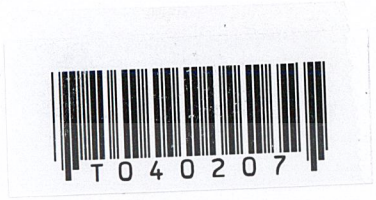
การควบคุมหุ่นยนต์ผ่านเครือข่ายอินเตอร์เน็ต



นายเบญจพล ตันฮู้
นายสันติชัย พวงแก้ว

โครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาฟิสิกส์ประยุกต์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

Internet Robot Control

Mr.Benchapol Tunhoo

Mr.Santichai Poungkaew

A Special Project Submitted on Partial Fulfillment of the

Requirement for the Degree of Bachelor of Science

Department of Applied Physics

Faculty of Science

King Mongkut 's Institute of Technology Ladkrabang

2000

หัวข้อโครงงานพิเศษ	การควบคุมหุ่นยนต์ผ่านเครือข่ายอินเตอร์เน็ต
โดย	นายเบญจพล ตันฮู้
	นายสันติชัย พวงแก้ว
ภาควิชา	ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา	ผศ.ดร.จิติ หนูแก้ว
	ผศ.ดร.เสน่ห์ เอกะวิภาต
	อ.อนุชิต จารุวนาวัฒน์

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
อนุมัติให้นับโครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิทยยาศาสตรบัณฑิต

.....หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ผศ. วิชาญ เตชะธีระ)

คณะกรรมการโครงงานพิเศษ

.....ประธานกรรมการ

(รศ.สุวรรณ คูสำราญ)

.....กรรมการ

(ผศ.ดร.จิติ หนูแก้ว)

.....กรรมการ

(ผศ.ดร.เสน่ห์ เอกะวิภาต)

.....กรรมการ

(ผศ.อนุพงศ์ สรงประภา)

.....กรรมการ

(อ.ภูมินทร์ จินดาจิธาวัฒน์)

ลิขสิทธิ์ของภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อโครงงานพิเศษ	การควบคุมหุ่นยนต์ผ่านเครือข่ายอินเตอร์เน็ต
นักศึกษา	นายเบญจพล ตันฮู้
	นายสันติชัย พวงแก้ว
อาจารย์ที่ปรึกษา	ผศ.ดร.จิติ หนูแก้ว
	ผศ.ดร.เสน่ห์ เอกะวิภาต
	อ.อนุชิต จารุวนาวัฒน์
ภาควิชา	ฟิสิกส์ประยุกต์
ปีการศึกษา	2543

บทคัดย่อ

โครงงานพิเศษนี้เป็นการสร้างหุ่นยนต์ที่ควบคุมผ่านเครือข่ายอินเตอร์เน็ต จากการนำเครื่องคอมพิวเตอร์แพลตฟอร์มไอบีเอ็มมาเป็นหน่วยประมวลผลหลัก โดยมีระบบปฏิบัติการลีนุกซ์ติดตั้งอยู่ การควบคุมนั้นทำได้โดยการใช้งานจากเครื่องคอมพิวเตอร์อื่นที่อยู่ในเครือข่ายอินเตอร์เน็ต มาควบคุมหุ่นยนต์ให้เคลื่อนที่ และรับภาพจากกล้องดิจิตอลที่ติดอยู่ด้านหน้าหุ่นยนต์ การประยุกต์ใช้งานหุ่นยนต์ที่สร้างขึ้นนี้คือเครื่องตรวจจับที่ควบคุมจากระยะไกล

Special Project Title	Internet Robot Control	
Name	Mr. Benchapol	Tunhoo
	Mr. Santichai	Poungkaew
Special Project Advisor	Asst.Prof.Dr.Jiti	Nukeaw
	Asst.Prof.Dr.Sanay	Akavipat
	Mr.Anuchit	Jaruvanawat
Department	Applied Physics	
Academic Year	2000	

Abstract

The objective of the special project is to construct the robot which is controlled by the internet network. The IBM computer platform with Linux operating system is used for main processing. The CCD camera is set up on the robot for the image sensing. The robot and CCD camera movement can be controlled by terminal in this internet network. For more application, the robot can use as the remote control sensing.

กิติกรรมประกาศ

โครงงานพิเศษนี้สำเร็จลุล่วงไปด้วยดีเนื่องด้วยความอนุเคราะห์จากบุคคลหลายๆ ฝ่ายดังนี้

บิดามารดา และผู้ปกครอง	ผู้ให้กำลังใจและคอยห่วงใยเสมอมา
ผศ.ดร.จิติ หนูแก้ว	ผู้ให้การสั่งสอน,ถ่ายถอดความรู้ ให้คำแนะนำและให้ โอกาสในการทำงานเสมอมา
ผศ.อนุพงศ์ สรงประภา	ที่เป็นห่วงใยเวลาทำงานดึก ๆ เสมอมา
คณาจารย์ทุกท่าน	ผู้ถ่ายถอดความวิชารู้โดยตลอดมา
เพื่อน ๆ ทุกคน	ที่คอยให้ความช่วยเหลือด้วยกันตลอดมา
ภาควิชาฟิสิกส์ประยุกต์ และคณะวิทยาศาสตร์ สจล.	ที่มอบสิ่งดี ๆ ตลอด 4 ปี

ขอขอบพระคุณทุกท่านเป็นอย่างสูง

เบญจพล ตันสู้

สันติชัย พวงแก้ว

สารบัญเรื่อง

หน้า

สารบัญตาราง

สารบัญรูป

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและที่มาของปัญหา

งานวิจัยทางด้านฟิสิกส์บางงานเช่น การวัดปริมาณสารพิษ การสำรวจ ฯลฯ มนุษย์ไม่สามารถเข้าไปดำเนินการ ประกอบ ติดตั้ง ตรวจวัด และทดสอบ ได้ด้วยตนเอง ประกอบกับการทำงานที่ต้องอาศัยความเที่ยงตรง แม่นยำ และ รวดเร็ว จึงเป็นอุปสรรค ต่อการดำเนินงานวิจัย ซึ่งในโครงงานฉบับนี้จะเป็นการประยุกต์ใช้โดยการนำสิ่งประดิษฐ์เข้าไปทำงานแทนโดยมีมนุษย์เป็นผู้ควบคุมอยู่ในอีกที่หนึ่ง โดยที่ระบบที่ออกแบบนั้นต้องมีการแลกเปลี่ยนข้อมูล และดำเนินงานตามคำสั่งของผู้ควบคุมได้เป็นอย่างดี

ในโครงงานพิเศษนี้จะเป็นการประยุกต์ การควบคุมระยะไกล (remote control) ผ่านทางระบบเครือข่ายอินเตอร์เน็ต โดยอาศัยระบบปฏิบัติการไลนักซ์ (Linux Operating Systems) ซึ่งเป็นระบบปฏิบัติการที่เสถียรภาพสูง และมีระบบรักษาความปลอดภัยที่ดี เหมาะในการนำมาใช้ในการควบคุมระยะไกล

1.2 วัตถุประสงค์ในการทำโครงงานพิเศษ

1. เพื่อเป็นการเรียนรู้การทำงานของระบบเครือข่ายอินเตอร์เน็ต
2. เพื่อเป็นการเรียนรู้กลไกการทำงานของ Microcomputer และ Microcontroller ในระบบงานควบคุม
3. สามารถควบคุมหุ่นยนต์จากระยะไกลได้เพื่อสามารถนำไปประยุกต์ใช้ในงานต่างๆได้
4. เพื่อให้เรียนรู้การทำงานที่เป็นระบบ รู้จักวิเคราะห์และแก้ปัญหาเฉพาะหน้าได้

1.3 ขอบเขตของงานที่ทำ

ในโครงงานพิเศษนี้จะเป็นจะทำการสร้างตัวหุ่นยนต์โดยมีระบบปฏิบัติการไลนักซ์ติดตั้งอยู่โดยจะทำหน้าที่เป็นเครื่องแม่ (Server)โดยการสั่งงานจะสามารถสั่งงานผ่านเครื่องลูก (Client) ทางระบบเครือข่ายอินเตอร์เน็ต โดยที่เครื่องแม่ (Server) จะไปสั่งงานวงจรไมโครคอนโทลเลอร์ AT89C52 ที่ทำการควบคุมการเคลื่อนที่ของตัวหุ่นยนต์ ผ่านทางพอร์ตอนุกรมอีกทีหนึ่ง

1.4 ประโยชน์ที่ได้รับ

1. มีความรู้ความสามารถในด้านการควบคุมด้วย Microcontroller
2. มีความรู้ทางด้านระบบเครือข่ายอินเตอร์เน็ต
3. นำหุ่นยนต์ไปประยุกต์ใช้ในงานสำรวจด้านต่างๆ ได้ และงานทางด้านตรวจจับระยะไกล
4. รู้จักวิธีการทำงานอย่างเป็นระบบ ทำงานร่วมกับผู้อื่น วิเคราะห์และแก้ไขปัญหาอย่างมีหลักการและเหตุผล

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 พื้นฐานระบบเน็ตเวิร์ก

2.1.1 ระบบเครือข่ายอินเตอร์เน็ต

มนุษย์เริ่มรู้จักการสื่อสารข้อมูลมานานแล้ว ตั้งแต่สมัยโบราณเรารู้จักการใช้จดหมายผูกไปกับนกพิราบสื่อสาร พัฒนาเป็นการส่งข้อมูลผ่านโทรเลข การพูดคุยผ่านโทรศัพท์และในที่สุดก็พัฒนาเป็นการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ การพัฒนาการสื่อสารข้อมูลระหว่างคอมพิวเตอร์เริ่มมีแนวคิดมานานแล้ว โครงการที่สำคัญของสหรัฐอเมริกา ซึ่งเป็นโครงการเกี่ยวกับทางทหาร ต้องการที่จะเชื่อมโยงระบบคอมพิวเตอร์ในประเทศเข้าด้วยกัน และคิดค้นหาวิธีที่จะป้องกันไม่ให้การสื่อสารของทั้งระบบต้องหยุดชะงัก หากเส้นทางการสื่อสารบางเส้นทางถูกทำลายไประบบจะต้องมีความสามารถในการเลือกหาเส้นทางการสื่อสารเส้นทางอื่นที่ใช้งานเป็นปกติได้ ในที่สุดระบบดังกล่าวนี้ก็ได้ถูกพัฒนาใช้ขึ้นมาเพื่อการศึกษาวิจัย และใช้ในเชิงพาณิชย์ในที่สุด ซึ่งก็คือระบบเครือข่ายอินเทอร์เน็ตในปัจจุบันนั่นเอง ปัจจุบันระบบเครือข่ายอินเตอร์เน็ตนับว่ามีบทบาทมากมายมหาศาล และนับวันจะยิ่งเปลี่ยนแปลงวิถีชีวิตของมนุษย์เข้าไปทุกที

การใช้งานเน็ตเวิร์ก การเชื่อมต่อคอมพิวเตอร์เข้าด้วยกัน จะมีจุดประสงค์เพื่อสามารถให้ใช้ทรัพยากรร่วมกัน ผู้ที่ใช้งานอยู่บนคอมพิวเตอร์เครื่องหนึ่ง จะสามารถใช้บริการทรัพยากรของเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งซึ่งอยู่ห่างไกลออกไปได้ สำหรับจุดมุ่งหมายอื่น เช่นเพื่อเน้นความน่าเชื่อถือของระบบ และการลดค่าใช้จ่ายลง

2.1.2 โครงสร้างของเครือข่าย

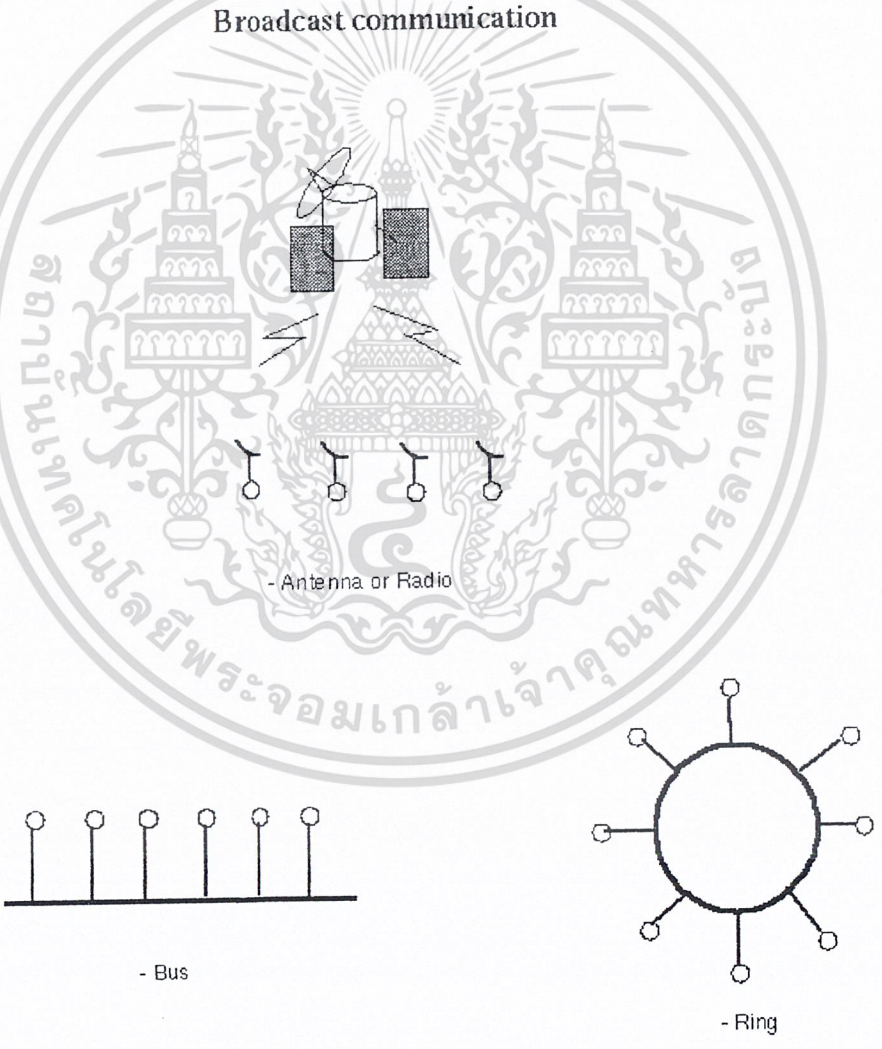
ภายในเน็ตเวิร์กจะมี เครื่องคอมพิวเตอร์ประเภทต่างๆ ที่ทำหน้าที่ตามคำสั่งของผู้ใช้ จะเรียกว่า Host (end system) ซึ่งระบบแรกๆจะถูกเชื่อมผ่าน subnet แต่ในระยะหลังเนื่องจากมีขนาดเครือข่ายกว้างขวางขึ้น รวมทั้งมีความซับซ้อนของการสื่อสารข้อมูลมากขึ้น จะมีระบบสลับข้อมูลเข้ามาช่วยในการสื่อสารข้อมูลระบบสลับข้อมูลนี้จะเรียกว่า IMP (Interface Message processor) บางทีอาจจะเรียกว่า package switching node หรือ intermediate system ก็มี

ระยะทางระหว่างโปรเซสเซอร์	โปรเซสเซอร์อยู่ที่	ตัวอย่างเช่น
0.1 เมตร	วงจรภายในบอร์ด	Data flow machine
1 เมตร	ภายในระบบ	Multiprocessor
10 เมตร	ภายในห้อง	
100 เมตร	ในอาคาร	Local area network
1 กิโลเมตร	ในสถาบันการศึกษา	
10 กิโลเมตร	ในเมือง	Wide area network
100 กิโลเมตร	ในประเทศ	Long Haul Network
1000 กิโลเมตร	ในทวีป	การต่อระหว่าง long haul
10,000 กิโลเมตร	ในดาวเคราะห์	Internet

ตารางที่ 2.1 แสดงเน็ตเวิร์กกับระยะห่างต่างๆ

รูปที่ 2.1 แผนภาพแสดงการเชื่อมต่อของเน็ตเวิร์กแบบจุดถึงจุด

ปกติรูปแบบการเชื่อมต่อจะเป็นแบบ จุดถึงจุด (point to point) และแบบกระจาย (broadcast channel)การเชื่อมต่อแบบจุดถึงจุด จะประกอบด้วยสายเคเบิล หรือสายสื่อสารเชื่อมต่อระหว่าง IMP ของแต่ละจุดเข้าด้วยกัน ในกรณีต้องการจะทำการสื่อสารระหว่าง IMP ซึ่งไม่มีสายสื่อสารเชื่อมต่อเข้าด้วยกัน จะต้องทำการส่งผ่าน IMP ตัวอื่นๆที่อยู่ใกล้เคียงแทน เมื่อกลุ่มของข้อมูล (message หรือ packet) ถูกส่งจาก IMP ออกมาแล้ว IMP ตัวกลางจะทำการเก็บแพคเก็ตเอาไว้ รอจนกระทั่งสายที่ใช้ส่งว่างแล้วจึงส่งแพคเก็ตออกไป วิธีแบบนี้จะถูกเรียกว่า store and forward หรือ packet switched



รูปที่ 2.2 แผนภาพแสดงการเชื่อมต่อของเน็ตเวิร์กแบบกระจาย

การเชื่อมต่อแบบกระจาย จะมีช่องทางเชื่อมสัญญาณเพียงช่องทางเดียวและต้องใช้ร่วมกับเน็ตเวิร์กอื่นๆ เมื่อแพคเก็ตถูกส่งออกไปแล้ว IMP อื่นๆจะได้รับสัญญาณในเวลาที่ใกล้เคียงกัน เมื่อ IMP ได้รับสัญญาณแล้วจะตรวจดูว่าเป็นแพคเก็ตของตนหรือไม่ ได้จาก address ภายในแพคเก็ตนั้น หากไม่ตรงกับ address ของตนเอง แพคเก็ตนั้นก็จะถูกทิ้งไป

ในการออกแบบเน็ตเวิร์กนั้น จะทำการแบ่งหน้าที่ของการทำงานต่างๆออกเป็นลำดับชั้น (layer) โดยที่แต่ละเลเยอร์จะทำการรับข้อมูลจาก เลเยอร์ที่สูงกว่าลงมา และส่งผ่านไปให้เลเยอร์ที่ต่ำกว่าลงไปเรื่อยๆ จนถึงเลเยอร์ลำดับล่างสุด ซึ่งจะเป็นตัวกลางทางกายภาพ (physical medium) หรือสายสัญญาณที่ใช้ทำการส่งข้อมูลกันจริงๆ เรามักจะมองว่า เกิดการสื่อสารกันระหว่างเลเยอร์ที่มีลำดับตรงกัน กับคอมพิวเตอร์ทั้งสองฝ่ายขึ้น ซึ่งจะเรียกว่าการสื่อสารเสมือน (virtual communication) แต่ในความเป็นจริงแล้ว การสื่อสารที่เกิดขึ้นจริงจะถูกส่งผ่านตามเลเยอร์ลงมาจนกระทั่งถึงเลเยอร์ลำดับล่างสุด ซึ่งเป็นตัวกลางการสื่อสาร และส่งผ่านไปให้คอมพิวเตอร์อีกด้านหนึ่งและผ่านเลเยอร์ จากชั้นล่างสุดไปจนถึงเลเยอร์ชั้นบนสุด วิธีการที่เลเยอร์ในลำดับที่ตรงกันของเครื่องที่ทำการสื่อสารกัน จะถูกเรียกว่าโปรโตคอล (protocol)

ตัวอย่างเช่นการสื่อสารแบบ IRC ผ่านอินเทอร์เน็ต เรามองดูเหมือนว่า เกิดการสื่อสารขึ้นจริงๆระหว่าง เรากับเพื่อนของเราที่อยู่ห่างไกลออกไป แต่ในความเป็นจริงเมื่อเราทำการพิมพ์ข้อความ ข้อความนั้นจะถูกดัดแปลงตกแต่งและถูกส่งลงไปในเลเยอร์การสื่อสารลำดับต่ำกว่า ในกรณีนี้โปรแกรม IRC ที่เราและเพื่อนเราใช้จะต้องมีวิธีการสื่อสารหรือโปรโตคอลแบบเดียวกัน (นี่เป็นเพียงการเปรียบเทียบ เนื่องจากการสื่อสารที่เกิดขึ้นจริงจะต้องผ่าน Chat server ก่อนที่จะมีการส่งข้อมูลของแต่ละฝ่ายไปให้กัน)

2.1.3 แบบจำลองของ OSI (The OSI Reference model)

เพื่อจะสามารถทำการอธิบาย สถาปัตยกรรมเครือข่ายคอมพิวเตอร์ได้เข้าใจอย่างละเอียด รวมทั้งเพื่อแสดงถึงตัวอย่างการออกแบบเลเยอร์ และรูปแบบของโปรโตคอลจึงได้มีการพัฒนาแบบจำลอง (model) ของเน็ตเวิร์กขึ้นมา แบบจำลองนี้ได้ถูกพัฒนาโดย International Standard Organization (ISO) และเรียกชื่อแบบจำลองนี้ว่า OSI (Open System Interconnection) ซึ่งจะมีทั้งหมด 7 เลเยอร์ โดยหลักการออกแบบเลเยอร์คือ

- เลเยอร์จะถูกกำหนดขึ้นมาเมื่อมีข้อแตกต่างด้านแนวคิด (abstraction)
- แต่ละเลเยอร์จะมีการกำหนดการทำงานอย่างละเอียด
- ฟังก์ชันภายในเลเยอร์จะพยายามมุ่งไปสู่ระดับมาตรฐานของโปรโตคอล

- ขอบเขตของเลเยอร์จะถูกเลือกและจำกัดให้มีปริมาณการเชื่อมต่อ ระหว่างเลเยอร์ให้น้อยที่สุด

- จำนวนของเลเยอร์จะต้องมากพอที่จะทำให้ฟังก์ชั่นที่จำเป็นและแตกต่างกันไม่อยู่ในเลเยอร์ เดียวกัน และจำนวนเลเยอร์จะต้องไม่มากจนเกินไป

เลเยอร์ 1 : Physical layer

เลเยอร์นี้จะเกี่ยวข้องกับการส่งข้อมูลระดับบิต ผ่านช่องสื่อสารข้อมูล โดยการออกแบบจะต้องแน่ใจว่าจะสามารถส่งข้อมูลออกไป และปลายทางจะต้องรับข้อมูลนั้นได้อย่างถูกต้อง โดยส่วนใหญ่จะเป็นข้อกำหนดเกี่ยวกับเรื่องของแรงดันไฟฟ้า ว่าจะต้องใช้แรงดันเท่าไรสำหรับแทนเลข "1" และเท่าใดสำหรับแทนเลข "0" ระยะเวลาในการส่งแต่ละบิตจะต้องห่างกันเท่าใด รวมถึงรูปแบบของคอนเนคเตอร์ วิศวกรไฟฟ้าจะเกี่ยวข้องโดยตรงกับเลเยอร์ในลำดับชั้นนี้

เลเยอร์ 2 : Data link layer

จุดประสงค์หลักของเลเยอร์นี้คือ จะทำการควบคุมการส่งข้อมูลดิบให้เหมือนกับว่าไม่มี ข้อผิดพลาดเกิดขึ้น ทำให้เลเยอร์ในลำดับถัดไปไม่ต้องสนใจในเรื่องนี้ วิธีการคือจะต้อง ทำการแตกข้อมูลออกเป็นก้อนๆเรียกว่า เฟรมข้อมูล (data-frame) แล้วทำการส่งออกไปทีละชุด และรอการตอบรับ (acknowledge frame) กลับมาในกรณีที่มีสัญญาณรบกวน ทำให้สัญญาณขาดหายไป จะต้องมีการบอกให้เครื่องต้นทางทำการส่งข้อมูลที่หายไปนั้นกลับมาให้ใหม่ นอกจากนี้จะต้องมีการพักข้อมูลไว้ในบัฟเฟอร์ หากความเร็วในการส่งข้อมูลของทั้งสองฝ่ายไม่เท่ากัน

เลเยอร์ 3 : Network layer

หน้าที่หลักของเลเยอร์นี้ จะเกี่ยวข้องกับการหาเส้นทาง (route) เพื่อพิจารณาว่าแพคเก็ตจะถูกส่งจากต้นทางไปยังปลายทางได้อย่างไร การกำหนดเส้นทางอาจจะกำหนดตั้งแต่เริ่มต้นติดต่อเลย หรืออาจจะสามารถเปลี่ยนแปลงตลอดเวลา (dynamic) ก็ได้ นอกจากนี้ก็มีเรื่องที่จะต้องพิจารณาหากมีการส่งข้อมูลข้ามเน็ตเวิร์ก แล้วมีความแตกต่างระหว่างเน็ตเวิร์ก หรือใช้โปรโตคอลแตกต่างกัน เน็ตเวิร์กเลเยอร์จะต้องทำการจัดการกับปัญหาเหล่านี้เพื่อให้แต่ละเน็ตเวิร์กสามารถเชื่อมต่อกันได้ เสมือนเป็นเน็ตเวิร์กเดียวกัน

เลเยอร์ 4 : Transport layer

เลเยอร์นี้จะคอยทำการติดต่อกับเลเยอร์ถัดไป (session layer) เพื่อคอยแยกข้อมูล ให้มีขนาดพอเหมาะและส่งต่อให้กับ network layer พร้อมทั้งตรวจสอบว่าข้อมูลได้ถูกส่งไปถึงยังปลายทางได้อย่างเรียบร้อยหรือไม่ โดยเลเยอร์นี้จะต้องจัดการได้อย่างมีประสิทธิภาพเพื่อเป็นการแยกให้ session layer เป็นอิสระจากการเปลี่ยนแปลงทางด้านฮาร์ดแวร์

เลเยอร์ 5 : Session layer

นอกจากจะทำการส่งข้อมูลแบบเดียวกับ transport layer แล้ว ยังมีการให้บริการอื่นๆ เช่น การยอมให้ผู้ใช้งานเข้าไปใช้งานยังเครื่องที่อยู่ห่างไกลออกไป (remote login) หรือทำการถ่ายโอนไฟล์ระหว่างเครื่อง นอกจากนี้ยังทำหน้าที่เกี่ยวกับการซิงโครไนซ์เซชั่น (synchronization) หรือทำให้สองระบบทำงานสัมพันธ์กัน

เลเยอร์ 6 : Presentation layer

เลเยอร์นี้จะสนใจในเรื่องของรูปแบบของข้อมูล เช่นการเปลี่ยนรหัสข้ามจาก ASCII เป็นรหัส EBCDIC เพื่อทำให้คอมพิวเตอร์ที่มีการแทนรหัสต่างกันสามารถสื่อสารกันได้ นอกจากนี้ก็อาจทำการลดขนาดของข้อมูล (data compression) หรือทำการเข้ารหัสของข้อมูล (data encryption) เพื่อป้องกันการโจรกรรมข้อมูลได้ด้วย

เลเยอร์ 7 : Application layer

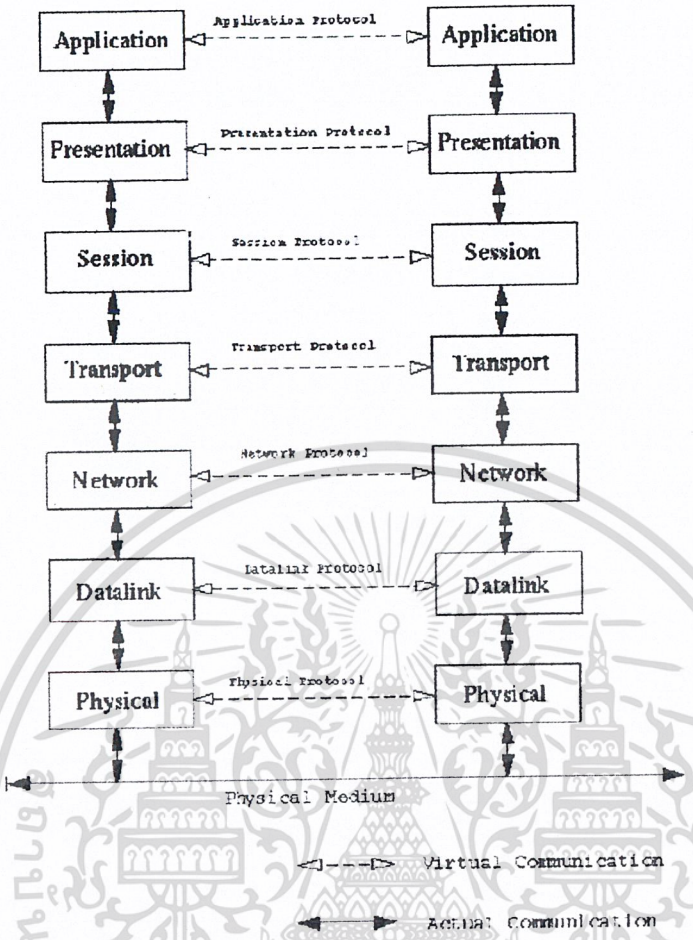
เลเยอร์บนสุดจะเกี่ยวข้องกับ โปรโตคอลมากมาย ซึ่งจะมีการใช้งานที่แตกต่างกันโดยเฉพาะ ตัวอย่างเช่นการควบคุมเทอร์มินอลชนิดต่างๆ รูปแบบการแสดงผลทางจอภาพอาจมีความแตกต่างกัน ก็อาจมีการกำหนดเทอร์มินอลเสมือน เพื่อเป็นตัวกลางในการควบคุมการทำงานของเทอร์มินอลจริงๆ (คล้ายกับ java bytecode ที่เป็นรหัสกลางสำหรับ java compiler จะทำการตีความให้สามารถทำงานที่เครื่องต่างชนิดกันได้)

ตัวอย่างของเน็ตเวิร์ก เปรียบเทียบกับโมเดลของ OSI

ARPANET : Advanced Research Project Agency Network เกิดจากโครงการวิจัยทางการทหารของ กระทรวงกลาโหมของสหรัฐอเมริกาโดยเริ่มต้นจากมหาวิทยาลัยบางแห่ง และเพิ่มจำนวนขึ้นมากเรื่อยๆ (ปัจจุบันกลายเป็นระบบอินเทอร์เน็ต)

ARPANET ไม่ได้มีการแบ่งรูปแบบตามแบบของ OSI เนื่องจากมีการใช้งานก่อนกำหนดมาตรฐาน OSI เกือบ 10 ปี โปรโตคอลการสื่อสารระหว่าง IMP จะเป็นการผสมผสานระหว่าง layer 2 และ layer 3 นอกจากนี้ยังมีกระบวนการตรวจสอบความถูกต้องของข้อมูลที่ฝั่งรับของ IMP ด้วย

ARPANET มีโปรโตคอลที่ทำงานคล้ายในแบบจำลองของ OSI ที่ network layer และ transport layer เรียกว่า IP (Internet Protocol) มีลักษณะแบบ connectionless และถูกออกแบบมาให้ต่อกับ LAN และ WAN ที่ต่อกับ ARPA internet โปรโตคอล ของ ARPANET ที่ transport layer ทำงานแบบ connection-oriented เรียกว่า TCP (Transmission Control Protocol) ซึ่งจะคล้ายๆกับโปรโตคอลของ OSI ที่ transport layer แต่มีความแตกต่างกันในด้านรายละเอียด และ TCP ถูกใช้งานอย่างแพร่หลายมากในระบบปฏิบัติการตระกูล UNIX



รูปที่ 2.3 แผนภาพแสดงการติดต่อของชั้นต่างๆ ของเน็ตเวิร์ก

ARPANET ในระยะแรกไม่มี session และ application protocol แต่ภายหลังก็มี application protocol ต่างๆ ซึ่งไม่ได้มีโครงสร้างแบบเดียวกับ OSI บริการที่มีเช่น FTP (File Transfer Protocol) หรือการถ่ายโอนไฟล์, SMTP (Simple Mail Transfer Protocol) สำหรับใช้งานจดหมายอิเลคโทรนิคส์ และ TELNET สำหรับการขอใช้งานจากระยะไกล ในระยะหลังๆเมื่อ ARPANET ได้กลายเป็นอินเทอร์เน็ต แล้ว ได้มีการพัฒนา application protocol ขึ้นมามากมายภายหลัง เช่น HTTP, NNTP เป็นต้น

2.1.4 TCP/IP

การจะใช้งาน TCP/IP ได้นั้นจำเป็นจะต้องมีการกำหนดค่าของ IP address หรือหมายเลขประจำตัวให้กับเครื่องคอมพิวเตอร์แต่ละเครื่องในเครือข่าย ซึ่งจะคล้ายกับเลขที่บ้านของคุณ ที่จะทำหน้าที่บอกตำแหน่งที่อยู่ของคุณให้คนอื่นๆได้รับทราบ

นอกจากนี้ก็ยังจะมีค่าอื่นๆอีก แต่ที่จำเป็นอีกอย่างก็คือค่าของ MTU หรือ Maximum Transfer Unit ซึ่งค่านี้ก็จะบอกถึงจำนวนสูงสุดของข้อมูล (datagram) ที่จะใช้ทำการ process

IP Address

IP address เป็นตัวเลขความยาว 32 บิท และจะต้องกำหนดให้แต่ละเครื่องมีความแตกต่างกันงานเน็ตเวิร์กของคุณเพียงแค่ในหน่วยงานของคุณเอง ในกรณีนี้คุณสามารถกำหนดหมายเลขของ IP ให้กับเครื่องต่างๆของคุณเองได้ แต่ถ้าหากเป็นเน็ตเวิร์กที่ติดต่อกับสาธารณะเมื่อใด จะต้องมีการกำหนดมาจากหน่วยงานที่ทำการควบคุมหมายเลข IP โดยเฉพาะซึ่งจะเรียกว่า the Network Information Centor (NIC) หมายเลข IP จะถูกแบ่งออกเป็นตัวเลข 8 บิทจำนวน 4 ส่วน (octets) ตัวอย่างเช่น IP address 0x954C0C04 จะถูกเขียนอย่างง่ายได้เป็น 149.76.12.4 การเขียนแบบนี้จะเรียกว่า dotted quad notation ประโยชน์อีกอย่างของการเขียนแบบแบ่งเป็นตัวเลข 4 ชุดนี้คือเพื่อแสดงถึงหมายเลขของเน็ตเวิร์ก และหมายเลขของโฮสต์ มีการแบ่งเน็ตเวิร์กออกเป็นขนาดต่างๆ ซึ่งจะมีผลต่อจำนวนของโฮสต์ที่มีอยู่ได้ ในเน็ตเวิร์กแต่ละชนิดด้วย

- Class A

Class A ประกอบด้วยเน็ตเวอร์หมายเลข 1.0.0.0 จนถึง 127.0.0.0 หมายเลขของโฮสต์จะประกอบด้วยตัวเลข 24 บิต ซึ่งจะทำให้มีโฮสต์ถึง 1.6 ล้านโฮสต์

- Class B

Class B ประกอบด้วยเน็ตเวิร์กหมายเลข 128.0.0.0 จนถึง 191.255.0.0 หมายเลขของโฮสต์จะมี 16 บิต มีเน็ตเวิร์กได้ 16320 เน็ตเวิร์ก และมีโฮสต์ได้ 65024 โฮสต์ สำหรับแต่ละเน็ตเวิร์ก

- Class C

Class C ประกอบด้วยเน็ตเวิร์กตั้งแต่ 192.0.0.0 จนถึง 223.255.255.0 จะมีเน็ตเวิร์กได้จำนวน 2 ล้านเน็ตเวิร์ก ซึ่งแต่ละเน็ตเวิร์กจะมีจำนวนโฮสต์ได้ 254 โฮสต์

- Class D, E และ F

มีแอดเดรสตั้งแต่ 224.0.0.0 ถึง 254.0.0.0 อยู่ในระหว่างการทดลองและสำรองไว้เผื่ออนาคต ไม่ได้กำหนดจำนวนเน็ตเวิร์กใดๆ

ตัวอย่างเช่น IP หมายเลข 149.76.12.4 จะหมายถึงโฮสต์ 12.4 บนเน็ตเวิร์ก Class B หมายเลขที่ 149.76.0.0 ในกรณีหมายเลขเป็น 149.76.255.255 จะหมายถึงโฮสต์ทุกตัวที่ อยู่บนเน็ตเวิร์กหมายเลข 147.76.0.0

มีเน็ตเวิร์กอีกสองหมายเลขที่ถูกสำรองไว้ใช้งานเฉพาะด้านคือ เน็ตเวิร์กหมายเลข 0.0.0.0 (default root) และ 127.0.0.0 (loopback address) loopback จะเป็นช่องทางพิเศษที่ใช้ติดต่อกับโฮสต์เครื่องนั้นเอง บางทีจะใช้สำหรับการทดสอบเน็ตเวิร์กบางอย่าง โดยไม่จำเป็นจะต้องต่อกับเน็ตเวิร์กจริง

เน็ตมาสก์ (Netmask)

