ควบคุมกล้องวิดีโอผ่านเครือข่ายอินเตอร์เน็ต

CAMERA CONTROL VIA INTERNET

โดย

นางสาวภาวิณี หรุ่นสูงเนิน 40010562

นางสาวศศิธร ตังทิวะนนท์ 40010769

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ควบคุมกล้องวิดีโอผ่านเครือข่ายอินเตอร์เน็ต

CAMERA CONTROL VIA INTERNET

โดย

นางสาวภาวิณี หรุ่นสูงเนิน 40010562

นางสาวศศิธร ตังทิวะนนท์ 40010769

อาจารย์ที่ปรึกษา

ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับวิศวกรรมศาสตรบัณฑิต สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ ปีการศึกษา 2543

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ควบคุมกล้องผ่านเครือข่ายอินเตอร์เน็ต

ผู้จัดทำ

นางสาวภาวิณี หรุ่นสูงเนิน 40010562

นางสาวศศิธร ตังทิวะนนท์ 40010769

.....

(ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

ปริญญานิพนธ์	ควบคุมกล้องผ่านเครือข่ายอินเตอร์เน็ต
	Camera Control via Internet
จัดทำโดย	นางสาวภาวิณี หรุ่นสูงเนิน
	นางสาวศศิธร ตังทิวะนนท์
อาจารย์ที่ปรึกษา	ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์)

วันที่ 21 / 3 / 01

กิตติกรรมประกาศ

ผู้จัดทำรายงานขอขอบพระคุณ ผศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน และคณาจารย์ท่านอื่นๆ ที่ได้ให้ความช่วยเหลือและสนับสนุนการทำโครงงานนี้โดยให้คำปรึกษา ให้ข้อคิดเห็นแนวทางในการแก้ไขปัญหา รวมทั้งอนุญาตให้ใช้เครื่องมืออุปกรณ์การทดลองต่างๆ จนสามารถทำให้โครงงานชิ้นนี้สำเร็จมาได้ด้วยดี และท้ายนี้ขอขอบพระคุณเพื่อนๆ ที่ได้ช่วยเหลือ สนับสนุน และเป็นกำลังใจให้โครงงานนี้สำเร็จลุล่วงด้วยดีไว้ ณ ที่นี้ด้วย

ภาวิณี หรุ่นสูงเนิน
.....
นางสาวภาวิณี หรุ่นสูงเนิน

ศศิธร ตังทิวนนท์
.....
นางสาวศศิธร ตังทิวะนนท์

ควบคุมกล้องวีดีโอผ่านเครือข่ายอินเตอร์เนต

นางสาวภาวิณี หรุ่นสูงเนิน 40010562
นางสาวศศิธร ตังทิวะนนท์ 40010769
ผศ.ดร. สุรพันธุ์ เอื่อไพบูลย์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2543

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ได้นำเอาความสามารถสามารถของคอมพิวเตอร์ และซอร์พแวร์เดลไฟมาประยุกต์ใช้งาน ในส่วนที่ทำหน้าที่ในการควบคุมการหมุนของกล้อง จะอาศัยสเตปปิ้งมอเตอร์ควบคุมโดยใช้ไมโครคอนโครลเลอร์ รับข้อมูลมาจากพอร์ตอนุกรมของคอมพิวเตอร์ โดยมี MAX232 เพื่อปรับระดับสัญญาณ ก่อนเข้าไมโครคอนโทรลเลอร์ แล้วผ่านวงจรไดร์เวอร์ เพื่อให้มีอัตราการขยายกระแสสูง ในส่วนของการควบคุมและการส่งภาพจะสามารถติดต่อทางอินเตอร์เนตได้

Camera Control via Internet

Miss. Pawinee Roonsungnoen
Miss. Sasithorn Tungtiwanon
Assist. Prof.Dr.Surapan Airphaiboon
Education year 2000

Abstract

This thesis present , using computer with software Delphi application. For the task about the rotate camera we use microcontroller to receive data from serial port of computer by using MAX232 to adjust signal , and the driver circuit is high current gain. We can control and send Image via internet.

สารบัญ

สารบัญรูปภาพ

หน้า

บทที่ 1

บทนำ

ในปัจจุบันวิทยาการทางด้านอิเล็กทรอนิกส์ และคอมพิวเตอร์ได้มีการพัฒนาอย่างรวดเร็ว ทำให้มีสิ่งอำนวยความสะดวกหลายอย่างเกิดขึ้นมากมาย ซึ่งโครงงานชิ้นนี้เป็นอีกรูปแบบหนึ่งในการประยุกต์ใช้ความสามารถของคอมพิวเตอร์ และใช้การเขียนโปรแกรมบนวินโดว์ (Window) ด้วย ภาษาเดลไฟ (Delphi) ซึ่งเป็นภาษาที่พัฒนามาจากภาษาปาสคาล (Pascal) ในการควบคุมให้สเตปปิ้งมอเตอร์ หมุนฐานกล้องติดตั้ง

ภาษาเดลไฟ มีรูปแบบการเขียนในลักษณะจำลอง (Virtual) คือ การนำองค์ประกอบต่างของโปรแกรมมาประกอบเข้าด้วยกัน โดยผู้เขียนโปรแกรม (Programmer) ทำหน้าที่เพียงเขียนชุดคำสั่งในการควบคุมองค์ประกอบเหล่านั้น นอกจากนี้การเขียนโปรแกรมบนวินโดว์ยังสนับสนุนการใช้งานแบบกราฟฟิก (Graphics Mode) ซึ่งทำให้โปรแกรมมีความสะดวกต่อผู้ใช้ และมีรูปแบบที่สวยงาม จุดเด่น อีกประการหนึ่งของการเขียนโปรแกรมในลักษณะนี้ก็คือ ช่วยลดความซับซ้อนในการเขียนโปรแกรม และช่วยลดเวลาในการพัฒนาโปรแกรมเนื่องจากสามารถนำออบเจคที่สร้างไว้แล้วมาใช้ใหม่ได้ตลอดเวลา

ในรายงานฉบับนี้จะกล่าวถึงทฤษฎีที่ต้องเรียนรู้ และหลักการออกแบบสร้างโครงสร้างโครงงานนี้ โดยแบ่งเนื้อหาออกเป็นบท ๆ ดังต่อไปนี้

บทที่1 : กล่าวถึงจุดประสงค์ในการทำโครงงาน และรายละเอียดคร่าวๆ

บทที่2 : กล่าวถึงระบบการรับและแสดงภาพ

บทที่3 : กล่าวถึงการรับส่งข้อมูลแบบอนุกรม ของไมโครคอนโทรลเลอร์ MCS-51

บทที่4 : กล่าวถึงทฤษฎีสเตปปิ้งมอเตอร์และวงจรขับ

บทที่5 : กล่าวถึงเครือข่ายอินเตอร์เน็ต

บทที่6 : กล่าวถึงหลักการออกแบบและการสร้าง

บทที่7 : ผลการทดลอง สรุป และวิจารณ์

ภาคผนวก : แสดงซอสโคดโปรแกรม , Datasheet ของทรานซิสเตอร์ , รูปวงจรที่ใช้ และแผนผังการทำงานของโปรแกรม

จุดประสงค์ของโครงงาน

<u>ส่วนของการทำงาน</u>

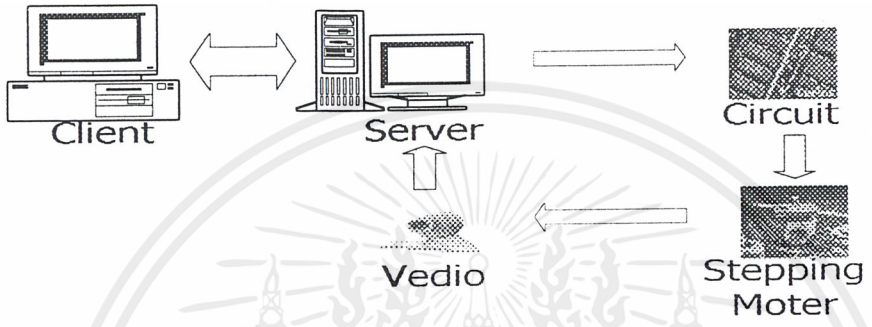
- เพื่อศึกษาทฤษฎีของ สเตปปิ้งมอเตอร์ และวงจรขับ รวมทั้งการนำไปใช้
- เพื่อศึกษาการเขียนโปรแกรมด้วยเดลไฟ ซึ่งมีพื้นฐานมาจากภาษาปาสคาล
- ได้ประยุกต์การใช้คอมพิวเตอร์ควบคุมอุปกรณ์ คือ สเตปปิ้งมอเตอร์ โดยการเชื่อมทางพอร์ตอนุกรม

ส่วนของผลงาน

- เป็นชิ้นงานที่สามารถนำไปใช้งานได้ โดยนำไปประยุกต์ใช้เป็นฐานติดตั้งกล้องโทรทัศน์วงจรปิด หรือกล้องวีดีโอ
- เป็นแนวทางเบื้องต้นในการนำไปพัฒนาให้มีความสามารถ และมีความสมบูรณ์มากขึ้น

แนวความคิด

ในหัวข้อนี้จะขอกล่าวถึงขั้นตอนการทำงานอย่างคร่าว ๆ โดยอาศัยการอธิบายตามการแสดงการทำงานด้วย Block diagram ดังแสดงด้วยรูปที่ 1.1



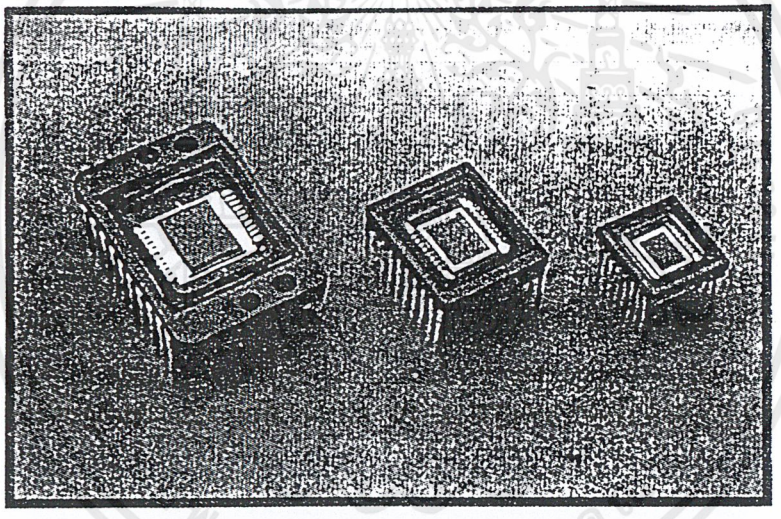
รูปที่ 1.1 แสดง Block diagram การทำงาน

ภาคคอมพิวเตอร์ส่วนบุคคล : ในภาคนี้เป็นส่วนที่ใช้ในการควบคุมการเคลื่อนที่ของกล้องวีดีโอ และแสดงภาพ

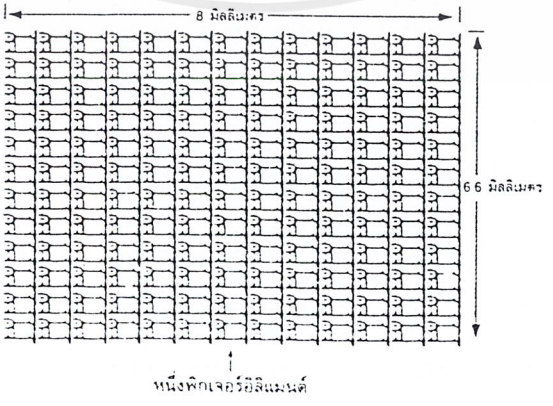
ภาคควบคุมการหมุนของกล้อง : ในภาคนี้จะประกอบด้วยบอร์ดควบคุมที่ใช้ IC MCS-51 , บอร์ดขับกระแสสำหรับสเตปปิ้งมอเตอร์ , ชุดเฟืองสำหรับหมุนกล้อง , และตัวกล้องวีดีโอขนาดเล็ก

บทที่ 2
ระบบการรับและแสดงภาพ

ในส่วนประกอบของระบบการรับและแสดงภาพนั้น สิ่งที่ขาดไม่ได้นั่นคือ กล้องโทรทัศน์ดิจิตอล และเครื่องคอมพิวเตอร์ที่ใช้แสดงผลภาพ ดังนั้น จึงควรทำความเข้าใจกับหลักการพื้นฐานของกล้องโทรทัศน์ดิจิตอล (Charge Coupled Device หรือ CCD) ส่วนในการนำสัญญาณภาพโทรทัศน์จากกล้องดิจิตอล มาแสดงบนหน้าจอคอมพิวเตอร์นั้น ต้องอาศัยซอฟแวร์ในการ Interface สัญญาณจากกล้องให้ติดต่อกับคอมพิวเตอร์ได้แล้วจึงนำภาพมาแสดงได้ ดังนั้น หลักการพื้นฐานของระบบรับและแสดงภาพสามารถที่จะแบ่งได้เป็นสองอย่าง คือ การทำงานของกล้องโทรทัศน์ดิจิตอลและการทำงานทางซอฟแวร์ที่ใช้แสดงผลภาพโทรทัศน์



รูปที่ 2.1 (ก) กล้องโทรทัศน์ CCD ขนาดต่างๆ



รูปที่ 2.1 (ข) ส่วนรับภาพและพิคเจอร์อิลีเมนต์

2.1 หลักการพื้นฐานของกล้องโทรทัศน์

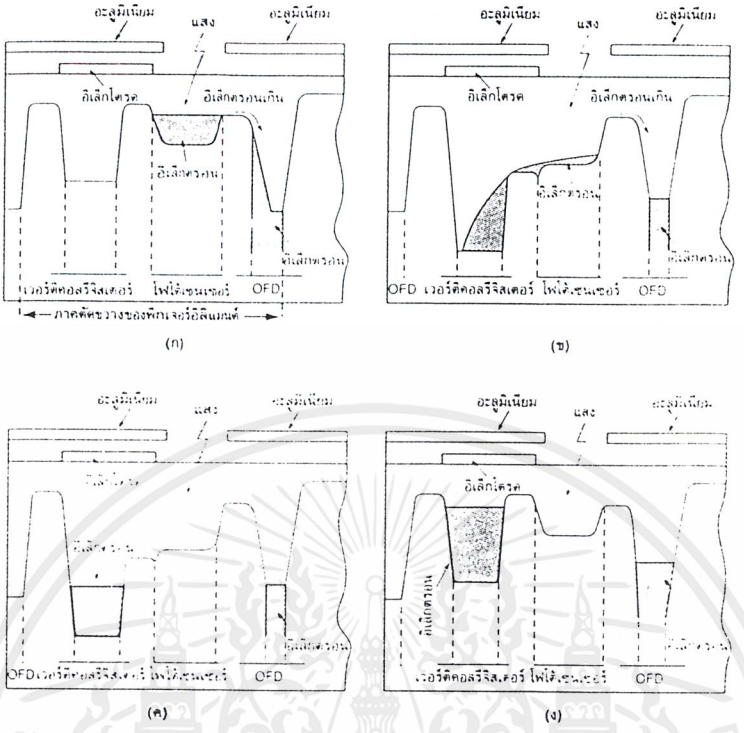
CCD หรือ Charge Couple Device ก็คือ VISI (Very Large Scale IC) ที่บรรจุด้วยจุดรับภาพมากกว่า 250,000 จุด อยู่บนแผ่นตรวจจับแสงในพื้นที่ขนาดเล็กเพียง 8.8*6.6 มิลลิเมตร เป็นผลจากการพัฒนาของเทคโนโลยีทางด้านอุปกรณ์สารกึ่งตัวนำ CCD จะให้อิเล็กตรอนเมื่อมีแสงสว่างตกกระทบตัวมัน ปริมาณอิเล็กตรอนที่เกิดขึ้นจะเปลี่ยนแปลงสัดส่วนกับความเข้มของแสงที่ได้รับ โดยแบ่งการทำงานภายใน CCD เป็น 3 ขั้นตอนดังนี้

1. กำเนิดอิเล็กตรอนจากแสงที่ได้รับ (Detect incident light) จุดรับภาพแต่ละจุดจะให้กำเนิดอิเล็กตรอนตามความเข้มของแสงที่ได้รับซีลด์อะลูมิเนียมของจุดรับภาพ ตามรูปที่ 2.2 จะทำหน้าที่บังแสงไม่ให้ตกกระทบไปยังส่วนอื่นๆ ยกเว้นบริเวณตังจับแสง

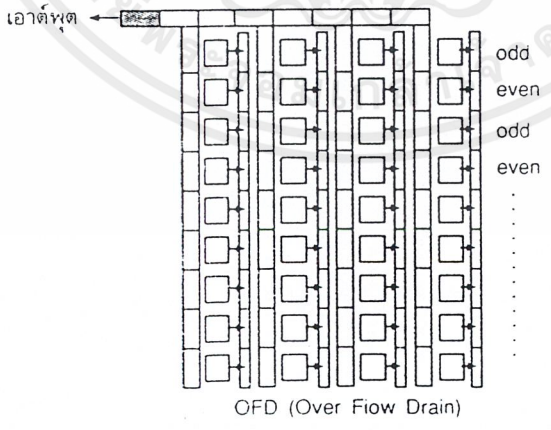
2. เก็บรักษาอิเล็กตรอน (Store induced charge) ค่าความต่างศักย์ดังแสดงในรูปที่ 2.2 จะอยู่ภายในของจุดรับภาพแต่ละจุด อิเล็กตรอนที่ถูกเก็บอยู่ในตัวจับแสงจำนวนมากจะไหลลงสู่ OFD และขจัดปัญหาการเกิดหางของภาพ ซึ่งเกิดขึ้นเสมอสำหรับภาพชนิดหลอดด้วยวิธีนี้จะสามารถรักษาระดับสูงสุดของอิเล็กตรอนให้คงที่แม้ว่าจะมีแสงสว่างมากเกินไป

3. ถ่ายเทอิเล็กตรอน ขณะที่มีแรงดันป้อนให้กับอิเล็กโตรของจุดรับภาพส่วนลึกของแต่ละแรงดันภายในอิเล็กโตรดจะเพิ่มขึ้น อิเล็กตรอนที่ถูกเก็บไว้จะเริ่มถ่ายเทลงมาให้กับรีจีสเตอร์ในแนวตั้ง ดังรูปที่ 2.2 (ก) ซึ่งเปรียบเสมือนประตูเขื่อนถูกเปิดออก ขณะที่อิเล็กตรอนถูกถ่ายทอดหมดแล้ว ตามความต่างศักย์ที่ป้อนให้กับอิเล็กโตรดจะหยุดด้วยแสดงดังรูป 2.2 (ค)นั่นคือความลึกของแต่ละแรงดันจะเหมือนเดิม การถ่ายเทอิเล็กตรอนจากโฟโตเซ็นเซอร์ไปสู่รีจีสเตอร์ในแนวตั้งทีสิ้นสุดลง ในระหว่างการถ่ายเทอิเล็กตรอนภายในอยู่ จะไม่มีการกำเนิดอิเล็กตรอนใหม่ ถึงแม้ว่าตัวจับแสงกำลังรับแสงอยู่ก็ตาม เป็นเพราะว่าความเร็วในการถ่ายเทนั้นสูงกว่าการเกิดอิเล็กตรอน การถ่ายเทอิเล็กตรอนของ CCD นี้สามารถตัดปัญหาการเผาไหม้และภาพที่มีลักษณะเป็นดาวหางได้

ทั้ง 3 ขั้นนี้เป็นหลักการการทำงานพื้นฐานของตัวรับภาพแบบ CCD เมื่อทำงานครบทั้งสามขั้นตอนแล้วส่งไปยังวงจรเอาท์พุทที่เรียกว่า อินเตอร์ไลน์ทรานสเฟอร์ซีซีดี อิเมจเซนเซอร์



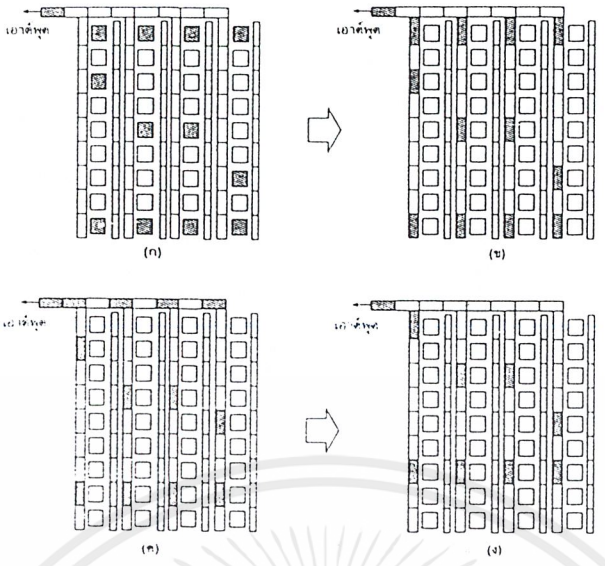
รูปที่ 2.2 ภายใน CCD พิกเจอร์อิลีเมนต์จะมีอิเล็กตรอนเกิดขึ้นและถ่ายเทไปทาง OFD



รูปที่2.3 ทิศทางการถ่ายเทของอิเล็กตรอนใน OFD

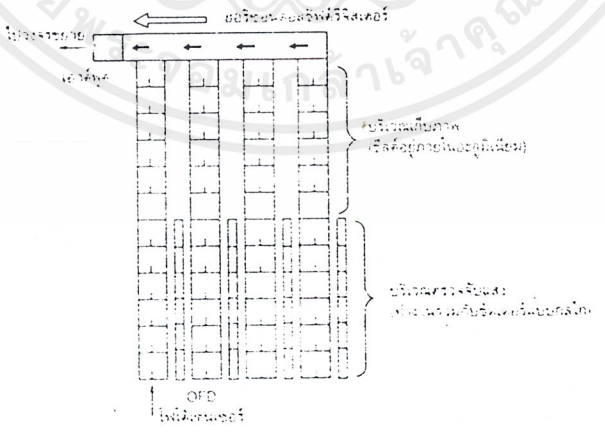
บริษัทโซนี่ได้ทำการอินเตอร์ไลน์ทรานสเฟอร์ให้กับตัวรับภาพ CCD เพื่อให้เห็นการฉายแสงลงบนพื้นที่ CCD นั้นเหมือนกับปรากฏบนจอมอนิเตอร์ จากรูปรีจีสเตอรีในแนวตั้งและ OFD ได้เชื่อมต่อประสานกับรีจีสเตอร์ในแนวระดับ ข้อพิเศษทางเทคนิกของโครงสร้าง CCD แบบนี้ก็คือ ลดอาการของภาพที่มีสีเลอะ หรือเป็นหิมะที่เกิดขึ้นถูกถ่ายเทไปยังรีจีสเตอร์ซึ่งเวอร์ติคอลชีฟต์รีจีสเติร์และฮอริซอนตอลชีฟรีจีสเตอร์เหมือรกับถนนและอิเล็กตรอนที่ถูกถ่ายเทก็เหมือนรถวิ่งบนถนน เคลื่อนที่ไปอย่างมีลำดับต่อเนื่องกันตลอด ตามรูปที่ 2.4 ไปสู่ภาคขยายอินเตอร์ไลน์ทรานเฟอร์ CCD อิเมจเซนเวอร์จะมีโครงสร้างที่ง่ายกว่า แต่มีข้อเสียอยู่หลายประการดังนี้คือ

1. มีขนาดใหญ่ทำให้ต้นทุนการผลิตสูง
2. เนื่องจากการถ่ายเทอิเล็กตรอนระหว่างโฟโตเซ็นซิ่งกับสตอเรจเป็นไปด้วยความรวดเร็วมากดังนั้นจึงทำให้รายละเอียดทางด้านแนวตั้งลดลง ซึ่งเปรียบเสมือนกับคนที่หิ้วน้ำเต็มถังแล้ววิ่งด้วยความเร็ว น้ำย่อมมีการกระฉอกออกไปบ้าง ทำให้ประสิทธิภาพที่ได้มานั้นเสียไป
3. ในระหว่างการถ่ายเทอิเล็กตรอนโฟโตเซนซิ่งรับหน้าที่เป็นรีจีสเตอร์แบบกลไก ใช้เพื่อตัดแสงในการถ่ายเทอิเล็กตรอน มิฉะนั้นภาพจะเลอะเลือน อย่างไรก็ตามชัตเตอร์แลลกลไก ทำให้เกิดปัญหาหลายประการ รูปที่ 2.6 เฟรมทรานสเฟอร์อิเมจเซนเซอร์ ข้อได้เปรียบระหว่างกล้อง CCD แบบ 3 ชิปเหนือกว่ากล้อง 3 หลอด ในเรื่องความไวแสงและภาพเกิดรอยไหม้น้อย ใช้กำลังไฟน้อยและมีความทนทาน แต่กลอง 3หลอดมีข้อดีตรงที่มีความชัดเจนสูงกว่า ดังนั้นกล้อง CCD แบบ 3 ชิป จึงเหมาะกับงานที่ถ่ายภาพนอกสถานที่ ซึ่งต้องการกล้องที่มีความต้านทานต่อการเผาไหม้สูง มีความไวและความแข็งแรงทนทานเป็นต้น ส่วนกล้อง 3 หลอด CCD นันใช้เมื่อต้องการคุณภาพของกล้องเท่านั้น

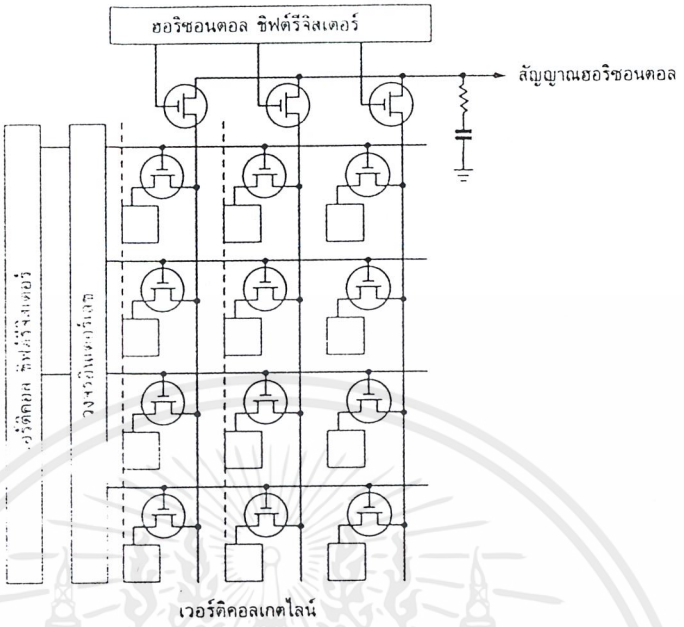


รูปที่ 2.4 การถ่ายเทอิเล็กตรอนโดยผ่านทางรีจีสเตอร์ในแนวตั้งและรีจีสเตอร์ในแนวระดับ

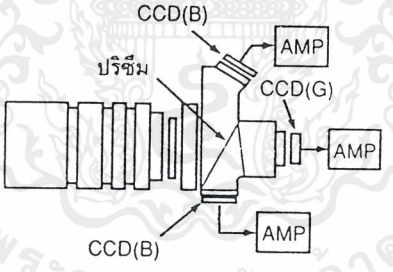
รูปที่ 2.5 การถ่ายเทอิเล็กตรอนทำให้เกิดมีอิเล็กตรอนสูญเสีย



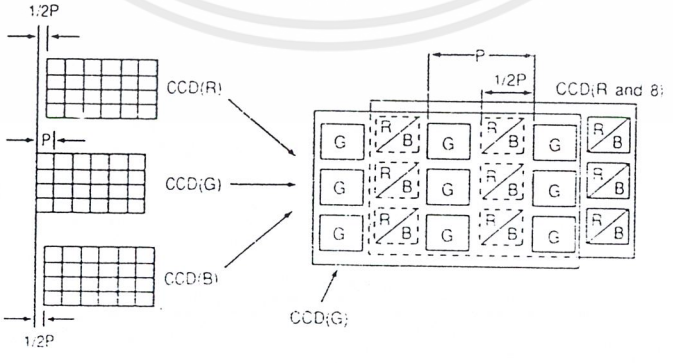
รูปที่2.6 โครงสร้างของ CCD แบบเฟรมทรานสเฟอร์ม



รูปที่ 2.7 โครงสร้างของตัวรับแบบ MOS



(ก) ตำแหน่งของ CCD ภายในกล้อง



(ข) CCD สีแดงและสีน้ำเงินจะมีระบบต่างกับ CCD สีเขียว 1/2 พิคซ์

รูปที่ 2.8เทคโนโลยีชดเชย Original Spatial Effect ของโซนี่

ข้อเปรียบเทียบระหว่างกล้อง CCD แบบ 3 ชิป กับกล้องแบบ MOS ทั่วไป

2.1.1 ตัวรับภาพแบบ MOS

จะใช้ทรานซิสเตอร์แบบ MOS เป็นตัวสวิทชิ่งคล้ายกับหน่วยความจำแบบ DRAM ดังรูปที่ 2.7 แม้ว่าอุปกรณ์ชนิดนี้จะสามารถเก็บประจุได้เป็นจำนวนมาก แต่มันก็ทำให้เกิดสัญญาณรบกวนเป็นจำนวนมากเช่นกัน และตัวรับภาพ CCD ก็มีคุณสมบัติเหนือกว่ามากในด้านของความไวแสง และนอยส์ โดยใช้เทคโนโลยี Original Spatial Effect ของโซนี่ โดยการติดตั้ง CCD สีแดงและสีน้ำเงินให้มีระยะแตกต่างจากสีเขียว ½ พิตช์ ดังแสดงในรูปที่ 2.8 ด้วยเทคนิคอันนี้จะทำให้ได้พิกเจอร์อิลีเมนต์เป็น 2 เท่า และ ทำให้ความชัดเจนของภาพสูงสุด

2.1.2 ประโยชน์และคุณภาพของกล้อง CCD

1. ภาพที่ได้มีการสูญเสียน้อย
2. มีความไวสูง
3. มีน้ำหนักเบา
4. ให้ภาพที่ละเอียดและคมชัด
5. กล้องไม่เป็นรูปรอยไหม้เมื่อถูกแสงอาทิตย์
6. ไม่เกิดภาพล้า
7. ไม่เกิดภาพเป็นรูปดาวหาง
8. เมื่อสัญญาณแสงต่ำไม่ต้องตั้งปรับ

ปัจจุบันกล้องส่วนใหญ่ใช้ CCD เป็นตัวรับภาพ จึงควรทราบหลักการและโครงสร้างของกล้องโทรทัศน์ CCD หากต้องการศึกษาลึกลงไปถึงระดับโครงสร้างของสารกึ่งตัวนำก็สามารถศึกษาเพิ่มเติมได้จากแหล่งข้อมูลทางอินเตอร์เน็ตก็มีมากมาย

