิต SMOD}}/32) * (\text{ความถี่สัญญาณนาฬิกา}/\{12x[256 - (TH1)]\})$$

- **โหมด 2**

ในโหมดนี้อัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น "0" อัตราบอดจะเท่ากับ 1/64 ของสัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

$$\text{อัตราบอด} = (2^{\text{ค่าของบิต SMOD}}/64) * \text{ความถี่สัญญาณนาฬิกา}$$

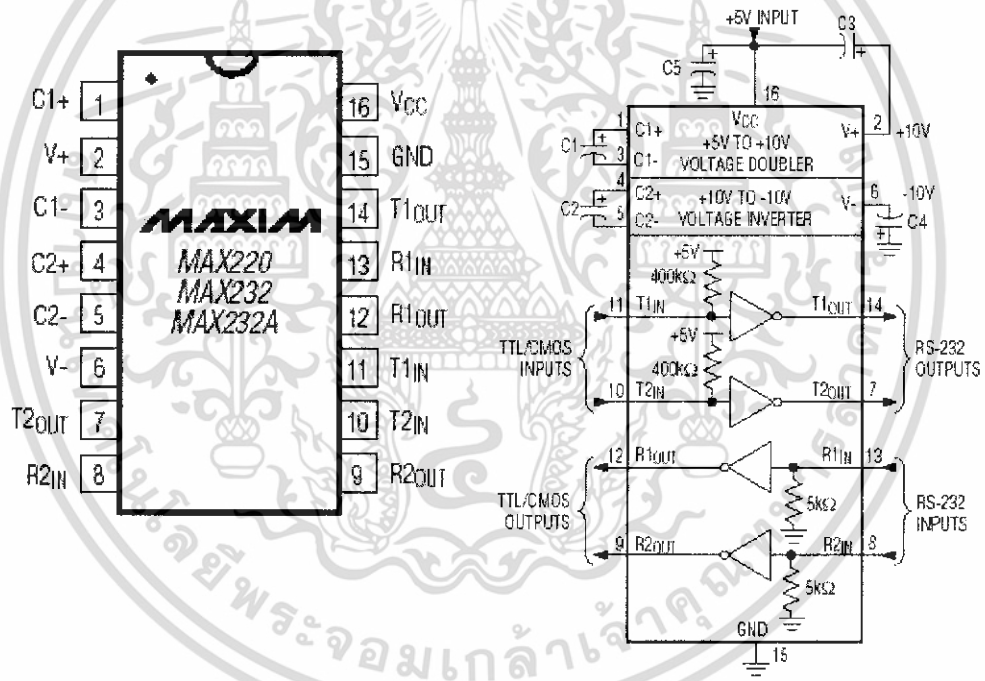
อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	×	×	×	×
โหมด 2 : สูงสุด 375 kHz	12 MHz	1	×	×	×
โหมด 1, 3 : 62.5 kHz	11.0592 MHz	1	0	2	FFH
19.2 k (19,200)	11.0592 MHz	1	0	2	FDH
9.6 k (9,600)	11.0592 MHz	0	0	2	FDH
4.8 k (4,800)	11.0592 MHz	0	0	2	FAH
2.4 k (2,400)	11.0592 MHz	0	0	2	F4H
1.2 k (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

รูปที่ 2.20 การเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS – 51

2.5.5 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS – 51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS – 232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS – 232 มีระดับตั้งแต่ ±3 ถึง ±12 V ในขณะที่ระดับสัญญาณไมโครคอนโทรลเลอร์ MCS – 51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS – 51 เข้ากับพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

ไอซีพิเศษทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS – 51 จากระดับทีทีแอลไปเป็นระดับของ RS – 232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS – 232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS – 51 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2.21 แสดงการจัดขาของไอซี ICL232 ซึ่งใช้ในการแปลงสัญญาณ RS – 232



รูปที่ 2.21 รายละเอียดเบื้องต้นของไอซีแปลงสัญญาณเพื่อเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

2.6 ความถี่ที่ใช้ในการสื่อสารวิทยุโทรทัศน์

การสื่อสารด้วยคลื่นวิทยุ แบ่งตามย่านความถี่คลื่นวิทยุ หรือเรียกว่าคลื่นพาห์ ดังรูปที่ 2.22 โดยแบ่งออกเป็นหลายๆ ย่านความถี่ เช่นย่านความถี่อินฟราเรด เริ่มจาก 30 – 300 GHz หรือเรียกว่าย่าน EHF ซึ่งมีความยาวคลื่นสั้นมาก ต่อจากย่านอินฟราเรดจะเป็นความถี่แสง เช่นรังสีอุลตราไวโอเรต , รังสีเอกซ์เรย์ รังสีแกรมมา และรังสีคอสมิก

ความถี่	ชื่อเต็ม	ชื่อย่อ	หมายเหตุ
30 – 300 Hz	Extremely low frequency	ELF	
300 – 3,000 Hz	Voice frequency	VF	ย่านความถี่เสียงพูดของคน
3 – 30 KHz	Very low frequency	VLF	ความถี่ต่ำมาก
30 – 300 KHz	Low frequency	LF	ความถี่ต่ำ
300 KHz – 3 MHz	Medium frequency	MF	ความถี่ระดับกลาง
3 – 30 MHz	High frequency	HF	ความถี่สูง
30 – 300 MHz	Very high frequency	VHF	ความถี่สูงมาก
300 MHz – 3 GHz	Ultra high frequency	UHF	ความถี่สูงยิ่ง
3 – 30 GHz	Super high frequency	SHF	
30 – 300 GHz	Extra high frequency	EHF	

รูปที่ 2.22 แถบความถี่วิทยุ

รายระเอียดการใช้งานความถี่แต่ละย่านจากรูปที่ 2.22 องค์กรควบคุมความถี่วิทยุ หรือ (Federal Communication Commission) ของอเมริกาเป็นผู้กำหนดเป็นมาตรฐาน โดยองค์กรควบคุมการใช้ความถี่ของแต่ละประเทศนำไปพิจารณาใช้งานตามความเหมาะสม กรมไปรษณีย์โทรเลข เป็นผู้ดูแล และจัดสรรความถี่ของไทย

2.6 ประวัติความเป็นมาของ PHP

PHP ย่อมาจาก Professional Home Page ซึ่งเป็นภาษาจำพวก Script Language คำสั่งต่างๆ จะเก็บอยู่ในไฟล์ที่เรียกว่า สคริปต์ (Script) และเวลาใช้งานต้องอาศัยตัวแปรชุดคำสั่ง ซึ่งทำงานโดยการสั่งงานจากเว็บเพจ แล้วส่งไปประมวลผลที่เว็บเซิร์ฟเวอร์สำหรับแสดงเว็บเพจอย่างหนึ่ง ที่จัดอยู่ในกลุ่มเซิร์ฟเวอร์ไซด์สคริปต์(Server Side Script)และจะทำงานในฝั่งเซิร์ฟเวอร์แล้วส่งการแสดงผลมายังเบราเซอร์ของตัวเครื่องลูกข่ายนอกจากนี้ยังเป็นสคริปต์ที่ติดตั้งบน HTML อีกด้วย ส่วนเลขที่ต่อท้ายก็หมายถึงเวอร์ชั่นนั่นเอง และในปัจจุบันนี้เป็นที่นิยมในกลุ่มนักสร้างเว็บทั่วโลก ตัวอย่างของภาษาสคริปต์ก็ เช่น จาวาสคริปต์, เพิร์ล, ASP (Active Server Page) เป็นต้น

2.6.1 ประโยชน์ที่ได้รับจาก PHP

ในปัจจุบัน เว็บไซต์ ต่างๆได้มีการพัฒนาในด้านต่างๆ อย่างรวดเร็ว เช่น เรื่องของความสวยงามและแปลกใหม่ การบริการข่าวสารข้อมูลที่ทันสมัย เป็นสื่อกลางในการติดต่อ และสิ่งหนึ่งที่กำลังได้รับความนิยม เป็นอย่างมากซึ่ง ถือได้ว่าเป็นการปฏิวัติรูปแบบการขายของก็คือ พาณิชย์อิเล็กทรอนิกส์ ซึ่งเจ้าของสินค้าต่างๆ ไม่จำเป็นต้องมีร้านค้าจริงและไม่จำเป็นต้องจ้างคนขายของอีกต่อไป ร้านค้าและตัวสินค้านั้น จะไปปรากฏอยู่บน เว็บไซต์ แทน และการซื้อขายก็เกิดขึ้นบนโลกของอินเตอร์เน็ต แล้ว PHP ช่วยเราให้เป็นเจ้าของร้านบน อินเตอร์เน็ต ได้อย่างไร PHP เป็นภาษาสคริปต์ที่มีความสามารถสูง สำหรับการพัฒนา เว็บไซต์ และความสามารถที่โดดเด่นอีกประการหนึ่งของ PHP คือ เว็บเพจที่ทำงานร่วมกับฐานข้อมูลได้ ทำให้เอกสารของ HTML สามารถที่จะเชื่อมต่อกับระบบฐานข้อมูล (database) ได้อย่างมีประสิทธิภาพและรวดเร็ว จึงทำให้ความต้องการในเรื่องการจัดรายการสินค้าและรับรายการสั่งของตลอดจนการจัดเก็บ ข้อมูลต่างๆ ที่สำคัญผ่านทาง อินเตอร์เน็ต เป็นไปได้อย่างง่ายดาย

2.6.2 รายการระบบฐานข้อมูลที่ PHP สามารถเชื่อมต่อได้คือ

Oracle, Sybase, mSQL, MySQL, SOLID, ODBC, PostgreSQL, Adabas D, FilePro, Velocis, Informix, dbase, Unix dbm

2.6.3 เหตุผลที่ PHP ได้รับความนิยมคือ

1. ไม่มีค่าลิขสิทธิ์ เนื่องจากเป็น Open Source ดังนั้นทำให้ ระบบปฏิบัติการ Linux, โปรแกรมเว็บ Apache, โปรแกรมฐานข้อมูล mySQL, และ Server side script อย่าง PHP เป็นที่นิยมอย่างแพร่หลาย

2. PHP นำเอาข้อดีของทั้งภาษาซี เพิร์ลและจาวามาผนวกเข้าด้วยกัน ทำให้ทำงานได้รวดเร็วกว่า CGI หรือแม้แต่ ASP และมีประสิทธิภาพ โดยเฉพาะเมื่อใช้กับอาปาเช่เซิร์ฟเวอร์ (Apache Server) เพราะไม่ต้องใช้โปรแกรมจากภายนอก

3. Open Source การพัฒนาของโปรแกรมไม่ได้ยึดติดกับบุคคลหรือกลุ่มคนเล็กๆ แต่เปิดโอกาสให้โปรแกรมเมอร์ทั่วไปได้เข้ามาช่วยกันพัฒนา ทำให้มีคนใช้งานจำนวนมาก และพัฒนาได้เร็วขึ้น

4. Crossable Platform ใช้ได้กับหลายๆระบบปฏิบัติการไม่ว่าวินโดวส์ ลินุกซ์และยูนิกส์ หรืออื่นๆ โดยแทบจะไม่ต้องเปลี่ยนแปลงโค้ดคำสั่ง

5. เรียนรู้ง่าย เนื่องจาก PHP ฝังเข้าไปใน HTML และใช้โครงสร้างและไวยากรณ์ภาษาง่ายๆ

6. ใช้ร่วมกับ XML ได้ทันที

7. ใช้ร่วมกับฐานข้อมูลได้เกือบทุกยี่ห้อ ดังกล่าวไปแล้วข้างต้น

8. ใช้กับระบบแฟ้มข้อมูลได้

9. ใช้ร่วมกับข้อมูลตัวอักษรได้อย่างมีประสิทธิภาพ

10. ใช้กับโครงสร้างข้อมูลได้ทั้งแบบสเกลาร์ อาเรย์และอาเรย์หลายมิติ

11. ใช้กับการประมวลผลภาพได้

ข้อแตกต่างระหว่าง PHP กับภาษา HTML คือ สคริปต์ของภาษา PHP เป็นเซิร์ฟเวอร์ไซต์สคริปต์ (Server Side Script) โดยถูกเรียกให้ทำงานทางด้านฝั่งเซิร์ฟเวอร์ส่วนสคริปต์ของภาษา HTML เป็นไคลเอนต์ไซด์สคริปต์ (Client Side Script) นั่นคือ สคริปต์จะถูกเรียกทำงานทางฝั่งไคลเอนต์หรือทางฝั่งของบราวเซอร์

2.7.4 หลักการของฐานข้อมูลบนเว็บ(Web Database)

ในการเขียนโปรแกรมบนเว็บในปัจจุบันส่วนมากจะต้องมีการเก็บข้อมูลบางอย่างเอาไว้เพื่อใช้ต่อไป ซึ่งการเขียนระบบฐานข้อมูลด้วยตัวเองนั้นจะต้องออกรูปแบบของการเก็บข้อมูลเอง และในการนำข้อมูลฐานข้อมูลไปใช้นั้นย่อมเกิดการผิดพลาดได้ถ้าการเขียนโปรแกรมไม่สมบูรณ์เพียงพอ

ในการเขียนโปรแกรมบนเว็บในยุคแรกๆ การเก็บข้อมูลนั้น โดยมากจะใช้เท็กซ์ไฟล์ เช่น โปรแกรมเอดิเตอร์หรือโปรแกรมประมวลผลข้อมูลในไฟล์ที่ใช้เท็กซ์ไฟล์เก็บฐานข้อมูลนั้น การควบคุมเท็กซ์ไฟล์นั้นลำยากกว่าการควบคุมไบนารีที่มีฟิลด์และเรคคอร์ดเข้ามาช่วยควบคุมและโอกาสการเกิดความผิดพลาดมนหารควบคุมเท็กซ์ไฟล์นั้นมากกว่า

เมื่อเว็บไซต์เป็นแหล่งรวบรวมข้อมูลที่มีคุณค่า การใช้ระบบการจัดการฐานข้อมูลมาช่วยจัดการกับข้อมูลต่างๆจึงเป็นการทำให้การบริหารข้อมูลบนเว็บนั้นมีความสะดวกสบายมากขึ้นและโอการในการผิดพลาดนั้นน้อยลงมาก โปรแกรมที่จะทำหน้าที่เป็นตัวกลางในการในการเรียกใช้ข้อมูลจากฐานข้อมูลและนำข้อมูลมาแสดงบนหน้าเว็บนั้นก็คือ โปรแกรมที่สร้างจากสคริปต์ PHP ดังที่ได้กล่าวเอาไว้ข้างต้นนั่นเอง

2.8 mySQL

mySQL เป็นฐานข้อมูลแบบเปิดเผยโค้ดที่ได้รับความนิยมในการใช้งานสูงสุดโปรแกรมหนึ่งบนเครื่องให้บริการ มีความสามารถในการจัดการกับฐานข้อมูลด้วยภาษา SQL (Structures Query Language) อย่างมีประสิทธิภาพ มีความรวดเร็วในการทำงาน รองรับการทำงานจากผู้ใช้หลายๆ คนและหลายๆ งานได้ในขณะเดียวกัน

mySQL ถูกพัฒนาขึ้นโดย mySQL AB โดยมีลิขสิทธิ์การใช้งาน 2 แบบ นั่นคือ ผู้ดูแลระบบสามารถใช้งานซอฟต์แวร์ mySQL ได้โดยไม่มีค่าใช้จ่ายใดๆ ภายใต้ลิขสิทธิ์ของ GNU General Public License (http://www.gnu.org/licenses/) หรืออาจเลือกใช้แบบที่มีลิขสิทธิ์ทางการค้าของ mySQL AB ซึ่งเป็นผู้ผลิตและพัฒนาซอฟต์แวร์โดยตรงก็ได้ หากไม่ต้องการเกี่ยวข้องกับข้อตกลงเรื่อง GPL รายละเอียดเพิ่มเติมเกี่ยวกับโปรแกรม mySQL สามารถหาข้อมูลได้จาก http://www.mySQL.com

คำอธิบายเพิ่มเติมเกี่ยวกับหน้าที่ ความสามารถและการทำงานของโปรแกรม mySQL มีดังต่อไปนี้

- mySQL ถือเป็นระบบจัดการฐานข้อมูล (DataBase Management System (DBMS))

ฐานข้อมูลมีลักษณะเป็นโครงสร้างของการเก็บรวบรวมข้อมูล การที่จะเพิ่มเติม เข้าถึงหรือประมวลผลข้อมูลที่เก็บในฐานข้อมูลจำเป็นจะต้องอาศัยระบบจัดการฐานข้อมูล ซึ่งจะทำหน้าที่เป็นตัวกลางในการจัดการกับข้อมูลในฐานข้อมูลทั้งสำหรับการใช้งานเฉพาะ และรองรับการทำงานของแอพลิเคชันอื่นๆ ที่ต้องการใช้งานข้อมูลในฐานข้อมูล เพื่อให้ได้รับความสะดวกในการจัดการกับข้อมูลจำนวนมาก mySQL ทำหน้าที่เป็นทั้งตัวฐานข้อมูลและระบบจัดการฐานข้อมูล

- mySQL เป็นระบบจัดการฐานข้อมูลแบบ เชิงความสัมพันธ์

ฐานข้อมูลแบบ เชิงความสัมพันธ์ จะทำการเก็บข้อมูลทั้งหมดในรูปแบบของตารางแทนการเก็บข้อมูลทั้งหมดลงในไฟล์เพียงไฟล์เดียว ทำให้ทำงานได้รวดเร็วและมีความยืดหยุ่น นอกจากนั้น แต่ละตารางที่เก็บข้อมูลสามารถเชื่อมโยงเข้าหากันทำให้สามารถรวมหรือจัดกลุ่มข้อมูลได้ตามต้องการ โดยอาศัยภาษา SQL ที่เป็นส่วนหนึ่งของโปรแกรม mySQL ซึ่งเป็นภาษามาตรฐานในการเข้าถึงฐานข้อมูล

- mySQLแจกจ่ายให้ใช้งานแบบเปิดเผยโค้ด

ผู้ใช้งาน mySQL ทุกคนสามารถใช้งานและปรับแต่งการทำงานได้ตามต้องการ สามารถดาวน์โหลดโปรแกรม mySQL ได้จากอินเทอร์เน็ตและนำมาใช้งานโดยไม่มีค่าใช้จ่ายใดๆ

2.9 เว็บเซิร์ฟเวอร์อาปาเช่ (Apache Web Server)

อาปาเช่เป็นเว็บเซิร์ฟเวอร์ที่ใช้งานมากที่สุดในอินเทอร์เน็ต โดยจากการสำรวจของ NetCraft.com ในเดือนกรกฎาคม 2544 พบว่า มีผู้ใช้อาปาเช่เป็นเว็บเซิร์ฟเวอร์ถึง 62.81% ในขณะที่ Microsoft's IIS และ Netscape มีผู้ใช้ราว 19.86% และ 6.91% ตามลำดับ (ข้อมูลจาก http://www.netcraft.com/survey/index-200007.html)

การเผยแพร่เวอร์ชั่นแรกของ Apache คือ v 0.6.2 ในเดือนเมษายน 1995 และจากนั้น Apache 1.0 ก็ได้ถูกเผยแพร่เมื่อ 1 ธันวาคม 1995 และได้รับความนิยมอย่างรวดเร็วภายในเวลา 1 ปี กลายเป็นเว็บเซิร์ฟเวอร์ที่มีผู้ใช้งานมากที่สุด ปัจจุบัน The Apache Software Foundation เป็นผู้ดูแลโครงการ Apache HTTP server ซึ่งมีจุดประสงค์เพื่อสร้างเว็บเซิร์ฟเวอร์ที่มีความทนทานต่อการใช้งาน มีคุณภาพในระดับของการใช้งานในเชิงพาณิชย์มีความสามารถที่น่าใช้งานและสามารถเปิดเผย โค้ดได้ ทั้งนี้สามารถใช้อาปาเช่เว็บเซิร์ฟเวอร์ได้ฟรีภายใต้ข้อกำหนดของ Apache Software License

อย่างไรก็ตามการที่เราจะติดตั้งเว็บเซิร์ฟเวอร์ให้มีความปลอดภัยนั้น ก็ไม่ควรที่จะติดตั้งเซอร์วิสอื่นๆ ที่ไม่มีความจำเป็น เช่น ftp, mail, DNS ซึ่งถ้ามีความจำเป็นต้องติดตั้งก็ควรติดตั้งแยกเครื่องกันต่างหาก ทั้งนี้รวมไปถึงการไม่ติดตั้งแอพลิเคชันที่ไม่จำเป็นรวมทั้งคอมไพเลอร์ด้วย

2.8.1 คุณสมบัติเด่นที่ทำให้เลือกใช้อาปาเช่

- เป็นโปรแกรมระบบเว็บเซิร์ฟเวอร์ตามมาตรฐานโปรโตคอล HTTP/1.1
- มีระบบโมเดลให้ผู้ใช้สามารถเขียนโปรแกรมเพิ่มเติมความสามารถให้กับอาปาเช่ได้เอง ซึ่งในปัจจุบัน