เน็ตมาสก์เป็นตัวบอกให้เราทราบได้ว่า IP ใดอยู่ในเน็ตเวิร์กเดียวกันบ้าง ตัวอย่าง ถ้าหากเน็ตมาสก์ มีค่าเป็น 255.255.255.0
ดังนั้นโฮสต์ที่มีหมายเลข IP เป็น 149.108.10.1 ที่มีค่าเน็ตมาสก์เป็นค่าข้างต้น เรารู้ว่าโฮสต์ตัวนี้จะอยู่ในเน็ตเวิร์กเดียวกันกับ โฮสต์ตัวอื่นๆที่มีหมายเลข IP ขึ้นต้นด้วย 149.108.10

โดเมนเนมเซอร์ฟเวอร์ (Domain Name Server : DNS)

โดเมนเนมเซอร์ฟเวอร์จะทำหน้าที่คล้ายกับสมุดโทรศัพท์ ซึ่งเราสามารถจะเปิดหาชื่อของบุคคลที่เราต้องการจะติดต่อ เพื่อค้นหาเบอร์โทรศัพท์ของเขาได้ ในทำนองเดียวกันหมายเลข IP address มีลักษณะเป็นตัวเลขที่มนุษย์ค่อนข้างจะจดจำได้ยาก จึงมีการเปลี่ยนมาใช้เป็นตัวอักษรสำหรับแทนหมายเลข IP address ต่างๆ และมีเครื่องเซอร์ฟเวอร์ ที่คอยทำหน้าที่บริการเปลี่ยนจากตัวเลข IP ให้กลายเป็นตัวอักษรได้

เกตเวย์ (Gateway)

เกตเวย์เป็นหนทางการเชื่อมต่อระหว่างเครือข่าย 2 เครือข่ายเข้าด้วยกัน สำหรับในกรณีที่มีโฮสต์ที่อยู่ในเครือข่ายที่เป็นซับเน็ต (subnet : เครือข่ายย่อย) เดียวกัน โฮสต์ต่างๆเหล่านั้นจะสามารถทำการติดต่อกันได้ แต่ถ้าหากจะต้องการติดต่อกับโฮสต์ ที่อยู่คนละซับเน็ตกันแล้วจำเป็นจะต้องอาศัยเกตเวย์เข้ามาช่วย เกตเวย์จะเปรียบเสมือนเป็นโฮสต์ที่เป็นที่รับรู้อยู่ ทั้งสองซับเน็ตและจะคอยทำการส่งผ่านข้อมูลระหว่างซับเน็ตทั้งสองไปมา สำหรับการหาเส้นทางการสื่อสารของเครือข่ายจำเป็นจะต้องอาศัยเครื่องหาเส้นทาง (router) เข้ามาช่วย

2.1.5 สายสัญญาณที่ใช้ในการส่งข้อมูล

1. สายคู่ตีเกลียว (twisted pair) สายคู่ตีเกลียวเป็น สายที่มีฉนวน 2 เส้นนำมาทำการตีเกลียวอย่างสม่ำเสมอตลอดระยะทางของสาย ปกติจะใช้กันมากในระบบโทรศัพท์ สายที่ลากจากชุมสายมายังบ้านจะเป็นสายคู่ตีเกลียว สายแบบนี้สามารถนำสัญญาณได้หลายกิโลเมตร โดยไม่ต้องมีการขยายสัญญาณ แต่ถ้ายาวกว่านี้มากๆจำเป็นจะต้องมีการขยายสัญญาณ สายคู่แบบมีเกลียวสามารถเดินไปด้วยกันหลายๆคู่เป็นมัดๆได้แล้วหุ้มด้วยฉนวนภายนอกอีกชั้นหนึ่ง การตีเกลียวจะช่วยลดสัญญาณรบกวนระหว่างคู่ที่เกิดขึ้นสายคู่แบบตีเกลียวสามารถใช้ได้กับทั้งระบบอนาล็อก

และดิจิตอล ขึ้นอยู่กับขนาดและความยาว ความหนาของสาย ขนาดความเร็วในระดับเมกกะบิทต่อวินาทีก็สามารถทำได้แต่เป็นในระยะทางสั้นๆเท่านั้น เนื่องจากมีราคาถูกและประสิทธิภาพพอใช้จึงทำให้มีการใช้งานอย่างแพร่หลายมาก สายสัญญาณแบบนี้รู้จักกันอีกชื่อหนึ่งว่า 10base-T

2. สายโคแอกเซียล (Coaxial Cable) สายโคแอกเซียล หรือเรียกสั้นๆว่าสายโคแอก แบ่งออกเป็น 2 ชนิดตามที่มีการใช้กันอย่าง แพร่หลายคือสายแบบหนา (Thick Ethernet) และสายแบบบาง (Thin Ethernet) สายโคแอกจะประกอบด้วยลวดทองแดงอยู่ตรงกลางหุ้มด้วยฉนวนพลาสติก 1 ชิ้น แล้วจึงหุ้ม ด้วยทองแดงที่ฉักเป็นผืน แล้วหุ้มภายนอกอีกชั้นหนึ่งด้วยฉนวนโครงสร้างของสายโคแอก สามารถทำให้ส่งข้อมูลมีความเร็วได้ถึง 10 Mbps ระยะทางสูงสุดคือ 200 และ 500 เมตรและมักจะเรียกกันอีกชื่อว่า 10base-2 และ 10base-5

วิธีการต่อสายโคแอกสามารถทำได้ 2 แบบคือใช้ทีจังชั่น (T-juction) และ vampire tap วิธีทีจังชั่นคือตัดสายออกใส่คอนเนคเตอร์ชนิด BNC ที่ปลายสายแล้วนำมา สวมกับทีจังชั่นนำสัญญาณไปใช้ทางด้านที่สามวิธีนี้มักจะใช้กับ Thin Ethernet แต่ถ้าใช้วิธีแวมไพร์แทป จะต้องไม่ตัดสายเพียงแต่เจาะรูตรงกลางและใส่คอนเนคเตอร์แบบพิเศษ ที่เรียกว่าแวมไพร์แทปเข้าไป ซึ่งก็จะทำหน้าที่คล้ายกับทีจังชั่น เพียงแต่ไม่ต้องตัดสายออกเป็นสองเส้น วิธีนี้มักจะใช้กับสายแบบ Thick Ethernet

นอกจากสายสัญญาณทั้งสองแบบนี้แล้ว ยังมีสายสัญญาณประเภทอื่นอีก เส้นใยแก้วนำแสง (Fiber obtics) เป็นต้น โดยปกติแล้วในระบบ LAN ทั่วไปใช้สายสัญญาณสองแบบข้างต้นดังกล่าวก็ถือว่าเพียงพอแล้ว

ถ้าหากใช้สายแบบสายคู่ตีเกลียว อาจจะซื้อสายที่มีการเข้าหัวต่อให้เรียบร้อยแล้วหรืออาจจะทำการเข้าหัวต่อเองก็ได้ ซึ่งจะต้องมีเครื่องสำหรับการเข้าหัวต่อโดยเฉพาะ โดยปกติสายแบบนี้มักจะต้องใช้คู่กับฮับ (HUP) แต่ถ้าใช้สายแบบสายโคแอก นอกจาก จะต้องทำการต่อแบบทีจังชั่นแล้วยังจะต้องหา ตัวกลืนสัญญาณ (Terminator) มาปิดระหว่างหัวท้ายของสายสัญญาณเส้นนั้นด้วยเพื่อไม่ให้เกิดการสะท้อนของสัญญาณขึ้น

2.2 พื้นฐานของระบบปฏิบัติการไลนักซ์ (Linux)

2.2.1 ระบบปฏิบัติการไลนักซ์ (Linux Operating System)

ไลนักซ์ระบบปฏิบัติการแบบ 32 บิต ที่เป็นยูนิกซ์โคลน สำหรับเครื่องพีซี และแจกจ่ายให้ใช้ฟรี สนับสนุนการใช้งานแบบหลากงาน หลายผู้ใช้ (MultiUser-MultiTasking) มีระบบ X วินโดวส์ ซึ่งเป็นระบบการติดต่อผู้ใช้แบบกราฟฟิก ที่ไม่ขึ้นกับโอเอส หรือฮาร์ดแวร์ใดๆ (มักใช้กัน

มากในระบบยูนิกซ์) และมาตรฐานการสื่อสาร TCP/IP ที่ใช้เป็นมาตรฐานการสื่อสารในอินเตอร์เน็ตมาให้ในตัว

ไลนักซ์มีความเข้ากันได้ (compatible) กับ มาตรฐาน POSIX ซึ่งเป็นมาตรฐานอินเทอร์เฟสที่ระบบยูนิกซ์ส่วนใหญ่จะต้องมีและมีรูปแบบบางส่วนที่คล้ายกับระบบปฏิบัติการยูนิกซ์จากค่าย Berkeley และ System V

โดยความหมายทางเทคนิคแล้วไลนักซ์ เป็นเพียงเคอร์เนล (kernel) ของระบบปฏิบัติการ ซึ่งจะทำหน้าที่ในด้านของการจัดสรรและบริหารโพรเซสงาน การจัดการไฟล์และอุปกรณ์ I/O ต่างๆ แต่ผู้ใช้ทั่วๆไปจะรู้จักไลนักซ์ผ่านทางแอพพลิเคชั่นและระบบอินเทอร์เฟสที่เขาเหล่านั้นเห็น (เช่น Shell หรือ X วินโดวส์) และนอกจากแพลตฟอร์มอินเทลแล้ว ปัจจุบันไลนักซ์ยังได้ทำการพัฒนาระบบเพื่อให้สามารถใช้งานได้บนแพลตฟอร์มอื่นๆด้วย เช่น DEC Alpha , Motorolla Power-PC , MIPS เมื่อสร้างแอพพลิเคชันขึ้นมาบนแพลตฟอร์มใดแพลตฟอร์มหนึ่งแล้วก็สามารถย้ายแอพพลิเคชันไปวิ่งบนแพลตฟอร์มอื่นได้ไม่ยาก

ไลนักซ์มีทีมพัฒนาโปรแกรมที่ต่อเนื่อง ไม่จำกัดจำนวนของอาสาสมัครผู้ร่วมงาน และส่วนใหญ่จะติดต่อกันผ่านทางอินเทอร์เน็ต เพราะที่อยู่อาศัยจริงๆของแต่ละคนอาจจะอยู่ไกลคนละซีกโลกก็ได้ และมีแผนงานการพัฒนาในระยะยาว ทำให้เรามั่นใจได้ว่า ไลนักซ์เป็นระบบปฏิบัติการที่มีอนาคต และจะยังคงพัฒนาต่อไปได้ตราบนานเท่านาน

2.2.2 ประวัติของไลนักซ์

ไลนักซ์ถือกำเนิดขึ้นในฟินแลนด์ ปี คศ. 1980 โดยลีนุส โทรวัลด์ส (Linus Trovalds) นักศึกษาภาควิชาวิทยาการคอมพิวเตอร์ (Computer Science) ในมหาวิทยาลัยเฮลซิงกิ ลีนุส เห็นว่าระบบมินิกซ์ (Minix) ที่เป็นระบบยูนิกซ์บนพีซีในขณะนั้น ซึ่งทำการพัฒนาโดย ศ. แอนดรูว์ ทาเนนบาวม์ (Andrew S. Tanenbaum) ยังมีความสามารถไม่เพียงพอแก่ความต้องการ จึงได้เริ่มต้นทำการพัฒนาระบบยูนิกซ์ของตนเองขึ้นมา โดยจุดประสงค์อีกประการ คือต้องการทำความเข้าใจในวิชาระบบปฏิบัติการคอมพิวเตอร์ด้วยเมื่อเขาเริ่มพัฒนาไลนักซ์ไปช่วงหนึ่งแล้ว เขาก็ได้ทำการชักชวนให้นักพัฒนาโปรแกรมอื่นๆมาช่วยทำการพัฒนาไลนักซ์ ซึ่งความร่วมมือส่วนใหญ่ก็จะเป็นความร่วมมือผ่านทางอินเทอร์เน็ต

2.2.3 ระบบ X Window

X Window (หรือเรียกย่อๆว่า X) เป็นโปรแกรมที่สร้างขึ้นมาเพื่อครอบบนระบบปฏิบัติการอีกทีหนึ่ง และทำให้เราสามารถใช้งานผ่านคอมพิวเตอร์โดยผ่านการแสดงผลที่เป็นรูปภาพแบบกราฟิกร่วมกับเมาส์ โดยมีการแบ่งพื้นที่แสดงผลออกเป็นหลาย "หน้าต่าง" ย่อยหรือ window นั่นเอง ซึ่งเป็นระบบอินเตอร์เฟซที่เรียกว่า GUI (Graphical User Interface)

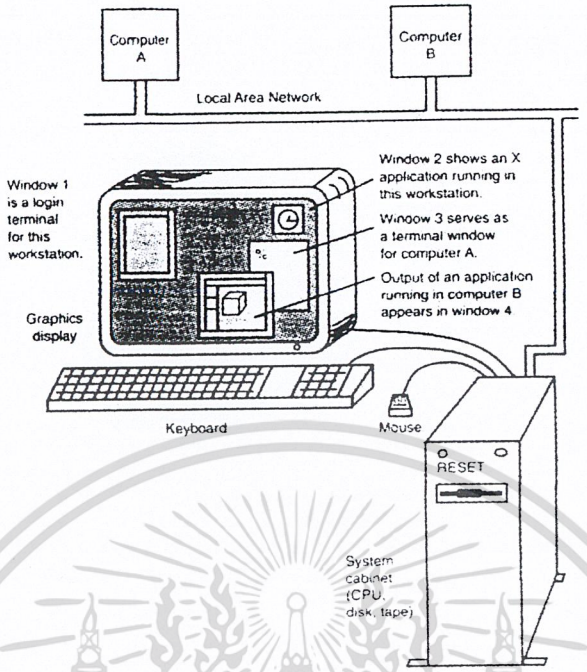
ระบบ X Window ถูกใช้งานกันอย่างมากในคอมพิวเตอร์ตระกูลยูนิกซ์ แต่จริงๆแล้วไม่เพียงแต่ตระกูลยูนิกซ์เท่านั้น แต่ยังมีการใช้งานกับระบบปฏิบัติการเกือบทุกตระกูล แม้กระทั่งบนระบบที่มีความสามารถในการแสดงผลแบบเป็นวินโดว์อยู่แล้ว เช่น Microsoft Windows ด้วย

การพัฒนาเรื่อง GUI บนยูนิกซ์ก็มีประวัติมายาวนานเช่นกัน ในปี คศ. 1987 สถาบันเทคโนโลยีแมสซาชูเซท (Massachusetts Institute of Technology,MIT) ได้ประกาศ snapshot แรกของเวอร์ชันที่ 11 ของระบบ X Window หรือที่รู้จักกันในนาม X11 ซึ่ง X11 ถือว่าเป็นเทคโนโลยีที่สำคัญที่สุดในทศวรรษที่ 1990 (รุ่นล่าสุดจะเป็น X11R6)

X เป็นโครงการที่ถูกพัฒนาร่วมกันระหว่างโปรเจ็กต์อะธีนาของ MIT (Athena Project) และบริษัท Digital Equipment Corporation (DEC) รวมทั้งได้การสนับสนุนจากบริษัทอื่นๆอีกมากมาย โครงการนี้อยู่ภายใต้การนำของ Robert Scheifler และทีมงานของเขาใน MIT

X คือการรวมกันของสิ่งต่างๆเหล่านี้ : X protocol, X display server, X clients และ X lib routines X client เป็นโปรแกรมประยุกต์ที่ใช้การแสดงผลของ workstation เราจะพูดถึงภาพโดยรวมของมัน โดยทั่วไปแล้ว เครื่อง workstation ของเราจะติดต่อกับเครื่อง workstation และเครื่อง minicomputer อื่นๆ โดยวิธี local area network (LAN) ด้วย X ที่ทำงานอยู่บน workstation ของเรา เราสามารถที่จะติดต่อกับมันด้วยกระบวนการหลายๆกระบวนการด้วยกัน แต่ละกระบวนการก็จะแสดงผลของมันออกมาทาง window บนจอภาพ (รูปที่2.4) window 1 เป็นการติดต่อของ workstation ของเรา window นี้ก็คือ terminal นั่นเอง window 2 แสดงผลลัพธ์ของโปรแกรม X ที่ทำงานบนเครื่อง workstation window 3 เป็นอีกหนึ่ง terminal แต่เป็นการติดต่อกับเครื่อง computer A แทน และ window 4 แสดงผลลัพธ์ของโปรแกรม X ที่ทำงานบนเครื่อง computer B

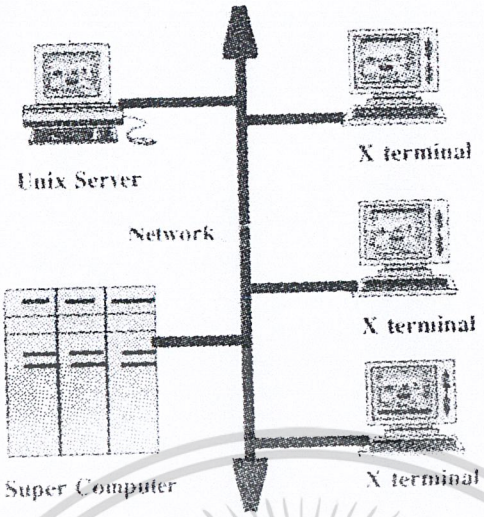
ระบบ X Window เป็นระบบที่พัฒนาขึ้นมาโดยไม่ขึ้นอยู่กับสถาปัตยกรรมแบบใดเลย รวมทั้งมีพื้นฐานอยู่บนระบบเครือข่าย คือมีลักษณะการทำงานแบบ Client-Server โปรแกรมหลักของ X window จะเรียกว่า X Server ซึ่งจะทำหน้าที่จัดทรัพยากรของระบบเช่น สี ภาพ และฟอนต์เป็นต้น



รูปที่ 2.4 เครื่อง workstation กับระบบ X-Window

X Server เป็นเสมือนรากฐานของระบบ แต่ระบบ X Window ที่มีเฉพาะ X Server เพียงอย่างเดียวจะไม่สามารถใช้ประโยชน์อะไรได้ จะต้องมีแอปพลิเคชั่นอื่นๆ ซึ่งแอปพลิเคชั่นเหล่านี้จะถูกจัดเป็น X Client ทั้งหมด โดยที่ X Client เหล่านี้สามารถทำงานข้ามระบบเครือข่ายได้อีกด้วยผลก็คือระหว่างตัว Client กับ Server อาจจะอยู่บนเครื่องเดียวกันหรืออยู่คนละเครื่องก็ได้ คือ คุณสามารถรันแอปพลิเคชั่นบนเครื่องอื่นๆและกำหนดให้มาแสดงผลบนเครื่องของคุณได้

การที่ระบบ X Window มีการแบ่ง X Server กับ X Client นั้นทำให้สถาปัตยกรรมของ X Window มีความแตกต่างจากระบบ window ที่ผูกอยู่กับระบบปฏิบัติการ อย่างเช่น ระบบปฏิบัติการ Windows, OS/2 และ MacOS เป็นต้น ซึ่งความแตกต่างนี้จะอยู่ที่ระบบจัดการวินโดว์ (Window Manager) กล่าวคือระบบปฏิบัติการข้างต้นมีระบบการจัดการวินโดว์ฝังมาในตัวเลย โดยที่เราไม่สามารถเปลี่ยนแปลงได้ แต่ระบบ X Window จะแยกระบบจัดการวินโดว์ ซึ่งทำหน้าที่ในการควบคุมการจัดการวินโดว์ของ X Client ทั้งหมดออกจากตัวมันเอง และมีให้เลือกใช้ได้หลายแบบด้วยกัน โดยเราสามารถเลือก Window Manager ได้มากกว่าหนึ่งแบบ ซึ่งจะทำให้เราไม่ถูกจำกัดการใช้อยู่กับรูปแบบของอินเตอร์เฟซแบบใดแบบหนึ่ง และรูปแบบของอินเตอร์เฟซเหล่านี้ (หมายถึงรูปแบบของการคลิกเมาส์ หรือรูปแบบของไตเติ้ลบาร์ หรืออื่นๆ) ก็สามารถดัดแปลงแก้ไขได้ตามความชอบของผู้ใช้ได้อีกด้วย



รูปที่ 2.5 แสดงระบบ X Client-Server

ดังที่ได้กล่าวไปแล้วว่า X Window เป็นระบบ Client-Server จึงประกอบด้วยส่วนสำคัญ 2 ส่วนคือ ส่วนที่ทำหน้าที่ในการจัดการเกี่ยวกับทรัพยากรต่างๆในระดับต่ำ เช่น ฟอนต์, การควบคุมสี, การแมปคีย์บอร์ด, อุปกรณ์อินพุตข้อมูลต่างๆ ซึ่งเรียกส่วนนี้ว่า ***X Server*** ส่วนประกอบอีกส่วนคือ ส่วนที่เป็นแอปพลิเคชั่นที่ใช้ทรัพยากรต่างๆจาก X Server ส่วนนี้ก็จะเรียกว่า ***X Client*** ตัวอย่างของแอปพลิเคชั่นเหล่านี้เช่น xterm ซึ่งทำหน้าที่เป็นหน้าต่างใช้ในการพิมพ์คำสั่งต่างๆ เป็นต้น

เนื่องจากระบบ X Window เป็นระบบที่ไม่ขึ้นกับระบบปฏิบัติการและฮาร์ดแวร์ใดๆ อีกทั้งยังมีความสามารถในลักษณะของการติดต่อข้ามเครือข่ายอีกด้วย นี่หมายความว่าเราสามารถจะใช้งานโปรแกรมแอปพลิเคชั่น X Client หลายๆตัวบนระบบลีนุกซ์พร้อมกันได้ โดยให้แต่ละแอปพลิเคชั่นมาแสดงผลบนวินโดว์ต่างๆกันผ่านทาง X Server ของเราได้ นอกจากนี้ยังไม่จำกัดว่าจะต้องเป็นแอปพลิเคชั่นบนเครื่องของเราเท่านั้น แต่ยังอาจจะจัดให้แอปพลิเคชั่นบนเครื่องอื่นๆที่อยู่ห่างไกลออกไปมาแสดงผลและรับคำสั่งที่เครื่องของเราได้ราวกับใช้งานอยู่บนเครื่องของเราเครื่องเดียว

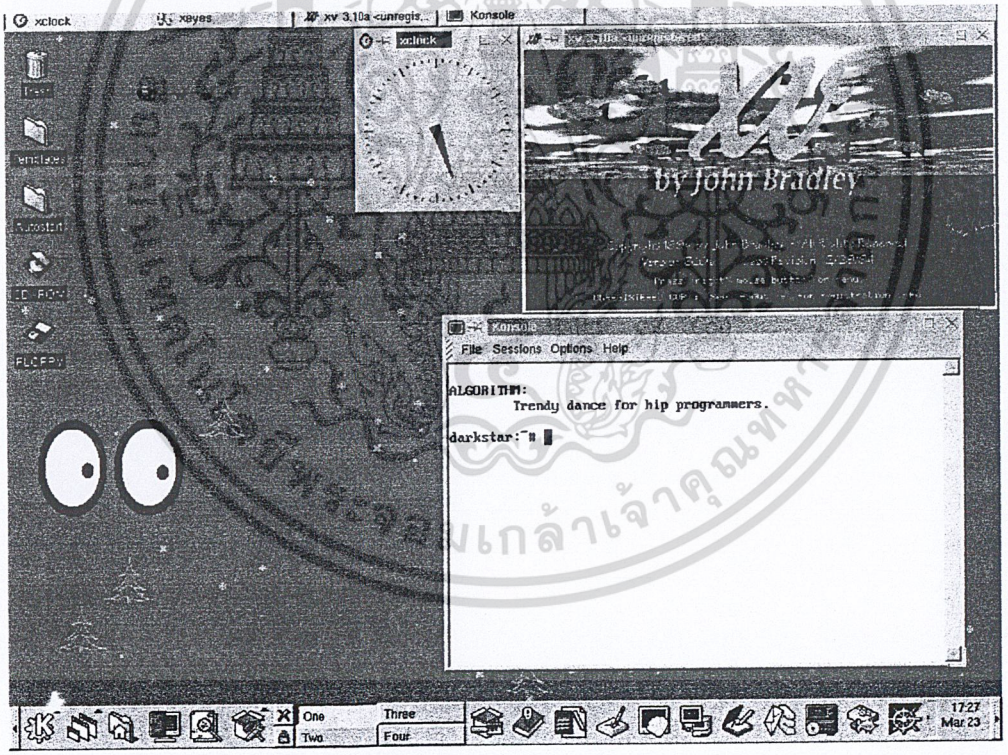
การรันโปรแกรม X Server เพียงโปรแกรมเดียว จะไม่สามารถทำให้เราใช้งานโปรแกรมอะไรได้เลย จะต้องมีสิ่งที่เรียกว่า ***X Environment*** ซึ่งจะควบคุมการใช้งานโปรแกรมในระบบ X อีกทีหนึ่ง X Environment นี้จะควบคุมทั้งลักษณะของหน้าต่าง, การใช้งานเมาส์ และอื่นๆ โดยภายในของ X Environment จะประกอบด้วยโปรแกรม X Client หลายตัว แต่ส่วนประกอบที่สำคัญคือ

Window Manager

ทำหน้าที่ควบคุมการทำงานของวินโดว์ เช่น สร้างวินโดว์ใหม่, ยุบวินโดว์ (Minimize), ขยายวินโดว์ (Maximize), เคลื่อนย้ายวินโดว์ (Move) เป็นต้น แต่เดิม X Environment จะมีแค่ Window Manager เช่น TWM เพียงตัวเดียว แต่มาภายหลังก็มีการเพิ่ม Tool และ Utility ต่างๆขึ้นมามากมาย เพื่อให้ใช้งานสะดวกและง่ายมากยิ่งขึ้น ระบบที่แพร่หลายที่สุดในปัจจุบันคือ Motif (ของ Open Software Foundation หรือ OSF)ซึ่งถือว่าเป็นมาตราฐานของระบบ X Window บนยูนิกซ์

Desktop Manager

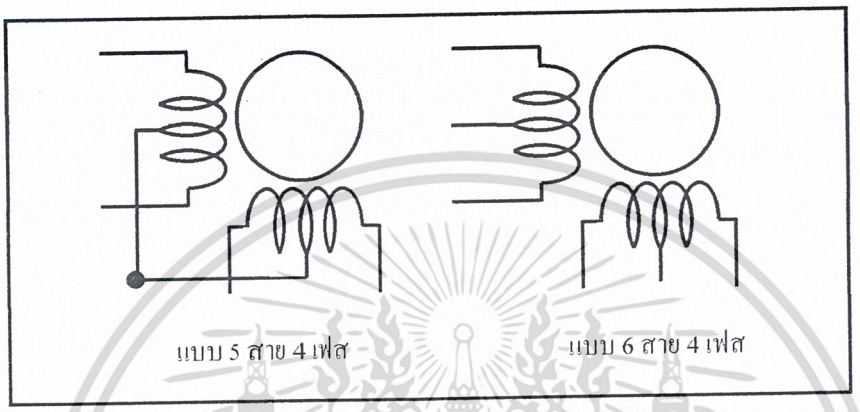
เป็นโปรแกรมที่เกิดขึ้นภายหลัง ทำหน้าที่ควบคุมเดสก์ทอปแบบใหม่ เช่น เรื่องการเปลี่ยน Wallpaper, Icon บนเดสก์ทอป เป็นต้น ซึ่งทำให้การใช้งาน X Window ง่ายและเป็นมิตรกับผู้ใช้มากยิ่งขึ้น ระบบแบบนี้ที่กำลังพัฒนาอยู่คือ KDE และ GNOME



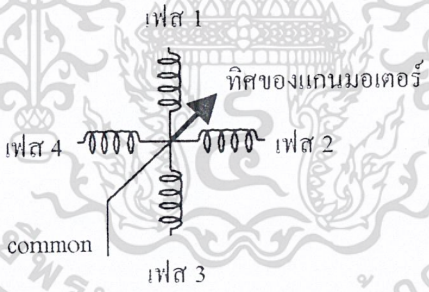
รูปที่ 2.6 หน้าตาของ desktop manager จาก KDE (K Desktop Environment)

2.3 สเต็ปปิ้งมอเตอร์ (Stepping Motor)

สเต็ปปิ้งมอเตอร์ได้รับการพัฒนามาอย่างต่อเนื่อง จนในปัจจุบันสเต็ปมอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุด และหาได้ง่ายคือ สเต็ปปิ้งมอเตอร์แบบยูนิโพล่าร์ (uni-polar stepping motor) มีลักษณะการพันขดลวดของมอเตอร์แสดงในรูปที่ 2.7



รูปที่ 2.7 แสดงลักษณะการพันขดลวดของมอเตอร์แบบยูนิโพล่าร์



รูปที่ 2.8 แสดงการวางของขดลวดแต่ละเฟสของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1,2,3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปปิ้งมอเตอร์แบบนี้มีทั้งแบบ 5 สายและ 6 สาย ถ้าเป็นแบบ 5 สาย จะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว

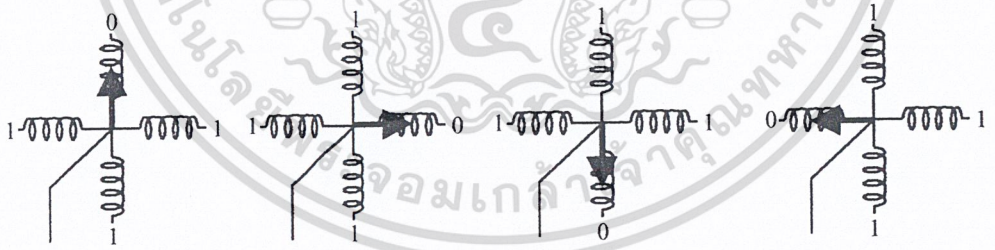
การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดทีละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้อง

ด้วยสามารถแบ่งได้เป็น 3 รูปแบบคือ แบบหนึ่งเฟส,แบบ 2 เฟส (two phase) และแบบครึ่งสเต็ป (half step)

แบบหนึ่งเฟสหรือแบบฟูลสเต็ป (full step) เป็นการกระตุ้นที่มีรูปแบบง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไล่เรียงถัดกันไป เช่น เริ่มต้นที่ขดที่ 1,2,3,4 แล้ววนกลับมาขดที่ 1 วนไปเรื่อยๆ หรือเริ่มที่ขดที่ 1 แล้วย้อนไปยังขดที่ 4,3,2 แล้วกลับมายังขดที่ 1 อีกครั้ง ซึ่งทำให้ทิศทางของการหมุนสวนกันในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังในตาราง 2.2

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

ตารางที่ 2.2 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบหนึ่งเฟส



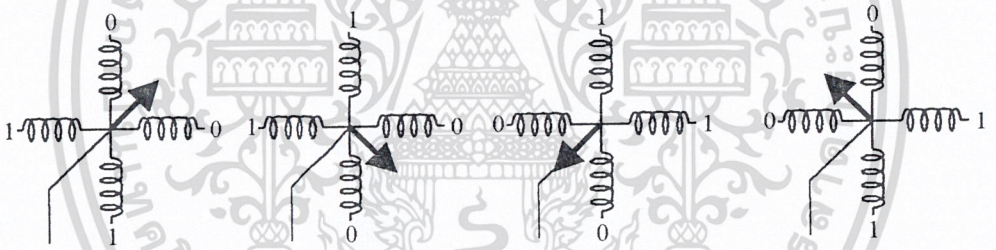
รูปที่ 2.9 แสดงการหมุนของแกนมอเตอร์เมื่อได้รับการกระตุ้นแบบหนึ่งเฟส

แบบ 2 เฟสเป็นการกระตุ้นซึ่งคล้ายกับแบบหนึ่งเฟส แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปเช่นเดียวกับแบบเวฟ ดังตัวอย่าง ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 ถัดไปเป็นขดที่ 4 และ 1 แล้วกลับมาที่ขดที่ 1 และ 2 วนไปตามลำดับเช่นนี้หรือเริ่มที่ขด 1 และ 4 ตามด้วยขดที่ 4 และ 3 ถัดไปเป็นขดที่ 3 และ 2 ต่อไปเป็นขด

ที่ 2 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 4 ทิศทางการหมุนจะสวนทางกัน การกระตุ้นสเต็ปปิ้งมอเตอร์แบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบ 1 เฟส โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือการกระตุ้นแบบนี้ต้องใช้กำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 2.3

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตารางที่ 2.3 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบสองเฟส



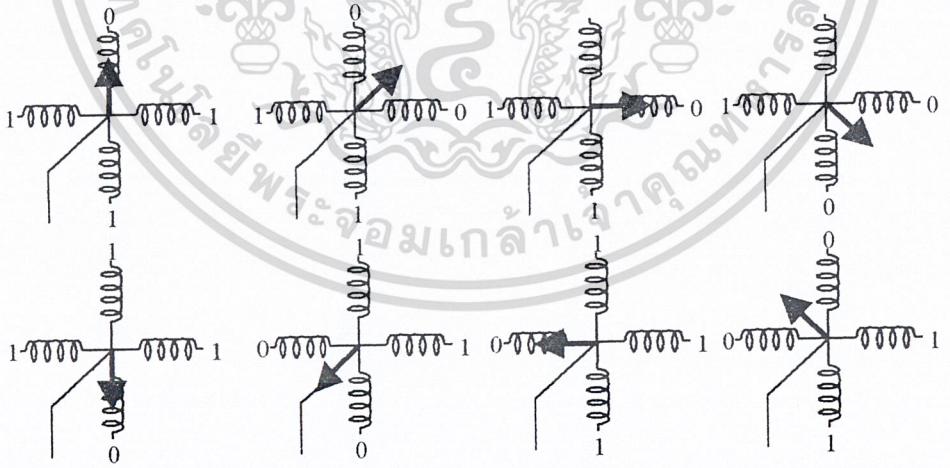
รูปที่ 2.10 แสดงการหมุนของแกนมอเตอร์เมื่อได้รับการกระตุ้นแบบสองเฟส

แบบครึ่งสเต็ปเป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบหนึ่งเฟสและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ เริ่มจากขดลวดที่ 1,1 และ 2,2,2 และ 3,3,3 และ 4,4,4 และ 1 แล้ววนกลับมายังขดลวดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังไว้อีกประการหนึ่งว่า เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2

สเต็ป จึงจะได้เท่ากับระยะเท่ากับ 1 สเต็ปเต็มของการควบคุมใน 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อย จึงจะเพียงพอ ขั้นตอนการทำงานต่างๆ แสดงดังในตารางที่ 2.4

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

ตารางที่ 2.4 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบครึ่งสเต็ป



รูปที่ 2.11 แสดงการหมุนของแกนมอเตอร์เมื่อได้รับการกระตุ้นแบบครึ่งสเต็ป

2.4 มอเตอร์ชนิดกระแสตรง(DC Motor)

DC มอเตอร์เป็นทรานสดิวเซอร์แรงบิดซึ่งมีคุณลักษณะพิเศษคือแรงบิดของเพลาของ DC มอเตอร์จะเป็นสัดส่วนโดยตรงกับกระแสอาร์มาเจอร์ แรงบิดของเพลาของ DC มอเตอร์จะได้จากผลระหว่างสนามแม่เหล็กและขดลวดตัวนำ หลักการนี้แสดงได้ในรูป ในที่นี้กระแสที่ไหลในขดลวดตัวนำจะสร้างสนามที่ประกอบด้วยเส้นแรงแม่เหล็กϕ และขดลวดตัวนำเหล่านั้นอยู่ห่างจากศูนย์กลางการหมุนเท่ากับ r ความสัมพันธ์ระหว่างแรงบิดของเพลาและกระแสเท่ากับ

$$T = K\phi I \qquad \text{----- (2)}$$

เมื่อ T คือแรงบิดของเพลามีหน่วยเป็นนิวตัน-เมตร

ϕ คือเส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

I คือกระแสเป็นแอมแปร์

และ K คือค่าคงตัว ดังนั้นแรงบิดของเพลาจะเป็นสัดส่วนโดยตรงกับผลคูณของเส้นแรงแม่เหล็กและกระแสเมื่อขดลวดตัวนำเคลื่อนที่ในสนามแม่เหล็กก็จะทำให้เกิดโวลท์เต็จตกคร่อมตัวมันเอง โวลท์เต็จนี้จะเป็นสัดส่วนกับความเร็วของเพลาของมอเตอร์และต้านการไหลของกระแส ความสัมพันธ์ระหว่างโวลท์เต็จย้อนกลับนี้และความเร็วของเพลามอเตอร์คือ

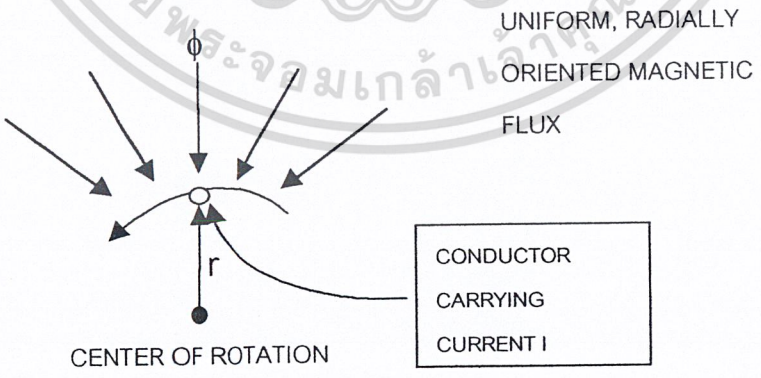
$$E = K\phi\omega \qquad \text{----- (2)}$$