บทที่ 3
การรับส่งข้อมูลแบบอนุกรม

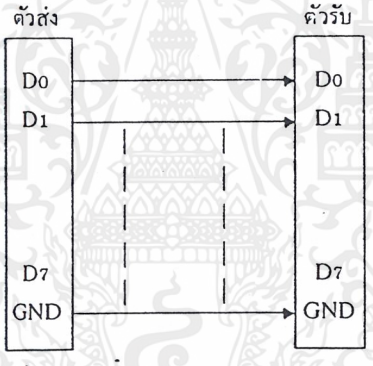
3.1 พื้นฐานการรับส่งข้อมูล

การรับส่งข้อมูลในระบบคอมพิวเตอร์โดยทั่วไปจะหมายถึงการรับส่งข้อมูลเป็นจำนวนไบต์ ๆ ให้กับอุปกรณ์ที่เกี่ยวข้องกับคอมพิวเตอร์ ซึ่งอาจแบ่งประเภทของการรับส่งข้อมูลได้ 2 แบบ

1) การรับส่งข้อมูลแบบขนาน (Parallel)

2) การรับส่งข้อมูลแบบอนุกรม (Serial)

การรับส่งข้อมูลแบบขนานจะเป็นการับส่งข้อมูลจำนวน 1 ไบต์ ออกไปทางพอร์ทในเวลาเดียวกันในระบบคอมพิวเตอร์ 1ไบต์จะมีจำนวน 8 บิต คือ D_0-D_7 ถ้ามีการส่งข้อมูลแบบขนานจะใช้สายสัญญาณอย่างน้อง 9 เส้น คือสาย Data 8 เส้น และสายกราวด์ 1 เส้นดังรูป



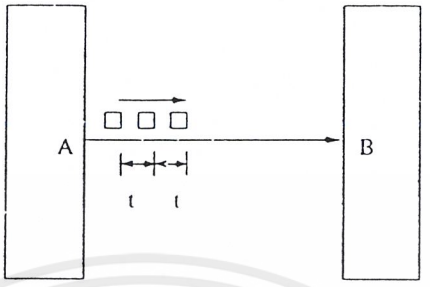
รูปที่ 3.1 การรับส่งข้อมูลแบบขนาน

การรับส่งข้อมูลแบบอนุกรม คือการรับส่งข้อมูลทีละบิต จนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์ คือ D_0-D_7 อาจจะส่งบิต D_0 ออกไปก่อนแล้วตามด้วย D_1 ไปเรื่อย ๆ จนถึง D_7 การส่งข้อมูลทั้งสองแบบมีข้อดีข้อเสียแตกต่างกันคือการส่งข้อมูลแบบขนาน สามารถส่งข้อมูลได้เร็วคือส่งทีเดียวจะได้ข้อมูลครบ 1 ไบต์ แต่ถ้าต้องส่งเป็นระยะไกล ๆ จะสิ้นเปลืองสายสัญญาณมาก ถ้าเป็นการส่งแบบอนุกรม เมื่อต้องการส่งข้อมูลเป็นระยะไกล ๆ จะช่วยประหยัดสายสัญญาณเนื่องจากจะใช้สายอย่างน้องเพียง 2 เส้น คือสายสัญญาณกับสายกราวด์ แต่การรับส่งข้อมูลจะใช้เวลานานเนื่องจากเป็นการส่งทีละบิต ในที่นี้จะกล่าวถึงพื้นฐานการรับส่งข้อมูลแบบอนุกรมโดยใช้ MCS-51

3.1.1 การรับส่งข้อมูลแบบซิงโครนัส (Synchromous Input / Output)

การรับส่งข้อมูลแบบนี้ไม่ว่าจะเป็นการส่งแบบอนุกรมหรือขนาน ข้อมูลแต่ละไบต์ที่ถูกส่งออกไปจะมีช่วงเวลาห่างกันแน่นอน เช่นการส่งข้อมูลจาก A ไป B ดังรูปที่ 3.2 Data 1 จะห่าง

จาก Data 2 เป็นเวลา t และ Data 3 จะห่างจาก Data 2 เป็นเวลา t เช่นกัน ระบบนี้เหมาะกับงานที่ไม่มีความยุ่งยากมาก



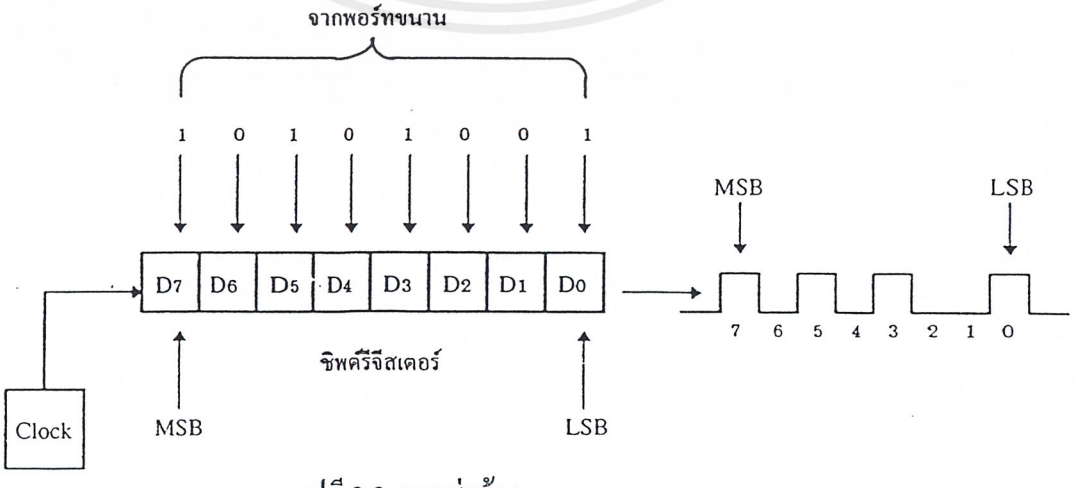
รูปที่ 3.2 การส่งข้อมูลแบบซิงโครนัส

3.1.2 การรับส่งข้อมูลแบบอะซิงโครนัส (Asynchromous Input / Output)

การรับส่งข้อมูลแบบนี้ข้อมูลที่ส่งออกไปจะไม่มีเวลาที่แน่นอน ซึ่งจะขึ้นอยู่กับความพร้อมของผู้ส่งและผู้รับ โดยจะมีสายสัญญาณตรวจสอบความพร้อมของระบบทั่งสองว่าพร้อมที่จะติดต่อกันหรือไม่ โดยสัญญาณที่เพิ่มขึ้นมาจากระบบแบบ ซิงโครนัสเรียกว่า สายสเตตัส (Status Line)

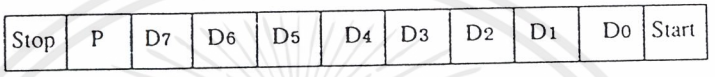
3.2 รูปแบบของการรังส่งข้อมูลแบบอนุกรม

เมื่อไมโครคอมพิวเตอร์ต้องการจะส่งข้อมูลแบบอนุกรม ตัวไมโครคอมพิวเตอร์จะส่งข้อมูลออกไปทางพอร์ทซึ่งเป็นพอร์ทแบบขนานก่อน จากนั่นจะมีอุปกรณ์มาต่อที่พอร์ท เพื่อแปลงข้อมูลแบบขนานให้เป็นแบบอนุกรมอีกทีหนึ่ง (Parallel - to – serial Conversion) ตัวแปลงข้อมูลนี้อาจพิจารณาง่าย ๆ ว่า เป็น Shift Register ดังรูปที่ 3.3 เมื่อข้อมูลที่จะส่งอยู่ใน Shift Register แล้วตัวสัญญาณนาฬิกาจะเป็นตัวกระตุ้นให้ส่งข้อมูลบิตต่ำออกไปในเวลาแรก จากนั้นจะส่งบิตต่Vไปตามออกมา จากรูปที่ 3.3 จะเป็นการส่งข้อมูล A9H ออกไป



รูปที่ 3.3 การส่งข้อมูลแบบอนุกรม

สำหรับตัวรับข้อมูลแบบอนุกรมเมื่อตัวรับข้อมูลจะเป็นการรับเข้ามาใน Shift Register แล้วส่งข้อมูลให้ไมโครคอมพิวเตอร์แบบขนนอีกทีหนึ่ง (Serial – to – parallel) ระบบคอมพิวเตอร์ในปัจจุบันจะมีตัวแปลง Parallel – to – serial และ Serial – to – parallel อยู่ในชิพไอซี เรียกว่า Universal Asynchronous Receiver Transmitter (UART) การส่งข้อมูลแบบอนุกรมนั้นจะต้องมีการเพิ่มเติมข้อมูลบางอย่างเข้าไปเพื่อให้การรับส่งข้อมูลสามารถทำงานได้ถูกต้องมากขึ้น โดยมีการเติมค่าบิตต่าง ๆ ลงไปตามรูปที่ 3.4



รูปที่ 3.4 บิตต่าง ๆ ของข้อมูลที่ส่งแบบอนุกรม

ถ้ามีการส่งข้อมูลแบบ 8 บิต จะต้องส่งบิตแรกออกไปก่อน เรียกว่า บิตเริ่มต้น (Start Bit) ถ้ามีการส่งข้อมูลหลาย ๆ ไบต์ออกมา บิตนี้จะเป็นตัวบอกว่า มีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้นมักมีระดับ ลอจิกเป็น “ 0” ต่อจากบิตเริ่มต้นจะเป็นข้อมูลบิต D0ถึง D7 จากนั้น จะตามด้วยบิตตรวจสอบความถูกต้อง (Parity Bit) ถ้าข้อมูล 8 บิตที่ส่งออกมา จำนวนของบิตที่มีค่าเป็น “1” เป็นจำนวนคู่ บิตนี้จะมีค่าเป็น 0 แต่ถ้าจำนวนของบิตที่มีค่าเป็น “1” เป็นคี่ บิตนี้จะมีค่าเป็น “1” จากนั้นข้อมูลที่ส่งออกไปจะตามด้วยบิตสิ้นสุดข้อมูล (Stop Bit) เพื่อเป็นการบอกว่าข้อมูลที่ส่งมา 8 บิตนั้นหมดแล้ว ตัวบิต Stop อาจมีจำนวนมากกว่า 1 บิต ก็ได้ เช่น 1.5 บิต,2 บิต

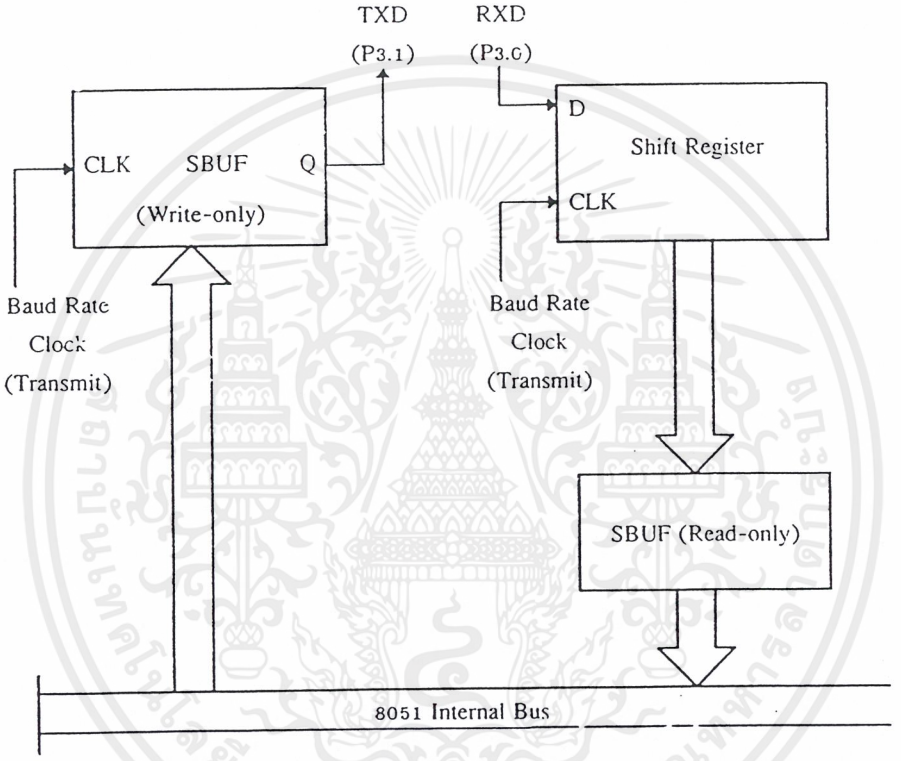
การส่งข้อมูลแบบอนุกรมนั้นความเร็วของการส่งจะมีค่าเป็นบิตต่อวินาที เรียกว่า “Baud Rate”

3.3 MCS – 51 กับการรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบอนุกรมกับไมโครคอนโทรลเลอร์ MCS-51 นั้น ภายในชิพ MCS – 51 จะมี UART อยู่ในตัว ซึ่งเป็นข้อดีของไมโครคอนโทรลเลอร์ถ้าเป็นไมโครโปรเซสเซอร์ เช่น เบอร์ Z – 80 ถ้าต้องการรับส่งข้อมูลแบบอนุกรมจะต้องนำชิพ UART มาประกอบด้วย

พอร์ทอนุกรมของ MCS – 51 จะใช้ขา TXD และ RXD ในการรับส่งข้อมูลโดยทั้งสองจะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TXD และ P3.0 หรือขา 10 เป็น RXD อนุกรมของ MCS – 51 สามารทำงานแบบ Full Duplex ได้คือสามารถส่งและรับข้อมูลในเวลาเดียวกันได้ โดยในการรับและส่งข้อมูลจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

รีจีสเตอร์ที่สำคัญในการรับส่งข้อมูลคือ SBUF และ SCPN ซึ่งเป็นรีจีสเตอร์ที่อยู่ใน Special Function Registers โดยรีจีสเตอร์ Serial Port Buffer (SBUF) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลไปที่ตำแหน่งนี้ จะเป็นการส่งข้อมูลออกทางพอร์ทอนุกรม และถ้าอ่านข้อมูลจากตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ทอนุกรมโดยใน SBUF จะประกอบด้วยบัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูลดังรูปที่ 3.5



รูปที่ 3.5 การรับส่งข้อมูลระหว่างรีจีสเตอร์กับบัสภายใน

สำหรับ Serial Port Control Register (SCON) ซึ่งอยู่ที่ตำแหน่ง 98H จะเป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ รีจีสเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่าง ๆ ของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล (Baud Rate) สามารถหาได้จากการหารสัญญาณนาฬิกาที่ใช้กับ MCS – 51

3.4 Serial Port Control Register

MCS –51 มีโหมดการทำงานของพอร์ทอนุกรมหลายโหมด ซึ่งสามารถโปรแกรมโหมดการทำงานได้โดยเขียนข้อมูลไปยังรีจีสเตอร์ SCON ความหมายของแต่ละบิตแสดงดังตารางที่ 3.1 และ 3.2

ตารางที่ 3.1 บิตต่าง ๆ ของรีจีสเตอร์ SCON

บิต	ชื่อ	ตำแหน่ง	ความหมาย
SCON.7	SM0	9FH	บิตเลือกโหมดการทำงานบิต 0
SCON.6	SM1	9EH	บิตเลือกโหมดการทำงานบิต 1
SCON.5	SM2	9DH	บิตเลือกโหมดการทำงานบิต 2
SCON.4	REN	9CH	บิตแฟลกกำหนดยอมให้มีการรับข้อมูล
SCON.3	TB8	9BH	ค่าของบิต 9 สำหรับการส่งข้อมูลในโหมด 2 และ 3 สามารถ Set และ Clear ได้โดย Software
SCON.2	RB8	9AH	ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา
SCON.1	TI	99H	บิตแฟลกแสดงการอินเทอร์รัพท์ภายหลังการส่งข้อมูลออกไปโดยจะ Set เมื่อส่งข้อมูลออกไปหมดแล้วและสามารถ Clear ได้ด้วย Software
SCON.0	RI	98H	แฟลกแสดงการอินเทอร์รัพท์ภายหลังรับข้อมูลเข้ามาสามารถ Clear ได้ด้วย Software

ตารางที่ 3.2 แสดงโหมดต่าง ๆ ของการรับส่งแบบอนุกรม

SM0	SM1	MODE	ความหมาย	BAUD TATE
0	0	0	Shift Register	เปลี่ยนแปลงไม่ได้ (Oscjllator Frequency ÷ 12)
0	1	1	8-bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจาก Timer
1	0	2	9-bit UART	เปลี่ยนแปลงไม่ได้ (Oscillator Frequency ÷ 12 หรือ ÷ 64)
1	1	3	9-bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจาก Timer

ก่อนที่จะใช้พอร์ทอนุกรมจะต้องโปรแกรมให้กับ SCON เสียก่อนเพื่อกำหนดโหมดการทำงานและลักษณะต่าง ๆ เช่น

MOV SCON,#01010010 B

เป็นการกำหนดให้พอร์ทอนุกรมทำงานในโหมด 1 และอีนาเบิลให้มีการรับข้อมูลพร้อมกับกำหนดให้ TI เป็น 1

ในการส่งข้อมูลทุกโหมดสามารถทำได้โดยเขียนข้อมูลไปยัง SBUF เมื่อข้อมูลถูกส่งไปแล้วบิต TI จะถูกเซตเป็น "1" ในการส่งข้อมูลจะต้องคอยตวจสอบบิต TI เพราะว่าถ้า TI ยังไม่เป็น "1" แสดงว่า ข้อมูลยังส่งไปไม่หมด ถ้าหากมีการเขียนข้อมูลไบต์ต่อไปยังง SBUF จะทำให้เกิดข้อผิดพลาดขึ้นสำหรับในการรับข้อมูลบิต REN จะต้องเซตให้เป็น "1" ยกเว้นโหมด 0 เพื่ออนุญาตให้รับข้อมูลได้ เมื่อข้อมูลรับเข้ามาเรียบร้อยแล้ว บิต RI จะถูกเซตเป็น "1"

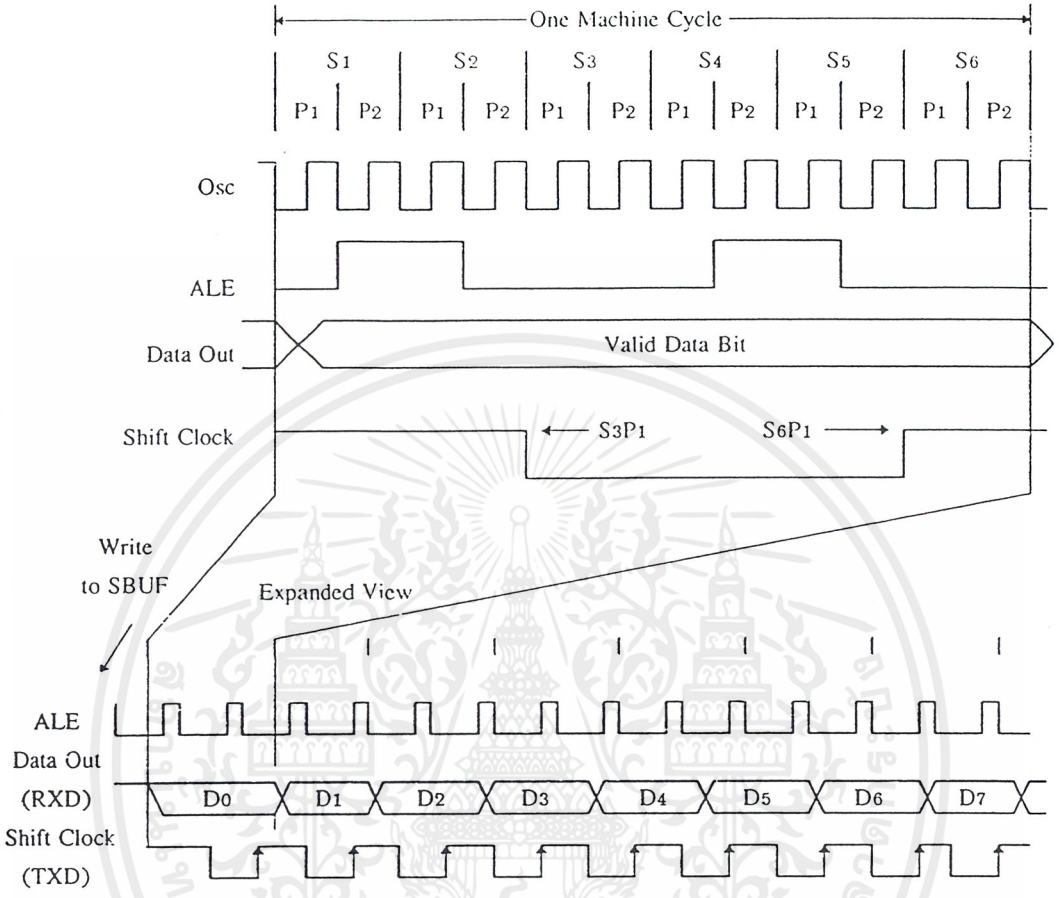
3.5 Mode Of Operation

ใน MCS –51 การสื่อสารทางพอร์ทอนุกรมจะมีอยู่ 4 ประเภท หรือ 4 โหมด ซึ่งจะกำหนดได้ที่บิต SM0 และ SM1 ใน SCON โดยจะมี 3 โหมดเป็นการสื่อสารแบบAsynchronous โดยลักษณะของข้อมูลที่ส่งจะมีบิตเริ่มต้น (Start Bit) และบิตจบ (Stop Bit) คล้ายกับการสื่อสารแบบ RS – 232 ในระบบคอมพิวเตอร์ อีกโหมดหนึ่งจะเป็นการใช้พอร์ทอนุกรมในลักษณะชิพต์รีจีสเตอร์

3.5.1 8 – Bit Shift Register (Mode 0)

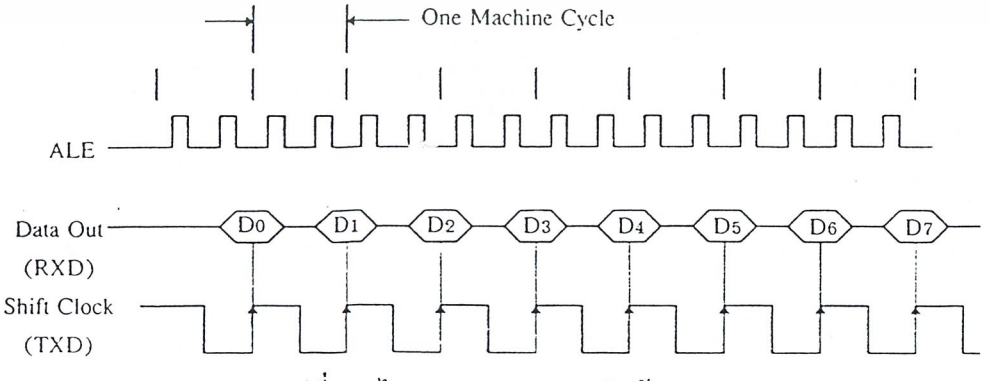
การทำงานในโหมดนี้จะใช้ขา RXD ในการรับส่งข้อมูลโดยต่อกับ Shift Register ภายนอก ส่วนขา TXD จะเป็น Output Shift Clock เพื่อกระตุ้นรีจีเตอร์ภายนอกให้เลื่อนบิตถ้ามีการส่งข้อมูลหรือรับข้อมูล 8 บิต จะเริ่มที่บิตต่ำสุดก่อน โดยมีค่า Baud Rate เท่ากัน 1/12 ของความถี่ที่ใช้บนชิพ

ในการส่งข้อมูลจะทำโดยเขียนข้อมูลไปที่รีจีสเตอร์ SBUF ข้อมูลจะถูกส่งออกมาทางขา RXD (P3.0) โดยจะสอดคล้องกับสัญญาณที่ออกมาทางขา TXD ซึ่งสัญญาณของขา TXD จะถูกส่งออกมาทุก ๆ Machine Cycle โดยจะเป็นลอจิก "0" ใน S3P1 และจะกลับเป็นลอจิก "1" ใน S6P1 ซึ่งแสดงได้ดังรูปที่ 3.6



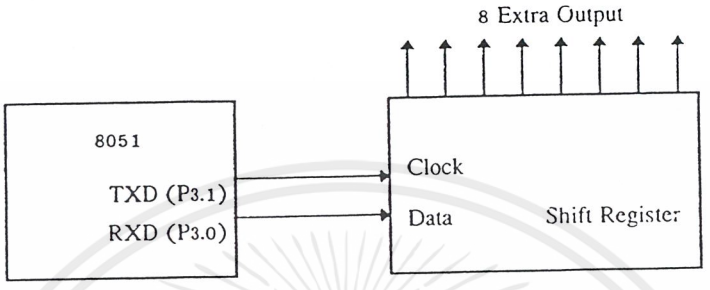
รูปที่ 3.6 ไดอะแกรมเวลาการส่งข้อมูล

สำหรับการรับข้อมูลจะรับได้เมื่อเซตขา Receiver Enable Bit (REN) เป็น "1" และเคลียร์ขา Receiver Interrupt Bit (RI) เป็น "0" ข้อมูลจะเข้าสู่ MCS – 51 เมื่อ Clock Shift ถูกส่งออกไปทาง TXD ที่ขอบขาขึ้นของ Clock Shift บิตต่ำจะถูกส่งเข้ามาก่อนดังรูปที่ 3.7



รูปที่ 3.7 ไดอะแกรมเวลาการรับข้อมูล

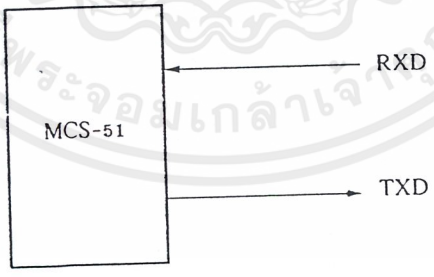
ในการประยุกต์ใช้งานโหมดนี้จะต้องมีไอซีชิพต์รีจีสเตอร์มาต่อภายนอก เช่น ถ้าหากต้องการส่งข้อมูลออกมาทางพอร์ทอนุกรม อาจต่อวงจรได้ดังรูป 3.8 โดยใช้ไอซี Serial – to – Parallel Shift Register โดยข้อมูลส่งออกมาทาง RXD และใช้ TXD เป็น Clock



รูปที่ 3.8 การส่งข้อมูลออกโดยใช้ชิพต์รีจีสเตอร์ช่วย

3.5.2 8 – Bit UART with Variable Baud Rate (Mode 1)

ในโหมดนี้จะเป็นการรับส่งข้อมูลแบบ 10 บิตซึ่งประกอบด้วยบิตเริ่มต้น (เป็น“0”) ข้อมูล 8 บิตและจบ(เป็น “1”) นอกจากนี้ยังสามารถกำหนดค่า Baud Rate ได้โดยค่า Baud Rate นี้จะแปรตามตัวจับเวลาตัวที่ 1 ในโหมดนี้ จะส่งข้อมูลออกทาง TXD และรับข้อมูลเข้าทาง RXD ถ้าเป็นการรับข้อมูลเข้าตัว Stop Bit จะเข้ามายังบิต RB8 ใน SCON



รูปที่ 3.9 การรับส่งข้อมูลในโหมด 1

ค่า Baud Rate ที่ใช้ในการรับส่งข้อมูลจะกำหนดโดย Timer 1 หลังจากโปรแกรมไปใน Timer 1 แล้วสามารถเลือกค่า Baud Tate ได้อีกสองค่าคือ ค่าจาก Timer 1 Overflow หาร 32 กับค่าจาก Timer 1 Overflow หาร 16

การส่งข้อมูลทำได้โดยการเขียนข้อมูล 8 บิตไปที่ SBUF โดยบิตที่ 9 (Stop Bit) ให้เขียนลงใน TB8 ใน SCLON จากนั้นข้อมูลจะถูกส่งออกมาทางขา TXD โดยส่ง Start Bit ออกมาก่อน ตามด้วยข้อมูล 8 บิต และจบด้วย Stop Bit เมื่อข้อมูลถูกส่งออกไปหมดแล้วบิต Interrupt Flag (TI) จะเป็น "1" ดังนั้นในการเขียนข้อมูลใหม่ลงไปจะต้องตรวจสอบบิตนี้

ในการรับข้อมูลจะเริ่มจากเมื่อมีการเปลี่ยนแปลงลอจิกจาก 1 เป็น 0 ทาง RXD หมายความว่า เริ่มรับบิตเริ่มต้น จากนั้นข้อมูลอีก 8 บิตจะถูกเก็บลงใน SBUF และ Stop Bit จะถูกเก็บในบิต RB8 ของรีจีสเตอร์ SCON เมื่อข้อมูลเข้ามาครบแล้วบิต Imterrupt Flag (RI) จะถูกเซต ดังนั้นในกรอ่านข้อมูลจะอ่านได้เมื่อบิต RI ถูกเซตแล้ว เมื่ออ่านข้อมูลไปแล้วจะต้องเคลียร์บิตนี้

3.5.3 9 – Bit UART with Fixed Baud Rate (Mode 2)

การทำงานในโหมดนี้ไม่สามารถกำหนดค่า Baud Rate ได้ ซึ่งค่า Baud Rate จะมีสองค่าคือ 1/64 และ 1/32 ของสัญญาณนาฬิกาบนชิพ การรับส่งข้อมูลจะเป็นชุดข้อมูล 9 บิต บิตเริ่มต้น บิตหยุด รวมเป็น 11 บิต โดยข้อมูล 9 บิตจะเป็นจำนวนข้อมูล 8 บิต และบิตที่โปรแกรมได้อีก 1 บิต โดยบิตนี้จะเป็นบิตที่ 9 ซึ่งจะใช้เป็น Parity บิต ในการส่งข้อมูลจะต้องเขียนไปที่ บิต TB8 ในรีจีสเตอร์ SCON สำหรับการรับข้อมูลบิตที่ 9 ถูกเก็บในบิต RB8

3.5.4 9 – Bit UART with Variable Baud Rate

การทำงานในโหมดนี้จะคล้ายกับโหมด 2 แต่สามารถกำหนดค่า Baud Rate ได้โดยการโปรแกรมไปที่ Timer 1 หลังจากโปรแกรมแล้วยังสามารถเลือกได้อีก 2 ค่าคือ ความถี่การ Overflow ของ Timer 1 หารด้วย 16 และหารด้วย 32

3.6 การกำหนดค่าเริ่มต้นให้รีจีสเตอร์ในการรับส่งข้อมูล

การรับข้อมูล ถ้าจะให้ MCS – 51 รับข้อมูลทางพอร์ทอนุกรมจะต้องโปรแกรมไปที่บิต Receiver Enable (REN) ในรีจีสเตอร์ SCON ให้เป็นลอจิก "1" ซึ่งอาจทำได้สองวิธีดังนี้

```
SETB   REN
```

เป็นการเซตบิต REN ให้เป็น "1" หรืออาจทำโดยใช้คำสั่ง

```
MOV  SCON , #xxx1xxxxB
```

ซึ่งเป็นการย้ายข้อมูลที่ทำให้บิต REN เป็น 1 สำหรับค่า x หมายความว่าเป็นอะไรก็ได้ขึ้นกับการใช้งานในโหมดต่าง ๆ

ข้อมูลแบบ 9 บิต ในการรับส่งข้อมูลที่มีบิตข้อมูลแบบ 9 บิต ได้แก่ การใช้งานในโหมด 2 และ โหมด 3 การส่งข้อมูลบิตที่ 9 จะถูกเขียนในบิต TB8 โดยการเขียนโปรแกรม สำหรับการรับข้อมูลเมื่อข้อมูลเข้ามาถึงบิตที่ 9 จะถูกเขียนลงในบิต RB8

