การควบคุมเครื่องใช้ไฟฟ้าผ่านเครือข่ายเวิลด์ไวด์เว็บ

Appliances Remote Controlling through World Wide Web Network

โดย

นางสาวปิยธิดา ดวงพิกุล 36014265

นายพรเทพ ชัยกิจวัฒนะ 36014281

ปริญญานิพนธ์สำหรับวิศวกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีการวัดคุมอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ภาควิชา : เทคโนโลยีการวัดคุมอุตสาหกรรม

คณะ : วิศวกรรมศาสตร์

สถาบัน : เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง : การควบคุมเครื่องใช้ไฟฟ้าผ่านเครือข่ายเวิลด์ไวด์เว็บ

โดย : นางสาวปิยธิดา ดวงพิกุล 36014265

นายพรเทพ ชัยกิจวัฒนะ 36014281

..... อาจารย์ที่ปรึกษา

(อาจารย์ทรงชัย วีระทวีมาศ)

บทคัดย่อ

เวิลด์ไวด์เว็บกำลังได้รับความนิยมเป็นอย่างสูง เนื่องจากความง่ายในการใช้งาน, ความสวยงามของภาพกราฟิก และความสะดวกในการเข้าถึงแหล่งข้อมูลขนาดใหญ่อย่างอินเตอร์เนต เวิลด์ไวด์เว็บครอบคลุมพื้นที่ทั้งโลก เราอาจควบคุมระบบหนึ่งๆ ได้จากที่ใดๆ การนำเครือข่ายนี้มาใช้ประโยชน์ในการควบคุมรีเลย์จะเป็นแนวทางในการควบคุมที่ซับซ้อนกว่าได้ต่อไป

abstract

World wide web is now very popular, because of its ease of use, nice graphics and convinient access to any Internet resource. Since it's worldwide, we may control any systems from anywhere in the world. Applying this remote relay switching project should result in many more sophisticated control systems.

สารบัญ

บทนำ

ช่วงปี 1970 คอมพิวเตอร์ส่วนบุคคลยังไม่ถือกำเนิดขึ้น คอมพิวเตอร์ที่ใช้กันจะเป็นแบบเมนเฟรม ซึ่งผู้ใช้หลายคนจะติดต่อผ่านทางเทอร์มินัลของตนเอง ทรัพยากรของระบบและการประมวลผลของโปรเซสเซอร์จะถูกจัดสรรให้ผู้ใช้ทั้งหมดที่ติดต่อเข้ามา

จากนั้นคอมพิวเตอร์ส่วนบุคคลได้ปฏิวัติรูปแบบการเชื่อมต่อเช่นนี้โดยสิ้นเชิง เพราะคอมพิวเตอร์ส่วนบุคคลจะบรรจุความสามารถในการประมวลผลทั้งหมดเท่าที่ผู้ใช้แต่ละคนต้องการ ไม่มีความจำเป็นสำหรับการแบ่งเวลาบนเมนเฟรมอีกต่อไป แต่การสื่อสารระหว่างผู้ใช้ไม่สะดวกนัก ทางแก้ปัญหาคืออีเทอร์เนต (Ethernet) แต่อีเทอร์เนตยังมีข้อจำกัด มันเพียงเชื่อมคอมพิวเตอร์ในไซต์เดียวกันเท่านั้น

และในท้ายที่สุดทั้งหมดถูกเชื่อมเข้าด้วยกันเป็นอินเตอร์เนต เราอาจมองอินเตอร์เนตเป็นระบบปฏิบัติการ และเวิลด์ไวด์เว็บเป็น command-line interpreter เว็บช่วยให้การใช้งานเทลเนต (telnet), FTP (File Transfer Protocol), HTTP (Hypertext Transfer Protocol) หรือโกเฟอร์ (Gopher) เป็นไปได้อย่างสะดวกสบาย

โครงงานนี้ถูกสร้างขึ้นเพื่อใช้ประโยชน์จากเว็บ ซึ่งมีขนาดครอบคลุมทั้งโลกในการควบคุมรีเลย์จากระยะไกล อาจประยุกต์โครงงานนี้ในการควบคุมแบบอื่นๆได้อีกด้วย

วัตถุประสงค์

1.ศึกษาการรับและส่งข้อมูลผ่านเครือข่ายเวิลด์ไวด์เว็บ

2.ศึกษาการนำข้อมูลที่รับได้ไปควบคุมรีเลย์เพื่อปิดเปิดเครื่องไฟฟ้า

3.ศึกษาวิธีการส่งข้อมูลสถานะของรีเลย์กลับเพื่อแจ้งแก่ผู้ใช้ซึ่งอยู่ห่างออกไป

ขอบเขตของโครงงาน

สร้างระบบซึ่งทำหน้าที่รับข้อมูลจากคอมพิวเตอร์ในการควบคุมรีเลย์ และออกแบบโปรแกรมสำหรับควบคุมระบบนั้นได้จากระยะไกลโดยผ่านทางเครือข่ายเวิลด์ไวด์เว็บ

บทที่ 1
ลักษณะของระบบโดยรวม

เป้าหมายของโครงงานนี้คือการควบคุมรีเลย์สำหรับเครื่องไฟฟ้าผ่านเครือข่ายเวิลด์ไวด์เว็บ

สิ่งที่ระบบนี้ต้องการ (system requirements) ได้แก่

-เว็บเซอร์ฟเวอร์ (web server) ซึ่งมีพอร์ตอนุกรมแบบ RS-232 DB-25

-ระบบปฏิบัติการ Windows'95 หรือ Window NT เวอร์ชั่น 4.0 ขึ้นไป

-จาวาเวอร์ชวลแมชชีน (Java Virtual Machine)เวอร์ชั่น 1.1ขึ้นไป

ทางฝั่งผู้ใช้ซึ่งควบคุมจากระยะไกลจะใช้โปรแกรมเบราเซอร์ (browser) ซึ่งสนับสนุนจาวา เช่น Netscape Navigator , Microsoft Internet Explorer หรือ HotJava! สำหรับส่งข้อมูลให้เว็บเซอร์ฟเวอร์ต่อไป

เราจะเขียนโปรแกรมจาวาแอพแพลท (Applet) แล้วฝากแอพเพลทนี้ไว้กับเว็บเพจบนเครื่องเซิร์ฟเวอร์ เมื่อผู้ใช้ซึ่งอยู่ห่างออกไปใช้โปรแกรมเบราเซอร์ติดต่อเข้ามาด้วยไฮเปอร์เท็กซ์ทรานเฟอร์โปรโตคอล (HyperText Transfer Protocol : HTTP) เซอร์ฟเวอร์จะโหลดแอพแพลทนี้ให้กับผู้ใช้พร้อมสถานะปัจจุบันของเครื่องไฟฟ้าซึ่งถูกควบคุม เฉพาะผู้ใช้ซึ่งทราบรหัสผ่านเท่านั้นจึงจะสามารถควบคุมเครื่องไฟฟ้าเหล่านั้นได้โดยการปิดเปิดสวิทช์จำลองบนตัวแอพแพลท แอพแพลทจะส่งข้อมูลสถานะใหม่กลับมาที่เว็บเซอร์ฟเวอร์ เว็บเซอร์ฟเวอร์จะ บันทึกข้อมูลสถานะใหม่นั้นลงในไฟล์และส่งต่อให้ฮาร์ดแวร์ซึ่งต่ออยู่กับพอร์ตอนุกรม com2

ขั้นตอนการติดต่อระหว่าง HOST กับ TERMINAL

ใช้ browser ติดต่อ

1. HOST ⟵ TERMINAL

ผ่าน HTTP

TERMINAL

download homepage

2. HOST ⟶ จะได้รับจาวาแอพเพลท แสดงสถานะของอุปกรณ์ไฟฟ้าขณะนั้น พร้อมที่จะควบคุมทันที

ส่งข้อมูลผ่านเครือข่าย TERMINAL

3. HOST ⟵

พร้อมป้อน port number ใช้แอพเพลทที่ได้รับจากโฮมเพจกำหนดสถานะรีเลย์ให้ Host

4. HOST ⟶ TERMINAL

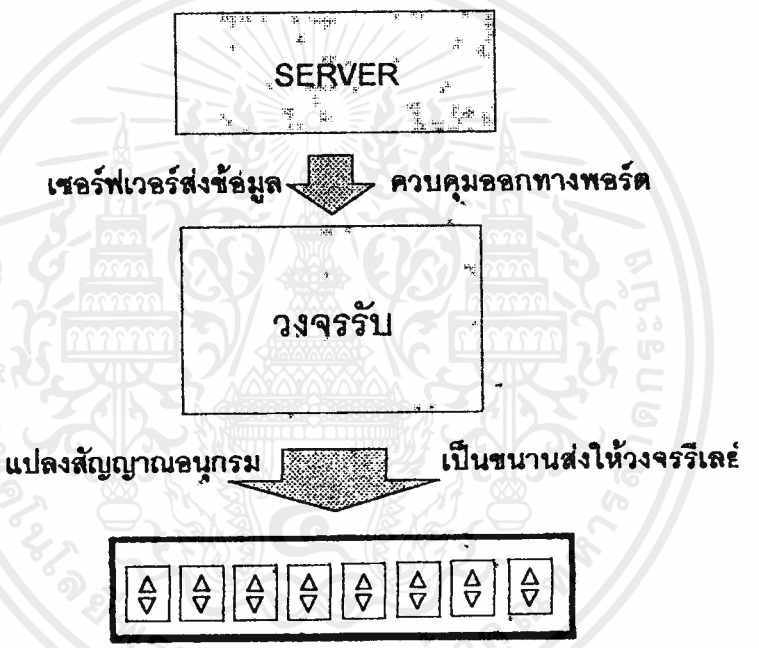
เมื่อ port number ถูกต้อง Host จึงเปลี่ยนแปลงสถานะของรีเลย์ตามข้อมูลที่ส่งมา

ฮาร์ดแวร์จะรับข้อมูลสถานะรีเลย์ขนาดหนึ่งไบต์จากพอร์ตอนุกรม com2 แต่ละบิตเป็นตัวแทนสถานะรีเลย์แต่ละตัว ฮาร์ดแวร์ประกอบด้วยวงจรหลักเพียงส่วน คือ

1.วงจรรับข้อมูลจากพอร์ตอนุกรม แปลงข้อมูลเป็นขนานเพื่อส่งต่อไปยังชุดวงจรควบคุมรีเลย์

2.ชุดวงจรรีเลย์เพื่อปิด-เปิดเครื่องไฟฟ้าและแหล่งจ่ายไฟ

ฮาร์ดแวร์และโปรแกรมควบคุมจะทำงานร่วมกันได้ด้วยแฮนด์เชคกิ้งในพอร์ตอนุกรม การทำงานของฮาร์ดแวร์เริ่มขึ้นเมื่อเซอร์ฟเวอร์ได้รับคำสั่งควบคุมรีเลย์จากผู้ใช้ เซอร์ฟเวอร์จะส่งสัญญาณรีเซทวงจรรับข้อมูล เพื่อให้วงจรรับข้อมูลพร้อมที่จะรับข้อมูลใหม่ จากนั้นเซอร์ฟเวอร์จะส่งข้อมูลใหม่ให้วงจรรับ เอาต์พุตจากวงจรรับข้อมูลเป็นแบบขนาน 8 บิต เรานำเอาต์พุตนี้ไปใช้ควบคุมรีเลย์สำหรับเครื่องไฟฟ้าแต่ละเครื่องได้