เมื่อ E คือโวลท์เต็จย้อนกลับ emf มีหน่วยเป็นโวลท์

ϕ คือเส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

ω คือความเร็วของมอเตอร์มีหน่วยเป็นเรเดียน/วินาที

สมการ (1) – (2) เป็นสมการที่แสดงถึงหลักการทำงานพื้นฐานของ DC มอเตอร์



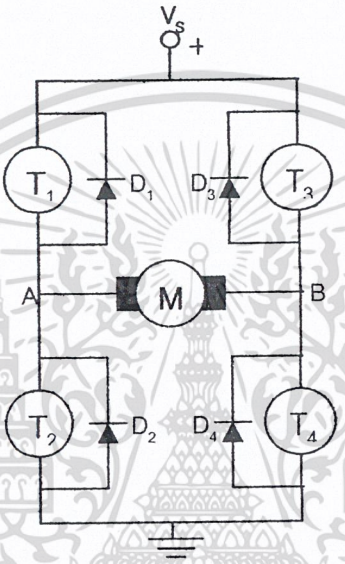
รูปที่ 2.12 แสดงถึงการเกิดแรงบิดในตัว DC มอเตอร์

2.4.1 การทำงานของแอมปลิไฟแบบพัลซ์วิดท์โมดูเลชั่น(Pulse Width Modulation)

แอมปลิไฟแบบ PWM แบบไบโพล่าร์จะดูการทำงานได้ตามรูป โดยที่จะกำหนดให้มีลิมิตความถี่การ สวิทซ์เป็น f_s t_{on} ที่เกิดขึ้นในส่วนแรกและ t_{off} เกิดในส่วนหลังโดย

t_{on} เมื่อ $0 \le t \le t_1$

t_{off} เมื่อ $t_1 \le t \le t_f$



รูปที่ 2.13 แสดงแอมปลิไฟแบบ PWM และ DC มอเตอร์

แบบไบโพล่าร์จะมี T_1 และ T_4 นำกระแสระหว่างเฟส on ส่วน T2 และ T3 จะนำกระแสขณะเฟส off จะได้ฟังก์ชั่นตกคร่อมมอเตอร์เป็น

$$V_m = V_{AB} \begin{cases} V_s & 0 \le t \le t_1 \\ -V_s & t_1 \le t \le t_f \end{cases}$$

แบบยูนิโพล่าร์ จะลดจำนวนทรานซิสเตอร์ในการสวิทซ์ลงการสวิทซ์ขึ้นกับ V_{in} เป็นบวกหรือลบ เมื่อ V_{in} เป็นบวก T_4 จะนำกระแสตลอดคาบ ในขณะที่ T_1 นำกระแสในช่วงเฟส on และ T_2 จะนำกระแสในช่วงเฟส off เมื่อ V_{in} เป็นลบ T_2 จะนำกระแสตลอด โดยมี T_3 และ T_4 สลับกันทำงาน เมื่อ V_{in} เป็นบวกได้

$$V_{in} \begin{cases} V_s & 0 \le t \le t_1 \\ 0 & t_1 \le t \le t_f \end{cases}$$

การแสดงค่า V_{in} ในทางลบจะเหมือนกันพียงแต่ V_{in} เป็นลบเท่านั้น

จากลักษณะของ 2 แบบดังกล่าวมานั้นมีประโยชน์เหมือนกัน ซึ่งในแต่ละกรณีจะมีทรานซิสเตอร์คู่หนึ่ง(T_1,T_2) หรือ (T_3,T_4)จะหยุดนำกระแสขณะที่อีกคู่นำกระแสซึ่งมีเวลาเก็บสะสมและเวลาที่ปล่อยออกของทรานซิสเตอร์เกิดขึ้นและมันอาจเป็นไปได้ที่ทรานซิสเตอร์ทั้งหมดนำกระแสในเวลาเดียวกัน ซึ่งจะทำให้เกิดการลัดวงจรของซัพพลาย เราจำเป็นต้องหลีกเลี่ยงภาวะดังกล่าว ซึ่งสามารถทำได้โดยการสร้างช่วง delay time ระหว่างการหยุดและการนำกระแสของทรานซิสเตอร์และด้วยเหตุผลดังกล่าว ความถี่ของการสวิทซ์จะถูกจำกัดในวงที่แคบลง

2.5 การทำงานของซีซีดี (CCD : Charge Coupled Device)

อุปกรณ์ซีซีดี (CCD : Charge Coupled Device) ถือกำเนิดขึ้นมาหลายทศวรรษแล้ว และได้มีการพัฒนาขึ้นมาอย่างรวดเร็วเมื่อประมาณปี พ.ศ. 2493 หลังจากมีการวิจัยเกี่ยวกับการผลิตทรานซิสเตอร์รูปแบบใหม่ประสบผลสำเร็จ ต่อจากนั้นจึงมีการพัฒนาไปสู่การผลิตวงจรรวมหรือไอซี (IC : Integrated Circuit) ด้วยเทคโนโลยีมอส (MOS : Metel Oxide Semiconductor) และในที่สุดก็ประสบผลสำเร็จอีกครั้งกับการผลิตวงจรโซลิดสเตด ซีซีดี สำหรับใช้งานเป็นอุปกรณ์ถ่ายภาพ เป็นอุปกรณ์ที่อาศัยหลักการทำงานโดยการถ่ายเทของประจุไฟฟ้า ประกอบด้วยส่วนสำคัญใหญ่ๆ 3 ส่วนคือ

1. ส่วนที่ไวแสง (storage) ซึ่งอาจเป็นรอยต่อ p-n ของโฟโตไดโอดหรือรอยต่อของ MOS

2. ส่วนที่ทำหน้าที่ถ่ายโอนสัญญาณ(transfer) ซึ่งผลิตจากแอนะลอกชิฟต์รีจีสเตอร์ (analog shift register)

3. วงจรสัญญาณเอาต์พุต

ถ้าแบ่งอิมเมจเซ็นเซอร์ชนิดซีซีดี ออกตามชนิดของการจัดเรียง (arrays) ของจุดภาพจะแบ่งได้เป็น 2 ชนิดคือ ชนิดหนึ่งมิติ และ ชนิดสองมิติ ในชนิดสองมิติยังแบ่งออกตามวิธีการถ่ายโอน (transfer) ข้อมูลได้อีก 2 ชนิดคือ ชนิด Frame transfer CCD (FT-CCD) และ ชนิด Inter Line Transfer (IL-CCD)

CCD เป็นสิ่งประศิษย์แบบแอกทีฟ ที่ทำงานโดยอาศัยการใช้แรงดันไฟฟ้าที่ทำให้เกิดชั้นปลอดพาหะซึ่งจะทำหน้าที่เปรียบเหมือนหลุมบ่อของศักย์ไฟฟ้าในสารกึ่งตัวนำ เพื่อเก็บพาหะ

อาศัยสัญญาณนาฬิกาพัลส์เพื่อช่วยให้บ่อศักย์ไฟฟ้าเหล่านี้เกิดการเคลื่อนย้ายตำแหน่งไปตามลำดับ ซึ่งจะทำให้เกิดการถ่ายโอนประจุไฟฟ้า

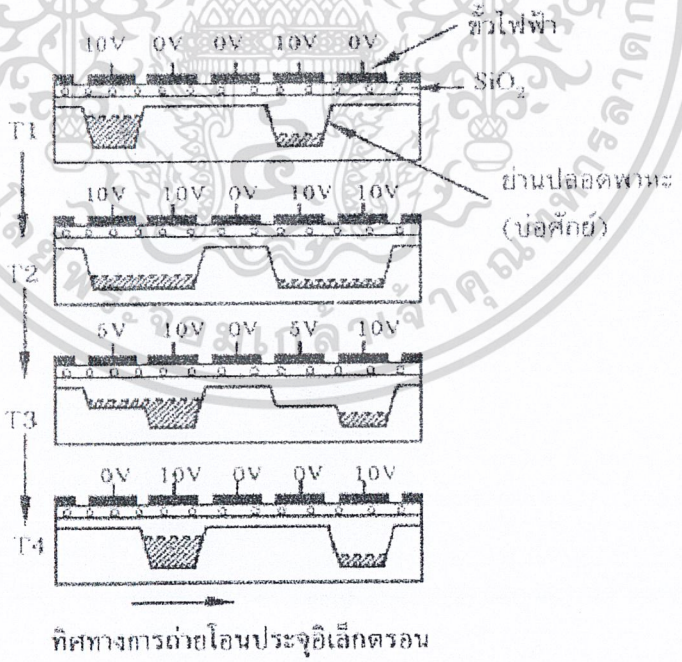
รูปที่ 2.14 แสดงหลักการถ่ายโอนประจุไฟฟ้าใน CCD บนแผ่นฐานสารกึ่งตัวนำมีชั้นฉนวนของ SiO_2 เคลือบอยู่ และมีขั้วถ่ายโอนจำนวนมากเคลือบอยู่ห่างกันเป็นระยะ ๆ ขั้นตอนการไหลถ่ายโอนประจุอิเล็กตรอนและการทำงานของ CCD อธิบายได้ดังนี้

ขั้นตอน T_1 เมื่อป้อนแรงดันบวก (เช่น 10 V) เข้าสู่ขั้วถ่ายโอนจะทำให้เกิดบ่อศักย์ไฟฟ้าลึกลงไปที่ใต้ขั้วนั้น และเมื่อมีแสงอินพุตเข้ามากระทบ CCD จะทำให้เกิดพาหะโฟโตและพาหะโฟโตเหล่านี้จะถูกเก็บอยู่ในบ่อศักย์ไฟฟ้าลึกๆ เท่านั้น

ขั้นตอน T_2 ใช้สัญญาณพัลส์ (เช่น 10 V) ป้อนไปที่ขั้วถ่ายโอนที่อยู่ข้างๆ จะทำให้บ่อกว้างขึ้นและอิเล็กตรอนก็จะไหลแผ่วงกว้างขึ้น

ขั้นตอน T_3 ลดขนาดแรงดันที่ขั้วเดิมให้เหลือประมาณ 5 V จะทำให้อิเล็กตรอนไหลลงไปสู่บ่อข้างเคียงที่มีแรงดัน 10 V

ขั้นตอน T_4 เมื่อตัดแรงดันที่ขั้วเดิมออกหมด จะทำให้อิเล็กตรอนทั้งหมดขยับไปอยู่ในบ่อข้างเคียง



รูปที่ 2.14 แสดงหลักการถ่ายโอนประจุไฟฟ้าใน CCD

ด้วยวิธีดังกล่าวข้างต้นนี้และการใช้ทรานส์เฟอร์พัลส์ป้อนเข้าสู่ขั้วถ่ายโอนตามลำดับทีละขั้ว เช่นจากขั้วด้านซ้ายไปสู่ขั้วด้านขวา จะทำให้อิเล็กตรอนถูกเคลื่อนย้ายไปตามบ่อศักย์ไฟฟ้าใน CCD จากซ้ายไปขวาและในที่สุดอิเล็กตรอน (สัญญาณอิมเมจ) ก็จะไหลออกไปสู่วงจรภายนอกเป็นสัญญาณเอาต์พุตของอิมเมจได้ตามต้องการ

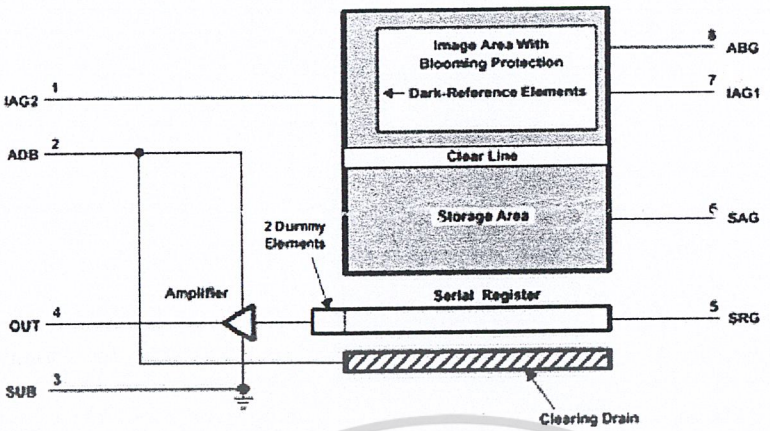
2.5.1 B/W Connectix Quickcam

B/W Connectix Quickcam เป็น กล้องซีซีดี (CCD Camera) ติดต่อกับคอมพิวเตอร์ได้โดยใช้พอร์ตเครื่องพิมพ์ และใช้ไฟเลี้ยงวงจรจากข้อต่อ ADB (Key Board) ตัวของ Image Sensor ชิบ TC255P ซึ่งเป็น CCD Image Sensor ขนาด 336x244 พิกเซล ผลิตโดย TEXAS INSTRUMENTS ตัวกล้อง B/W Connectix Quickcam มีลักษณะดังรูปที่ 2.15

รูปที่ 2.15 แสดงตัวกล้อง B/W Connectix Quickcam

2.5.2 TC255P

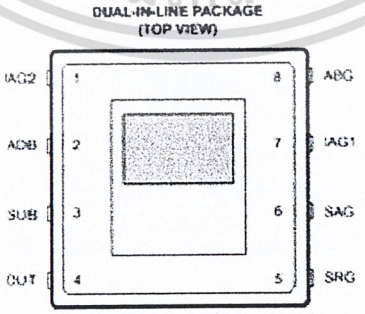
TC255P เป็นซีซีดีชนิด FT-CCD (Frame-Transfer charge-coupled device) ออกแบบสำหรับใช้เป็น B/W NTSC TV และ การประยุกต์ใช้กับงานต่างที่ออกแบบให้มีขนาดเล็ก พื้นที่ของ image-sensing มีจำนวน 243x336 sensor สามารถทำงานได้ในโหมด noninterace ที่ 324 แกนนอน และ 243 แกนตั้ง TC255P ทำงานได้ในช่วงอุณหภูมิ $-10^{\circ}C$ ถึง $45^{\circ}C$



รูปที่ 2.16 แสดงการทำงานของ TC255P (FT-CCD)

คุณสมบัติของ TC255P

1. มีความละเอียดปานกลาง เป็น Solid-State Image Sensor สำหรับการประยุกต์ Low-cost B/w TV
2. มีขนาดของ Active Element ในพื้นที่ Image Sensor 324(H)x243(V)
3. พิกเซลมีขนาด 10x10 μm
4. การทำงานของชัตเตอร์อิเล็กทรอนิกส์ใช้เวลา 1/60 –1/50000 วินาที
5. มีกระแสมืดต่ำ (Low dark current)
6. มีความไวสูง
7. ตัวถังของ TC255P เป็นแบบ DIP 8 ขา



รูปที่ 2.17 แสดงตัวถังและขาของ TC255P

2.6 ระบบการสื่อสารข้อมูลของคอมพิวเตอร์

2.6.1 ระบบการส่งข้อมูลแบบดิจิตอล

ลักษณะข้อมูลที่ใช้ในระบบนี้จะอยู่ในรูปฟอร์มของตัวเลข '0' กับ '1' เช่น ตัวเลขฐานสอง, ตัวเลขฐานสิบหก เป็นต้น บางครั้งอาจมีความต้องการส่งสัญญาณอนาลอกผ่านทางระบบดิจิตอลจึงต้องมีการเปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลเสียก่อนเรียกว่าการสุ่มตัวอย่าง (Sampling) ซึ่งเป็นวิธีทางคณิตศาสตร์ ค่าที่ได้จากการสุ่มตัวอย่างจัดเป็นเลขฐานสอง (Binary Code) ที่สามารถจัดตามเทคนิคทางสัญญาณดิจิตอลได้เช่น การส่งข้อมูลแบบขนานหรืออนุกรม และ แบบซิงโครนัสหรือแบบอะซิงโครนัส เป็นต้น

ทิศทางของการส่งสัญญาณ

เราสามารถแบ่งทิศของการส่งสัญญาณได้ 3 วิธีคือ

1. การส่งผ่านแบบทิศทางเดียว (Simplex)
2. การส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex)
3. การส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex)

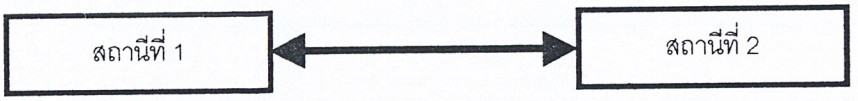
การส่งผ่านแบบทิศทางเดียว (Simplex)

หมายถึง รูปแบบการส่งสัญญาณให้ด้านรับได้ฝ่ายเดียว โดยไม่สามารถโต้ตอบผ่านทางการติดต่อได้เช่น การกระจายเสียงของวิทยุหรือสัญญาณโทรทัศน์ เป็นต้น

รูปที่ 2.18 แสดงการส่งสัญญาณแบบทิศทางเดียว (Simplex)

การส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex)

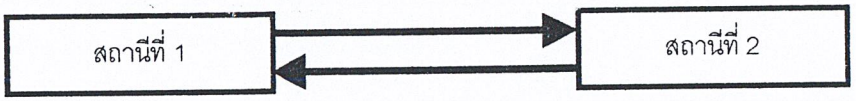
หมายถึง รูปแบบสัญญาณที่สถานีทั้งสองฝ่ายสามารถรับและส่งสัญญาณระหว่างกันได้โดยกำหนดว่าต้องมีด้านใดด้านหนึ่งเป็นตัวรับเสมอเช่น การใช้วิทยุสมัครเล่นในการติดต่อสื่อสาร



รูปที่ 2.19 แสดงการส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex)

การส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex)

หมายถึง รูปแบบการส่งสัญญาณที่ทั้งด้านส่งและด้านรับสามารถที่จะส่งสัญญาณในเวลาเดียวกันได้โดยไม่จำเป็นต้องสลับด้านกันด้วย เช่นการสนทนาทางโทรศัพท์ เป็นต้น



รูปที่ 2.20 แสดงการส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex)

2.6.2 ลักษณะของการรับส่งข้อมูลแบบดิจิตอล

โดยทั่วๆไปแล้วหลักใหญ่ของการรับส่งข้อมูลในรูปแบบของสัญญาณดิจิตอลมีลักษณะการส่งข้อมูล 2 แบบคือ

1. การส่งแบบอนุกรม
2. การส่งแบบขนาน

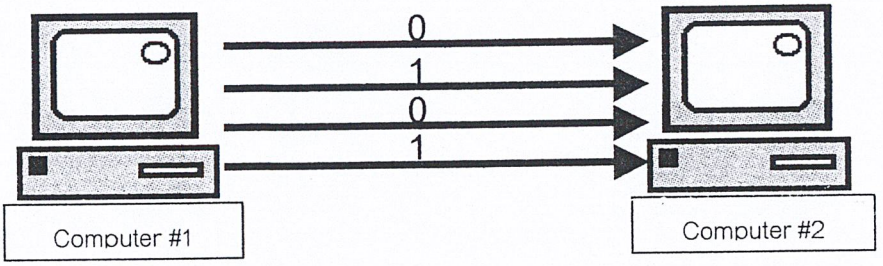
การส่งแบบอนุกรม

ข้อมูลแต่ละบิทจะถูกส่งเรียงออกไปเป็นลำดับต่อเนื่องกันทีละบิท เช่น ข้อมูล 0101 เลข 1 ทางขวาสุดซึ่งเป็นบิทนัยสำคัญต่ำสุดจะถูกส่งออกไปก่อน ตามด้วยเลข 0 เลข 1 และเลข 0 ซึ่งเป็นบิทนัยสำคัญสูงสุดตามลำดับ โดยมีสายส่งอยู่เพียงเส้นเดียว

รูปที่ 2.21 แสดงการส่งข้อมูลแบบอนุกรม

การส่งข้อมูลแบบขนาน

ข้อมูลทุกๆ บิทจะถูกส่งออกไปพร้อมๆ กัน ในครั้งเดียว เช่น ถ้าข้อมูลเป็น 0101 ทั้งสี่บิทจะถูกส่งออกไปพร้อมกันหมด โดยผ่านส่งสัญญาณจำนวน 4 เส้น โดยแต่ละบิทจะส่งในสายคนละเส้นกัน

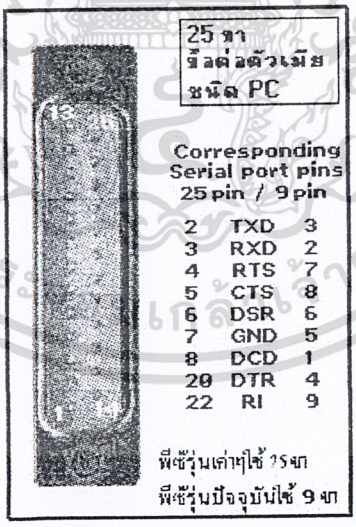


รูปที่ 2.22 แสดงการส่งข้อมูลแบบขนาน

2.6.3 สายเคเบิลของ RS-232C

มาตรฐานของการรับ-ส่งข้อมูลอนุกรม (RS-232C) นี้ ได้กำหนดขึ้นมาเพื่อให้คอมพิวเตอร์ต่างยี่ห้อกัน หรืออุปกรณ์ต่อพ่วงแต่ละชนิดรับ-ส่งข้อมูลกันได้ เมื่อทำตามมาตรฐานนี้ โดยไม่สนใจว่าอุปกรณ์หรือคอมพิวเตอร์นั้นจะผลิตมาจากที่ใด โดยมีการกำหนดรายละเอียดในเรื่องหัวต่อ (Connector) ที่ใช้ สัญญาณในแต่ละเส้น

ข้อต่อ (Connector) ระหว่างสายเคเบิลทั้งสองปลายจะใช้ข้อต่อแบบ 25 ขารูปร่างหน้าตัดคล้ายตัว 'D' เรียกว่า DB-25 ดังรูปที่ 2.23



รูปที่ 2.23 แสดงข้อต่อแบบ DB-25

สายเคเบิลที่ใช้ในการรับ-ส่งข้อมูลส่วนมากจะใช้สายเพียง 9-10 เส้นเท่านั้น โดยขาสัญญาณที่ 1 (Protective Ground) นั้น มักจะไม่จำเป็นต้องต่อใช้งาน จึงเหลือจำนวนสายสัญญาณที่ใช้เพียง 9 เส้น จากข้อต่อแบบ DB-25 สัญญาณแต่ละเส้นเรียงตามลำดับดังนี้

ขาที่ 1 (Protective Ground) เป็นสายดินของอุปกรณ์

ขาที่ 2 (Transmitted Date) ใช้สำหรับส่งข้อมูล

ขาที่ 3 (Received to Send) ใช้สำหรับรับข้อมูล

ขาที่ 4 (Request to Send) เป็นสัญญาณขอทำการส่งข้อมูล

ขาที่ 5 (Clear to Send) เป็นสัญญาณตอบรับว่าเริ่มส่งข้อมูลได้

ขาที่ 6 (Data Set Ready) เป็นสัญญาณแสดงว่าตัวรับพร้อมที่จะรับข้อมูล

ขาที่ 7 (Signal Ground) เป็นสายดินของสัญญาณรับส่ง

ขาที่ 8 (Data Carrier Detect) เป็นตัวบอกว่าทั้งตัวรับและตัวส่งต่อถึงกันเรียบร้อยแล้ว และพร้อมที่จะทำการรับ-ส่งข้อมูล

ขาที่ 20 (Data Terminal Ready) เป็นสัญญาณแสดงว่าตัวส่งพร้อมที่จะส่งข้อมูล

ขาที่ 22 (Ring Indicator) เป็นขาแสดงกริ่งโทรศัพท์ที่เรียกเข้ามา

ข้อต่อแบบ DB-9 และ DB-25

การต่อสายเคเบิลสัญญาณต่างๆของข้อต่อแบบ DB-9และDB-25 แสดงในตารางที่ 2.5

DB-9 Pin	DB-25 Pin	Assignment / Function
1	8	Data Carrier Detect
2	3	Received to Send
3	2	Transmitted Date
4	20	Data Terminal Ready
5	7	Signal Ground
6	6	Data Set Ready
7	4	Request to Send
8	5	Clear to Send
9	22	Ring Indicator

ตารางที่ 2.5 แสดงการเปรียบเทียบขาของข้อต่อแบบ DB-9 กับ DB-25

2.7 การ์ด ET-DIO

การ์ด ET-DIO เป็นลักษณะของ PC CARD ใช้เชื่อมต่อกับเครื่อง PC เพื่อขยายระบบอินพุทและเอาท์พุท ให้ใช้งานได้มากยิ่งขึ้นซึ่ง ET-DIO CARD สามารถที่จะรับสัญญาณอินพุทและให้สัญญาณเอาท์พุทออกมาได้ทั้งในรูปของ Analog และ Digital ทำให้มีความอ่อนตัวในการนำไปประยุกต์ใช้งานด้านต่างๆได้มากยิ่งขึ้น ตัวการ์ด ET-DIO ผลิตโดยบริษัท อีทีที จำกัด

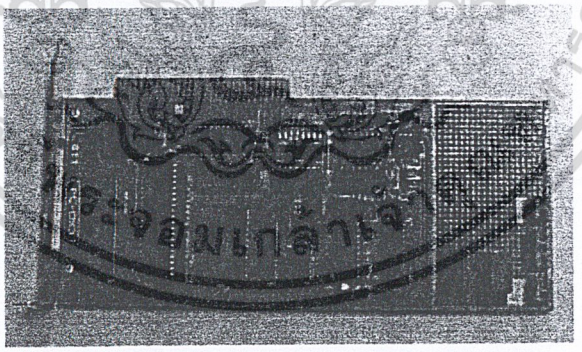
การ์ด ET-DIO มีอุปกรณ์ร่วมดังนี้

1. มีไอซี 8255 (Programmable Pheripheral Interface) จำนวนหนึ่งตัว ซึ่งสามารถที่จะโปรแกรมให้เป็นได้ทั้งอินพุทและเอาท์พุตตามความต้องการของผู้ใช้เองได้ทั้งหมด 3 พอร์ต หรือ 24 บิต I/O ในรูปของสัญญาณ Digital นั่นเอง

2. มีไอซี 8253 (Programmable Interval Timer) จำนวน 1 ตัวทำให้ ET-DIO CARD สามารถที่จะประยุกต์ใช้งานกับระบบฐานเวลาต่างๆ ได้มากมาย

3. มีไอซี ADC (Analog to Digital Converter) จำนวน 1 ตัว ซึ่งสามารถที่จะเลือกใช้ได้ถึง 2 เบอร์คือ ADC0804 และ ADC1001 (10 บิต) ทำให้สามารถทำการรับค่า Analog มาคำนวณได้

4. มีไอซี DAC (Digital to Analog Converter) จำนวน 1 ตัว ซึ่งสามารถที่จะส่งค่า Analog ไปควบคุมการทำงานของอุปกรณ์ภายนอกได้



รูปที่ 2.24 แสดงภาพของการ์ด ET-DIO

2.8 AT89C52 8-Bit Microcontroller with 8 Kbytes Flash

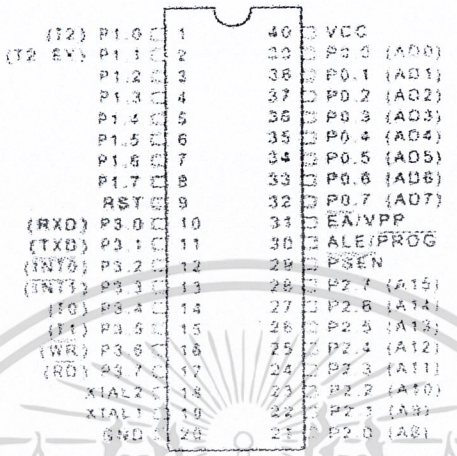
AT89C52 เป็นไมโครคอนโทรเลอร์ 8 Bits ขนาด 40 ขา สร้างด้วยเทคโนโลยี Atmel's High Densiy Nonvolatile Memory Technology โดยมีหน่วยความจำโปรแกรมภายในขนาด 8 Kbytes แบบ Flash Memory หรือที่เรียกว่า Programmable and Erasable Read Only Memory (PEROM) หน่วยความจำข้อมูลภายในขนาด 256 Bytes สร้างด้วยเทคโนโลยี CMOS AT89C52 เป็นไมโครคอนโทรเลอร์ผลิตโดย ATMEL ชุดคำสั่งและสถาปัตยกรรมภายในเหมือนไมโครคอนโทรเลอร์ตระกูล MCS-51 ซึ่งผลิตโดย INTEL ที่ใช้พลังงานต่ำและหน่วยความจำภายใน สามารถเขียนและลบได้ไม่น้อยกว่า 1,000 ครั้ง AT89C52 เป็นไมโครคอนโทรเลอร์ที่มีความสามารถในการเปลี่ยนแปลงสูงและผลที่ได้คุ้มค่าซึ่งสามารถนำมาประยุกต์ใช้ในการควบคุมและสามารถเลือกใช้ซอร์ฟแวร์ในการประหยัดพลังงานได้ 2 โหมด คือ โหมด Idle และ โหมด Power Down

2.8.1 คุณสมบัติของ AT89C52

1. เข้ากันได้กับไมโครคอนโทรเลอร์ตระกูล MCS-51
2. หน่วยความจำโปรแกรมเป็นแบบ Flash Memory ขนาด 8 Kbytes สามารถเขียนหรือลบโปรแกรมได้ถึง 1,000 ครั้งและเก็บข้อมูลได้นานถึง 10 ปี
3. ใช้ไฟเลี้ยงตั้งแต่ 2.7-6 Volts
4. ทำงานได้ในช่วงความถี่ 0 Hz ถึง 24 MHz
5. ระบบหน่วยความจำโปรแกรมมี 3 ระดับ
6. มี I/O พอร์ต 32 บิต
7. มี Counter/Timer ขนาด 16 Bits 3 ตัว
8. พอร์ตอนุกรมแบบ Full Duplex
9. มี Interrupt 8 แหล่ง
10. มีทั้งโหมด Low power Idle และ Power Down

2.8.2 ตัวถังและขาตั้งของชิป AT89C52

ลักษณะเป็นแบบ PDIP(Plastic Dual Inline Package) ขนาด 40 ขา แสดงดังรูปที่ 2.25



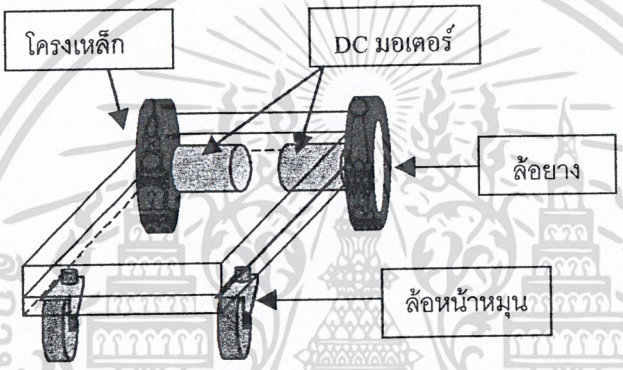
รูปที่ 2.25 แสดงขาของไมโครคอนโทรเลอร์ AT89C52

บทที่ 3
การดำเนินการวิจัย

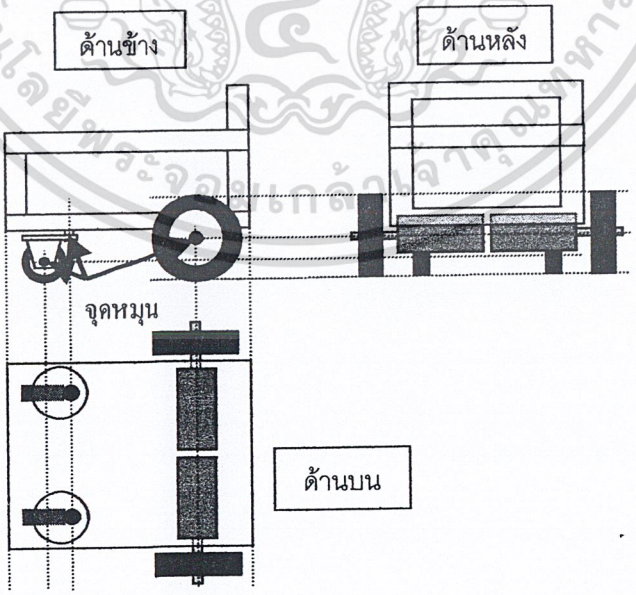
3.1 การออกแบบในส่วนของฮาร์ดแวร์ (Hardware Designed)

3.1.1 การออกแบบตัวรถ เป็นส่วนตัวถังของหุ่นยนต์

ตัวรถ ออกแบบเป็นรถ 4 ล้อ ขับเคลื่อนด้วย 2 ล้อหลังมี DC มอเตอร์ เป็นตัวขับเคลื่อนทั้ง 2 ล้อ ส่วนล้อหน้า 2 ล้อหมุนได้อย่างอิสระ



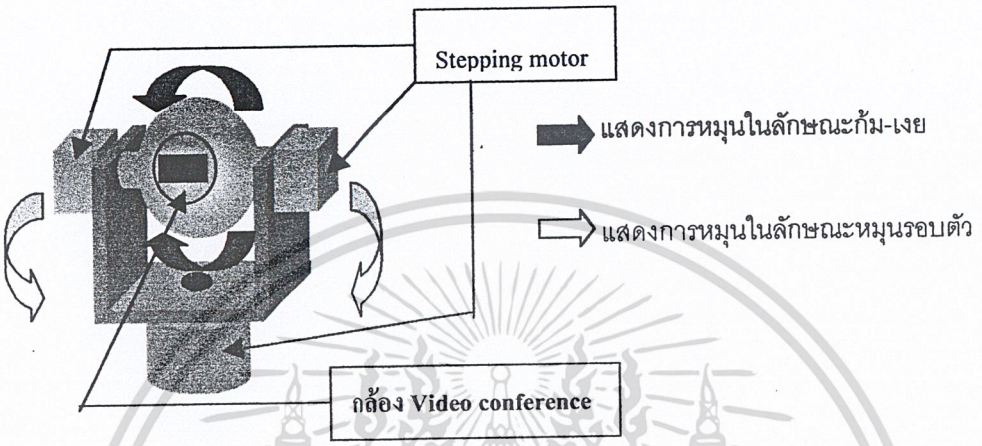
รูปที่ 3.1 แสดงตัวรถที่ออกแบบ



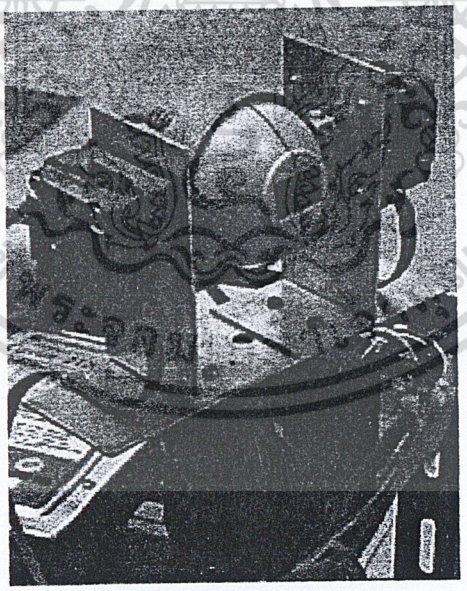
รูปที่ 3.2 แสดงแบบร่างในมุมมองต่างๆ ของตัวรถ

3.1.2 การออกแบบตัวกล้อง เป็นส่วนตาของหุ่นยนต์

ตัวกล้อง ออกแบบให้หมุนได้รอบทิศทาง คือ หมุนรอบตัว และ หมุนเป็นมุมก้มกับเงยได้ โดยใช้ Stepping มอเตอร์เป็นตัวขับ



รูปที่ 3.3 แสดงตัวกล้องที่ออกแบบ



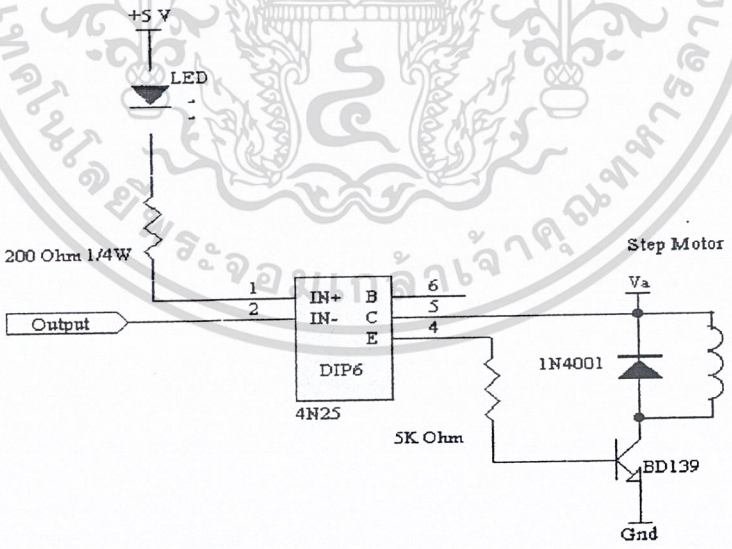
รูปที่ 3.4 แสดงรูปของตัวกล้องที่สร้างขึ้น