การเพิ่มบิต Parity การส่งข้อมูลแบบ 9 บิต สามารถใช้บิตที่ 9 เป็นบิต Parity ได้ซึ่งบิต Parity จะอยู่ใน Program Status Word (PSW) โดยจะถูกเซตหรือเคลียร์ทุก ๆ แมชชีนไซเคิลที่เกี่ยวข้องกับ Accumulator เช่น ถ้าจะส่งข้อมูลแบบ 8 บิต ตามด้วยบิต Even Parity เป็นบิตที่ 9 สามารถเขียนโปรแกรมได้ดังนี้

```
MOV  C,P          ; อ่านค่าบิต P มาเก็บใน C
MOV  TB8 , C      ; นำค่าบิต Parity เขียนลงใน TB8
MOV  SBUF , A     ; ส่งข้อมูลไปทางพอร์ทอนุกรม
```

ถ้าเป็นแบบ Odd Parity ให้แก้ไขข้อมูลที่อ่านได้จากบิต Parity เสียก่อนที่จะส่งออกไป ซึ่งเขียนโปรแกรมได้ดังนี้

```
MOV  C,P          ; อ่านค่าบิต P มาเก็บใน C
CPL  C            ; กลับค่าให้เป็น Odd Parity
MOV  TB8,C        ; เขียนค่าลงใน TB8
MOV  SBUF,A       ; ส่งข้อมูลออกไปทางพอร์ทอนุกรม
```

การส่งข้อมูลแบบมี Parity บิตด้วยไม่ใช่จะส่งได้แบบ 9 บิตหรือโหมด 2 และ 3 เท่านั้น ในโหมด 1 ซึ่งส่งข้อมูลแบบ 8 บิตก็สามารถทำได้ อย่างเช่นการส่งรหัส ASCII จะใช้บิตข้อมูล 7 บิต สำหรับบิตที่เหลืออีหนึ่งบิตจะเป็นบิต Parity รวมเป็น 8 บิตซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
CLR  ACC.7        ; เคลียร์ค่าบิต 7 เพื่อใช้เป็น Parity บิต
MOV  C , P        ; นำบิต Parity มาเก็บใน C
MOV  ACC.7 , C    ; เขียนค่าบิต Parity ลงในรีจีสเตอร์ A
```

```
MOV SBUF , A          ; ส่งข้อมูลออกไปทางพอร์ทอนุกรม
```

แฟลกอินเทอร์รัพท์ เมื่อมีการรับส่งข้อมูลเสร็จสิ้นจะมีผลต่อแฟลกอินเทอร์รัพท์ (RI และ TI) ในรีจีสเตอร์ SCON ซึ่งบิตหล่านี้จะถูกเซตโดย Hardware แต้องเคลียร์ด้วย Software

บิต RI ถ้าถูกเซตหมายความว่าบัฟเฟอร์ที่ใช้รับข้อมูลเต็มให้อ่านไปได้แล้ว และบิตนี้สามารถใช้อินเทอร์รัพท์ MCS –51 แต่ถ้าเขียนโปรแกรมจะใช้วิธีตรวจเช็กบิตนี้ ถ้าเป็น "1" หมายความว่าให้อ่านข้อมูลมาเก็บใน รีจีสเตอร์ A ได้แต่ก่อนอ่านจะต้องเคลียร์ RI เสียก่อนเพื่อจะได้รับข้อมูลถัดไปได้ซึ่งเขียนโปรแกรมได้ดังนี้

```
WAIT : JNV RI , WAIT      ; ถ้าบิตนี้ไม่เป็น "1" จะทำงานอยู่ที่เดิม
       CLR RI             ; เคลียร์ RI
       MOV A , SBUF       ; อ่านค่ามาเก็บใน A
```

บิต TI เมื่อส่งข้อมูลออกไปแล้วบิตนี้จะถูกเซต เป็นการบอกว่าบัฟเฟอร์ส่งข้อมูลว่างแล้วให้ส่งข้อมูลใหม่เข้าไปได้ ซึ่งสามารถใช้บิตนี้อินเทอร์รัพท์ MSC – 51 ได้เช่นกัน แต่ถ้าเขียนโปรแกรมคอยตรวจเช็กอาจเขียนได้ดังนี้

```
WAIT : JNV TI , WAIT      ; ตรวจบิต TI ว่าเป็น "1" หรือยัง
       CLR TI             ; เคลียร์ TI
       MOV SBUF ,A        ; เขียนข้อมูลลงไป
```

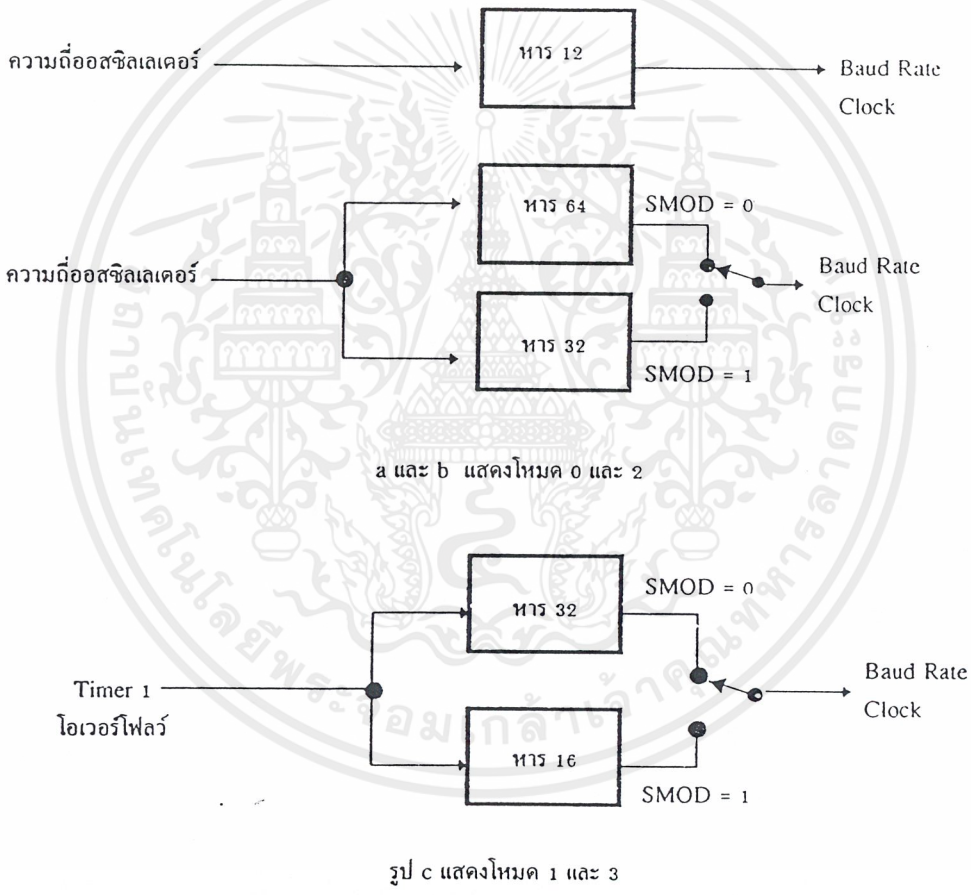
3.7 อัตราการส่งข้อมูลของพอร์ทอนุกรม

จากการศึกษาการรับส่งข้อมูลในโหมดต่าง ๆ พบว่าในโหมด 0 และโหมด 2 ไม่สามารถกำหนด Baud Rate เองได้ โดยในโหมด 0 ค่า Baud Rate จะมีค่าเท่ากับความถี่ของ Oscillator หารด้วย 12 ใน โหมด 1 จะมีสองค่าคือ ความถี่ Oscillator หารด้วย 32 และหารด้วย 64 สองค่านี้เรียกว่า SMOD 0 และ SMOD 1 ซึ่งสามารถกำหนดได้ในรีจีสเตอร์ PXON บิตที่ 7 ในรีจีสเตอร์ PCON นี้ไม่สามารถเข้าถึงข้อมูลระดับบิตได้ การเขียนข้อมูลลงไปทีละบิตจะต้องใช้วิธีที่เรียกว่า "Read-Modify-Write" คืออ่านค่าขึ้นมาแก้ไขแล้วเขียนลงไปใหม่ ตัวอย่างเช่น