บทที่ 2
มาตรฐาน RS-232

มาตรฐาน RS-232 เป็นรูปแบบการรับส่งข้อมูลอนุกรมระหว่างอุปกรณ์แบบหนึ่ง ตามมาตรฐานนี้เราแบ่งอุปกรณ์เป็นสองประเภท ได้แก่ Data Terminal Equipment (DTE) และ Data Communication Equipment (DCE) ตามลักษณะการจัดตำแหน่งขาของคอนเนกเตอร์ คอมพิวเตอร์ส่วนใหญ่ในปัจจุบันเป็นอุปกรณ์ประเภท DTE

การจัดตำแหน่งขาคอนเนกเตอร์ของพอร์ตอนุกรมตามมาตรฐาน RS-232 เป็นดังนี้

RS-232 SERIAL PORT CONNECTOR (DTE DEVICE)

Pin Number	*Definition*	*Signal*	*Direction*
1	Protective ground		
2	Transmitted data	TD	Out
3	Received data	RD	In
4	Request to send	RTS	Out
5	Clear to send	CTS	In
6	Data set ready	DSR	In
7	Signal ground	SG	
8	Received line signal detector	DCD	In
20	Data terminal ready	DTR	Out
22	Ring indicator	RI	In

แรงดันไฟฟ้าบวก (สถานะ space) มีค่าอยู่ระหว่าง +5 ถึง +15 โวลต์สำหรับเอาต์พุต และ ระหว่าง +3 ถึง +15 โวลต์สำหรับอินพุต ความแตกต่างมีไว้เผื่อกรณีที่แรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณ ในทำนองเดียวกันแรงดันไฟฟ้าลบ (สถานะ mark) ถูกกำหนดไว้ระหว่าง -5 ถึง -15 โวลต์สำหรับเอาต์พุต และ -3 ถึง -15 โวลต์สำหรับอินพุต

บนสายข้อมูล เช่น สาย 2 และ 3 แรงดันไฟฟ้าบวกแสดงถึงค่าลอจิก 0 และแรงดันไฟฟ้าลบแสดงค่าลอจิก 1 บนสายแฮนด์เช็คกิ้ง เช่น DTR และ DSR แรงดันไฟฟ้าบวกแสดงว่าส่งข้อมูลได้ ส่วนแรงดันไฟฟ้าลบหมายถึงหยุดส่งข้อมูล

สิ่งต่อไปที่เราจะขาดไม่ได้คือ วิธีการควบคุมการทำงานของอุปกรณ์สองตัวให้สัมพันธ์กันในการรับส่งข้อมูลในช่วงเวลาและใช้สัญญาณที่เหมาะสม ซึ่งก็คือกระบวนการแฮนด์เชคกิ้งนั่นเอง

แฮนด์เชคกิ้งแบ่งได้เป็นสองประเภท คือ

-แฮนด์เชคกิ้งทางซอฟท์แวร์ เป็นวิธีการหนึ่งในการควบคุมการทำงานของอุปกรณ์รับข้อมูลโดยส่งผ่านสัญญาณควบคุมไปพร้อมกับตัวข้อมูลที่ต้องการส่ง ตัวอย่างเช่น ในการพิมพ์ข้อความทางเครื่องพิมพ์ คอมพิวเตอร์จะส่งข้อมูลไปทีละบรรทัด และเมื่อสิ้นสุดแต่ละบรรทัด มันจะแทรกอักษรควบคุมแสดงว่าสิ้นสุดบรรทัดเพื่อแจ้งเครื่องพิมพ์ว่าขณะนี้คอมพิวเตอร์กำลังรอสัญญาณตอบกลับอยู่ก่อนจะส่งข้อมูลบรรทัดถัดไป และเมื่อเครื่องพิมพ์รับข้อมูลเข้ามาและพิมพ์ไปจนจบบรรทัด ก็จะส่งสัญญาณไปบอกคอมพิวเตอร์ว่ามันพร้อมจะรับข้อความบรรทัดใหม่แล้ว ซึ่งเพียงเท่านี้เราก็สามารถควบคุมการทำงานของเครื่องพิมพ์ได้แล้ว แต่ในความเป็นจริงแล้ว เครื่องพิมพ์บางรุ่นไม่สามารถแยกอักษรควบคุมออกจากตัวข้อความได้

-ในทางตรงกันข้าม แฮนด์เชคกิ้งทางฮาร์ดแวร์สามารถควบคุมเครื่องพิมพ์ได้ตั้งแต่ระดับฮาร์ดแวร์ โดยใช้การเปลี่ยนระดับแรงดันในสายสัญญาณควบคุมเป็นตัวระงับไม่ให้คอมพิวเตอร์ส่งข้อมูลเพิ่มเข้ามาอีก ซึ่งเป็นการหลีกเลี่ยงการใช้รหัสหรือโปรแกรม แต่แฮนด์เชคกิ้งทางฮาร์ดแวร์นั้นมีข้อจำกัด คือจำเป็นต้องมีสายสัญญาณควบ

คุมต่างหากสำหรับงานนี้โดยเฉพาะ ทำให้วิธีนี้ไม่เหมาะที่จะนำมาใช้ในการอินเตอร์เฟสกับโมเด็ม

เราจะใช้ฮาร์ดแวร์แฮนด์เชคกิ้งสำหรับโครงงานนี้

สายที่เราจะนำมาใช้งานจริงได้แก่สายข้อมูลออก TxD(2), สายข้อมูลเข้า สายแฮนด์เช็คกิ้ง RTS(4), CTS(5), และสายกราวนด์ SG(7) เนื่องจากคอสต้องการสัญญาณแฮนด์เช็คกิ้งทุกกรณี ในขณะที่ฮาร์ดแวร์ซึ่งเราจะสร้างนั้นไม่มีสัญญาณแฮนด์เชคกิ้งจริงๆออกมา จึงจำเป็นต้องอินพุตบางขาด้วยเอาต์พุตจากตัวพอร์ตเองเป็นการชดเชย

สำหรับโครงงานนี้เราเลือกใช้พอร์ต com2 โดยกำหนดหน้าที่ของแต่ละขาดังนี้

-TxD(2) เป็นขาส่งข้อมูลออกจากคอมพิวเตอร์

-RTS(4) เอาต์พุตเอนกประสงค์ เราใช้เพื่อรีเซทเอาต์พุตของวงจรรับข้อมูล และเป็นแฮนด์เชคกิ้งหลอกสำหรับ CTS(5)

-CTS(5) ต้องการสัญญาณแฮนด์เชคกิ้ง เมื่อจะส่งข้อมูลออก

-SG(7) เป็นกราวนด์ของสัญญาณออก ต้องถูกเชื่อมเข้ากับกราวนด์ของฮาร์ดแวร์

เนื่องจากวงจรรับ-ส่งข้อมูลอนุกรมใช้สัญญาณภายในแบบ TTL จึงจำเป็นต้องแปลงสัญญาณแบบ RS-232 ไปเป็นแบบ TTL ด้วยไอซี MAX232 ซึ่งมีลักษณะวงจรและการจัดขาเป็นดังรูปที่ 1

MAX232 มีข้อดีคือมีทั้ง driver และ receiver ในตัวเอง และใช้โวลเตจเพียงระดับเดียว คือ 5 โวลต์ การนำเอา MAX232 ไปใช้นั้นต้องต่อตัวเก็บประจุเพิ่มเข้าไปอีก เล็กน้อยตามรูปที่ 2

สัญญาณแบบ RS-232 ที่จะถูกแปลงเป็น TTL ได้แก่

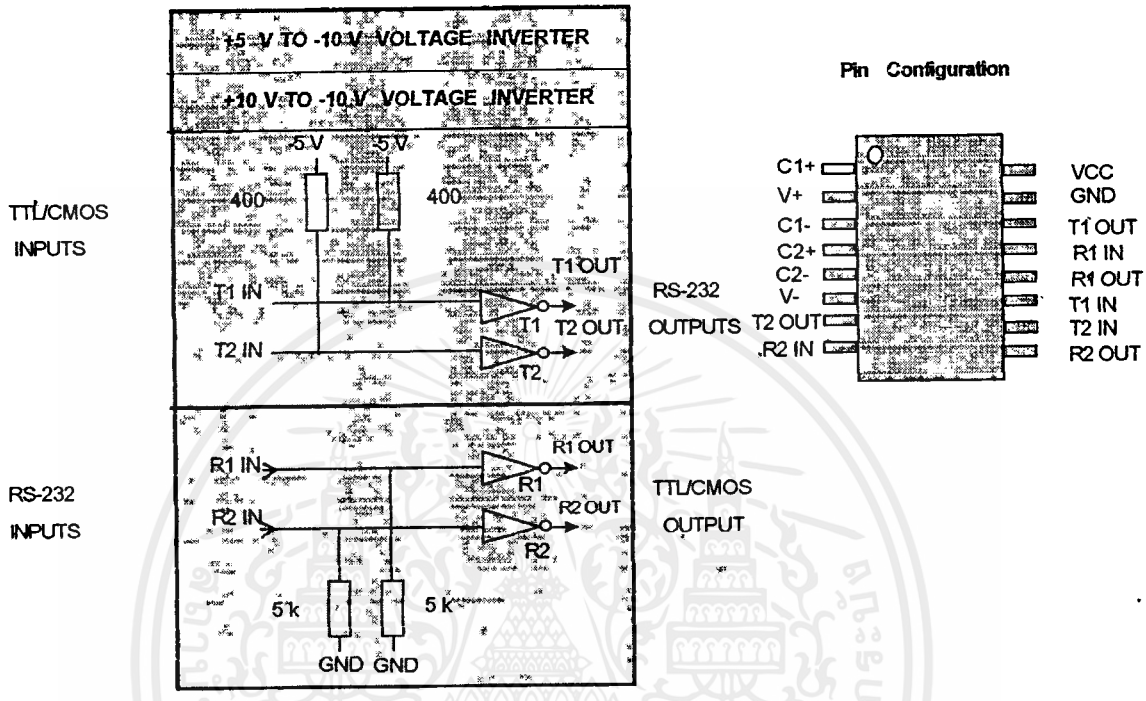
-TxD(2) สัญญาณข้อมูล สำหรับส่งให้วงจรรับข้อมูลอนุกรม

-RTS(4) ใช้รีเซทเอาต์พุตของวงจรรับข้อมูล เพื่อให้วงจรรับข้อมูลพร้อมสำหรับข้อมูลใหม่