3.1.3 การออกแบบบอร์ดควบคุมสเต็ปปิ้งมอเตอร์ และ ดีซีมอเตอร์ ด้วยไมโครคอนโทรเลอร์ AT89C52

3.1.3.1 การควบคุมสเต็ปปิ้งมอเตอร์

ในการควบคุมสเต็ปปิ้งมอเตอร์ จะเป็นกระทำโดยการการสวิตซ์แรงดันแก่ขดลวดต่างๆ ของสเต็ปปิ้งมอเตอร์ ตามลักษณะการกระตุ้นไม่ว่าจะเป็นแบบฮาล์ฟสเตป (Half Step) หรือแบบฟูลสเตป(Full Step) ในการควบคุมสเต็ปปิ้งมอเตอร์ในโครงงานพิเศษฉบับนี้ จะมีวงจรควบคุมอยู่สองชุดคือในชุดแรกใช้ไอซี ULN2803 ซึ่งเป็นไอซีที่ใช้ในการขับสเต็ปปิ้งมอเตอร์โดยเฉพาะซึ่งสามารถขับสเต็ปปิ้งมอเตอร์ขนาด 4 เฟสได้สองตัว ซึ่งจะใช้ในการขับ สเต็ปปิ้งมอเตอร์ที่บังคับการก้มเงยของตัวกล้องกินซึ่งจะกำลังไม่มากนัก

ชุดที่สองใช้ ทรานซิสเตอร์ ชนิด NPN เบอร์ BD139 เป็นตัวสวิตซ์ โดยจะมีไอซี 74LS244 ซึ่งทำหน้าที่เป็นบัฟเฟอร์คั่นระหว่าง ไมโครคอนโทลเลอร์ AT89C52 กับชุดวงจรขับ ในวงจรขับจะประกอบด้วยไอซีออปโตคัปเปิล (Opto Couple) เบอร์ 4N25 ทำหน้าที่เป็นตัวกั้นทางแสง โดยมี LED บอกสถานะต่ออยู่ ถ้าสัญญาณที่ออกจาก 74LS244 เป็น 5 โวลต์ (High level) LED จะดับ แต่ถ้าเป็น 0 โวลต์ (Low level) LED จะสว่าง ที่เอาท์พุทของ 4N25 จะไปต่อกับขาเบสโดยมีตัวต้านทาน 5 กิโลโอห์ม ต่อคั่นอยู่



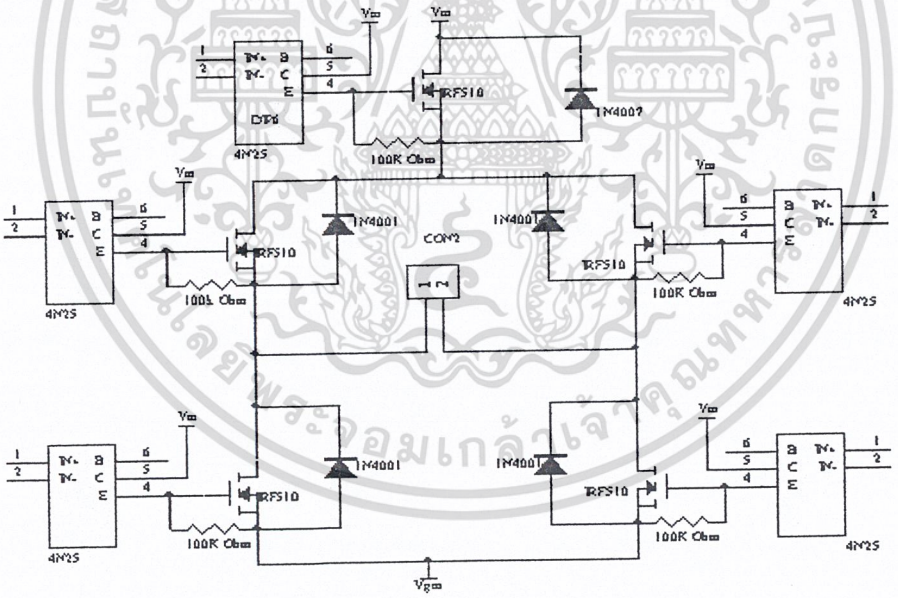
รูปที่ 3.5 แสดงวงจรขับสเต็ปปิ้งมอเตอร์

เมื่อเอาท์พุทจาก 74LS244 เป็น 0 โวลต์ จะทำให้มีกระแสไหลที่ขาเบสโดยจะทำให้ทรานซิสเตอร์อยู่ในสถานะอิ่มตัว (saturation) ทำให้มีกระแสไหลจาก คอลเลกเตอร์ไป อิมิตเตอร์

มากที่สุด ซึ่งจะเป็นช่วงที่เป็นการให้การกระตุ้นกับสเต็ปปิ้งมอเตอร์ และเมื่อเอาท์พุทจาก 74LS244 เป็น 5 โวลต์จะทำให้ไม่มีกระแสไหลเข้าที่ขาเบสทำให้ทรานซิสเตอร์อยู่ในสถานะปิด (Cutoff) คือ ไม่มีกระแสไหลจากคอลเลกเตอร์ไป อิมิตเตอร์ เป็นช่วงที่หยุดการให้การกระตุ้นกับ สเต็ปปิ้งมอเตอร์ ซึ่งในวงจรนี้จะสามารถใช้ขับ สเต็ปปิ้งมอเตอร์ที่มีกำลังสูงได้ ซึ่งจะใช้ในการขับสเต็ปปิ้งมอเตอร์ที่บังคับทิศทางซ้าย – ขวาของตัวกล้อง

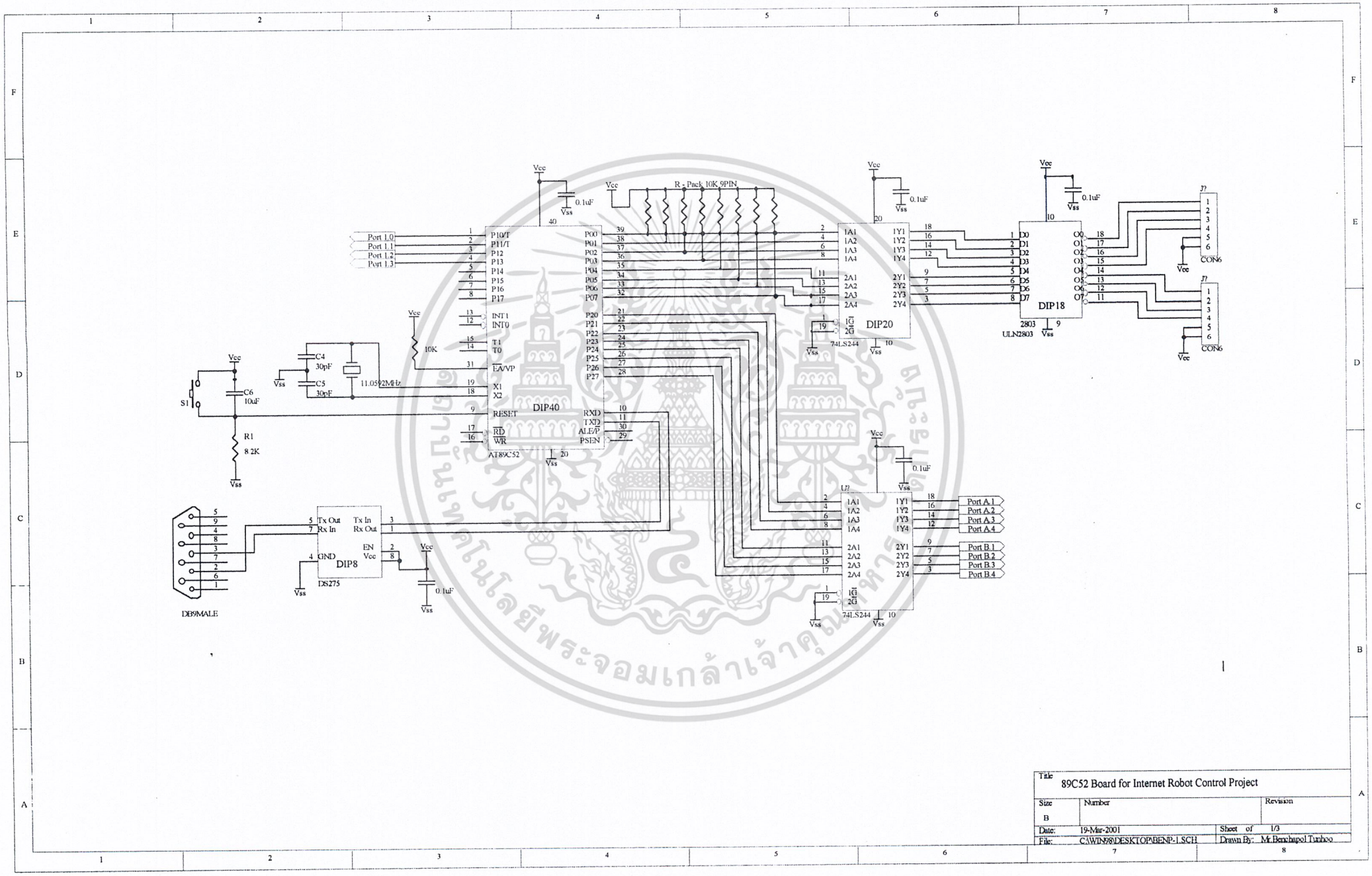
3.1.3.2 การควบคุม DC Motor

ในการควบคุมมอเตอร์กระแสตรงจะใช้วงจรที่เรียก H- Bridge ซึ่งเราสามารถบังคับทิศทางการเคลื่อนที่และความเร็วของมอเตอร์กระแสตรงโดยใช้ขาเอาท์พุทเพียง 2 ขาเท่านั้น ในโครงงานพิเศษนี้ ใช้ไอซี MOSFET ชนิด n – channel เบอร์ IRF510 ต่อเป็นวงจร H – Bridge โดยในการทำงานจะต้องให้ MOSFET ที่อยู่ตรงข้าม 2 ตัวทำงานพร้อมกัน เพื่อให้สามารถบังคับทิศทางได้โดยจะมีไอซี 74LS244 ซึ่งทำหน้าที่เป็นบัฟเฟอร์คั่นระหว่าง ไมโครคอนโทลเลอร์ AT89C52

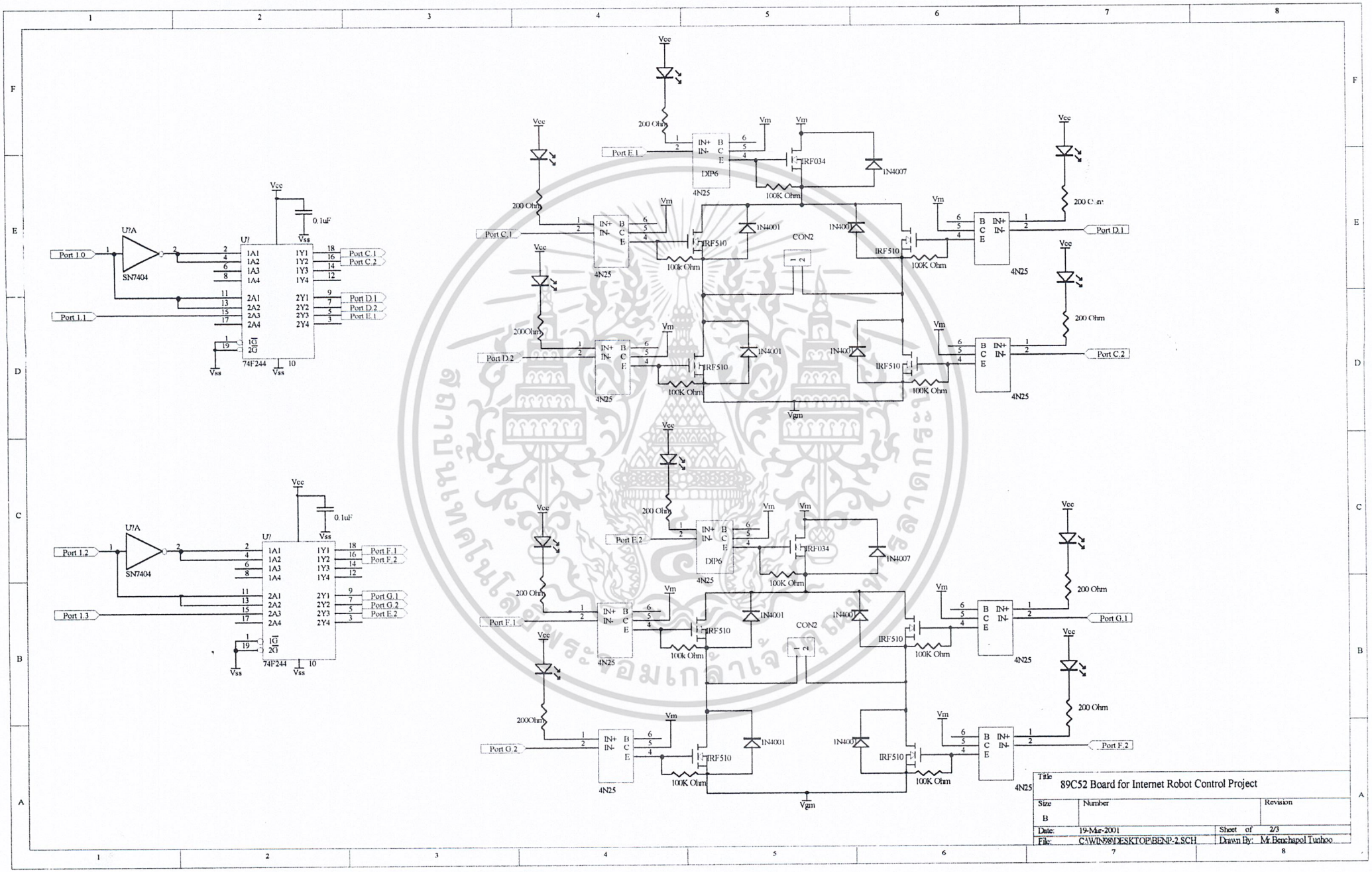


รูปที่ 3.6 แสดงส่วนของวงจรขับ DC Motor ด้วย IRF510

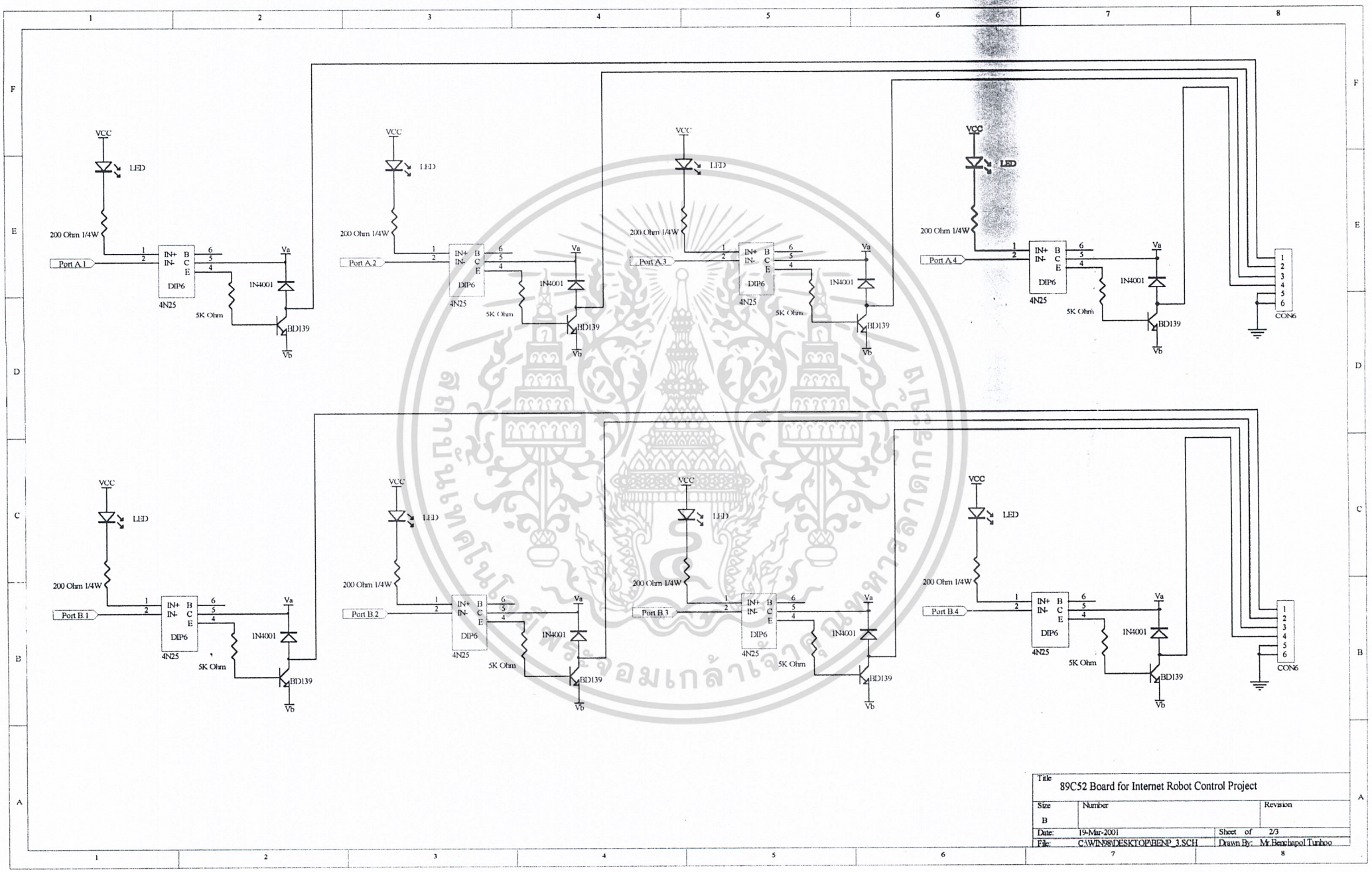
ในวงจรขับจะประกอบด้วย จะประกอบด้วยไอซีออปโตคัปเปิล (Opto Couple) เบอร์ 4N25 ทำหน้าที่เป็นตัวกั้นทางแสง โดยมี LED บอกสถานะต่ออยู่ ถ้าสัญญาณที่ออกจาก 74LS244 เป็น 5 โวลต์ (High level) LED จะดับ แต่ถ้าเป็น 0 โวลต์ (Low level) LED จะสว่าง ที่เอาท์พุทของ 4N25 จะไม่ต่อกับขาเกตของ IRF510



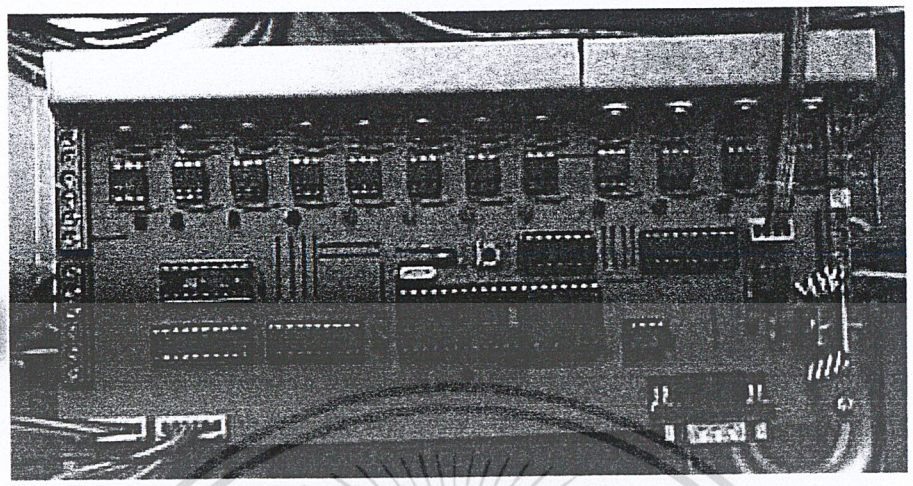
รูปที่ 3.7 (ก) แสดงวงจรควบคุมสเต็ปปิ้งมอเตอร์ และ มอเตอร์กระแสตรงด้วย AT89C52



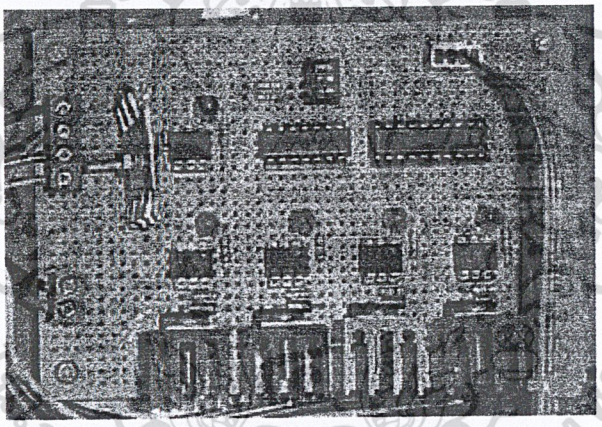
รูปที่ 3.7 (ข) แสดงวงจรควบคุมสเต็ปปิ้งมอเตอร์ และ มอเตอร์กระแสตรงด้วย AT89C52



รูปที่ 3.7 (ค) แสดงวงจรควบคุมสเต็ปปิ้งมอเตอร์ และ มอเตอร์กระแสตรงด้วย AT89C52



(ก)

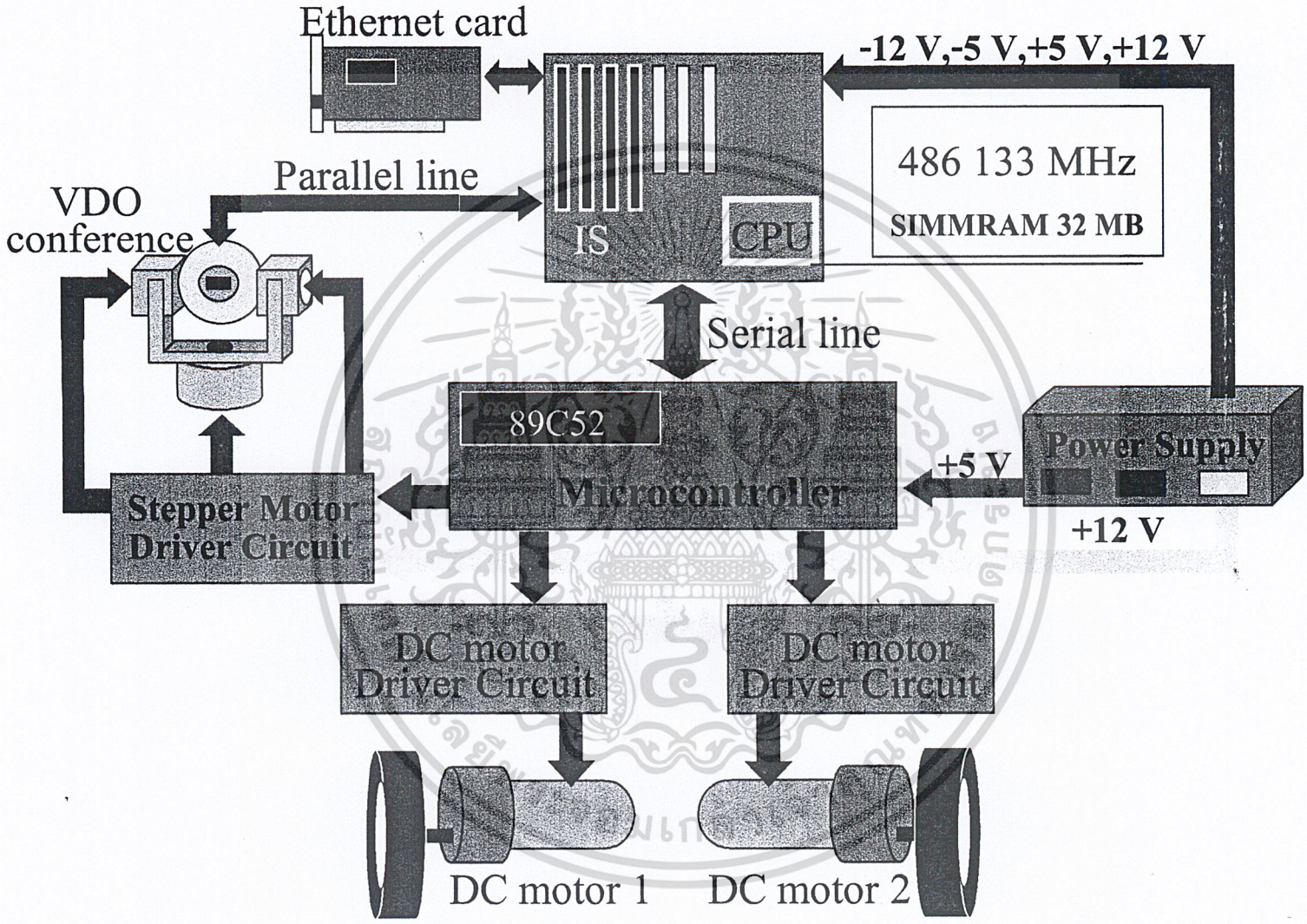


(ข)

รูปที่ 3.8 แสดงภาพถ่ายของวงจรควบคุมสเต็ปปิ้งมอเตอร์และมอเตอร์กระแสตรงด้วย AT89C52

(ก) ส่วนของบอร์ดหลักควบคุม สเต็ปปิ้งมอเตอร์จำนวน 4 ตัว และ มอเตอร์กระแสตรง 1 ตัว

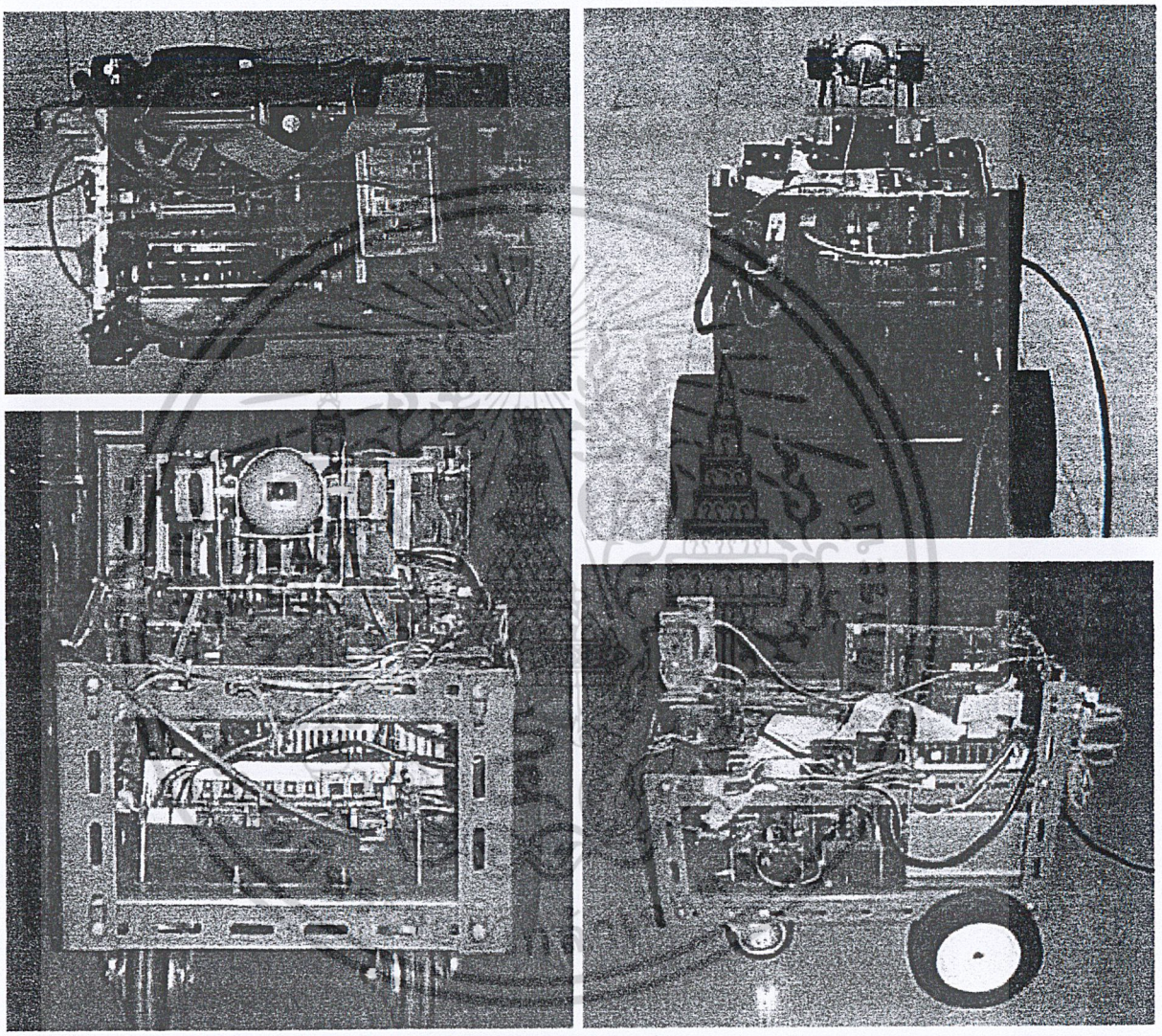
(ข) ส่วนเพิ่มเติม สำหรับขับมอเตอร์กระแสตรง 1 ตัว



รูปที่ 3.9 แสดงแผนผังการเชื่อมต่อฮาร์ดแวร์ของหุ่นยนต์

3.1.4 การประกอบชิ้นส่วนต่างๆ เป็นหุ่นยนต์

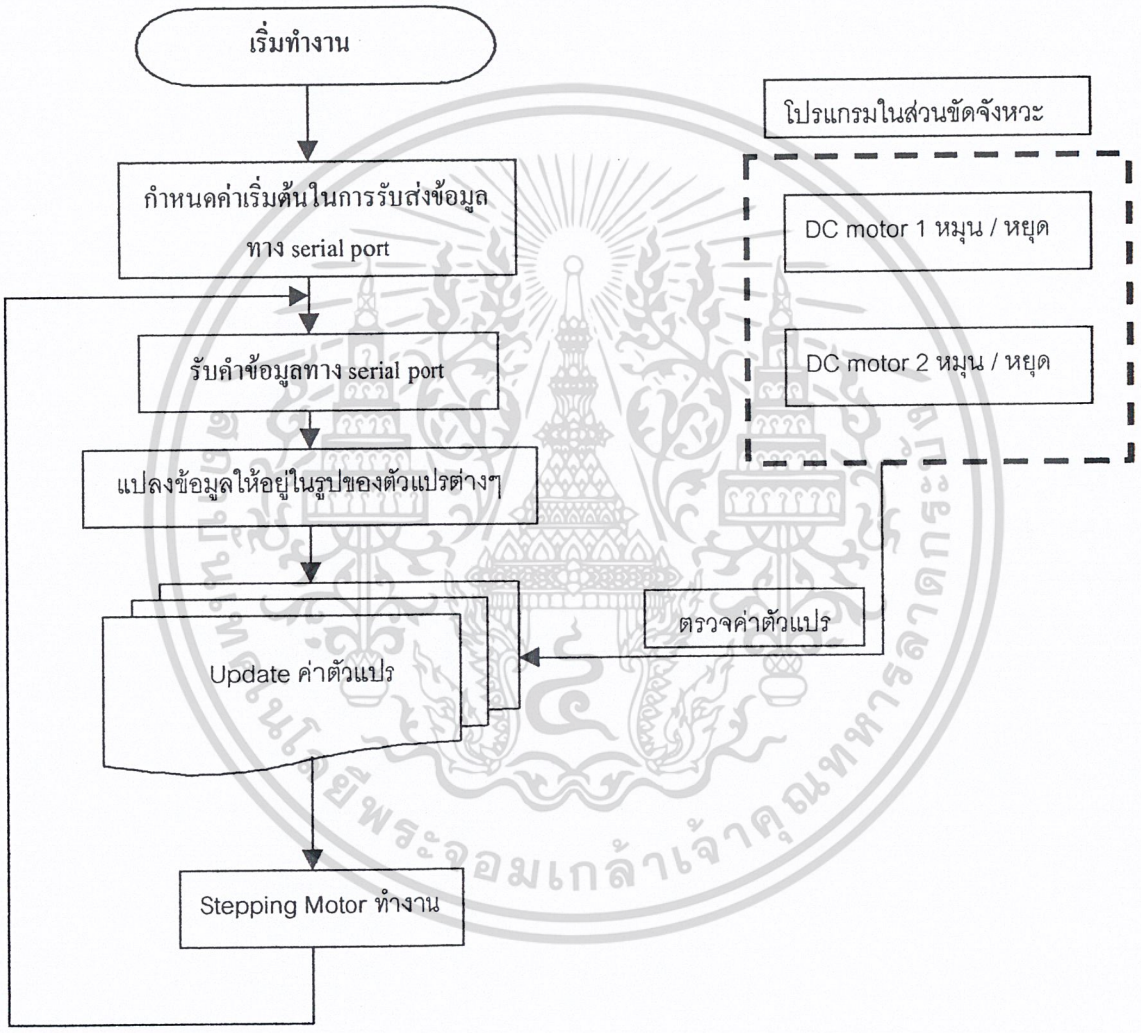
หลังจากที่ได้มีการออกแบบในส่วนต่างๆ เสร็จเรียบร้อยจึงทำการประกอบตัวรถ กล้อง บอร์ดควบคุมมอเตอร์ และคอมพิวเตอร์ เข้าด้วยกันดังรูปที่ 3.10



รูปที่ 3.10 แสดงภาพถ่ายของตัวหุ่นยนต์ที่ประกอบเสร็จเรียบร้อยแล้ว

3.2 การออกแบบในส่วนของซอฟต์แวร์ (Software Designed)

3.2.1 การออกแบบซอฟต์แวร์ในส่วนของวงจรไมโครคอนโทรเลอร์ AT89C52 ทำหน้าที่รับคำสั่งจาก Microprocessor ที่ส่งมาทางพอร์ตอนุกรมในมาตรฐาน RS232 แล้วไปทำการเปิด/ปิด กำหนดทิศทางการหมุน และ ความเร็วของ DC มอเตอร์ 2 ตัว(สำหรับขับเคลื่อน) กับ Stepping มอเตอร์จำนวน 3 ตัว(สำหรับเปลี่ยนตำแหน่งของกล้อง Video conference)



รูปที่ 3.11 แผนภาพแสดงการทำงานของโปรแกรมในส่วนของ Microcontroller

3.2.2 การออกแบบซอฟต์แวร์ในส่วนของคอมพิวเตอร์ Linux Robot Server

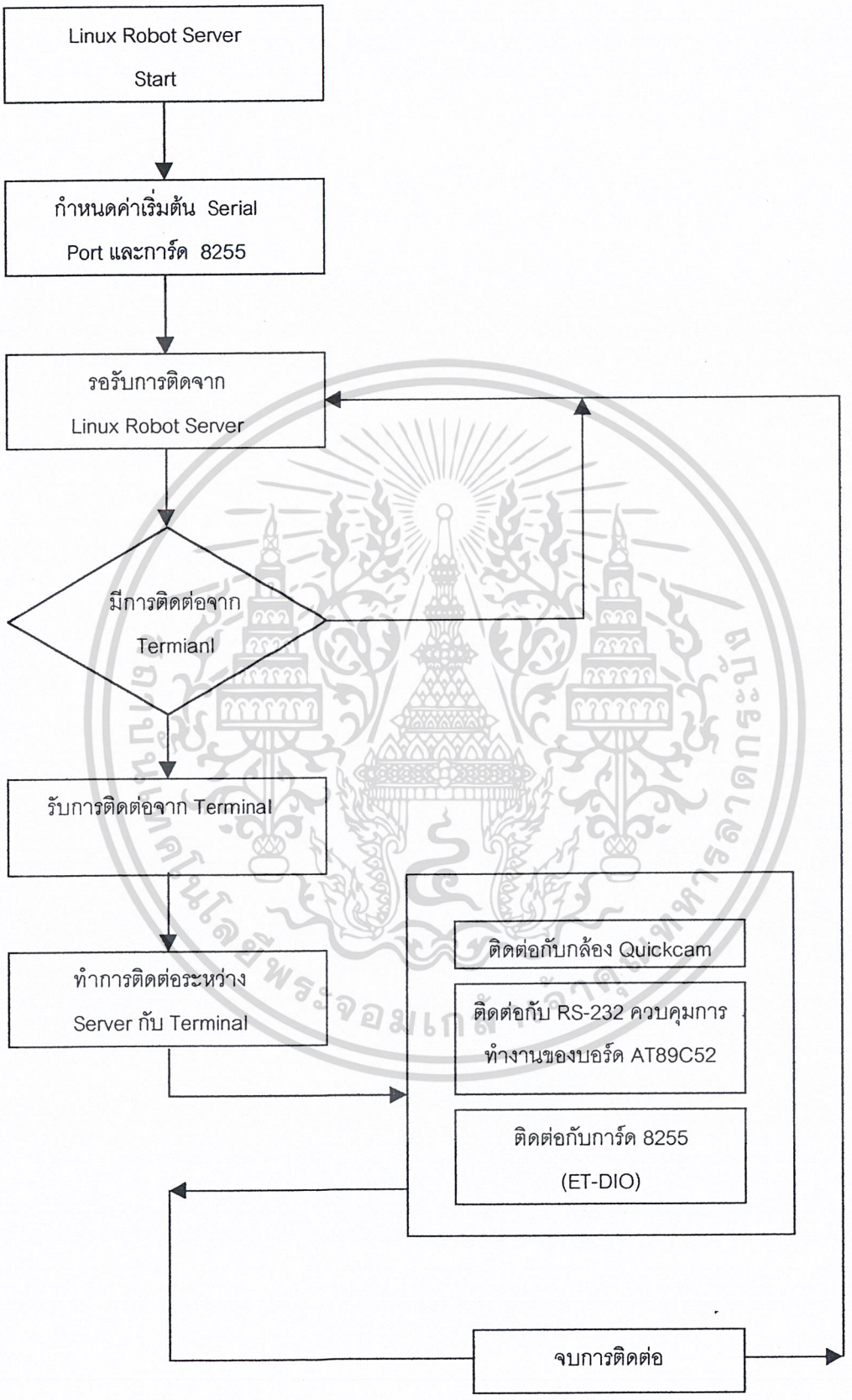
ในส่วนของโปรแกรม Linux Robot Server จะเป็นโปรแกรมลักษณะที่ทำงานตลอดเวลาเพื่อรอให้มีการติดต่อจาก ฝ่าย Terminal ที่เป็นเครื่องปลายทาง แล้วทำการตรวจสอบว่าเป็น user name และ password ที่ได้รับอนุญาติ ให้เข้ามาใช้งาน เมื่อตรวจสอบว่าถูกต้องแล้วจะทำหน้าที่เป็นตัวรับคำสั่งที่ส่งมาจาก Terminal แล้วส่งผ่านทาง RS – 232 ไปยังบอร์ด AT89C52 หรือส่งผ่าน Parallel port ทางการ์ด 8255 โดยภาษาที่ใช้ในการเขียนโปรแกรมจะใช้ ภาษาซี ในการออกแบบโปรแกรม Linux Socket Serverจะประกอบด้วย