```
MOV A, PCON               ; อ่านค่าจาก PCON มาเก็บในรีจีสเตอร์A
```

```
SETB ACC.7          ; เซตบิต 7 (SMOD)
MOV  PXON , A       ; เขียนค่าลงไปใหม่ใน PCON
```

สำหรับโหมด 1 และโหมด 3 สามารถกำหนดค่า Baud Rates ได้โดยการโปรแกรมลงใน Timer 1 ในการโปรแกรมแต่ละครั้งจะมี SMOD สองค่าเช่นกัน ค่า Baud Tates ของโหมดต่าง ๆ แสดงได้ดังรูปที่ 3.10



รูปที่ 3.10 แสดงการกำหนด Baud Rate ในโหมดต่าง ๆ

การใช้ Timer 1 กำหนด Baud Tate Clock

การกำหนดค่าลงใน Timer 1 ทำได้โดยการโปรแกรมไปที่ TMOD ให้ทำงานแบบ 8 – bit Auto Reload Mode (โหมด 2) โดยเขียนค่าไปที่ TH 1 ซึ่งโปรแกรมที่รีจีสเตอร์ TMOD ได้ดังนี้

MOV TMOD , #0010xxxxB

ค่า x หมายความว่าเป็นอะไรก็ได้เพราะบิตเหล่านี้ใช้ใน Timer 0

ถ้าต้องการ Baud Rate ต่ำ ๆ สามารถใช้ 16 – bit Mode ได้ โดยโปรแกรมเป็น TMOD = 0001xxxxB ค่า Baud Rate ที่ส่งออกมาจะมีค่าเท่ากับ ความถี่ของ Timer 1 เกิด Overflow หารด้วย 32 (หรือหารด้วย 16 ถ้าเป็น SMOD =1)

รูปแบบทั่วไปของการหาค่า Baud Rate ในโหมด 1 และ 3 สามารถหาได้ดังนี้

BAUD RATE = TIMER 1 OVERFLOW RATE /32

ถ้าต้องการค่า Baud Rate เท่ากับ 1,200 สามารถคำนวณค่าความถี่ Overflow ของ Timer 1 ได้ดังนี้

1,200 = Timer 1Overflow Rate / 32

จะได้ Timer 1 Overflow Rate เท่ากับ 38.4 kHz

ถ้าระบบ MCS –51 ใช้ความถี่สัญญาณนาฬิกาจาก Crystal เท่ากับ 12 MHz ตัว Timer 1 จะได้รับ Clock เท่ากับ 1 MHz หรือ 1,000 kHz ถ้าเราต้องการ Timer 1 Overflow เท่ากับ 38.4 kHz ดังนั้นค่าอัตรา Overflow มีค่าเท่ากับ 1,000/38.4 = 26.04 Clock โดยค่า Overflow จะเกิดเมื่อเกิดการเปลี่ยนจาก FFH เป็น 00H ดังนั้นจะต้องให้ Timer 1 นับไป 26 Count ดังนั้นค่าที่จะให้รีจีสเตอร์ TH 1 มีค่าเท่ากับ –26 ซึ่งเป็นค่า Reload ดังนั้นเขียนคำสั่งได้ดังนี้

MOV TH1 , #-26

ตัวโปรแกรมแอสเซมเบอร์ทั่วไปจะแปลงค่า –26 เป็น 0E6H เอง จากที่ผ่านมาจะเห็นว่าความถี่ Vaud Rate จะมีความสัมพันธ์กับค่าสัญญาณนาฬิกาที่ใช้จาก Crystal ในตารางที่ 3.3 จะเป็นค่าที่ต้องกำหนดใน Timer 1 เมื่อต้องการค่า Baud Rate ต่าง ๆ

ตารางที่ 3.3 แสดงความถี่สัญญาณนาฬิกาใช้กำหนด Baud Rate ค่าต่าง ๆ

ค่า Baud Rate	Crystal	SMOD โหมด	ค่าใน TH1	ค่า BaudRate ที่ได้	ผิดพลาด
9,600	12.000	1	-7(F9H)	8,923	7%
2,400	12.000	0	-13(F3H)	2,404	0.16%
1,200	12.000	0	-16(E6H)	1,202	0.16%
19,200	11.059	1	-3(FDH)	19,200	0
9,600	11.059	0	-3(FDH)	9,600	0
2,400	11.059	0	-12(F4H)	2,400	0
1,200	11.059	0	-24(E8H)	1,200	0

บทที่ 4
ทฤษฎีสเตปปิ้งมอเตอร์ และวงจรขับ

สเตปมอเตอร์ถูกนำมาใช้งานหลายประเภทในปัจจุบัน โดยเฉพาะงานที่ต้องใช้ไมโครโปรเซสเซอร์ในการควบคุม เนื่องจากง่ายต่อการควบคุม โดยการใช้โปรแกรมเพียงอย่างเดียวก็สามารถควบคุมการหมุนของสเตปมอเตอร์ได้ ไม่ต้องใช้วงจรที่ซับซ้อน และเป็นการควบคุมแบบลูปเปิด ทำให้ประหยัดค่าใช้จ่ายกว่าการใช้มอเตอร์กระแสตรง ซึ่งเป็นการควบคุมแบบลูปปิดที่ต้องใช้อุปกรณ์ในการควบคุมมากกว่า

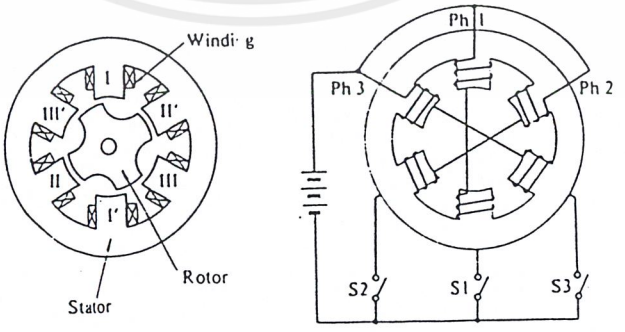
อย่างไรก็ตามมอเตอร์กระแสตรงจะมี แรงบิดที่สูงกว่า และมีการหมุนที่ราบเรียบกว่าสเตปมอเตอร์ ในการเลือกมอเตอร์เพื่อใช้งาน จึงควรพิจารณาความเหมาะสมในการใช้งานของมอเตอร์แต่ละชนิด

4.1 ชนิดของ Stepping Motor

สเตปมอเตอร์สามารถแบ่งออกได้เป็น 3 ประเภทตามโครงสร้าง ดังนี้

4.1.1 Variable Reluctance

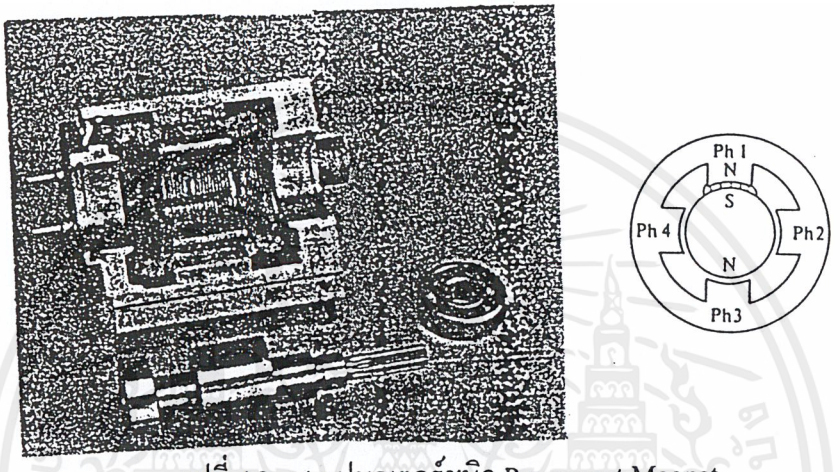
เป็นสเตปมอเตอร์ที่มีลักษณะโครงสร้างและการใช้งานที่เป็นพื้นฐานของแบบอื่นๆ ในรูปที่ 3.1 เป็นVR motor แบบ 3 เฟส โดยที่แต่ละฟันของสเตเตอร์จะถูกพันด้วยขดลวดและขดตรงกันข้ามจะถูกต่อกันแบบอนุกรมหรือขนาน ส่วนโรเตอร์จะเป็นฟันยื่นออกมา ทั้งสเตเตอร์โรเตอร์ทำจากเหล็กผสมซิลิคอน ซึ่งเป็นวัสดุที่มีค่าความซึมซับสูง (high permiable) ให้เส้นแรงแม่เหล็กผ่านได้มาก มีความเฉื่อยต่ำกว่าแบบอื่น ทำให้มีการตอบสนองที่เร็วกว่าแบบอื่น เนื่องจากมอเตอร์ไม่มีส่วนของแม่เหล็ก ขณะที่ไม่ถูกกระตุ้นจะสามารถหมุนได้อิสระ



รูปที่ 4.1 สเตปมอเตอร์ชนิด Variable Reluctance

4.1.2 Permanent Magnet

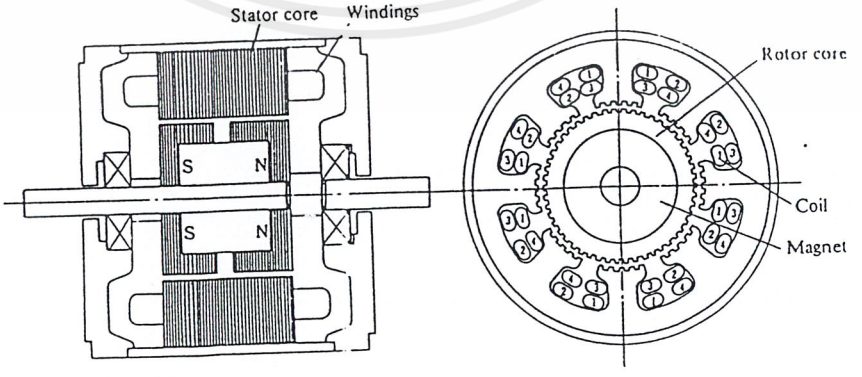
เป็นสเตปมอเตอร์ที่ใช้แม่เหล็กถาวรเป็นโรเตอร์ ทำให้มี holding torque ขณะยังไม่หมุนแต่ละฟันของสเตเตอร์จะถูกพันด้วยขดลวดทำให้เกิดเป็นขั้วแม่เหล็กเหนือและใต้สลับกัน บางครั้งอาจจะเพิ่มแม่เหล็กถาวรเข้าในส่วนของสเตเตอร์เพื่อเพิ่มสนามแม่เหล็ก ทำให้มีแรงบิดสูงขึ้น สเตปมอเตอร์ชนิดนี้จะใช้พลังงานต่ำกว่า และมีการตอบสนองที่ดีกว่าชนิดอื่น โดยทั่วไปมอเตอร์ชนิดนี้จะมีความกว้างของมุมสเตปขนาด 1.8,7.5,15,30 และ 90



รูปที่ 4.2 สเตปมกเตอร์ชนิด Permanent Magnet

4.1.3 Hybrid

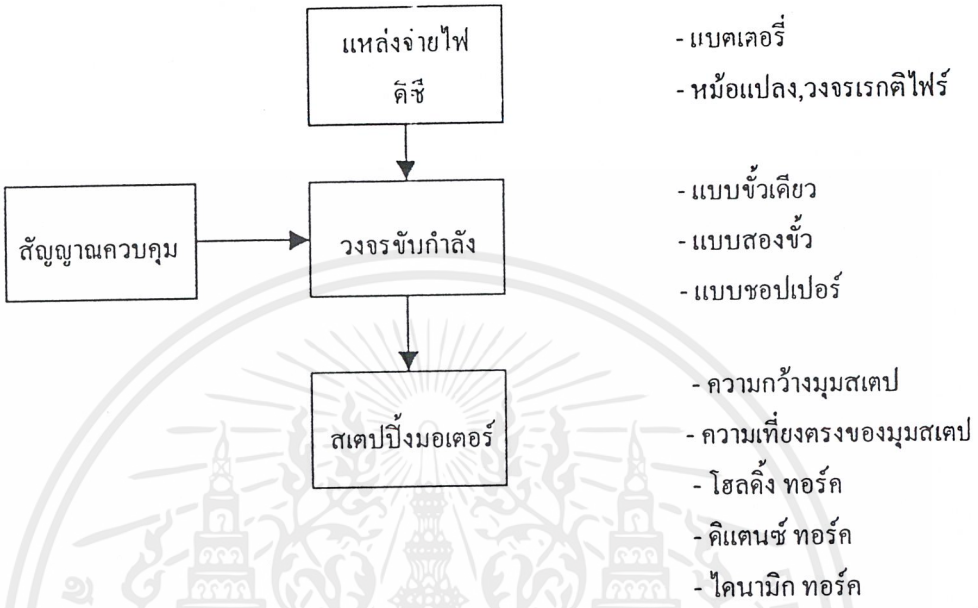
เป็นสเตปมอเตอร์ที่มีลักษณะโครงสร้างรวมกันระหว่างแบบ VR motor และแบบแม่เหล็กถาวร โดยมีโรเตอร์เป็นแม่เหล็กถาวร แต่มีลักษณะการทำงานเหมือนกับ VR motor ดังในรูปที่ 4.17 ข้อแตกต่างจาก VR motor คือใน VR motorจะมีขดลวดบนขั้วของสเตเตอร์เพียง 1 ขด แต่ในไฮบริดจะมีขดลวด 2 ขดในขั้วของ สเตเตอร์เดียว



รูปที่ 4.3 เสตปมอเตอร์แบบ Hybrid

4.2 คุณลักษณะการทำงานของสเตปปิ้ง

การเลือกใช้สเตปปิ้งมอเตอร์ เพื่อนำไปใช้ประโยชน์นั้น จะต้องมีความเข้าใจคุณลักษณะของมอเตอร์และวงจรที่ใช้ขับมอเตอร์ ในรูปที่ 4.4 เป็นแผนผังระบบสเตปปิ้งมอเตอร์



รูปที่ 4.4 แผนผังระบบสเตปปิ้งมอเตอร์

ความถูกต้องของมุมสเตปขณะไม่มีโหลด จะถูกระบุสำหรับมอเตอร์แต่ละชนิด เช่น มอเตอร์ที่มีสเตป 7.5° ความผิดพลาด ±10 ลิปดา ขณะเคลื่อนที่ไปหนึ่งสเตป เป็นต้น มอเตอร์ที่มีจำนวนสเตปต่อรอบเท่ากับ 4 จะมีค่าความผิดพลาดเป็นศูนย์ เมื่อหมุนครบ 1 รอบ เพราะขณะที่หมุนมา ณ ตำแหน่ง เดิมขณะเริ่มต้นขั้วแม่เหล็กและทิศทางของเส้นแม่เหล็ก (flux) วงเดิม ด้วยเหตุนี้การเปลี่ยนแปลงของสเตปปิ้งมอเตอร์ที่ต้องการความถูกต้องสูง จะต้องแบ่งจำนวนสเตปต่อรอบเป็นจำนวนเท่าของสี่สเตป เพื่อลดการสะสมของค่าความผิดพลาด (step angle error) ซึ่งเป็นรูปแบบการทำงานแบบสี่สเตป ตัวอย่างของมุมสเตปแสดงดังตาราง

มุมสเตป	จำนวนสเตปต่อรอบ
0.9°	400
1.8°	200
3.6°	100
3.75°	96
7.5°	48

ตารางที่ 4.1 ตัวอย่างของมุมสเตป

โดยทั่วไปแล้วมอเตอร์ที่มีความกว้างของมุมสเตปน้อย จะมีความแม่นยำดีกว่าที่มีความกว้างของมุมสเตปมาก

4.2.1 แรงบิด (torque)

การทำงานของสเตปปิ้งมอเตอร์มีแรงบิดเกี่ยวข้องอยู่ 3 ชนิด คือ

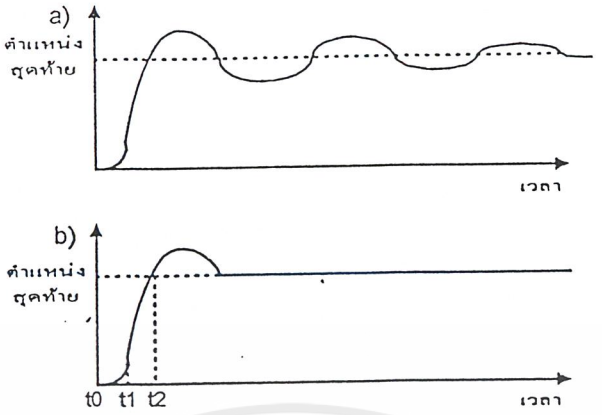
- โฮลดิ้งทอร์ค (Holding Torque) คือ แรงบิดที่ทำให้สเตปปิ้งมอเตอร์ เริ่มหมุนไป สองสเตปขณะหยุดนิ่งถ้าแรงบิดที่ทำให้สเตปปิ้งมอเตอร์มีขนาดมากกว่าโฮลดิ้งทอร์ค จะทำให้มอเตอร์สูญเสียการหมุนแบบต่อเนื่อง โดยปกติแรงบิดขณะการทำงานของมอเตอร์ จะน้อยกว่าระดับโฮลดิ่ง

- ดิเตนซ์ทอร์ค (Detent Torque) ซึ่งเป็นสเตปปิ้งมอเตอร์แบบไฮบริด และแม่เหล็กถาวรจะมีส่วนประกอบของมอเตอร์เป็นแม่เหล็กถาวร ซึ่งจะสร้างแรงบิดมาเบรคการหมุนของมอเตอร์อย่างสม่ำเสมอ ในขณะที่ไม่มีการป้อนกระแสเข้าขดสเตเตอร์ แรงบิดดังกล่าวนี้เรียกว่า ดิเตนซ์ทอร์ค

- ไดนามิกทอร์ค (Dynamic torque) คือแรงบิดขณะทำงานซึ่งอาจเกิดการเปลี่ยนแปลงได้เนื่องมาจากการปรับเปลี่ยนอัตราเร็วของมอเตอร์ โดยปกติการเปลี่ยนอัตราเร็วของมอเตอร์จะอยู่ในย่านระหว่าง เส้นโค้งพูลอิน (Pull-in curve) และเส้นโค้งพูลเอ้าท์ (Pull-out curve) เพราะถ้าปรับอัตราเร็ว ณ จุดนอกโค้งพูลเอ้าท์มอเตอร์จะเสียการหมุนแบบเป็นสเตปได้ หรือเกิดการหมุนแบบต่อเนื่องนั่นเอง

4.2.2 การแกว่งเข้าสู่สภาวะคงตัว (Over shoot)

ขณะที่มอเตอร์หมุนไปในแต่ละสเตป และหยุด ณ สเตปใด ๆ จะเกิดการแกว่งหรือสั่นของโรเตอร์ เข้าสู่ตำแหน่งสุดท้ายนั้นๆ และใช้เวลาช่วงหนึ่งในการเข้าสู่สภาวะคงตัว แสดงเปรียบเทียบได้ดังรูปที่ 4.16(a) ซึ่งเป็นพฤติกรรมปกติ ของระบบที่ใช้สัญญาณพัลซ์ โดยทั้งนี้ขึ้นอยู่กับโหลด และกำลังงานที่ได้รับจากภาคขับมอเตอร์ ผลตอบสนองที่เกิดขึ้นนี้สามารถเปลี่ยนแปลง คือ ลดเวลาในการเข้าสู่สภาวะคงตัวได้โดยการเพิ่มโหลดที่เป็นแรงเสียดทานเข้าไปในระบบ ซึ่งเป็นลักษณะการแก้ไปทางกล การแก้ไขทางไฟฟ้า ทำได้โดยการหน่วงสัญญาณพัลซ์ลูกสุดท้ายในขบวนพัลซ์ทั้งหมด



รูปที่ 4.5 ผลตอบสนองในการเข้าสู่สภาวะคงตัว (a) ผลตอบสนองเมื่อการหน่วงน้อย (b) ผลตอบสนองเมื่อมีการหน่วงทางไฟฟ้า

4.2.3 เฟส

จำนวนเฟสของสเตปปิ้งมอเตอร์ ขึ้นอยู่กับจำนวนขดลวดบนสเตเตอร์ โดยทั่วไปจำนวนฟันของสเตเตอร์และโรเตอร์จะสัมพันธ์กันโดย

$$N_S = N_R \pm P \tag{4.1}$$

โดยที่ N_S = จำนวนซี่ฟันบนสเตเตอร์

N_R = จำนวนซี่ฟันบนโรเตอร์

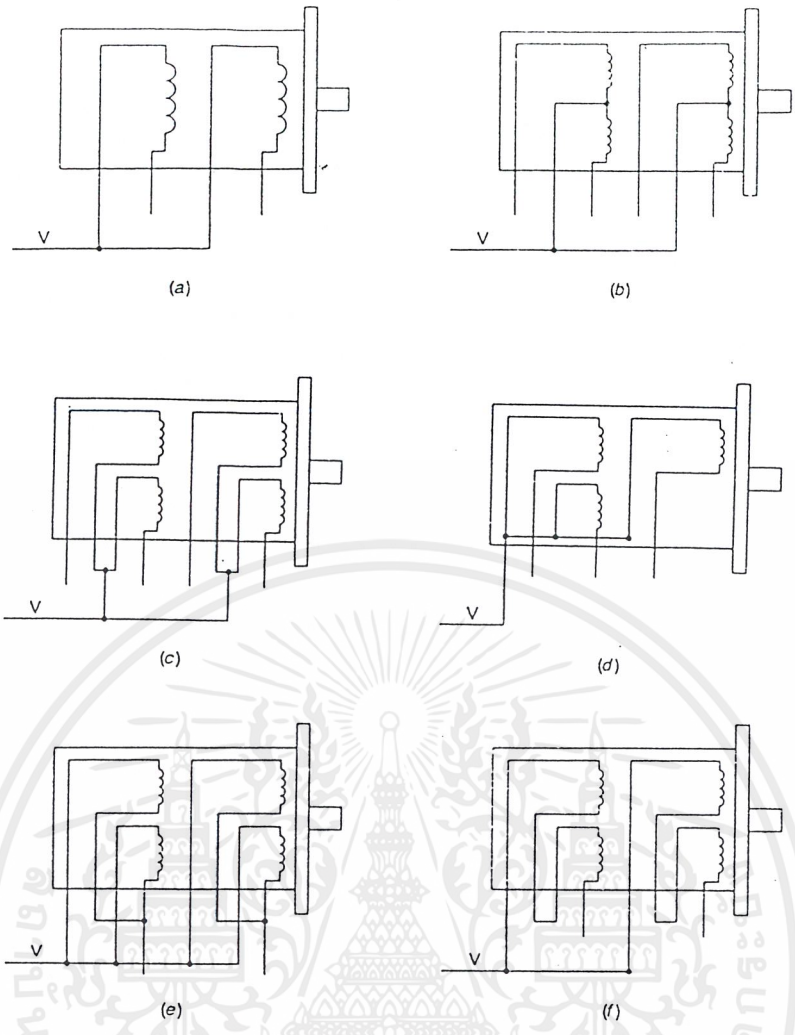
P = จำนวนฟันของสเตเตอร์ต่อเฟส

และความกว้างของมุมสเตปกำหนดโดย

$$\theta = \frac{360}{N} \tag{4.2}$$

เมื่อ N เป็นจำนวนสเตปต่อหนึ่งรอบ

บางครั้งสเตปมอเตอร์อาจแบ่งประเภท ตามจำนวนสายไฟที่ต่อกับมอเตอร์ ซึ่งไม่เกี่ยวข้องกับจำนวนเฟส ดังแสดงในรูปที่ 4.6 มอเตอร์2 เฟส 4 สายเรียกว่า ไบโพลาร์(bipolar) มอเตอร์



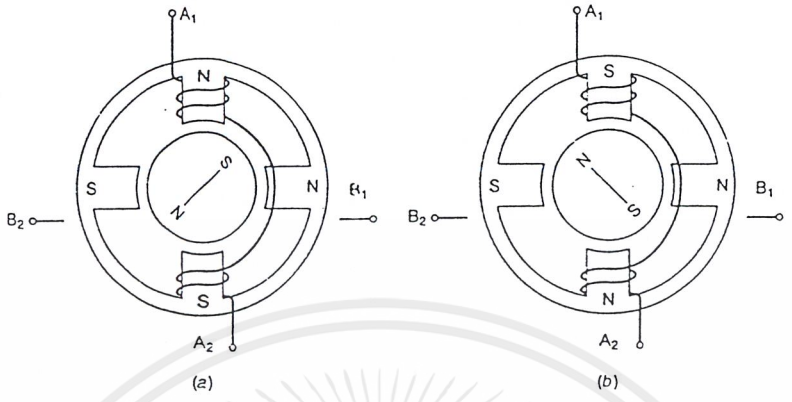
รูปที่ 4.6 แสดงความสัมพันธ์ของสายไฟกับเฟส ของมอเตอร์

4.3 หลักการทำงานของสเตปมอเตอร์

พิจารณาโรเตอร์ชนิดแม่เหล็กถาวรอย่างง่ายซึ่งมีขั้วเหนือและใต้เพียงขั้วเดียว ส่วนของสเตเตอร์มี 4 ฟัน โดยมีคู่ของขดลวด A_1-A_2 และ B_1-B_2 พันอยู่ เมื่อจ่ายแรงดันให้แก่ขดลวด A_1และ A_2 โดย A_2 ต่อกับกราวด์ จะทำให้ฟันซี่ที่ขดลวด A_1 พันอยู่เป็นขั้วเหนือ และ A_2เป็นขั้วใต้ ดังรูปที่ 4.7 จะทำให้โรเตอร์ถูกผลักไปเป็นมุม 45° และสมดุลที่ตำแหน่ง +45° ในทางตรงกันข้าม ถ้าให้สลับขั้ว ไฟที่จ่ายให้ขดลวด A_1และA_2จะทำให้โรเตอร์หมุนไป 90° และสมดุลที่ตำแหน่ง -45° นั่นคือ มอเตอร์หมุนไปเป็นมุมสเตป 90° และถ้าหากเราให้แรงดันอย่างต่อเนื่องตามลำดับ ดังตารางที่ 4.1 จะทำให้โรเตอร์หมุนครบหนึ่งรอบ เมื่อเราจ่ายแรงดันตามลำดับเช่นนี้อย่างต่อเนื่อง ก็จะทำให้มอเตอร์หมุนอย่างต่อเนื่อง ถ้าหากเราต้องการให้มอเตอร์หมุนกลับทิศทาง ก็สามารถทำได้โดยการจ่ายแรงดันที่มีลำดับย้อนกลับจากตารางที่ 4.1

วิธีการจ่ายแรงดันตามลำดับดังกล่าว เรียกว่า Full Step Mode เพราะแต่ละสเตปที่มอแตอร์หมุนไปจะหมุนไปทีละมุมสเตป(step angle) หากเป็นการทำให้มอเตอร์หมุนทีละครึ่งมุมสเตปใน

แต่ละสเตป จะเรียกวิธีการนี้ว่า Half Step Mode แต่การเพิ่มสเตปการหมุนจะทำให้มอเตอร์มีแรงบิดที่ต่ำกว่า การหมุนที่สเตปน้อยกว่า แต่จะมีความราบเรียบในการหมุนมากกว่า



รูปที่ 4.7 ลำดับการทำงานของสเตปมอเตอร์ (a) สเตป 1 (b) สเตป 2

Full-step mode

Step	A_1	A_2	B_1	B_2
1	H	L	H	L
2	L	H	H	L
3	L	H	L	H
4	H	L	L	H
5	Repeat 1 to 4.			

(a)

Half-step mode

Step	A_1	A_2	B_1	B_2
1	H	L	H	L
2	L	L	L	H
3	L	H	H	L
4	H	L	L	L
5	L	H	L	H
6	L	L	H	L
7	H	L	L	H
8	L	H	L	L

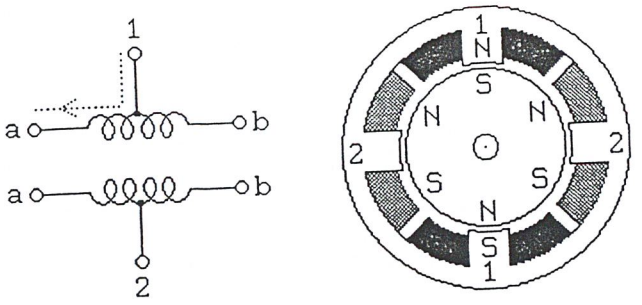
(b)

ตารางที่ 4.2 ลำดับการกระตุ้น (a)แบบ Full-step mode (b)แบบHalf-step mode

4.4 การขับสเตปปิ้งมอเตอร์

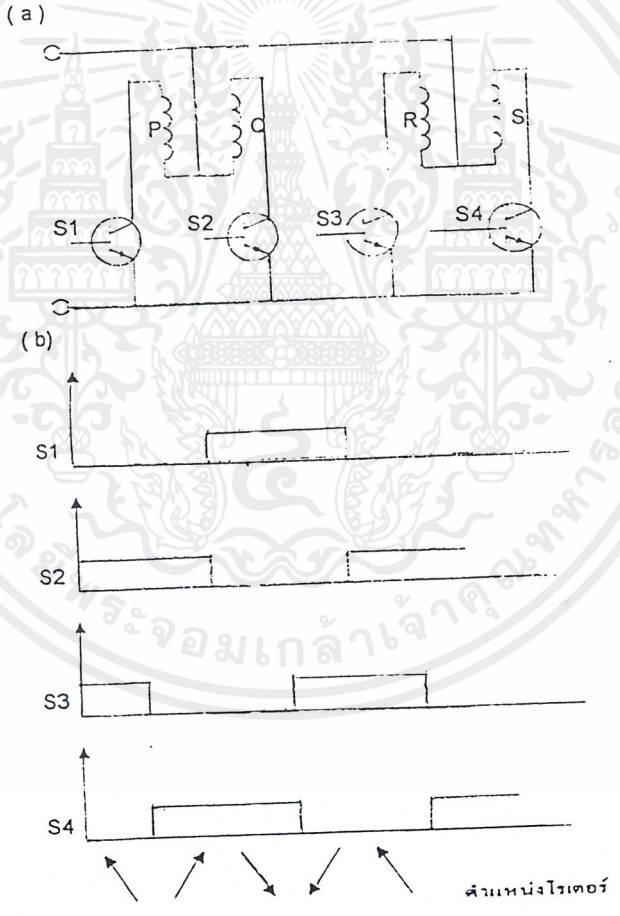
4.4.1 การขับมอเตอร์แบบขั้วเดียว

มอเตอร์แบบขั้วเดียวมีทั้งที่เป็นแบบแม่เหล็กถาวรและไฮบริด โดยทั่วไปจะมีรูปแบบการพันขดลวด ดังรูปที่ 4.8 ขดลวดสเตเตอร์ถูกออกแบบให้มีจุดแบ่งกึ่งกลางบนขดลวด จุดนี้จะต่อไปยังขั้วใดขั้วหนึ่งของแหล่งจ่ายไฟ ทิศทางของเกระแสจะถูกกำหนดโดยการต่อปลายขดลวดเข้ากับขั้วใดขั้วหนึ่งที่เหลืออยู่ของแหล่งจ่ายไฟ โดยใช้อุปกรณ์ประเภทสวิตช์ชิ่งในการตัดต่อวงจร เช่น ทรานซิสเตอร์ และการสลับการทำงานระหว่างขดลวดมีผลทำให้ขั้วแม่เหล็กบนสเตเตอร์ เปลี่ยนแปลง



รูปที่ 4.8 ลักษณะการพันขดลวดของมอเตอร์แบบ Unipolar

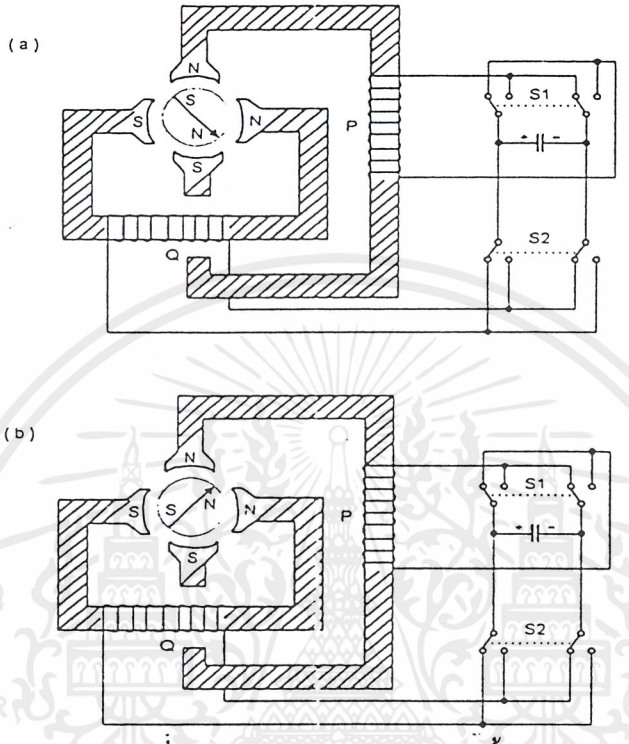
วงจรสวิตช์ซึ่งดังรูปที่ 4.9 (a) โดยให้สัญญาณพัลส์ป้อนเข้าไปยังขาเบสของทรานซิสเตอร์แต่ละตัว จะเห็นว่าที่เวลาใดเวลาหนึ่งจะมีขดลวด 2 เฟส ที่ได้รับกระแสในเวลาที่ต่อเนื่องกัน นั่นคือจะเกิดสนามแม่เหล็กหมุนที่สเตเตอร์เหนี่ยวนำให้มอเตอร์หมุนเคลื่อนที่เป็นลำดับได้



รูปที่ 4. 9 การขับแบบขั้วเดียว (a) วงจรขับมอเตอร์ (b) สัญญาณควบคุม

4.4.2 การขับมอเตอร์แบบสองขั้ว

ขดลวดสเตเตอร์ของมอเตอร์แบบสองขั้วที่ออกแบบมาสำหรับการขับแบบสองขั้วนั้นจะไม่มีจุดแบ่งกึ่งกลาง หลักการทำงานของมอเตอร์ที่ขับแบบสองมีลักษณะเช่นเดียวกับแบบขั้วเดียว ดังรูปที่ 4.10 นั่นคือเมื่อกระแสในเฟส P เปลี่ยนทิศทางโดยการสับสวิตช์ S_1 สนามแม่เหล็กในสเตเตอร์ก็จะเปลี่ยนทิศทางและโรเตอร์จะเคลื่อนที่ไปหนึ่งลำดับ

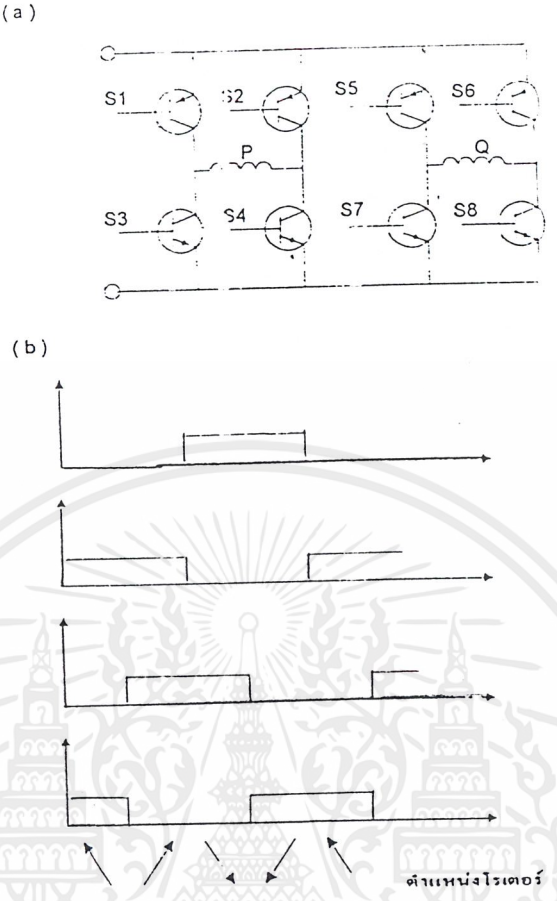


รูปที่ 4.10 มอเตอร์ 4 เฟสแบบ 2 ขั้ว

รูปที่ 4.11 (a) เป็นวงจรขับมอเตอร์ 4 เฟส แบบสองขั้ว ทรานซิสเตอร์ในวงจรจะทำงานเป็นคู่ เช่น S_1 กับ S_4ทำงาน กระแสจะไหลจากขั้วบวกของแหล่งจ่ายไฟไปเข้าทางปลายขดลวด P หรือ S_6 กับ S_7 ทำงาน กระแสจะกลับทิศทางการไหลคือ ไหลจากขั้วลบของแหล่งจายไฟมาเข้าขดลวดและออกทางปลายสาย P เป็นต้น

รูปที่ 4.11 (b) เป็นลักษณะสัญญาณควบคุมที่ป้อนให้กับวงจรขับมอเตอร์ ช่วงเวลาการทำงานของทรานซิสเตอร์(Switching Time) ในวงจรขับแบบสองขั้ว จะต้องมีความเที่ยงตรงกว่าการขับแบบขั้วเดียว มิฉะนั้นแล้วอาจจะทำให้เกิดการลัดวงจรแหล่งจ่ายไฟเพราะทรานซิสเตอร์ตัวบนและล่างนำกระแสพร้อมกัน

การขับแบบสองขั้วมีข้อดี คือให้แรงบิดหรือทอร์ค(Torque)ได้มากกว่า การขับแบบขั้วเดียวที่อัตราช่วงการหมุนต่ำๆ แต่จะมีค่าของแรงบิดใกล้เคียงกันที่อัตราช่วงการหมุนสูงๆ

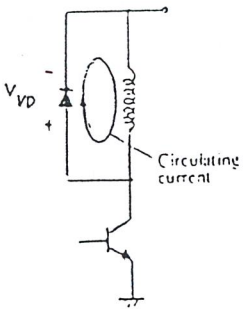


รูปที่ 4.11 (b) สัญญาณควบคุมในการขับมอเตอร์แบบสองขั้ว

4.5 ปัญหาที่เกิดขึ้นในวงจรขับกระแส

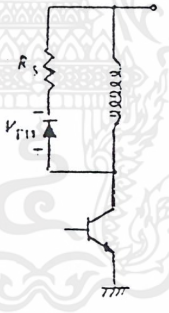
ในขดลวดของสเตปมอเตอร์ ประกอบด้วยตัวต้านทานและความเหนี่ยวนำ ในการขับสเตปมอเตอร์ ขณะที่ทรานซิสเตอร์ในวงจรขับมอเตอร์หยุดนำกระแส จะมี Back E.M.F. ถูกผลิตขึ้นในขดลวด ทำให้เกิดแรงดันค่าสูง ซึ่งเกิดจากการคายพลังงานที่สะสมอยู่ในขดลวด ตามสมการ $V = L\, di/dt$ แรงดันที่เกิดขึ้นนี้ อาจทำให้รอยต่อของทรานซิสเตอร์เสียหายได้ถ้าไม่จำกัดออกไป การปัญหาเหล่านี้แก้ได้หลายวิธี ดังนี้

4.5.1 การกำจัดโดยใช้ไดโอด (Diode Suppression) คือการนำไดโอดมาต่อขนานกันกับขดลวด ดังรูปที่ 4.12 หลังจากที่ทรานซิสเตอร์ หยุดนำกระแส จะเกิดกระแสไหลวนอยู่เป็นวงกลมรอบขดลวดและไดโอด จึงเรียกไดโอดนี้ ว่า free-wheeling diode กระแสนี้จะลดค่าตามเวลา วิธีนี้เป็นวิธีที่ง่ายที่สุด แต่การทำให้กระแสไหลวนหมด ต้องใช้เวลานาน



รูปที่ 4.12 แสดงการต่อ free-wheeling diode

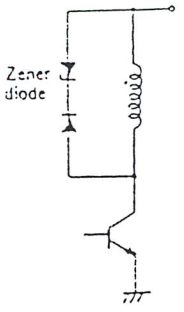
4.5.2 การกำจัดโดยใช้ไดโอดและความต้านทาน (Diode and Resister Suppression) คือ การนำค่าความต้านทานมาอนุกรมกับไดดอด ดังรูปที่ 4.13 ค่าความต้านทานช่วยให้กระแสไหลวนลดลงเร็วขึ้น เพราะมีค่าคงตัวเวลา(Time constant = L/R) น้อยลง



รูปที่ 4.13 การกำจัดโดยใช้ได้โอดและตัวต้านทาน

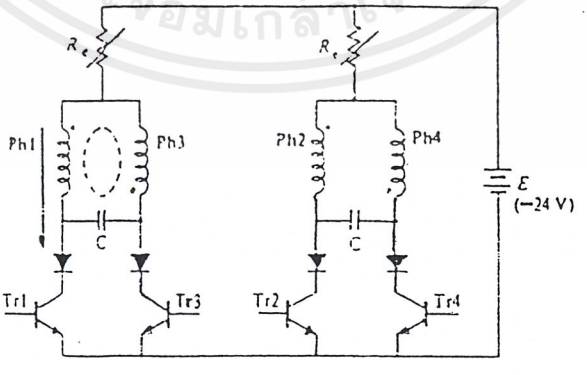
4.5.3 การกำจัดโดยใช้ไดโอดและซีเนอร์ไดโอด (Zener diode Suppression) การใช้ไดโอดเพียงอย่างเดียว ในการำจัดแรงดันสไปร์คมีข้อเสีย คือ แรงบิดที่ได้รับจะน้อยลงไป นอกเสียจากว่าแรงดันตกคร่อมทรานวิสเตอร์จะมีค่าเพิ่มขึ้นประมาณ 2 เท่า ของแหล่งจ่ายแรงดัน ซึ่งแรงดันที่สูงขึ้นนี้จะทำให้สนามแม่เหล็กและกระแสไหลวนลดค่าลงอย่างรวดเร็ว ทำให้แรงบิดดีขึ้น ด้วยเหตุนี้จึงใช้ควมต้านทานต่ออนุกรมกับไดโอดหรือซีเนอร์ไดดอดต่ออนุกรมกับไดโอดแทน ดังรูปที่ 4.14

ซึ่งวิธีการใช้ซีเนอร์ไดโอดจะให้ผลดีที่สุด เมื่อเปรียบเทียบเวลาที่ใช้ในการลดค่ากระแส



รูปที่ 4.14 การกำจัดโดยใช้ไดโอดและซีเนอร์ไดโอด