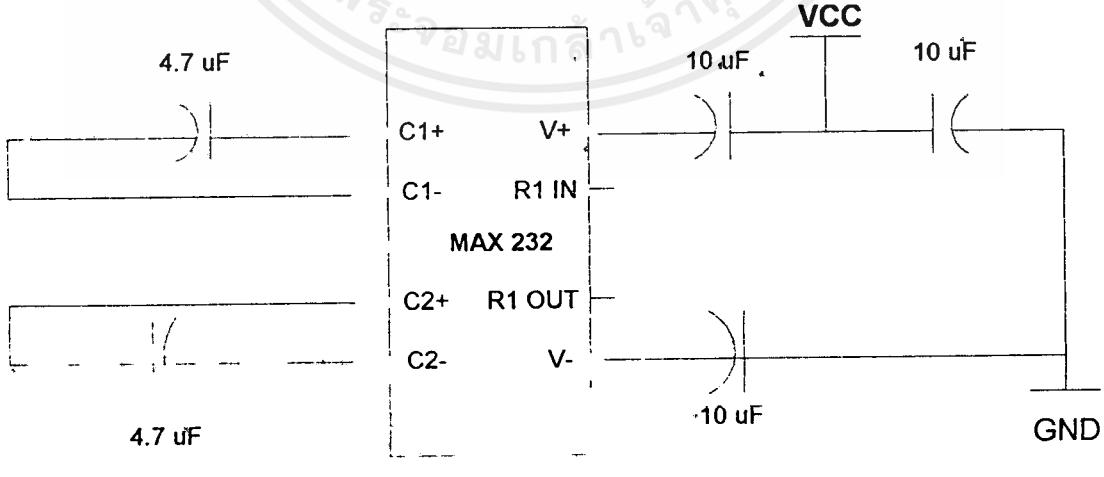
MAX-232

RS-232C DRIVERS/RECEIVERS

BLOCK DIAGRAM



รูปที่ 1 ลักษณะวงจรภายในและขาของ ไอซี MAX232



รูปที่ 2 วงจรประกอบร่วมของ MAX232

คอนเนกเตอร์ที่ใช้กับพอร์ต com2 เป็นคอนเนกเตอร์แบบ DB-25 สายสัญญาณที่ต้องต่อออกมาใช้ ได้แก่ สายข้อมูลออก TxD(2), สาย RTS(4) สำหรับใช้รีเซทวงจรรับข้อมูลอนุกรม และสายกราวนด์ SG(7) ให้นำมาต่อเข้ากับกราวนด์แหล่งจ่ายไฟของวงจรรับข้อมูล และต้องไม่ลืมที่จะป้อนแฮนด์เชคกิ้งหลอกให้ CTS(5) เด็ดขาด

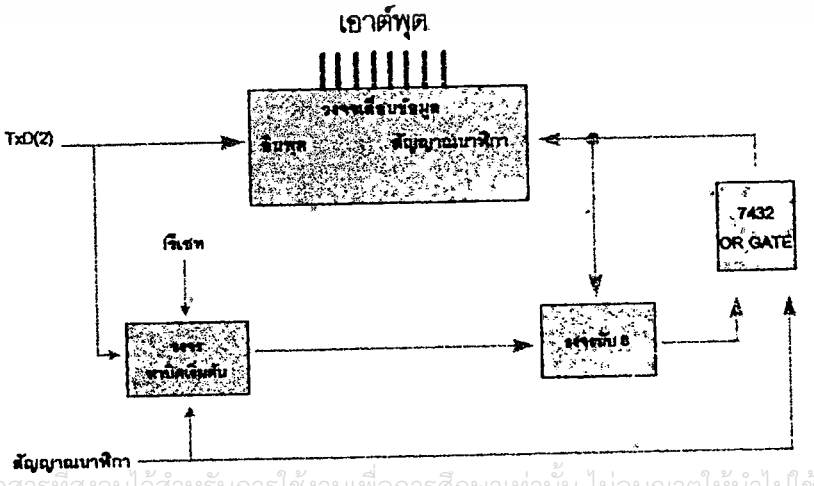
บทที่ 3
วงจรรับข้อมูลอนุกรม

วงจรรับข้อมูลอนุกรมจะทำหน้าที่รอรับข้อมูลอนุกรมหนึ่งไบต์ซึ่งถูกส่งออกจากเซอร์ฟเวอร์แล้วแปลงเป็นข้อมูลขนาน เพื่อส่งต่อไปควบคุมชุดรีเลย์ทั้งแปดตัว

วงจรหาบิตเริ่มต้นจะคอยตรวจหาบิตที่มีระดับลอจิกเป็น 0 ซึ่งมีความหมายว่ามีข้อมูลถูกส่งมาแล้ว เมื่อพบมันจะบอกไปที่วงจรนับว่าตอนนี้มีข้อมูลเข้ามาแล้ว วงจรนับก็จะเริ่มนับจำนวนสัญญาณนาฬิกา (ซึ่งสัญญาณนาฬิกาของวงจรรับต้องมีความถี่ตรงกับคอมพิวเตอร์ผู้ส่งข้อมูล) และจะเปิดเกตให้สัญญาณนาฬิกาผ่านเข้ามาได้ สัญญาณนาฬิกาจะถูกป้อนให้วงจรเลื่อนข้อมูล วงจรเลื่อนข้อมูลก็จะเลื่อนข้อมูลที่ได้จากอินพุตเข้ามาเก็บทีละบิต จนวงจรนับสัญญาณนาฬิกาครบ 8 ลูก ก็จะปิดเกตไม่ให้สัญญาณนาฬิกาผ่านเข้าไปที่วงจรเลื่อนข้อมูลได้อีกต่อไป วงจรเลื่อนข้อมูลจึงหยุดเลื่อน ผลลัพท์ที่ได้ก็จะเลื่อนมาที่เอาต์พุตทั้ง 8 ขา

แต่เราไม่ควรใช้เอาต์พุตจากชิฟท์รีจิสเตอร์ไปควบคุมรีเลย์โดยตรง เนื่องจากในระหว่างการเลื่อนบิตข้อมูลนั้น เอาต์พุตของชิฟท์รีจิสเตอร์จะมีการเปลี่ยนแปลงตลอดเวลา จำเป็นต้องนำแลตช์เข้ามาแทรก แลตช์จะดึงข้อมูลจากชิฟท์รีจิสเตอร์ก็ต่อเมื่อการรับข้อมูลนั้นเสร็จสิ้นลงแล้ว ช่วยให้การทำงานของรีเลย์แน่นอนขึ้น

ทุกครั้งก่อนที่พอร์ตจะส่งค่าใหม่ วงจรนี้จะถูกรีเซทด้วยสัญญาณจาก RTS(4) เพื่อให้วงจรหาบิตเริ่มต้นพร้อมจะเริ่มทำงานใหม่ได้



รายละเอียดวงจรรับข้อมูล

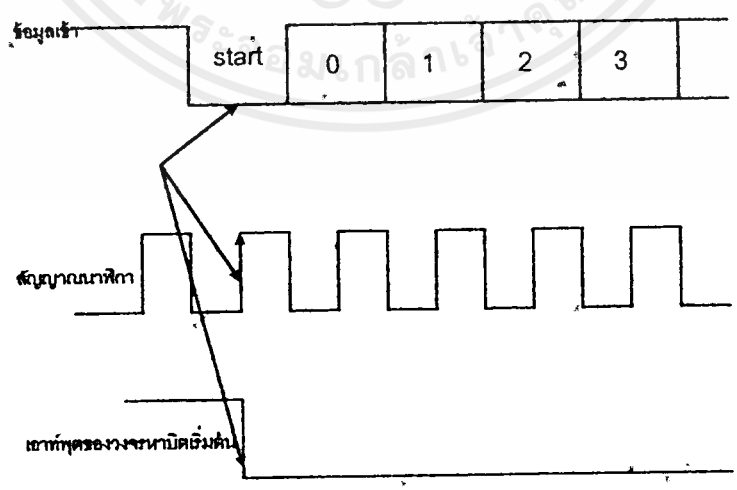
วงจรเลื่อนข้อมูล

เราใช้ไอซีชิฟทีรีจิสเตอร์ 74164 ซึ่งรับข้อมูลเลื่อนเข้าแบบอนุกรมแล้วให้เอาต์พุตออกมาในแบบขนาน ไอซีนี้มีขาที่ต้องควบคุมเพียงขาเดียวคือขาเคลียร์ สำหรับเคลียร์เอาต์พุตทั้งแปดขา เราไม่ได้ใช้ ให้ต่อเข้ากับลอจิก 1

ต่อเอาต์พุตของชิฟทีรีจิสเตอร์เข้ากับแลตช์ และต่อสัญญาณ READY ที่ออกจาก Q3 ของ 7493 เข้ากับขาสัญญาณนาฬิกาของแลตช์ เมื่อชิฟทีรีจิสเตอร์เลื่อนบิตข้อมูลเสร็จแล้ว แลตช์จะดึงค่าออกมาที่เอาต์พุตโดยอัตโนมัติ

วงจรหาบิตเริ่มต้น

ปกติสายสัญญาณที่รับข้อมูลเข้ามา หากยังไม่มีการส่งข้อมูลเข้ามาจะมีระดับลอจิกเป็น 1 และหากมีข้อมูลถูกส่งเข้ามา ข้อมูลนั้นจะถูกนำด้วยบิตเริ่มต้นซึ่งมีลอจิก 0 ดังนั้นวงจรหาบิตเริ่มต้นก็คือ วงจรตรวจจับลอจิก 0 นั่นเอง แต่เมื่อตรวจพบข้อมูลซึ่งมีลอจิก 0 แล้วจะส่งสัญญาณเอาต์พุตทันทีไม่ได้ จะต้องรอสัญญาณนาฬิกาลูกต่อไปของระบบเสียก่อน ขอให้สังเกตจากไดอะแกรมเวลาในรูปที่ 3



รูปที่ 3 แสดงไดอะแกรมเวลาของวงจรหาบิตเริ่มต้น

จากรูปจะเห็นว่าข้อมูลที่เข้ามาและสัญญาณนาฬิกาของระบบจะมีเฟสไม่ตรงกันซึ่งเป็นเรื่องปกติ แต่เราต้องจัดการให้ความถี่ทั้งคู่เท่ากัน จะสังเกตได้ว่าเมื่อมีสัญญาณที่เป็นลอจิก 0 เข้ามาแล้ว หากสัญญาณนาฬิกามีพัลส์ที่ขอบขาขึ้นในลูกต่อไป ที่เอาต์พุตของวงจรหาบิตเริ่มต้นจึงจะเริ่มทำงานได้ วงจรที่ทำงานตามสถานะของสัญญาณนาฬิกาอย่างนี้จะต้องมีส่วนประกอบเป็นฟลิปฟลอปเสมอ