ผู้ใช้ทั้งหลายได้เขียนโมดูลต่างๆออกมาเป็น Third-party module อย่างมากมาย

- มีระบบ DBM หรือ Database for Authentication ให้เรากำหนดรหัสผ่านสำหรับอนุญาตและป้องกันการเรียกดูเพจต่างๆ ของโฮมเพจแต่ละหน้าให้เฉพาะผู้ใช้ที่ต้องการและไม่ต้องการ
- มีระบบ Multiple Directory Index คือสามารถกำหนดชื่อไฟล์เพื่อเชื่อมกับ URL ที่เป็นแบบไดเร็กทอรี่ได้มากกว่าหนึ่งชื่อไฟล์
- มีระบบ Multiple-homed server ความสามารถนี้เป็นที่ต้องการอย่างยิ่งในปัจจุบัน คือ อาปาเช่สามารถตอบสนองต่อเครื่องไคลเอนต์ต่างๆให้แลดูเสมือนเป็นเว็บเซิร์ฟเวอร์หลายๆเครื่องพร้อมกันได้โดยติดตั้งอาปาเช่ให้กับเครื่องเซิร์ฟเวอร์เพียงเครื่องเดียว

2.9 Thick film gas sensor

เป็นอุปกรณ์สารกึ่งตัวนำชนิดหนึ่ง ซึ่งนำไปใช้ในงานประเภทเซ็นเซอร์ก๊าซ ซึ่งเป็นอุปกรณ์ที่มีคุณภาพสูง และได้รับความเชื่อถือมายาวนาน

อุปกรณ์เซ็นเซอร์ชนิดนี้จะมีฐานรอง (Substrate) ซึ่งมีชั้นของเซรามิคที่มีลักษณะเป็นรูพรุนเล็กๆ ปกคลุมอยู่เพื่อจะใช้สำหรับป้องกันฝุ่นละออง สสารต่างๆ หรือ ก๊าซที่ไม่พึงประสงค์ นอกจากนั้นยังมี Micro heater และตัวเซ็นเซอร์เชื่อมต่อกับพื้นรอง (Substrate) Micro heater จะใช้เป็นแหล่งกำเนิดความร้อนในช่วงอุณหภูมิที่จะทำให้ตัวเซ็นเซอร์เกิดการเปลี่ยนแปลงความต้านทาน

พื้นฐานในการวัดด้วยก๊าซเซ็นเซอร์

ในสภาวะอากาศปกติตัวก๊าซเซ็นเซอร์จะมีค่าความต้านทานทางไฟฟ้าสูง (High electrical) การตรวจวัดปริมาณสารมีจุดมุ่งหมายสำคัญเพื่อวิเคราะห์ผลิตภัณฑ์หรือตัวแปรต่างๆ ทั้งในระหว่างกระบวนการผลิตและเมื่อสิ้นสุดกระบวนการ รูปแบบและวิธีการวัดมีการพัฒนาอย่างรวดเร็ว ไม่ว่าจะเป็นการวัดตัวแปรทางกายภาพและทางเคมี ตัวอย่างการวัดทางกายภาพ เช่น การวัดปริมาตร มวล ความหนาแน่น อัตราการไหล ความดันและอุณหภูมิ เป็นต้น ในการวัดตัวแปรทางเคมี ในปัจจุบันมีเครื่องมือต่างๆ เช่น แก๊สและลิควิดโครมาโตกราฟี (Gas – Liquid chromatography) พีเอชมิเตอร์ (pH meter) อิออนซีเลคทีฟอิเล็กโตรด (Ion – selective electrode) และสเปคโตรโฟโตมิเตอร์ (Spectro photo meter) เป็นต้น เครื่องมือวัดเหล่านี้ทำงานได้ดี ทั้งการวัดตัวแปร หรือ สารตัวอย่างโดยตรงหรือมีการทำตัวอย่างเปลี่ยนสภาพที่ทำให้เครื่องมือวัดหรือเซ็นเซอร์ตอบสนอง

การใช้วัสดุในการวิเคราะห์ปริมาณสารที่เรียกว่า Chemical assay มักใช้เวลาในการวิเคราะห์ เช่น TGS sensors ของ Figaro Engineering Company ที่ใช้เนื้อสารของดีบุกออกไซด์ (SnO_2) เป็นตัวเพื่อวิเคราะห์คาร์บอนมอนอกไซด์ (CO) จากท่อไอเสียรถ ปัจจุบันมีการใช้หัววัดทางเคมีไปติดเข้ากับ

เครื่องมือทางไฟฟ้า ในการหาปริมาณสารตัวอย่างได้ในระยะเวลาอันรวดเร็ว เครื่องมือวัดแบบนี้เรียกว่า เคมีเซ็นเซอร์ (Chemsensors)

นิยามของ Gas Sensors

เซ็นเซอร์จะเป็นส่วนสำคัญสำหรับระบบการควบคุมทางอิเล็กทรอนิกส์ที่สามารถแสดงผลออกมาแทนการตอบสนองของประสาทสัมผัสของมนุษย์ที่นอกจากเราจะสามารถได้ยิน สัมผัส ดมกลิ่น การลิ้มรส และการมองเห็น เราอาจจะได้รับอันตรายจากการสัมผัสหรือได้รับสิ่งเหล่านั้นโดยตรงในปริมาณที่มากเกินขีดจำกัดที่ร่างกายจะรับได้ เซ็นเซอร์จะสามารถแสดงเป็นทางอิเล็กทรอนิกส์ที่ให้ค่าที่เท่ากับการตอบสนองของร่างกาย อย่างไรก็ตาม อาจกล่าวได้ว่าการพัฒนาเซ็นเซอร์ มีการพัฒนาเป็นอุปกรณ์ไมโครอิเล็กทรอนิกส์ ที่ให้สัญญาณเป็นดิจิตอลอย่างมาก โดยเฉพาะการทำตัวตรวจจับที่สามารถตอบสนองต่อสารเคมี และในกรณีที่มีการทำกล้อง CCD ที่สามารถแยกประสาทในการมองได้สูงมาก หรือแม้แต่การทำไมโครโฟนที่มีความไวในการตอบสนองเสียงได้ดี ซึ่งสิ่งเหล่านี้ยังเป็นขั้นเริ่มต้น

สัญญาณดิจิตอลในทางอิเล็กทรอนิกส์ของตัวตรวจจับทางเคมี ซึ่งจะสามารรู้รสและได้กลิ่นของสิ่งไม่มีชีวิต ถึงแม้แต่ในระบบชีววิทยายังมีระบบภูมิคุ้มกัน และการวัดปริมาณกลูโคส การวัดความเข้มข้นสารเคมี และการควบคุมปริมาณที่ต้องใช้ความถูกต้องสูง เพื่อที่จะแทนที่หรือเป็นตัววัดระบบทางชีววิทยาเหล่านี้ จนในที่สุดจะได้ตัวตรวจจับที่มีการตอบสนองและการแยกแยะที่ดี วัสดุตรวจจับทางเคมีในเชิงพาณิชย์จะต้องสามารถปรับค่าเริ่มต้น แต่ยังไม่สามารถแยกแยะ และมีความไวในการตอบสนองได้ดีพอ อย่างไรก็ตาม ถ้ามีสารชีววิทยาที่สามารถเป็นตัวตรวจจับทางเคมีและทางชีววิทยาที่สามารถแยกแยะสารและตอบสนองที่ดีที่เหมือนระบบชีววิทยานั้นได้รับการยอมรับในช่วงทศวรรษที่ผ่านมาได้มีการวิจัยทางนี้อย่างกว้างขวางและจนกระทั่งในปัจจุบันได้มีเซ็นเซอร์ออกสู่ตลาดหลายชนิด

ตัวอย่างเซ็นเซอร์ที่ได้รับการแพร่หลายมากที่สุดก็คือ กลูโคสเซ็นเซอร์ ในปัจจุบันเซ็นเซอร์ชนิดนี้สามารถสร้างได้ง่าย มีราคาถูก จนสามารถใช้แล้วทิ้งได้ เพื่อให้การตอบสนองที่ดีและมีความถูกต้องและยังมีการแสดงผลเป็นสัญญาณดิจิตอลเพื่อง่ายต่อการใช้

การแยกชนิดของเซ็นเซอร์และอุปกรณ์ในการตรวจจับสามารถแบ่งออกเป็นหลักกว้างๆ คือ ทรานสดิวเซอร์ (Transducer) เซ็นเซอร์ (Sensors) และแอคชิวเอเตอร์ (Actuators)

ความหมายของคำว่า ทรานสดิวเซอร์ (Transducers) มีรากศัพท์มาจากภาษาลาติน จากคำว่า "transduco - traduco" ซึ่งหมายถึง "การเปลี่ยนสภาพ - การแปลงสภาพ" ดังนั้น อุปกรณ์จะต้องเป็นการเปลี่ยนแปลงหรือการแปลงสภาพ พลังงานจากระบบหนึ่งไปเป็นระบบอื่น โดยที่รูปแบบอาจจะเปลี่ยนไปก็ได้ ซึ่งจะเป็นความหมายของ ทรานสดิวเซอร์ การวัดค่าต่างๆ ในสิ่งแวดล้อม ทรานสดิวเซอร์จะเป็นอุปกรณ์ที่แสดงการเปลี่ยนแปลงสัญญาณจากพลังงานรูปหนึ่งไปเป็นพลังงานอีกรูป โดยพลังงานมีอยู่หลายรูป เช่น ทางไฟฟ้า ทางกลศาสตร์ ทางแสง ทางความร้อน ทางแม่เหล็ก และทางการแผ่รังสี

เซ็นเซอร์ (Sensors) เป็นทรานซิสเตอร์ ที่แปลงสัญญาณในรูปแบบต่างๆ เป็นสัญญาณทางไฟฟ้า ส่วนแอคชิวเอเตอร์ (Actuators) จะเป็นทรานสดิวเซอร์ที่แปลงสัญญาณไฟฟ้าไปเป็นสัญญาณในรูปอื่น โดยทั่วไปจะเป็นสัญญาณในรูปกลศาสตร์ ตัวอย่างของเซ็นเซอร์ เช่น เซ็นเซอร์วัดความดัน pH เซ็นเซอร์

และโฟโต้ทรานซิสเตอร์ ตัวอย่างของแอคชิวเอเตอร์ (Actuators) เช่น โซลินอยย์ อุปกรณ์เพียร์อิเล็กตริก และเลเซอร์ไดโอด ส่วนในเคมีคอลแอคชิวเอเตอร์ (Chemical Actuators) เมื่อป้อนศักดาทางไฟฟ้าจะทำให้เกิดไฮโดรเจนและออกซิเจน จอภาพจะเป็นทรานสดิวเซอร์ชนิดพิเศษที่แปลงสัญญาณทางไฟฟ้าไปเป็นรูปแบบที่มองเห็น เช่น จอภาพ CRT,จอ LCD,LCD array ซึ่งจะเป็นอุปกรณ์แสดงผล ที่สามารถพบเห็นได้ทั่วไป

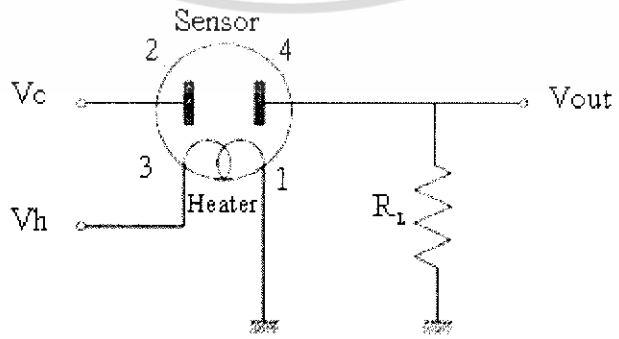
คุณลักษณะของ Thick film gas sensor

- มีความไวสูง และมีความตอบสนองได้อย่างรวดเร็ว
- เมื่ออุณหภูมิเปลี่ยนแปลงไป คุณลักษณะต่างๆมีการเปลี่ยนแปลงน้อยมาก
- มีอายุการใช้งานยาวนาน
- สามารถนำแรงดันที่ได้จากการเปลี่ยนแปลงของค่าความต้านทานของตัวเซ็นเซอร์ไปใช้งานได้เลย
- มีเสถียรภาพสูง

วงจรภายในของ Gas Sensor AF-Series ซึ่งจะเห็นได้ว่า วงจรภายในจะประกอบด้วย Heater และตัว Sensor และเป็นวงจรพื้นฐานที่จะนำไปต่อใช้งานขา1 กับ ขา3 จะต่อกับส่วนของ Heater และ ขา2 และขา4 จะต่อกับส่วนของ Sensor ซึ่ง Gas Sensor AF-Series นี้จะเสถียรภาพที่ดีเมื่อใช้งานที่แรงดัน Vc มีค่าเท่ากับ 5 โวลต์

เงื่อนไขการทดสอบ

- เป็นอากาศที่บริสุทธิ์ สะอาด ไม่มีก๊าซเจือปน อุณหภูมิ $25^{\circ}C \pm 2^{\circ}C$ และค่า $R.H.50 \pm 5\%$
- แรงดันของวงจร $5V \pm 0.05V$
- แรงดันของ Heater $5V \pm 0.05V$
- ความต้านทายนของโหลด R_L ประมาณ $5k\Omega - 10k\Omega \pm 10\%$
- ระยะเวลาที่ใช้ในการทดสอบมากกว่า 48 ชั่วโมง



รูปที่ 2.22 แสดงวิธีการใช้งานแก๊สเซ็นเซอร์แบบเบื้องต้นพื้นฐาน

จากรูป 2.23 เป็นการแสดงค่าความต้านทานที่แปรผันไปตามค่าความเข้มข้นของแก๊สโดยจะคิดความเข้มข้นในหน่วยของหนึ่งในล้าน

ความไวสามารถหาได้จาก

$$\text{Sensitivity} = \frac{R_{Gas}}{R_{Air}}$$

โดยที่

R_{Gas} : ความต้านทานของ Sensors (Rs) ในขณะที่อยู่ในบรรยากาศที่มีความหนาแน่นของก๊าซอยู่

R_{Air} : ความต้านทานของ Sensors (Rs) ในขณะที่อยู่ในบรรยากาศที่สะอาด บริสุทธิ์ ไม่มีก๊าซผสมอยู่

ลักษณะโครงสร้างของแก๊สเซ็นเซอร์

แก๊สเซ็นเซอร์ถูกสร้างขึ้นมาจาก สารกึ่งตัวนำชนิดเอ็น (N-Type) ซึ่งประกอบด้วยดีบุกออกไซด์ (Tin - Oxide : SnO_2) ซึ่งจะมีคุณสมบัติคือค่าความนำทางไฟฟ้า (Conductivity) จะเพิ่มขึ้นเมื่อมีพวกแก๊สติดไฟ (Combustible Gases) เข้ามาทำปฏิกิริยา อาทิเช่น ไฮโดรเจน คาร์บอนมอนอกไซด์ มีเทน ฯลฯ และรอบๆ ชิ้นสารกึ่งตัวนำจะมีลวดความร้อนล้อมรอบอยู่เพื่อทำหน้าที่ให้แก๊สและไอถูกขับและระเหยได้เร็วขึ้นและยังเป็นการลดผลของอุณหภูมิภายนอกและความชื้นที่จะมามีผลต่อการนำไฟฟ้าของชิ้นสารกึ่งตัวนำ

วิธีใช้งานของตัวแก๊สเซ็นเซอร์

การใช้งานแก๊สเซ็นเซอร์จะใช้แรงดัน 5 โวลท์ มาป้อนให้แก่ขดลวดแล้วต่อปลายอิเล็กโทรด (electrode) ทั้งสองด้านชิ้นสารกึ่งตัวนำไปหาไฟเลี้ยง (V_{cc}) โดยผ่านความต้านทานโหลด (R_L) ดังรูปที่ 2.23 ไฟเลี้ยงนี้ต้องไม่เกินกว่า 12 โวลท์ อย่างไรก็ตามโดยทั่วไปแล้วมักใช้ไฟเลี้ยงมีค่าไม่น้อยกว่า 5 โวลท์ และไม่มากกว่า 12 โวลท์

เนื่องจากคุณสมบัติด้านความไวของตัวแก๊สเซ็นเซอร์ขึ้นกับอุณหภูมิ และความชื้นมาก ดังนั้นเพื่อให้การทำงานได้ดีขึ้น และเชื่อถือได้สูงไฟเลี้ยงขดลวดความร้อนจึงควรทีค่าคงที่

จากรูป 2.23 เป็นวงจรพื้นฐานที่จะนำไปต่อใช้งานขา 1 กับ ขา 3 จะต่อกับส่วนของ heater และขา 2 กับขา 4 จะต่อกับส่วนของ Sensors ซึ่ง Gas Sensor AF series นี้จะมีเสถียรภาพที่ดีเมื่อใช้งานที่แรงดัน Vc มีค่าเท่ากับ 5 โวลท์ เมื่อมีแก๊สมาตกกระทบบริเวณตัวเซ็นเซอร์ จะทำให้ความต้านทานของตัวเซ็นเซอร์มีค่าลดลง ค่าความต้านทานของตัวเซ็นเซอร์ (Rs) สามารถคำนวณได้จากแรงเอ้าท์พุท (V_{out}) ตามสูตร

$$R_s = \left[\frac{V_c - V_{out}}{V_{out}}\right] R_L$$

เงื่อนไขในการทดสอบ

- เป็นอากาศบริสุทธิ์ สะอาด ไม่มีก๊าซอื่นเจือปน อุณหภูมิ $25°C \pm 2°C$ และค่า R.H. $50 \pm 5\%$
- แรงดันของวงจร $5V \pm 0.05V$

บทที่ 3
การคำนวณและการสร้าง

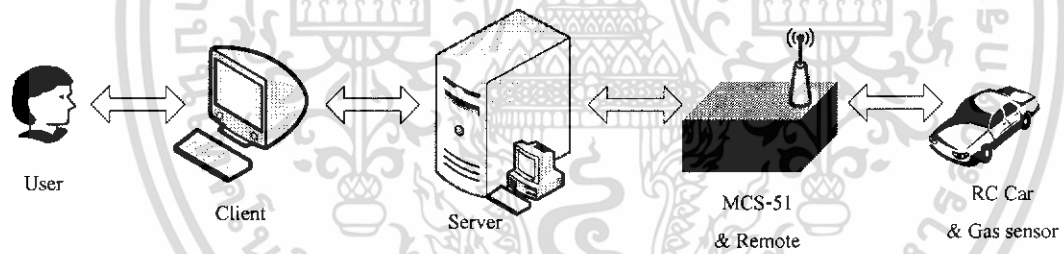
3.1 บล็อกไดอะแกรมการทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

โครงสร้างโดยรวมของโครงงานนี้มีหลักการทำงานดังนี้

1. ฝั่ง Client ควบคุมการเคลื่อนที่ของรถบังคับวิทยุโดย Visual C++ Version 6.0
2. ฝั่ง Server จะควบคุมไมโครคอนโทรลเลอร์ที่เชื่อมต่อกับ Remote control
3. ส่วนวงจรตรวจจับควันและส่งสัญญาณเตือนไปยัง Server ให้ทราบ

รูปที่ 3.1 บล็อกไดอะแกรมของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

3.2 การทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์



รูปที่ 3.2 การทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

การทำงานจะเริ่มต้นเมื่อผู้ใช้งานทำการเชื่อมต่อระหว่าง Client และ Server ผ่านทางหน้าเว็บไซต์เพื่อทำการควบคุมรถวิทยุบังคับ ผู้ใช้จะเป็นผู้สั่งการเคลื่อนที่ของรถวิทยุบังคับโดยการกดปุ่มทิศทางการเคลื่อนที่ทางฝั่ง Client และเมื่อทำการเชื่อมต่อไปยัง Server แล้ว Server จะส่งข้อมูลผ่านพอร์ตอนุกรมจากโปรแกรม Visual C++ เพื่อบังคับ Remote ให้รถเคลื่อนที่ตามที่ผู้ใช้งานบังคับจากฝั่ง Client ซึ่งการสั่งงานไปยัง Remote จะควบคุมโดยไมโครคอนโทรลเลอร์ และเมื่อทางรถบังคับที่มีวงจรตรวจจับควันสามารถจับควันไฟได้จะทำการส่งสัญญาณกลับไปยังตัวไมโครคอนโทรลเลอร์ที่ฝั่งเครื่องคอมพิวเตอร์เซิร์ฟเวอร์เพื่อแสดงขึ้นบนเว็บไซต์