4.5.4 การใช้ตัวเก็บประจุ (Condensor Suppressor) ตัวเก็บประจุจะถูกใส่ระหว่างเฟส 1 และเฟส 3 และเฟส 2 และเฟส 4 ดังรูปที่ 4.15 เมื่อทรานซิสเตอร์หยุดนำกระแส ตัวเก็บประจุที่เชื่อมต่อระหว่างไดดอดจะถูกดูดซับกระแสที่เกิดจากขดลวด เพื่อที่จะป้องกันทรานซิสเตอร์ พิจารณาเมื่อ ทรานซิสเตอร์ Tr_1 หยุดนำกระแส Tr_2 และ Tr_4 แต่ Tr_3 ยังคง OFF อยู่ ดังนั้นขดลวดเฟสที่ 1 และเฟสที่ 3 ซึ่งเป็นกระแสออสซิเลทที่มีทิศทางตรงกันข้าม ทำให้เกิดกระแสทรานเซี้ยน (Transient Current) จะไหลวนในลูปตามเส้นประในรูป ถ้า Tr_3 ON เมื่อกระแสทรานเซี้ยนกลายเป็นศูนย์ นั่นคือ ตัวเก็บประจุจะเก็บประจุจนมีประจุมากที่สุด ดังนั้นกระแสด้านบวกจากประจุ C ก็จะ discharge
ผ่านขดลวดเฟสที่ 1 โดยเกิดการ Resonance ซึ่งการใช้ตัวเก็บประจุนี้เหมาะสมที่จะใช้ขับสเตบ่มอเตอร์ที่พิกัดที่จำกัดในขอบเขตที่แคบๆ



รูปที่ 4.15 การกำจัดโดยใช้ตัวเก็บประจุ

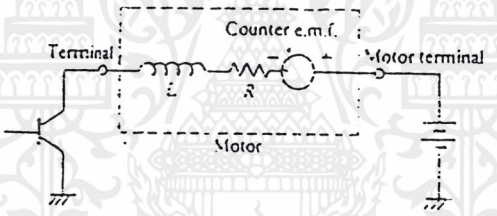
4.5.5 การต่อความต้านทานอนุกรมในวงจรขับมอเตอร์

ในสภาวะที่มอเตอร์ทำงานด้วยระดับแรงดันคงที่ค่าหนึ่ง แรงบิดที่ได้จะลดลงถ้ามีการเพิ่มอัตราสเตป เนื่องมาจากการเพิ่มแรงดันต้านกลับ (Back EMF) และช่วงเวลาขึ้น(Rise Time) ของกระแสในขดลวดถูกจำกัดหรือมีค่ามาก จากวงจรเทียบเคียงของสเตปปิ้งมอเตอร์ดังรูปที่ 2.2? (a) จะได้สมการ ของแรงดัน

$$L\frac{dI}{dt} + RI = Vs \qquad \text{.....(1)}$$

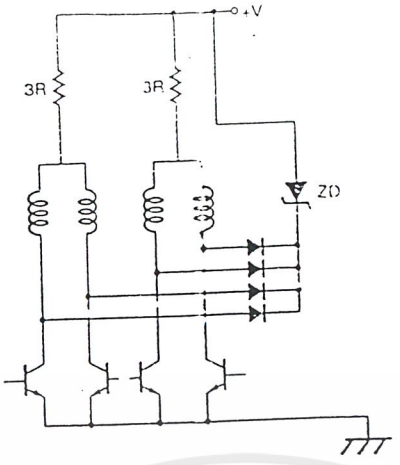
และสมการของกระแส

$$I(t) = \frac{Vs}{R}[1 - e^{-t/\tau}] \qquad \text{.....(2)}$$



รูปที่ 4.15 การต่อความต้านทานอนุกรมในวงจรขับมอเตอร์

วิธีแก้ไขให้เกิดการเปลี่ยนแปลงของกระแสในขดลวดเร็วขึ้น ทำได้โดยการเพิ่มแรงดันของแหล่งจ่ายไฟให้สูงขึ้นเพื่อรักษาระดับแรงดันให้คงที่ และขณะทำงานที่อัตราสเตป สูงๆ หรืออาจคงค่าของแรงดันเอาไว้ด้วย แต่เพิ่มค่าความต้านทานอนุกรมเข้าไปในวงจรดังรูปที่ 4.16 ค่าความต้านทานนี้เรียกว่า Forcing Resistance ค่าความต้านทานที่ต่อเพิ่มนี้จะลดค่าเวลาคงตัว($\tau = L/R$) ทำให้ใช้งานที่ความเร็วสูงได้ดี ในรูปที่ 4.16 เพิ่มความต้านทานมีค่า 3 เท่าของความต้านทานขดลวด จะทำให้มีค่าเวลาคงตัวเท่ากับ L/4R และควรเพิ่มขนาดแรงดัน แหล่งจ่ายไฟเป็น 4 เท่าด้วย เพื่อรักษาระดับกระแสที่ผ่านขดลวดให้คงที่ ซึ่งในกรณีนี้อาจมีปัญหาในเรื่องแหล่งจ่ายไฟ หรือเกิดการสูญเสียที่ forcing resistance ในรูของความร้อน ดังนั้นที่ความเร็วต่ำๆ จึงไม่เหมาะที่จะนำวิธีการนี้มาใช้



รูปที่ 4.15 การต่อความต้านทานสำหรับวงจรขับแบบ L\4R

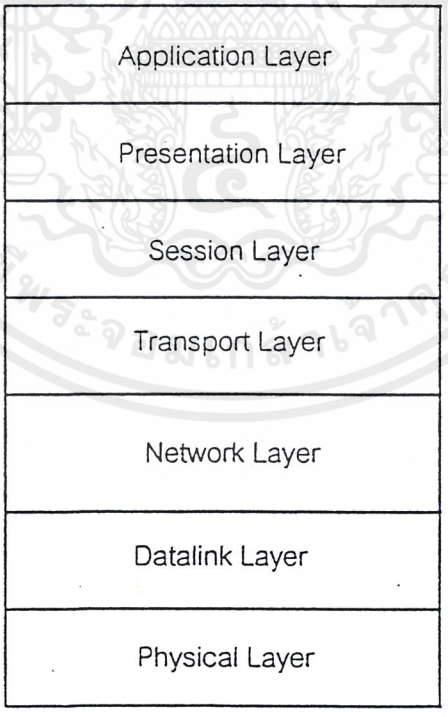
วงจรที่ใช้ในการขับสเตปปิ้งมอเตอร์ที่เลือกใช้นั้น จะเหมือนดังรูปที่ 4.15 แต่จะไม่ใส่ความต้านทานอนุกรม

บทที่ 5
เครือข่ายอินเตอร์เน็ต

Internet คือ การที่เครือข่าย 2 หรือมากกว่าเชื่อมต่อเข้าด้วยกัน และทำงานเสมือนเป็นเครือข่ายเดียวกัน โดย Network ที่เป็นส่วนประกอบของอินเตอร์เน็ต คือ Subnetwork ซึ่งอาจจะเป็นเครือข่าย Local area network (LAN) หรือ Wide area network (WAN) อุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่ายเข้าด้วยกันคือ Intermediate system (IS) หรือ internetworking unit (IWU) การเชื่อมโยงระหว่างระบบที่แตกต่างกัน จำเป็นต้องมีมาตรฐานในการติดต่อกัน ซึ่งเรียกเป็นศัพย์เฉพาะว่าโพรโทคอล

5.1 OSI โมเดล

องค์การมาตรฐานสากล ISO ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อยๆ และกำหนดโมเดลแบ่งเป็นชั้นๆ ตามลำดับเรียกว่า OSI โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อยๆ ซึ่งจะช่วยในการออกแบบ และการใช้งานเครือข่ายรวมถึงการเชื่อมโยงกัน เป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

Application Layer เป็นชั้นบนสุดของโมเดล เป็นส่วนที่ทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้เป็นไปได้ตามต้องการ เช่น ระบบ e-mail การโอนถ่ายข้อมูล การขอเข้าใช้คอมพิวเตอร์ในระบบเครือข่าย เป็นต้น

Presentation Layer มีการกำหนดหน้าที่ไม่ชัดเจนนักและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือ เป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมไปถึงการจัดแปลงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC การลดขนาดข้อมูล การเข้ารหัส หรือถอดรหัสของข้อมูล แต่ส่วนใหญ่แล้ว แอปพลิเคชั่นจะจัดการแทนได้

Session Layer เป็นชั้นที่จัดการในเรื่องของ การติดต่อแต่ละครั้ง หรือ Session ให้ระบบคอมพิวเตอร์ทั้งสองฝ่าย โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อแต่ละครั้งนั้นๆเป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณ และรายละเอียดวิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้าย ที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP ในโพรโทคอล TCP / IP ทำงานที่ระดับนี้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน packet ข้อมูล ผ่านอุปกรณ์ต่างๆไปยังเครือข่ายย่อยได้อย่าถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูลและกั่นหรือกรอง packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่วิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP , TCP / IP , IPX เป็นโพรโทคอลที่ทำงานอยู่ใน Layer นี้

Data Link Layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet , Tokenring หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ ที่จะส่งข้อมูลไป ทั้งนี้ชั้นนี้จะเป็นชั้นแรกที่จัดการแปลงข้อมูลแบบ bit ให้เป็น packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลเข้ามา ก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ packet ข้อมูลที่ไม่ถูกตองก็จะไม่เอาข้อมูลนั้นไปใช้งาน และจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

Physical Layer รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณผ่านสายสัญญาณแบบต่างๆ การเชื่อมต่อเข้ากับเครือข่ายแบบต่างๆ โดยใช้ Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า สัญญาณเสียง หรือสัญญาณที่จำเป็นในการสื่อสารโดยตรง

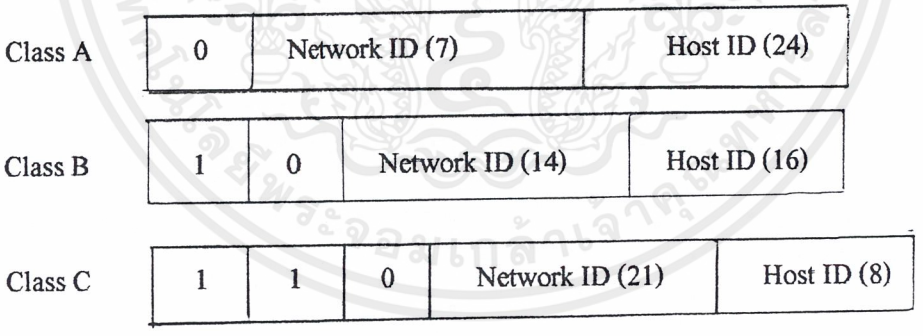
Network Layer จะเป็นตัวจัดการการติดต่อแบบ end-to-end ของการบริการ internetwide ไปยังผู้ใช้บริการ ISOได้จัด network Layer เป็น 3 Protocol ซึ่งทำงานร่วมกันเพื่อให้บริการใน Network Layer ได้แก่ 1. Subnetwork independent convergence protocol (SNICP)

2. Subnetwork dependent convergence protocol (SNDCE)
3. Subnetwork dependent access protocol (SNDAP)

5.2 Address Structure

ในศัพท์ของ ISO เมื่อ 2 network ติดต่อกันด้วย Host / ES ที่ติดต่อกับอินเตอร์เน็ต network เหล่านี้ติดต่อกันได้โดยใช้ network service access point (NSAP) address และ subnet pooint of attachment (SNPA) สำหรับใน TCP / IP ก็จะมี NPA address ตามลำดับ โดย NPA address จะต่างกันในแต่ละชนิดของ network / subnet ขณะที่ IP address จะเป็นรูปแบบเดียวกัน

IP address นี้จะมีการจัดแบ่งออกเป็นทั้งหมด 5 ลำดับ แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ลำดับ คือ Class A , Class B และ Class C ซึ่งจะแบ่งตามขนาดความใหญ่ของเครือข่าย ถ้าเครือข่ายใดมีจำนวนเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ใน Class A และจะลดหลั่นกันลงมาตามลำดับ



รูปที่ 5.2 แสดงคลาสของไอพีแอดเดรส

จากรูปจะเห็นว่าหมายเลข IP ของ Class A มีตัวแรกเป็น 0 และหมายเลขของเครือข่าย ขนาด 7 bit และมีหมายเลขเครื่องคอมพิวเตอร์ขนาด 24 bit ทำให้หนึ่งเครือข่ายของ Class A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ในเครือข่ายได้ถึง 16 ล้านเครื่อง แต่ใน Class A นี้ จะมีหมายเลขเครือข่ายได้ 128 ตัวเท่านั้นทั่วโลก ซึ่งก็คือมีเครือข่ายใหญ่แบบนี้ 128 เครือเท่านั้น

สำหรับ Class B จะมีหมายเลขเครือข่ายแบบ 14 bit และหมายเลขเครื่องคอมพิวเตอร์แบบ 16 bit ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่ายในแต่ละเครือข่ายได้ถึง 65000 เครื่อง

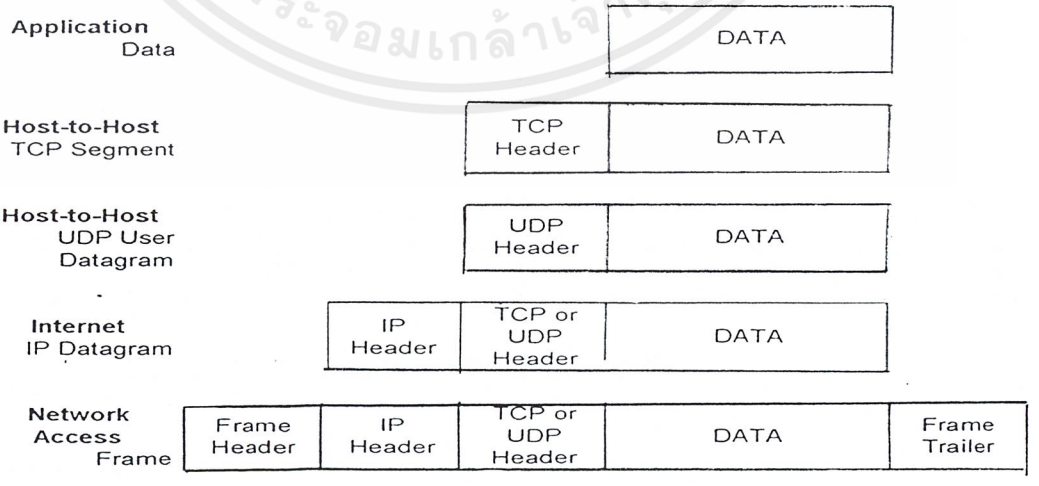
Class C มีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit ดังนั้นในแต่ละเครือข่ายจะมีเครื่องเชื่อมต่อกับได้ไม่เกิน 254 เครื่องในแต่ละเครือข่าย

เมื่อเครือข่ายและเครื่องคอมพิวเตอร์ ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกัน และมีความหมายให้ทราบถึงขนาดเครือข่ายแล้ว การติดต่อส่งผ่านข้อมูลจึงทำได้ไม่สับสน

5.3 ชุดโพรโตคอลทีซีพี / ไอพี

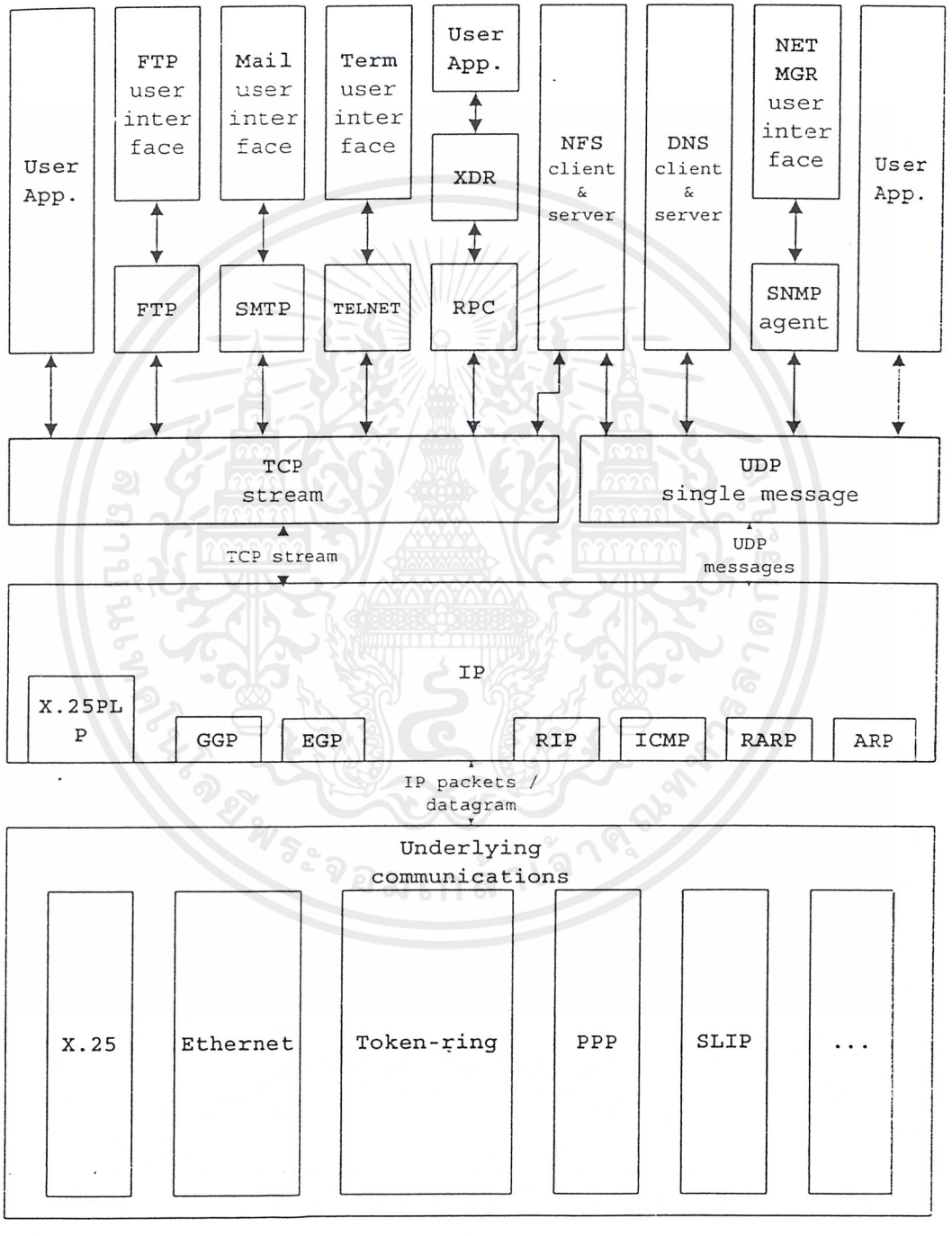
โพรโตคอลทีซีพี / ไอพี (TCP / IP – Transmission Control Protocol / Internet Protocol) เป็นกลุ่มโพรโตคอลที่พัฒนาขึ้นมาเพื่อให้คอมพิวเตอร์สามารถใช้ทรัพยากรและบริการฟังก์ชั่นพื้นฐานสำหรับการใช้งานบนระบบสื่อสารข้อมูลคอมพิวเตอร์ได้ ในสมัยก่อนนิยมใช้ทีซีพี / ไอพี ในการสื่อสารที่ในเครื่องระดับมินิคอมพิวเตอร์หรือเมนเฟรม ซึ่งจะมีบริการอยู่หลายแบบ เช่น การล็อกอินจากที่อื่น การแลกเปลี่ยนไฟล์ข้อมูล จดหมายอิเล็กทรอนิกส์ เป็นต้น ปัจจุบันชุดโพรโตคอลทีซีพี / ไอพี ได้รับความนิยมในการใช้งานอย่างแพร่หลาย เนื่องจากใช้เป็นโพรโตคอลหลักในการติดต่อสื่อสารบนระบบเครือข่ายคอมพิวเตอร์ขนาดใหญ่ที่สุดในโลกที่เรียยกว่าระบบอินเตอร์เน็ต

ข้อมูลที่ทีซีพี /ไอพี นำส่งจะถูกแบ่งออกเป็นข้อมูลย่อยหลายๆส่วน เพื่อทยอยส่งไปตามลำดับเพื่อให้เหมาะสมกับระบบเครือข่ายในชั้นถัดไปที่อาจไม่สามารถส่งข้อมูลขนาดใหญ่ได้ทันที และเมื่อส่งไปถึงปลายทางก็จะรวบรวมข้อมูลนั้นกลับเป็นชุดเดิมอีกครั้งหนึ่ง ซึ่งจะมีการจัดรูปแบบเพ็กเก็ตในการสื่อสารดังรูป



รูปที่ 5.3 การจัดเตรียมข้อมูลเป็นแพ็กเก็ตเพื่อทำการส่ง

ชุดโพรโตคอลทีซีพี / ไอพี จะมีทีซีพี / ไอพีเป็นหลักและโพรโตคอลอื่นๆ ที่ทำงานร่วมกับทีซีพี / ไอพีในชั้นอื่นๆของทีซีพี / ไอพีโมเดล ซึ่งมีทั้งโพรโทคอลช่วยเหลือ เช่น ICMP , ARP, RIP และโพรโตคอลที่ใช้ทำงานหลัก เช่น TELNET , FTP , SMTP , HTTP เป็นต้น



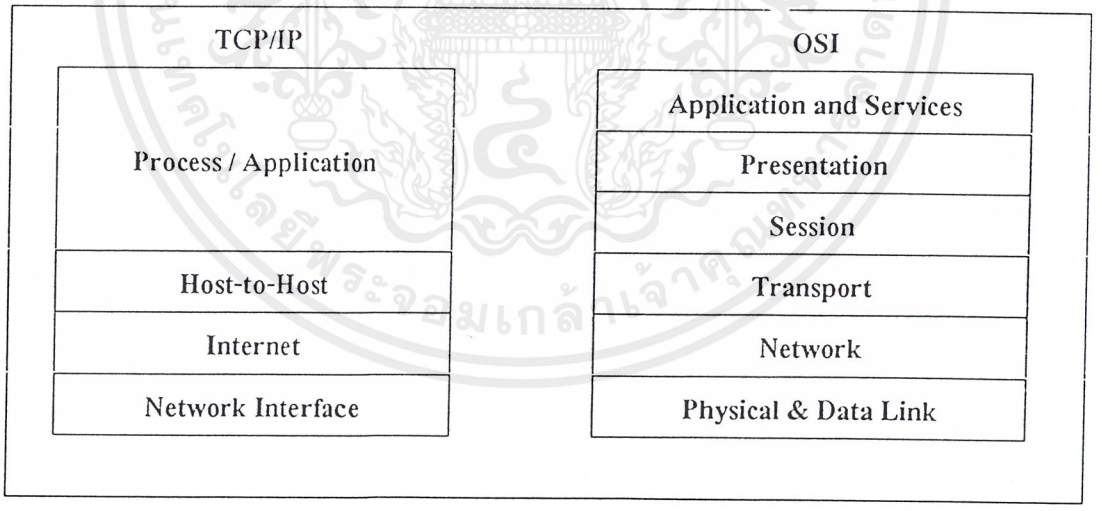
รูปที่ 5.4 การใช้งานชุดโพรโตคอลทีซีพี / ไอพี

5.3.1 ชั้นต่างๆของทีซีพี / ไอพี

การติดต่อสื่อสารของทีซีพี / ไอพี ถูกกำหนดให้มีการทำงานเป็นระดับชั้น และเพื่อให้มีตอนการทำงานในการแลกเปลี่ยนข้อมูลระหว่างคอมพิวเตอร์เป็นไปได้อย่างถูกต้องดังนี้

- กำหนดรูปแบบข้อมูล
- จัดเตรียมชุดข้อมูล
- กำหนดเส้นทางการส่งข้อมูล
- กำหนดอัตราเร็วในการส่งข้อมูล
- ทำการส่งข้อมูลผ่านตัวกลาง
- รวบรวมและจัดลำดับชุดข้อมูลที่ส่งมา
- ตรวจสอบว่ามีชุดข้อมูลซ้ำหรือไม่
- ตอบกลับไปให้ผู้ส่งรู้ว่าได้รับข้อมูลแล้ว
- ส่งผ่านข้อมูลไปให้ชั้นการทำงานถัดไป

เมื่อเปรียบเทียบกับโมเดลอ้างอิงการเชื่อมต่อระบบเปิด(Open System Interconnection Reference Model : OSI-RM) โดย ISO จะได้ดังนี้



รูปที่ 5.5 การแบ่งระดับการทำงานของทีซีพี / ไอพี และ OSI

5.3.2 ชั้นเชื่อมต่อระบบเครือข่าย

ทำงานในชั้นเดียวกับ OSI Physical Layer และ Data Link Layer ชั้นนี้จะทำหน้าที่ในการสื่อสารข้อมูลทางกายภาพ ในระดับสัญญาณนำส่ง ตัวนำที่ใช้ในการส่ง ระบบสื่อสัญญาณ และรูปแบบสัญญาณที่ใช้ในการแทนข้อมูล ว่าเป็นสัญญาณลอจิก "0" หรือ "1" ตัวอย่างของระบบที่ทำงานในชั้นนี้ เช่น ระบบเครือข่ายแบบอีเทอร์เน็ต หรือระบบเครือข่ายแบบโทเก้นริง และจัดข้อมูลเป็นกลุ่มที่เรียกว่าเฟรม เฟรมจะมีส่วนหัวใช้แสดงตำแหน่งของต้นทางและปลายทาง ข่าวสารที่ในการควบคุม และส่วนท้ายที่ใช้ในการตรวจสอบข้อผิดพลาดการติดต่อสื่สารระดับล่างสุดเฟรมจะถูกส่งจากอุปกรณ์เชื่อมต่อระบบเครือข่าย ของเครื่องต้นทางผ่านระบบสื่อสารต่างๆ ไปอุปกรณ์เชื่อมต่อระบบเครือข่ายของเครื่องปลายทาง

5.3.3 ชั้นอินเตอร์เน็ต

ชั้นนี้จะมีอินเตอร์เน็ตโพรโทคอล หรือไอพี เป็นโพรโทคอลหลัก คอยทำการหาเส้นทางที่เหมาะสมให้ในการสื่อสารข้อมูลระหว่างระบบ ข้อมูลที่ส่งเรียกว่าเดต้าแกรม ซึ่งจะถูกส่งไปในระบบที่อาจจะเชื่อมต่อกันโดยตรงหรือเชื่อมต่อกันผ่านระบบสื่อสารอื่นๆอยู่ก็ได้

ไอพีจะทำงานแบบไม่มีการเชื่อมต่อกันก่อน เดต้าแกรมแต่ละตัวจะถูกจัดเส้นทางส่งเป็นอิสระต่อกัน ไอพีไม่มีการรับประกันความถูกต้อง ความน่าเชื่อถือ หรือแม้แต่ การจัดเรียงลำดับเดต้าแกรมให้อยู่ในลำดับที่ถูกต้อง

ชุดข้อมูลจะถูกส่งไปในระบบเครือข่าย โดยแต่ละเครือข่ายจะมีเครื่องที่ทำหน้าที่จัดเส้นทาง ซึ่งจะดูหมายเลขปลายทางแล้วตัดสินใจว่าจะส่งข้อมูลไปยังเส้นทางไหน ตัวจัดเส้นทางนี้อาจจะเป็นเครื่องคอมพิวเตอร์ธรรมดาซึ่งเพิ่มหน้าที่การหาเส้นทางเข้าไป หรือใช้เครื่องที่ทำหน้าที่จัดหาเส้นทางโดยเฉพาะ กว่าที่ข้อมูลจะไปถึงปลายทาง อาจจะต้องผ่านตัวจัดเส้นทางของหลายเครือข่าย จึงต้องมีการผนวกหมายเลขของเครื่องต้นทางและเครื่องปลายทางเข้าไปในชุดข้อมูล เพื่อให้เราท์เตอร์รู้ว่าข้อมูลที่ผ่านเข้ามา ต้องการจะไปทางไหน ถ้าไม่ใช้หมายเลขเครือข่ายตัวเอง ก็จะส่งต่อไปยังเครือข่ายที่อื่น แต่ถ้าใช่ก็จะส่งไปให้สมาชิกทั้งหมดของเครือข่าย เครื่องที่อยู่ในเครือข่ายจะตรวจสอบชุดข้อมูลที่ผ่านมาทั้งหมดว่าเป็นข้อมูลของตัวเองหรือไม่ ถ้าใช่ก็รับข้อมูลนั้นไว้แล้วส่งไปให้กับส่วนการทำงานในชั้นโฮสต์ทูโฮสต์อีกทีหนึ่ง

5.3.4 ชั้นโฮสต์ทูโฮสต์ และ UDP

โพรโทรคอลที่ทำงานในชั้นโฮสต์ทูโฮสต์นี้มี 2 แบบ คือแบบที่เรียกว่า ทีซีพี จะทำงานแบบมีการเชื่อมต่อก่อน ซึ่งจะเป็นส่วนในการทำงานภายในตัวคอมพิวเตอร์แต่ละเครื่อง มีหน้าที่นำส่งข้อมูลโดยรับประกันความน่าเชื่อถือให้ด้วย ว่าข้อมูลที่นำส่งจะไม่มีข้อผิดพลาด และเรียงอยู่ในลำดับที่ถูกต้อง โพรโทคอลทีซีพีจะทำการเพิ่มส่วนหัวของชั้นโพรโทคอลให้กับข้อมูลเพื่อสร้างเป็นเซกเมนต์ โพรโทคอลอีกแบบหนึ่งได้แก่ ยูดีพี ซึ่งจะทำงานแบบไม่มีการเชื่อมต่อก่อน และไม่มีการรับประกันความถูกต้องของข้อมูล เรียกข้อมูลที่ส่งโดยยูพีดีว่า User Datagram ตัวอย่างการทำงานโดยใช้ยูพีดี เช่น การตรวจสอบข้อมูลชื่อจากฐานข้อมูลในระบบ Domain Name System

5.3.5 ชั้นโปรแกรมประยุกต์ (Application Layer)

ชุดโพรโทคอลทีซีพี / ไอพี จะมีโพรโทคอลในชั้นโปรแกรมประยุกต์ใช้งานอยู่มาก ที่นิยมใช้กันมากและจัดเป็นบริการพื้นฐานของทีซีพี / ไอพี เช่น

การล็อกอินระยะไกล ทำให้ผู้ใช้เครื่องคอมพิวเตอร์ในระบบเครือข่ายสามารถทำการล็อกอินเข้าไปใช้ทรัพยากรในคอมพิวเตอร์อื่นที่ต่อเชื่อมกันอยู่ในระบบเครือข่ายได้จากเทอร์มินอลของตัวเองที่มีความแตกต่างกัน โดยใช้เทลเน็ตโพรโทคอล ทำให้เกิดโปรแกรม Telnet ที่ระบบเทอร์มินอลจำลอง ซึ่งสามารถใช้งานกับคอมพิวเตอร์ได้เกือบทุกระบบ

การส่งผ่านแฟ้มข้อมูล ทำให้ผู้ใช้เครื่องคอมพิวเตอร์เครื่องใดก็ตามสามารถรับส่งไฟล์จากเครื่องคอมพิวเตอร์อื่นๆได้ โดยใช้ไฟล์ทรานสเฟอร์โพรโทคอล ทำหน้าที่ในการคัดลอกแฟ้มข้อมูลระหว่างเครื่อง และทำงานทั่วไปเกี่ยวกับแฟ้มข้อมูลเช่น การเปลี่ยนชื่อแฟ้ม การลบแฟ้ม เป็นต้น

การส่งไปรษณีย์อิเล็กทรอนิกส์ ทำให้ผู้ส่งข้อความไปหาผู้ใช้คนอื่นในระบบเครือข่ายได้ โดยการกำหนดรูปแบบข้อความที่จะส่งให้เป็นมาตรฐานเดียวกันในกระบวนการในการรับและการส่งระหว่างเครื่องต่างๆ

บริการเวิลด์ไวด์เว็บ จัดเป็นบริการที่มีความสามารถมากที่สุดในโปรแกรมประยุกต์ที่ทำงานแบบ Client / Server ของทีซีพี / ไอพี และได้รับความนิยมสูงมากในปัจจุบัน ทำให้ผู้ใช้สามารถสืบค้นข้อมูลในลักษณะของ Hypermedia โดยการใช้เอชทีทีพีโพรโทคอล

เน็ตเวิร์คไฟล์ซิสเต็ม เป็นการอนุญาติให้ระบบเข้าถึงข้อมูลจากคอมพิวเตอร์เครื่องอื่นได้โดยผ่านระบบไฟล์ของระบบจัดการนั้นๆ เอง

การพิมพ์ระยะไกล ให้เราสามารถใช้งานเครื่องพิมพ์โดยผ่านทางระบบเครือข่ายได้

นอกจากนี้ก็มีการให้บริการค้นหาชื่อสมาชิกเครือข่าย การจัดระบบเครือข่าย โดยใช้ SNMP (Simple Network Manangement Protocol)

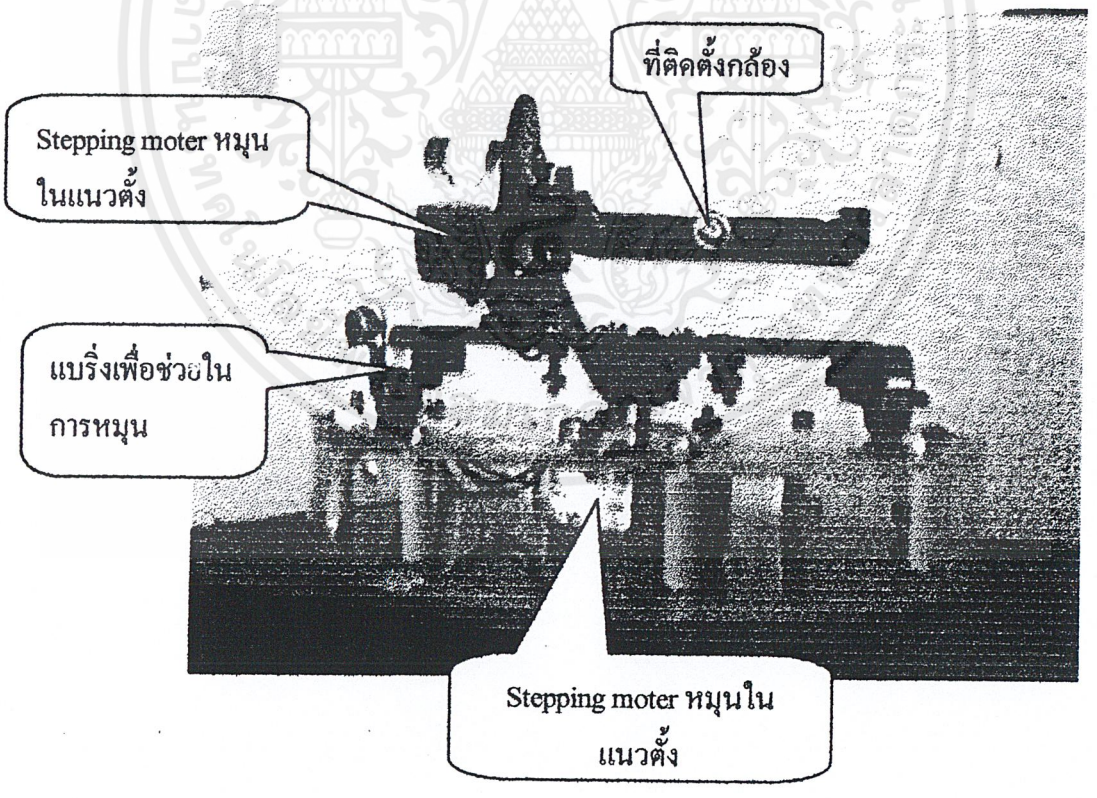
ในปัจจุบันได้มีการสร้างโปรแกรมที่ทำงานบนเครือข่ายโดยใช้ ทีซีพี / ไอพี จำนวนมาก เช่น ระบบฐานข้อมูลที่ให้บริการระหว่างเครื่องในเครือข่าย หรือการทำงานบนระบบประมวลผลแบบกระจาย ซึ่งแสดงให้เห็นว่า ทีซีพี / ไอพี มีบทบาทเป็นโพรโทคอลพื้นฐานที่สำคัญของการใช้งานระบบเครือข่ายในปัจจุบัน