จะเห็นว่าเมื่อมีบิตเริ่มต้นเข้ามาวงจรจะต้องรอให้สัญญาณนาฬิกาทำงานที่ขอบขาขึ้นก่อนมันจึงจะทำงาน หากใช้วงจรชิฟท์รีจิสเตอร์ โดยต่อให้สัญญาณข้อมูลเข้ามาเป็นสัญญาณนาฬิกาของฟลิปฟลอปตัวแรก แล้วต่อขา D ลงลอจิก 0 หากมีข้อมูลที่เป็นลอจิก 0 เข้ามาก็จะทำให้เอาต์พุต Q ออก 0 ได้และเมื่อต่อวงจรซ้อนกันอีกชั้นหนึ่งเพื่อนำลอจิก 0 นี้ออกไปเป็นเอาต์พุตของวงจรหาบิตเริ่มต้น ก็จะทำให้เอาต์พุตเป็นลอจิก 0 ได้ก็ต่อเมื่อมีสัญญาณนาฬิกาขอบขาขึ้นเข้ามาก่อน ซึ่งตรงกับไดอะแกรมเวลาในรูปที่ 3 พอดี

เราใช้ไอซีเบอร์ 7474 เป็น D ฟลิปฟลอป ขาสัญญาณนาฬิกาของไอซีเบอร์นี้จะทำงานที่ขอบขาขึ้นดังนั้นจึงต้องใส่อินเวอร์เตอร์ที่ขาสัญญาณนาฬิกาของฟลิปฟลอปตัวแรก เพื่อให้มันทำงานที่ขอบขาลง ส่วนฟลิปฟลอปตัวที่สองจะใช้สัญญาณนาฬิกาของระบบมาต่อ เพื่อให้มันทำงานไปพร้อมกับระบบ การทำงานจะเริ่มจากมีสัญญาณเคลียร์เข้ามาที่ขาปรีเซทของทั้งสองตัว ทำให้ขา Q ของมันกลายเป็นลอจิก 1 ถ้าหากมีข้อมูลเข้ามา ที่เอาต์พุตของมันจะเปลี่ยนแปลงจากลอจิก 1 มาเป็นลอจิก 0 และหากมีสัญญาณนาฬิกาขอบขาขึ้นเข้ามา ลอจิก 0 ก็จะไปปรากฏที่เอาต์พุตของฟลิปฟลอปตัวที่สองซึ่งจะตรงกับไดอะแกรมเวลาในรูปที่ 3 พอดี

วงจรนับ

วงจรนี้ทำหน้าที่รับคำสั่งจากวงจรหาบิตเริ่มต้น แล้วเริ่มนับสัญญาณนาฬิกาจนครบแปดลูก จากนั้นส่งสัญญาณไปปิดเกตเพื่อหยุดสัญญาณนาฬิกาไม่ให้เข้าไปที่วงจรเลื่อนข้อมูลได้ วงจรนี้จะใช้ 7493 การนำเอาเอาต์พุตที่จะไปปิดเกตจะใช้เฉพาะเอาต์

พุต D เท่านั้น ไอซีเบอร์นี้ที่ขาสัญญาณนาฬิกาจะทำงานที่พัลส์ขอบขาลง แต่ไอซี 74164 ขาสัญญาณนาฬิกาทำงานที่พัลส์ขอบขาขึ้น ดังนั้นสัญญาณนาฬิกาที่ผ่านเกตไปเข้าไอซีวงจรนับจะต้องผ่านอินเวอร์เตอร์ เพื่อให้ไอซีทั้งสองทำงานที่ขอบขาขึ้นเหมือนกัน

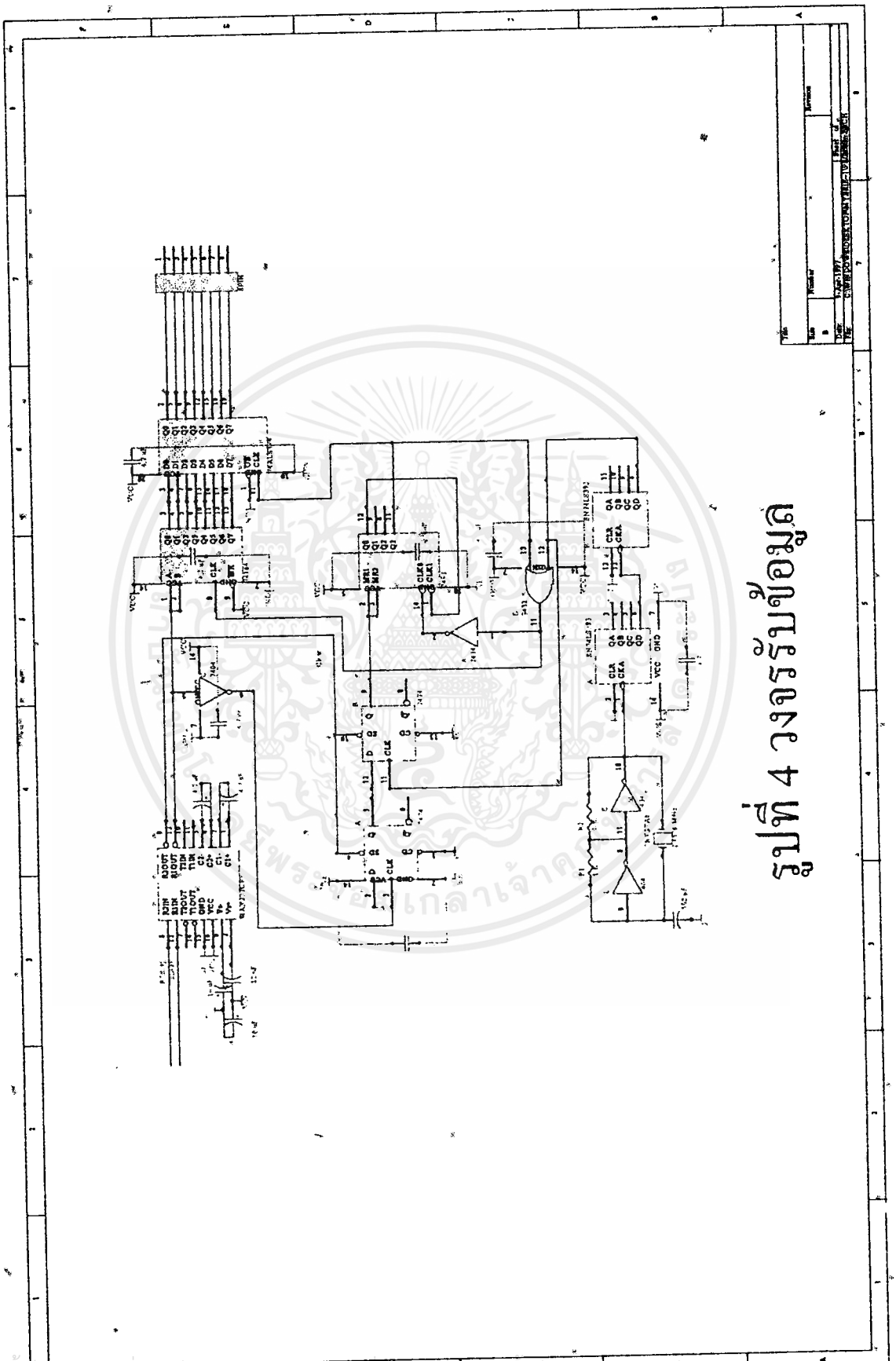
ที่ขาเอาต์พุตของวงจรนับที่เขียนว่า READY ใช้สำหรับแจ้งว่าการรับข้อมูลเรียบร้อยแล้ว เราจะนำสัญญาณนี้ไปต่อเข้ากับแลตช์ 74374 เพื่อให้แลตช์ดึงข้อมูลเมื่อพร้อม

วงจรกำหนดสัญญาณนาฬิกา

ใช้ผลึกคริสตอลความถี่ 2.4576 MHz เป็นตัวออสซิลเลต ความถี่นี้เมื่อผ่านวงจรหารสิบหกสองครั้ง จะเป็นตัวกำหนดอัตราบอดของพอร์ตอนุกรมให้เป็น 9800 บิตต่อวินาที

ประกอบวงจรทั้งหมดเข้าด้วยกันดังรูปที่ 4

อินพุตของวงจรนี้คือสัญญาณแบบ TTL ของ TxD(2) และ RTS(4) เอาต์พุตเป็นข้อมูลขนานแปดบิตสำหรับส่งต่อให้วงจรรีเลย์



รูปที่ 4 วงจรร่างต้นฉบับ

บทที่ 4
วงจรรีเลย์และแหล่งจ่ายไฟ

วงจรรีเลย์

เอาต์พุทจากแลตช์ 74374 ทั้งแปดขาถูกต่อเข้ากับขาเบสของทรานซิสเตอร์ เมื่อเอาต์พุตจากแลตช์บิตใดมีลอจิก 1 ทรานซิสเตอร์นั้นจะยอมให้กระแสไหลผ่านหน้าสัมผัสคอลเลกเตอร์-อีมิตเตอร์ซึ่งอนุกรมอยู่กับขดลวดของรีเลย์ ขดลวดของรีเลย์จะสร้างสนามแม่เหล็กดูดหน้าสัมผัสรีเลย์เข้าหากัน หน้าสัมผัสนี้จะถูกนำไปใช้เพื่อควบคุมการจ่ายกำลังให้กับเครื่องไฟฟ้าที่เราต้องการจะควบคุม