บทที่ 3
การคำนวณและการสร้าง

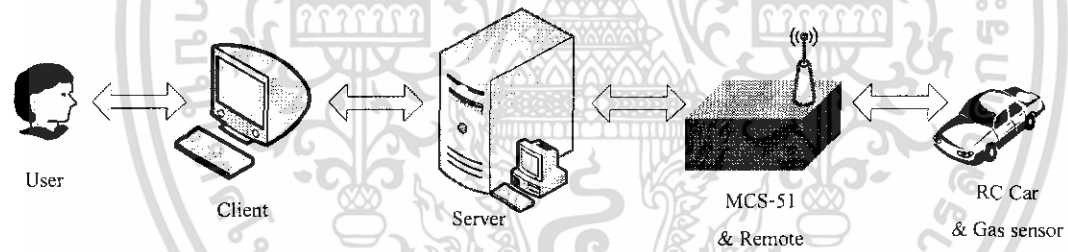
3.1 บล็อกไดอะแกรมการทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

โครงสร้างโดยรวมของโครงงานนี้มีหลักการทำงานดังนี้

1. ฝั่ง Client ควบคุมการเคลื่อนที่ของรถบังคับวิทยุโดย Visual C++ Version 6.0
2. ฝั่ง Server จะควบคุมไมโครคอนโทรลเลอร์ที่เชื่อมต่อกับ Remote control
3. ส่วนวงจรตรวจจับควันและส่งสัญญาณเตือนไปยัง Server ให้ทราบ

รูปที่ 3.1 บล็อกไดอะแกรมของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

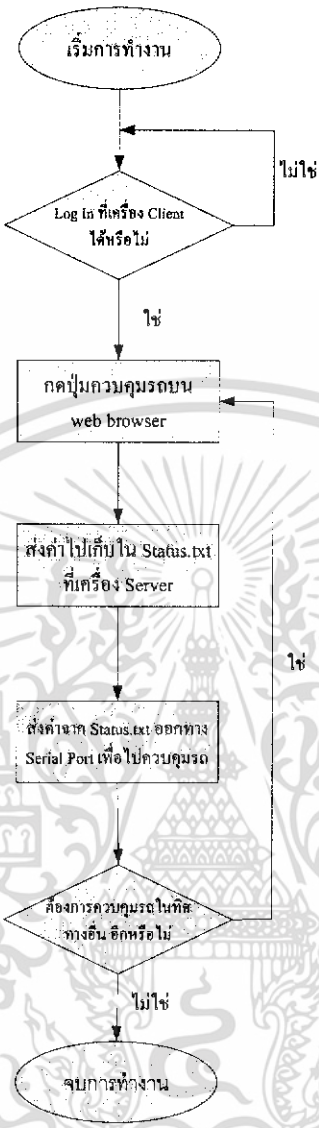
3.2 การทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์



รูปที่ 3.2 การทำงานของระบบควบคุมรถตรวจการณ์ผ่านเครือข่ายคอมพิวเตอร์

การทำงานจะเริ่มต้นเมื่อผู้ใช้งานทำการเชื่อมต่อระหว่าง Client และ Server ผ่านทางหน้าเว็บไซต์เพื่อทำการควบคุมรถวิทยุบังคับ ผู้ใช้จะเป็นผู้สั่งการเคลื่อนที่ของรถวิทยุบังคับโดยการกดปุ่มทิศทางการเคลื่อนที่ทางฝั่ง Client และเมื่อทำการเชื่อมต่อไปยัง Server แล้ว Server จะส่งข้อมูลผ่านพอร์ตอนุกรมจากโปรแกรม Visual C++ เพื่อบังคับ Remote ให้รถเคลื่อนที่ตามที่ผู้ใช้งานบังคับจากฝั่ง Client ซึ่งการสั่งงานไปยัง Remote จะควบคุมโดยไมโครคอนโทรลเลอร์ และเมื่อทางรถบังคับที่มีวงจรตรวจจับควันสามารถจับควันไฟได้จะทำการส่งสัญญาณกลับไปยังตัวไมโครคอนโทรลเลอร์ที่ฝั่งเครื่องคอมพิวเตอร์เซิร์ฟเวอร์เพื่อแสดงขึ้นบนเว็บไซต์

3.3 โฟลว์ชาร์ตการทำงานของ PHP และ Visual C++ ในส่วนการควบคุมการเคลื่อนที่ของรถวิทยุบังคับ



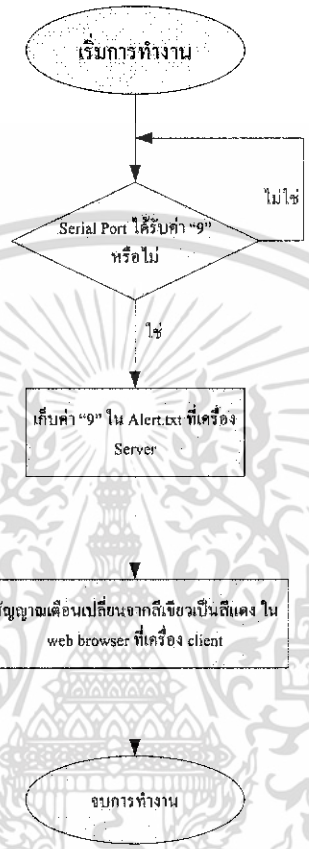
รูปที่ 3.3 โฟลว์ชาร์ตการทำงานของ PHP และ Visual C++
ในส่วนการควบคุมการเคลื่อนที่ของรถวิทยุบังคับ

หลักการทำงานของโปรแกรม

- เริ่มต้นด้วยที่ ผู้ใช้ทำการ Log in ในการควบคุมการทำงานของรถ
- ผู้ใช้ทำการกดปุ่มเพื่อควบคุมการเคลื่อนที่ของรถวิทยุบังคับ
- เมื่อผู้ใช้ทำการกดปุ่มบังคับรถวิทยุบังคับแล้ว PHP จะทำการสร้าง Text fie ขึ้นมา เพื่อให้โปรแกรม Visual C++ ทำการส่งค่าที่ผู้ใช้ทำการกดปุ่มบังคับไปส่งค่าให้พอร์ตอนุกรม
- รอการควบคุมรถวิทยุบังคับครั้งต่อไป

3.3.1 โฟลว์ชาร์ตการทำงานของ PHP และ Visual C++ ในส่วนส่งสัญญาณกลับไปที่เครื่องคอมพิวเตอร์เซิร์ฟเวอร์เมื่อเกิดควัน

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์เมื่อเกิดการตรวจจับควัน สามารถแสดงได้ดังนี้

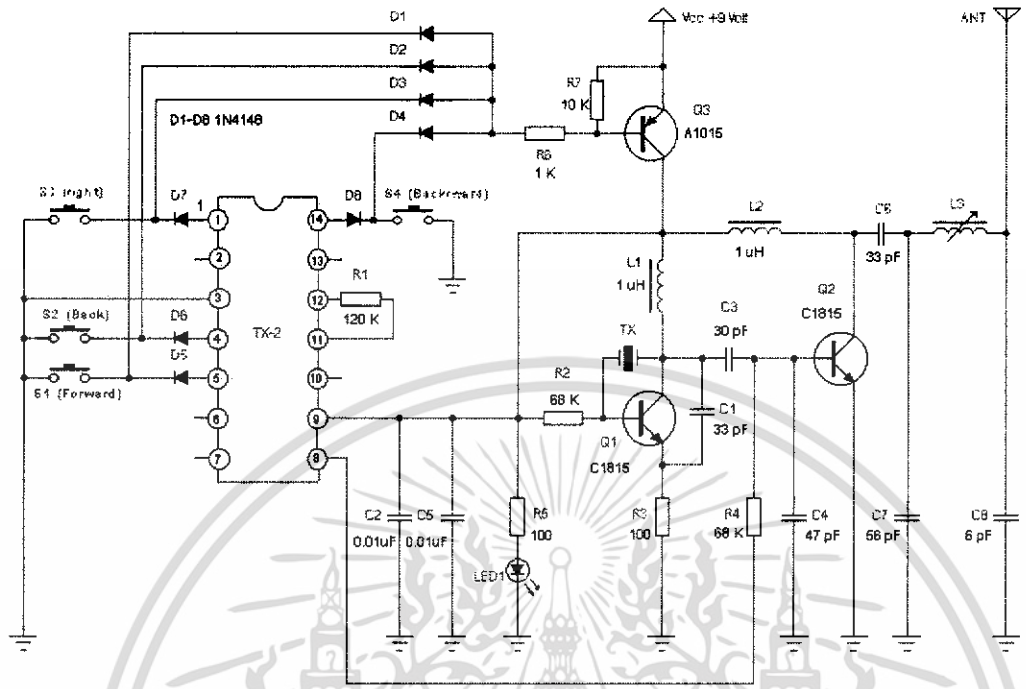


รูปที่ 3.4 โฟลว์ชาร์ตการทำงานของ PHP และ Visual C++
ในส่วนการแสดงผลการได้รับสัญญาณวิทยุจากวงจรตรวจจับควัน

หลักการทำงานของโปรแกรม

- เริ่มต้นด้วยที่ พอร์ตอนุกรมมีค่า 9 ส่งมาหรือไม่
- เมื่อมีค่า 9 ส่งมาที่พอร์ตอนุกรมโปรแกรม Visual C++ ทำการ Text file เพื่อให้ PHP นำไปแสดงผล
- รอรับค่า 9 ที่พอร์ตอนุกรมครั้งต่อไป

3.4 วงจรรีโมทคอนโทรลรถวิทยุบังคับภาคส่ง

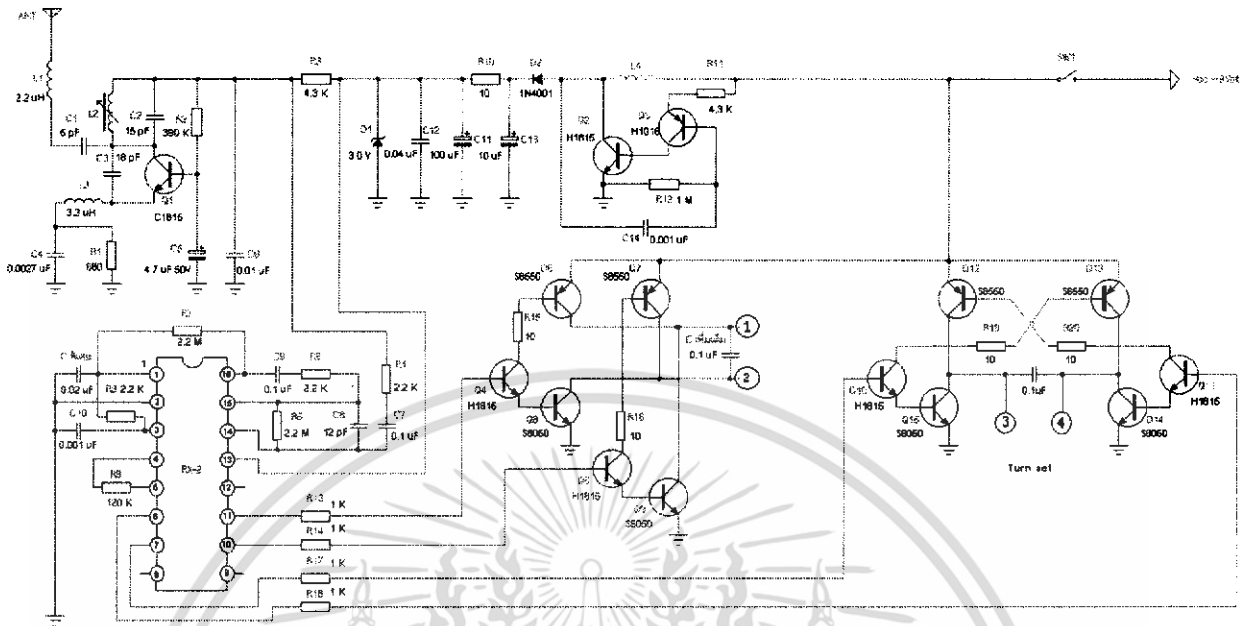


รูปที่ 3.5 วงจรรีโมทคอนโทรลรถวิทยุบังคับภาคส่ง

จะมีหลักการทำงานดังนี้

- สวิตช์ S1 ใช้สำหรับควบคุมรถให้เดินหน้า
- สวิตช์ S2 ใช้สำหรับควบคุมรถให้ถอยหลัง
- สวิตช์ S3 ใช้สำหรับควบคุมรถให้เลี้ยวขวา
- สวิตช์ S4 ใช้สำหรับควบคุมรถให้เลี้ยวซ้าย

3.5 วงจรวงจรรีโมทคอนโทรลรถวิทยุบังคับภาครับ



รูปที่ 3.6 วงจรรีโมทคอนโทรลรถวิทยุบังคับภาครับ

จะมีหลักการทำงานดังนี้

- เมื่อมีการควบคุมรถให้เดินหน้า จะทำให้ทรานซิสเตอร์ Q4, Q6 และ Q8 ทำงาน จึงทำให้เกิดค่าแรงดันที่จุดที่จุด 1

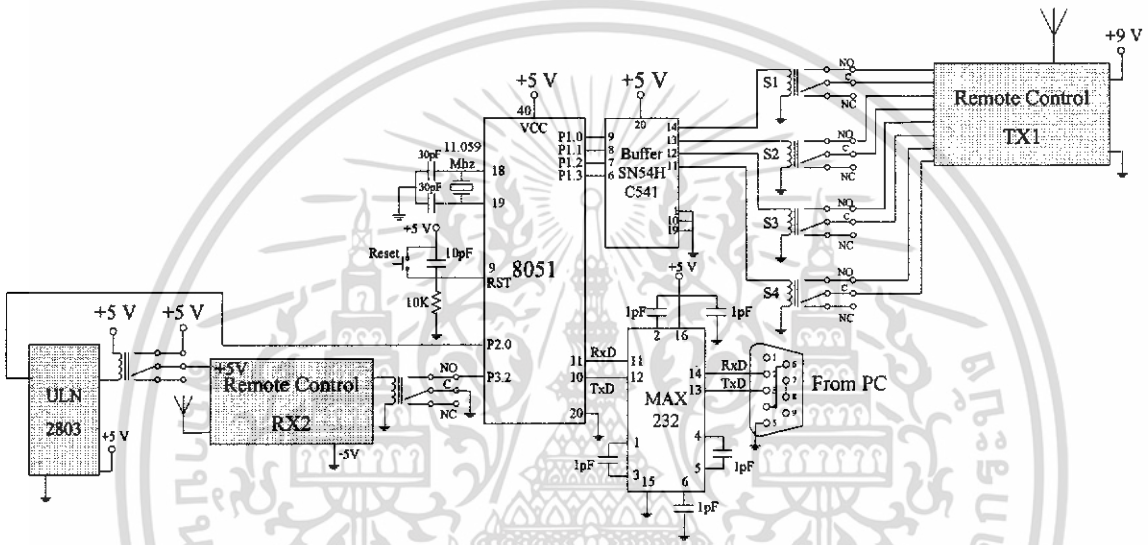
- เมื่อมีการควบคุมรถให้ถอยหลัง จะทำให้ทรานซิสเตอร์ Q5, Q7 และ Q9 ทำงาน จึงทำให้เกิดค่าแรงดันที่จุดที่จุด 2

- เมื่อมีการควบคุมรถให้เลี้ยวขวา จะทำให้ทรานซิสเตอร์ Q11, Q12 และ Q14 ทำงาน จึงทำให้เกิดค่าแรงดันที่จุดที่จุด 3

- เมื่อมีการควบคุมรถให้เลี้ยวซ้าย จะทำให้ทรานซิสเตอร์ Q10, Q13 และ Q15 ทำงาน จึงทำให้เกิดค่าแรงดันที่จุดที่จุด 4

3.6 วงจรเชื่อมต่อ MAX-232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์ MCS-51

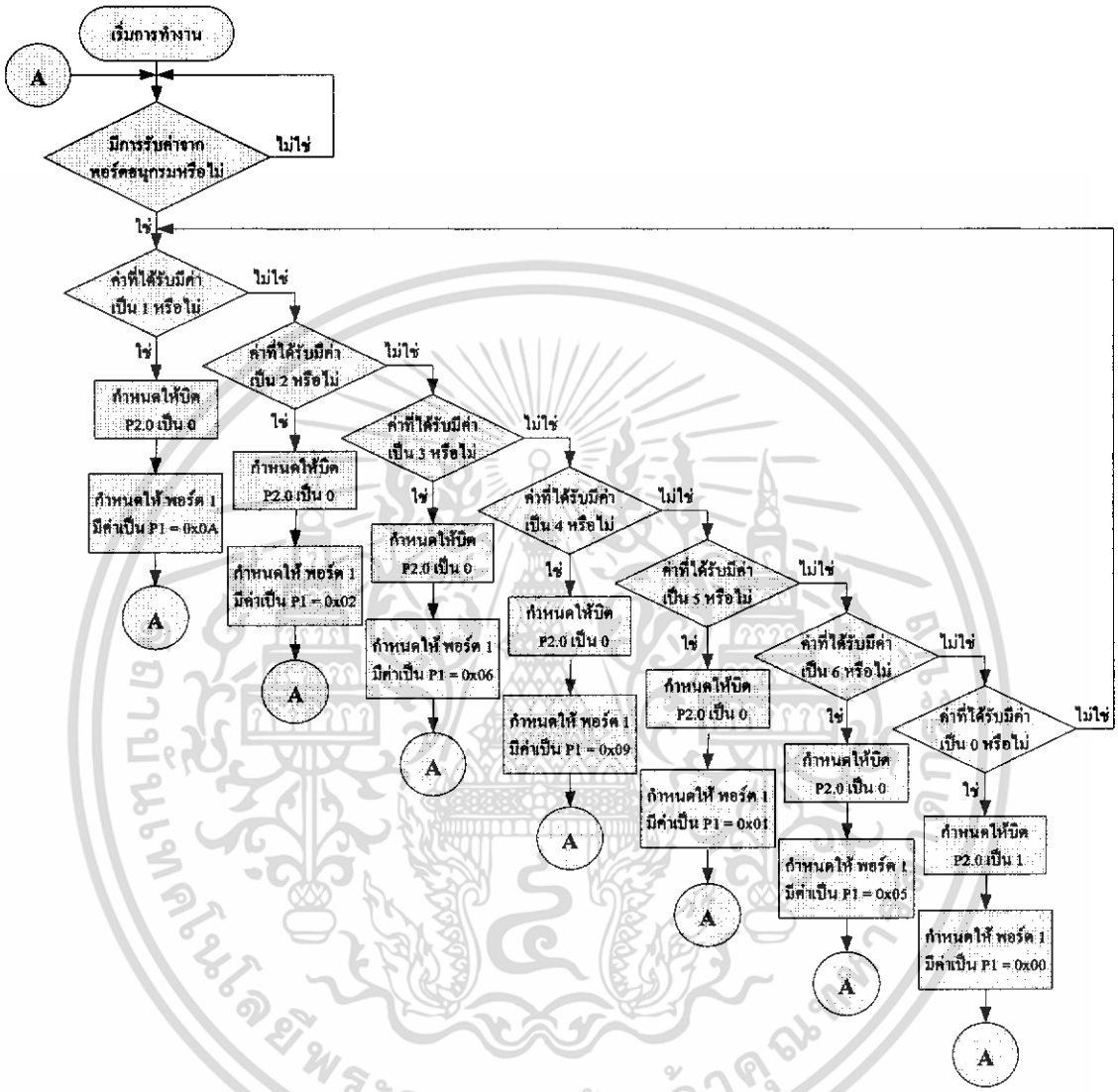
ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 ติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 จะต้องทำการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลกระดับสัญญาณ สำหรับโครงงานนี้จะเลือกไอซีหมายเลย MAX232 ซึ่งใช้ในการแปลงข้อมูลที่รับมาที่ไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลที่ส่งจากคอมพิวเตอร์จากระดับของ RS-232 ไปเป็นระดับทีทีแอลเพื่อให้สามารถทำการถ่ายทอดต่อไปยังไมโครคอนโทรลเลอร์ได้อย่างสมบูรณ์ โดยในรูปที่ 3.7 แสดงวงจรการต่อกับไมโครคอนโทรลเลอร์ MCS-51 และไมโครคอนโทรลเลอร์จะทำการควบคุมการทำงานของรีโมทต่อไป



รูปที่ 3.7 วงจรรวมของการเชื่อมต่อของอุปกรณ์ต่างๆกับไมโครคอนโทรลเลอร์และไมโครคอนโทรลเลอร์กับพอร์ตอนุกรม

3.7 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ MCS-51 ในส่วนของการควบคุมรีโมทรถบังคับ

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ในการควบคุมรีโมท สามารถแสดงได้ดังนี้