บทที่ 6
หลักการออกแบบและการสร้าง

ในการสร้างโครงงานชิ้นนี้ได้ทดสอบคุณสมบัติต่างๆของโครงงาน โดยจะแยกผลการทดลองแต่ละส่วนเพื่อให้เห็นคุณสมบัติต่างกันอย่างชัดเจน

6.1 ฐานติดตั้งกล้อง

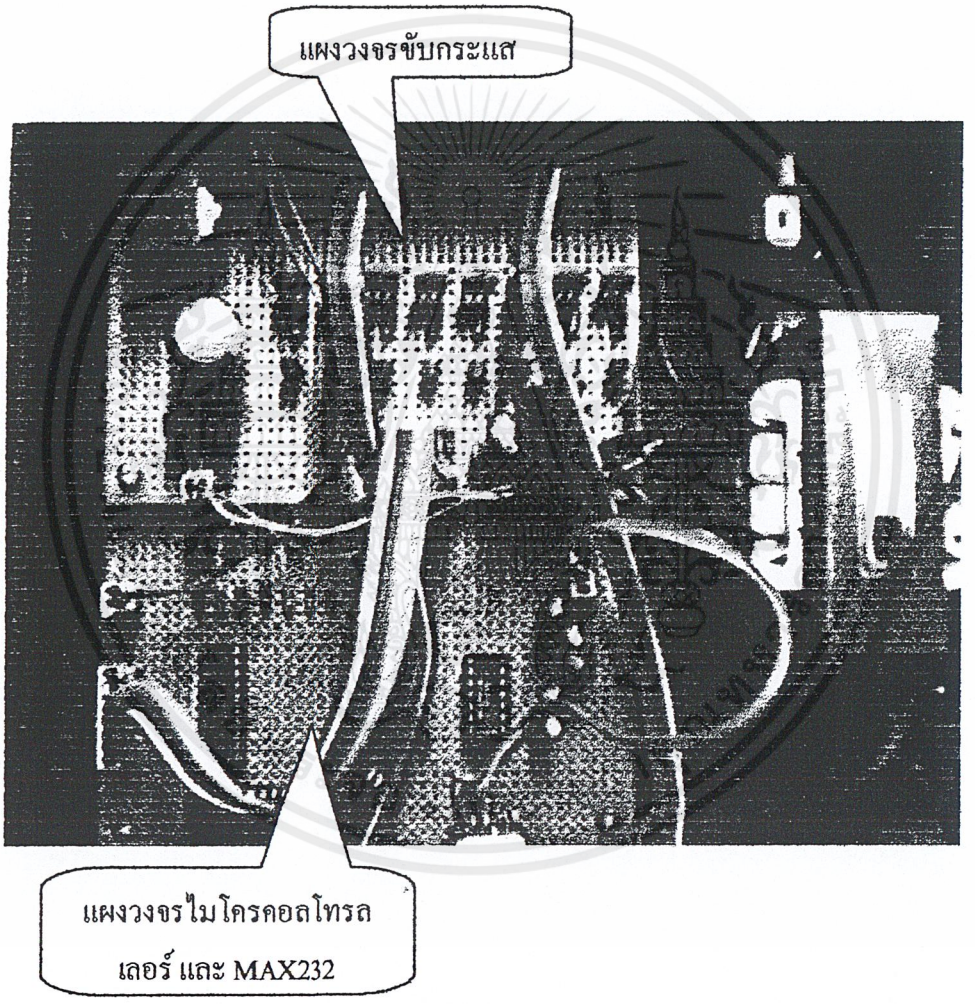
มีลักษณะคล้ายๆแขนกล แต่ได้ออกแบบมาเฉพาะกล้องเท่านั้น โดยใช้พลาสติกทำเป็นส่วนใหญ่ เพราะมีราคาถูกและง่ายต่อการสร้าง เราได้ทดสอบการเคลื่อนที่ของฐานติดตั้งกล้องแต่ละส่วนโดยโปรแกรมทดสอบเบื้องต้น การขับมอเตอร์แบบสองเฟส จะให้แรงบิดมากกว่าการขับมอเตอร์แบบเฟสเดียว ดังนั้นเราจึงใช้การขับมอเตอร์แบบสองเฟส สำหรับชิ้นงานนี้ฐานกล้องจะมีลักษณะดังรูปข้างล่าง



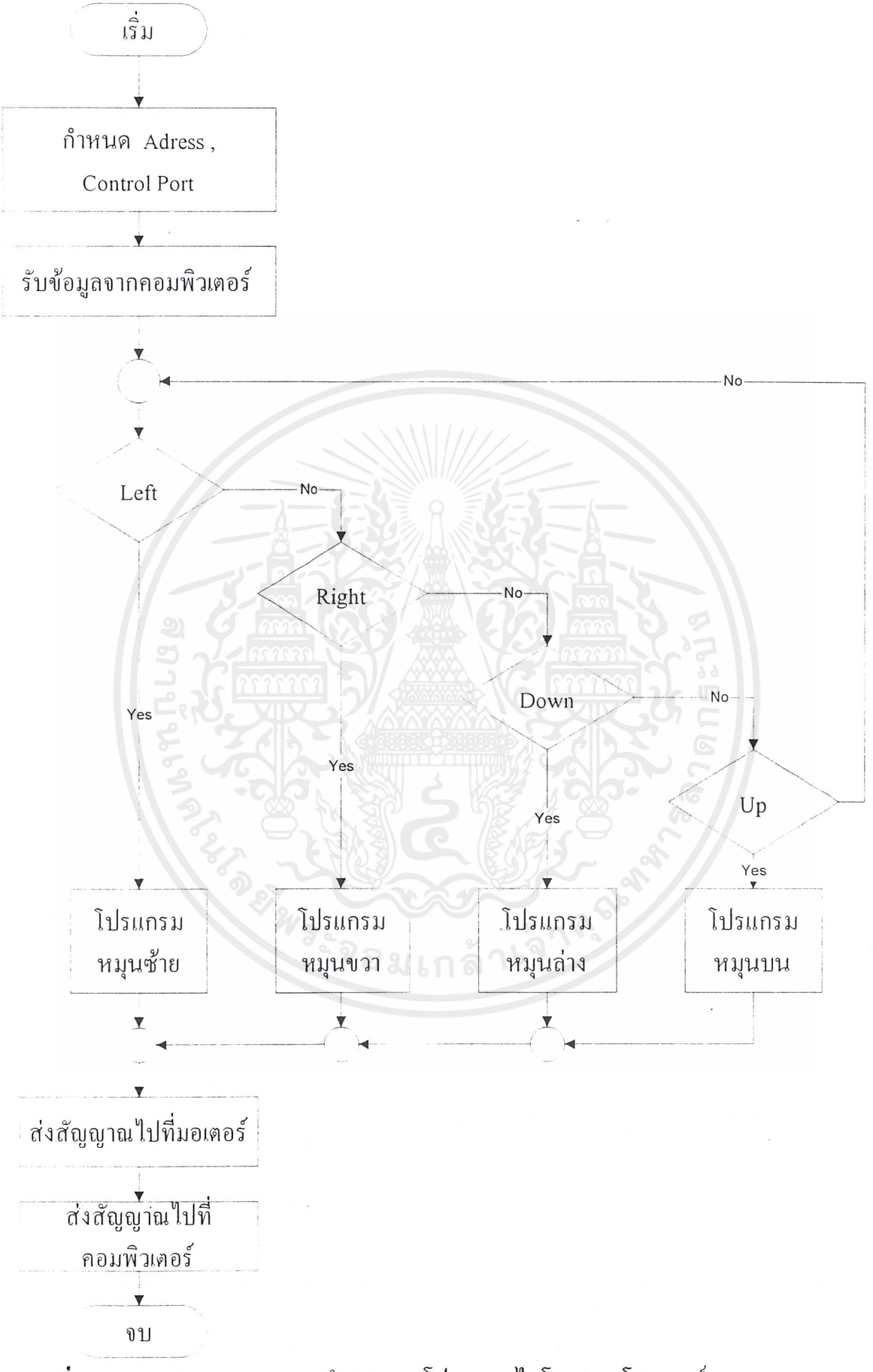
รูปที่ 6.1 ฐานติดตั้งกล้อง

6.2 ชุดควบคุมการหมุน

จะมีองค์ประกอบหลักคือ ไมโครคอนโทรเลอร์ วงจรขับกระแส และ MAX232 ใช้มอเตอร์ 2 ตัว แต่ละตัวมี 4 เฟส เพราะฉะนั้นจึงใช้ชุดขับมอเตอร์ 8 ชุด ส่วนที่ทำหน้าที่ในการควบคุมการหมุนของกล้องจะใช้ไมโครคอนโทลเลอร์รับข้อมูลจากพอร์ทอนุกรมของคอมพิวเตอร์ โดยมี MAX232 เพื่อปรับระดับแรงดันก่อนเข้าไมโครคอนโทลเลอร์ รูปวงจรจะเป็นดังรูป 6.2



รูปที่ 6.2 ชุดควบคุมการหมุน



รูปที่ 6.3 Flow Chart แสดงการทำงานของโปรแกรมไมโครคอนโทรเลอร์

เริ่มจากเรากำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆ ที่จำเป็นในการกำหนดรูปแบบการทำงาน เมื่อกำหนดค่าพารามิเตอร์ต่างๆแล้ว จะเป็นส่วนของการรอรับค่าจากคอมพิวเตอร์ โดยผ่านทาง RS232 ซึ่งเป็นพอร์ทอนุกรม การรับค่าจากคอมพิวเตอร์จะรับเพียงแค่ Byte เดียว ซึ่งจะเป็นโค้ดที่เป็นการกำหนดให้วงจรทำงานตามโค้ดที่ส่งมา

เมื่อได้รับโค้ดแล้วก็จะทำการตรวจสอบว่าเป็นโค้ดสำหรับอะไร แล้วก็จะเข้าไปทำในโปรแกรมย่อยที่ใช้ในการเคลื่อนที่ของแต่ละส่วนของมอเตอร์ ต่อไป

6.3 โปรแกรมการติดต่อใช้งานระหว่างผู้ควบคุมและผู้ให้บริการ

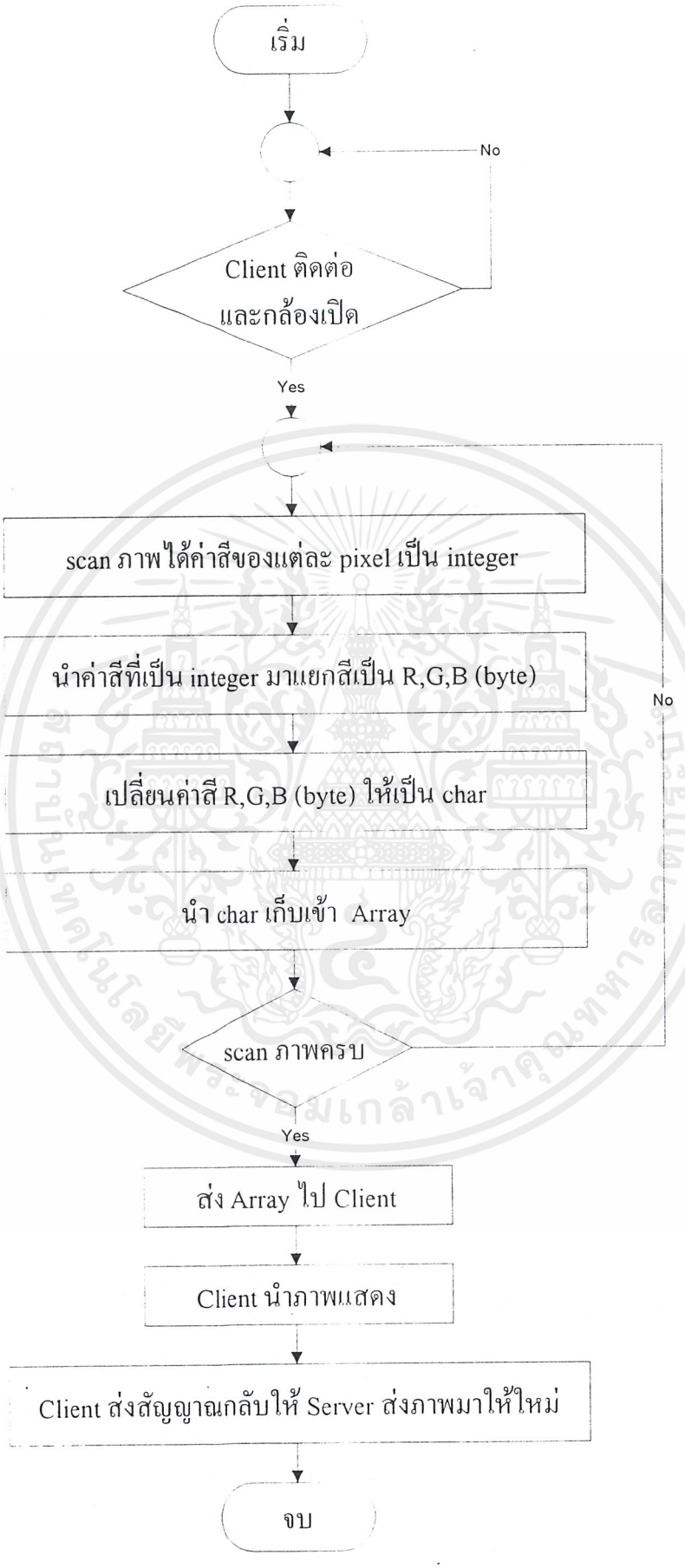
โปรแกรมที่ใช้ในโครงงานนี้จะเป็นโปรแกรมในส่วนที่ติดต่อกับผู้ใช้งานมีสองโปรแกรม คือ โปรแกรมทางด้านผู้ควบคุม(ผู้ใช้งาน) และโปรแกรมทางด้านผู้ให้บริการ (Server) โดยเริ่มจากโปรแกรมในส่วนที่เป็นผู้ใช้ก่อนดังนี้

6.3.1 โปรแกรมควบคุมด้านผู้ใช้ (Client Side Program)

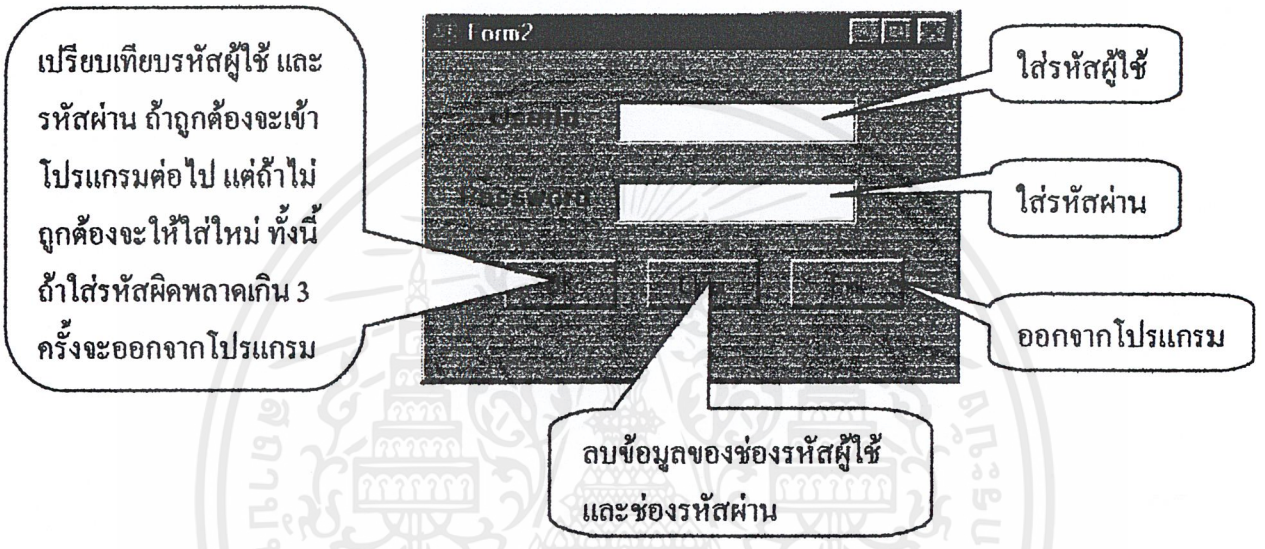
เริ่มจากการออกแบบโครงสร้างของโปรแกรมนี้ก่อน โดยคิดจากฟังก์ชั่นการทำงานทั้งหมดที่จำเป็นในการควบคุมการหมุนของกล้องโดยมีการทำงานดังนี้

เริ่มจากการที่ Client ติดต่อไปยัง Server โดยระบุ IP Address ทางด้าน Server เมื่อติดต่อได้แล้ว ก็จะรอการรับภาพ โดยวงเช็คใน Array ว่าครบตามที่กำหนดหรือยัง เมื่อครบแล้วก็จะทำการเปลี่ยนจาก Character เป็น byte แล้วรวมแต่ละค่าของ R,G,B จากนั้นก็นำขึ้นแสดงภาพ นอกจากนี้ยังมีส่วนของการควบคุมทิศทางการหมุน ซึ่งสามารถควบคุมกล้องให้เคลื่อนที่ไปตามทิศทางที่กำหนดได้ สำหรับแผนผังแสดงการนำภาพขึ้นจะเป็นดังรูป 6.4

นอกจากนี้เรายังเพิ่มระบบรักษาความปลอดภัยโดยที่ผู้ที่รู้ Password เท่านั้นที่จะสามารถเข้าไปใช้โปรแกรมได้มีการวางรูปแบบดังรูปที่ 6.5



รูปที่ 6.4 แสดงการทำงานของโปรแกรมส่งภาพ

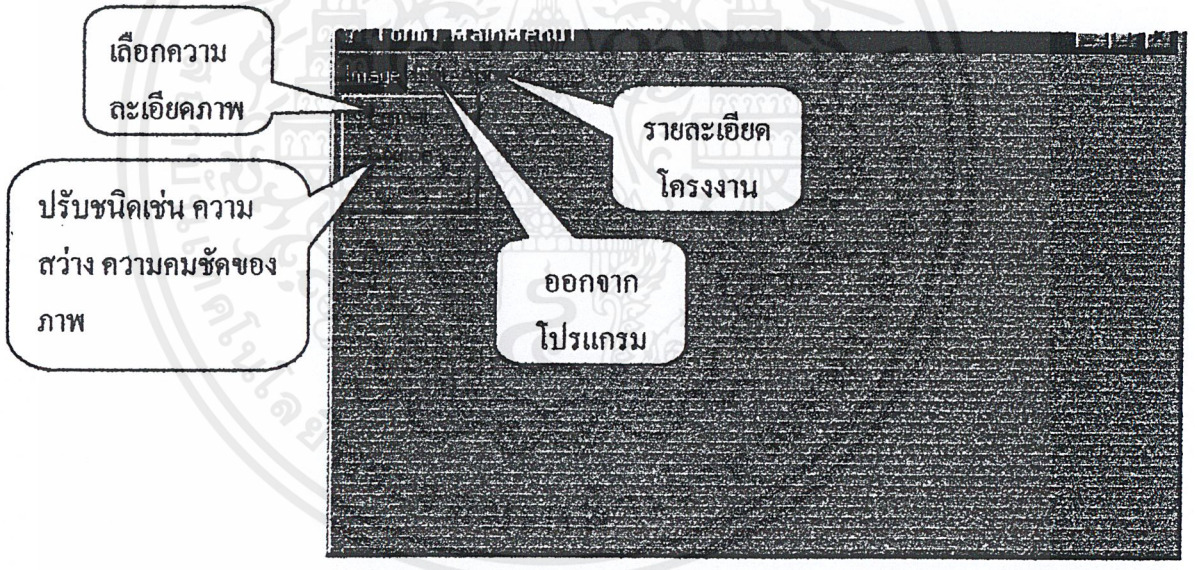


รูปที่ 6.5 แสดงการวางรูปแบบของโปรแกรมด้านความปลอดภัย

6.3.2 โปรแกรมทางด้านผู้ให้บริการ (Server Side Program)

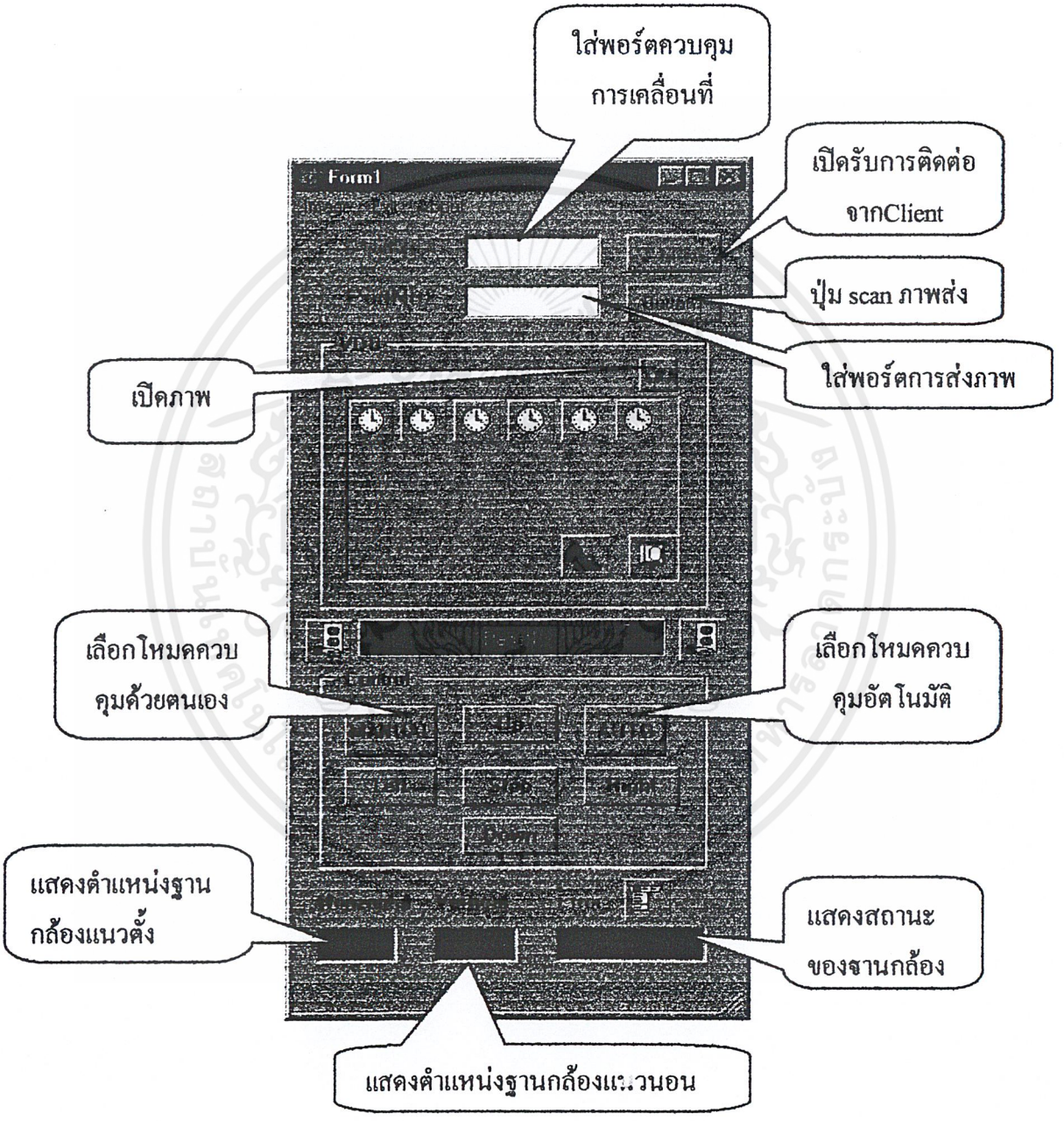
ส่วนนี้เป็นส่วนที่เป็นตัวจัดการเกี่ยวกับการควบคุม การรับส่งข้อมูล การจัดเกี่ยวกับภาพที่ส่งไปยังจุดหมายปลายทาง โดยมีชุดจับกล้องรวมอยู่ที่ส่วนนี้ด้วย โปรแกรมที่ใช้ควบคุมในส่วนนี้ จะเป็นตัวควบคุมคอยติดต่อกับชุดจับกล้องโดยผ่านทางพอร์ทอนุกรม รอรับภาพจากกล้องวีดีโอเข้ามา ต่อมาก็จะเป็นการรับคำสั่งจากผู้ใช้แล้วรายงานผลตำแหน่งของมอเตอร์ และแสดง Error ขึ้นมา และสุดท้ายจะเป็นโปรแกรมการติดต่อผ่านอินเตอร์เน็ต ซึ่งจะใช้ Component ที่มีใน delphi 5 แล้ว โดยใช้ TClientSocket และ TserverSocket ทั้งสองตัวนี้จะเป็นตัวที่ทำการรับคำสั่งควบคุม ส่งข้อมูลตำแหน่งของมอเตอร์ไปยังผู้สั่งงาน และทำการรับส่งภาพด้วย

การวางรูปแบบส่วนเมนูมีดังนี้

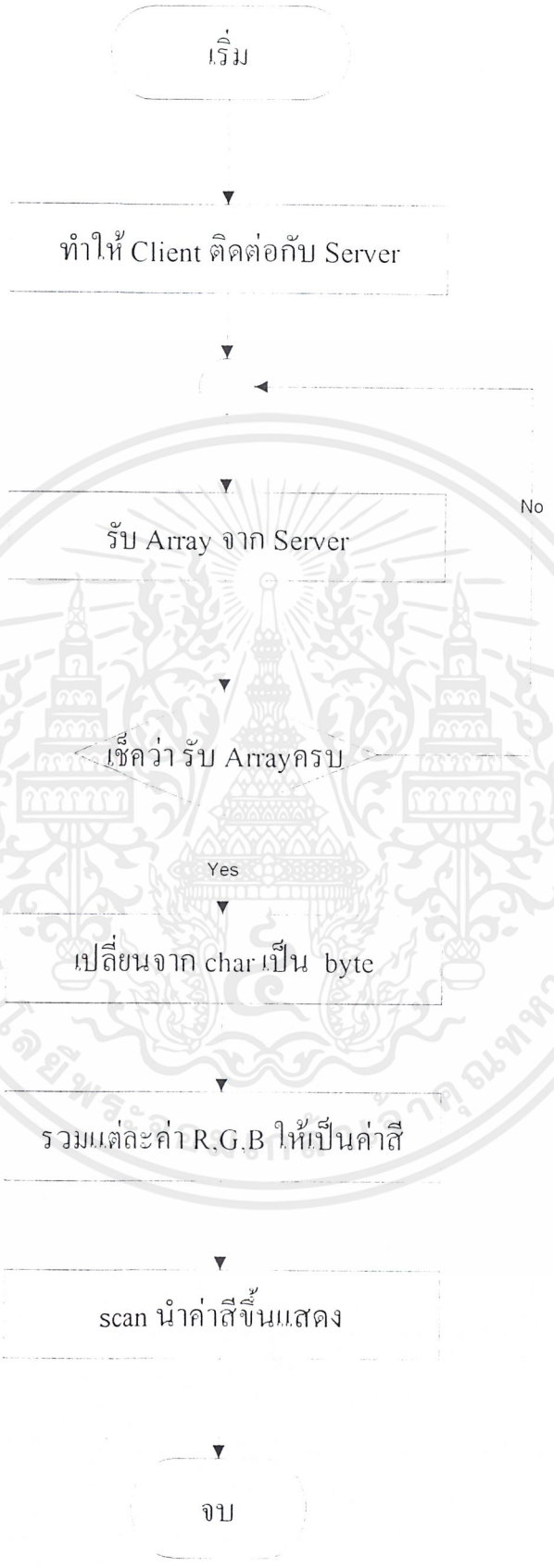


รูปที่ 6.6 แสดงการวางรูปแบบของเมนู

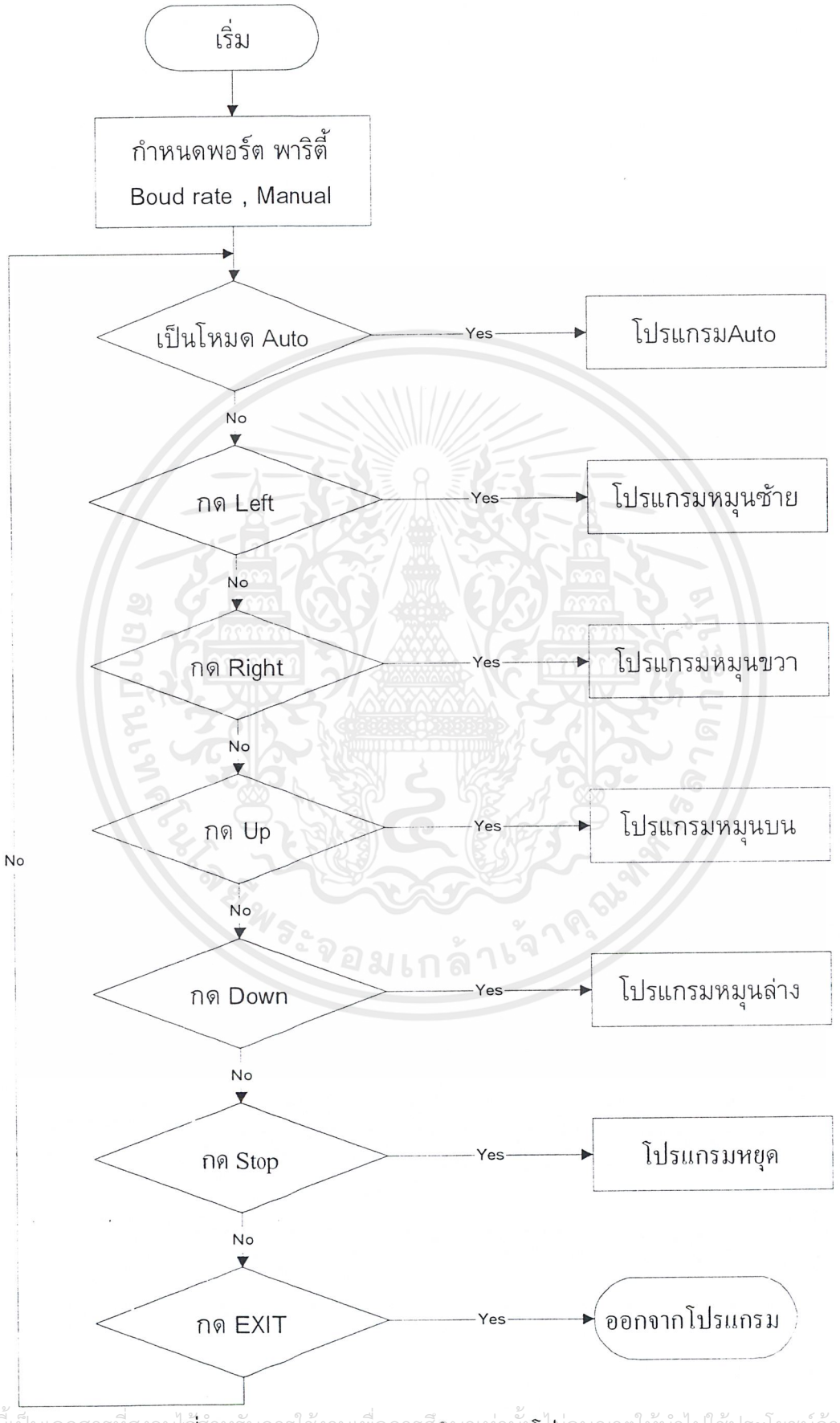
เมื่อเขียนโปรแกรมทั้งหมดแล้วส่วนนี้จะมีการวางรูปแบบดังรูป



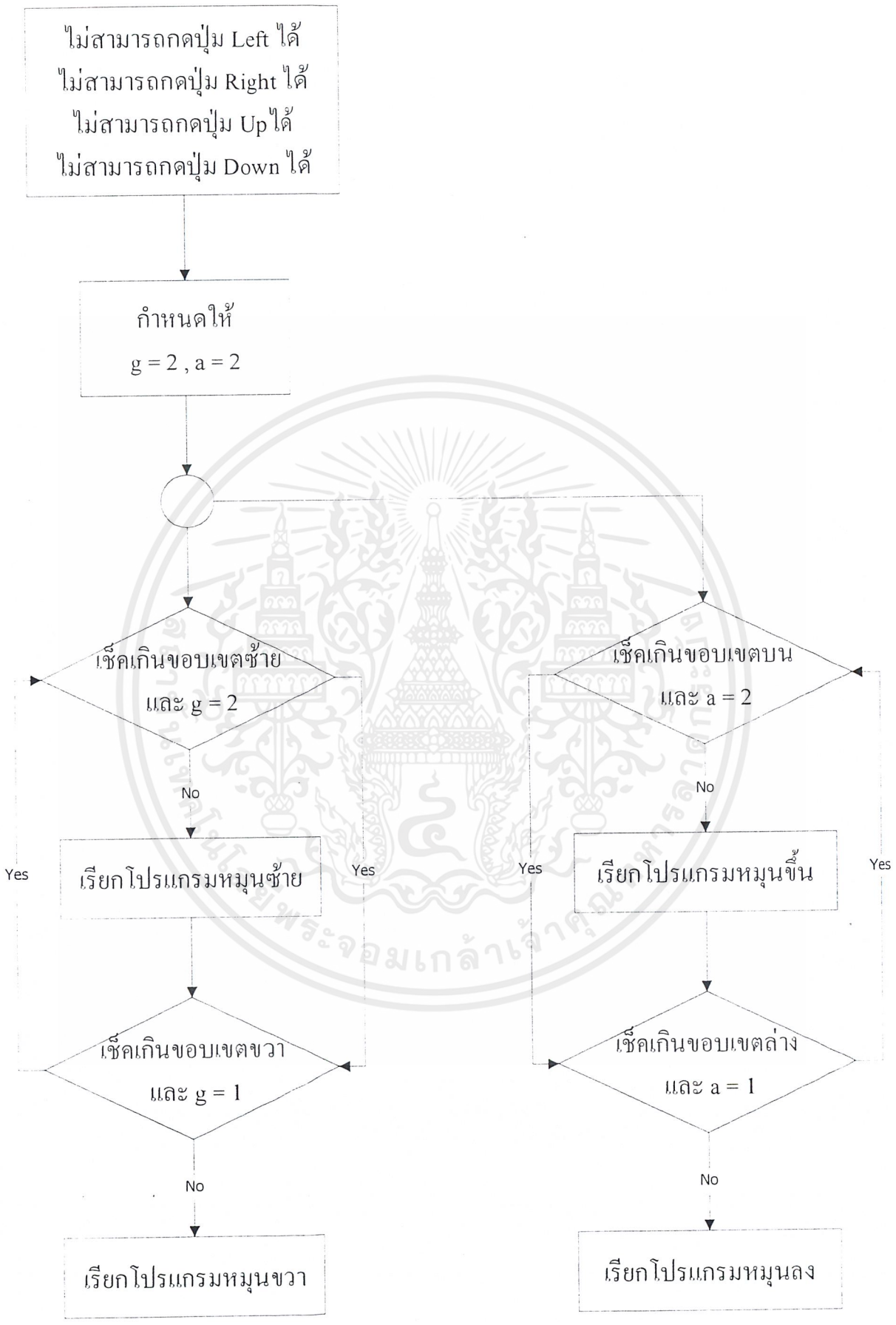
รูปที่ 6.7 แสดงการวางรูปแบบของโปรแกรมด้าน Server



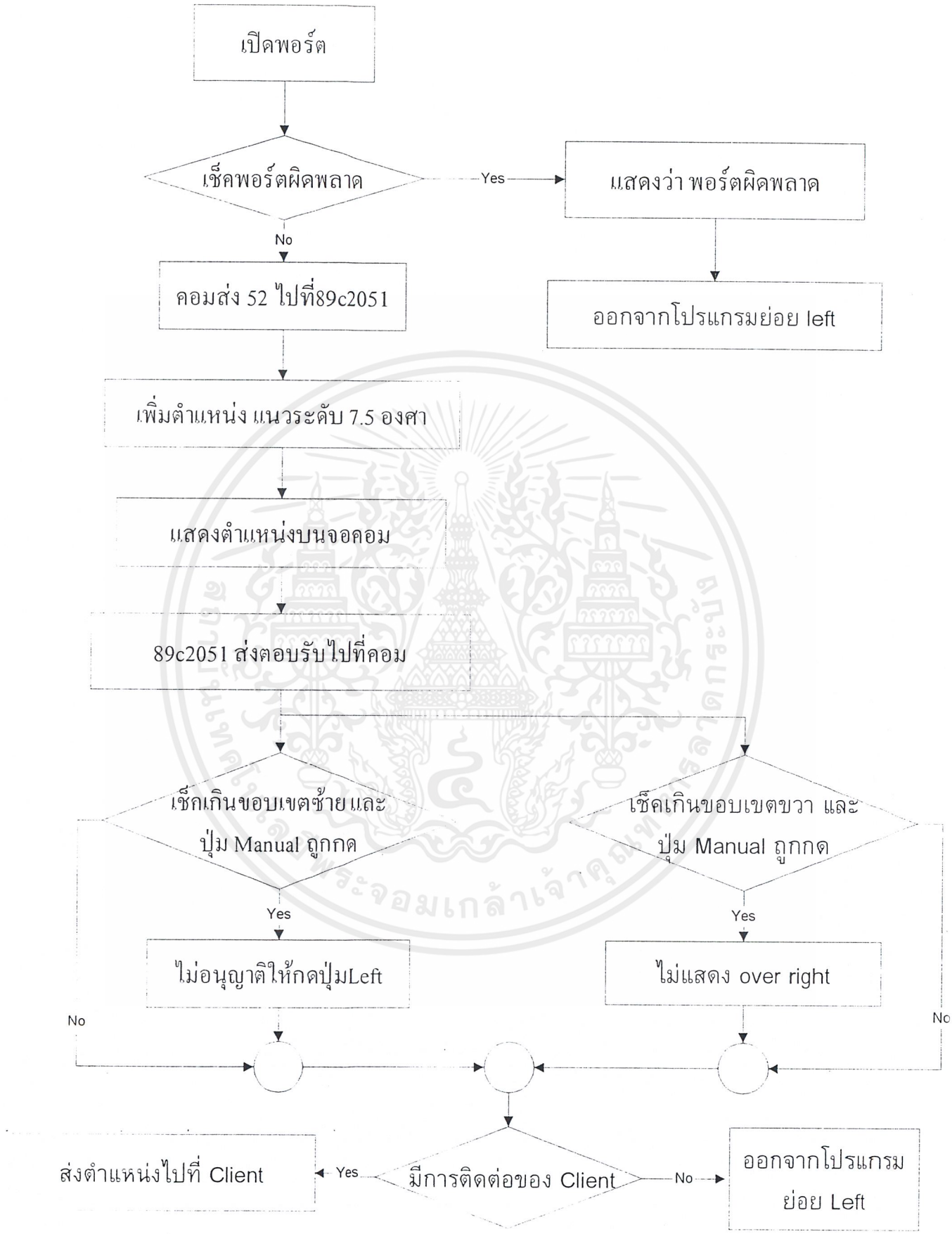
รูปที่ 6.8 แสดงการทำงานของโปรแกรมรับภาพ



รูปที่ 6.9 Flow Chart แสดงการทำงานของโปรแกรม MANUAL



รูปที่ 6.10 Flowchart แสดงการ Control ในส่วนของ Auto



รูปที่ 6.11 Flowchart แสดงโปรแกรมย่อยของการหมุนซ้าย

(ส่วนการหมุนด้านอื่นๆก็ทำเช่นเดียวกัน)

บทที่ 7
ผลการทดลอง สรุป และวิจารณ์

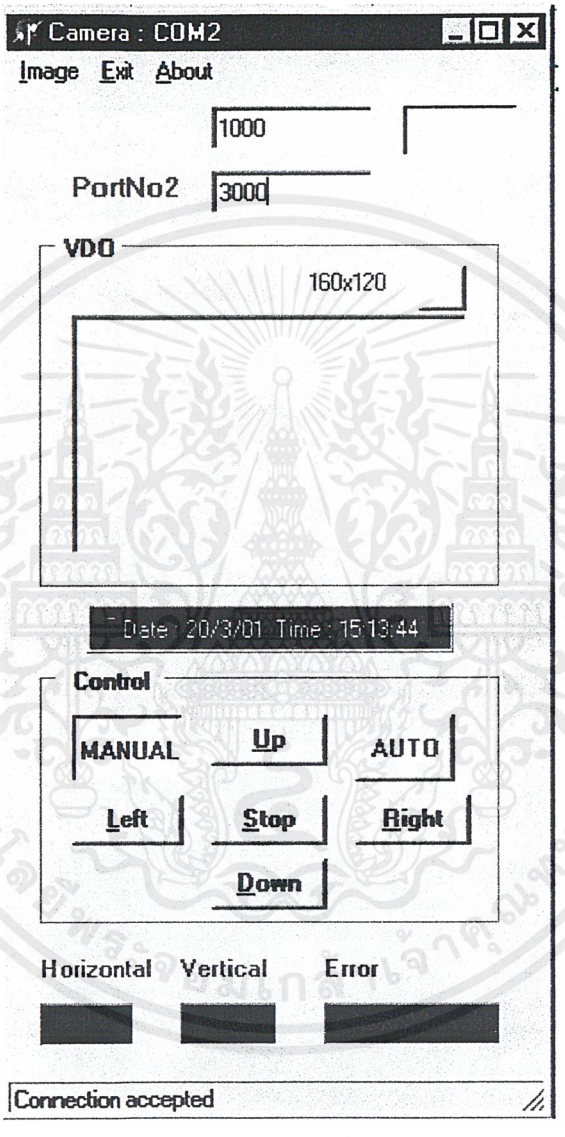
7.1 การทดลองในส่วนของการควบคุมกล้อง

สำหรับการทดลองในส่วนนี้เราทำการสำรวจผลการทดลองโดยดูจาก การหมุนของกล้อง และการแสดงผลจาก LED โดยถ้าหมุนซ้าย ไฟดวงที่1จะสว่าง หมุนขวา ไฟดวงที่2จะสว่าง หมุนขึ้น ไฟดวงที่3 จะสว่าง หมุนลง ไฟดวงที่4 จะสว่าง ได้ผลดังนี้

กดปุ่ม	กล้องหมุนทางด้าน	ไฟดวงที่ 1	ไฟดวงที่ 2	ไฟดวงที่ 3	ไฟดวงที่ 4
ซ้าย	ซ้าย	สว่าง	ดับ	ดับ	ดับ
ซ้าย	ซ้าย	สว่าง	ดับ	ดับ	ดับ
ขวา	ขวา	ดับ	สว่าง	ดับ	ดับ
ขวา	ขวา	ดับ	สว่าง	ดับ	ดับ
บน	บน	ดับ	ดับ	สว่าง	ดับ
บน	บน	ดับ	ดับ	สว่าง	ดับ
ล่าง	ล่าง	ดับ	ดับ	ดับ	สว่าง
ล่าง	ล่าง	ดับ	ดับ	ดับ	สว่าง
ซ้าย	ซ้าย	สว่าง	ดับ	ดับ	ดับ
บน	บน	ดับ	ดับ	สว่าง	ดับ
ขวา	ขวา	ดับ	สว่าง	ดับ	ดับ
ล่าง	ล่าง	ดับ	ดับ	ดับ	สว่าง
ขวา	ขวา	ดับ	สว่าง	ดับ	ดับ
บน	บน	ดับ	ดับ	สว่าง	ดับ
ล่าง	ล่าง	ดับ	ดับ	ดับ	สว่าง
ซ้าย	ซ้าย	สว่าง	ดับ	ดับ	ดับ

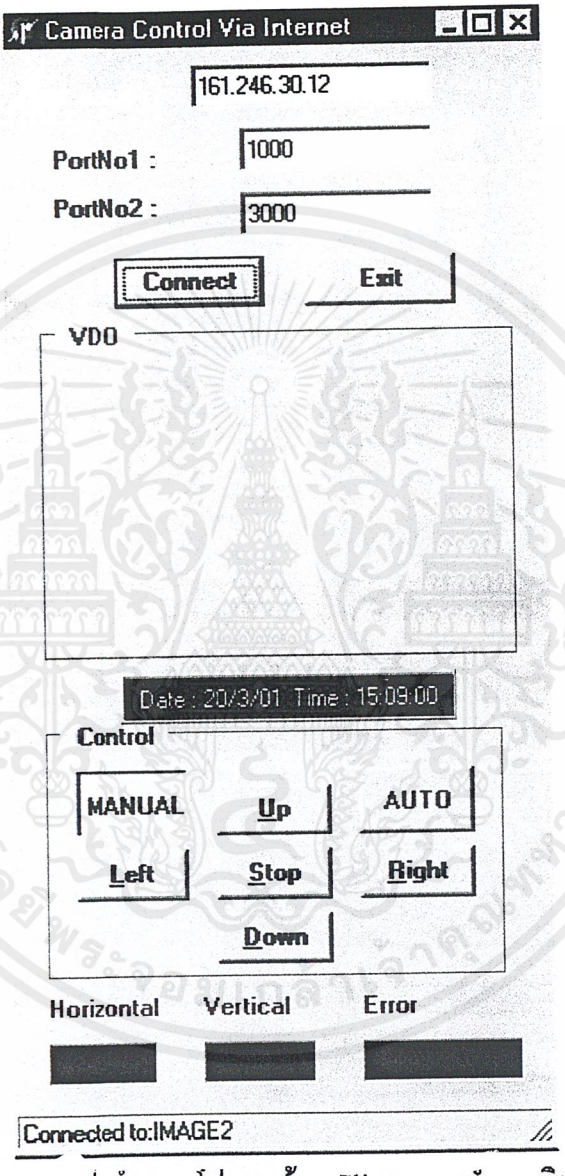
7.2 การทดลองการรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์บนเครือข่ายอินเตอร์เน็ต

ทางด้าน Server จะทำการใส่หมายเลขพอร์ต ทั้ง 2 แห่งแล้วทำการกดปุ่ม Listen เพื่อรอการ Connect จาก Client แสดงดังรูปที่ 7.1



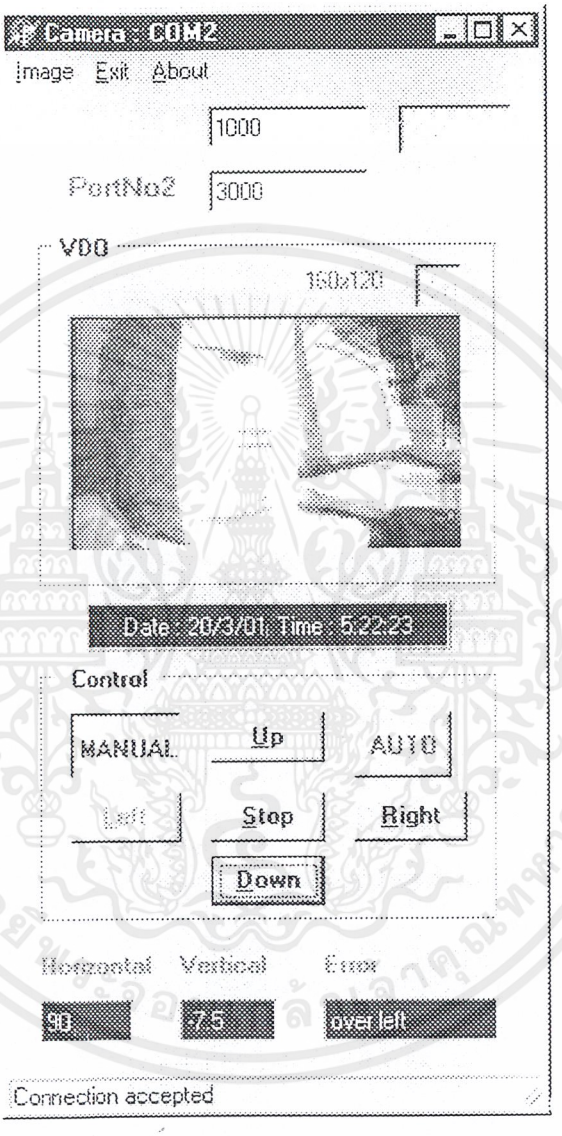
รูปที่ 7.1 แสดงฟอร์มของโปรเจคด้าน Server ขณะรอการติดต่อ

ทางด้าน Client ผู้ใช้ต้องกำหนด ไอพีแอดเดรสของเครื่องที่ต้องการติดต่อ และใส่หมายเลขของพอร์ตที่ต้องการใช้ในการสื่อสาร เมื่อใส่เรียบร้อยแล้วก็จะกดปุ่ม Connect เพื่อทำการติดต่อไปยังด้าน Server แสดงดังรูป 7.2



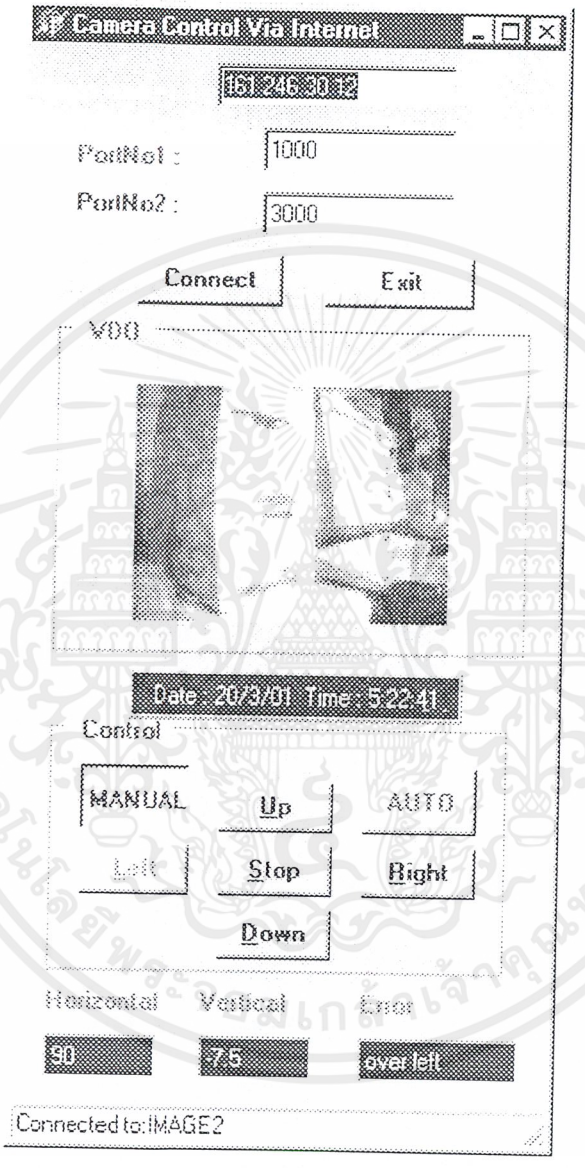
รูปที่ 7.2 แสดงฟอร์มของโปรเจคด้าน Client ขณะทำการติดต่อ

ทางด้าน Server เมื่อมีการติดต่อมาเรียบร้อยแล้วก็จะทำการส่งภาพไปให้ได้ และมีการควบคุมกล้องจากด้าน Client ได้ ดังรูป



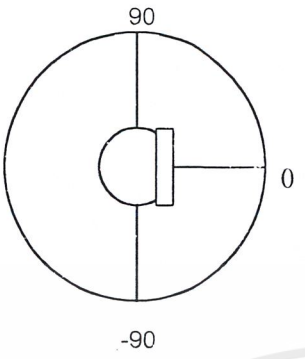
รูปที่ 7.3 แสดงฟอร์มของโปรเจคด้าน Server เมื่อทำการส่งภาพเรียบร้อยแล้ว

ส่วนทางด้าน Client เมื่อทำการติดต่อเรียบร้อยแล้วก็จะสามารถควบคุมกล้องได้ และยังสามารถรับภาพจากด้าน Server ได้อีกด้วย ดังรูป

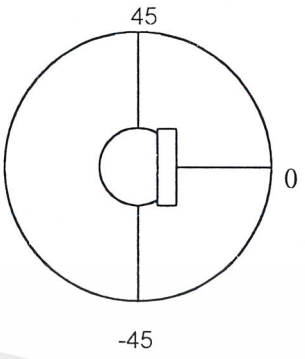


รูปที่ 7.4 แสดงฟอร์มของโปรเจคด้าน Client เมื่อทำการรับภาพเรียบร้อยแล้ว

7.3 การทดลองความถูกต้องของมุมที่หมุนของกล้อง



ขอบเขตซ้าย และขวา



ขอบเขตบน และล่าง

พิจารณาที่	ตำแหน่งที่แสดงบนจอคอมพิวเตอร์	ตำแหน่งฐานกล้องจริง
แนวระดับ	90	86
แนวระดับ	-90	-85
แนวดิ่ง	45	43
แนวดิ่ง	-45	-43

7.4 สรุปและวิจารณ์ผลการทดลอง

จากการทดลองสามารถควบคุมกล้องโดยผ่านทางอินเตอร์เน็ตได้ และสามารถส่งภาพจากด้าน Server มายังด้าน Client ได้อย่างถูกต้อง สำหรับปัญหาที่พบจากการทดลองนี้คือ ในการรับส่งไฟล์ที่เครื่องด้านรับ ต้องมีการเปิดโปรแกรมเพื่อรอการติดต่อก่อน เครื่องด้านส่งจึงจะสามารถทำการติดต่อมาได้

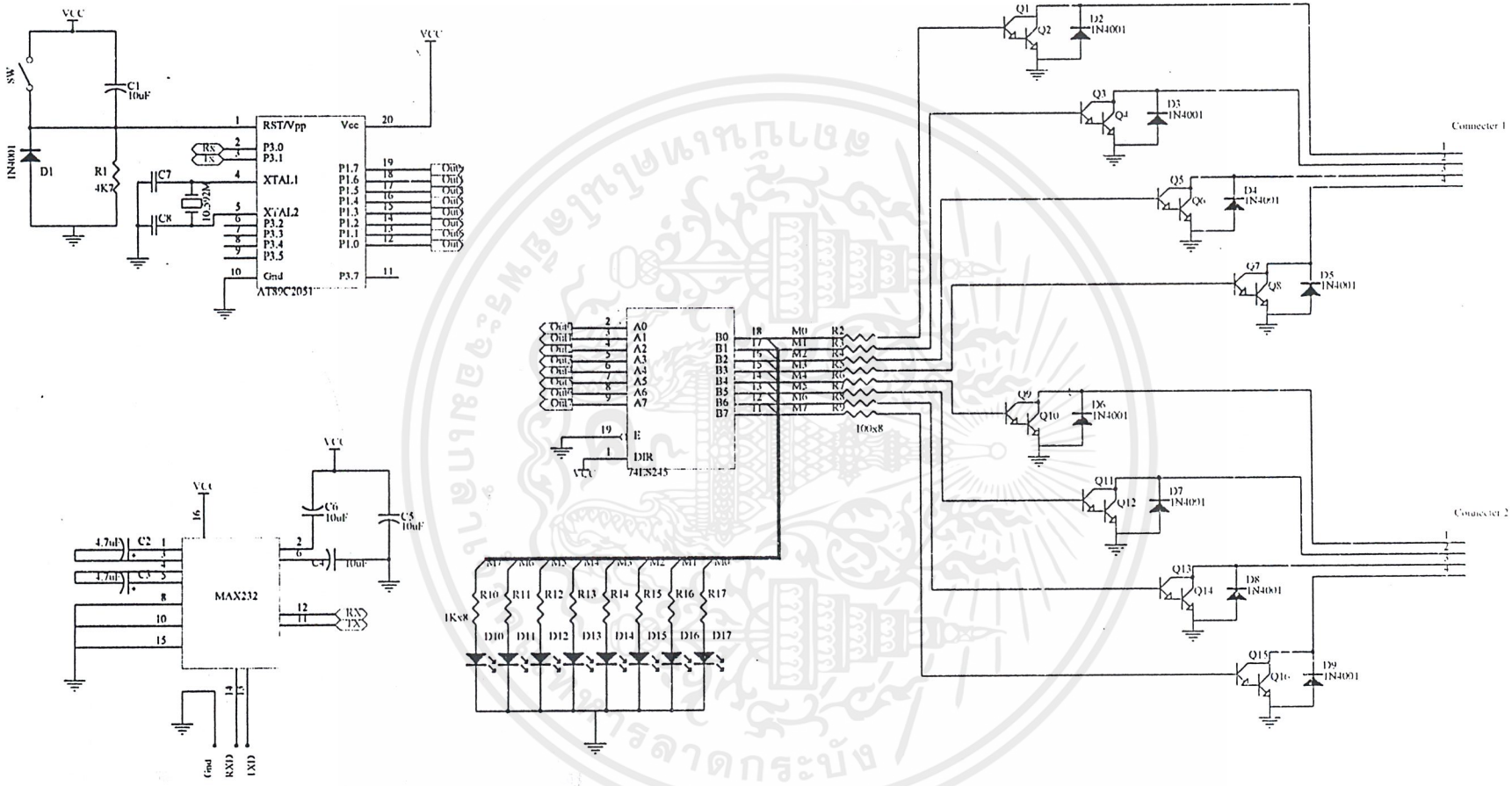
ในการสร้างโครงงานนี้ได้ทำให้เกิดทักษะมากมาย อาทิเช่น การค้นหาข้อมูลจากแหล่งต่างๆ การทำงานร่วมกัน การแก้ปัญหาต่างๆที่เกิดขึ้น การที่ได้ศึกษาความรู้เรื่องใหม่ๆที่ไม่มีในบทเรียน และเป็นการศึกษาในหลายๆด้าน ซึ่งสามารถนำไปประยุกต์ใช้ในงานอื่นต่อไปได้ ทำให้มีความรู้เพิ่มขึ้นมากมายจากเดิมที่มีอยู่

สำหรับแนวทางในการพัฒนาในอนาคตนั้น ควรจะปรับปรุงแก้ไขโปรแกรมเพื่อแก้ปัญหาที่เกิดขึ้น และพัฒนาโปรแกรมให้สามารถติดต่อสื่อสารกันได้อย่างสะดวก รวดเร็วยิ่งขึ้น

บรรณานุกรม

1. ธีรวัฒน์ ประกอบผล , การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ , กรุงเทพ : สมาคมส่งเสริมเทคโนโลยี (ไทย – ญี่ปุ่น)
2. กิตติ เปรมพินิจ , คู่มือการใช้งาน Borland Delphi 4 , ห้างหุ้นส่วนจำกัดเคพีเอ็น ซิสเต็มอินทิเกรเตอร์ , นครราชสีมา , 2541
3. กมลมาศ กำจรกิจการ , คู่มือ Borland Delphi 5 ,ฉบับสมบูรย์ , กรุงเทพ : โปรวิชั่น , 2543 , 520 หน้า
4. Tokashi Kenjo , 'Stepping Motor and their Microprocessor control', The University of press, 1986
5. Markus Pope , "programming Internet Control" , Prima Publishing , ISBN 0-7615-0773-6 PP.167-184
6. Pat Bonner , " Network Programming With Windows Socket", Prentice-Hall INC ISBN 0-13-2301152-0 ,PP.49-78

ภาคผนวก



รูปแสดงวงจรควบคุมการหมุนของกล้อง

Features

- Compatible with MCS-51™ Products
- 2K Bytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-chip Analog Comparator
- Low-power Idle and Power-down Modes

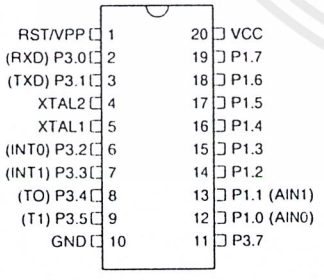
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

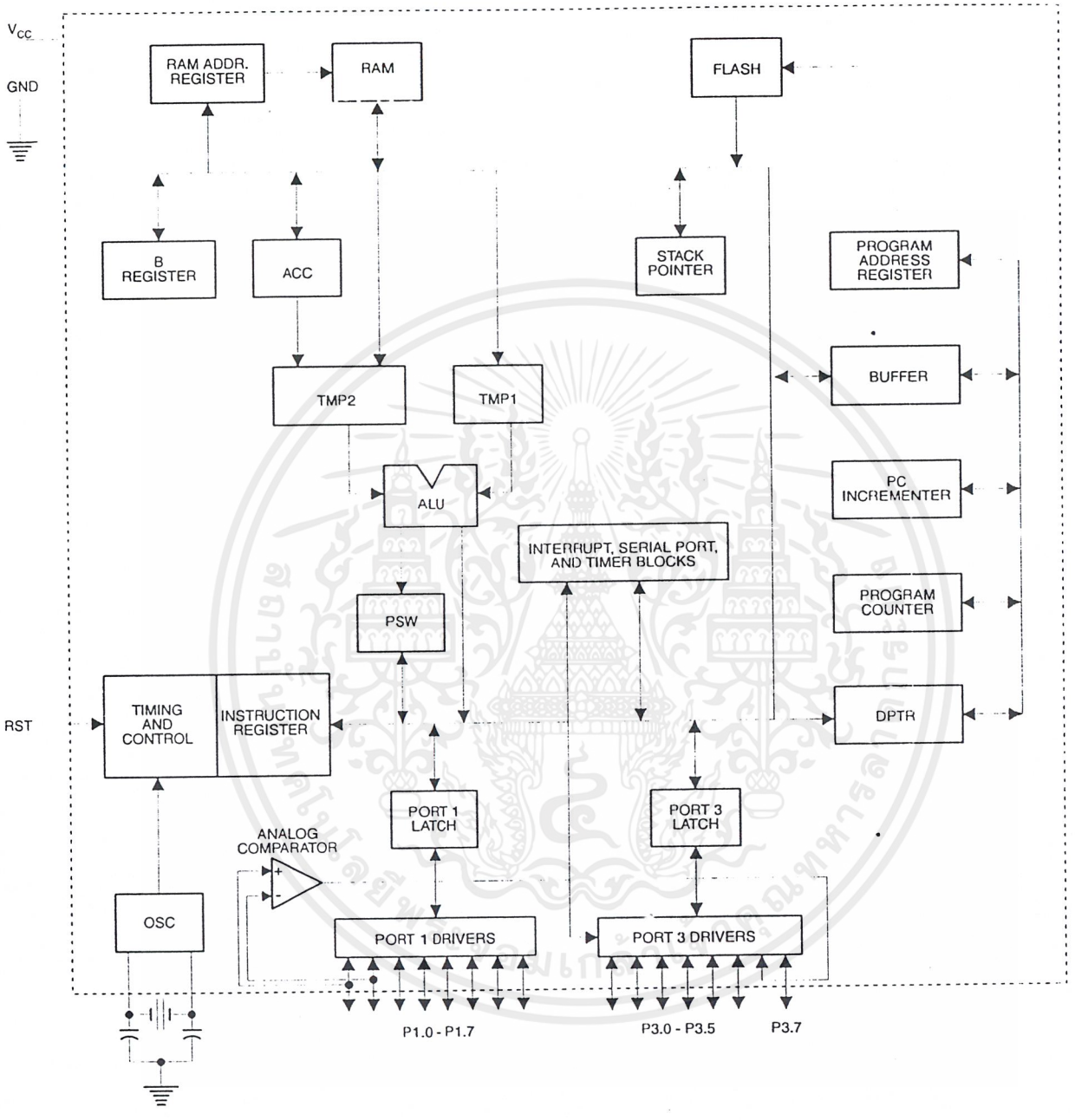


8-bit Microcontroller with 2K Bytes Flash

AT89C2051

Rev. 0368E–02/00

Block Diagram



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 1

Port 1 is an 8-bit bi-irectional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (I_{IL}) because of the internal pullups.

Port 1 also receives code data during Flash programming and verification.

Port 3

Port 3 pins P3.0 to P3.5, P3.7 are seven bi-irectional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a general purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89C2051 as listed below:

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)

Port 3 also receives some control signals for Flash programming and verification.

RST

Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the oscillator is running resets the device.

Each machine cycle takes 12 oscillator or clock cycles.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

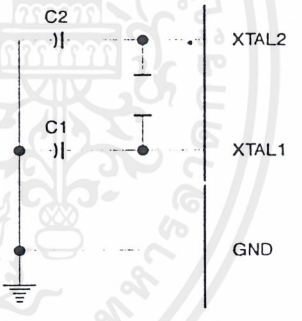
XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

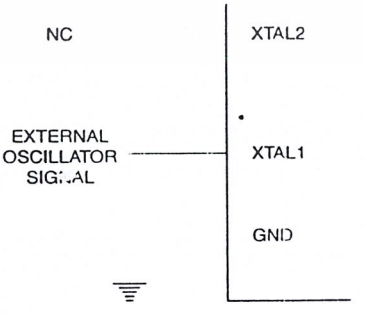
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



National Semiconductor

June 1989

54LS245/DM54LS245/DM74LS245 TRI-STATE® Octal Bus Transceiver

General Description

These octal bus transceivers are designed for asynchronous two-way communication between data buses. The control function implementation minimizes external timing requirements.

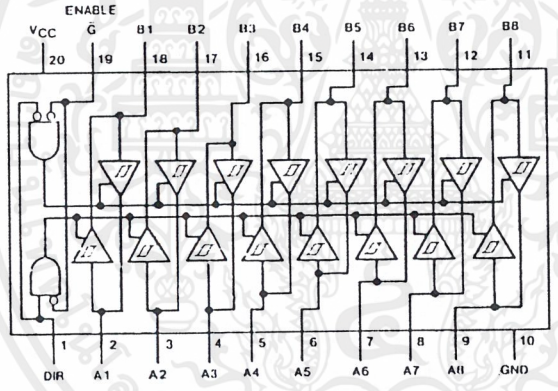
The device allows data transmission from the A bus to the B bus or from the B bus to the A bus depending upon the logic level at the direction control (DIR) input. The enable input ($\overline{G}$) can be used to disable the device so that the buses are effectively isolated.

Features

- Bi-Directional bus transceiver in a high-density 20-pin package
- TRI-STATE outputs drive bus lines directly
- PNP inputs reduce DC loading on bus lines
- Hysteresis at bus inputs improve noise margins
- Typical propagation delay times, port-to-port 8 ns
- Typical enable/disable times 17 ns
- I_{OL} (sink current)
  - 54LS 12 mA
  - 74LS 24 mA
- I_{OH} (source current)
  - 54LS 12 mA
  - 74LS 15 mA
- Alternate Military/Aerospace device (54LS245) is available. Contact a National Semiconductor Sales Office/ Distributor for specifications

Connection Diagram

Dual-In-Line Package



TL/F/6413-1

Order Number 54LS245DMQB, 54LS245FMQB, 54LS245LMQB, DM54LS245J, DM54LS245W, DM74LS245WM or DM74LS245N
See NS Package Number E20A, J20A, M20B, N20A or W20A

Function Table

Enable $\overline{G}$	Direction Control DIR	Operation
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

H = High Level, L = Low Level, X = Irrelevant

TRI-STATE® is a registered trademark of National Semiconductor Corporation

 TL/F/6413

RRD-B30M105/Printed in U.S.A.

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage	
DIR or $\overline{G}$	7V
A or B	5.5V
Operating Free Air Temperature Range	
DM54LS and 54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS245			DM74LS245			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−12			−15	mA
I_{OL}	Low Level Output Current			12			24	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions			Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA					−1.5	V
HYS	Hysteresis ($V_{T+} - V_{T-}$)	V_{CC} = Min			0.2	0.4		V
V_{OH}	High Level Output Voltage	V_{CC} = Min, V_{IH} = Min, V_{IL} = Max, I_{OH} = −1 mA		DM74	2.7			V
		V_{CC} = Min, V_{IH} = Min, V_{IL} = Max, I_{OH} = −3 mA		DM54/DM74	2.4	3.4		
		V_{CC} = Min, V_{IH} = Min, V_{IL} = 0.5V, I_{OH} = Max		DM54/DM74	2			
V_{OL}	Low Level Output Voltage	V_{CC} = Min, V_{IL} = Max, V_{IH} = Min	I_{OL} = 12 mA	DM74			0.4	V
			I_{OL} = Max	DM54			0.4	
				DM74			0.5	
I_{OZH}	Off-State Output Current, High Level Voltage Applied	V_{CC} = Max, V_{IL} = Max, V_{IH} = Min	V_O = 2.7V				20	μA
I_{OZL}	Off-State Output Current, Low Level Voltage Applied		V_O = 0.4V				−200	μA
I_I	Input Current at Maximum Input Voltage	V_{CC} = Max	A or B	V_I = 5.5V			0.1	mA
			DIR or $\overline{G}$	V_I = 7V			0.1	
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V					20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V					−0.2	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)			−40		−225	mA
I_{CC}	Supply Current	Outputs High		V_{CC} = Max		48	70	mA
		Outputs Low				62	90	
		Outputs at Hi-Z				64	95	

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, not to exceed one second duration.

Switching Characteristics V_{CC} = 5V, T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	Conditions	DM54/74 LS245		Units
			Min	Max	
t_{PLH}	Propagation Delay Time, Low-to-High-Level Output	C_L = 45 pF, R_L = 667Ω		12	ns
t_{PHL}	Propagation Delay Time, High-to-Low-Level Output			12	ns
t_{PZL}	Output Enable Time to Low Level			40	ns
t_{PZH}	Output Enable Time to High Level			40	ns
t_{PLZ}	Output Disable Time from Low Level	C_L = 5 pF, R_L = 667Ω		25	ns
t_{PHZ}	Output Disable Time from High Level			25	ns
t_{PLH}	Propagation Delay Time, Low-to-High-Level Output	C_L = 150 pF, R_L = 667Ω		16	ns
t_{PHL}	Propagation Delay Time, High-to-Low-Level Output			17	ns
t_{PZL}	Output Enable Time to Low Level			45	ns
t_{PZH}	Output Enable Time to High Level			45	ns

SOURCE CODE ของโปรแกรม