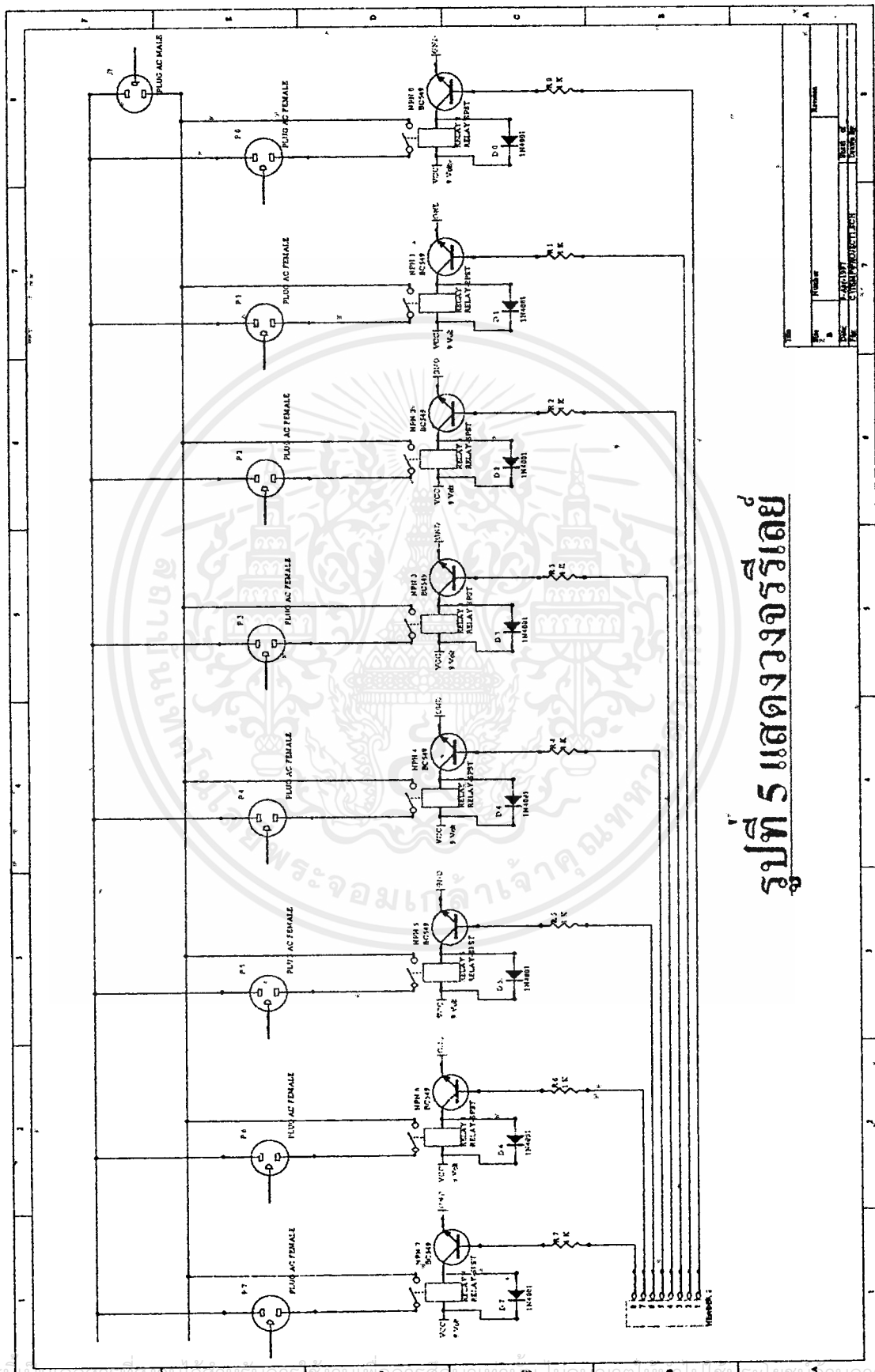
ให้อนุกรมหน้าสัมผัสนี้เข้ากับแหล่งจ่ายแรงดัน 220 โวลต์และปลั๊กตัวเมีย นำเครื่องไฟฟ้าต่อเข้ากับปลั๊กตัวเมียนี้ เราก็สามารถควบคุมการเปิดปิดเครื่องไฟฟ้านั้นได้จากเทอร์มินัลซึ่งอยู่ห่างออกไปได้

รูปที่ 5 แสดงส่วนของวงจรรีเลย์

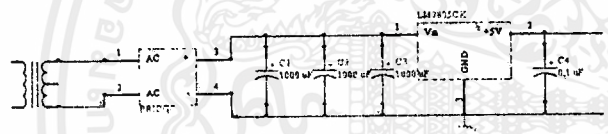
แหล่งจ่ายไฟ

จากหม้อแปลงกระแสสลับ 220 V/7.5 V เราใช้ไดโอดบริดจ์, ไอซี 7805 และตัวเก็บประจุ ร่วมกันเป็นเรกูเลเตอร์ เพื่อจ่ายโวลเตจกระแสตรง 5 Vdc เป็นไฟเลี้ยงให้วงจรรับ-ส่งข้อมูลอนุกรมและส่วนควบคุมรีเลย์

รูปที่ 6 แสดงวงจรแหล่งจ่ายไฟ



รูปที่ 5 แสดงวงจรสั่งงาน



รูปที่ 6 แสดงวงจรแหล่งจ่ายไฟ

บทที่ 5
โปรแกรมควบคุม

ภาษาโปรแกรมใหม่ - จาวา (Java)

จาวาเป็นภาษาโปรแกรมซึ่งพัฒนาขึ้นโดยบริษัท Sun Microsystems เดิมตั้งใจจะใช้จาวาเป็นภาษาโปรแกรมในเครื่องใช้ไฟฟ้าทั่วไปในบ้าน โครงการนั้นล้มเหลว ต่อมาเมื่อเวิลด์ไวด์เว็บและเบราเซอร์ที่ชื่อ Netscape Navigator ได้รับความนิยมเป็นอย่างสูง Sun จึงสนใจที่จะนำจาวากลับมาใช้กับเวิลด์ไวด์เว็บอีกครั้ง ด้วยการแจกโปรแกรมเบราเซอร์ตัวใหม่ที่ชื่อ HotJava! ไปพร้อมๆกับเครื่องมือออกแบบโปรแกรม Java Developer's Kit และประสบความสำเร็จเป็นอย่างสูงเช่นเดียวกัน ปัจจุบันการสนับสนุนจาวาเป็นมาตรฐานอย่างหนึ่งของเบราเซอร์ไปแล้ว

เดิมนั้นบนเวิลด์ไวด์เว็บจะมีเพียงรูปภาพและข้อความ แต่จาวาช่วยให้เว็บสนับสนุนอนิเมชัน กราฟิก เกม และรูปแบบพิเศษอื่นๆ มากมาย จาวายังช่วยให้การรันโปรแกรมข้ามเครือข่ายเป็นไปอย่างมีประสิทธิภาพ นอกจากโปรแกรมซึ่งฝังลงในเว็บเพจซึ่งเราเรียกว่า แอพเพลทแล้ว อันที่จริงจาวาได้ถูกออกแบบให้เป็นภาษาโปรแกรมเต็มรูปแบบสำหรับคอมพิวเตอร์ standalone เช่นกัน เบราเซอร์ HotJava! เป็นตัวอย่างหนึ่งของจาวาแอพพลิเคชัน

เราสามารถสรุปข้อดีของจาวาได้ดังนี้

1.จาวาถูกออกแบบให้มี syntax และหลักการคล้ายคลึงกับ C++ จึงง่ายสำหรับโปรแกรมเมอร์ส่วนใหญ่ นอกจากนี้จาวายังได้ตัดฟีเจอร์บางอย่างซึ่งมีปัญหาตลอดมาออกจาก C++ เช่น multiple-class inheritance, operator overloading และการจัดการเมโมรีโดยตรง

2.จาวาเป็นภาษาแบบ object-oriented ช่วยให้การออกแบบโปรแกรมเป็นไปอย่างมีประสิทธิภาพ เราถือเสมือนว่าโปรแกรมเป็นออบเจกต์อย่างหนึ่งซึ่งมีความสมบูรณ์ในตัวเอง เราสามารถนำมาใช้งานได้โดยไม่จำเป็นต้องทราบหรือสนใจรายละเอียดภายใน (modularity และ information hiding) เราอาจจะใช้คลาสเดียวสร้าง

ออบเจกต์ซ้ำแล้วซ้ำเล่า (reusability) และเราอาจจะสร้างออบเจกต์ใหม่ซึ่งมีรายละเอียดต่างไปจากเดิมเล็กน้อยได้ด้วยการสับคลาส (inheritance)

3.ข้อที่สำคัญที่สุดคือ จาวาเป็นภาษาโปรแกรมแรกซึ่งไม่ยึดติดกับระบบปฏิบัติการใดๆ (platform-independent) ดังนั้นโปรแกรมซึ่งคอมไพล์ขึ้นเพียงครั้งเดียว อาจจะรันได้บนคอมพิวเตอร์ทุกเครื่องในโลก โปรแกรมเมอร์ไม่จำเป็นต้องพอร์ตโปรแกรมหนึ่งๆไปยังระบบปฏิบัติการอื่นๆอีกต่อไป และช่วยลดปัญหาความไม่เข้ากันได้เป็นอย่างดี

4.จาวาถูกออกแบบมาให้มีความน่าเชื่อถือสูง ตัวอย่างเช่น การจัดการเมโมรีโดยตรงซึ่งสร้างปัญหาใน C มาตลอดเป็นไปโดยอัตโนมัติ

5.จาวามีระบบรักษาความปลอดภัยที่ดี เราจะมั่นใจได้ว่าแอพเพลทที่เราได้รับนั้นจะถูกตรวจสอบจนมั่นใจว่าจะไม่ทำความเสียหายให้กับระบบของเรา

6.จาวาเป็นภาษาแบบ multithreaded งานหลายๆ งานอาจจะถูกจัดการได้ในเวลาเดียวกัน ในขณะที่ภาษาโปรแกรมอื่นเช่น C/C++ เป็น single-threaded ช่วยลดเวลาเหลื่อมซึ่งเคยเกิดขึ้น และยังเหมาะกับ multiprocessor machine เป็นพิเศษ

7.จาวามีความยืดหยุ่นเป็นอย่างสูง เหมาะกับการสร้างโปรแกรมต้นแบบ (prototyping)

โปรแกรมควบคุม

โปรแกรมควบคุมถูกแบ่งออกเป็นสามส่วน ได้แก่

-โปรแกรมเขียนข้อมูลหนึ่งไบต์บนพอร์ตอนุกรม com 2

-โปรแกรมเซอร์ฟเวอร์ คอยรับข้อมูลจากผู้ใช้ซึ่งอยู่ไกลออกไป

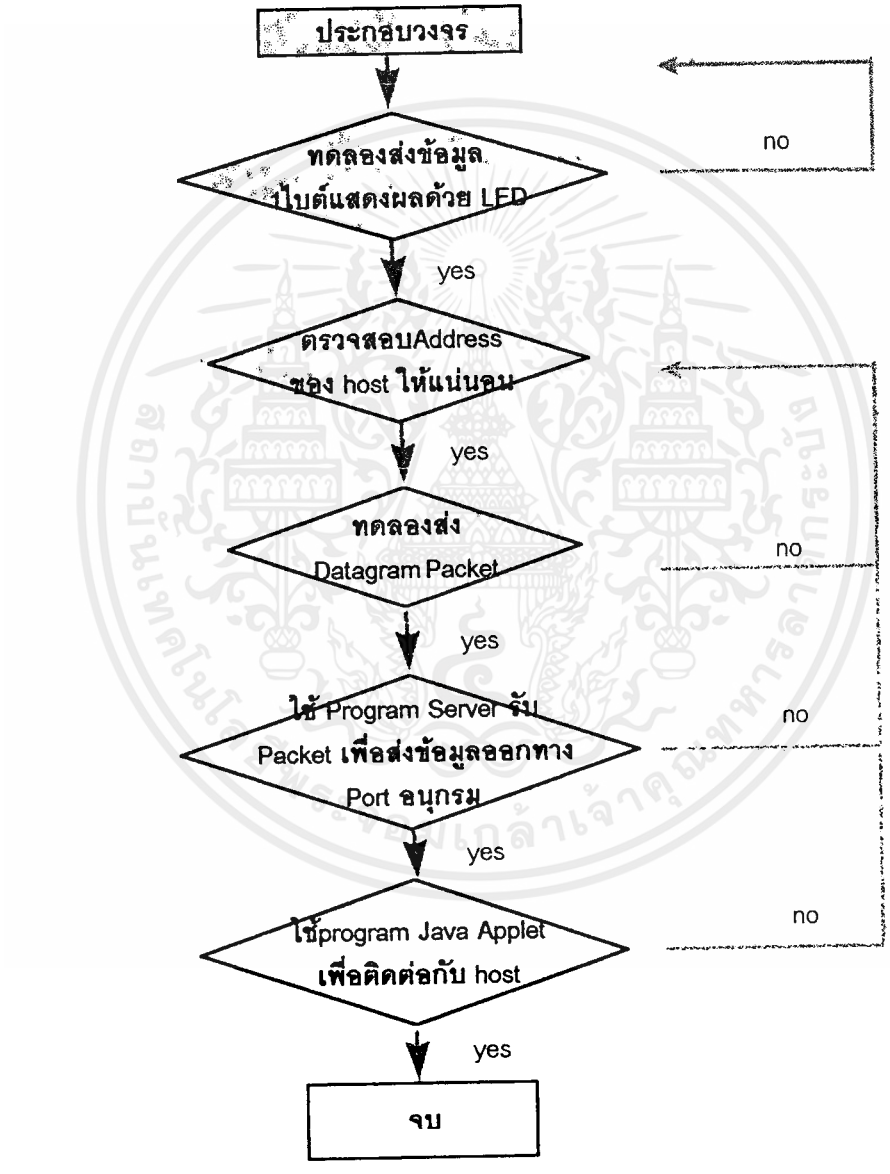
-จาวาแอพเพลทซึ่งฝังตัวอยู่ในเวบเพจทำหน้าที่เก็บข้อมูลสถานะรีเลย์จากผู้ใช้เพื่อส่งกลับมายังเซอร์ฟเวอร์