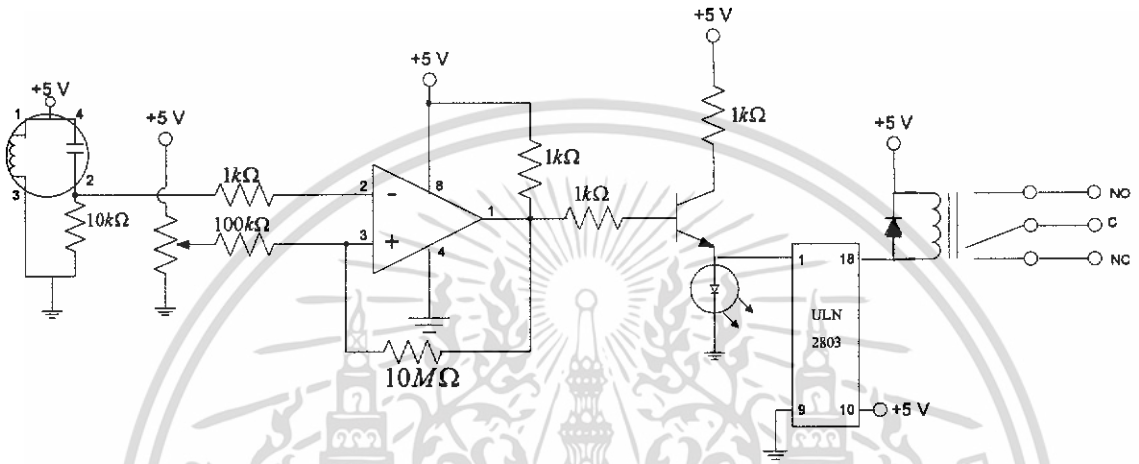
รูปที่ 3.8 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ ในการควบคุมรีโมท

หลักการทำงานของไมโครคอนโทรลเลอร์

- เริ่มต้นด้วยการรับค่าจากพอร์ตอนุกรม
- ทำการเปรียบเทียบค่าที่รับมากับค่าที่ตั้งไว้ในโปรแกรม
- ในกรณี เมื่อได้รับค่า 0 จากพอร์ตอนุกรม จะทำการเปิดวงจรรับสัญญาณวิทยุจากวงจรตรวจจับควัน โดยการ ให้บิต 2.0 เป็น 1 นอกเหนือจากกรณีนี้ จะทำการตัดวงจรรับสัญญาณวิทยุจากวงจรตรวจจับควัน โดยการ ให้บิต 2.0 เป็น 0
- ทำการเปรียบเทียบค่าทีละคำสั่ง จากนั้นจึงส่งค่าออกพอร์ต1 ของ MCS-51
- รอรับค่าที่ส่งมาใหม่

3.8 วงจรตรวจจับควัน

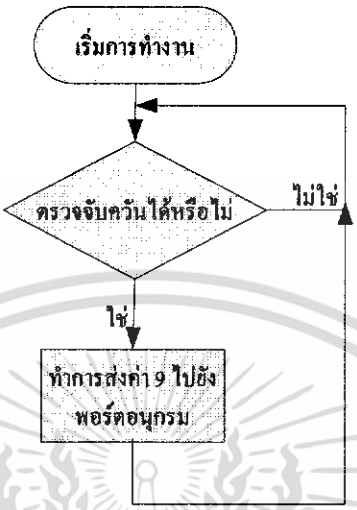
หลักการทำงานของวงจรตรวจจับควัน เมื่อเริ่มจ่ายไฟเข้าวงจรต้องรอประมาณ 2 นาที เพื่อให้อุณหภูมิหัววัดใช้งานได้ ทำการทดสอบวงจรโดยเมื่อมีควันเข้ามาที่หัวจับควันจะทำให้ค่าความต้านทานของหัววัดลดลง จะทำให้แรงดันที่ขาลบของออปแอมป์สูงขึ้น จนทำให้วงจรเปรียบเทียบค่าแรงดันมีกระแสออกที่ขาเอาท์พุต เพื่อจะนำไปเริ่มการทำงานของลีเรย์ ดังรูป



รูปที่ 3.9 วงจรตรวจจับควัน

3.9 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ในส่วนการตรวจจับควัน

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ในรับค่าการตรวจจับควัน สามารถแสดงได้ดังนี้



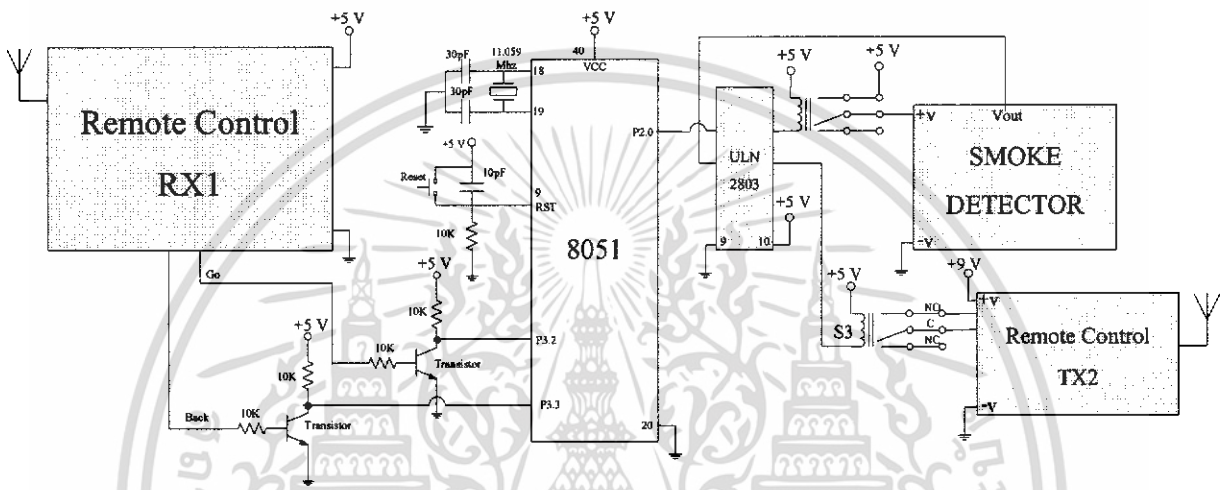
รูปที่ 3.10 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ ในการตรวจจับควัน

หลักการทำงานของไมโครคอนโทรลเลอร์

- เริ่มต้นด้วยการตรวจว่ามีการจับควันได้หรือไม่
- ทำการส่งค่า 9 ไปยังพอร์ตอนุกรม เพื่อนำไปแสดงผล
- รอรับค่าในการตรวจจับควันครั้งต่อไป

3.10 วงจรรีโมทคอนโทรลและไมโครคอนโทรลเลอร์ MCS-51 บนตัวรถวิทยุบังคับ

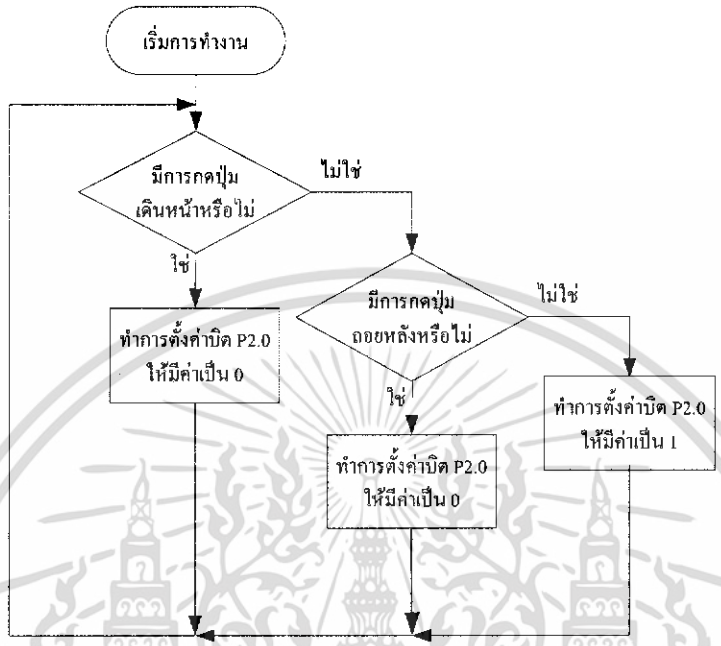
ในการใช้งานไมโครคอนโทรลเลอร์ จะทำการรับค่าในการเดินหน้าและถอยหลังจากตัวรับสัญญาณบนตัวรถวิทยุบังคับ และเมื่อไมโครคอนโทรลเลอร์ได้รับสัญญาณจะทำการควบคุมให้วงจรตรวจจับควันทำงาน เมื่อบริเวณรถวิทยุบังคับเกิดควันขึ้น ก็จะส่งสัญญาณกลับไปที่เครื่องคอมพิวเตอร์เซิร์ฟเวอร์ต่อไป และไมโครคอนโทรลเลอร์จะทำการตัดวงจรตรวจจับควันก็ต่อเมื่อไม่มีการควบคุมการเดินหน้าและถอยหลังจากตัวรับสัญญาณบนตัวรถวิทยุบังคับ



รูปที่ 3.11 วงจรรวมของการเชื่อมต่อของอุปกรณ์ต่างๆกับไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับ

3.11 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์ MCS-51 บนตัวรถวิทยุบังคับ

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับ สามารถแสดงได้ดังนี้



รูปที่ 3.12 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับ
ในส่วนการเคลื่อนที่เดินหน้าถอยหลัง

หลักการทำงานของไมโครคอนโทรลเลอร์

- เริ่มต้นด้วยตรวจสอบว่า มีการควบคุมให้รถ เดินหน้า หรือ ถอยหลังหรือไม่
- เมื่อไม่มีการควบคุมให้รถ เดินหน้า หรือ ถอยหลัง จะทำการเปิดวงจรตรวจจับควัน โดยควบคุมให้บิต 2.0 เป็น 1 นอกเหนือจากกรณีนี้ จะทำการปิดวงจรตรวจจับควันออกจากวงจรรวม โดยควบคุมให้บิต 2.0 เป็น 0
- รอรับค่าที่ส่งมาใหม่

3.12 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับในส่วนของการตรวจจับควัน

โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับในส่วนการตรวจจับควัน สามารถแสดงได้ดังนี้

รูปที่ 3.13 โฟลว์ชาร์ตการทำงานของไมโครคอนโทรลเลอร์บนตัวรถวิทยุบังคับในส่วนการตรวจจับควัน

หลักการทำงานของไมโครคอนโทรลเลอร์

- เริ่มต้นด้วยการตรวจว่ามีการจับควันได้หรือไม่
- ทำการส่งสัญญาณวิทยุกลับไปยังเครื่องคอมพิวเตอร์เซิร์ฟเวอร์
- ทำการตรวจจับควันครั้งต่อไป

บทที่ 4
การทดลองและผลการทดลอง

4.1 ผลการทดลองการควบคุมการเคลื่อนที่ของรถวิทยุบังคับด้วยชุดควบคุมภายในโปรแกรมหลัก

การควบคุมรถวิทยุบังคับ โดยเมื่อกดปุ่มควบคุมการเคลื่อนที่จากโปรแกรม โปรแกรมจะทำการส่งรหัสซึ่งทางไมโครคอนโทรลเลอร์เข้าใจออกไป โดยโปรแกรมนี้เราใช้รหัสแอสกี (ASCII) เป็นตัวกลาง โดยส่งผ่านทางพอร์ตอนุกรม RS-232 โดยปุ่มต่างๆ แทนรหัสแอสกีดังนี้

- ชุดควบคุมรถวิทยุบังคับ

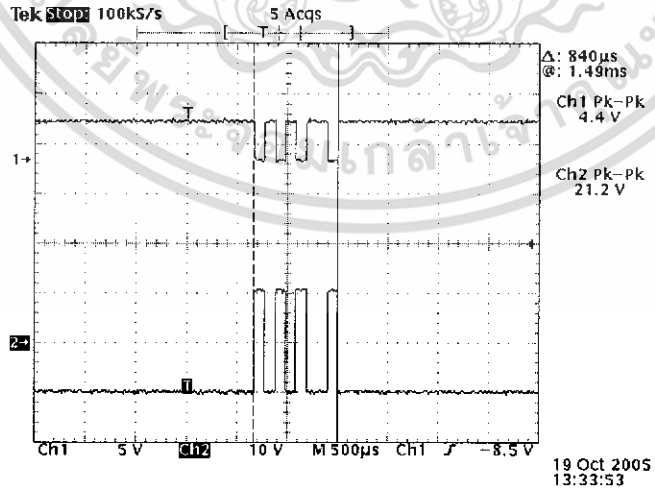
ปุ่ม เดินหน้า	ใช้รหัสแอสกี '5'	แทนด้วย 0x35
ปุ่ม ถอยหลัง	ใช้รหัสแอสกี '2'	แทนด้วย 0x32
ปุ่ม เดินหน้าและเลี้ยวขวา	ใช้รหัสแอสกี '6'	แทนด้วย 0x36
ปุ่ม เดินหน้าและเลี้ยวซ้าย	ใช้รหัสแอสกี '4'	แทนด้วย 0x34
ปุ่ม ถอยหลังและเลี้ยวขวา	ใช้รหัสแอสกี '3'	แทนด้วย 0x33
ปุ่ม ถอยหลังและเลี้ยวซ้าย	ใช้รหัสแอสกี '1'	แทนด้วย 0x31
ปุ่ม หยุด	ใช้รหัสแอสกี '0'	แทนด้วย 0x30

4.1.1 ผลการทดลองเมื่อกดปุ่มควบคุมบนโปรแกรมหลัก

ในส่วนนี้ได้ทำการวัดผลด้วยออสซิลเลเตอร์ โดยวัดที่ขา 9 ของไอซีเบอร์ MAX-232 เพื่อดูสัญญาณของรหัสแอสกีต่างๆ ที่ส่งออกมาจากคอมพิวเตอร์เพื่อควบคุมไมโครคอนโทรลเลอร์

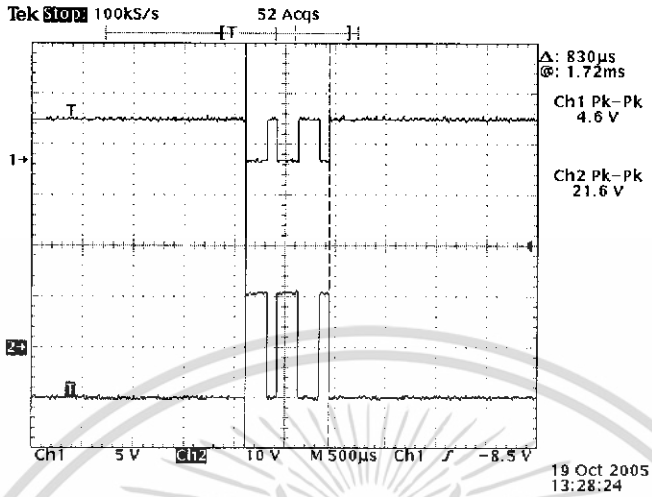
- ผลการทดลองเมื่อกดปุ่มเดินหน้า ของชุดควบคุมรถวิทยุบังคับ

สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x35 หรือ 00110101 ดังรูป



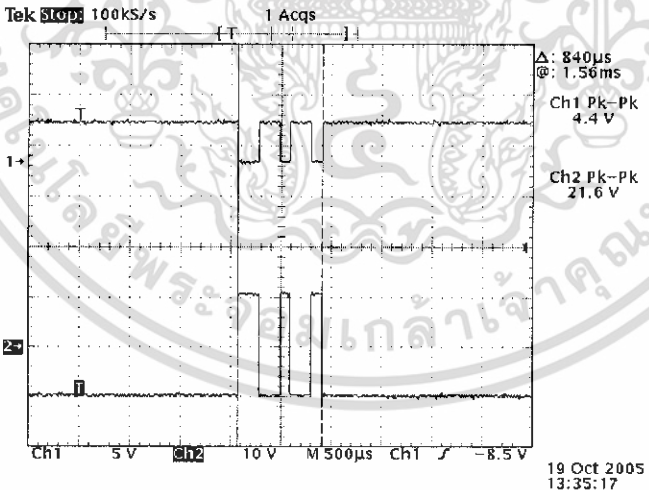
รูปที่ 4.1 สัญญาณเมื่อกดเดินหน้า ส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

- ผลการทดลองเมื่อกดปุ่มถอยหลัง ของชุดควบคุมรถวิทยุบังคับ
สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x32 หรือ 00110010 ดังรูป



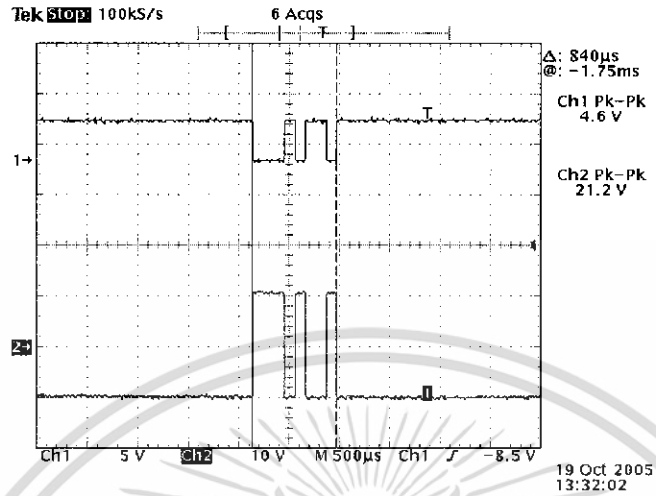
รูปที่ 4.2 สัญญาณเมื่อกดถอยหลัง ส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

- ผลการทดลองเมื่อกดปุ่มเดินหน้าและเลี้ยวขวา ของชุดควบคุมรถวิทยุบังคับ
สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x36 หรือ 00110110 ดังรูป



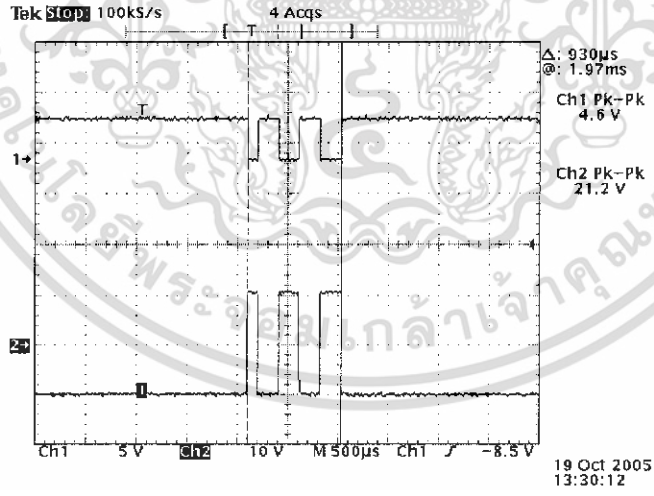
รูปที่ 4.3 สัญญาณเมื่อกดเดินหน้าและเลี้ยวขวา ส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

- ผลการทดลองเมื่อกดปุ่มเดินหน้าและเลี้ยวซ้าย ของชุดควบคุมรถวิทยุบังคับ
สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x34 หรือ 00110100 ดังรูป



รูปที่ 4.4 สัญญาณเมื่อกดเดินหน้าและเลี้ยวซ้าย ส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

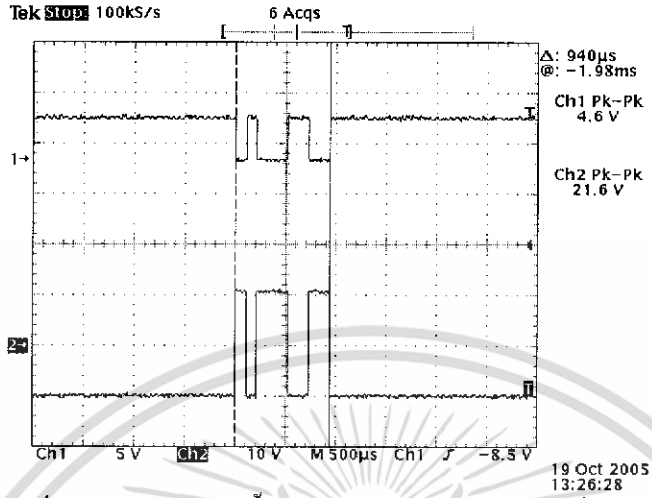
- ผลการทดลองเมื่อกดปุ่มถอยหลังและเลี้ยวขวา ของชุดควบคุมรถวิทยุบังคับ
สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x33 หรือ 00110011 ดังรูป



รูปที่ 4.5 สัญญาณเมื่อกดถอยหลังและเลี้ยวขวาส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