1. ส่วนการติดต่อผ่านทาง Internet Socket ซึ่ง Internet Socket คือรูปแบบการติดต่อระหว่างเครื่องคอมพิวเตอร์สองเครื่อง โดยในโปรแกรมนี้จะใช้ Internet Socket เป็นตัวรอรับการติดต่อจาก Terminal โดยใช้ Internet port ที่ 3002 เมื่อมีการติดต่อจาก Terminal จะต้องการตรวจสอบ user name และ password ว่าถูกต้องหรือไม่ แล้วจึงจะมีการอนุญาติ ให้มีการติดต่อ จะกระทั้งเมื่อจบการติดต่อแล้ว Linux Robot Server จะกลับไปคอยรอรับติดต่ออีก โดยในการติดต่อแต่ละครั้งจะมีผู้ติดต่อได้เพียงคนเดียวเท่านั้น

2. ส่วนการติดต่อกับ Serial Port เมื่อมีการติดต่อจาก Terminal เสร็จเรียบร้อยแล้ว ค่าคำสั่งที่ใช้ในการควบคุม Stepping Motor ที่ใช้ในการหมุนกล้อง CCD กับ DC Motor จะต้องส่งผ่านไปที่ บอร์ด AT89C52 ซึ่งจะทำหน้าที่ควบคุมอีกทีหนึ่ง

3. ส่วนการติดต่อกับการ์ด 8255 จะเป็นการติดต่อจะเป็นแบบ Parallel Port ซึ่งจะใช้ในการควบคุมการเปิดปิดไฟที่ด้านหน้ารถ

4. ส่วนการรับภาพจากกล้อง B/W Connectix Quickcam ที่มีการส่งข้อมูลภาพทางพอร์ตเครื่องพิมพ์



รูปที่ 3.12 แผนภาพแสดงการทำงานของ Linux Robot Server

3.2.3 การออกแบบซอฟต์แวร์ในส่วนของคอมพิวเตอร์ Terminal

ในส่วนของ Terminal จะทำหน้าที่เป็นตัวควบคุม server ในระยะไกล

3.2.3.1 MS Windows Terminal

ส่วนการควบคุมการหมุนของกล้อง และควบคุมการเคลื่อนที่ของตัวหุ่นยนต์ผ่านระบบเครือข่าย ในการออกแบบส่วนของ MS Windows Terminal จะใช้โปรแกรม Visual Basic โดยในการควบคุมจะใช้ คีย์บอร์ด หรือ จอยสติก ก็ได้ ในส่วนของการติดต่อ จอยสติก จะอาศัยการเรียกผ่าน Application Program Interface (API) ของตัวระบบปฏิบัติการ MS Windows ดังไดอะแกรมดังรูปที่ 3.13

ในส่วนของการออกแบบโปรแกรม MS Windows 98 Terminal จะแบ่งการทำงานออกเป็น 2 ส่วนการทำงานย่อยดังต่อไปนี้คือ

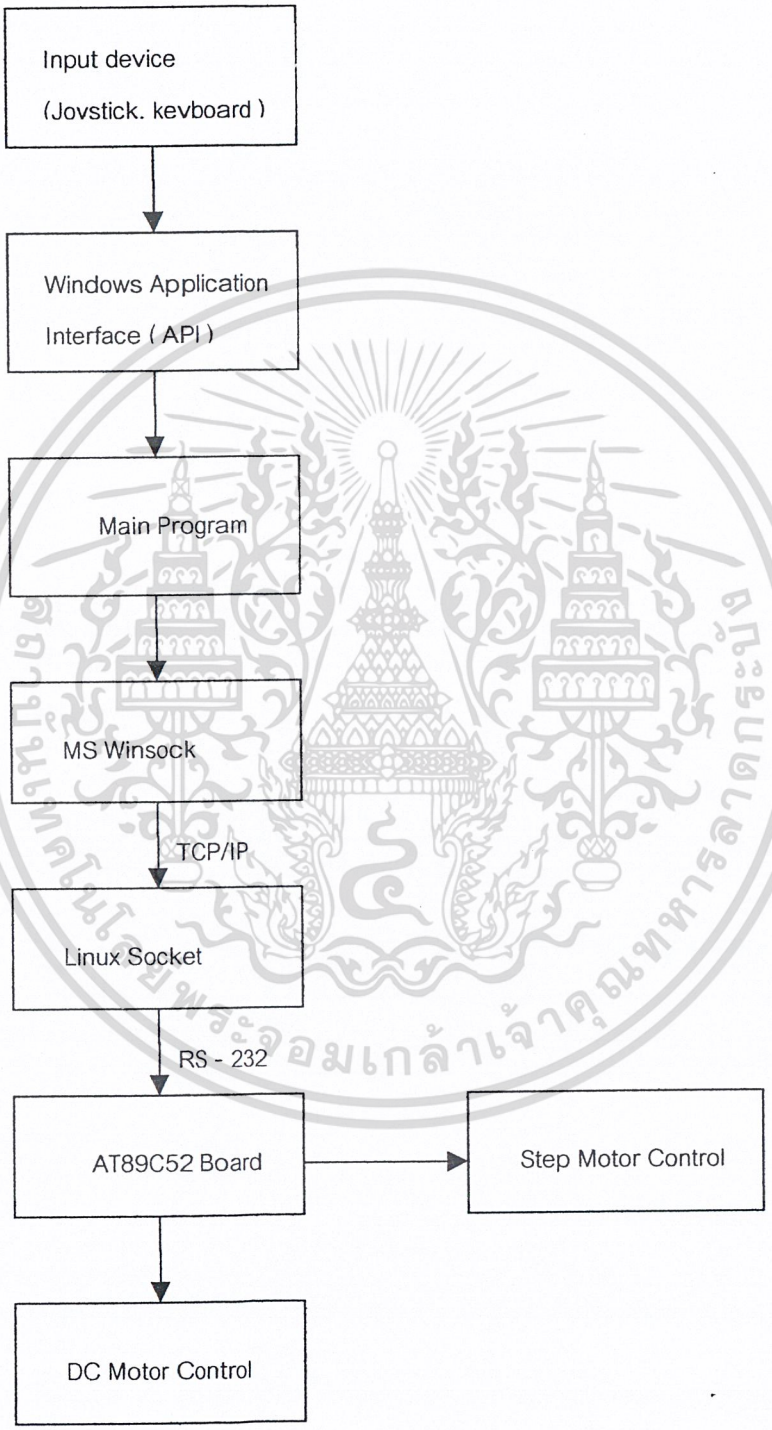
1. สร้าง Form สำหรับการติดต่อกับส่วน Linux Robot Server มีชื่อว่า frmconnect โดยใน Form นี้จะประกอบด้วย

- Input box สำหรับใส่ชื่อ Host ที่ต้องการติดต่อ
- Input box สำหรับใส่ User name ที่ใช้ Login เข้าไปใช้งานในระบบ
- Input box สำหรับใส่ Password ที่ใช้ Login เข้าไปใช้งานในระบบ
- Command Control เป็นปุ่มใช้สำหรับกดเพื่อเริ่มการติดต่อกับ Linux Robot Server มีชื่อว่า cmd_connect และปุ่มสำหรับยกเลิกการติดต่อ

2. สร้าง Form สำหรับเป็นโปรแกรมหลักในการควบคุมมีชื่อว่า frmmain โดยใน Form นี้จะประกอบด้วย

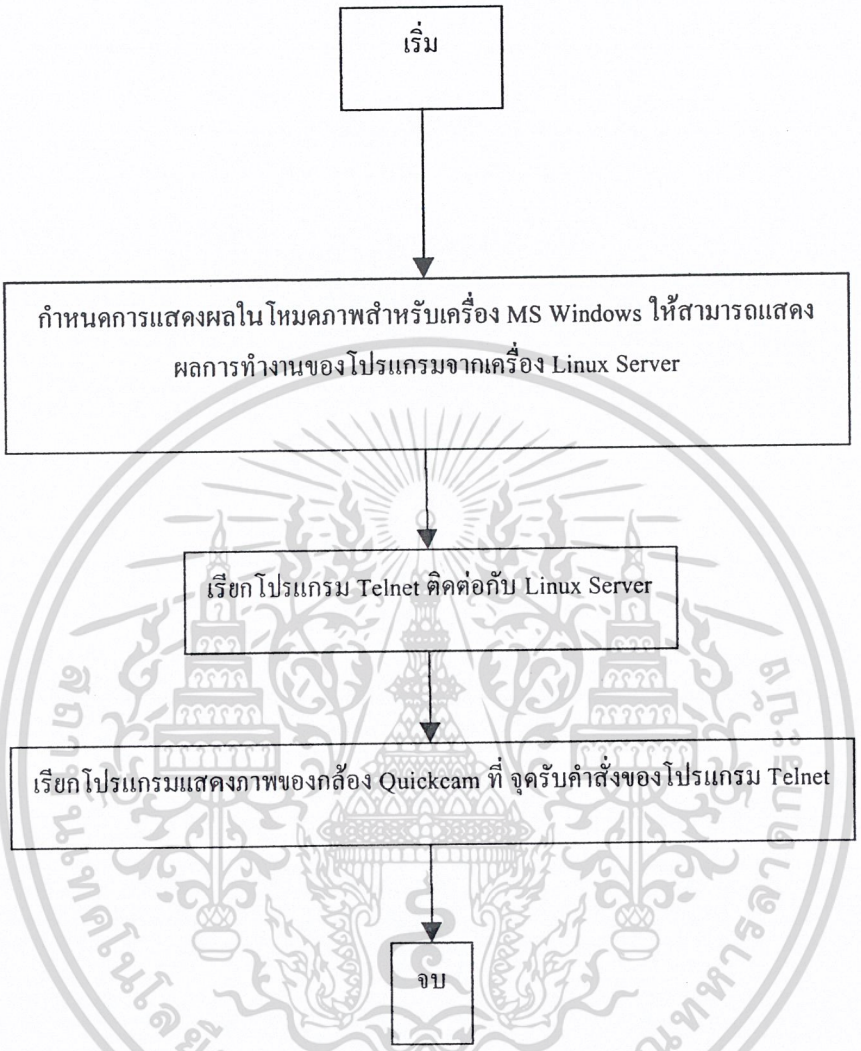
- Frame ที่ชื่อว่า DC_motor_control ที่ภายในประกอบด้วย Command Control ที่ใช้สำหรับการควบคุม DC Motor โดยจะถูก diable เอาไว้จนกระทั้งติดต่อกับ server จึงจะ enable ให้สามารถส่งค่าไปควบคุมที่ server ได้
- Frame ที่ชื่อ ว่าStep_motor_control ที่ภายในประกอบด้วย Command Control ที่ใช้สำหรับการควบคุม Stepping Motor โดยจะถูก diable เอาไว้จนกระทั้งติดต่อกับ server จึงจะ enable ให้สามารถส่งค่าไปควบคุมที่ server ได้
- Command Control มีทั้งหมด 3 ปุ่ม ปุ่มที่หนึ่งชื่อว่า connect_buttom เป็นปุ่มสำหรับให้แสดง Form ในข้อ 1 ออก มาซึ่งจะถูก disable เมื่อมีการติดต่อกับ Linux Robot Server เรียบร้อยแล้ว โดยจะ enable เมื่อจบการติดต่อกับตัว server ปุ่มที่สองชื่อว่า Disconnect_buttom เป็นปุ่มสำหรับยกเลิกการติดต่อกับตัว server โดยในช่วงที่ยังไม่ได้ติดต่อกับตัว server จะถูกกำหนดให้ disable จนเมื่อมีการติดต่อกับ server เรียบร้อยแล้วจึงจะ enable ปุ่มที่สามชื่อว่า Exit เป็นปุ่มสำหรับออกจากโปรแกรม

- Status bar จะเป็นแถบแสดงสถานะคือ ในช่องแรกจะแสดงว่า มี Joystick ได้ต่ออยู่กับเครื่อง Terminal หรือไม่ ช่องที่สองแสดงวันที่ ช่องที่สามแสดงเวลา



รูปที่ 3.13 แผนภาพแสดงการทำงานของโปรแกรม MS Windows terminal

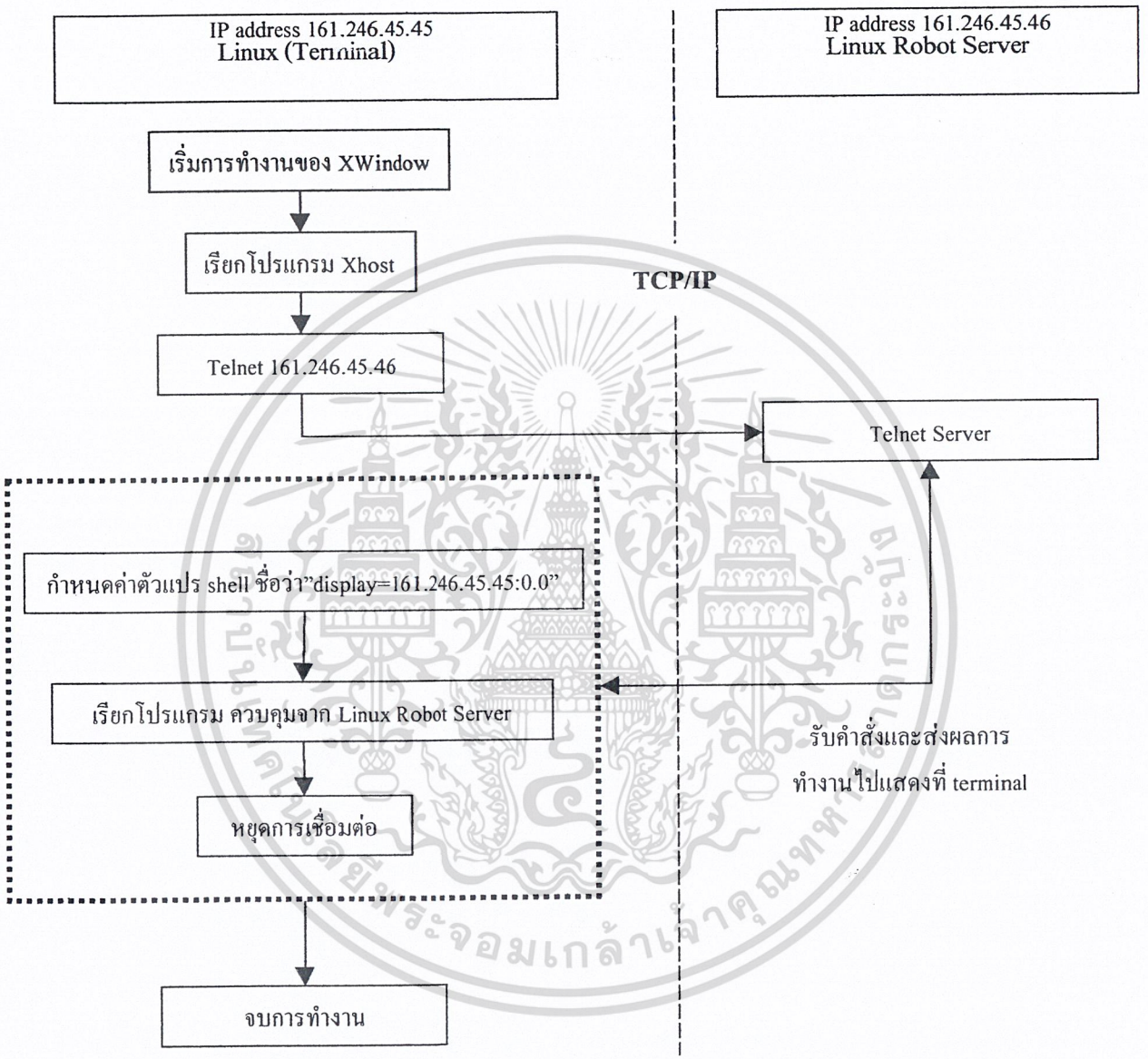
การรับภาพมาแสดงจาก Linux Robot Server



รูปที่ 3.14 แผนภาพแสดงการขั้นตอนการทำงานของ MS Windows Terminal ส่วนการแสดงภาพ

3.2.3.2 Linux Terminal

การติดต่อกับ Linux Robot Server



รูปที่ 3.15 แผนภาพแสดงการทำงานของ Linux Terminal ในการติดต่อกับ Linux Robot Server

ดังรูปที่ 3.15 เป็นการแสดงการทำงานของการติดต่อกันระหว่างเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการ Linux OS ทั้งสองเครื่อง โดยที่เครื่องหนึ่งทำหน้าที่เป็น Server ซึ่งเปิดการให้บริการการทำงาน Telnet อีกเครื่องหนึ่งทำหน้าที่เป็น Terminal ซึ่งจะส่งคำสั่งและรับการแสดงผลของการทำงานของเครื่อง Server (การทำงานของโปรแกรมควบคุมRobot ที่เครื่อง Linux Robot Server จะไปแสดงผลที่เครื่อง Linux Termin

บทที่ 4
ผลการทดลองวิจัย

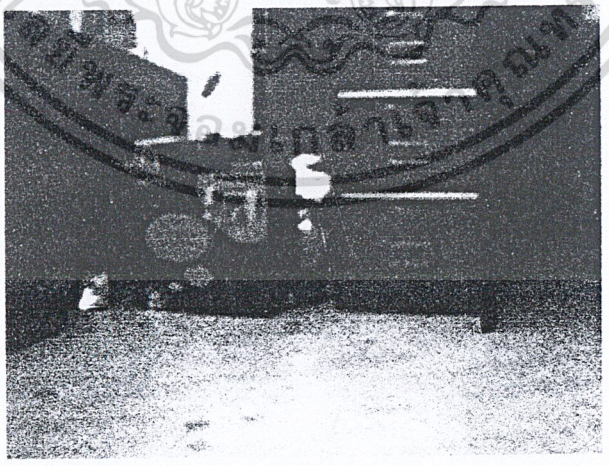
ในการทดลองวิจัยโครงงานนี้เนื่องจากประกอบด้วยส่วนต่างๆ หลายส่วน จึงได้ทำการแบ่งการทดลองวิจัยออกเป็น ส่วนใหญ่ ซึ่งจะได้ตรวจสอบข้อผิดพลาดและแก้ไขได้ง่าย แต่ละส่วนมีผลการทดลองวิจัยดังนี้

4.1 ผลการทดลองในส่วนการรับส่งข้อมูลกับบอร์ดควบคุมมอเตอร์

เราสามารถติดต่อกับบอร์ดควบคุมมอเตอร์ที่ใช้ ไมโครคอนโทรเลอร์เบอร์ AT89C52 ได้ โดยการติดต่อจะยึดมาตรฐาน EIA-232 ในการรับส่งข้อมูล ซึ่งผลการทดลองสามารถทำการหมุนมอเตอร์กระแสตรงทั้ง 2 ตัว ได้ในทิศทาง และ ความเร็วที่กำหนดได้ และ สามารถทำการหมุนสเต็ปปิ้งมอเตอร์ทั้ง 4 ตัวได้ตามจำนวนสเต็ปที่ระบุและในทิศทางที่ต้องการได้

4.2 ผลการทดลองในส่วนการติดต่อกับกล้องซีซีดี B/W Connectix Quickcam

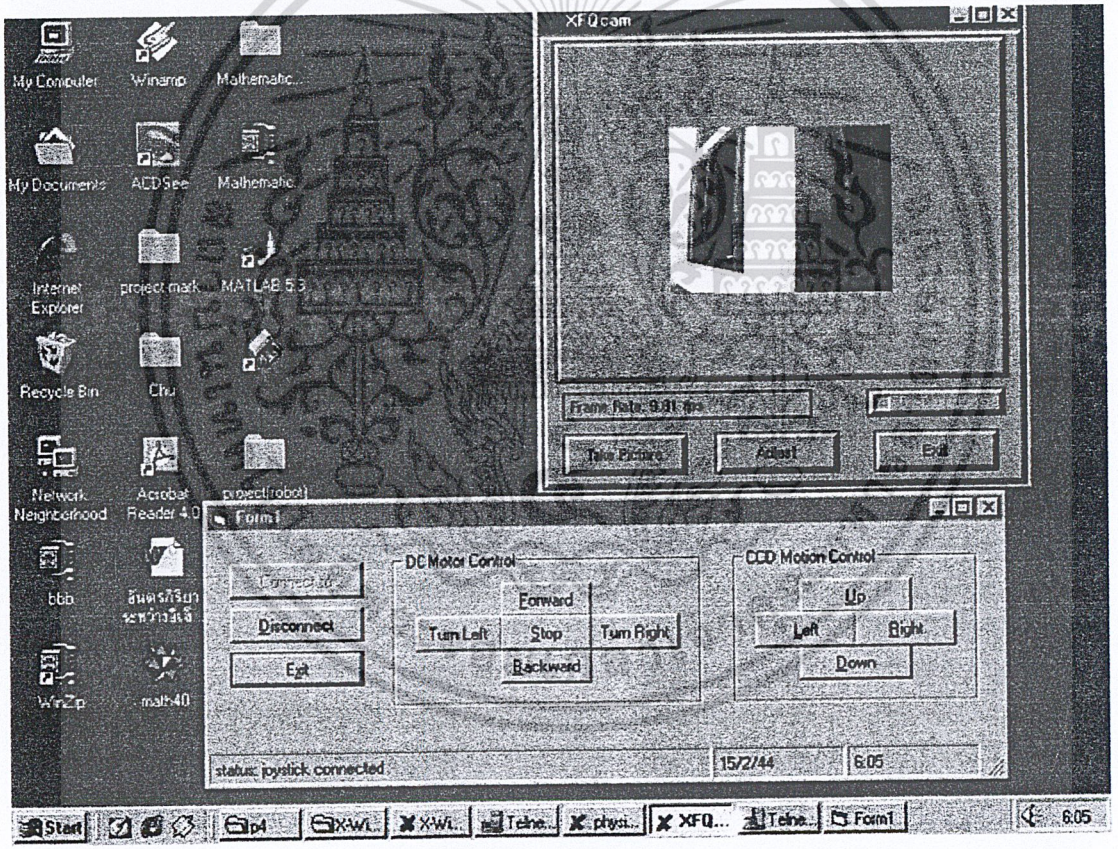
เราสามารถติดต่อกับ กล้องซีซีดี B/W Connectix Quickcam ได้ โดยการติดต่อผ่านทางพอร์ตเครื่องพิมพ์ ลักษณะการติดต่อกับกล้องคือ มีการส่งข้อมูลแบบขนานไปทำการกำหนดค่า Register ของกล้อง ให้ทำงานตามที่ต้องการ คือ กำหนดจำนวนพิกเซลของความกว้างและยาวเพื่อการสแกนภาพ กำหนดความลึกของสี โดยที่รูปแบบของภาพที่ได้จะอยู่ในแบบ PGM ดังรูปที่ 4.1



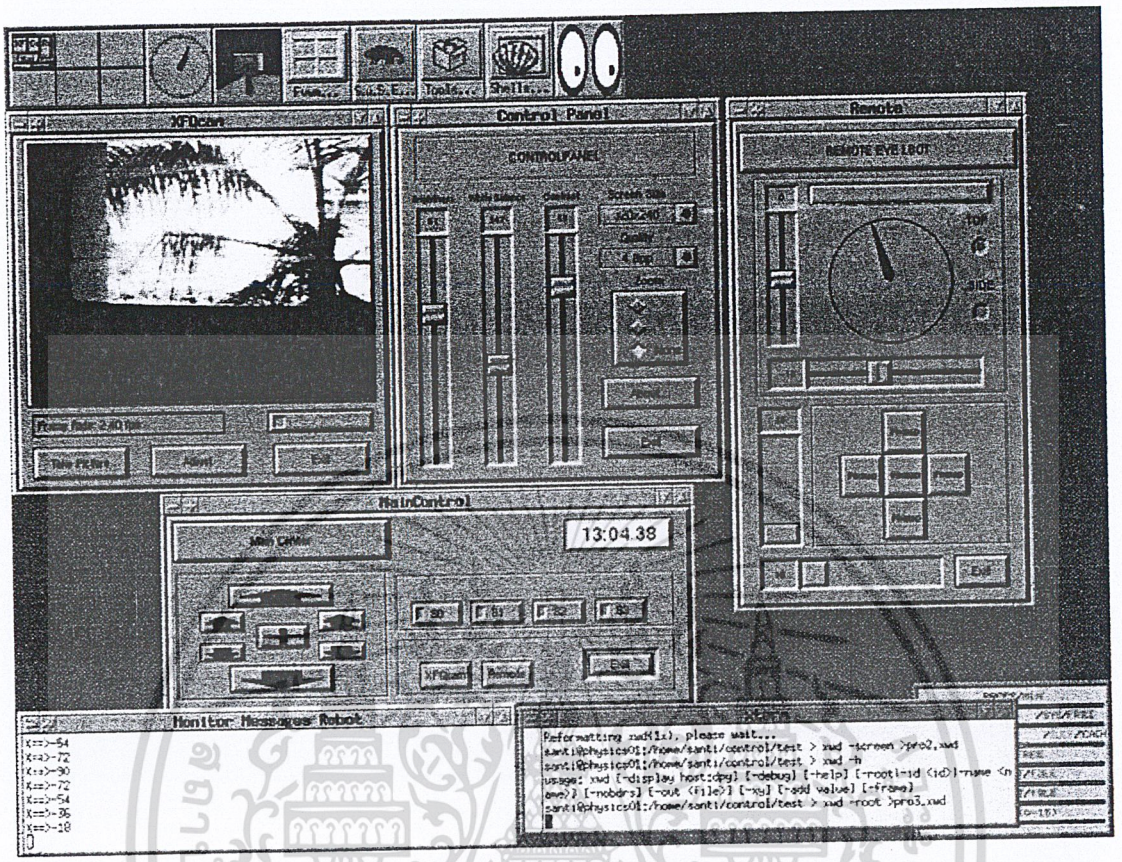
รูปที่ 4.1 แสดงตัวอย่างของข้อมูลภาพที่ได้จากกล้อง ซีซีดี B/W Connectix Quickcam

4.3 ผลการทดลองในส่วนการติดต่อสื่อสารผ่านอินเตอร์เน็ต

เราสามารถติดต่อกับ Linux Robot Server ได้ โดยการติดต่อสื่อสารผ่านเครือข่ายอินเตอร์เน็ต ที่มีการใช้โปรโตคอลชนิด TCP/IP เครื่อง Linux Robot Server ได้มีการกำหนด IP Address เป็น 161.246.45.46 และ กำหนดการช่องการทำงานของ Socket Port 3002 ส่วนเครื่องคอมพิวเตอร์ที่ทำการควบคุม ได้มีการทดลองทั้งเครื่องที่มีระบบปฏิบัติการ MS Windows98 มี IP Address เป็น 161.246.13.60 และ เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Linux มี IP Address เป็น 161.246.45.45 ในการติดต่อกันระหว่างเครื่อง Linux Robot Server กับ เครื่องที่เป็น Terminal นั้น มีหน้าตาของโปรแกรมดังรูปที่ 4.2 อันประกอบด้วน ส่วนรับภาพ ส่วนควบคุมกล้อง และส่วนควบคุมการเคลื่อนที่ของหุ่นยนต์



(ก)



(ข)

รูปที่ 4.2 แสดงหน้าของโปรแกรมที่ควบคุมการทำงานของ Linux Robot Server

(ก) ควบคุมด้วยคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการ MS Windows98

(ข) ควบคุมด้วยคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการ Linux

4.4 ผลการทดลองในส่วนการควบคุมหุ่นยนต์ให้เคลื่อนที่ผ่านเครือข่ายอินเตอร์เน็ต

เราสามารถควบคุมการเคลื่อนที่ของหุ่นยนต์ให้ไปได้ 6 ลักษณะคือ

1. ตรงไปข้างหน้า โดยการสั่งให้มอเตอร์กระแสตรงทั้งสองตัว ซ้าย(M1) และ ขวา (M2) หมุนไปข้างหน้าพร้อมกันดังรูปที่ 4.3 (ก)

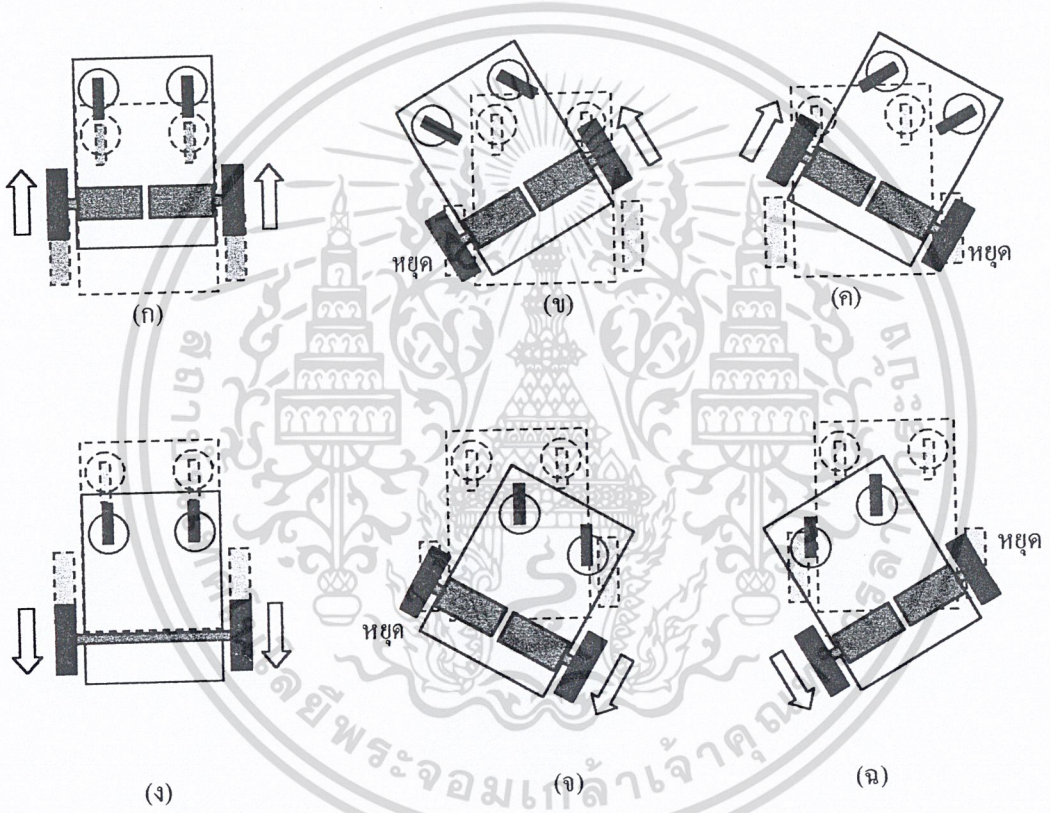
2. เลี้ยวซ้าย โดยการสั่งให้มอเตอร์กระแสตรง ซ้าย(M1) หยุดหมุน และ ขวา(M2) หมุนไปข้างหน้า ดังรูปที่ 4.3 (ข)

3. เลี้ยวขวา โดยการสั่งให้มอเตอร์กระแสตรง ซ้าย(M1)หมุนไปข้างหน้า และ ขวา (M2) หยุดหมุน ดังรูปที่ 4.3 (ค)

4. ถอยหลัง โดยการสั่งให้มอเตอร์กระแสตรงทั้งสองตัว ซ้าย(M1) และ ขวา(M2) หมุนไปข้างหลังพร้อมกันดังรูปที่ 4.3 (ง)

5. ถอยหลังไปทางซ้าย โดยการสั่งให้มอเตอร์กระแสตรง ซ้าย(M1)หยุดหมุน และ ขวา(M2) หมุนไปข้างหลัง ดังรูปที่ 4.3 (จ)

6. ถอยหลังไปทางขวา โดยการสั่งให้มอเตอร์กระแสตรง ซ้าย(M1) หมุนไปข้างหลัง และ ขวา(M2) หยุดหมุน ดังรูปที่ 4.3 (ฉ)



รูปที่ 4.3 แสดงทิศการหมุนของมอเตอร์กระแสตรงในลักษณะที่ทำให้เคลื่อนที่ในแนวต่างๆ

ก) ตรงไปข้างหน้า

ข) เลี้ยวซ้าย

ค) เลี้ยวขวา

ง) ถอยหลัง

จ) ถอยหลังไปทางซ้าย

ฉ) ถอยหลังไปทางขวา

บทที่ 5
สรุปผลการทดลองวิจัยและแนวทางการพัฒนา

โครงงานพิเศษนี้สามารถควบคุมหุ่นยนต์ให้สามารถเคลื่อนได้ตามที่ต้องการ โดยสามารถควบคุมได้ทั้งการเดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา และสามารถบังคับ กล้อง CCD ที่ทำหน้าที่ส่งภาพในขณะที่บังคับรถ ให้สามารถเคลื่อนหมุนซ้าย ขวา หรือก้มเงยได้ตามที่ต้องการ โดยในการการควบคุมสามารถควบคุมได้จากเครื่องลูก (client) ที่ใช้ระบบปฏิบัติการต่างๆไม่ว่าจะเป็น ระบบปฏิบัติการ MS Windows หรือระบบปฏิบัติการ Unix

ข้อเสนอแนะและแนวทางการพัฒนา

1 เนื่องจากระบบเครือข่ายอินเตอร์เน็ตเป็นระบบเปิด ดังนั้นในการควบคุมหุ่นยนต์ จะไม่สามารถนำไปใช้ในการที่ต้องการความเร็วสูงได้ เนื่องจากอาจมีการล่าช้า (delay) ที่มาจากระบบเครือข่ายอินเตอร์เน็ต

2 หุ่นยนต์ที่สร้างขึ้น สามารถนำไปใช้ใน การประยุกต์ใช้งานในด้านตรวจจับระยะไกล (Remote Sensing) ในที่ ที่มนุษย์ ไม่สามารถเข้าไปได้เช่นในที่ ที่มีสารพิษ หรือ ก๊าซพิษ

3 ในการปรับปรุงหุ่นยนต์ที่สร้างขึ้นอาจทำให้เป็นระบบที่ไร้สาย (Wireless) โดยต้องมีการปรับปรุงในด้านพลังงานที่ให้กับหุ่นยนต์ และระบบการติดต่อระหว่างหุ่นยนต์กับระบบเครือข่ายที่ต้องมีความเร็วสูงที่พอจะสามารถบังคับหุ่นยนต์ได้

4 ในการปรับปรุงในการเคลื่อนที่ของหุ่นยนต์ให้มีความเร็วสูงขึ้นแต่ต้องคำนึงถึงปัจจัยทางด้านระบบเครือข่ายด้วย

ภาคผนวก

ภาคผนวก ก.