ทั้งหมดนี้เป็นโปรแกรมภาษาจาวา ซึ่งต้องใช้ JDK (Java Developer's Toolkit) 1.1 ในการรันและคอมไพล์

source code ภาษาจาวามีชื่อไฟล์เป็น Program.java เราต้องคอมไพล์

ด้วย Java Compiler (javac.exe) เสียก่อน ผลการคอมไพล์ได้ไฟล์รหัสคำสั่ง Program.class จากนั้น เราจะเรียกใช้ไฟล์รหัสคำสั่งใดๆ ด้วย Java Interpreter (java.exe)

ลำดับขั้นการทดสอบและออกแบบโปรแกรมควบคุมของเราเป็นดังนี้



โปรแกรมเขียนข้อมูลหนึ่งไบต์บนพอร์ตอนุกรม com2

เพราะว่าจาวาเป็นภาษาโปรแกรมซึ่งไม่ยึดติดกับระบบปฏิบัติการใดๆ จาวาจึงไม่รู้จักพอร์ตอนุกรม com2 ของดอส เราแก้ปัญหานี้ด้วยการอินเตอร์เฟสจาวาเข้ากับภาษาซีมาตรฐานซึ่งเข้าถึงพอร์ตอนุกรมได้

สร้างออบเจกต์ที่เรียกว่า SecondComport (SecondComport.java) เพื่อเป็นตัวแทนของพอร์ต com 2 จริงๆ ให้ใช้โปรแกรม javah.exe ในการสร้าง header (SecondComport.h) และ stub (SecondComport.c) สำหรับอินเตอร์เฟสเข้ากับซี สังเกตว่าใน SecondComport.java นั้น method ที่ชื่อว่า write() นั้นไม่มีตัวคำสั่งจริง ตัวคำสั่งจริงจะเป็นรหัสภาษาซีในไฟล์ SecondComportImp.c ซึ่งเราเขียนขึ้นมาต่างหาก

ต่อไปให้ใช้ Visual C++ คอมไพล์ SecondComport.c และ SecondComportImp.c ได้ออบเจกต์ไฟล์ SecondComport.obj และ SecondComportImp.obj ลิงก์ออบเจกต์ไฟล์และ header ทั้งหมดเข้าด้วยกันเป็นไลบรารีไฟล์ SecondComport.lib, SecondComport.dll และ SecondComport.exp

ทุกครั้งที่เราเรียกใช้ออบเจกต์ SecondComport มันจะไปโหลดไลบรารี SecondComport.dll เป็นอันว่าเราจะเขียนข้อมูลบนพอร์ตอนุกรม com2 ได้ด้วยออบเจกต์นี้

ต่อไปจะทดสอบวงจรรับข้อมูลอนุกรม ประกอบวงจรรับข้อมูลอนุกรม โดยเพิ่ม LED ที่เอาต์พุตของแลตช์ ให้เอาต์พุตจากแลตช์ผ่าน LED ลงกราวนด์ การต่อแบบนี้ LED จะติดสว่างเมื่อเอาต์พุตจากแลตช์เป็น 1 ประกอบเสร็จแล้วนำมาทดสอบกับโปรแกรม TestSecondComport (TestSecondComport.java) ได้ TestSecondComport จะทดสอบการเขียนข้อมูลหนึ่งไบต์ไปยังพอร์ตอนุกรม com2 ด้วย method ที่ชื่อ write()

write() จะตั้งค่าพารามิเตอร์สำหรับการส่งข้อมูล (9600 บิตต่อวินาที,

1บิตจบ,ไม่มีพาริตี้บิต) และรีเซทวงจรรับข้อมูลอนุกรมด้วยสัญญาณ RTS(4) จากนั้นจึงส่งข้อมูลหนึ่งไบต์ไปยังพอร์ต ซึ่งเราจะควบคุมสัญญาณ RTS(4) นี้ได้ด้วยการเข้าถึงรีจิสเตอร์โมเด็มคอนโทรลของพอร์ต com 2

ลองเปรียบเทียบข้อมูลที่ส่งออกและข้อมูลที่รับได้ว่าตรงกันหรือไม่

โปรแกรมเซอร์ฟเวอร์

ก่อนจะเขียนโปรแกรมเซอร์ฟเวอร์ได้นั้นเราต้องมั่นใจเสียก่อนว่า เราสามารถส่งและรับข้อมูลผ่านเครือข่ายอินเตอร์เนตได้จริง

การส่งข้อมูลข้ามเครือข่ายของโครงงานนี้จะใช้วิธีส่งแบบ UDP (Unreliable Datagram Protocol) ซึ่งข้อมูลจะเดินทางได้เร็วแต่ไม่ปลอดภัยมากนัก packet ข้อมูลที่ส่งอาจจะสูญหายได้ระหว่างเดินทางมายังเซอร์ฟเวอร์

เขียนและรันโปรแกรม LocalHost (LocalHost.java) เพื่อตรวจสอบอินเตอร์เนตแอดเดรสที่แน่นอนของเว็บเซอร์ฟเวอร์นั้น

นำแอดเดรสที่ได้ไปใช้ทดสอบการรับและส่ง DatagramPacket ให้รันโปรแกรม DatagramReceive (DatagramReceive.java) ทิ้งเอาไว้ที่เว็บเซอร์ฟเวอร์ รันโปรแกรม DatagramSend (DatagramSend.java)บนคอมพิวเตอร์อีกเครื่องหนึ่งบนเครือข่ายอินเตอร์เนต

packet ที่รับได้จะถูกแสดงบนหน้าจอของเว็บเซอร์ฟเวอร์ หากรับข้อมูลไม่ได้ให้ตรวจสอบหมายเลขพอร์ตและแอดเดรสของเซอร์ฟเวอร์ให้ถูกต้อง สำหรับหมายเลขพอร์ตนั้นผู้ดูแลเซอร์ฟเวอร์สามารถเลือกใช้หมายเลขพอร์ตที่ว่างได้เสมอ และหมายเลขพอร์ตนี้จะทำหน้าที่แทนรหัสผ่านสำหรับควบคุมรีเลย์

ต่อไปเป็นโปรแกรมเซอร์ฟเวอร์ที่เราจะใช้งานจริงประกอบด้วยสองไฟล์คือ ServerThread (ServerThread.java) และ Server (Server.java) ServerThread เป็นสับคลาสของ Thread เนื่องจากโปรแกรมเซอร์ฟเวอร์ของเราจะต้องรันอยู่ตลอดเวลา ServerThread จะคอยรับ DatagramPacket อยู่ตลอดเวลาเฉพาะ packet ที่นำหน้าด้วยอักษร 'a' เท่านั้นจึงจะมีผลกับการทำงานของ ServerThread

ServerThread จะนำตัวเลขฐานสิบ ซึ่งตามหลังอักษร 'a' ใน packet ที่รับได้ไปเขียนสถานะใหม่ของรีเลย์ลงในไฟล์ และส่งข้อมูลที่รับได้ไปยังพอร์ตอนุกรม com 2

โปรแกรม Server ทำหน้าที่เพียงเริ่มการทำงานของ ServerThread เท่านั้น

ทดสอบโปรแกรม ServerThread ด้วยการติดตั้งวงจรรับข้อมูลเข้ากับพอร์ต com 2 รันโปรแกรม Server ทิ้งไว้เพื่อรอรับ packet ที่จะถูกส่งเข้ามา

ส่ง packet ด้วยโปรแกรม DatagramSend โดยกำหนดข้อมูลที่ส่งเป็นสตริง ซึ่งขึ้นต้นด้วยอักษร 'a' และต่อท้ายด้วยเลขฐานสิบตั้งแต่ 0 - 255

หากการทำงานของ ServerThread ถูกต้องวงจรรับจะต้องแสดงผลและไฟล์ Status.txt จะต้องถูกเขียนขึ้นด้วย ถ้า DatagramSend ส่งสตริงซึ่ง ServerThread ไม่เข้าใจมันจะไม่สนใจ packet นั้นหรือเกิดข้อผิดพลาดขึ้น (exception) จนโปรแกรมหยุดทำงาน

จาวาแอพเพลท

แม้ว่า DatagramSend อาจจะทำการควบคุมรีเลย์ได้แล้วก็ตาม จาวาแอพเพลทจะเพิ่มความสะดวกให้กับผู้ใช้ได้มาก ClientApplet (ClientApplet.java) เป็นสับคลาสของแอพแพลทซึ่งเราสามารถนำ ClientApplet ไปรันด้วยโปรแกรมเบราเซอร์ทั่วไปได้