- ผลการทดลองเมื่อกดปุ่มถอยหลังและเลี้ยวซ้าย ของชุดควบคุมรถวิทยุบังคับ

สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x31 หรือ 00110001 ดังรูป

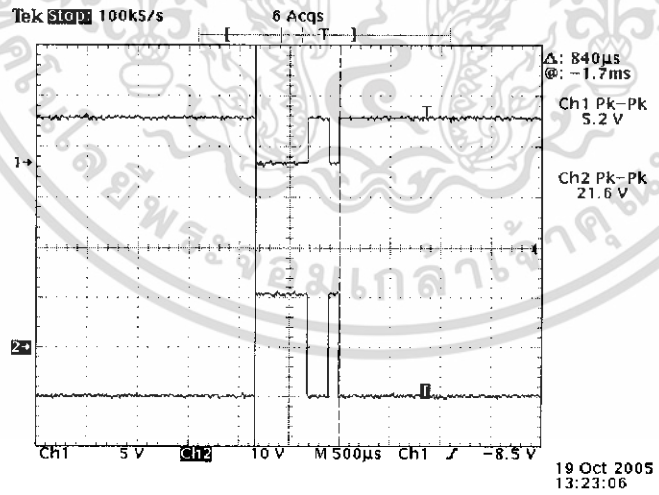


รูปที่ 4.6 สัญญาณเมื่อกดถอยหลังและเลี้ยวซ้ายส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

- ผลการทดลองเมื่อกดปุ่มหยุด ของชุดควบคุมรถวิทยุบังคับ

ผลที่ได้จะทำให้ไมโครคอนโทรลเลอร์สั่งงานให้วงจรตรวจจับควันเริ่มทำงานได้

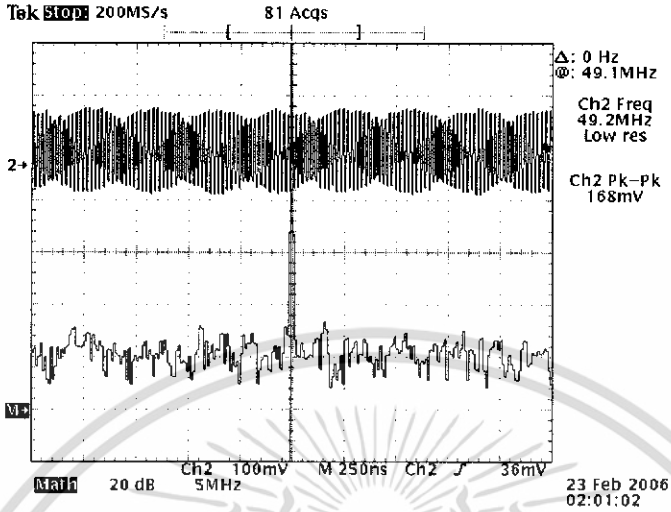
สัญญาณที่วัดได้จะเป็นสัญญาณของรหัส 0x30 หรือ 00110000 ดังรูป



รูปที่ 4.7 สัญญาณเมื่อกดหยุด ส่วนบนแสดงสัญญาณที่ออกมาจาก MAX-232
ส่วนล่างแสดงสัญญาณที่ออกมาจากพอร์ตอนุกรม

4.1.2 ผลการทดลองของวงจรส่งคลื่นบังคับรถวิทยุ

เมื่อรับสัญญาณจากพอร์ตอนุกรม ไมโครคอนโทรลเลอร์จะควบคุมรีเลย์ให้วงจรส่งสัญญาณควบคุมรถบังคับดังรูปที่ 4.8



รูปที่ 4.8 สัญญาณที่ส่งออกมาจากวงจรส่งบังคับรถวิทยุความถี่ 49 MHz

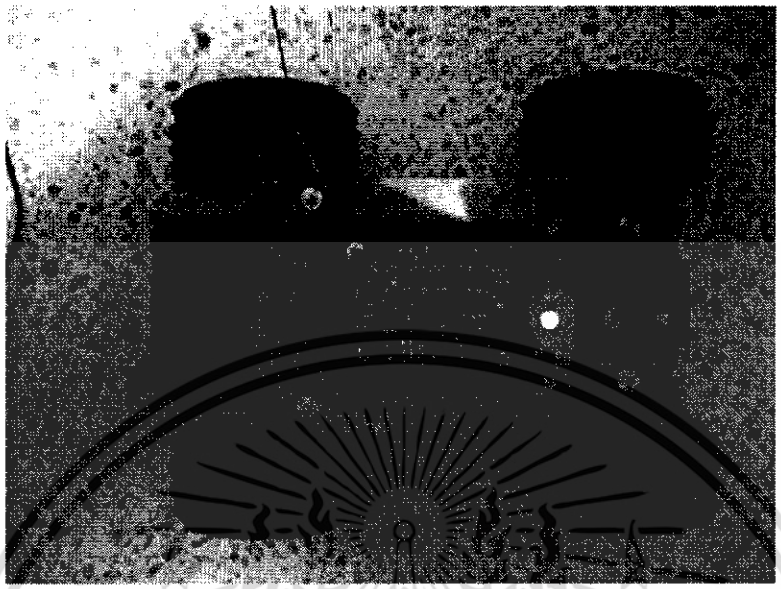
4.2 ผลการทดลองการเคลื่อนที่ของรถวิทยุบังคับ

เมื่อทำการสัญญาณของรหัสแอสกีต่างๆ ที่ส่งออกมาจากคอมพิวเตอร์เพื่อควบคุมไมโครคอนโทรลเลอร์เพื่อบังคับการเคลื่อนที่ของรถวิทยุบังคับโดยการเคลื่อนที่ของรถวิทยุบังคับจะแสดงผลได้จากหลอด LED

- ผลการทดลองเมื่อกดปุ่มเดินหน้า

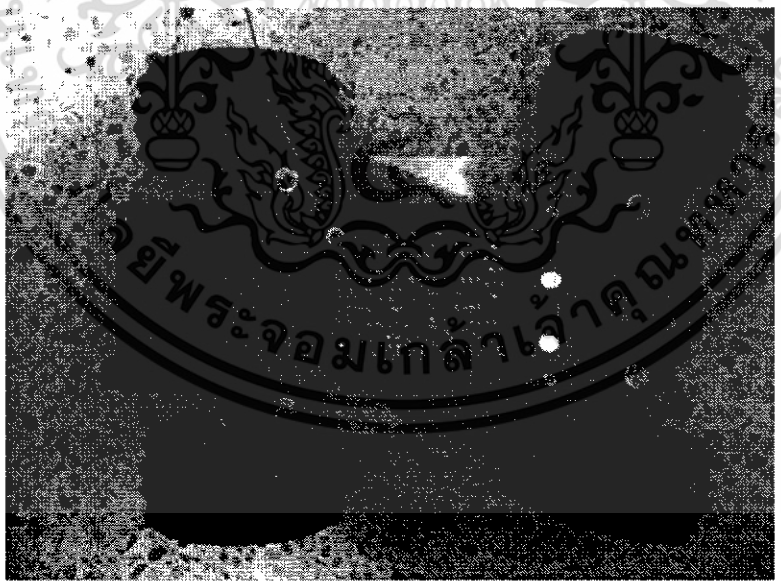
รูปที่ 4.9 ลักษณะหลอด LED เมื่อกดเดินหน้า

- ผลการทดลองเมื่อกดปุ่มถอยหลัง



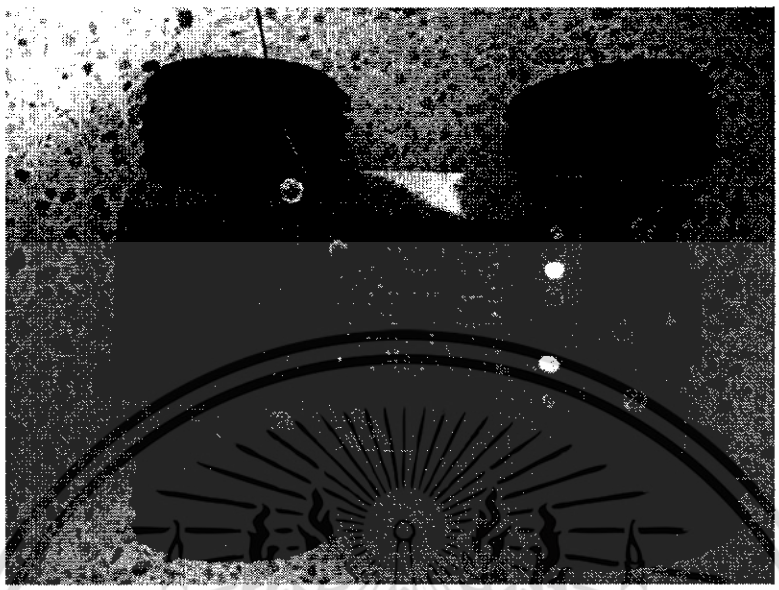
รูปที่ 4.10 ลักษณะหลอด LED เมื่อกดถอยหลัง

- ผลการทดลองเมื่อกดปุ่มเดินหน้าและเลี้ยวขวา



รูปที่ 4.11 ลักษณะหลอด LED เมื่อกดเดินหน้าและเลี้ยวขวา

- ผลการทดลองเมื่อกดปุ่มเดินหน้าและเลี้ยวซ้าย

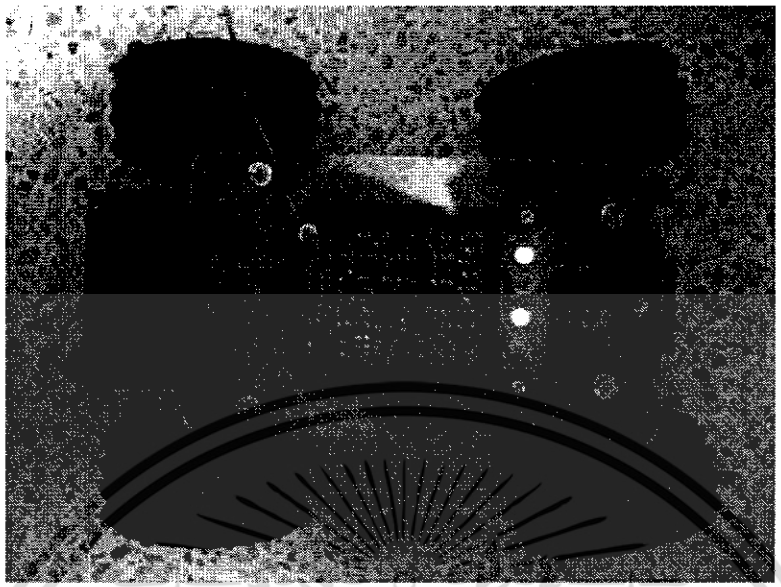


รูปที่ 4.12 ลักษณะหลอด LED เมื่อกดเดินหน้าและเลี้ยวซ้าย

- ผลการทดลองเมื่อกดปุ่มถอยหลังและเลี้ยวขวา

รูปที่ 4.13 ลักษณะหลอด LED เมื่อกดถอยหลังและเลี้ยวขวา

- ผลการทดลองเมื่อกดปุ่มถอยหลังและเลี้ยวซ้าย



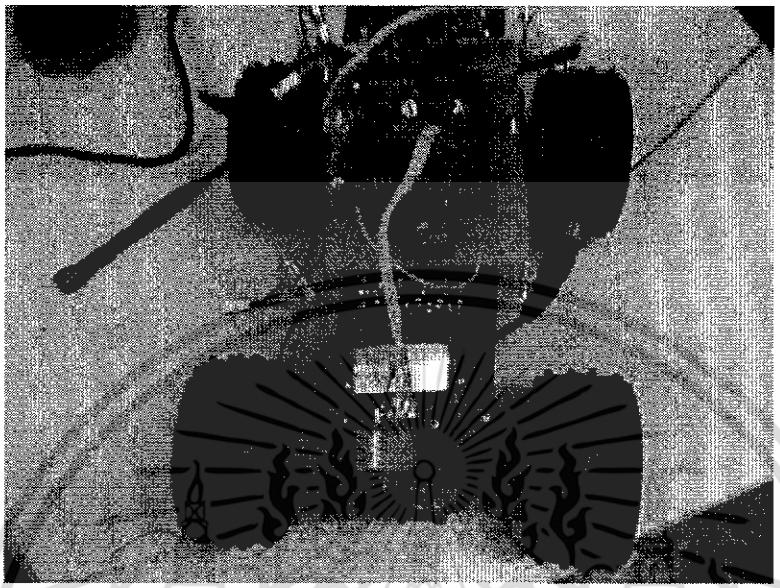
รูปที่ 4.14 ลักษณะหลอด LED เมื่อกดถอยหลังและเลี้ยวซ้าย

- ผลการทดลองเมื่อกดปุ่มหยุด

รูปที่ 4.15 ลักษณะหลอด LED เมื่อกดหยุด

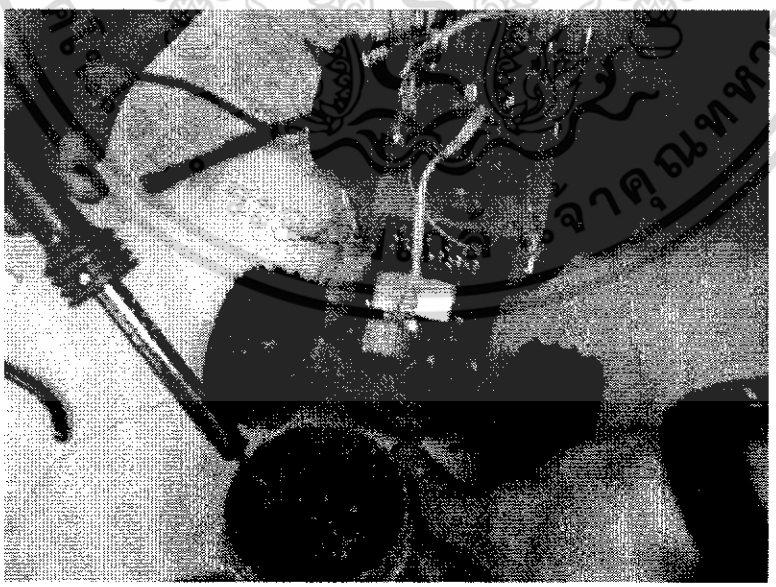
4.3 ผลการทดลองส่วนของวงจรตรวจจับควัน

เมื่อรถได้รับสัญญาณหยุดรถจะส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ให้วงจรตรวจจับควันเริ่มทำงานดังรูป



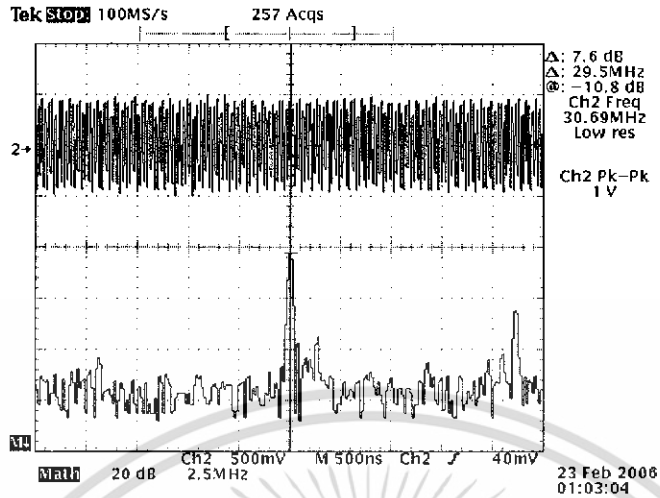
รูปที่ 4.16 แสดงสถานะพร้อมที่จะทำงานของวงจรตรวจจับควัน

เมื่อมีควันเข้ามาทางหัวตรวจจับควันวงจรตรวจจับควันจะแสดงด้วยหลอดLED



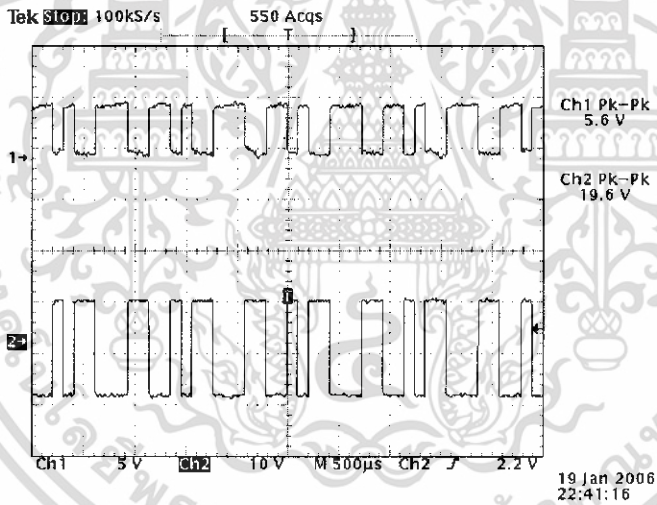
รูปที่ 4.17 แสดงการทำงานของวงจรเมื่อได้รับควัน

- วงจรตรวจจับควันจะส่งสัญญาณไปยังวงจรส่งคลื่นวิทยุเพื่อส่งสัญญาณกลับไปยังวงจรรับคลื่นวิทยุที่เครื่องคอมพิวเตอร์เซิร์ฟเวอร์



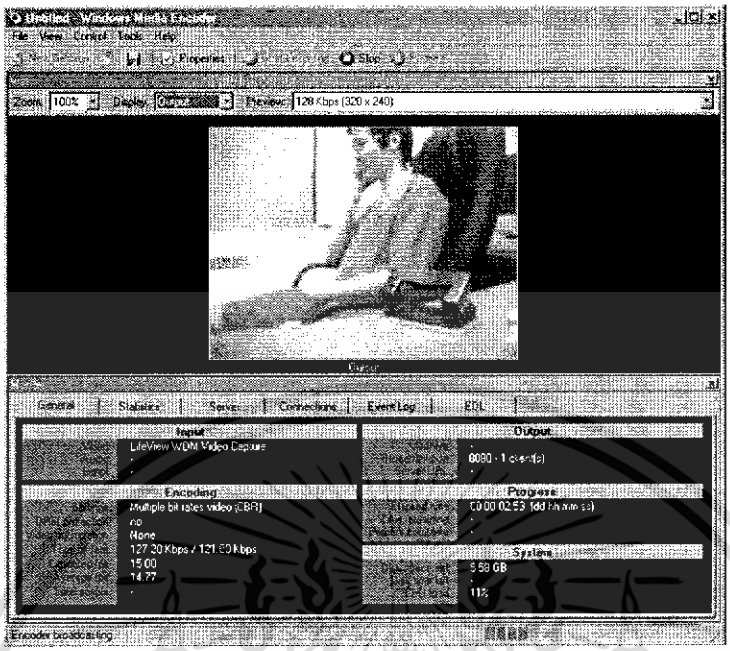
รูปที่ 4.18 สัญญาณที่ส่งออกมาจากวงจรส่งเตือนควันความถี่ 29 MHz

- จากนั้นจะส่งสัญญาณเลข "9" ผ่านพอร์ตอนุกรมดังรูป

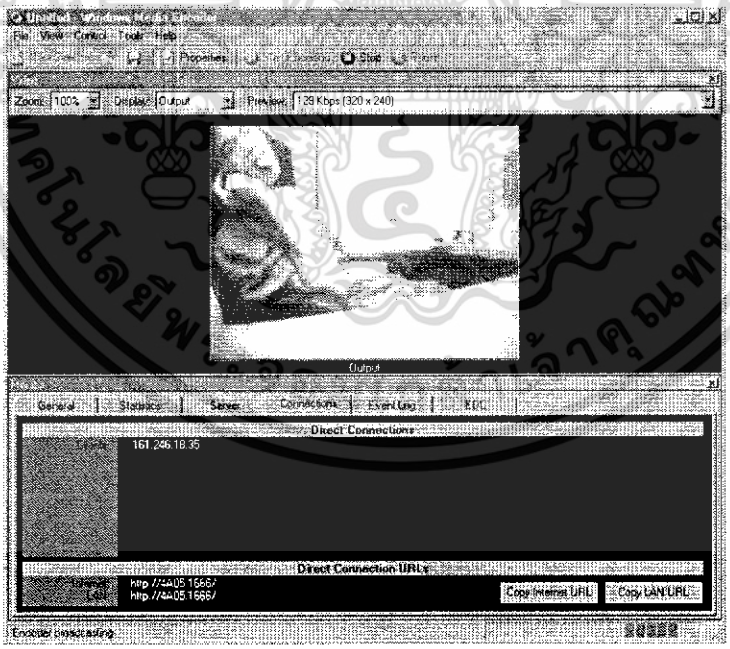


รูปที่ 4.19 สัญญาณเลข "9" ที่ส่งผ่านพอร์ตอนุกรมเข้าคอมพิวเตอร์

4.4 ผลการทดลองโปรแกรม Windows Media Encoder



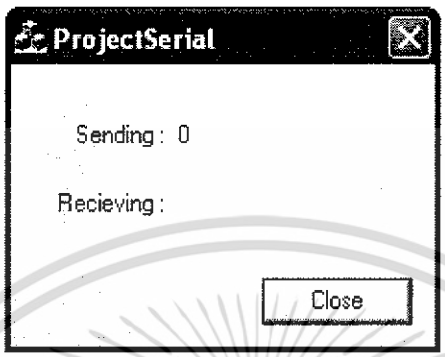
รูปที่ 4.20 หน้าต่างโปรแกรม Window Media Encoder ทางเครื่อง Server
(บอกจำนวนของเครื่อง Client ที่เรียกใช้)