โปรแกรมควบคุมมอเตอร์ด้วย AT89C52

```
/*
       Linux Robot AT89C52
  By Santichai Poungkaew
   Create 10/17/2000 00.30 AM.
   Modified 1/11/2001 01.32 PM.
*/
#include "c:\mc\8051io.h"
#include "c:\mc\8051reg.h"
#include "c:\mc\8051bit.h"

extern register char cputick;
extern register int PWM1,PWM2;
/* ---- variable --*/
register int xdest,xuser;
register int ydest,yuser;
register int odest,ouser;
register int pdest,puser;
register int zdest,zuser;
register int adest,auser;
register int bdest,buser;
register int value,negative;
register char dest;

register int od1,od2;

unsigned register char OUTP0,OUTP2;
register char delay1,delay2,delay3,delay4; // delay for step1,2,3,4
unsigned register char c_delay1,c_delay2,c_delay3,c_delay4;
unsigned register char bit_con,bit_stop; //bit controll 4 step motor
register char count1,count2,count3,count4;
register int c1,c2,c3,c4,o1,o2,o3,o4;

/* ---- Table -- */
unsigned steptab1[]={0x01,0x02,0x04,0x08};
unsigned steptab2[]={0x10,0x20,0x40,0x80};

unsigned int power[] = {0x0000,0x0001,0x0101,0x0841,0x1111,0x1249,0x2949
,0x2A93,
0x5555,0xD555,0xD5D5,0xEDB6,0xEEEE,0xF7BE,0xFEFE,0xFFFE,0xFFFF};

/*                step1     step2     step3     step4
 dir   bit_con    0x10      0x20      0x40      0x80
  en   bit_stop   0x01      0x02      0x04      0x08
*/

main()
{
 serinit(9600);
 init_value();
// printf("hello lbot! >>");
 dc1(0);
 dc2(0);
 while(1)
 {
  while(cputick<1)
  ;
  cputick = 0;
  // -----------------
  control();

  delay_count();
  step_run();
```

```
  //--------------------
 }
}

init_value()
{
 dest=' ';
 value=0;
 negative=0;
 xdest=0;
 ydest=0;
 odest=0;
 pdest=0;
 zdest=0;
 adest=0;
 bdest=0;
 xuser=0;
 yuser=0;
 ouser=0;
 puser=0;
 zuser=0;
 auser=0;
 buser=0;
 c1=0;c2=0;c3=0;c4=0;
 count1=4;
 count2=4;
 count3=4;
 count4=4;
 OUTP0=0x11;
 OUTP2=~0x11;
 delay1 = 2;  // in unit 10 msec
 delay2 = 2;
 delay3 = 5;
 delay4 = 5;
 bit_stop = 0x00;
 od1=1;od2=1;

}

int get_one_char(char c)
{
 // if((c>'a') && (c<'z')) {c = c- 'a'+'A';}
 switch(c) {

  case 'A':  adest=1;auser=1;   break;
  case 'B':  bdest=1;buser=1;   break;
  case 'C':  adest=-1;auser=-1;  break;
  case 'D':  bdest=-1;buser=-1;  break;
  case 'M':  adest=0; auser=0; break;
  case 'N':  bdest=0; buser=0; break;

  case 'X':  adest=0; bdest=0; auser=0;buser=0;break;
  case 'F':  adest=1; bdest=1; auser=1;buser=1;break;
  case 'R':  adest=-1;bdest=-1; auser=-1;buser=-1; break;

  case 'x':
  case 'y':
  case 'o':
  case 'p':
  case 'z':
  case 'a':
  case 'b':
    dest=c;
```

```
 break;
case '=':
 negative = 0;
 break;
case '-':
 negative = 1;
 break;
case '+':
 negative = 0;
case '0':
case '1':
case '2':
case '3':
case '4':
case '5':
case '6':
case '7':
case '8':
case '9':
  value *=10;
  value +=(c-'0');
  break;

case '?':
case '\r':
case '\n':
case ',':

   if(negative) {
    value = -value;
   }
   if(dest=='x') {
     xuser=value;
   } if(dest == 'y') {
     yuser=value;
   } if(dest == 'o') {
     ouser=value;
   } if(dest == 'p') {
     puser=value;
   } if(dest == 'z') {
     zuser=value;
   } if(dest == 'a') {
     auser=value;
   } if(dest == 'b') {
     buser=value;
   }
   dest = ' ';
   value = 0;
   negative = 0;

   if( ( c=='\n')||( c=='\r') ){
      xdest = xuser;
      ydest = yuser;
      odest = ouser;
      pdest = puser;
      zdest = zuser;
      adest = auser;
      bdest = buser;
      xuser=0;yuser=0;ouser=0;puser=0;zuser=0;
      //auser=0;buser=0;
      return 1;
     }
     break;
```

```
    }
    return 0;
}



delay_count()
{
 c_delay1++;
 c_delay2++;
 c_delay3++;
 c_delay4++;
 if(c_delay1 >= delay1) { c_delay1 = 0; bit_con |= 0x01;}
 if(c_delay2 >= delay2) { c_delay2 = 0; bit_con |= 0x02;}
 if(c_delay3 >= delay3) { c_delay3 = 0; bit_con |= 0x04;}
 if(c_delay4 >= delay4) { c_delay4 = 0; bit_con |= 0x08;}
}

step_run()
{
 //step 1
  if( ((bit_con & 0x01) != 0) && ((bit_stop & 0x01)!=0) )
  {
  bit_con &= ~0x01;
   if((bit_con & 0x10) != 0)
   { count1++; }
   else {count1--;}

    OUTP2 |= 0x0f;
    OUTP2 &= ~steptab1[count1%4];
    c1++;
    if( c1 >= o1 ) { c1=0;bit_stop &= ~0x01;putch('i');}
   }
  //---------

  //step 2
  if( ((bit_con & 0x02) != 0) && ((bit_stop & 0x02)!=0) )
  {
  bit_con &= ~0x02;
   if((bit_con & 0x20) != 0)

   { count2++;}
   else { count2--;}

    OUTP2 |= 0xf0;
    OUTP2 &= ~steptab2[count2%4];
    c2++;
    if( c2 >= o2 ) { c2=0;bit_stop &= ~0x02;putch('j');}
   }
  //---------

 //step 3
  if( ((bit_con & 0x04) != 0) && ((bit_stop & 0x04)!=0) )
  {
  bit_con &= ~0x04;
   if((bit_con & 0x40) != 0)
   { count3++;}
   else { count3--;}

    OUTP0 &= 0xf0;
    OUTP0 |= steptab1[count3%4];
    c3++;
```

```
   if( c3 >= o3 ) { c3=0;bit_stop &= ~0x04;putch('*');}
  }
  //---------

  //step 4
  if( ((bit_con & 0x08) != 0) && ((bit_stop & 0x08)!=0) )
  {
  bit_con &= ~0x08;
   if((bit_con & 0x80) != 0)
   { count4++;}
   else { count4--;}
    OUTP0 &= 0x0f;
    OUTP0 |= steptab2[count4%4];
    c4++;
   if( c4 >= o4 ) { c4=0;bit_stop &= ~0x08;putch('*');}
  }
  //---------

 P0 = OUTP0;  // load value to P0
 P2 = OUTP2;  // load value to P2
}


step1(int a)
{
  if(a)
  {
  if (a < 0)
  {  bit_con |= 0x10; }
  else
  {  bit_con &=~0x10; }
  bit_stop |= 0x01;
  o1 = abs(a);
  }
}
step2(int a)
{
  if(a)
  {
  if (a < 0)
  {  bit_con |= 0x20; }
  else
  {  bit_con &=~0x20; }
  bit_stop |= 0x02;
  o2 = abs(a);
  }
}
step3(int a)
{
  if(a)
  {
  if (a < 0)
  {  bit_con |= 0x40; }
  else
  {  bit_con &=~0x40; }
  bit_stop |= 0x04;
  o3 = abs(a);
  }
}


step4(int a)
{
```

```
  if(a)
  {
  if (a < 0)
  {  bit_con |= 0x80; }
  else
  {  bit_con &=~0x80; }
  bit_stop |= 0x08;
  o4 = abs(a);
  }
}

motor1(int s)
{
  if(s<0)
   setbit(P1.0)
  else clrbit(P1.0)

  clrbit(IE.5)
  PWM1=power[16-abs(s)];
  setbit(IE.5)

}

motor2(int s)
{
 if(s<0)
   setbit(P1.2)
 else clrbit(P1.2)

  clrbit(IE.5)
  PWM2=power[16-abs(s)];
  setbit(IE.5)

}

dc1(int d)
{
 if(d!=od1)
 { motor1(d);od1=d;}
}

dc2(int d)
{
  if(d!=od2)
  { motor2(d);od2=d;}
}

control()
{
 char ch;
 if(ch=chkch())
 {

   dc_(adest);
   dc2(bdest);

  if(get_one_char(ch))
  {
   step1(xdest);step2(ydest);
   if(zdest)
   {
    step3(zdest);step4(-zdest);
   }
```

```
   else
   { step3(odest);step4(pdest);}

//   printf("xdest=%d,ydest=%d,odest=%d,pdest=%d,zdest=%d,adest=%d,bdest
=%d\n",
//                  xdest,ydest,odest,pdest,zdest,adest,bdest);

   xdest=0;
   ydest=0;
   odest=0;
   pdest=0;
   zdest=0;
  // adest=0;
  // bdest=0;
  }
 }
}
```

ภาคผนวก ข.

โปรแกรม MS – Windows Terminal

คู่มือการใช้งานโปรแกรม MS – Windows Terminal

โปรแกรมที่ออกแบบสำหรับควบคุมหุ่นยนต์แบ่งได้ออกเป็น

1. ส่วนติดต่อกับ Linux Robot Server
2. ส่วนที่ใช้ควบคุมการเคลื่อนที่

สิ่งที่จำเป็นสำหรับการติดตั้งโปรแกรม

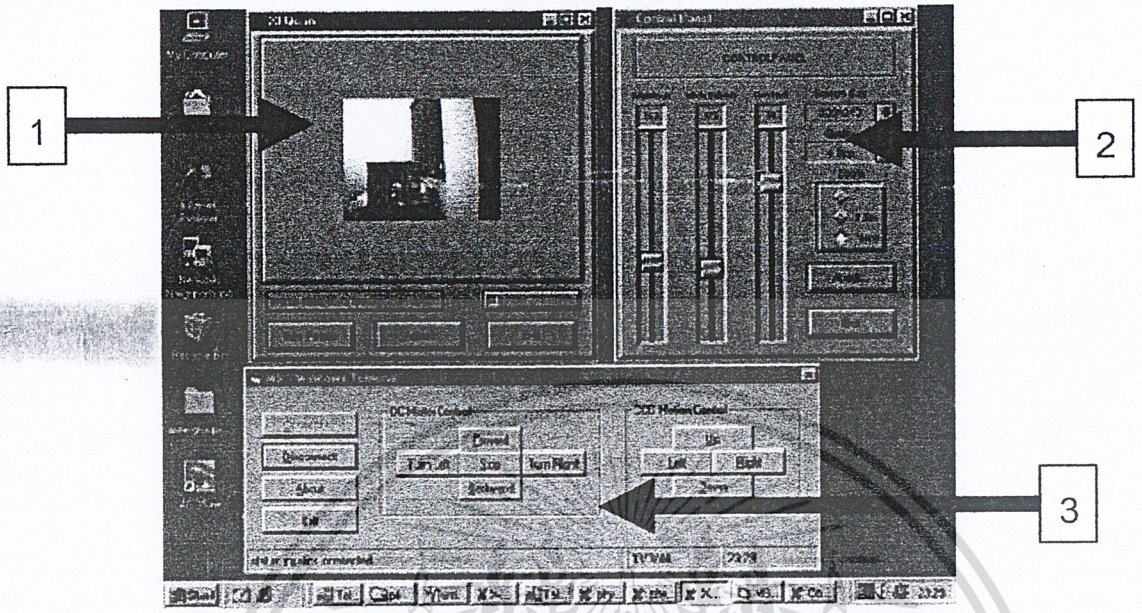
1. คอมพิวเตอร์ 80486 Dx 100 หรือสูงกว่า
2. ระบบปฏิบัติการ Windows 95 / 98 / 2000 / Me
3. หน่วยความจำ (RAM) 16 เมกะไบต์
4. ฮาร์ดดิสก์ 15 เมกะไบต์
5. จอยสติก สำหรับในกรณีที่ต้องการควบคุมด้วย จอยสติก

การติดตั้งโปรแกรม

1. เข้าสู่ระบบปฏิบัติการ Windows
2. ใส่แผ่น setup ลงในไดร์ฟ 3.5 นิ้ว
3. กดปุ่ม Start เลื่อนเมาท์มาที่ Run แล้วกด Enter
4. พิมพ์ A:\Setup กดปุ่ม OK

หลังจากนั้นโปรแกรมจะทำการติดตั้งอย่างอัตโนมัติ

การใช้งานโปรแกรม MS window Terminal ในการควบคุมหุ่นยนต์



จากรูปด้านบนจะเห็นว่าตัวโปรแกรมจะประกอบไปด้วย 3 หน้าต่างคือ

1. หน้าต่างแสดงภาพ เป็นส่วนแสดงภาพถ่ายจากตัวกล้อง จะแสดงความเร็วในการแสดงภาพได้ในหน่วย frame / sec
2. หน้าต่างควบคุมเกี่ยวกับภาพ เป็นส่วนควบคุมภาพที่ได้จากตัวกล้อง คือ ความสว่าง ความคมชัด ความสมดุลย์ทางสีขาว ขนาดภาพ และความละเอียดของสีภาพ
3. หน้าต่างควบคุมการเคลื่อนที่ของหุ่นยนต์ เป็นส่วนหลักในการควบคุมการเคลื่อนที่ของตัวหุ่นยนต์ และการเคลื่อนไหวของตัวกล้อง

```
'-----------------------------------------
' MS – Windows Terminal
' main_frm.frm
'-----------------------------------------
Option Explicit
Dim ji As JOYINFOEX     ' joystick state buffer
Dim caps As JOYCAPS     ' joystick capabilities
Dim flag_xjoy As Boolean
Dim flag_yjoy As Boolean
Dim flag_butt As Boolean
Dim xaxis_num As Integer
Dim yaxis_num As Integer
Dim rc As Long          ' return code
Dim i As Long           ' index

Private Sub About_Click()

main.Enabled = False
about_p.Show
End Sub

Private Sub backward_Click()
Dim data As String
data = "b" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub connect_but_Click()
connect.login.Text = ""
connect.password.Text = ""
flag_xjoy = False
flag_yjoy = False
flag_butt = False
connect.Show
main.Enabled = False
```

```
End Sub

Private Sub disconnect_but_Click()
Winsock1.Close
Timer1.Enabled = False
connect_but.Enabled = True
Frame1.Enabled = False
Frame2.Enabled = False
connect.host = ""
connect.login = ""
connect.password = ""
disconnect_but.Enabled = False
End Sub

Private Sub down_Click()
Dim data As String
data = "d" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub exit_Click()
Winsock1.Close
Unload main
End Sub

Private Sub Form_Load()
disconnect_but.Enabled = False

' Start with timer disabled
  Timer1.Enabled = False
  ' Get capabilities of joystick1
  rc = joyGetDevCaps(JOYSTICKID1, caps, Len(caps))
  If (rc <> 0) Then
     'MsgBox "Couldn't detect the joystick"
      StatusBar1.Panels(1).Text = "Couldn't detect the joystick"
```

```
  End If
'----------------
  ji.dwSize = Len(ji)
  ji.dwFlags = JOY_RETURNALL

  ' Get the current joystick data
  rc = joyGetPosEx(JOYSTICKID1, ji)

  ' Display the status
  If (rc = 0) Then
    'Status.Caption = "status: joystick connected"
    StatusBar1.Panels(1).Text = "status: joystick connected"
  Else
    If (rc = JOYERR_UNPLUGGED) Then
      'Status.Caption = "status: joystick unplugged"
     StatusBar1.Panels(1).Text = "status: joystick unplugged"
    Else
      'Status.Caption = "status: joyGetPosEx error, rc = "& rc
   StatusBar1.Panels(1).Text = "status: joyGetPosEx Error"
    End If
  End If

'----------------
Frame1.Enabled = False
Frame2.Enabled = False


End Sub

Private Sub Form_Unload(Cancel As Integer)
MsgBox "ต้องการออกจากโปรแกรม", vbOKCancel, "Exit"
End Sub

Private Sub forward_Click()
Dim data As String
```

```
data = "f" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub left_Click()
Dim data As String
data = "l" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub right_Click()
Dim data As String
data = "r" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub stop_Click()
Dim data As String
data = "s" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub Timer1_Timer()
  'Dim i As Long
  ' Initialize struct
  ji.dwSize = Len(ji)
  ji.dwFlags = JOY_RETURNALL

  ' Get the current joystick data
  rc = joyGetPosEx(JOYSTICKID1, ji)

  ' Display the status
  If (rc = 0) Then
    'Status.Caption = "status: joystick connected"
    StatusBar1.Panels(1).Text = "status: joystick connected"
```

```
Else
  If (rc = JOYERR_UNPLUGGED) Then
    'Status.Caption = "status: joystick unplugged"
   StatusBar1.Panels(1).Text = "status: joystick unplugged"
  Else
    'Status.Caption = "status: joyGetPosEx error, rc = "& rc
 StatusBar1.Panels(1).Text = "status: joyGetPosEx Error"
  End If
End If

' Display the data on the form
'xa.Caption = ji.dwXpos
'ya.Caption = ji.dwYpos
'bt.Caption = ji.dwButtons
Dim data_buttom As String
Select Case ji.dwButtons
 Case 1
        If flag_butt = True Then
          data_buttom = "f" + Chr(10)
          main.Winsock1.SendData (data_buttom)
          flag_butt = False
        End If
 Case 2
        If flag_butt = True Then
          data_buttom = "1" + Chr(10)
          main.Winsock1.SendData (data_buttom)
          flag_butt = False
        End If
 Case 4
        If flag_butt = True Then
          data_buttom = "b" + Chr(10)
          main.Winsock1.SendData (data_buttom)
          flag_butt = False
        End If
 Case 8
```

```
        If flag_butt = True Then
           data_buttom = "2" + Chr(10)
           main.Winsock1.SendData (data_buttom)
           flag_butt = False
        End If
Case 16
        If flag_butt = True Then
           data_buttom = "s" + Chr(10)
           main.Winsock1.SendData (data_buttom)
           flag_butt = False
        End If
Case 256
        If flag_butt = True Then
           data_buttom = "z" + Chr(10)
           main.Winsock1.SendData (data_buttom)
           flag_butt = False
        End If
Case 64
        If flag_butt = True Then
           data_buttom = "0" + Chr(10)
           main.Winsock1.SendData (data_buttom)
           flag_butt = False
        End If
Case 128
        If flag_butt = True Then
           data_buttom = "3" + Chr(10)
           main.Winsock1.SendData (data_buttom)
           flag_butt = False
        End If
Case Else
        If flag_butt = False Then
           flag_butt = True
        End If
End Select
```

```
If (ji.dwXpos > 65000) Then
        If flag_xjoy = True Then
           Dim data_x1 As String
           data_x1 = "r" + Chr(10)
           main.Winsock1.SendData (data_x1)
           flag_xjoy = False
        End If
  Else
    If (ji.dwXpos < 100) Then
        If flag_xjoy = True Then
           Dim data_x2 As String
           data_x2 = "l" + Chr(10)
           main.Winsock1.SendData (data_x2)
           flag_xjoy = False
        End If
    Else
    'CENTER
        If flag_xjoy = False Then
           flag_xjoy = True
        End If
    End If
  End If


  If ji.dwYpos > 65000 Then
        If flag_yjoy = True Then
           Dim data_y1 As String
           data_y1 = "d" + Chr(10)
           main.Winsock1.SendData (data_y1)
           flag_yjoy = False
        End If
  Else
    If ji.dwYpos < 100 Then
        If flag_yjoy = True Then
           Dim data_y2 As String
```

```
            data_y2 = "u" + Chr(10)
            main.Winsock1.SendData (data_y2)
            flag_yjoy = False
          End If
      Else
      'CENTER
          If flag_yjoy = False Then
            flag_yjoy = True
          End If
      End If
End If
'Debug.Print ji.dwYpos, flag_yjoy


End Sub

Private Sub turn_left_Click()
Dim data As String
data = "2" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub turn_right_Click()
Dim data As String
data = "1" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub up_Click()
Dim data As String
data = "u" + Chr(10)
main.Winsock1.SendData (data)
End Sub

Private Sub Winsock1_Close()
```

```
Winsock1.Close

End Sub

Private Sub Winsock1_Connect()
MsgBox "Connected. Success", vbOKOnly, "Success"
Timer1.Enabled = True
connect_but.Enabled = False
Frame1.Enabled = True
Frame2.Enabled = True
disconnect_but.Enabled = True
connect.Hide
End Sub
```

```
'-------------------------------------------------------------------------

' connect_frm.frm

'-------------------------------------------------------------------------

Private Sub connect_cancal_Click()

 main.Winsock1.Close

main.Enabled = True

 connect.Hide

End Sub


Private Sub connect_ok_Click()

main.Enabled = True

If host = "" Then

   MsgBox "No host name specified", vbOKOnly, "Error"

Else

    main.Winsock1.connect host, 3002

End If

End Sub
```

ภาคผนวก ค.