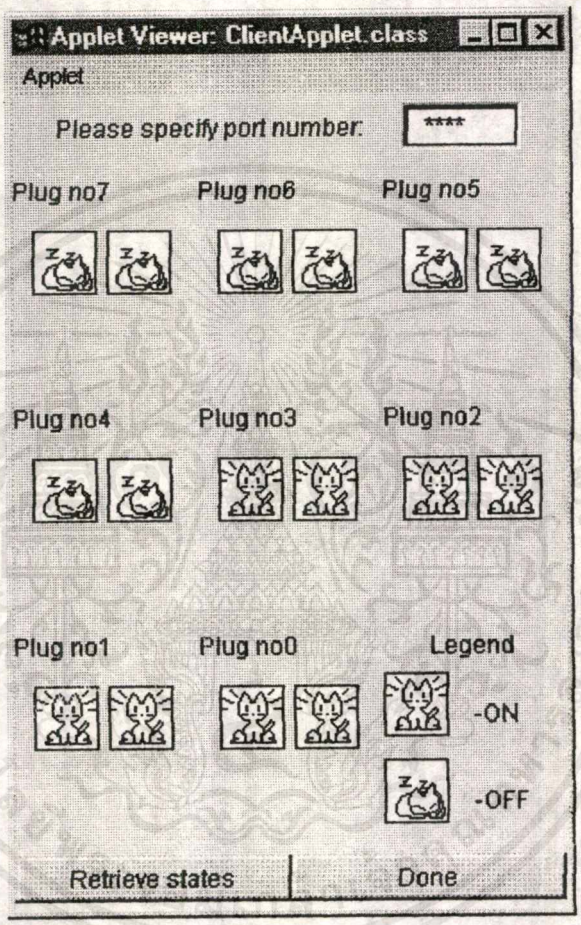
ClentApplet จะควบคุมรีเลย์ผ่านทางโปรแกรม Server และ ServerThread เช่นเดียวกับการทดสอบบังคับรีเลย์ด้วยโปรแกรม DatagramSend

ClientApplet จะใช้ ImageCanvas (ImageCanvas.java) เพื่อแสดงรูปภาพทั้งหมดบนแอพเพลท

เมื่อผู้ใช้ติดต่อเว็บเซอร์ฟเวอร์เข้ามาด้วยโปรแกรมเบราเซอร์ ผู้ใช้จะได้รับจาวาแอพแพลทดังรูปที่ 7 จากนั้นผู้ใช้สามารถควบคุมรีเลย์จากระยะไกลได้โดยผ่านทางแอพแพลทนี้

ปุ่ม Retrieve states จะอ่านค่าสถานะปัจจุบันของเครื่องไฟฟ้าจากไฟล์ States.txt บนเครื่องเซอร์ฟเวอร์ ค่าที่อ่านได้จะถูกแสดงด้วยภาพทางซ้ายมือ

ผู้ใช้จะควบคุมรีเลย์ซึ่งควบคุมเครื่องไฟฟ้าด้วยการคลิกเมาส์บนรูปภาพแต่ละคู่ ภาพทางขวามือจะเปลี่ยนกลับไปมา โดยไม่มีผลกับรูปภาพทางซ้ายมือ (ค่าสถานะเดิมที่อ่านได้)



รูปที่ 7 แสดงแอพแพลทที่เราได้รับเมื่อติดต่อกับเซอร์ฟเวอร์

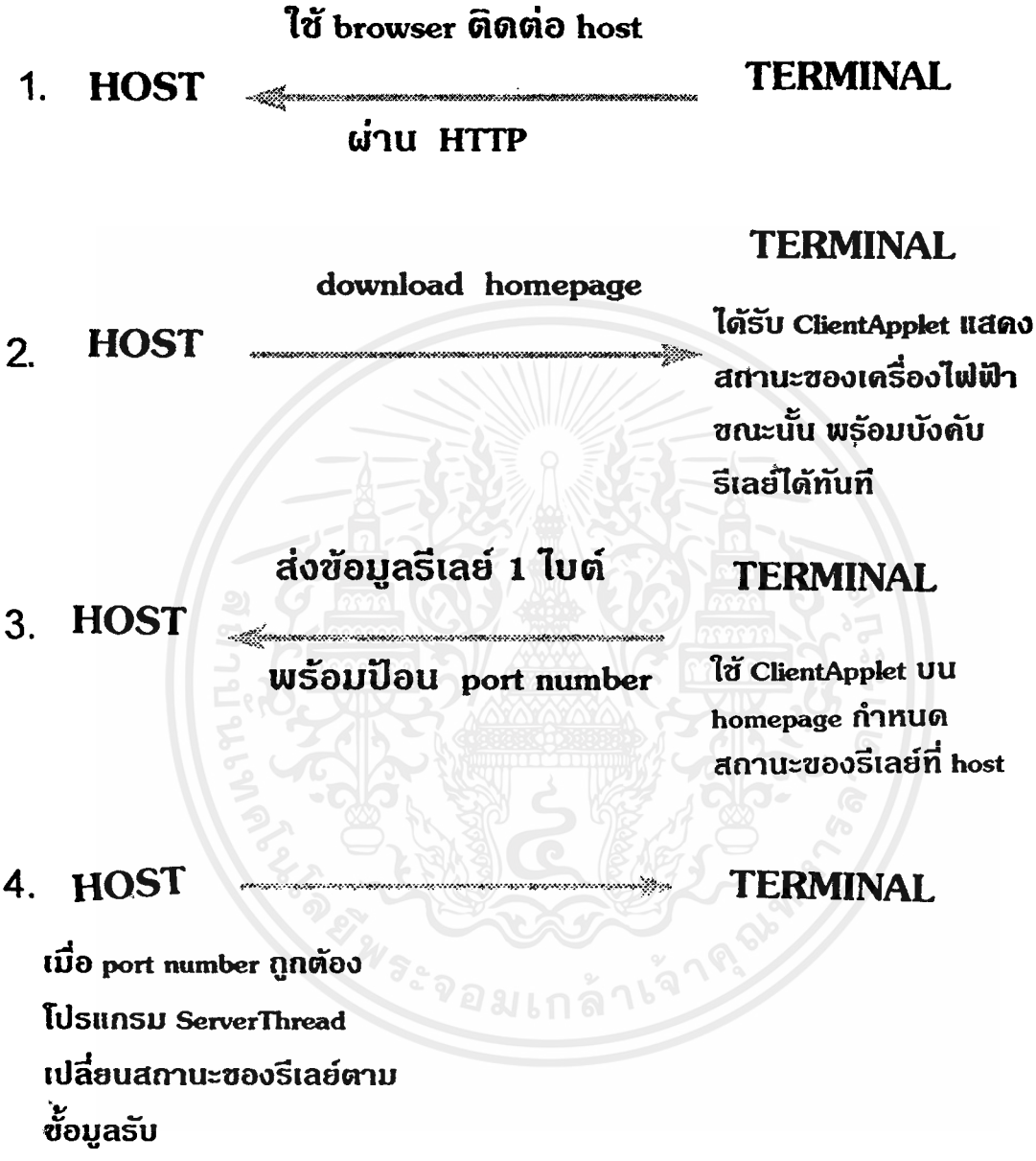
ปุ่ม Done จะส่งค่าที่ผู้ใช้เลือกขึ้นมาใหม่ไปควบคุมรีเลย์ซึ่งต่ออยู่กับพอร์ตอนุกรม com2 ของเครื่องเซอร์ฟเวอร์ในลักษณะของ DatagramPacket

มีความเป็นไปได้ว่า DatagramPacket จะสูญหายระหว่างทาง ดังนั้นหลังจากการส่ง packet แต่ละครั้ง แล้วเราควรจะตรวจสอบสถานะเครื่องไฟฟ้าด้วยการคลิกบนปุ่ม Retrieve states

ผู้ใช้ที่ไม่ทราบพอร์ตซึ่งเซอร์ฟเวอร์คอยรับค่าอยู่จะไม่มีสิทธิในการควบคุม เนื่องจาก packet ที่ส่งจะไปไม่ถึงเซอร์ฟเวอร์ หมายเลขพอร์ตนี้เป็นระบบรักษาความปลอดภัยแบบหนึ่ง เราจะระบุพอร์ตของเซอร์ฟเวอร์ได้ด้วย TextField ด้านบน

หากต้องการเปลี่ยนหมายเลขพอร์ตนี้ จะต้องแก้ source code ของ ServerThread คอมไพล์ใหม่แล้วจึงรันโปรแกรมได้ทันที

ผังแสดงการติดต่อระหว่าง ClientApplet และ ServerThread



บทที่ 6
การติดตั้งและใช้งาน

หลังจากประกอบวงจรรับข้อมูล, วงจรรีเลย์, แหล่งจ่ายไฟ และคอนเนกเตอร์ DB-25 เข้าด้วยกันแล้ว ให้ต่อคอนเนกเตอร์เข้ากับพอร์ตอนุกรม com2

รันโปรแกรม WWW server ตามปกติ เพื่อคอย load เว็บเพจให้ผู้ใช้ที่ติดต่อเข้ามา

รันโปรแกรมจาวาที่ชื่อ Server โปรแกรม Server จะไปสตาร์ท ServerThread ให้รอรับ packet ซึ่งผู้ใช้จะส่งมา

นำเครื่องไฟฟ้าที่ต้องการควบคุมจากระยะไกลเสียบเข้ากับปลั๊ก ซึ่งต่ออยู่กับวงจรรีเลย์ และให้เปิดสวิทช์เครื่องไฟฟ้านั้นไว้ด้วย

เนื่องจาก packet ที่ถูกส่งโดย ClientApplet อาจจะมาไม่ถึงเซอร์ฟเวอร์ เพราะฉนั้นผู้ใช้ต้องตรวจสถานะของรีเลย์อีกครั้งว่ามีการเปลี่ยนแปลงหรือไม่ หากไม่มีการเปลี่ยนแปลงให้ส่ง packet ซ้ำอีกครั้ง

การตรวจสอบสถานะของรีเลย์นั้นเราใช้การอ่านค่าจากไฟล์ Status.txt ซึ่งถูกเขียนขึ้นทุกครั้งที่มีการส่งข้อมูลไปยังพอร์ต com2 เราทดสอบแล้วว่าวงจรรับข้อมูลนี้ทำงานได้ค่อนข้างแน่นอน

จุดที่ต้องระวังจะมีเพียงสองจุด คือ packet มาไม่ถึงเซอร์ฟเวอร์ หรือเครื่องไฟฟ้ามีปัญหาเกี่ยวกับสวิทช์หรือทางเดินกระแส

เราทดสอบวงจรของเรา ด้วยการกำหนดตัวเลขสุ่มค่าตั้งแต่ 0-255 ส่งออกทางพอร์ตอนุกรมไปยังรีเลย์ซึ่งควบคุมเครื่องไฟฟ้า แล้วตรวจสอบความถูกต้องของข้อมูลที่รับได้ปรากฏว่า ถูกต้อง 229 ครั้งจากทั้งหมด 255 ครั้งผิดพลาดประมาณ 8.4%

souce codeในภาคผนวกนั้น ถูกเขียนขึ้นด้วย JDK 1.1 ซึ่งเป็นเวอร์ชันล่าสุดปัจจุบันยังไม่มีเบราเซอร์ใดนอกจาก HotJava! (ซึ่งพัฒนาขึ้นโดย Sun Microsystems เช่นกัน) จะรันแอพเพลท ClientApplet ได้