รูปที่ 4.21 หน้าต่างโปรแกรม Window Media Encoder ทางเครื่อง Server
(บอก IP Address ของเครื่อง Client ที่เรียกใช้)

4.5 ผลการทดลองโปรแกรม Visual C++

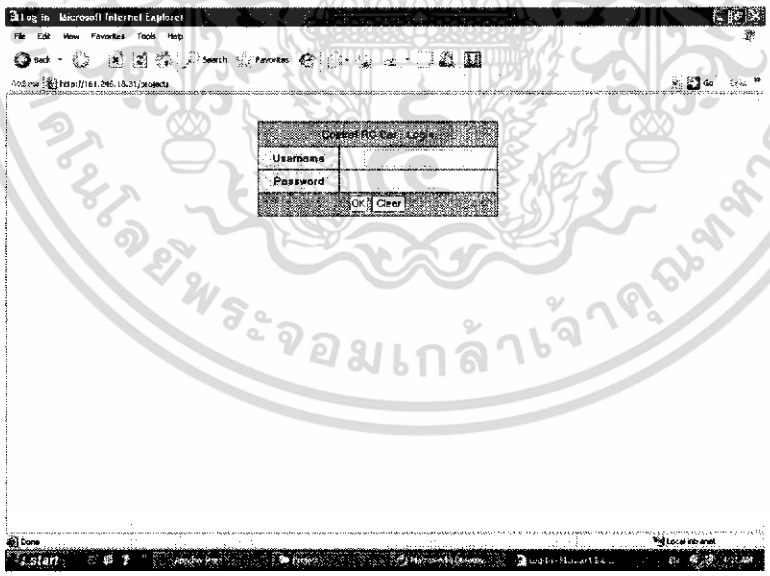
เมื่อค่าการควบคุมรถ ถูกเก็บเข้ามาที่ Status.txt ที่เครื่อง Server แล้ว โปรแกรม Visual C++ ซึ่งหน้าต่างดังรูป ก็จะนำค่าใน Status.txt ส่งออกทาง Serial Port



รูปที่ 4.22 หน้าต่างโปรแกรม Visual C++

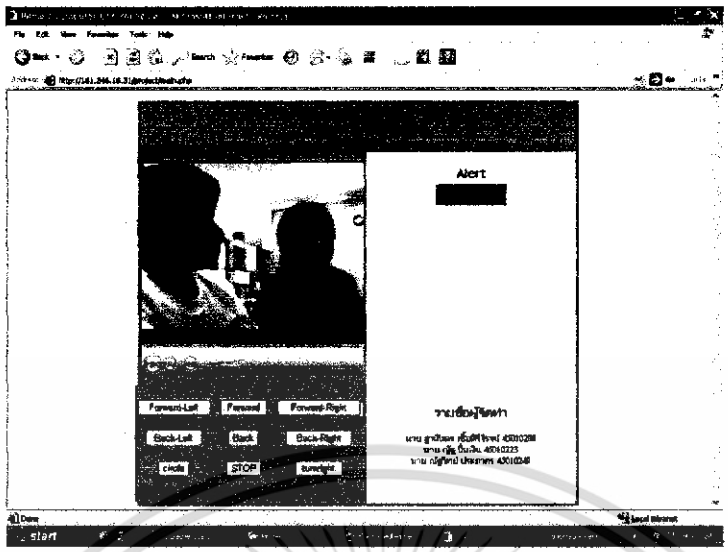
4.6 ผลการทดลองที่เครื่อง Client

เมื่อเครื่อง Client เรียก IP address ของ Server จะแสดง Web Browser ดังรูป



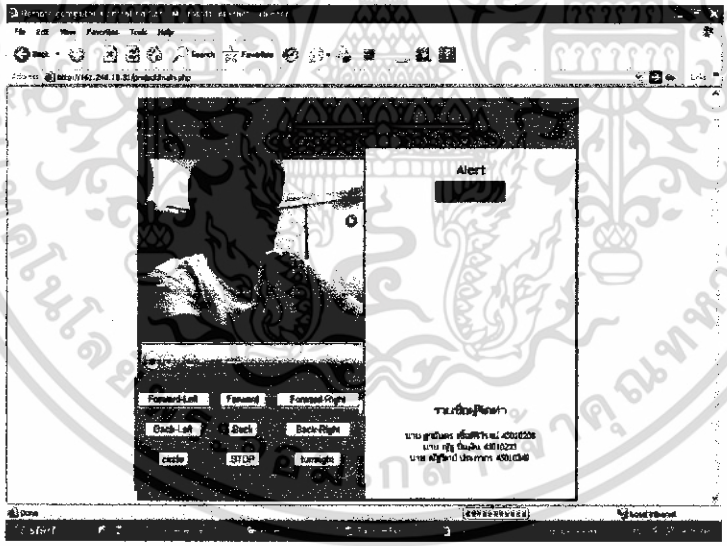
รูปที่ 4.23 หน้า Log In เข้าสู่ระบบ

เมื่อเข้าสู่ระบบ และรถอยู่ในสภาวะปกติ (ไม่สามารถตรวจจับควันได้) หน้า Web Browser ทางเครื่อง Client จะแสดงดังรูป



รูปที่ 4.24 Web Browser ที่เครื่อง Client เมื่อค่าใน alert.txt มีค่าที่ไม่เป็น “9”

เมื่อรถตรวจจับควันได้ จะส่งค่า “9” กลับมาที่ Alert.txt ที่เครื่อง Server ทำให้ หน้า Web Browser ทางเครื่อง Client แสดง ดังรูป



รูปที่ 4.25 Web Browser ที่เครื่อง Client เมื่อค่าใน Alert.txt มีค่าที่เป็น “9”

- ทดสอบแบนวิทด์ที่ใช้ที่เครื่อง Client จากรูปที่ 4.26

รูปที่ 4.26 รูปแสดงการแบนวิทด์ที่เครื่องฝั่ง Client

คำนวณค่าการส่งผ่านข้อมูล

0.14 % จากความเร็ว 100 Mbps

คิดเป็นแบนวิดท์ที่ใช้คือ 140 kbps

บทที่ 5
บทวิจารณ์และสรุปผล

5.1 สรุปผลการทดลอง

5.1.1 การทำงานในส่วนของการรับส่งข้อมูลผ่านอินเตอร์เน็ต

- ผู้ใช้สามารถเข้าไปสั่งงานรถผ่านหน้าเว็บไซต์โดยผ่านหน้า log in จากนั้นก็จะสามารถกดปุ่มทิศทางที่จะไปใช้บังคับรถ และสามารถแสดงสัญญาณเตือนเมื่อได้รับสัญญาณจากวงจรตรวจจับควันได้
- สามารถส่งภาพจากกล้องที่ติดอยู่บนรถ แสดงออกทางเว็บไซต์ได้
- สามารถเก็บค่าการบังคับรถไว้ใน Text file ที่สร้างขึ้นมาเพื่อจะส่งออกทางพอร์ตอนุกรมได้

5.1.2 การทำงานของการรับส่งข้อมูลผ่านพอร์ตอนุกรม

- สามารถเรียกค่าจาก Text file ที่เก็บไว้ และทำการส่งออกทางพอร์ตอนุกรมได้
- สามารถรับค่าจากไมโครคอนโทรลเลอร์ และทำการเก็บข้อมูลลงใน Text file ได้

5.1.3 การทำงานระหว่างรถกับไมโครคอนโทรลเลอร์

- สามารถสั่งงานควบคุมทิศทางการเคลื่อนที่ของรถได้
- วงจรตรวจจับควันสามารถส่งสัญญาณกลับไปบอกยังไมโครคอนโทรลเลอร์ที่ต่ออยู่กับพอร์ตอนุกรม เมื่อเกิดการตรวจพบควันได้

หนังสืออ้างอิง

ยุทธนา ลีลาศวัฒนกุล. **เริ่มต้นการเขียนโปรแกรมด้วยภาษา C++** กรุงเทพฯ : บริษัท ดวงกมลสมัย จำกัด , พิมพ์ครั้งที่ 2 , 2547.

สมยศ จุฌณะปิยะ. **การประยุกต์ใช้ไมโครคอนโทรลเลอร์** กรุงเทพฯ : คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2543.

นคร ภักดีชาติ. **ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษา C** กรุงเทพฯ : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด , 2521

กิตติ ภักดีวัฒนะกุล – อังศุมาลิน เวชนารายณ์. **คัมภีร์PHP** กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์ พิมพ์ครั้งที่ 3 , 2546

ภาคผนวก

Source code C programming for Microcontroller

Server

```
#include <AT89X51.H>
#include "serialport_h.h"
#define SENSOR_PIN P3_2
void intSerial(void) interrupt 4
{
      unsigned char dat;
      RI = 0;
      dat = SBUF;
      switch(dat)
            {
                  case '0':   P1 = 0x00;  break;
                  case '1':   P1 = 0x0A;  break;
                  case '2':   P1 = 0x02;  break;
                  case '3':   P1 = 0x06;  break;
                  case '4':   P1 = 0x09;  break;
                  case '5':   P1 = 0x01;  break;
                  case '6':   P1 = 0x05;  break;
            }// end switch
}
void main(void)
{
      unsigned char sensorBuff;
      serial_init();
      INT0 = 1;
      IE = 0x90;
      SENSOR_PIN = sensorBuff = 1;
      while(1)
      {
            if ((SENSOR_PIN == 0) && (sensorBuff = 1))
            {
                  sndSerial('9');
                  sensorBuff = SENSOR_PIN;
            }
            else if ((SENSOR_PIN == 1) && (sensorBuff = 0))
            {
                  sndSerial('8');
                  sensorBuff = SENSOR_PIN;
            }
      }//end while
}
#include <AT89X52.H>
#include "serialport_h.h"
void serial_init(void)
{
      PCON = 0x00;
      SCON = 0x50;
      TMOD = 0x20;
      TH1 = 0xFD;
      TR1 = 1;
}
void sndSerial(unsigned char dat)
{
      SBUF = dat;
      while(~TI);
      TI = 0;
}
unsigned char recSerial(void)
{
```

```
        unsigned char dat;
        while(~RI);
        RI = 0;
        dat = SBUF;
        return(dat);
}
```

Car

```
#include <AT89X51.H>
#define sw_go P3_2
#define sw_back P3_3
#define smoke_on P2_0
void main(void)
{
        smoke_on = 0;
        sw_go = 1;
        sw_back = 1;
        while(1)
        {
                while(~sw_go)
                {
                        smoke_on = 0;
                }
                while(~sw_back)
                {
                        smoke_on = 0;
                }
                smoke_on = 1;
        }
}
```

PHP Programming

```
<?php
  $host = "localhost";
  $username = "root";
  $passwd = "";
  $db1 = "rccar";
?>
```

Log In

```
<html>
<head>
<title>Log-In</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
874">
<link href="../css/dance-inner.css" rel="stylesheet" type="text/css">
</head>
<body background="page_bg.gif">
<table width="95%"  border="0" align="center" cellpadding="0"
cellspacing="0">
</table>
<div align="center"></div>
<table width="95%" height="191"  border="0" align="center"
cellpadding="0" cellspacing="0">
  <tr>
```

```
    <td height="191"><form name="form1" method="post"
action="check.php">
        <table width="35%" height="129"  border="1" align="center"
cellpadding="0" cellspacing="0" bordercolor="#666666">
          <tr bgcolor="#FF9900">
            <td height="27" colspan="3"><div align="center"
class="header"><font size="2" face="MS Sans Serif"><strong>Control RC
Car - Login</strong></font></div></td>
          </tr>
          <tr>
            <td width="34%" bgcolor="#FFDFB0"
class="sidebarFooter"><div align="center"><strong><font size="2"
face="MS Sans Serif">Username</font></strong></div></td>
            <td width="66%" colspan="2"><div align="center"><font
size="2"><font face="MS Sans Serif">
                <input name="uname" type="text" class="box1"
id="uname" maxlength="10">
            </font></font></div></td>
          </tr>
          <tr>
            <td bgcolor="#FFDFB0" class="sidebarFooter"><div
align="center"><strong><font size="2" face="MS Sans
Serif">Password</font></strong></div></td>
            <td colspan="2"><div align="center"><font size="2"><font
face="MS Sans Serif">
                <input name="pwd" type="password" class="box1"
id="pwd">
            </font></font></div></td>
          </tr>
          <tr bgcolor="#FF9900">
            <td colspan="3"><div align="center"><font size="2"><font
face="MS Sans Serif">
                <input type="submit" name="Submit" value="OK">
                </font></font><font size="2"><font face="MS Sans
Serif">
                <input type="reset" name="Submit2" value="Clear">
            </font></font></div></td>
          </tr>
        </table>
    </form></td>
  </tr>
</table>
</body>
</html>
```

Log Out

```
<?php
      session_start();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>logout</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
874">
<link href="../css/dance-inner.css" rel="stylesheet" type="text/css">
<script language="JavaScript">
    parent.bottomFrame.location = "../menu.php"
</script>
</head>
<body background="page_bg.gif">
<?php
      $old_user = $valid_user;
      session_unregister("privilege");
      session_unregister("ads_id");
      $result = session_unregister("valid_user");
      session_destroy();
      if (isset($old_user)){
            if($result){
                  $filename = 'status.txt';
                  $somecontent = 0;
//if (is_writable($filename)) {
    if (!$handle = fopen($filename, 'w')) {
         echo "Cannot open file ($filename)";
         exit;
    }
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Cannot write to file ($filename)";
        exit;
    }
    fclose($handle);
                  echo "<p align=\"center\"><font size=\"2\"><strong>
ออกระบบอย่างสมบูรณ์</strong><br><p align=\"center\"><font
size=\"2\"><strong><a href=\"index.php\">กลับสู่หน้าแรก</a></strong>";
            }else {
?>
<p align="center"><font size="2"><strong>ไม่สามารถออกจากระบบได้
</strong><br>
    <a href="logout.php">ออกจากระบบอีกครั้ง</a>  <br>
  <br>
  <?php
      }
}else echo " <p align=\"center\"><font size=\"2\"><strong><a
href=\"index.php\">กรุณา Login ก่อน</a></strong><br>";
?>
</font></p>
</body>
</html>
```

Check User

```
<?php
      session_start();
      $uname = $HTTP_POST_VARS['uname'];
      $pwd = $HTTP_POST_VARS['pwd'];
      $pwd = addslashes($pwd);
?>
<html>
<head>
<title>Check</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
874">
<link href="../css/dance-inner.css" rel="stylesheet" type="text/css">
</head>

<body background="../images/page_bg.gif">
<?php
      if ($uname && $pwd){
      $pwd_md5 = md5($pwd);
       include 'hup.php';
      $db_conn = mysql_connect($host,$username,$passwd);
      mysql_select_db($db1,$db_conn);

            /*$queryX="SELECT `privilege`,`ads_id` FROM `ads_tb`
WHERE `username`='$uname' and `password`='$pwd_md5';";
            $resultX=mysql_query($queryX);
            $rowX= mysql_fetch_row($resultX);
            $privilege = $rowX[0];
            $ads_id = $rowX[1];*/
            $query = "SELECT * FROM `login` WHERE `username`='$uname'
and `password`='$pwd_md5'";
      $result = mysql_query($query,$db_conn);
      $rows = mysql_num_rows($result);
      if($rows <= 0){
?>
<div align="center">
  <table width="30%" height="55"  border="0" cellpadding="0"
cellspacing="0">
    <tr>
      <td><div align="center" class="header"><font size="2">รหัสผ่านไม่
ถูกต้อง</font></div></td>
    </tr>
    <tr>
      <td><div align="center"><a href="index.php"><font
size="2">Login Again </font></a></div></td>
    </tr>
  </table>
<?php
      }
      else{
            $valid_user = $uname;
            session_register("valid_user");
            session_register("privilege");
            session_register("ads_id");
?>
<br>
<br>
<?php
```

```
      header("Location: main.php");
?>
<center><br>ยินดีต้อนรับคุณ <?php echo $uname; ?><br><a
href="main.php"><font size="2">ไปยังหน้าแรก ...</font></a></center>
<br>
  <?php
            }
      }
      else{
?>
  <br>
  <table width="30%" height="55"  border="0" cellpadding="0"
cellspacing="0">
    <tr>
      <td><div align="center" class="header"><font size="2">ข้อมูลไม่ครบ
</font></div></td>
    </tr>
    <tr>
      <td><div align="center"><a href="index.php"></a><a
href="index.php"><font size="2">Login Again </font></a></div></td>
    </tr>
  </table>
  <p><br>
    <?php
      }
?>
</p>
  <p> </p>
</div>
</body>
</html>
```

Control

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>

<body bgcolor="#FF9900" leftmargin="0" topmargin="0">
<br>
<table width="100%" height="129"  border="0" cellpadding="0"
cellspacing="0">
  <tr valign="middle" bgcolor="#FF9900">
    <td width="31%"><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="f1" type="submit" id="f12" value="Forward-
Left">
          </div>
        </form>
    </div></td>
    <td width="36%"><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="f" type="submit" id="f" value="Forward">
```

```
          </div>
        </form>
    </div></td>
    <td width="33%"><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="fr" type="submit" id="fr" value="Forward-
Right">
          </div>
        </form>
    </div></td>
  </tr>
  <tr valign="middle" bgcolor="#FF9900">
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="bl" type="submit" id="bl" value="Back-Left">
          </div>
        </form>
    </div></td>
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="b" type="submit" id="b" value="Back">
          </div>
        </form>
    </div></td>
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="br" type="submit" id="br" value="Back-
Right">
          </div>
        </form>
    </div></td>
  </tr>
  <tr valign="middle" bgcolor="#FF9900">
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="circle" type="submit" id="circle"
value="circle">
          </div>
        </form>
    </div></td>
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="stop" type="submit" id="stop" value="STOP">
          </div>
        </form>
    </div></td>
    <td><div align="center">
        <form name="form1" method="post" action="process.php">
          <div align="center">
            <input name="turnright" type="submit" id="turnright"
value="turnright">
          </div>
        </form>
    </div></td>
  </tr>
```

```
</table>
</body>
</html>
```

Main

```
<?php
      session_start();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Remote computer-Controlled Car</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
874">
</head>
<body background="page_bg.gif">
<?php
      if (session_is_registered("valid_user")){
?>
<table width="526" height="514" border="3" align="center"
cellpadding="0" cellspacing="0" bordercolor="#666666">
  <tr bgcolor="#FF9900">
    <td height="67" colspan="2"><div align="center">
      <p><font color="#000000" size="3" face="Verdana, Arial,
Helvetica, sans-serif"><strong><font size="4">Control RC Car
</font><br>
            </strong><font size="2">-- control by <strong><?php echo
$valid_user?></strong> -- <br>
            <strong><a href="logout.php">logout here</a></strong><br>
        </font></font></p>
    </div></td>
  </tr>
  <tr>
    <td width="320" height="19" bgcolor="#FF9900">
      <OBJECT id=MediaPlayer
codeBase=http://activex.microsoft.com/activex/controls/mplayer/en/nsm
p2inf.cab#Version=5,1,52,701
type=application/x-oleobject height=305
standby="Loading Microsoft Windows Media Player components..."
width=320
classid=CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6 VIEWASTEXT>
      <PARAM NAME="url" VALUE="http://161.246.18.31:1710/">
      <PARAM NAME="uimode" VALUE="mini">
      <PARAM NAME="stretchToFit" VALUE="true">
      <PARAM NAME="AutoStart" VALUE="-1">
    </OBJECT></td>
    <td width="196" rowspan="2" valign="top"><div align="center">
      <p>
        <iframe src="alert.php" frameborder="0" width="300"
height=500></iframe>
      </p>
      </div></td>
  </tr>
  <tr>
    <td height="134" valign="top" bgcolor="#FF9900"><iframe
src="control.html" frameborder="0" width="100%">
      <div align="left"><br>
      </div></td>
  </tr>
```

```
</table>
<?php
}
else
?>
<div align="center"><a href="index.php"><font size="2">กรุณาLogin ก่อน
</font></a>
  <?
;
?>
</div>
</body>
</html>
```

Process