```
        ORG     0000H
OUT_M   EQU 30H
CON_M   EQU 31H,32H
POS_M   EQU 33H,34H
ERROR   EQU 35H
        MOV     PCON,#00H
        MOV     SCON,#50H
        MOV     TMOD,#20H
        MOV     TH1,#0FDH
        SETB    TR1
        SETB    PEN
        MOV     A,#11H
        MOV     R0,#CON_M
        MOV     @R0,A
        INC     R0
        MOV     A,#33H
        MOV     @R0,A
        MOV     A,#07FH
        MOV     R0,#POS_M
        MOV     @R0,A
        INC     R0
        MOV     @R0,A
        MOV     ERROR,#41H
        MOV     R3,#0H
        CLR     RI
        CLR     TI
        MOV     R4,#02H
WAIT:   MOV     P1,#0FFH
        LCALL   DELAY
        LCALL   DELAY
        MOV     P1,#0H
        LCALL   DELAY
        LCALL   DELAY
        DJNZ    R4,WAIT
MAIN:   JNB     RI,MAIN
        MOV     A,SBUF
        MOV     R3,A
        CLR     RI
MAIN1:  CJNE    R3,#36H,MAIN2
        LCALL   R_L
MAIN2:  CJNE    R3,#34H,MAIN3
        LCALL   R_L
MAIN3:  CJNE    R3,#38H,MAIN4
        LCALL   U_D
MAIN4:  CJNE    R3,#32H,MAIN
        LCALL   U_D
R_L:    MOV     R1,POS_M
        MOV     A,CON_M
        CJNE    R3,#36H,SUBT1
SUBT:   RL      A
        INC     R1
        LJMP    STORE
SUBT1:  RR      A
        DEC     R1
STORE:  MOV     ERROR,#0H
        MOV     CON_M,A
        MOV     POS_M,R1
        LCALL   DELAY
        LCALL   MOTOR
        MOV     A,POS_M
        MOV     SBUF,A
WAIT1:  JNB     TI,WAIT1
        CLR     TI
        MOV     R3,#0H
        RET
U_D:    MOV     R1,POS_M+1
        MOV     A,CON_M+1
        CJNE    R3,#38H,SUBT3
SUBT2:  RL      A
        INC     R1
        LJMP    STORE1
SUBT3:  RR      A
        DEC     R1
STORE1: MOV     ERROR,#0H
        MOV     CON_M+1,A
        MOV     POS_M+1,R1
        LCALL   DELAY
        LCALL   MOTOR
        MOV     A,POS_M+1
        MOV     SBUF,A
WAIT2:  JNB     TI,WAIT2
        CLR     TI
        MOV     R3,#0H
        RET
MOTOR:  PUSH    ACC
        PUSH    PSW
        MOV     R0,#CON_M
        MOV     OUT_M,@R0
        ANL     OUT_M,#0FH
        INC     R0
        MOV     A,#0F0H
        ANL     A,@R0
        ORL     A,OUT_M
        MOV     P1,A
WAIT3:  JNB     TI,WAIT3
        CLR     TI
        LCALL   DELAY
        LCALL   DELAY
        POP     PSW
        POP     ACC
        RET
DELAY:  MOV     R7,#0FFH
DELAY1: MOV     R6,#0FFH
        DJNZ    R6,$
        DJNZ    R7,DELAY1
        RET
        END
```

โปรแกรมการทำงานของไมโครคอลโทลเลอร์