โปรแกรม Liux Robot Server

```
/*
        Makefile for Linux Robot
*/
CC = cc
project := lbot
objects := main.o lbotsock.o c52lbot.o
cflags  :=
libs    :=

$(project) : $(objects)
        $(CC) $(cflags) -o $(project) $(objects) $(libs)

main.o : main.c lbotsock.o c52lbot.o
        $(CC) $(cflags) -c main.c

lbotsock.o : lbotsock.c lbotsock.h
        $(CC) $(cflags) -c lbotsock.c

c52lbot.o : c52lbot.c  c52lbot.h
        $(CC)  $(cflags)  -c  c52lbot.c
```

```
/*
        lbotsock.h
        create 22.55 ,2 Jan 2001
        By Benchapol Tunhoo 40054028 (Rv.B)
*/
#define NSTRS       3        /* number. of strings  */
#define PORT    3002

void initsock();
void acept_con();
int Send_str(char *ch,int len_str);
char  Recv_ch();
char Rec_ch();
void Close_sock();
```

```
/*          lbotsock.c
            create 22.55 ,2 Jan 2001
            by Benchapol Tunhoo 40054028 (Rv.B)
*/

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <string.h>
#include <netinet/in.h>
#include <errno.h>
#include <sys/wait.h>
#include <stdio.h>
#include "lbotsock.h"



int  ns, len,sockfd;
struct sockaddr_un saun, fsaun;
struct sockaddr_in my_addr;
struct sockaddr_in their_addr;
int sin_size;
FILE *fp;
/*
* Strings we send to the client.
*/
  char *strs[NSTRS] = {
            "Welcome to Internet Robot Control.\n",
                  "\n",
                  "login :"
                        };



void initsock()
{
```

```
        if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            perror("server: socket");
            exit(1);

          }
          my_addr.sin_family = AF_UNIX;
         my_addr.sin_port=htons(PORT);
                my_addr.sin_addr.s_addr=INADDR_ANY;
                bzero(&(my_addr.sin_zero),8);
        if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr))==-1) {
            perror("server: bind");
            exit(1);
          }

          if (listen(sockfd, 5) < 0) {
            perror("server: listen");
            exit(1);
          }
}

void acept_con()
{
                sin_size=sizeof(struct sockaddr_in);
    if ((ns = accept(sockfd,(struct sockaddr *)&their_addr,&sin_size))< 0)
        {
            perror("server: accept");
            exit(1);
          }
                fp=fdopen(ns,"r");

}

int Send_str(char *ch,int len_str)
{
         send(ns,ch,len_str,0);
```

```
}

char  Recv_ch()
{
        char *buf;
        recv(ns,buf,1,0);
        return (char)*buf;
}

char Rec_ch()
{
        char ch;
        while((ch=fgetc(fp))!=EOF)
        {
        return ch;
        }
}

void Close_sock()
{
        close(ns);
}
```

```
/*  c52lbot.h        */

int open_port(void);
void init_rs232(void);
void end_rs232(void);

void stepX(int step);  // use step = 0,1,2,...,255
void stepY(int step);
void stepZ(int step);  //use

void motor1(int dir);  // use dir = -16,-15,...,0,...,15,16
void motor2(int dir);
void all_motor(int dir);

void M1(int op); // use op =-1,0,1
void M2(int op); //use
void MB(int op); //use
```

```
/*    Driver RS 232 for c52 linux robot */
 /*    By SantiChar PoungKaew */
 /*    Create 8:50 P.M. 19/12/00 */
 /*    Update 4:20 P.M. 21/12/00 */

 #include <stdio.h>   /* Standard input/output definitions */
 #inciude <string.h>  /* String function definitions */
 #include <unistd.h>  /* UNIX standard function definitions */
 #include <fcntl.h>   /* File control definitions */
 #include <errno.h>   /* Error number definitions */
 #include <termios.h> /* POSIX terminal control definitions */


static struct termios orig, new;
static int peek = -1;


int  mainfd;
struct termios options;

 /*
  * 'open_port()' - Open serial port 1.
  *
  * Returns the file descriptor on success or -1 on error.
  */

 int open_port(void)
 {
   int fd;                      /* File descriptor for the port */

   fd = open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NDELAY);

   if (fd == -1)
   {                                  /* Could not open the port */
     fprintf(stderr, "open_port: Unable to open /dev/ttyS1 - %s\n",
```

```
        strerror(errno));
  }

  return (fd);
}

init_rs232()
{

 mainfd = open_port();

 fcntl(mainfd, F_SETFL, FNDELAY);            /* Configure port reading */
                        /* Get the current options for the port */
 tcgetattr(mainfd, &options);
 cfsetispeed(&options, B9600);          /* Set the baud rates to 9600 */
 cfsetospeed(&options, B9600);

                        /* Enable the receiver and set local mode */
 options.c_cflag |= (CLOCAL | CREAD);
 options.c_cflag &= ~PARENB; /* Mask the character size to 8 bits, no parity */
 options.c_cflag &= ~CSTOPB;
 options.c_cflag &= ~CSIZE;
 options.c_cflag |=  CS8;                    /* Select 8 data bits */
 options.c_cflag &= ~CRTSCTS;           /* Disable hardware flow control */

                    /* Enable data to be processed as raw input */
 options.c_lflag &= ~(ICANON | ECHO | ISIG);

                        /* Set the new options for the port */
 tcsetattr(mainfd, TCSANOW, &options);

 }


 end_rs232()
```

```
{
 close(mainfd);
}


stepX(int step)
{
 int i;
 char command[10];

 sprintf(command,"x=%d\n",step);

  for(i=0;command[i]!= 0;i++)
 {
  write(mainfd,&command[i],1);
  usleep(100000);
 }
}

 stepY(int step)
 {
  int i;
  char command[10];

  sprintf(command,"y=%d\n",step);

   for(i=0;command[i]!= 0;i++)
  {
   write(mainfd,&command[i],1);
   usleep(100000);
  }

 }

 stepZ(int step)
```

```
{
 int i;
 char command[10];

 sprintf(command,"z=%d\n",step);

  for(i=0;command[i]!= 0;i++)
 {
  write(mainfd,&command[i],1);
  usleep(100000);
 }
}

void motor1(int dir)
{
 int i;
 char command[10];

  sprintf(command,"a=%d\n\n",dir);

   for(i=0;command[i]!= 0;i++)
  {
   write(mainfd,&command[i],1);
   usleep(100000);
  }

 }

 void motor2(int dir)
 {
  int i;
  char command[10];

   sprintf(command,"b=%d\n\n",dir);
```

```
 for(i=0;command[i]!= 0;i++)
{
 write(mainfd,&command[i],1);
 usleep(100000);
 }

}

void all_motor(int dir)
{
 int i;
 char command[10];

 sprintf(command,"a=%d,b=%d\n\n",dir,dir);

  for(i=0;command[i]!= 0;i++)
 {
  write(mainfd,&command[i],1);
  usleep(100000);
 }
}

 void M1(int op)
 {
  int i;
  char command[10];
  if(op==1)
  {
  sprintf(command,"A\n");
  }
  if(op==0)
  {
  sprintf(command,"M\n");
  }
  if(op==-1)
```

```
{
sprintf(command,"C\n");
}
 for(i=0;command[i]!= 0;i++)
{
 write(mainfd,&command[i],1);
 usleep(100000);
}

}


void M2(int op)
{
 int i;
 char command[10];
 if(op==1)
 {
 sprintf(command,"B\n");
 }
 if(op==0)
 {
 sprintf(command,"N\n");
 }
 if(op==-1)
 {
 sprintf(command,"D\n");
 }
  for(i=0;command[i]!= 0;i++)
 {
  write(mainfd,&command[i],1);
  usleep(100000);
 }

}
```

```
void MB(int op)
{
 int i;
 char command[10];
 if(op==1)
 {
 sprintf(command,"F\n");
 }
 if(op==0)
 {
 sprintf(command,"X\n");
 }
 if(op==-1)
 {
 sprintf(command,"R\n");
 }
  for(i=0;command[i]!= 0;i++)
 {
  write(mainfd,&command[i],1);
  usleep(100000);
 }

}
```

```
/*
        main.c  for  Linux Robot
        create by Benchapol Tunhoo 40054028 (Rv.B)
*/
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include "lbotsock.h"
#include "c52lbot.h"
#include "i8255.h"

int fda;
struct i8255_struct A;

main()
{
        char c;
                int i;
                int flag_x,flag_z;
                int xaxis_num,zaxis_num;
                initsock();
                init_rs232();
        init_i8255();
  while(1)
        {
                acept_con();
         xaxis_num=0;
                zaxis_num=0;
                Send_str("Good\n",5);
      for(i=0;i<1024;i++)
        {
                c=Rec_ch();

        switch(c)
        {
```

```
        case 'l':
        {
        printf("Step turn left\n");
        xaxis_num++;
        if(xaxis_num == 5)
        {
        xaxis_num=4;
        printf("xaxis_num = 5\n");
        break;
        }
        else
        {
        printf("xaxis_num : %d\n",xaxis_num);
        stepX(10);
        break;
        }
        }
        case 'r':
{
printf("Step turn right\n");
        xaxis_num=xaxis_num-1;
        if(xaxis_num == -5)
{
xaxis_num= -4;
printf("xaxis_num = -4\n");
break;
}
else
{
        printf("xaxis_num : %d\n",xaxis_num);
stepX(-10);
break;
        }
}
        case 'u':
```

```
{
printf("Step turn up\n");
        zaxis_num++;
         if(zaxis_num == 5)
{
zaxis_num=4;
printf("zaxis_num = 5\n");
break;
}
else
{
        printf("zaxis_num : %d\n",zaxis_num);
stepZ(10);
break;
        }
}
         case 'd':
{
printf("Step turn down\n");
        zaxis_num=zaxis_num-1;
         if(xaxis_num == -5)
{
xaxis_num= -4;
printf("zaxis_num = -5\n");
break;
}
else
{
        printf("zaxis_num : %d\n",zaxis_num);
stepZ(-10);
break;
        }
}
         case 'f':
{
```

```
printf("DC Motor go forward\n");
MB(1);
break;
}
        case 's':
{
printf("DC Motor stop\n");
MB(0);
break;
}
        case 'b':
{
printf("DC Motor go back\n");
MB(-1);
break;
}
        case '1':
{
printf("DC Motor turn right\n");
MB(0);
        M1(1);
break;
}
        case '2':
{
printf("DC Motor turn left\n");
MB(0);
M2(1);
        break;
}


        printf("return zero\n");
/*------ LED on ----------------*/
        case '0':
```

```
        {
        printf("LED on\n");
                Led_On(0);
        break;
        }
/*-------------------------------*/
/*------ LED off ------------------*/
        case '3':
        {
        printf("LED off\n");
        Led_Off(0);
        break;
        }
/*-------------------------------*/
/*------ return to zero x xais ----------*/
                        case 'z':
                {
        printf("return zero\n");
                        if(xaxis_num>0)
                        {
                        for(flag_x=0;flag_x < xaxis_num;flag_x++)
                                {
                                stepX(-10);
                                usleep(500000);
                                printf(">0 : %d\n",flag_x);
                                }
                                Send_str("complete\n",9);
                        }
                        else
                        {
                        if(xaxis_num < 0)
                                {
                for(flag_x=0 ; flag_x < abs(xaxis_num);flag_x++)
                                {
                                stepX(10);
```

```
                usleep(500000);
                printf("<0 ;%d\n",flag_x);
            }
            Send_str("complete\n",9);
                }
            else
                {
                printf("=0\n");
                }
            }
/*------------end  return x axis ---------*/
/*------------return z axis-------------*/
            if(zaxis_num>0)
        {
        for(flag_z=0;flag_z<zaxis_num;flag_z++)
            {
            stepZ(-10);
            usleep(500000);
            printf(">0 : %d\n",flag_z);
            }
            Send_str("complete\n",9);
        }
        else
                {
        if(zaxis_num < 0)
            {
        for(flag_z=0;flag_z < abs(zaxis_num);flag_z++)
        {
            stepZ(10);
            usleep(500000);
            printf("<0 ;%d\n",flag_z);
        }
        Send_str("complete\n",9);
            }
        else
```

```
                                        {
                        printf("=0\n");
                        }
                }
/*----------end  return z axis ----------*/
                        xaxis_num=0;
                        zaxis_num=0;
                        break;
                        }
                        case 'q':
                        {
                        printf("quit");
                        /*xaxis_num=0;
                        zaxis_num=0;*/
                        Close_sock();
            exit(0);
                        }
        }
            }
            }
/*
                        Close_sock();
                        exit(0);
*/
            end_rs232();
}

Led_On(int bit) // PORTA BIT0 .. BIT7
{
  struct i8255_struct AA;
  AA.regnum = REG_A0;
  AA.bitno = bit;
    ioctl(fda, I8255_CLRBIT, &AA);
}
```

```
Led_Off(int bit)
{
   struct i8255_struct AA;
   AA.regnum = REG_A0;
   AA.bitno = bit;
   ioctl(fda, I8255_SETBIT, &AA);
}

init_i8255()
{
  fda = open("/dev/i8255a", O_RDWR);
  if( fda == -1) {fprintf(stderr,"error :< \n"); }

  A.regnum = REG_D0;
  A.value = 0x80;   // out portA = 0x80
  ioctl(fda, I8255_WRITE, &A);
}
```

```
/*****************************************************************************
 *
 *          i8255 Loadable Module for Linux
 *
 *
 * $Id: i8255.h,v 1.10 1998/10/14 20:28:20 sanne Exp $
 *
 * Copyright (c) 1998 by Sanne Graaf (sanne@mmm.xs4all.nl).
 *   All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the Free
 * Software Foundation; either version 2 of the License, or (at your option)
 * any later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
 * more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; see the file COPYING.  If not, write to
 * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
 *
 *****************************************************************************/
#ifndef INCLUDED_8255_DRIVER
  #define INCLUDED_8255_DRIVER

  #define MAX_DEVICES           10          /* max numbe- of 8255's */


/*****************************************************************************
 * Functions
 *
 */
```

```
#define I8255_RESET        0        /* RESET/INIT CARD*/
#define I8255_DEBUG        1            /* Set Debug level        */
#define I8255_EXCL         2            /* Set Device Exclusive*/
#define I8255_TEST         3        /* Display all values in syslog*/

#define i8255_READ         4        /* Read i8255 Register        */
#define I8255_WRITE        5        /* Write i8255 Register*/

#define I8255_SETBIT       6        /* Set one bit on i8255*/
#define I8255_CLRBIT       7        /* Clear one bit on i8255*/
#define I8255_READBIT      8        /* Read one bit on i8255*/

/*****************************************************************************
* Register Mask bits and status register masks
*
*/

 #define BIT0              0xFE    /* inverse mask bit 0        */
 #define BIT1              0xFD    /*                           */
 #define BIT2              0xFB    /*                           */
 #define BIT3              0xF7    /*                           */
 #define BIT4              0xEF    /*                           */
 #define BIT5              0xDF    /*                           */
 #define BIT6              0xBF    /*                           */
 #define BIT7              0x7F    /* inverse mask bit 7        */

 #define I8255_USED        0x01    /* 8255 is in use  */
 #define I8255_IRQOK       0x02    /* IRQ is configured         */
 #define I8255_PORTOK      0x04    /* IO port configures        */

/*****************************************************************************
* Various defines
*
```

```
*/

//  #ifdef GENERIC_I8255

    #define DEV_NAME  "i8255"
    #define I8255_PORTNUM          4          /* 8255 number of ports      */

    #define I8255_MAJOR        64         /* device major number       */

    #define REG_A0                 0          /* Reg A0 offset*/
    #define REG_B0                 1          /* Reg B0 offset*/
    #define REG_C0                 2          /* Reg C0 offset*/
    #define REG_D0                 3          /* Reg D0 offset*/

    #define I8255_A0(num)          num+REG_A0     /* 8255 port A    */
    #define I8255_B0(num)          num+REG_B0     /* 8255 port B    */
    #define I8255_C0(num)          num+REG_C0     /* 8255 port C    */
    #define I8255_D0(num)          num+REG_D0     /* 8255 status register*/

//  #endif


  #define ON             1        /* general ON   */
  #define OFF            0        /* general OFF  */

  #define I8255_VERSION"0.2"          /* version number */

/*****************************************************************************
 *  General Structures
 *
 *
 */

struct i8255_struct {              /* communication struct      */
  int regnum;                      /* register (REG_A0 ..)      */
```

```
  u_char value;                    /* value to write       */
  u_char bitno;                    /* bitmask for set/clr bit  */
};

struct i8255_cfg {                 /* One 8255 Cfg structure   */
  int iobase;                      /* 8255 IO base address     */
  int irq;                 /* 8255 IRQ number          */
  u_char data_regs[I8255_PORTNUM];   /* current register values  */
  u_char i8255stat;                /* bitmapped status byte    */
  u_char exclusive;                /* allow multi usage or not */
  u_char usenum;                   /* number of uses           */
};


static struct i8255_cfg i8255_dev[MAX_DEVICES]; /* i8255 config struct array */
static struct wait_queue *i8255_wait;

  #ifdef __KERNEL__


/*****************************************************************************
 *  Module Function Prototypes.
 *
 */

static int i8255_read(struct inode *inode, struct file *file, char *buffer, int count);
static int i8255_open(struct inode *inode, struct file *file);
static int i8255_write(struct inode *inode, struct file *file, const char *buffer, int count);
static int i8255_ioctl(struct inode *inode, struct file *file, u_int cmd, u_long arg);
static void i8255_release(struct inode *inode, struct file *file);

/*****************************************************************************
 * /proc/i8255 stuff
 *
 */
```

```
static int i8255_get_info(char *, char**, off_t, int, int);

static struct proc_dir_entry i8255_proc_entry = {
   0, 5, "i8255", S_IFREG | S_IRUGO, 0, 0, 0, 0, NULL, i8255_get_info, NULL,
   NULL, NULL, NULL
};



/*****************************************************************************
 *  Module file operation structure.
 *
 *
 */

struct file_operations i8255_fops = {
   NULL,            /* lseek          */
   i8255_read,      /* read                    */
   i8255_write,     /* write          */
   NULL,            /* readdir                 */
   NULL,            /* select         */
   i8255_ioctl,     /* ioctl          */
   NULL,            /* mmap                    */
   i8255_open,      /* open                    */
   i8255_release,   /* release                 */
   NULL,            /* fsync          */
   NULL,            /* fasync                  */
   NULL,            /* check_media_change      */
   NULL,            /* revalidate              */
};

   #endif //__KERNEL__
#endif //INCLUDED_8255_DRIVER
```

```
 *
 * Reads one byte from ioport
 *
 * Args: ioport -> ioport to read from
 *
 * Return: byte read
 */

u_char ioread(int ioport)
{
   u_char byte_read;

   if (slow8255)
      byte_read = inb_p(ioport);
   else
      byte_read = inb(ioport);

   my_printk(3, "ioread-> I/O Port 0x%X -> 0x%X\n", ioport, byte_read);
   return byte_read;
}


/*****************************************************************************
 * IO write routine
 *
 * Writes one byte to ioport
 *
 * Args: outbyte -> byte to send
 *       ioport  -> ioport to send to
 *
 * Return: always 0
 */

u_char iowrite(int ioport, u_char outbyte)
{
```

```
  my_printk(3, "iowrite-> I/O Port 0x%X -> 0x%X\n", ioport, outbyte);

  if (slow8255)
    outb_p(outbyte, ioport);
  else
    outb(outbyte, ioport);

  return (0);
}

/*****************************************************************************
 * Set one bit
 *
 * Sets one bit within a byte
 *
 * Args: reg -> byte to set bit in
 *       bit -> bitmask
 *       value -> local copy of register
 *
 * Return: new register value
 *
 */

u_char set_onebit(int reg, u_char bit, u_char value)
{
  u_char val;

  val = value;
  my_printk(2, "set_onebit-> origional val: 0x%X (mask 0x%X)\n", val, bit);
  val |= ~bit;
  my_printk(2, "set_onebit-> new val: 0x%X\n", val);
  iowrite(reg, val);
  return (val);
}
```

```
/*****************************************************************************
 * Clear one bit
 *
 * Clear's a bit in a byte
 *
 * Args: reg -> byte to clear bit in
 *          bit -> bitmask
 *          value -> local copy of register
 *
 * Return: new register value
 *
 */

u_char clr_onebit(int reg, u_char bit, u_char value)
{
  u_char val;

  val = value;
  my_printk(2, "clr_onebit-> origional val: 0x%X (mask 0x%X)\n", val, bit);
  val &= bit;
  my_printk(2, "clr_onebit-> new val: 0x%X\n", val);
  iowrite(reg, val);
  return (val);
}


/*****************************************************************************
 * Read from device
 *
 * routine to handle things like 'cat /dev/i8255x' ...
 *
 * Args: inode -> inode structure
 *          file -> file structure
```

```
*           buffer -> buffer voor gelezen data
*           count -> aantal te lezen bytes
*
* Return: data from buffer in user-space allocated memory
*
*/

static int i8255_read(struct inode *inode, struct file *file, char *buffer,
                      int count)
{
  int offset, i;

  my_printk(1, "i8255_read-> Request for %d bytes, minor: %d\n",
          count, MINOR(inode->i_rdev));
  if (count<=0) return -EINVAL;
  if (count > I8255_PORTNUM) count = I8255_PORTNUM;

  for (i=0; i < I8255_PORTNUM; i++)
    i8255_dev[MINOR(inode->i_rdev)].data_regs[i] =
    ioread(i8255_dev[MINOR(inode->i_rdev)].iobase+i);

  my_printk(2, "i8255_read-> Bytes: A:0x%X B:0x%X C:0x%X D:0x%X\n",
          i8255_dev[MINOR(inode->i_rdev)].data_regs[0],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[1],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[2],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[3]);

  for (offset=0; offset<count; offset++)
    put_fs_byte(i8255_dev[MINOR(inode->i_rdev)].data_regs[offset],
              buffer+offset);
  return offset;
}

/*****************************************************************************
```

```
* Write to device
*
* routine to handle things like 'cat [whatever] > /dev/i8255' ...
*
* Args: inode -> inode structure
*       file -> file structure
*       buffer -> buffer voor te schrijven data
*       count -> aantal te schrijven bytes
*
* Return: data from buffer in user-space allocated memory
*
*/

static int i8255_write(struct inode *inode, struct file*file,
                const char *buffer, int count)
{

  int orig_count, offset, i;

  my_printk(1, "i8255_write-> Request for %d bytes\n", count);
  orig_count = count;
  if (count > I8255_PORTNUM) count = I8255_PORTNUM;
  for (offset=0; offset<count; offset++) {
    i8255_dev[MINOR(inode->i_rdev)].data_regs[offset] =
    get_fs_byte(buffer+offset);
    iowrite(i8255_dev[MINOR(inode->i_rdev)].iobase+offset,
          i8255_dev[MINOR(inode->i_rdev)].data_regs[offset]);
  }
  my_printk(2, "i8255_write-> Bytes: A:0x%X B:0x%X C:0x%X D:0x%X\n",
          i8255_dev[MINOR(inode->i_rdev)].data_regs[0],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[1],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[2],
          i8255_dev[MINOR(inode->i_rdev)].data_regs[3]);
  return orig_count;
}
```

```
/*****************************************************************************
 * Open and Lock device
 *
 * Orig: Routine to prevent more filehandles to be opened to the device
 * Now: only counts devices who use the device
 *
 * Args: inode -> inode structure
 *       file -> file structure
 *
 * Return: nothing (0).
 *
 */

static int i8255_open(struct inode *inode, struct file *file)
{
  my_printk(1, "i8255_open-> Request, minor: %d\n",
        MINOR(inode->i_rdev));
  if ((i8255_dev[MINOR(inode->i_rdev)].i8255stat & I8255_USED) &&
     (i8255_dev[MINOR(inode->i_rdev)].exclusive == 1)) {
    my_printk(1, "i8255_open-> Request on exclusive device, minor: %d\n",
          MINOR(inode->i_rdev));
    return -EBUSY;
  }
  MOD_INC_USE_COUNT;
  i8255_dev[MINOR(inode->i_rdev)].i8255stat |= I8255_USED;
  i8255_dev[MINOR(inode->i_rdev)].usenum++;
  my_printk(2, "i8255_open-> Opened %d times, minor: %d\n",
        i8255_dev[MINOR(inode->i_rdev)].usenum,
        MINOR(inode->i_rdev));
  return 0;
}

/*****************************************************************************
```

```
* Unlock device
*
* Routine to unlock device / decrement number of users
*
* Args: inode -> inode structure
*       file -> file structure
*
* Return: nothing (0).
*
*/

static void i8255_release(struct inode *inode, struct file *file)
{
  my_printk(1, "i8255_release-> Request, minor: %d\n",
        MINOR(inode->i_rdev));
  MOD_DEC_USE_COUNT;
  i8255_dev[MINOR(inode->i_rdev)].usenum--;
  if (i8255_dev[MINOR(inode->i_rdev)].usenum == 0)
    i8255_dev[MINOR(inode->i_rdev)].i8255stat &= ~I8255_USED;
}


/*****************************************************************************
 * ioctl function
 *
 * Command based ioctl routine. This will do all the work to/from the 8255
 *
 * Args: inode -> inode structure
 *       file -> file structure
 *       cmd -> command to execute (read/write/clear/set/reset)
 *       arg -> argument, is used as pointer to i8255_struct
 *
 */

static int i8255_ioctl(struct inode *inode, struct file *file,
```

```
                u_int cmd, u_long arg)
{
  int rc, i, temp, portnum;
  struct i8255_struct request;

  my_printk(1, "i8255_ioctl-> func=0x%X, arg=0x%X, minor=%d\n",
        cmd, (int) arg, MINOR(inode->i_rdev));

  if (cmd > I8255_TEST) {
    rc = verify_area(VERIFY_READ, (struct request*) arg, sizeof(request));
    if (rc)
      return rc;
    memcpy_fromfs(&request, (struct i8255_struct*) arg, sizeof(request));
    portnum = i8255_dev[MINOR(inode->i_rdev)].iobase + request.regnum;
    if ((portnum < i8255_dev[MINOR(inode->i_rdev)].iobase) ||
      (portnum > i8255_dev[MINOR(inode->i_rdev)].iobase+I8255_PORTNUM-1)) {
      my_printk(0, "i8255_ioctl -> ERROR: Wrong IOport address: 0x%X, minor: %d\n",
          portnum, MINOR(inode->i_rdev));
      return -EINVAL;
    }
  }

  switch (cmd) {
    case I8255_READ:                    /* Read register   */
        i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum] =
          ioread(portnum);
      return (i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum]);

    case I8255_WRITE:                   /* Write register   */
      iowrite(portnum, request.value);
      i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum] =
      request.value;
      break;

    case I8255_SETBIT:                  /* Sets one bit     */
```

```
  i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum] =
  set_onebit(portnum, request.bitno,
          i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum]);
  break;

case I8255_CLRBIT:                    /* Clears one bit  */
  i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum] =
  clr_onebit(portnum, request.bitno,
          i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum]);
  break;

case I8255_READBIT:                   /* Reads one bit  */
  temp = 0;
  i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum] =
  ioread(portnum);
  temp = i8255_dev[MINOR(inode->i_rdev)].data_regs[request.regnum]
       & ~request.bitno;
  my_printk(2, "i8255_ioctl -> READBIT -> 0x%X\n", temp);
  if (temp)
    return (1);
  else
    return (0);
  break;

case I8255_DEBUG:
       if ((arg > 0) && (arg < 4))
         debug8255 = arg;
       else
         debug8255 = 0;
           my_printk(0, "ioctl -> new DEBUG level: %d\n", debug8255);
  break;

case I8255_EXCL:
           if (arg != 0)
             i8255_dev[MINOR(inode->i_rdev)].exclusive = 1;
```

```
            else
                i8255_dev[MINOR(inode->i_rdev)].exclusive = 0;
        break;

    case I8255_RESET:           /* Init 8255, all input     */

#ifdef GENERIC_I8255
        iowrite(I8255_D0(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x9B);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_D0] = 0x9B;
#endif
#ifdef ADVANTECH_PCL731
        iowrite(I8255_D0(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x9B);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_D0] = 0x9B;
        iowrite(I8255_D1(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x9B);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_D1] = 0x9B;
#endif
#ifdef ADVANTECH_PCM3724
        iowrite(I8255_D0(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x9B);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_D0] = 0x9B;
        iowrite(I8255_D1(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x9B);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_D1] = 0x9B;
        iowrite(I8255_DIR(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x00);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_DIR] = 0x00;
        iowrite(I8255_GATE(i8255_dev[MINOR(inode->i_rdev)].iobase), 0x00);
        i8255_dev[MINOR(inode->i_rdev)].data_regs[REG_GATE] = 0x00;
#endif

        for (i=0; i < I8255_PORTNUM; i++)
          i8255_dev[MINOR(inode->i_rdev)].data_regs[i] =
                    ioread(i8255_dev[MINOR(inode->i_rdev)].iobase+i);

          break;
      default:                      /* invalid function number */
        return -EINVAL;
    }
```

```
  return 0;
}


/*****************************************************************************
 * I8255 Interrupt routine
 *
 */

static void i8255_interrupt(int irq, void *dev_id, struct pt_regs *regs)
{
  struct i8255_dev *dev = (struct i8255_dev *) dev_id;
  int i;
  static int isr_active = 0;

  my_printk(1, "Interrupt: %d\n", irq);
  if (isr_active) {
    my_printk(2, "!!__Nested Interrupt__!!.\n");
    return;
  }
  isr_active++;
  if (dev == NULL) {
    my_printk(0, "Interrupt from unknown device.\n");
  }
  for (i=0; i < num_devices; i++) {
    my_printk(1, "port: %X\n", i8255_dev[i].iobase);
    if (i8255_dev[i].irq == irq) {
      my_printk(1, "Got Interrupt from device %d, io 0x%X.\n", i, i8255_dev[i].iobase);
          i8255_dev[i].data_regs[REG_C0] = ioread(i8255_dev[i].iobase+REG_C0);
    }
  }
  wake_up_interruptible(&i8255_wait);
  isr_active--;
}
```

```
/*****************************************************************************
 * Static device init
 *
 * Init routine for kernel module, grabs I/O range and if used interrupt
 *
 */

int i8255_init(void)
{
  int i, j;

  if (register_chrdev(I8255_MAJOR, DEV_NAME, &i8255_fops)) {
    my_printk(0, "Cannot register to major device: %d\n",
        I8255_MAJOR);
    return -EIO;
  }
  my_printk(1, "Registered to major device %d.\n", I8255_MAJOR);

  if (proc > 0) {
    i8255_proc_entry.namelen = strlen(DEV_NAME);
    i8255_proc_entry.name = DEV_NAME;
    j = proc_register_dynamic(&proc_root, &i8255_proc_entry);
    my_printk(1, "Registered /proc/%s, ret: %d, num: %d\n", DEV_NAME,j,
i8255_proc_entry.low_ino);
  }

  for (i = 0; i < num_devices; i++) {
    i8255_dev[i].iobase = ioport[i];
    i8255_dev[i].irq = irq[i];
    i8255_dev[i].i8255stat = 0;
    i8255_dev[i].usenum = 0;
    i8255_dev[i].exclusive = exclusive[i];

    if (check_region(i8255_dev[i].iobase, I8255_PORTNUM) < 0) {
```

```
      my_printk(0, "I/O PORT region (0x%X) is taken by other device\n",
            i8255_dev[i].iobase);
      continue;
    } else i8255_dev[i].i8255stat |= I8255_PORTOK;

    request_region(i8255_dev[i].iobase, I8255_PORTNUM, DEV_NAME);

    my_printk(2, "i8255_init-> Request I/O ports (0x%x - 0x%x), exclusive: %d.\n",
            i8255_dev[i].iobase, i8255_dev[i].iobase+I8255_PORTNUM-1,
            i8255_dev[i].exclusive);

    if (i8255_dev[i].irq > 0) {
      if (request_irq(i8255_dev[i].irq, i8255_interrupt, 0, DEV_NAME, i8255_dev)) {
        my_printk(0, "IRQ %d taken by other device.\n",
              i8255_dev[i].irq);
        continue;
      } else i8255_dev[i].i8255stat |= I8255_IRQOK;

      my_printk(1, "i8255_init-> Request IRQ %d\n", i8255_dev[i].irq);
    }
  }
  return 0;
}


/*****************************************************************************
 * Register and init dynamic module
 *
 */

int init_module(void)
{
   int errnum=0;

   if ((debug > 0) && (debug < 4))
```

```
    debug8255 = debug;
  else
    debug8255 = 0;

  if (slowio > 0)
    slow8255 = 1;

  my_printk(0, "Generic 8255 Kernel module v%s loaded.\n", I8255_VERSION);
  my_printk(1, "$Id: i8255.c,v 1.9 1998/10/07 21:26:10 sanne Exp $.\n");
  my_printk(0, "by Sanne Graaf, <sanne@mmm.xs4all.nl>.\n");
  my_printk(1, "Debug level: %d.\n", debug8255);
  if (slow8255)
    my_printk(0, "Slow IO mode.\n");

  while (ioport[num_devices] > 0) {
    if (num_devices >= MAX_DEVICES) {
      my_printk(0, "Init 8255, to many devices.\n");
          break;
    }
    my_printk(1, "8255 num: %d, BasePort: 0x%X, IRQ: %d\n",
          num_devices, ioport[num_devices], irq[num_devices]);
    num_devices++;
  }
  my_printk(1,"Start Init 8255's, total: %d\n", num_devices);

  errnum = i8255_init();
  if (errnum != 0) {
     my_printk(0,"Error(s) during Init 8255's: %d\n", errnum);
  }
  return 0;
}


/*****************************************************************************
 * Unregister device dynamic module and cleanup.
```

```
 *
 */

void cleanup_module(void)
{
  int i = 0, j;

  unregister_chrdev(I8255_MAJOR, DEV_NAME);
  while (i < num_devices) {
    if (i8255_dev[i].i8255stat & I8255_PORTOK) {
      release_region(i8255_dev[i].iobase, I8255_PORTNUM);
          my_printk(1, "Released IObase 0x%02X-0x%02X\n",
            i8255_dev[i].iobase, i8255_dev[i].iobase+I8255_PORTNUM);
    }
    if ((irq[i] != 0) && (i8255_dev[i].i8255stat & I8255_IRQOK)) {
        my_printk(1, "Freeing IRQ %d.\n", i8255_dev[i].irq);
      free_irq(i8255_dev[i].irq, i8255_dev);
    }
    my_printk(1, "cleanup_module-> dev: %d, Released Baseport: 0x%02X, IRQ: %d\n",
          i, i8255_dev[i].iobase, i8255_dev[i].irq);
    i++;
  }

  if (proc > 0) {
     j = proc_unregister(&proc_root, i8255_proc_entry.low_ino);
     my_printk(1, "Unregistered /proc/%s, ret: %d, num: %d\n", DEV_NAME, j,
i8255_proc_entry.low_ino);
  }

  my_printk(0, "Module unloaded.\n");
}


/*****************************************************************************
 * /proc info routines.
```

```
*
*/

int i8255_get_info(char *buf, char **start, off_t fpos, int length, int dummy)
{
   char *p;
   int i;

   p = buf;

   p += sprintf(p, "Type     : %s\nNumdevices: %d\nDebug    : %d\nSlow I/O  : %d\nMajor    :
%d\n\n",
                  DEV_NAME, num_devices, debug8255, slow8255, I8255_MAJOR);

#ifdef GENERIC_I8255
   p += sprintf(p, "Num IOBase IRQ X Use Stat [A0] [B0] [C0] [D0]\n");
   for (i=0; i < num_devices; i++) {
   p += sprintf(p, "%3d  0x%03X  %02d %d   %d 0x%02X 0x%02X 0x%02X 0x%02X 0x%02X\n",i,
i8255_dev[i].iobase, i8255_dev[i].irq,
                  i8255_dev[i].exclusive, i8255_dev[i].usenum, i8255_dev[i].i8255stat,
                  i8255_dev[i].data_regs[REG_A0],  i8255_dev[i].data_regs[REG_B0],
                  i8255_dev[i].data_regs[REG_C0],  i8255_dev[i].data_regs[REG_D0]);
   }
#endif
#ifdef ADVANTECH_PCL731
   p += sprintf(p, "Num IOBase IRQ X Use Stat [A0] [B0] [C0] [D0] [A1] [B1] [C1] [D1]\n");
   for (i=0; i < num_devices; i++) {
   p += sprintf(p, "%3d  0x%03X  %02d %d   %d 0x%02X 0x%02X 0x%02X 0x%02X 0x%02X
0x%02X 0x%02X 0x%02X 0x%02X\n",i, i8255_dev[i].iobase, i8255_dev[i].irq,
                  i8255_dev[i].exclusive, i8255_dev[i].usenum, i8255_dev[i].i8255stat,
                  i8255_dev[i].data_regs[REG_A0],  i8255_dev[i].data_regs[REG_B0],
                  i8255_dev[i].data_regs[REG_C0],  i8255_dev[i].data_regs[REG_D0],
                  i8255_dev[i].data_regs[REG_A1],  i8255_dev[i].data_regs[REG_B1],
                  i8255_dev[i].data_regs[REG_C1],  i8255_dev[i].data_regs[REG_D1]);
   }
```

```
#endif
#ifdef ADVANTECH_PCM3724
    p += sprintf(p, "Num IOBase IRQ X Use Stat [A0] [B0] [C0] [D0] [A1] [B1] [C1] [D1] [DR] [GT]\n");
    for (i=0; i < num_devices; i++) {
    p += sprintf(p, "%3d  0x%03X  %02d %d   %d 0x%02X 0x%02X 0x%02X 0x%02X 0x%02X
0x%02X 0x%02X 0x%02X 0x%02X 0x%02X 0x%02X\n",i, i8255_dev[i].iobase, i8255_dev[i].irq,
                    i8255_dev[i].exclusive, i8255_dev[i].usenum, i8255_dev[i].i8255stat,
                    i8255_dev[i].data_regs[REG_A0],  i8255_dev[i].data_regs[REG_B0],
                    i8255_dev[i].data_regs[REG_C0],  i8255_dev[i].data_regs[REG_D0],
                    i8255_dev[i].data_regs[REG_A1],  i8255_dev[i].data_regs[REG_B1],
                    i8255_dev[i].data_regs[REG_C1],  i8255_dev[i].data_regs[REG_D1],
                    i8255_dev[i].data_regs[REG_DIR], i8255_dev[i].data_regs[REG_GATE]);
    }
#endif

    p += sprintf(p, "\n");
    return p - buf;
};
```

ภาคผนวก ง.

คู่มือของอุปกรณ์

Features

- Compatible with MCS-51™ Products
- 8 Kbytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 256 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-Bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

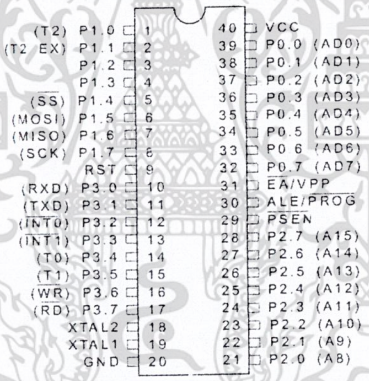
Description

The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8 Kbytes 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 80C51 and 80C52 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

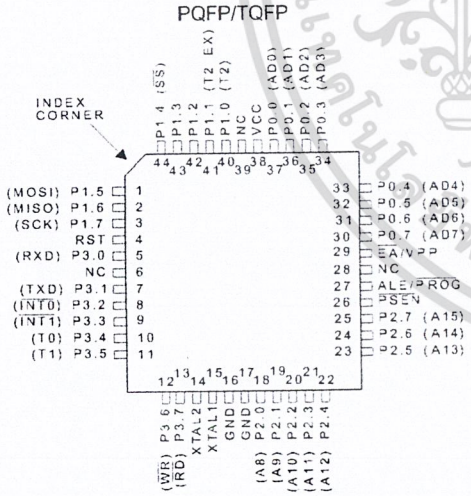
(continued)

Pin Configurations

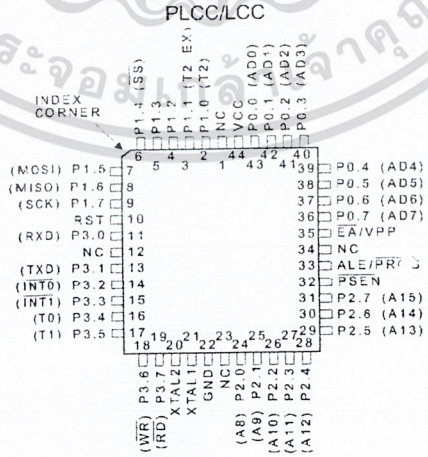
PDIP/Cerdip



PQFP/TQFP



PLCC/LCC

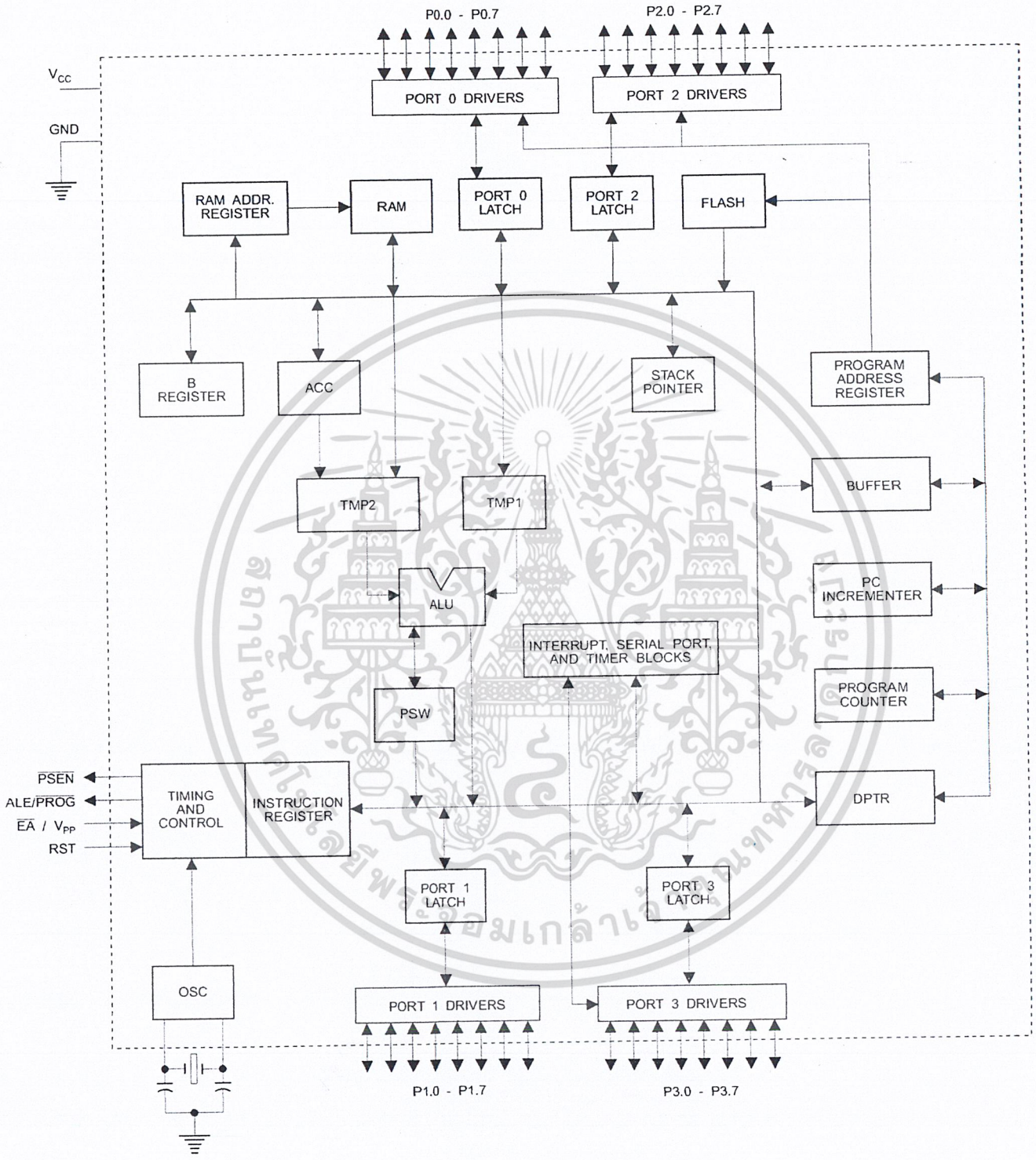


8-Bit Microcontroller with 8 Kbytes Flash

AT89C52

0313E

Block Diagram



Description (Continued)

The AT89C52 provides the following standard features: 8 Kbytes of Flash, 256 bytes of RAM, 32 I/O lines, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89C52 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 Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)

Port 1 also receives the low-order address bytes during Flash programming and program verification.

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. 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 AT89C51, as shown in the following table.

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)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

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

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

(continued)

Pin Description (Continued)

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcrontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89C52 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to VCC for internal program executions.

This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming when 12-volt programming is selected.

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89C52 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		0CFH
0C0H								0C7H
0B8H	IP XX000000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0X000000							0AFH
0A0H	P2 11111111							0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000			PCON 0XXX0000	87H

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 4) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.

Table 2. T2CON—Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
Bit	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/RL2 = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Data Memory

The AT89C52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions that use direct addressing access SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

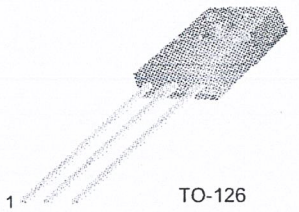
Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

FAIRCHILD
SEMICONDUCTOR TM

BD135/137/139

Medium Power Linear and Switching Applications

- Complement to BD136, BD138 and BD140 respectively



1

TO-126

1. Emitter 2.Collector 3.Base

NPN Epitaxial Silicon Transistor

Absolute Maximum Ratings T_C=25°C unless otherwise noted

Symbol	Parameter		Value	Units
V_{CBO}	Collector-Base Voltage	: BD135	45	V
		: BD137	60	V
		: BD139	80	V
V_{CEO}	Collector-Emitter Voltage	: BD135	45	V
		: BD137	60	V
		: BD139	80	V
V_{EBO}	Emitter-Base Voltage		5	V
I_C	Collector Current (DC)		1.5	A
I_{CP}	Collector Current (Pulse)		3.0	A
I_B	Base Current		0.5	A
P_C	Collector Dissipation (T_C=25°C)		12.5	W
P_C	Collector Dissipation (T_a=25°C)		1.25	W
T_J	Junction Temperature		150	°C
T_{STG}	Storage Temperature		- 55 ~ 150	°C

Electrical Characteristics T_C=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
V_{CEO}(sus)	Collector-Emitter Sustaining Voltage					
	: BD135	I_C = 30mA, I_B = 0	45			V
	: BD137		60			V
	: BD139		80			V
I_{CBO}	Collector Cut-off Current	V_{CB} = 30V, I_E = 0			0.1	μA
I_{EBO}	Emitter Cut-off Current	V_{EB} = 5V, I_C = 0			10	μA
h_{FE1}	DC Current Gain : ALL DEVICE	V_{CE} = 2V, I_C = 5mA	25			
h_{FE2}	: ALL DEVICE	V_{CE} = 2V, I_C = 0.5A	25			
h_{FE3}	: BD135	V_{CE} = 2V, I_C = 150mA	40		250	
	: BD137, BD139		40		160	
V_{CE}(sat)	Collector-Emitter Saturation Voltage	I_C = 500mA, I_B = 50mA			0.5	V
V_{BE}(on)	Base-Emitter ON Voltage	V_{CE} = 2V, I_C = 0.5A			1	V

h_{FE} Classification

Classification	6	10	16
h_{FE3}	40 ~ 100	63 ~ 160	100 ~ 250

Rev. A, February 2000

Typical Characteristics

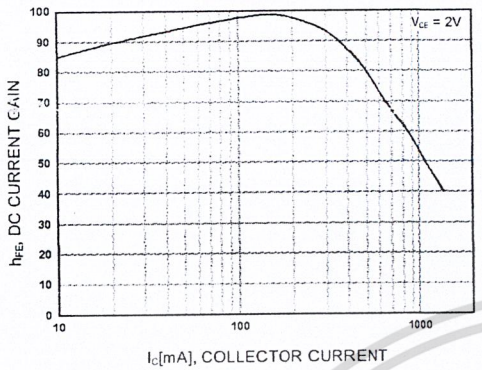


Figure 1. DC current Gain

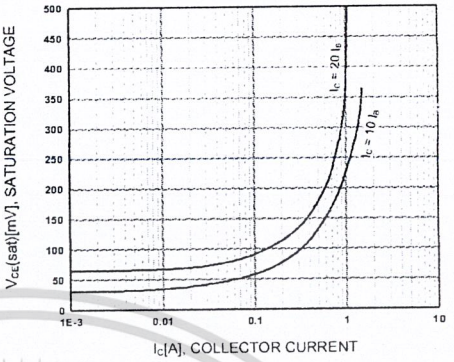


Figure 2. Collector-Emitter Saturation Voltage

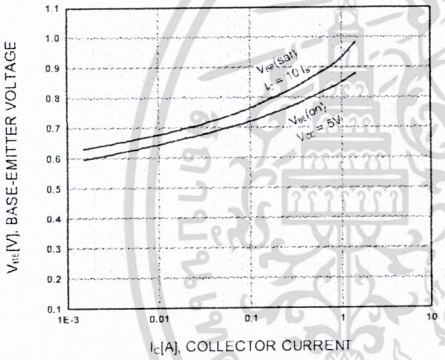


Figure 3. Base-Emitter Voltage

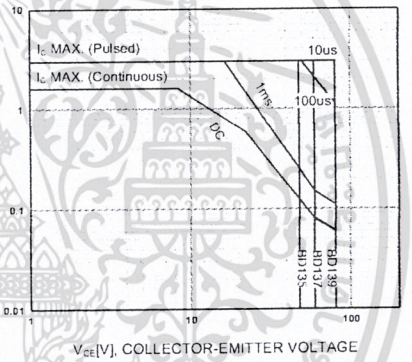


Figure 4. Safe Operating Area

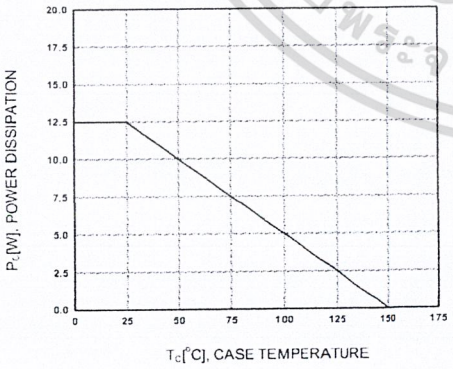
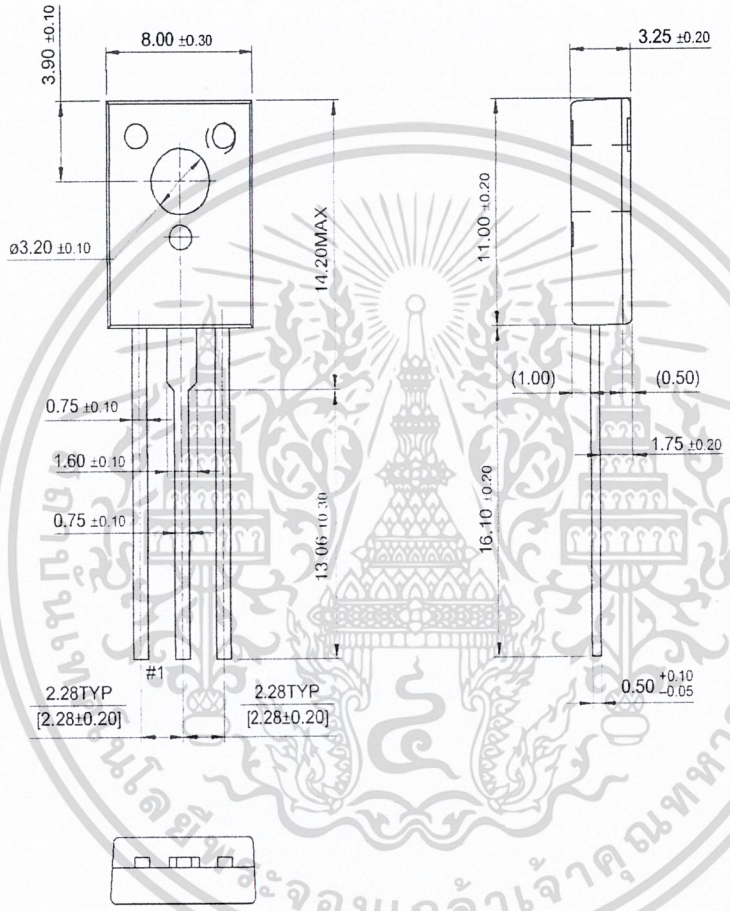


Figure 5. Power Derating

Package Demensions

TO-126



Dimensions in Millimeters

TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

ACEx™
Bottomless™
CoolFET™
CROSSVOLT™
E^2CMOS™
FACT™
FACT Quiet Series™
FAST®
FASTr™
GTO™
HiSeC™
ISOPLANAR™
MICROWIRE™
POP™
PowerTrench®
QFET™
QS™
Quiet Series™
SuperSOT™-3
SuperSOT™-6
SuperSOT™-8
SyncFET™
TinyLogic™
UHC™
VCX™

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR INTERNATIONAL.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

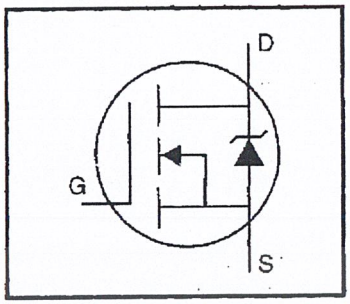
Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

International IOR Rectifier

IRF510

HEXFET® Power MOSFET

- Dynamic dv/dt Rating
- Repetitive Avalanche Rated
- 175°C Operating Temperature
- Fast Switching
- Ease of Paralleling
- Simple Drive Requirements



V_{DSS} = 100V

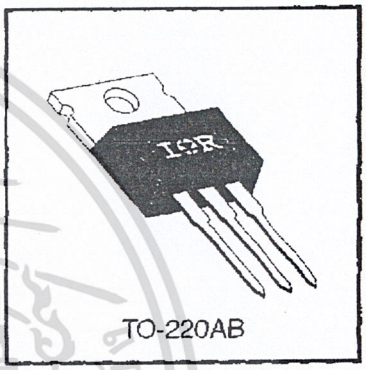
$R_{DS(on)}$ = 0.54Ω

I_D = 5.6A

Description

Third Generation HEXFETs from International Rectifier provide the designer with the best combination of fast switching, ruggedized device design, low on-resistance and cost-effectiveness.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.