```
<?php
if ($circle) {
      echo "Circle";
      $filename = 'status.txt';
      if (!$handle = fopen($filename, 'w')) {
            echo "Cannot open file ($filename)";
            exit;
       }
      if (fwrite($handle, "6") === FALSE) {
           echo "Cannot write to file ($filename)";
           exit;
      }
      fclose($handle);

      sleep(2);
      $filename = 'status.txt';
      if (!$handle = fopen($filename, 'w')) {
            echo "Cannot open file ($filename)";
            exit;
       }
      if (fwrite($handle, "0") === FALSE) {
           echo "Cannot write to file ($filename)";
           exit;
      }
      fclose($handle);
      sleep(1);
      $filename = 'status.txt';
      if (!$handle = fopen($filename, 'w')) {
            echo "Cannot open file ($filename)";
            exit;
       }
      if (fwrite($handle, "6") === FALSE) {
           echo "Cannot write to file ($filename)";
           exit;
      }
      fclose($handle);
      sleep(2);
      $filename = 'status.txt';
      if (!$handle = fopen($filename, 'w')) {
            echo "Cannot open file ($filename)";
            exit;
       }
      if (fwrite($handle, "0") === FALSE) {
           echo "Cannot write to file ($filename)";
```

```
            exit;
        }
        fclose($handle);
        echo "<meta http-equiv=\"REFRESH\" content=\"0.1;
 url=control.html\">";
        exit;
 };
 if ($turnright) {
        echo "Turn Right";
        $filename = 'status.txt';
        if (!$handle = fopen($filename, 'w')) {
             echo "Cannot open file ($filename)";
             exit;
         }
        if (fwrite($handle, "5") === FALSE) {
            echo "Cannot write to file ($filename)";
            exit;
        }
        fclose($handle);
        sleep(2);
        $filename = 'status.txt';
        if (!$handle = fopen($filename, 'w')) {
             echo "Cannot open file ($filename)";
             exit;
         }
        if (fwrite($handle, "6") === FALSE) {
            echo "Cannot write to file ($filename)";
            exit;
        }
        fclose($handle);
        sleep(2);
        $filename = 'status.txt';
        if (!$handle = fopen($filename, 'w')) {
             echo "Cannot open file ($filename)";
             exit;
         }
        if (fwrite($handle, "5") === FALSE) {
            echo "Cannot write to file ($filename)";
            exit;
        }
        fclose($handle);
        sleep(2);
        $filename = 'status.txt';
        if (!$handle = fopen($filename, 'w')) {
             echo "Cannot open file ($filename)";
             exit;
         }
        if (fwrite($handle, "0") === FALSE) {
            echo "Cannot write to file ($filename)";
            exit;
        }
        fclose($handle);
        echo "<meta http-equiv=\"REFRESH\" content=\"0.1;
 url=control.html\">";
        exit;
 };
 if ($fl) {$somecontent = 4;};
        if ($f) {$somecontent = 5;};
        if ($fr) {$somecontent = 6;};
        if ($bl) {$somecontent = 1;};
        if ($b) {$somecontent = 2;};
```

```
      if ($br) {$somecontent = 3;};
      if ($stop) {$somecontent = 0;};
$filename = 'status.txt';
//if (is_writable($filename)) {
    if (!$handle = fopen($filename, 'w')) {
         echo "Cannot open file ($filename)";
         exit;
    }
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Cannot write to file ($filename)";
        exit;
    }
    fclose($handle);
      echo "<meta http-equiv=\"REFRESH\" content=\"0.1;
url=control.html\">";
//    header("Location: http://localhost/baas/control.html");
/*} else {
    echo "The file $filename is not writable";
}*/
?>
```

Alert

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<meta http-equiv="REFRESH" content="2; url=alert.php">
</head>
<body>
<table width="100" height="61" align="center" cellpadding="0"
cellspacing="0">
        <tr>
          <td height="20"><div align="center"><font color="#000000"
size="3" face="Verdana, Arial, Helvetica, sans-
serif"><strong>Alert</strong></font></div></td>
        </tr>
        <tr>
          <td <?php
                        $filename = "alert.txt";
                        $handle = fopen($filename, "r");
                        $contents = fread($handle,
filesize($filename));
                        fclose($handle);
                        if ($contents == "9")
                        echo "bgcolor=\"#FF0000\"";
                        else echo "bgcolor=\"#009900\"";
              ?>
              > </td>
        </tr>
</table>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<div align="center">
```

```
<p><strong>&#3619;&#3634;&#3618;&#3594;&#3639;&#3656;&#3629;&#3612;&#
3641;&#3657;&#3592;&#3633;&#3604;&#3607;&#3635;</strong></p>
  <p><font size="2"> &#3609;&#3634;&#3618;
&#3600;&#3634;&#3609;&#3633;&#3609;&#3604;&#3619;
&#3648;&#3594;&#3639;&#3657;&#3629;&#3624;&#3636;&#3619;&#3636;&#3650
;&#3619;&#3592;&#3609;&#3660; 45010208 <br>
  &#3609;&#3634;&#3618; &#3597;&#3633;&#3600;
&#3611;&#3636;&#3656;&#3609;&#3648;&#3591;&#3636;&#3609; 45010223<br>
  &#3609;&#3634;&#3618;
&#3603;&#3633;&#3600;&#3623;&#3636;&#3607;&#3618;&#3660;
&#3611;&#3619;&#3632;&#3616;&#3634;&#3585;&#3619; 45010249</font></p>
</div>
</body>
</html>
```

ProjectSerialDlg.h : header file

```
//

#if
!defined(AFX_PROJECTSERIALDLG_H__945701DF_9639_45EF_8C37_7213111742B9
__INCLUDED_)
#define
AFX_PROJECTSERIALDLG_H__945701DF_9639_45EF_8C37_7213111742B9__INCLUDE
D_

#include "SerialPort.h" // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////
////////
// CProjectSerialDlg dialog

class CProjectSerialDlg : public CDialog
{
// Construction
public:
      //LONG OnCommunication(WPARAM ch, LPARAM port);
      CSerialPort m_port1;
      CProjectSerialDlg(CWnd* pParent = NULL);  // standard
constructor

// Dialog Data
      //{{AFX_DATA(CProjectSerialDlg)
      enum { IDD = IDD_PROJECTSERIAL_DIALOG };
            // NOTE: the ClassWizard will add data members here
      //}}AFX_DATA

      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CProjectSerialDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support
      //}}AFX_VIRTUAL

// Implementation
protected:
      HICON m_hIcon;

      // Generated message map functions
```

```
	//{{AFX_MSG(CProjectSerialDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg LONG OnCommunication(WPARAM ch, LPARAM port);
	afx_msg void OnButton1();
	afx_msg void OnTimer(UINT nIDEvent);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_PROJECTSERIALDLG_H__945701DF_9639_45EF_8C37_7213111742B9
__INCLUDED_)
```

CSerialPort.h : header file

```
#ifndef __SERIALPORT_H__
#define __SERIALPORT_H__
#define WM_COMM_BREAK_DETECTED          WM_USER+1   // A break was
detected on input.
#define WM_COMM_CTS_DETECTED        WM_USER+2   // The CTS (clear-to-
send) signal changed state.
#define WM_COMM_DSR_DETECTED        WM_USER+3   // The DSR (data-set-
ready) signal changed state.
#define WM_COMM_ERR_DETECTED        WM_USER+4   // A line-status
error occurred. Line-status errors are CE_FRAME, CE_OVERRUN, and
CE_RXPARITY.
#define WM_COMM_RING_DETECTED       WM_USER+5   // A ring indicator
was detected.
#define WM_COMM_RLSD_DETECTED       WM_USER+6   // The RLSD (receive-
line-signal-detect) signal changed state.
#define WM_COMM_RXCHAR                  WM_USER+7   // A character
was received and placed in the input buffer.
#define WM_COMM_RXFLAG_DETECTED         WM_USER+8   // The event
character was received and placed in the input buffer.
#define WM_COMM_TXEMPTY_DETECTED    WM_USER+9   // The last character
in the output buffer was sent.
class CSerialPort
{

public:
	// contruction and destruction
	CSerialPort();
	virtual             ~CSerialPort();
	// port initialisation
		BOOL        InitPort(CWnd* pPortOwner, UINT portnr = 1,
UINT baud = 19200, char parity = 'N', UINT databits = 8, UINT
stopsbits = 1, DWORD dwCommEvents = EV_RXCHAR | EV_CTS, UINT
nBufferSize = 512);

	// start/stop comm watching
	BOOL        StartMonitoring();
	BOOL        RestartMonitoring();
	BOOL        StopMonitoring();
	DWORD GetWriteBufferSize();
```

```
	DWORD GetCommEvents();
	DCB		GetDCB();
	void		WriteToPort(char* string);
protected:
	// protected memberfunctions
	void		ProcessErrorMessage(char* ErrorText);
	static UINT CommThread(LPVOID pParam);
	static void ReceiveChar(CSerialPort* port, COMSTAT comstat);
	static void WriteChar(CSerialPort* port);
	CWinThread*			m_Thread;
	// synchronisation objects
	CRITICAL_SECTION  m_csCommunicationSync;
	BOOL				m_bThreadAlive;
	HANDLE				m_hShutdownEvent;
	HANDLE				m_hComm;
	HANDLE				m_hWriteEvent;
	// Event array.
	// One element is used for each event. There are two event
handles for each port.
	// A Write event and a receive character event which is located
in the overlapped structure (m_ov.hEvent).
	// There is a general shutdown when the port is closed.
	HANDLE				m_hEventArray[3];
	// structures
	OVERLAPPED			m_ov;
	COMMTIMEOUTS		m_CommTimeouts;
	DCB				m_dcb;
	// owner window
	CWnd*				m_pOwner;
	// misc
	UINT				m_nPortNr;
	char*				m_szWriteBuffer;
	DWORD				m_dwCommEvents;
	DWORD				m_nWriteBufferSize;
};
#endif __SERIALPORT_H__
```

ProjectSerialDlg.cpp : implementation file

```
//
#include "stdafx.h"
#include "ProjectSerial.h"
#include "ProjectSerialDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
	//}}AFX_VIRTUAL
// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CProjectSerialDlg dialog
CProjectSerialDlg::CProjectSerialDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CProjectSerialDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CProjectSerialDlg)
		// NOTE: the ClassWizard will add member initialization
here
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon
in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
```

```
void CProjectSerialDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CProjectSerialDlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CProjectSerialDlg, CDialog)
	//{{AFX_MSG_MAP(CProjectSerialDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_MESSAGE(WM_COMM_RXCHAR, OnCommunication)
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_WM_TIMER()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
// CProjectSerialDlg message handlers
BOOL CProjectSerialDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// Add "About..." menu item to system menu.
	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}
	// Set the icon for this dialog.  The framework does this
automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
	// TODO: Add extra initialization here
	m_port1.InitPort(this,1,9600);
	m_port1.StartMonitoring();
	SetTimer(1, 2000, NULL);
	return TRUE;  // return TRUE  unless you set the focus to a
control
}
void CProjectSerialDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
```

```
}
// If you add a minimize button to your dialog, you will need the
code below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.
void CProjectSerialDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}
HCURSOR CProjectSerialDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
LONG CProjectSerialDlg::OnCommunication(WPARAM ch, LPARAM port)
{
	//CString strCh;
	//strCh.Format("%c from port %d",(char)ch,port);
	//MessageBox(strCh);
	CString strCh;
	strCh.Format("%c",ch);
	CFile myFile;
	if ( myFile.Open( "alert.txt", CFile::modeReadWrite  ) )
	{
		myFile.Write(strCh,1);
		SetDlgItemText(IDC_STATIC_REC,strCh);
		SetTimer(2,3000,NULL);
	}
	return 0;
}
void CProjectSerialDlg::OnButton1()
{
	// TODO: Add your control notification handler code here
	CFile myFile;
	char cData[2];
	cData[1] = 0;
	if ( myFile.Open( "status.txt", CFile::modeRead  ) )
	{
		//MessageBox("Open successful");
		myFile.Read(cData,1);
		m_port1.WriteToPort(cData);
		SetDlgItemText(IDC_STATIC_SEND,cData);
		for( int i = 0; i < 6000000; ++i );
```

```
            m_port1.WriteToPort("0");
            SetDlgItemText(IDC_STATIC_SEND,cData);
            for( int j = 0; j < 6000000; ++j );
      }
}
void CProjectSerialDlg::OnTimer(UINT nIDEvent)
{
      // TODO: Add your message handler code here and/or call default
      switch(nIDEvent)
      {
      case 1:
            OnButton1();
            break;
      case 2:
            CFile myFile;
            if ( myFile.Open( "alert.txt", CFile::modeReadWrite  ) )
            {
                  myFile.Write("8",1);
                  SetDlgItemText(IDC_STATIC_REC,"");
            }
            break;
      }
      CDialog::OnTimer(nIDEvent);
}
```

GAS SENSORS : TYPE AF63

ETHANOL (C_2H_5OH) SENSOR

DESCRIPTION:

Alcohol gas sensor made with thick film sensing element.

FEATURES:

- Constant heater voltage
- Tight resistance tolerance
- Short initial stabilisation time
- Typical applications include gas leak detection, alcohol detection

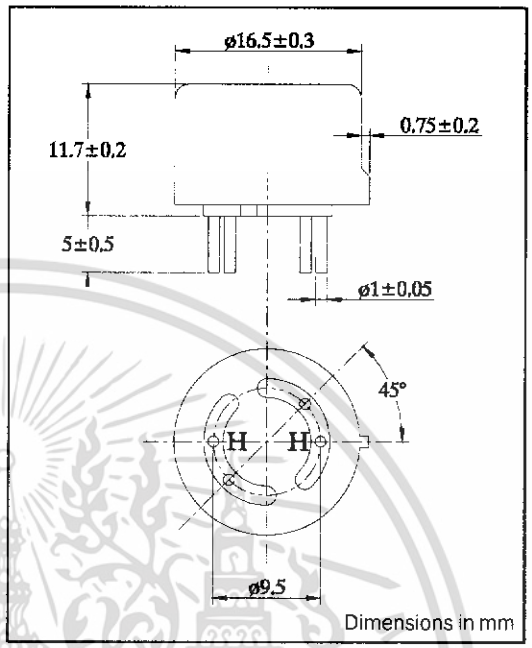
DATA:

Operating conditions:
Operating temperature ..... -10 to +55°C
Storage temperature ..... -30 to +60°C
Load Resistor R_L ..... Variable
Heater resistance ..... 19Ω (nom)
Rated power consumption P_s ..... <15mW
Rated working voltage of circuit V_c .....
..... 5V d.c. or 5V rms a.c. (max 12V)
Rated working voltage of heater .....
..... 5 ±0.2V d.c.
..... 5 ±0.2 V rms a.c.
Parts and material:
Sensing element ..... Semi-conducting oxide
Thick film heater ..... Platinum
Case ..... Nylon 66
Pin ..... Nickel alloy
Flame arrestor .....
... Double 100-mesh stainless gauze (SUS316)

Sensitivity characteristics:

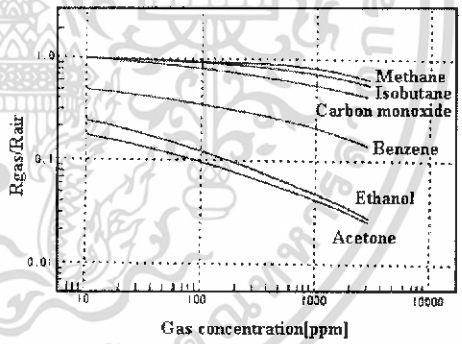
	Specification	Conditions
Sensor resistance Rgas	3k to 12k Ω	In clean air
Gas sensitivity Rgas/Rair	0.07 to 0.20	Resistance ratio at 100ppm ethanol to clean air
Power consumption	680mW (max)	

DIMENSIONS:



Dimensions in mm

Typical gas sensitivity:



Mechanical characteristics:

Test	Condition		Performance
Vibration	Frequency:	10 - 500 Hz	Should satisfy the specifications shown in the sensitivity characteristics
	Amplitude (10 - 50Hz):	2 mm	
	Acceleration (50 - 500 Hz)	10G	
	Recipricol scanning time:	5 min	
	Test time: 2 hours each for X, Y and Z directions		
Shock	Acceleration:	100G	
	Number of impacts:	5	

Crown Industrial Estate, Priorswood Rd
Taunton, Somerset TA2 8QY UK
Tel +44 (0)1823 335200
Fax +44 (0)1823 332637

808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

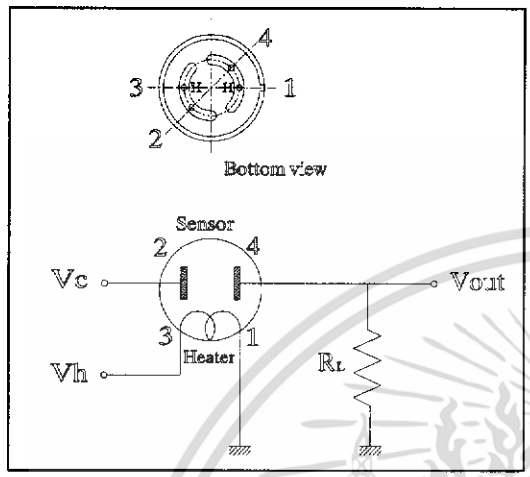
967 Windfall Road
St Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

GAS SENSORS : TYPE AF63

ALCOHOL (C_2H_5OH) SENSOR

NOTES:

Pin allocation and standard test circuit:



Test conditions:

Atmosphere

Clean air at 25± 2°C and 50±5% RH without noise gas.

Circuit condition

Vc (circuit voltage) ..... 5±0.05V
Vh (heater voltage) ..... 5±0.05V
Preheat time ..... 48 hours

Test gas

Ethanol 100ppm

WARNING:

Do not use if the case or wire netting is damaged, otherwise built-in heater may cause explosions or fires.
Do not disassemble or change any parts.
Use only within specified conditions.

Crown Industrial Estate, Priorswood Rd
Taunton, Somerset TA2 8QY UK
Tel +44 (0)1823 335200
Fax +44 (0)1823 332637

808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

967 Windfall Road
St Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

Order this document by ULN2803/D

MOTOROLA

ULN2803
ULN2804

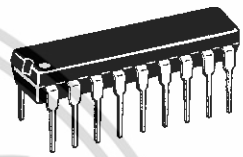
Octal High Voltage, High Current Darlington Transistor Arrays

OCTAL PERIPHERAL DRIVER ARRAYS

SEMICONDUCTOR TECHNICAL DATA

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.



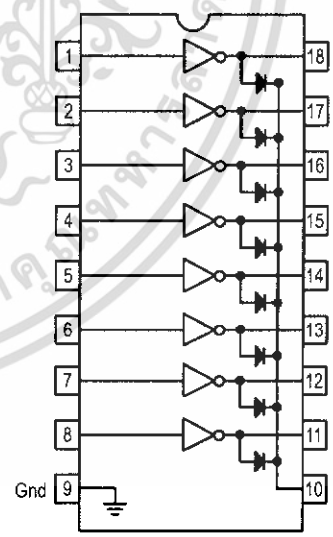
A SUFFIX
PLASTIC PACKAGE
CASE 707

MAXIMUM RATINGS (T_A = 25°C and rating apply to any one device in the package, unless otherwise noted.)

Rating	Symbol	Value	Unit
Output Voltage	V_O	50	V
Input Voltage (Except ULN2801)	V_I	30	V
Collector Current – Continuous	I_C	500	mA
Base Current – Continuous	I_B	25	mA
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Junction Temperature	T_J	125	°C

$R_{\theta JA}$ = 55°C/W
Do not exceed maximum current limit per driver.

PIN CONNECTIONS



ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	V_{CE}(Max)/I_C(Max)	Operating Temperature Range
ULN2803A ULN2804A	TTL, 5.0 V CMOS 6 to 15 V CMOS, PMOS	50 V/500 mA	T_A = 0 to + 70°C

ELECTRICAL CHARACTERISTICS (T_A = 25°C, unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Output Leakage Current (Figure 1)		I_{CEX}				µA
(V_O = 50 V, T_A = +70°C)	All Types		–	–	100	
(V_O = 50 V, T_A = +25°C)	All Types		–	–	50	
(V_O = 50 V, T_A = +70°C, V_I = 6.0 V)	ULN2802		–	–	500	
(V_O = 50 V, T_A = +70°C, V_I = 1.0 V)	ULN2804		–	–	500	
Collector–Emitter Saturation Voltage (Figure 2)		$V_{CE(sat)}$				V
(I_C = 350 mA, I_B = 500 µA)	All Types		–	1.1	1.6	
(I_C = 200 mA, I_B = 350 µA)	All Types		–	0.95	1.3	
(I_C = 100 mA, I_B = 250 µA)	All Types		–	0.85	1.1	
Input Current – On Condition (Figure 4)		$I_{I(on)}$				mA
(V_I = 17 V)	ULN2802		–	0.82	1.25	
(V_I = 3.85 V)	ULN2803		–	0.93	1.35	
(V_I = 5.0 V)	ULN2804		–	0.35	0.5	
(V_I = 12 V)	ULN2804		–	1.0	1.45	
Input Voltage – On Condition (Figure 5)		$V_{I(on)}$				V
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2802		–	–	13	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2803		–	–	2.4	
(V_{CE} = 2.0 V, I_C = 250 mA)	ULN2803		–	–	2.7	
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2803		–	–	3.0	
(V_{CE} = 2.0 V, I_C = 125 mA)	ULN2804		–	–	5.0	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2804		–	–	6.0	
(V_{CE} = 2.0 V, I_C = 275 mA)	ULN2804		–	–	7.0	
(V_{CE} = 2.0 V, I_C = 350 mA)	ULN2804		–	–	8.0	
Input Current – Off Condition (Figure 3) (I_C = 500 µA, T_A = +70°C)	All Types	$I_{I(off)}$	50	100	–	µA
DC Current Gain (Figure 2) (V_{CE} = 2.0 V, I_C = 350 mA)	ULN2801	h_{FE}	1000	–	–	–
Input Capacitance		C_I	–	15	25	pF
Turn–On Delay Time (50% E_I to 50% E_O)		t_{on}	–	0.25	1.0	µs
Turn–Off Delay Time (50% E_I to 50% E_O)		t_{off}	–	0.25	1.0	µs
Clamp Diode Leakage Current (Figure 6) (V_R = 50 V)	T_A = +25°C T_A = +70°C	I_R	–	–	50 100	µA
Clamp Diode Forward Voltage (Figure 7) (I_F = 350 mA)		V_F	–	1.5	2.0	V

TEST FIGURES

(See Figure Numbers in Electrical Characteristics Table)

Figure 1.