```
unit client1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
  c:integer;
implementation
uses Client1_2, Client1_3;
{$R *.DFM}
procedure TForm2.Button1Click(Sender: TObject);
var a,b:string;
begin
    a:='kmitl';
    b:='kmitl' ;
    if (Edit1.text = a) and (Edit2.text=b)
    then
    begin
    form3.show;
    form2.Visible := false;
    end   else
    begin
    showmessage('Password Incorrect');
    c:=c+1
    end;
    if c=3 then
    begin
    showmessage('PLEASE TRY AGAIN');
    Application.Terminate; end;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
  Edit1.clear;
  Edit2.Clear;
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
    Application.Terminate;
end;
procedure TForm2.FormCreate(Sender: TObject);
begin
    Form2.Caption := 'Welcome' ;
end;
end.
```

โปรแกรมส่วนพาสเวิร์ด

```
unit Client1_2;
interface
uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls, jpeg, ExtCtrls;
type
  TForm3 = class(TForm)
    Button1: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Panel1: TPanel;
    Image1: TImage;
    Label6: TLabel;
    Label7: TLabel;
    Panel2: TPanel;
    Label11: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    procedure Button1Click(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
implementation
uses Client1_3;
{$R *.DFM}
procedure TForm3.Button1Click(Sender:
TObject);
begin
     FOrm1.Show;
     Form3.Visible:=false;
end;
procedure TForm3.FormCreate(Sender:
TObject);
begin
     Form3.Caption := 'About Project' ;
end;
end.
```

โปรแกรมส่วน **About Project**

```
unit Client1_3;
interface
uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, ScktComp, ExtCtrls,
ImageEnProc, ImageEnView,
  VideoCap, Buttons;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    ClientSocket: TClientSocket;
    Exit: TButton;
    GroupBox1: TGroupBox;
    Up: TButton;
    Left: TButton;
    Stop: TButton;
    Right: TButton;
    Down: TButton;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    Label5: TLabel;
    Edit5: TEdit;
    StatusBar1: TStatusBar;
    Connect: TButton;
    ClientSocket1: TClientSocket;
    Label6: TLabel;
    Edit6: TEdit;
    TmGetImage: TTimer;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    GroupBox2: TGroupBox;
    Panel1: TPanel;
    Timer1: TTimer;
    PaintBox1: TPaintBox;
    procedure ConnectClick(Sender:
TObject);
    procedure ExitClick(Sender: TObject);
    procedure ClientSocketConnect(Sender:
TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketError(Sender:
TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var
ErrorCode: Integer);
    procedure ClientSocketDisconnect
(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure StopClick(Sender: TObject);
    procedure UpClick(Sender: TObject);
    procedure DownClick(Sender: TObject);
    procedure LeftClick(Sender: TObject);
    procedure RightClick(Sender: TObject);
    procedure ClientSocketRead(Sender:
TObject; Socket: TCustomWinSocket);
    procedure ClientSocket1Connect(Sender:
TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket1Read(Sender:
TObject; Socket: TCustomWinSocket);
    procedure TmGetImageTimer(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender:
TObject);
    procedure SpeedButton1Click(Sender:
TObject);
```

```
  procedure SpeedButton2Click(Sender:
TObject);
 private
  { Private declarations }
   W:String;
   booFlag:boolean;
 public
  { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.ConnectClick(Sender:
TObject);
var HostName:ShortString;
  PortNo:integer;
  PortNo1:integer;
begin
 HostName:=Edit1.Text;
 PortNo:=StrToInt(Edit2.Text);
 ClientSocket.Host:=HostName;
 ClientSocket.Port:=PortNo;
 ClientSocket.Open;
 PortNo1:=StrToInt(Edit6.Text);
 ClientSocket1.Host:=HostName;
 ClientSocket1.Port:=PortNo1;
 ClientSocket1.Open;
end;
procedure TForm1.ExitClick(Sender:
TObject);
begin
 ClientSocket.Close;
 Application.Terminate;
end;
procedure TForm1.ClientSocketConnect
(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 Statusbar1.Panels[0].Text:='Connected
to:'+Socket.RemoteHost;
end;
procedure TForm1.ClientSocketError
(Sender: TObject;
 Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;
 var ErrorCode: Integer);
begin
 Statusbar1.Panels[0].Text:='Error';
end;
procedure TForm1.ClientSocketDisconnect
(Sender: TObject;
 Socket: TCustomWinSocket);
 begin
 Statusbar1.Panels[0].Text:='Now Server
Disconnect';
end;
procedure TForm1.StopClick(Sender:
TObject);
begin
 ClientSocket.Socket.SendText('7');
 SpeedButton1.down:=true;
 SpeedButton1.click;
end;
procedure TForm1.UpClick(Sender:
TObject);
begin
 ClientSocket.Socket.SendText('2');
end;
procedure TForm1.DownClick(Sender:
TObject);
```

```
begin
  ClientSocket.Socket.SendText('4');
end;
procedure TForm1.LeftClick(Sender:
TObject);
begin
  ClientSocket.Socket.SendText('6');
end;
procedure TForm1.RightClick(Sender:
TObject);
begin
  ClientSocket.Socket.SendText('8');
end;
procedure TForm1.ClientSocketRead
(Sender: TObject;
  Socket: TCustomWinSocket);
var S:String;
begin
  S:=Socket.ReceiveText;
  case S[1] of
   'H':begin delete(S,1,1); Edit3.Text:=S;
end;
   'V':begin delete(S,1,1); Edit4.Text:=S;
end;
   'E':begin delete(S,1,1); Edit5.Text:=S;
end;
  end;
  if edit5.Text='over right'
  then Right.Enabled:=false
  else if Speedbutton2.Down = false then
Right.Enabled:=true;
  if edit5.Text='over left'
  then Left.Enabled:=false
  else if Speedbutton2.Down = false then
Left.Enabled:=true;
  if edit5.Text='over up'
  then Up.Enabled:=false
  else if Speedbutton2.Down = false then
Up.Enabled:=true;
  if edit5.Text='over down'
  then Down.Enabled:=false
  else if Speedbutton2.Down = false then
Down.Enabled:=true;
end;
procedure TForm1.ClientSocket1Connect
(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  TmGetImage.Enabled:=true;
  W:='';
end;
  procedure TForm1.ClientSocket1Read
  (Sender: TObject;
    Socket: TCustomWinSocket);
  begin
    W:=W+socket.ReceiveText;
  if length(W)=56763 then booFlag:=true
    else booFlag:=false;
  end;
  procedure TForm1.TmGetImageTimer
  (Sender: TObject);
  type
    TRGB = packed record
      b: byte;
      g: byte;
      r: byte;
    end;
    frame3 = array [1..56763] of byte;
var  array1:array[1..56763] of char;
    i:integer;
    x,y,g : integer;
    pixframe3 : frame3;
```

```
begin
   TmGetImage.Enabled := false;
   if booFlag then
   begin
      for i:=1 to 56763 do
        begin
         array1[i]:=w[i];
         pixframe3[i]:=byte(Array1[i]);
        end;
      g:=1;
      for y:=1 to 119 do
        begin
          for x:= 1 to 159 do
            begin
            PaintBox1.Canvas.Pixels
[x,y]:= RGB(pixframe3[g],pixframe3
[g+1],pixframe3[g+2]);
            g:=g+3;
            end;
        end;
      W := '';
      ClientSocket1.Socket.SendText
('Comp');
   end;
   booFlag := false;
   TmGetImage.Enabled := true;
end;
procedure TForm1.FormCreate(Sender:
TObject);
begin
   booFlag:=false;
   SpeedButton1.Down := true;
   Form1.Caption := 'Camera Control Via
Internet';
end;
procedure TForm1.Timer1Timer(Sender:
TObject);
begin
   Panel1.Caption:='Date : '+Datetostr(Now)+'
Time : '+timetostr(Now);
end;
procedure TForm1.SpeedButton1Click
(Sender: TObject);
begin
   ClientSocket.Socket.SendText('3');
   Left.Enabled:=true;
   Right.Enabled:=true;
   Up.Enabled:=true;
   Down.Enabled:=true;
end;
procedure TForm1.SpeedButton2Click
(Sender: TObject);
begin
   ClientSocket.Socket.SendText('5');
   Left.Enabled:=false;
   Right.Enabled:=false;
   Up.Enabled:=false;
   Down.Enabled:=false;
end;
end.
```

โปรแกรมส่วน Client

```
unit Server3;
interface
uses
  Display,
  Windows, Messages, SysUtils, WinTypes,
WinProcs, Classes, Graphics, Controls,
  Forms, Dialogs, wsc, ExtCtrls, StdCtrls,
IEOpenSaveDlg, ImageEnIO,
  ImageEnProc, ImageEnView, VideoCap,
Menus, ComCtrls,  ScktComp,
  Buttons;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    ServerSocket: TServerSocket;
    GroupBox1: TGroupBox;
    Up: TButton;
    Stop: TButton;
    Left: TButton;
    Right: TButton;
    Down: TButton;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    StatusBar1: TStatusBar;
    Speedbutton1: TSpeedButton;
    Speedbutton2: TSpeedButton;
    GroupBox2: TGroupBox;
    Panel1: TPanel;
    ImageEnVideoView1:
  TImageEnVideoView;
    Timer1: TTimer;
    Timer2: TTimer;
    Timer3: TTimer;
    Timer4: TTimer;
    Timer5: TTimer;
    ImageEnProc1: TImageEnProc;
    ImageEnIO1: TImageEnIO;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    MainMenu1: TMainMenu;
    In1: TMenuItem;
    Format1: TMenuItem;
    Source1: TMenuItem;
    Exit1: TMenuItem;
    Edit5: TEdit;
    Label5: TLabel;
    ServerSocket1: TServerSocket;
    Label6: TLabel;
    ImageEnView1: TImageEnView;
    About1: TMenuItem;
    Button1: TButton;
    Timer6: TTimer;
    procedure LeftClick(Sender: TObject);
    procedure ServerSocketAccept(Sender:
  TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientError
  (Sender: TObject;
      Socket: TCustomWinSocket;
  ErrorEvent: TErrorEvent;
      var ErrorCode: Integer);
    procedure ServerSocketListen(Sender:
  TObject;
      Socket: TCustomWinSocket);
    procedure StopClick(Sender: TObject);
    procedure RightClick(Sender: TObject);
    procedure UpClick(Sender: TObject);
```

```
    procedure DownClick(Sender: TObject);
    procedure ServerSocketClientRead
(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure SpeedButton1Click(Sender:
TObject);
    procedure SpeedButton2Click(Sender:
TObject);
//    procedure Timer5Timer(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender:
TObject);
    procedure Timer2Timer(Sender:
TObject);
 //  procedure Timer3Timer(Sender:
TObject);
  // procedure Timer4Timer(Sender:
TObject);
    procedure SpeedButton3Click(Sender:
TObject);
    procedure SpeedButton4Click(Sender:
TObject);
    procedure Format1Click(Sender:
TObject);
    procedure Source1Click(Sender:
TObject);
    procedure ServerSocket1ClientRead
(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure Exit1Click(Sender: TObject);
    procedure About1Click(Sender:
TObject);
    procedure Button1Click(Sender:
TObject);
    procedure Timer6Timer(Sender:
TObject);
    procedure ServerSocketClientDisconnect
(Sender: TObject;
      Socket: TCustomWinSocket);
  private
  ConnectSocket:TCustomWinSocket;
    { Private declarations }
    Port : Integer;
    Baud : Integer;
    Parity : Integer;
    DataBits : Integer;
    StopBits : Integer;
    Array1:array[1..56763]of char;
  public
    { Public declarations }
    procedure DisplayVideoSize;
    procedure ImageEnProc(Sender: TObject;
  Bitmap: TBitmap );
    end;
  var
    Form1: TForm1;
    chspig:integer;
    count_LR : real;
    count_UD : real;
    test :integer;
    scan,gade,ao:integer;
  implementation
  uses server3_2;
  {$R *.DFM}
  procedure TForm1.ServerSocketAccept
  (Sender: TObject;
   Socket: TCustomWinSocket);
  begin
   ConnectSocket:=Socket;
```

```
 Statusbar1.Panels[0].Text:='Connection
accepted';
 if (scan=2) and (speedbutton4.down) then
button1.click;
end;
procedure TForm1.ServerSocketClientError
(Sender: TObject;
 Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;
var ErrorCode: Integer);
begin
 Statusbar1.Panels[0].Text:='Error';
end;
procedure TForm1.ServerSocketListen
(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 Statusbar1.Panels[0].Text:='Listening...';
end;
procedure TForm1.StopClick(Sender:
TObject);
begin
 SpeedButton1.Down := true;
 SpeedButton1.Click;
 timer2.Enabled := false;
 timer3.Enabled := false;
 timer4.Enabled := false;
 timer5.Enabled := false;
end;
procedure TForm1.LeftClick(Sender:
TObject);
var S:String;
   code,check: Integer;
begin
 code := SioReset(Port,2048,512);
 if code < 0 then
begin
 showmessage('PORT COM Error');
 timer2.Enabled := false;
 timer3.Enabled := false;
 timer4.Enabled := false;
 timer5.Enabled := false;
 exit;
end;
sioBaud(Port,Baud);
sioparms(Port,parity,stopbits,databits);
siodtr(Port,'S');
siorts(Port,'S');
if count_LR < 90 then
begin
 SioPutc(Port,chr(52));
 count_LR := count_LR+7.5;
end;
 edit2.text:=floattostr(count_LR);
 if count_LR = 90 then gade:=1;
 check := 0;
 repeat
 code := SioGetc(Port);
 check := check+1;
 if check > 10000 then  break;
 until code >= 0 ;
 if count_LR >= 90 then
 begin
if SpeedButton2.Down = false then
left.Enabled:=false;
   edit4.text:='over left';
   end
   else if count_LR >= -90 then
   begin
    if SpeedButton2.Down = false then
right.Enabled:=true;
    edit4.text:='';
```

```
  end;
  SioDone(Port);
  if (speedbutton3.Down = true) and
(Statusbar1.Panels[0].Text='Connection
accepted') then
  begin
   S:='H'+Edit2.Text;
   ServerSocket.Socket.Connections
[0].SendText(S);
   S:='E'+Edit4.Text;
   ServerSocket.Socket.Connections
[0].SendText(S);
  end;
end;
procedure TForm1.RightClick(Sender:
TObject);
var S:String;
  code,check: Integer;
begin
 code := SioReset(Port,2048,512);
 if code < 0 then
 begin
  showmessage('PORT COM Error');
  timer2.Enabled := false;
  timer3.Enabled := false;
  timer4.Enabled := false;
  timer5.Enabled := false;
  exit;
 end;
 sioBaud(Port,Baud);
 sioparms(Port,parity,stopbits,databits);
 siodtr(Port,'S');
 siorts(Port,'S');
 if count_LR > -90 then
 begin
  SioPutc(Port,chr(54));
   count_LR:= count_LR-7.5;
 end;
 edit2.text:=floattostr(count_LR);
 if count_LR = -90 then gade:=2;
 check := 0;
 repeat
 code := SioGetc(Port);
 check := check+1;
 if check > 10000 then  break;
 until code >= 0 ;
 if count_LR <= -90 then
 begin
  if SpeedButton2.Down = false then
right.Enabled:=false;
  edit4.text:='over right';
 end
 else if count_LR <= 90 then
 begin
  if SpeedButton2.Down = false then
left.Enabled:=true;
  edit4.text:='';
 end ;
 SioDone(Port);
 if (speedbutton3.Down = true) and
(Statusbar1.Panels[0].Text='Connection
accepted') then
 begin
  S:='H'+Edit2.Text;
  ServerSocket.Socket.Connections
[0].SendText(S);
  S:='E'+Edit4.Text;
  ServerSocket.Socket.Connections
[0].SendText(S);
 end;
end;
```

```
procedure TForm1.UpClick(Sender:
TObject);
var S:String;
   code,check : Integer;
begin
 code := SioReset(Port,2048,512);
 if code < 0 then
 begin
  showmessage('PORT COM Error');
  timer2.Enabled := false;
  timer3.Enabled := false;
  timer4.Enabled := false;
  timer5.Enabled := false;
  exit;
 end;
 sioBaud(Port,Baud);
 sioparms(Port,parity,stopbits,databits);
 siodtr(Port,'S');
 siorts(Port,'S');
 if count_UD < 45 then
 begin
  SioPutc(Port,chr(50));
  count_UD:= count_UD+7.5;
 end;
 edit3.text:=floattostr(count_UD);
 if count_UD=45 then ao:=1;
 check := 0;
 repeat
 code := SioGetc(Port);
 check := check+1;
 if check > 10000 then  break;
 until code >= 0 ;
 if count_UD >= 45 then
 begin
  if SpeedButton2.Down = false then
up.Enabled:=false;
  edit4.text:='over up';
 end
 else if count_UD >= -45 then
 begin
  if SpeedButton2.Down = false then
down.Enabled:=true;
  edit4.text:='';
 end  ;
 SioDone(Port);
 if (speedbutton3.Down = true) and
(Statusbar1.Panels[0].Text='Connection
accepted') then
  begin
   S:='V'+Edit3.Text;
   ServerSocket.Socket.Connections
 [0].SendText(S);
   S:='E'+Edit4.Text;
   ServerSocket.Socket.Connections
 [0].SendText(S);
  end;
end;
procedure TForm1.DownClick(Sender:
TObject);
var S:String;
  code,check: Integer;
begin
 code := SioReset(Port,2048,512);
 if code < 0 then
 begin
  showmessage('PORT COM Error');
  timer2.Enabled := false;
  timer3.Enabled := false;
  timer4.Enabled := false;
  timer5.Enabled := false;
  exit;
 end;
```

```
  sioBaud(Port,Baud);
  sioparms(Port,parity,stopbits,databits);
  siodtr(Port,'S');
  siorts(Port,'S');
  if count_UD > -45 then
  begin
   SioPutc(Port,chr(56));
   count_UD:= count_UD-7.5;
  end;
  edit3.text:=floattostr(count_UD);
  if count_UD = -45 then ao:=2;
  check := 0;
  repeat
  code := SioGetc(Port);
  check := check+1;
  if check > 10000 then  break;
  until code >= 0 ;
  if count_UD <= -45 then
  begin
   if SpeedButton2.Down = false then
down.Enabled:=false;
   edit4.text:='over down';
  end
  else if count_UD <= 45 then
  begin
   if SpeedButton2.Down = false then
up.Enabled:=true;
   edit4.text:='';
  end ;
  SioDone(Port);
  if (speedbutton3.Down = true) and
(Statusbar1.Panels[0].Text='Connection
accepted') then
  begin
   S:='V'+Edit3.Text;
   ServerSocket.Socket.Connections
[0].SendText(S);
   S:='E'+Edit4.Text;
   ServerSocket.Socket.Connections
[0].SendText(S);
  end;
end;
procedure TForm1.ServerSocketClientRead
(Sender: TObject;
  Socket: TCustomWinSocket);
var A:String;
begin
  A:=Socket.ReceiveText;
  if (A='3')  then  SpeedButton1.Click
  else if (A='5')then  SpeedButton2.Click
  else if (A='7')then Stop.Click
  else if (A='2')then Up.Click
  else if (A='4')then Down.Click
  else if (A='6')then Left.Click
  else if (A='8')then Right.Click;
end;
procedure TForm1.SpeedButton1Click
(Sender: TObject);
begin
  SpeedButton1.Down := true;
  left.Enabled:=true;
  right.Enabled:=true;
  up.Enabled:=true;
  down.Enabled:=true;
  timer2.Enabled := false;
  timer3.Enabled := false;
  timer4.Enabled := false;
  timer5.Enabled := false;
end;
procedure TForm1.SpeedButton2Click
(Sender: TObject);
```

```
begin
  SpeedButton2.Down := true;
  left.Enabled:=false;
  right.Enabled:=false;
  up.Enabled:=false;
  down.Enabled:=false;
  timer2.Enabled:=true;
  {timer3.Enabled:=false;
  timer4.Enabled:=true;
  timer5.Enabled:=false;}
end;
{procedure TForm1.Timer5Timer(Sender:
TObject);
begin
  if count_UD > -45 then down.click;
  if count_UD = -45 then
  begin
    timer5.Enabled := false;
    timer4.Enabled := true;
    //if count_LR = 90 then timer3.Enabled :=
true;
    //if count_LR = -90 then timer2.Enabled
:= true;
  end;
end; }
procedure TForm1.FormCreate(Sender:
TObject);
begin
  Port := COM2;
  Form1.Caption := 'Camera : COM' + Chr
($31+Port);
  Baud := Baud9600;
  Parity := NoParity;
  DataBits := Wordlength8;
  StopBits := OneStopBit;
  count_LR:=0;
  count_UD:=0;
  SpeedButton1.Down := true;
  scan:=0;
  gade:=2;
  ao:=2;
end;
procedure TForm1.Timer1Timer(Sender:
TObject);
begin
  Panel1.Caption:='Date : '+Datetostr
(Now)+'  Time : '+timetostr(Now);
end;
procedure TForm1.Timer2Timer(Sender:
TObject);
begin
  // if count_LR = 90 then gade:=1;
  // if count_UR = -90 then gade:=2;
    if (count_LR < 90) and (gade=2) then
begin left.Click;
    { if count_LR = 90 then gade:=1;} end
    else  if (count_LR > -90) and (gade=1)
then begin right.click;
    { if count_LR = -90 then gade:=2;} end;
    if (count_UD < 45) and (ao=2) then begin
up.click;
    {if count_UD=45 then ao:=1; }end
    else if (count_UD > -45) and (ao=1) then
begin down.click;
    {if count_UD = -45 then ao:=2} end;
    //Edit6.text:=inttostr(ao);
  {if count_LR = 90 then
  begin
    timer2.Enabled := false;
    timer3.Enabled := true;
  end;}
end;
```

```
{procedure TForm1.Timer3Timer(Sender:
TObject);
begin
 if count_LR > -90 then right.click;
 if count_LR = -90 then
 begin
  timer3.Enabled := false;
  timer2.Enabled := true;
 end;
end;
procedure TForm1.Timer4Timer(Sender:
TObject);
begin
 if count_UD < 45 then up.click;
 if count_UD = 45 then
 begin
  timer4.Enabled := false;
  timer5.Enabled := true;
 end;
end;
procedure TForm1.Timer5Timer(Sender:
TObject);
begin
 if count_UD > -45 then down.click;
 if count_UD = -45 then
 begin
  timer5.Enabled := false;
  timer4.Enabled := true;
  //if count_LR = 90 then timer3.Enabled :=
true;
  //if count_LR = -90 then timer2.Enabled
:= true;
 end;
end;}
procedure TForm1.SpeedButton3Click
(Sender: TObject);
var PortNo:integer;
  PortNo1:integer;
begin
 PortNo:=StrToInt(Edit1.Text);
 ServerSocket.Port:=PortNo;
 ServerSocket.Open;
 PortNo1:=StrToInt(Edit5.Text);
 ServerSocket1.Port:=PortNo1;
 ServerSocket1.Open;
end;
procedure TForm1.SpeedButton4Click
(Sender: TObject);
begin
ImageEnVideoView1.ShowVideo:=SpeedB
utton4.Down;
 DisplayVideoSize;
 if (scan=2) and (Statusbar1.Panels
[0].Text='Connection accepted')
 then button1.Click;
 // edit6.Text:=inttostr(scan);
end;
procedure TForm1.Format1Click(Sender:
TObject);
begin
 if not
ImageEnVideoView1.DoConfigureFormat
then
 MessageDlg('Configure Format dialog not
available',mtInformation,[mbOK],0)
 else
  DisplayVideoSize;
end;
procedure TForm1.Source1Click(Sender:
TObject);
begin
```

```
 if not
ImageEnVideoView1.DoConfigureSource
then
 MessageDlg('Configure Source dialog not
available',mtInformation,[mbOK],0)
 else
  DisplayVideoSize;
end;
procedure
TForm1.ServerSocket1ClientRead(Sender:
TObject;
 Socket: TCustomWinSocket);
var Z:string;
begin
 Z:=socket.ReceiveText;
 if (Z='Comp') then  Button1.click;
end;
procedure TForm1.Exit1Click(Sender:
TObject);
begin
 ServerSocket.Close;
 Close;
end;
procedure TForm1.ImageEnProc(Sender:
TObject; Bitmap: TBitmap);
type frame1 = array [1..160,1..120] of
integer;
var
  pixframe1 : frame1;
  RED,GREEN,BLUE : char;
  x,y,n,a :integer;
  r,g,b : byte;
  L : Longint;
begin
 ImageEnView1.assign(Bitmap);
 ImageEnView1.Fit;
 if scan=1 then
 begin
  n:=1;
  for y:=1 to 119 do
  begin
   for x:=1 to 159 do
   begin
   a :=
ImageEnView1.Bitmap.Canvas.Pixels[x,y];
    pixframe1[x,y] := a ;
    L:=ColorToRGB(a);
    r:=GetRValue(L);
    g:=GetGValue(L);
    b:=GetBValue(L);
    RED:=chr(r);
    GREEN:=chr(g);
    BLUE:=chr(b);
    Array1[n]:=RED;
    Array1[n+1]:=GREEN;
    Array1[n+2]:=BLUE;
    n:=n+3;
   end;
  end;
  scan:=2;
 if  Statusbar1.Panels[0].Text='Connection
accepted' then
ServerSocket1.Socket.Connections
[0].Sendtext(Array1);
 end;
end;
procedure TForm1.DisplayVideoSize;
var
        r:TRect;
begin
 r:=ImageEnVideoView1.GetVideoSize;
```

```
Label6.caption:=inttostr(r.right+1)+'x'+intto
str(r.bottom+1);
end;
procedure TForm1.About1Click(Sender:
TObject);
begin
    Form2.show;
end;
procedure TForm1.Button1Click(Sender:
TObject);
begin
 if SpeedButton4.Down then
 begin
  scan:=1;
ImageEnVideoView1.OnVideoFrame:=Ima
geEnProc
 end
 else
 ImageEnVideoView1.OnVideoFrame:=nil;
end;
```

```
procedure TForm1.Timer6Timer(Sender:
TObject);
begin  timer6.Enabled:=false;
    if
(Statusbar1.Panels[0].Text='Connection
accepted')
    and (Speedbutton4.Down = true) and
(scan=0)
    then button1.click
    else timer6.Enabled:=true;
end;
procedure
TForm1.ServerSocketClientDisconnect
(Sender: TObject;
 Socket: TCustomWinSocket);
begin
 Statusbar1.Panels[0].Text:='Now Client
Disconnect';
end;
end.
```

โปรแกรมส่วน Server

```
unit server3_2;
interface
uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls, jpeg, ExtCtrls;
type
  TForm2 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Panel2: TPanel;
    Image1: TImage;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Button1: TButton;
    Label11: TLabel;
    procedure Button1Click(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
{$R *.DFM}
procedure TForm2.Button1Click(Sender:
TObject);
begin
      form2.Visible:=false;
end;
procedure TForm2.FormCreate(Sender:
TObject);
begin
      form2.Caption := 'About Project' ;
end;
end.
```

โปรแกรมส่วน About project ฝั่ง Server