TO-220AB

DATA SHEETS

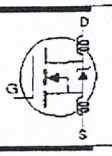
Absolute Maximum Ratings

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10 V	5.6	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10 V	4.0	
I_{DM}	Pulsed Drain Current ①	20	
P_D @ T_C = 25°C	Power Dissipation	43	W
	Linear Derating Factor	0.29	W/°C
V_{GS}	Gate-to-Source Voltage	±20	V
E_{AS}	Single Pulse Avalanche Energy ②	100	mJ
I_{AR}	Avalanche Current ①	5.6	A
E_{AR}	Repetitive Avalanche Energy ①	4.3	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.5	V/ns
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to +175	°C
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting Torque, 6-32 or M3 screw	10 lbf•in (1.1 N•m)	

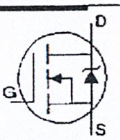
Thermal Resistance

	Parameter	Min.	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	—	—	3.5	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	—	0.50	—	
$R_{\theta JA}$	Junction-to-Ambient	—	—	62	

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Test Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	100	—	—	V	V_{GS}=0V, I_D= 250μA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	—	0.12	—	V/°C	Reference to 25°C, I_D= 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	—	—	0.54	Ω	V_{GS}=10V, I_D=3.4A ④
$V_{GS(th)}$	Gate Threshold Voltage	2.0	—	4.0	V	V_{DS}=V_{GS}, I_D= 250μA
g_{fs}	Forward Transconductance	1.3	—	—	S	V_{DS}=50V, I_D=3.4A ④
I_{DSS}	Drain-to-Source Leakage Current	—	—	25	μA	V_{DS}=100V, V_{GS}=0V
		—	—	250		V_{DS}=80V, V_{GS}=0V, T_J=150°C
I_{GSS}	Gate-to-Source Forward Leakage	—	—	100	nA	V_{GS}=20V
	Gate-to-Source Reverse Leakage	—	—	-100		V_{GS}=-20V
Q_g	Total Gate Charge	—	—	8.3	nC	I_D=5.6A
Q_{gs}	Gate-to-Source Charge	—	—	2.3		V_{DS}=80V
Q_{gd}	Gate-to-Drain ("Miller") Charge	—	—	3.8		V_{GS}=10V See Fig. 6 and 13 ④
$t_{d(on)}$	Turn-On Delay Time	—	6.9	—	ns	V_{DD}=50V
t_r	Rise Time	—	16	—		I_D=5.6A
$t_{d(off)}$	Turn-Off Delay Time	—	15	—		R_G=24Ω
t_f	Fall Time	—	9.4	—		R_D=8.4Ω See Figure 10 ④
L_D	Internal Drain Inductance	—	4.5	—	nH	Between lead, 6 mm (0.25in.) from package and center of die contact 
L_S	Internal Source Inductance	—	7.5	—		
C_{iss}	Input Capacitance	—	180	—	pF	V_{GS}=0V
C_{oss}	Output Capacitance	—	81	—		V_{DS}=25V
C_{rss}	Reverse Transfer Capacitance	—	15	—		f=1.0MHz See Figure 5

Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Test Conditions
I_S	Continuous Source Current (Body Diode)	—	—	5.6	A	MOSFET symbol showing the integral reverse p-n junction diode. 
I_{SM}	Pulsed Source Current (Body Diode) ①	—	—	20		
V_{SD}	Diode Forward Voltage	—	—	2.5	V	T_J=25°C, I_S=5.6A, V_{GS}=0V ④
t_{rr}	Reverse Recovery Time	—	100	200	ns	T_J=25°C, I_F=5.6A
Q_{rr}	Reverse Recovery Charge	—	0.44	0.88	μC	di/dt=100A/μs ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is neglegible (turn-on is dominated by L_S+L_D)				

Notes:

① Repetitive rating; pulse width limited by max. junction temperature (See Figure 11)

② V_{DD}=25V, starting T_J=25°C, L=4.8mH R_G=25Ω, I_{AS}=5.6A (See Figure 12)

③ I_{SD}≤5.6A, di/dt≤75A/μs, V_{DD}≤$V_{(BR)DSS}$, T_J≤175°C

④ Pulse width ≤ 300 μs; duty cycle ≤2%.

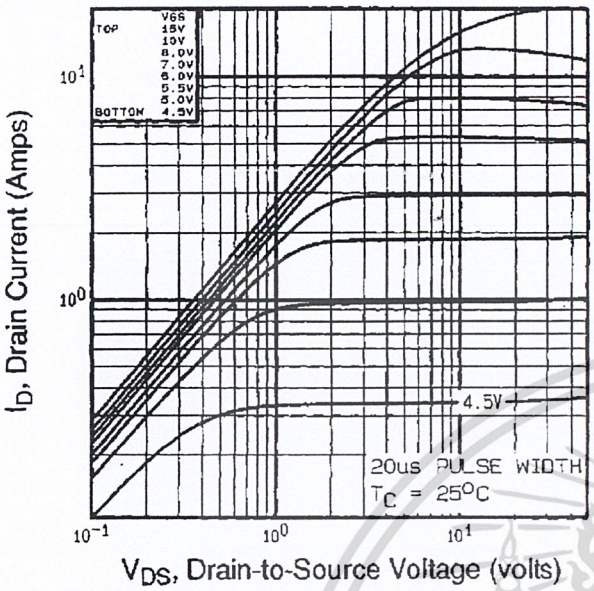


Fig 1. Typical Output Characteristics, T_C=25°C

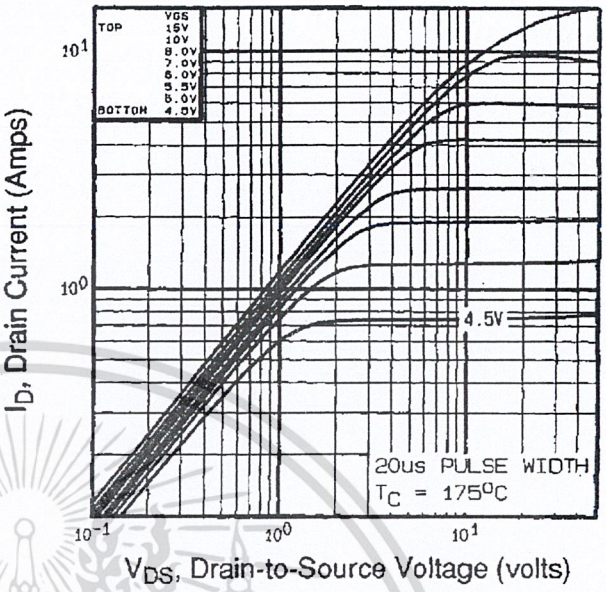


Fig 2. Typical Output Characteristics, T_C=175°C

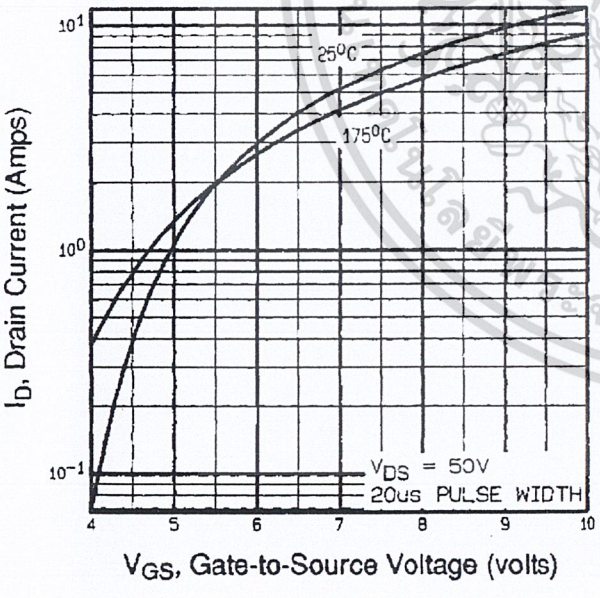


Fig 3. Typical Transfer Characteristics

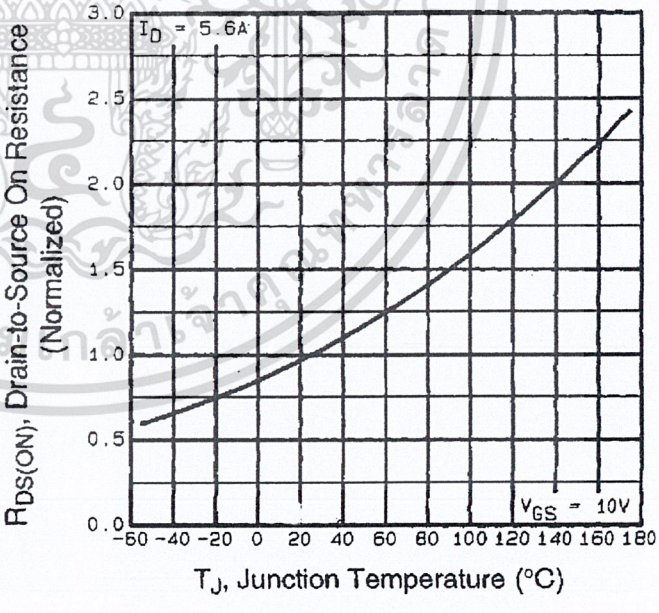


Fig 4. Normalized On-Resistance Vs. Temperature

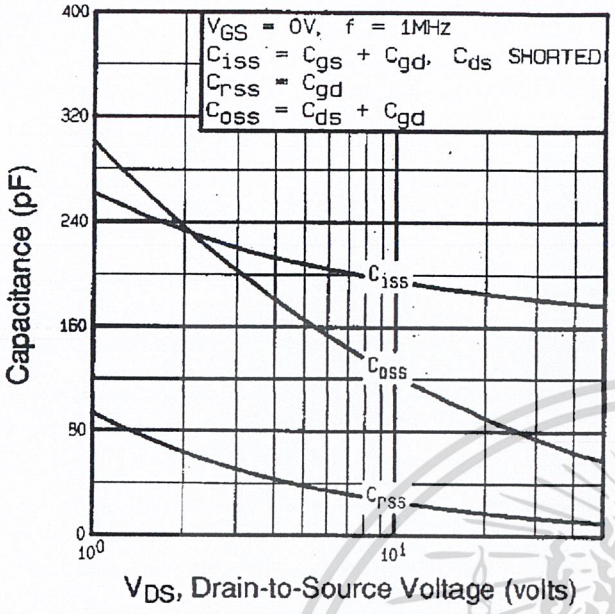


Fig 5. Typical Capacitance Vs. Drain-to-Source Voltage

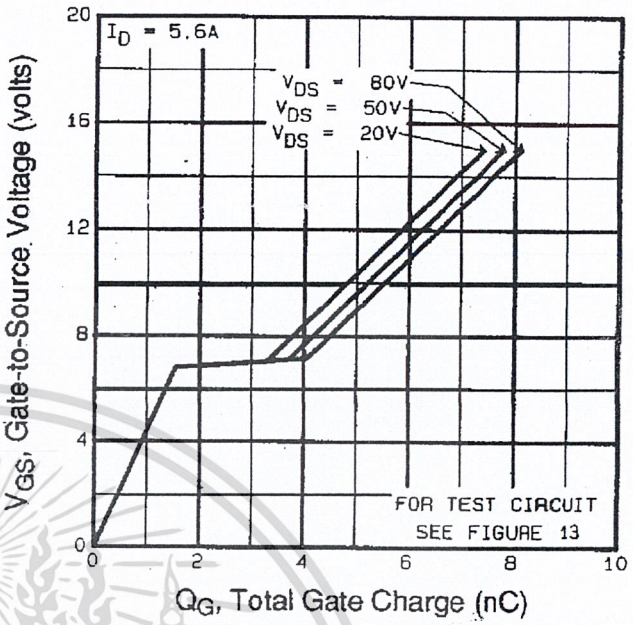


Fig 6. Typical Gate Charge Vs. Gate-to-Source Voltage

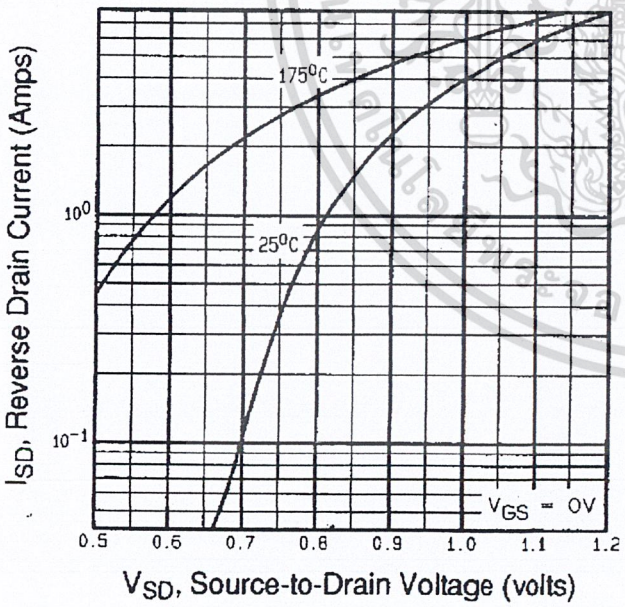


Fig 7. Typical Source-Drain Diode Forward Voltage

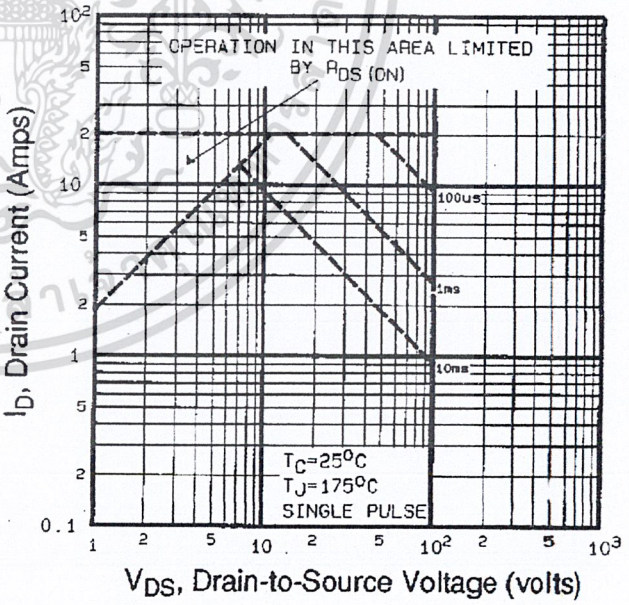


Fig 8. Maximum Safe Operating Area

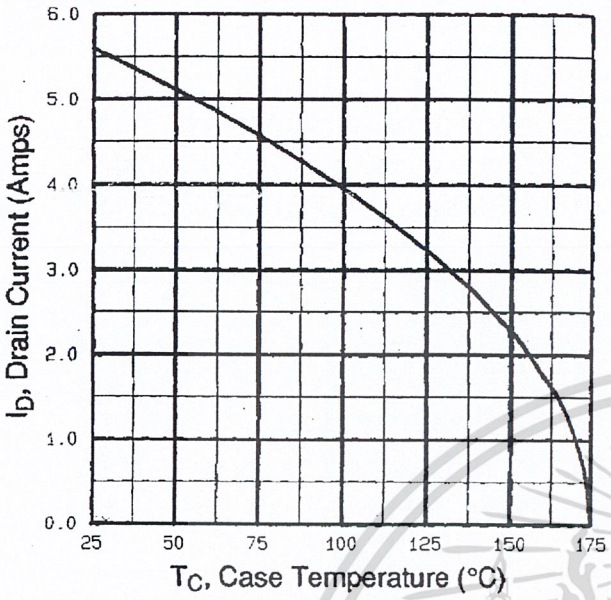


Fig 9. Maximum Drain Current Vs. Case Temperature

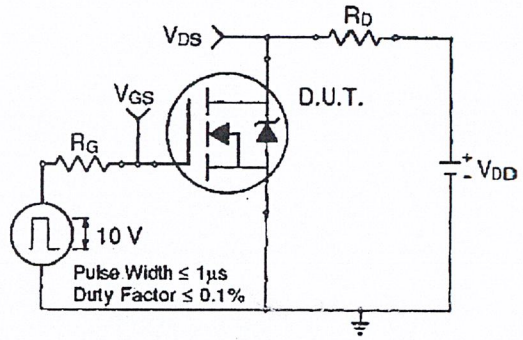


Fig 10a. Switching Time Test Circuit

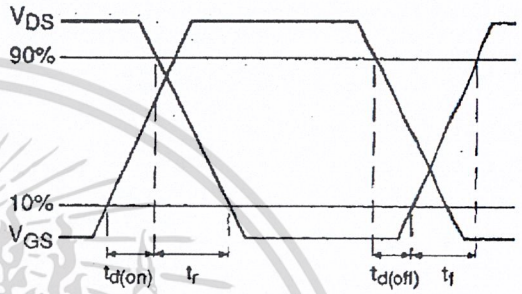


Fig 10b. Switching Time Waveforms

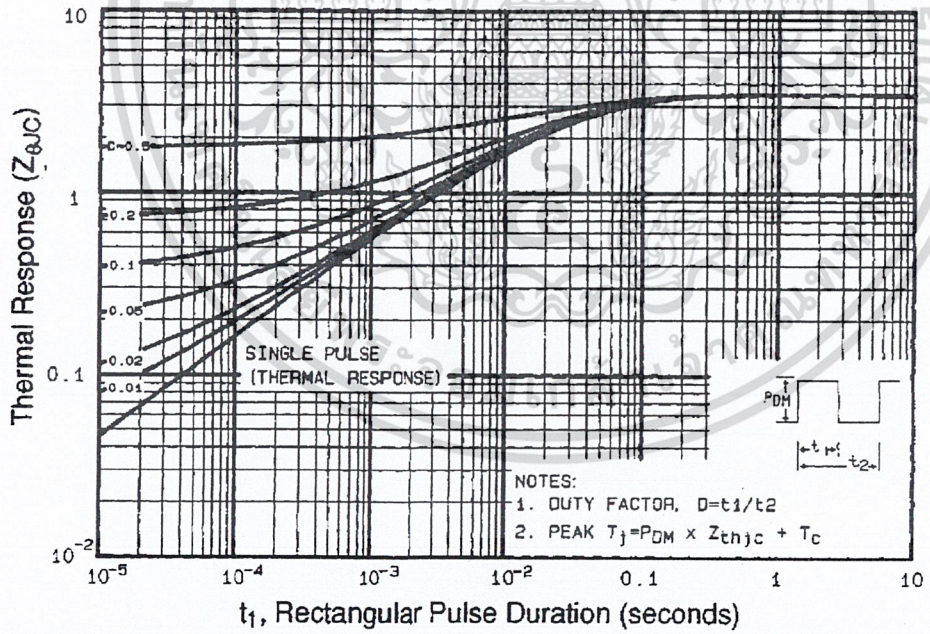


Fig 11. Maximum Effective Transient Thermal Impedance, Junction-to-Case

DATA SHEETS

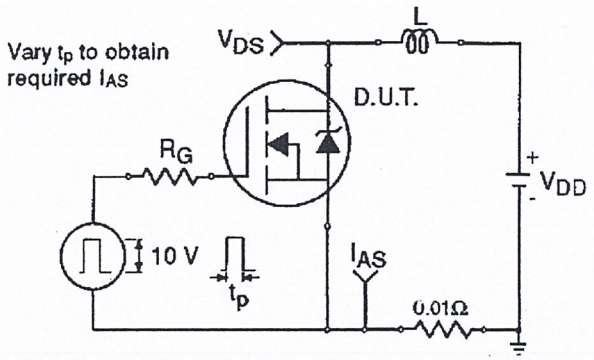


Fig 12a. Unclamped Inductive Test Circuit

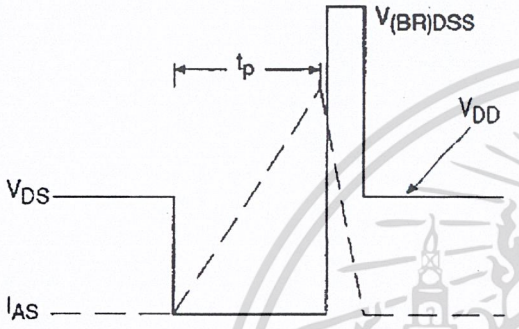


Fig 12b. Unclamped Inductive Waveforms

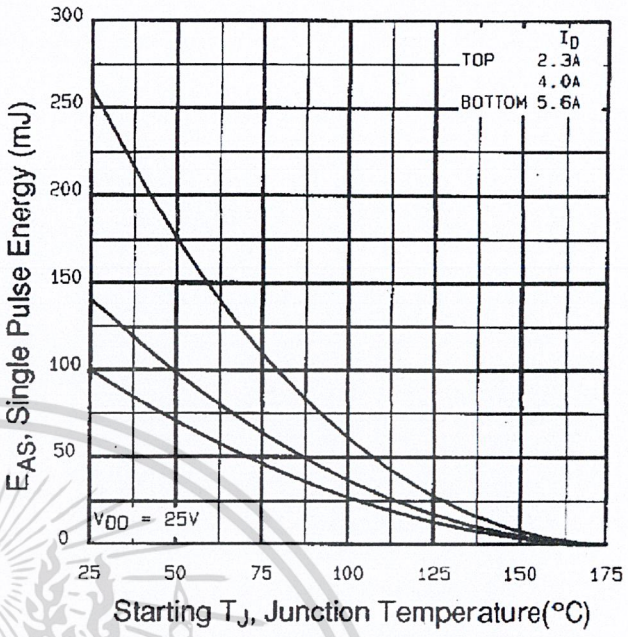


Fig 12c. Maximum Avalanche Energy Vs. Drain Current

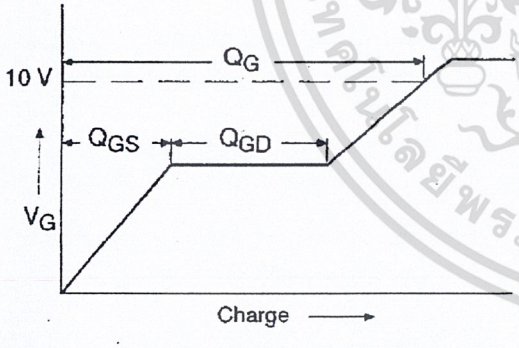


Fig 13a. Basic Gate Charge Waveform

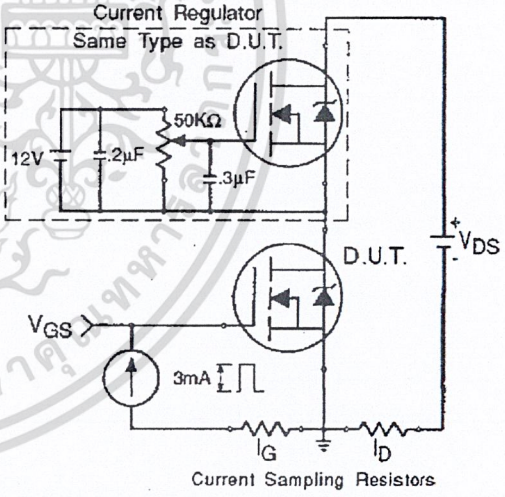


Fig 13b. Gate Charge Test Circuit

Appendix A: Figure 14, Peak Diode Recovery dv/dt Test Circuit – See page 1505

Appendix B: Package Outline Mechanical Drawing – See page 1509

Appendix C: Part Marking Information – See page 1516

Appendix E: Optional Leadforms – See page 1525

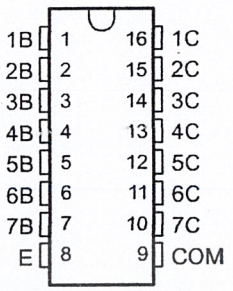
International
IOR Rectifier

SLRS027 – D2624, DECEMBER 1976 – REVISED APRIL 1993

HIGH-VOLTAGE HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

- **500-mA Rated Collector Current (Single Output)**
- **High-Voltage Outputs . . . 50 V**
- **Output Clamp Diodes**
- **Inputs Compatible With Various Types of Logic**
- **Relay Driver Applications**
- **Designed to Be Interchangeable With Sprague ULN2001A Series**

D OR N PACKAGE
(TOP VIEW)

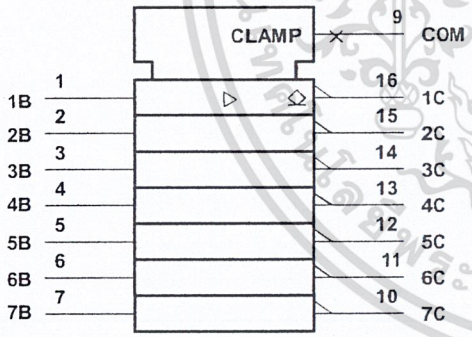


description

The ULN2001A, ULN2002A, ULN2003A, and ULN2004A are monolithic high-voltage, high-current Darlington transistor arrays. Each consists of seven npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of a single Darlington pair is 500 mA. The Darlington pairs may be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. For 100-V (otherwise interchangeable) versions, see the SN75465 through SN75469.

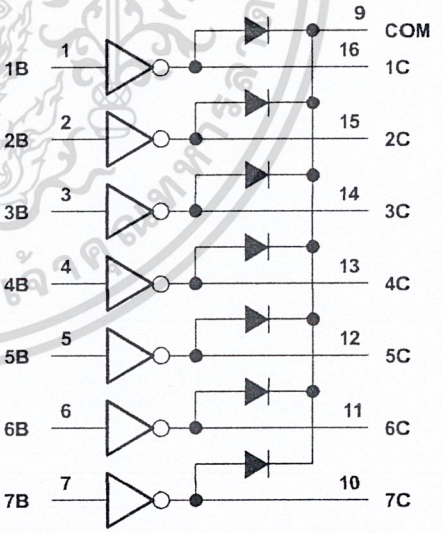
The ULN2001A is a general-purpose array and can be used with TTL, P-MOS, CMOS, and other MOS technologies. The ULN2002A is specifically designed for use with 14- to 25-V P-MOS devices. Each input of this device has a zener diode and resistor in series to control the input current to a safe limit. The ULN2003A has a 2.7-kΩ series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices. The ULN2004A has a 10.5-kΩ series base resistor to allow its operation directly from CMOS or P-MOS devices that use supply voltages of 6 to 15 V. The required input current of the ULN2004A is below that of the ULN2003A, and the required voltage is less than that required by the ULN2002A.

logic symbol†

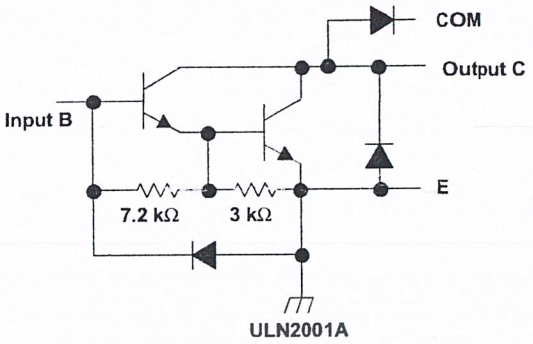


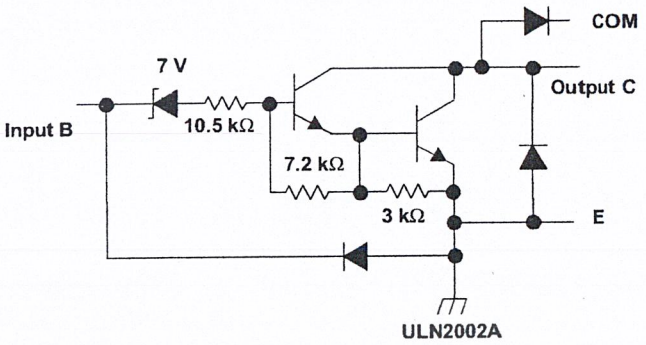
† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

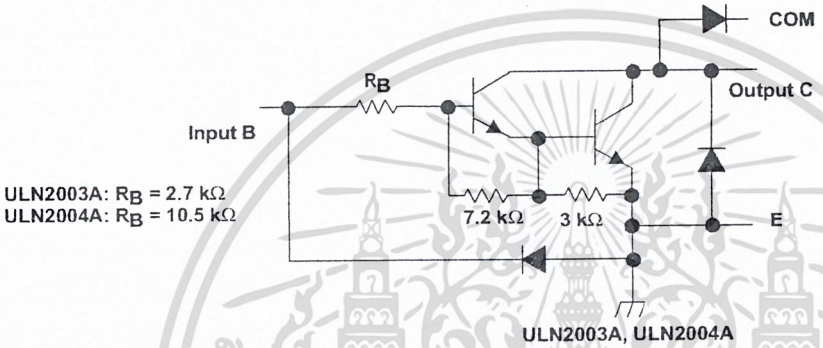
logic diagram



schematics (each Darlington pair)







All resistor values shown are nominal.

absolute maximum ratings at 25°C free-air temperature (unless otherwise noted)

Collector-emitter voltage	50 V
Input voltage, V_I (see Note 1)	30 V
Peak collector current (see Figures 14 and 15)	500 mA
Output clamp current, I_{OK}	500 mA
Total emitter-terminal current	−2.5 A
Continuous total power dissipation	See Dissipation Rating Table
Operating free-air temperature range	−20°C to 85°C
Storage temperature range	−65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C

NOTE 1: All voltage values are with respect to the emitter/substrate terminal E, unless otherwise noted.

DISSIPATION RATING TABLE

PACKAGE	T_A = 25°C POWER RATING	DERATING FACTOR ABOVE T_A = 25°C	T_A = 85°C POWER RATING
D	950 mW	7.6 mW/°C	494 mW
N	1150 mW	9.2 mW/°C	598 mW

electrical characteristics, T_A = 25°C (unless otherwise noted)

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2001A			ULN2002A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	6	V_{CE} = 2 V,	I_C = 300 mA						13	V
$V_{CE(sat)}$	Collector-emitter saturation voltage	5	I_I = 250 µA,	I_C = 100 mA		0.9	1.1		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.3		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.6		1.2	1.6	
V_F	Clamp forward voltage	8	I_F = 350 mA			1.7	2		1.7	2	V
I_{CEX}	Collector cutoff current	1	V_{CE} = 50 V,	I_I = 0			50			50	µA
		2	V_{CE} = 50 V, T_A = 70°C	I_I = 0			100			100	
				V_I = 6 V						500	
$I_{I(off)}$	Off-state input current	3	V_{CE} = 50 V, T_A = 70°C	I_C = 500 µA,	50	65		50	65		µA
I_I	Input current	4	V_I = 17 V						0.82	1.25	mA
I_R	Clamp reverse current	7	V_R = 50 V,	T_A = 70°C			100			100	µA
h_{FE}	Static forward current transfer ratio	5	V_{CE} = 2 V,	I_C = 350 mA	1000						
I_R	Clamp reverse current	7	V_R = 50 V				50			50	µA
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25		15	25	pF

electrical characteristics, T_A = 25°C (unless otherwise noted)

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2003A			ULN2004A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	6	V_{CE} = 2 V	I_C = 125 mA						5	V
				I_C = 200 mA			2.4			6	
				I_C = 250 mA			2.7				
				I_C = 275 mA						7	
				I_C = 300 mA			3				
				I_C = 350 mA						8	
$V_{CE(sat)}$	Collector-emitter saturation voltage	5	I_I = 250 µA,	I_C = 100 mA		0.9	1.1		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.3		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.6		1.2	1.6	
I_{CEX}	Collector cutoff current	1	V_{CE} = 50 V,	I_I = 0			50			50	µA
		2	V_{CE} = 50 V, T_A = 70°C	I_I = 0			100			100	
				V_I = 1 V						500	
V_F	Clamp forward voltage	8	I_F = 350 mA			1.7	2		1.7	2	V
$I_{I(off)}$	Off-state input current	3	V_{CE} = 50 V, T_A = 70°C	I_C = 500 µA,	50	65		50	65		µA
I_I	Input current	4	V_I = 3.85 V			0.93	1.35				mA
			V_I = 5 V						0.35	0.5	
			V_I = 12 V						1	1.45	
I_R	Clamp reverse current	7	V_R = 50 V				50			50	µA
			V_R = 50 V,	T_A = 70°C			100			100	
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25		15	25	pF

switching characteristics, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	See Figure 9		0.25	1	µs
t_{PHL}	Propagation delay time, high-to-low-level output			0.25	1	µs
V_{OH}	High-level output voltage after switching	V_S = 50 V, I_O = 300 mA, See Figure 10	V_S–20			mV

PARAMETER MEASUREMENT INFORMATION

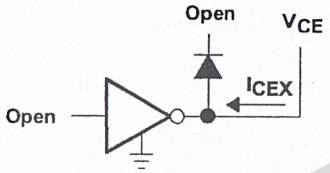


Figure 1. I_{CEX} Test Circuit

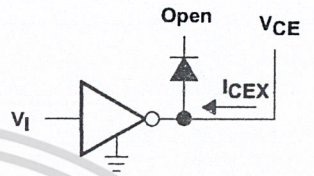


Figure 2. I_{CEX} Test Circuit

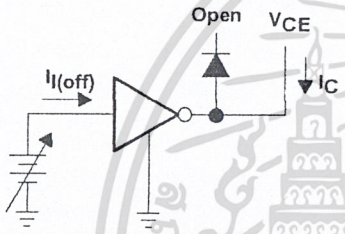


Figure 3. $I_{I(off)}$ Test Circuit

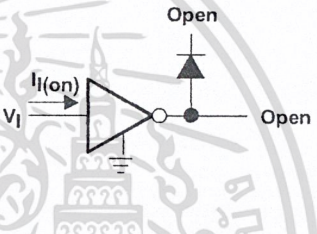
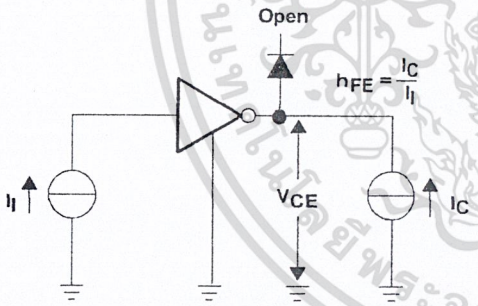


Figure 4. I_I Test Circuit



NOTE: I_I is fixed for measuring $V_{CE(sat)}$, variable for measuring h_{FE}.

Figure 5. h_{FE}, $V_{CE(sat)}$ Test Circuit

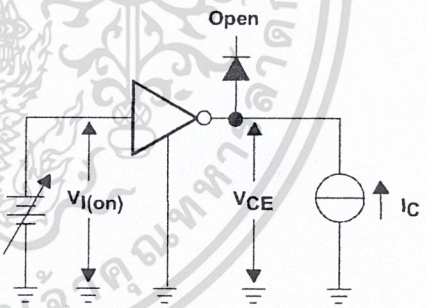


Figure 6. $V_{I(on)}$ Test Circuit

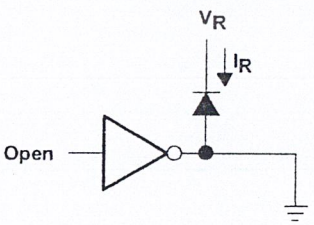


Figure 7. I_R Test Circuit

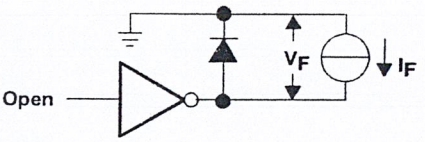


Figure 8. V_F Test Circuit

เอกสารอ้างอิง

ดุสิต เครืองาม, **สิ่งประดิษฐ์ออปโตอิเล็กทรอนิกส์ ฟิสิกส์ เทคโนโลยี่ และการใช้ งาน เล่ม 1**, หน้า 526 – 530, จุฬาลงกรณ์มหาวิทยาลัย, ปี 2542

สุทธิศักดิ์ พงศ์ธนาพานิช, Visual BASIC 6.0 Professional, บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน), ปี 2542

Warren W. Gay, Linux Socket Programming by Example, pp 200 – 250, Indianapolis, c2000