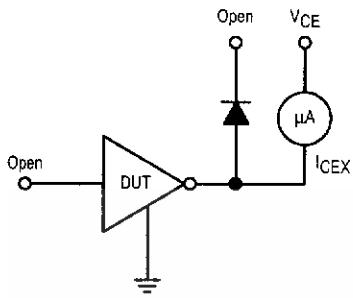


Figure 2.

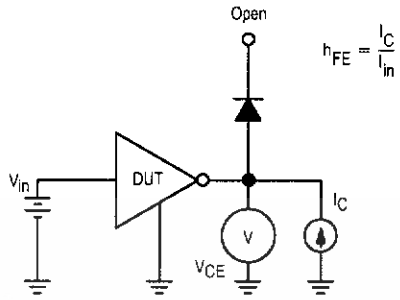


Figure 3.

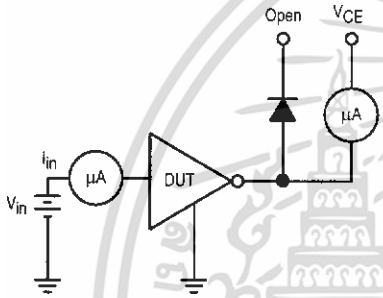


Figure 4.

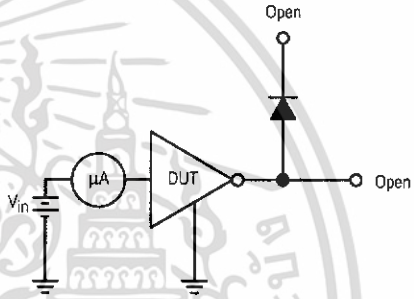


Figure 5.

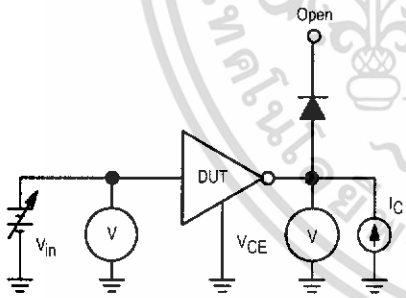


Figure 6.

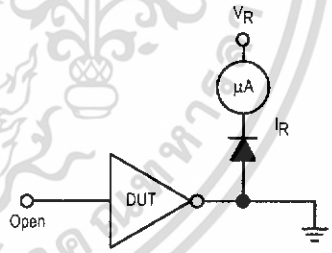
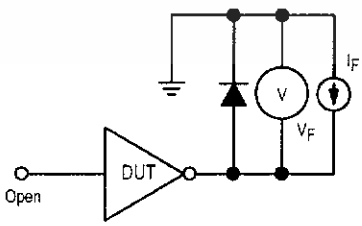


Figure 7.



TYPICAL CHARACTERISTIC CURVES – T_A = 25°C, unless otherwise noted

Output Characteristics

Figure 8. Output Current versus Saturation Voltage

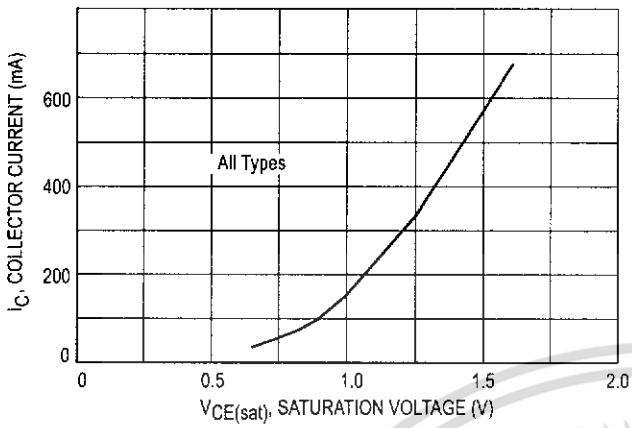
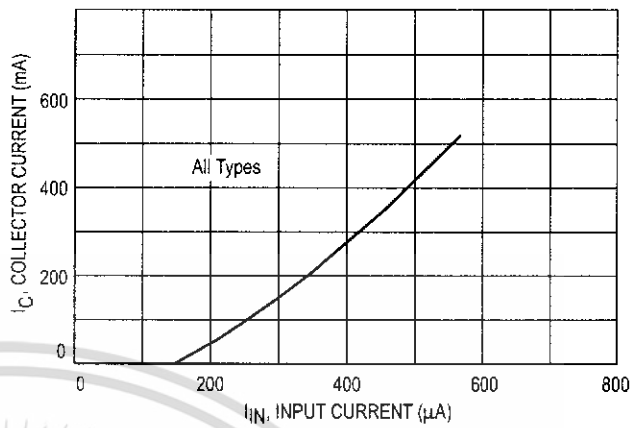


Figure 9. Output Current versus Input Current



Input Characteristics

Figure 10. ULN2803 Input Current versus Input Voltage

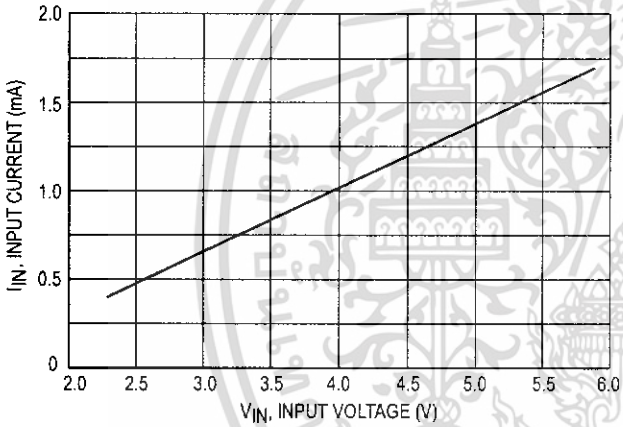


Figure 11. ULN2804 Input Current versus Input Voltage

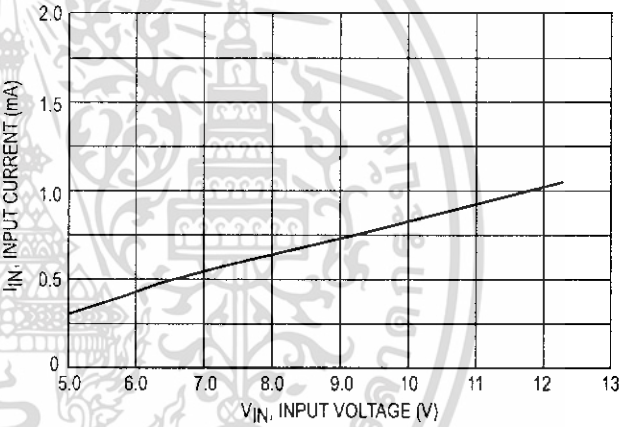
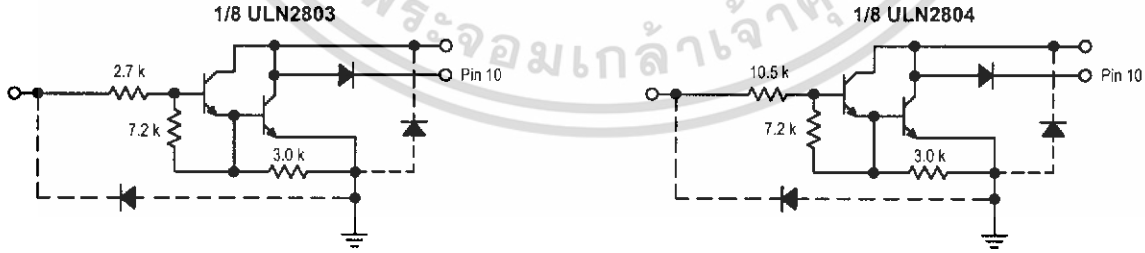
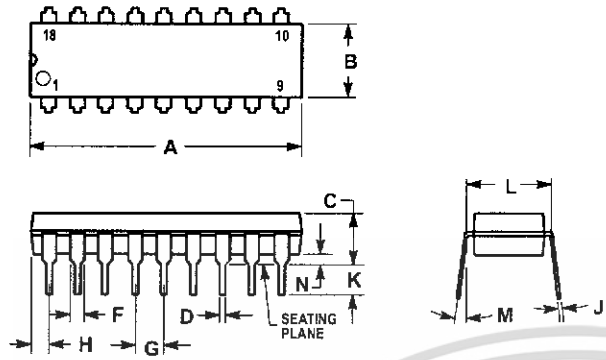


Figure 12. Representative Schematic Diagrams



OUTLINE DIMENSIONS

A SUFFIX
PLASTIC PACKAGE
CASE 707–02
ISSUE C



NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	22.22	23.24	0.875	0.915
B	6.10	6.60	0.240	0.260
C	3.56	4.57	0.140	0.180
D	0.36	0.56	0.014	0.022
F	1.27	1.78	0.050	0.070
G	2.54 BSC		0.100 BSC	
H	1.02	1.52	0.040	0.060
J	0.20	0.30	0.008	0.012
K	2.92	3.43	0.115	0.135
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

- **High-Current 3-State Outputs Drive Bus Lines Directly or up to 15 LSTTL Loads**
- **Data Flow-Through Pinout (All Inputs on Opposite Side From Outputs)**
- **Package Options Include Plastic Small-Outline (DW), Thin Shrink Small-Outline (PW), and Ceramic Flat (W) Packages, Ceramic Chip Carriers (FK), and Standard Plastic (N) and Ceramic (J) 300-mil DIPs**

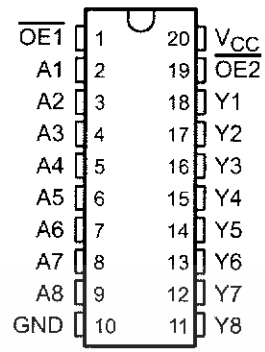
description

These octal buffers and line drivers feature the performance of the 'HC240 and a pinout with inputs and outputs on opposite sides of the package. This arrangement greatly enhances printed circuit board layout.

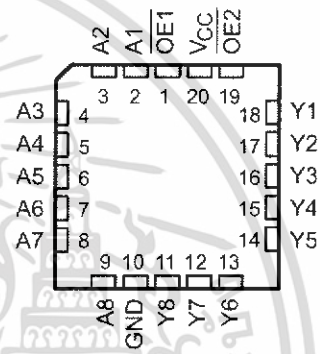
The 3-state control gate is a 2-input NOR. If either output-enable ($\overline{OE1}$ or $\overline{OE2}$) input is high, all eight outputs are in the high-impedance state. The 'HC541 provide true data at the outputs.

The SN54HC541 is characterized for operation over the full military temperature range of –55°C to 125°C. The SN74HC541 is characterized for operation from –40°C to 85°C.

SN54HC541 . . . J OR W PACKAGE
SN74HC541 . . . DW, N, OR PW PACKAGE
(TOP VIEW)



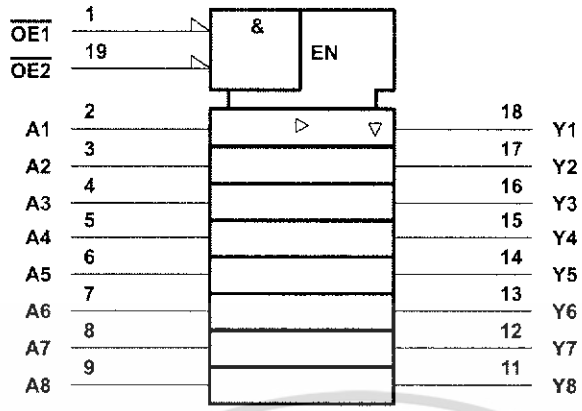
SN54HC541 . . . FK PACKAGE
(TOP VIEW)



FUNCTION TABLE
(each buffer/driver)

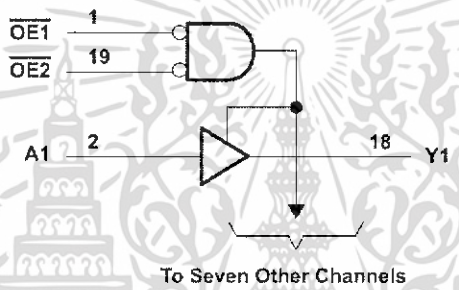
INPUTS			OUTPUT
$\overline{OE1}$	$\overline{OE2}$	A	Y
L	L	L	L
L	L	H	H
H	X	X	Z
X	H	X	Z

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range‡

Supply voltage range, V_{CC}	–0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see Note 1)	±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$) (see Note 1)	±20 mA
Continuous output current, I_O ($V_O = 0$ to V_{CC})	±35 mA
Continuous current through V_{CC} or GND	±70 mA
Package thermal impedance, θ_{JA} (see Note 2): DW package	97°C/W
N package	67°C/W
PW package	128°C/W
Storage temperature range, T_{stg}	–65°C to 150°C

‡ Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

recommended operating conditions

			SN54HC541			SN74HC541			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V	0		0.5	0		0.5	V
		V_{CC} = 4.5 V	0		1.35	0		1.35	
		V_{CC} = 6 V	0		1.8	0		1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
t_t	Input transition (rise and fall) time	V_{CC} = 2 V	0		1000	0		1000	ns
		V_{CC} = 4.5 V	0		500	0		500	
		V_{CC} = 6 V	0		400	0		400	
T_A	Operating free-air temperature		–55		125	–40		85	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		V_{CC}	T_A = 25°C			SN54HC541		SN74HC541		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
V_{OH}	V_I = V_{IH} or V_{IL}	I_{OH} = –20 µA	2 V	1.9	1.998		1.9		1.9		V
			4.5 V	4.4	4.499		4.4		4.4		
			6 V	5.9	5.999		5.9		5.9		
		I_{OH} = –6 mA	4.5 V	3.98	4.3		3.7		3.84		
		I_{OH} = –7.8 mA	6 V	5.48	5.8		5.2		5.34		
V_{OL}	V_I = V_{IH} or V_{IL}	I_{OL} = 20 µA	2 V		0.002	0.1		0.1		0.1	V
			4.5 V		0.001	0.1		0.1		0.1	
			6 V		0.001	0.1		0.1		0.1	
		I_{OL} = 6 mA	4.5 V		0.17	0.26		0.4		0.33	
		I_{OL} = 7.8 mA	6 V		0.15	0.26		0.4		0.33	
I_I	V_I = V_{CC} or 0		6 V		±0.1	±100		±1000		±1000	nA
I_{OZ}	V_O = V_{CC} or 0		6 V		±0.01	±0.5		±10		±5	µA
I_{CC}	V_I = V_{CC} or 0,	I_O = 0	6 V			8		160		80	µA
C_i			2 V to 6 V		3	10		10		10	pF

switching characteristics over recommended operating free-air temperature range, C_L = 50 pF (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC}	T_A = 25°C			SN54HC541		SN74HC541		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A	Y	2 V		40	115		171		144	ns
			4.5 V		12	23		34		29	
			6 V		10	20		29		25	
t_{en}	$\overline{OE}$	Y	2 V		80	150		224		188	ns
			4.5 V		17	30		45		38	
			6 V		15	26		38		32	
t_{dis}	$\overline{OE}$	Y	2 V		40	150		224		188	ns
			4.5 V		18	30		45		38	
			6 V		17	26		38		32	
t_t		Y	2 V		28	60		90		75	ns
			4.5 V		8	12		18		15	
			6 V		6	10		15		13	

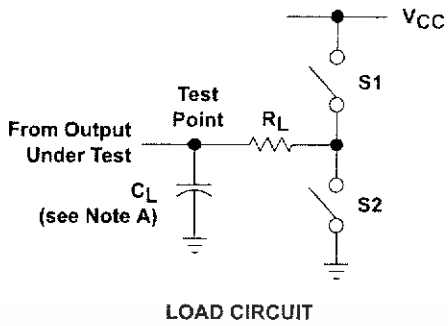
switching characteristics over recommended operating free-air temperature range, C_L = 150 pF (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC}	T_A = 25°C			SN54HC541		SN74HC541		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A	Y	2 V		65	165		246		206	ns
			4.5 V		16	33		49		41	
			6 V		14	28		42		35	
t_{en}	$\overline{OE}$	Y	2 V		100	200		298		250	ns
			4.5 V		20	40		60		50	
			6 V		17	34		51		43	
t_t		Y	2 V		45	210		315		265	ns
			4.5 V		17	42		63		53	
			6 V		13	36		53		45	

operating characteristics, T_A = 25°C

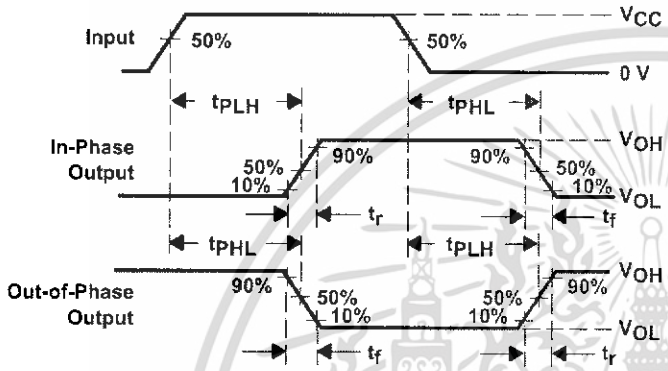
PARAMETER		TEST CONDITIONS	TYP	UNIT
C_{pd}	Power dissipation capacitance per buffer/driver	No load	35	pF

PARAMETER MEASUREMENT INFORMATION



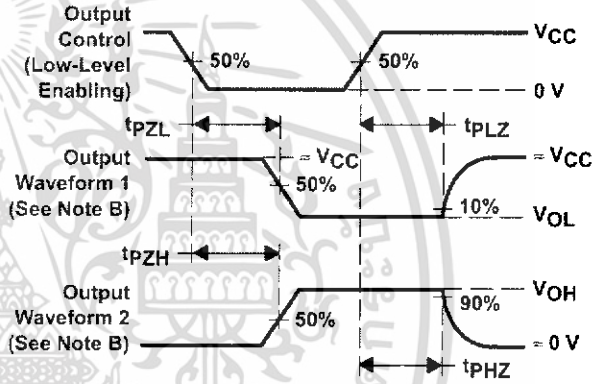
LOAD CIRCUIT

PARAMETER		R_L	C_L	S1	S2
t_{en}	t_{PZH}	1 kΩ	50 pF or 150 pF	Open	Closed
	t_{PZL}			Closed	Open
t_{dis}	t_{PHZ}	1 kΩ	50 pF	Open	Closed
	t_{PLZ}			Closed	Open
t_{pd} or t_t		—	50 pF or 150 pF	Open	Open



VOLTAGE WAVEFORMS
PROPAGATION DELAY AND OUTPUT TRANSITION TIMES

VOLTAGE WAVEFORM
INPUT RISE AND FALL TIMES



VOLTAGE WAVEFORMS
ENABLE AND DISABLE TIMES FOR 3-STATE OUTPUTS

NOTES: A. C_L includes probe and test-fixture capacitance.
B. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.
C. Phase relationships between waveforms were chosen arbitrarily. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, Z_O = 50 Ω, t_r = 6 ns, t_f = 6 ns.
D. The outputs are measured one at a time with one input transition per measurement.
E. t_{PLZ} and t_{PHZ} are the same as t_{dis}.
F. t_{PZL} and t_{PZH} are the same as t_{en}.
G. t_{PLH} and t_{PHL} are the same as t_{pd}.

Figure 1. Load Circuit and Voltage Waveforms