รายชื่อไฟล์สำคัญ

1) SecondComport.class คลาสตัวแทนของพอร์ต com2 จริงๆ เพื่อให้จาวาคลาสอื่นๆ เรียกใช้งานพอร์ต com2 ได้
2) SecondComport.dll เป็นไฟล์ dynamic library สำหรับใช้งานร่วมกับ SecondComport.class
3) SecondComport.lib เช่นเดียวกับ SecondComport.dll
4) SecondComport.exp เช่นเดียวกับ SecondComport.dll
5) Server.class เป็นจาวาคลาสสำหรับเริ่มการทำงานของ ServerThread.class
6) ServerThread.class เป็นคลาสหลักทำหน้าที่คอยรับ packet ข้อมูลซึ่งส่งโดยแอพเพลท ClientApplet.class
7) ClientApplet.class ตัวแอพเพลทซึ่งคอยรับข้อมูลจากผู้ใช้ รวบรวมสร้างเป็น packet ส่งข้ามเครือข่ายมายัง ServerThread.class ซึ่งรอรับอยู่
8) ClientApplet.html เว็บเพจที่อยู่ของ ClientApplet.class เพื่อให้เบราเซอร์โหลดแอพเพลทไปรันบนเครื่องของผู้ใช้ได้
9) ImageCanvas.class สำหรับแสดงรูปภาพต่างๆ บนแอพเพลท ทำงานร่วมกับ ClientApplet.class
10) awake.gif ไฟล์รูปภาพ
11) sleep.gif ไฟล์รูปภาพ

ไฟล์ทั้งหมดจะต้องถูกเก็บไว้ในไดเรคทอรีที่เป็น document root ของโปรแกรมเวิลด์ไวด์เว็บเซอร์ฟเวอร์ (ไดเรกทอรีซึ่งใช้เก็บเว็บเพจ)

โครงงานนี้ใช้ประโยชน์จากเวิลด์ไวด์เว็บในการควบคุมเครื่องไฟฟ้าได้สำเร็จด้วยการแยกส่วนของฮาร์ดแวร์ออกจากซอฟท์แวร์ และใช้พอร์ตอนุกรม com2 ในการเชื่อมสองส่วนนี้เข้าด้วยกัน

เราอาจนำโครงงานนี้ไปพัฒนาต่อเพื่อการควบคุมซึ่งมีความซับซ้อนกว่านี้ เช่น การควบคุมระดับ หรืออัตราไหล ด้วยการส่งเซทพ้อยนท์ข้ามเครือข่าย เป็นต้น

บรรณานุกรม

I. ทรงชัย วีระทวีมาศ, "เทคนิคการออกแบบวงจรดิจิตอล-วงจรรับส่งข้อมูลแบบอนุกรม", เซมิคอนดักเตอร์อิเลกทรอนิกส์ , บ. ซีเอ็ดยูเคชัน จำกัด, พ.ย.2534

II. Peter Gofton, "Mastering serial communications", Sybex Incoperation, 1994.

III. Laura Lemay & Charles Perkins, "Teach yourself Java in 21 days", Sams.net Publication, 1996.

IV. Glenn Vandenburg, "Tricks of the Java programming gurus", Sams.net Publication, 1996.

V. Alexander Newman, et al., "Special Edition Using Java", Que Coperation, 1996.

VI. Colin Fraizer & Jill Bond, "Java API Reference", New Riders Publishing, 1996.

กิตติกรรมประกาศ

โปรเจคท์นี้ลุล่วงไปได้ ด้วยความช่วยเหลือของอาจารย์หลายท่าน โดยเฉพาะอาจารย์ที่ปรึกษา อาจารย์ทรงชัย วีระทวีมาศ ซึ่งเป็นเจ้าของวงจรรับข้อมูลอนุกรมนี้ และคอยให้ความช่วยเหลือเสมอมา

อาจารย์ประสิทธิ์ จุลเสรีวงศ์ ที่ช่วยแบ่งปันทรัพยากรมาให้กลุ่มผู้จัดทำ

เพื่อนๆ 4J ที่น่ารักทุกคน

พี่ๆ ห้องอิมเมจ ที่คอยให้ยืมเกือบทุกอย่าง

ขอขอบพระคุณเป็นอย่างสูง

ปิยธิดา และ พรเทพ

SecondComport.java

```
public class SecondComport {
  private short status;
  public native void write();

  public void setStatus(short inpStatus)  {
    status = inpStatus;
  }
  static {
    System.loadLibrary("SecondComport");
  }
}
```

SecondComport.h

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <native.h>
/* Header for class SecondComport */

#ifndef _Included_SecondComport
#define _Included_SecondComport

#pragma pack(4)

typedef struct ClassSecondComport {
    long status;
} ClassSecondComport;
HandleTo(SecondComport);

#pragma pack()

#ifdef __cplusplus
extern "C" {
#endif
extern void SecondComport_write(struct HSecondComport *);
#ifdef __cplusplus
}
#endif
#endif
```

SecondComport.c

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <StubPreamble.h>

/* Stubs for class SecondComport */
/* SYMBOL: "SecondComport/write()V", Java_SecondComport_write_stub */
__declspec(dllexport) stack_item *Java_SecondComport_write_stub(stack_item *_P_,struct execenv
*_EE_) {
        extern void SecondComport_write(void *);
        (void) SecondComport_write(_P_[0].p);
        return _P_;
}
```

SecondComportImp.c

```
#include <StubPreamble.h>
#include "SecondComport.h"
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#define TxRx  0x2f8
#define STATUS 0x2fd
#define MCTRL 0x2fc
#define RCV_RDY 1
void SecondComport_write(struct HSecondComport *hthis){
   int i,sentNumber= unhand(hthis)->status;
   unsigned char sentChar ;
   sentChar = (unsigned char) sentNumber;
   printf("%x written to com2\n",sentChar);
   _asm{
      mov ax,0x00E3
      mov dx,1
      int 14h
   }
   _outp(MCTRL,0x2);
   while(i<100) i++;
   _outp(MCTRL,0x0);
   _outp(TxRx,sentChar);
}
```

TestSecondComport.java

```
class TestSecondComport{
    public static void main(String argv[]).
    {
        SecondComport com2 = new SecondComport();
        com2.setStatus((short)Integer.parseInt(argv[0]));
        com2.write();

    }
}
```

LocalHost.java

```
import java.net.*;

class LocalHost {
    public static void main(String argv[]) throws UnknownHostException {
        InetAddress localHostObject =  InetAddress.getLocalHost();

        System.out.println("getHostName() returns "+localHostObject.getHostName());
        System.out.println("getHostAddress() returns "+ localHostObject.getHostAddress());
    }
}
```

DatagramSend.java

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;


public class DatagramSend {
   public static void main(String argv[]) throws Exception {


      String String_Christie = argv[0];
      byte[] byte_Christie = new byte[String_Christie.length()];
      byte_Christie = String_Christie.getBytes();
      int port = 10000;


      InetAddress server =
           InetAddress.getByName("161.246.20.16");
      DatagramPacket packet = new DatagramPacket(byte_Christie,
           String_Christie.length(),server,port);
      DatagramSocket socket = new DatagramSocket();
      socket.send(packet);
      socket.close();
   }
}
```

DatagramReceive.java

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;

public class DatagramReceive {
   public static void main(String argv[]) throws Exception {

      String receivedMessage;
      byte[] buffer = new byte[2048];
      int endIndex=100;
      DatagramPacket rcv_packet = new DatagramPacket(buffer,
           buffer.length);
      DatagramSocket rcv_socket = new DatagramSocket(10000);

      rcv_socket.receive(rcv_packet);
      receivedMessage = new String(buffer);
foo:
      for (int i = 0; i<buffer.length; i++) {
        if ((buffer[i]==0) & (buffer[i+1]==0)) {
           endIndex = i;
           break foo;
        }
      }
      System.out.println(receivedMessage.substring(0,endIndex));
      SecondComport com2 = new SecondComport();
      com2.setStatus((short) Integer.parseInt(receivedMessage.substring(0,endIndex)));
      com2.write();
      rcv_socket.close();

   }
}
```

ServerThread.java

```
import java.net.*;
import java.io.*;

class ServerThread extends Thread {

    private volatile int port=10000;
    private         String receivedMessage;
    private         byte[] buffer;
    private         DatagramPacket rcv_packet;
    private         DatagramSocket rcv_socket;
    private         DataOutputStream aDO;
    ServerThread() {
      super("Server");
      try {
        rcv_socket = new DatagramSocket(port);
        System.out.println("Server listening on port: "+rcv_socket.getLocalPort());
      } catch (java.io.IOException e) {
        System.out.println("Specified port is unavailable");
      }
    }

    public void run() {
      System.out.println("Program runnable...");
      System.out.println();
      while(true) {
        System.out.println();
        System.out.println("waiting for packet");
        byte[] buffer = new byte[20];

        rcv_packet = new DatagramPacket(buffer,buffer.length);
        try {
          rcv_socket.receive(rcv_packet);
        }catch(Exception e){};
```

```
        receivedMessage = new String(rcv_packet.getData());
        receivedMessage = receivedMessage.trim();
        System.out.println("receivedMessage = "+receivedMessage);
        if(receivedMessage.charAt(0)=='a') {
          SecondComport com2 = new SecondComport();

com2.setStatus((short)Integer.parseInt(receivedMessage.substring(1,receivedMessage.length())));
          com2.write();
          try {
          File statusFile = new File("Status.txt");
          aDO = new DataOutputStream(new FileOutputStream(statusFile));
          aDO.writeBytes("                    ");
          aDO= new DataOutputStream(new FileOutputStream(statusFile));
          aDO.writeBytes(receivedMessage.substring(1,receivedMessage.length()));
          } catch(Exception e){};
        System.out.println("one datagrampacket managed.");
        }
      }


    }
  }
```

Server.java

```
class Server {
    public static void main(String argv[]) {
        new ServerThread().start();

    }

}
```

ClientApplet.java

```
import java.awt.*;
import java.applet.*;
import java.net.*;
import java.util.*;
import java.io.*;

public class ClientApplet extends Applet {

    private final    Color MYBLUE= new Color(200,200,255);
    private volatile boolean[] statusBoolean = new boolean[8];
    private volatile boolean[] selectionBoolean = new boolean[8];
    private          int port;
    private          TextField portTextField;
    private          String line;
    private          String hostName = "161.246.20.16";
    private          Image sleepImage,awakeImage,dukeImage;
    private          Panel panel70,panel60,panel50,panel40,
                         panel30,panel20,panel10,panel00;

    public void init() {

        sleepImage = getImage(this.getCodeBase(),"sleep.gif");
        awakeImage = getImage(this.getCodeBase(),"awake.gif");
        dukeImage  = getImage(getCodeBase(),"duke.gif");

        setBackground(MYBLUE);
        setLayout(new BorderLayout(3,3));
        Panel northPanel = new Panel();
        Panel southPanel = new Panel();
        Panel centerPanel = new Panel();

        add(northPanel,"North");
        add(southPanel,"South");
        add(centerPanel,"Center");

        northPanel.setLayout(new FlowLayout());
        portTextField = new TextField("*****",5);
```

```
northPanel.add(new Label("Please specify port number:",Label.LEFT));
northPanel.add(portTextField);
southPanel.setLayout(new GridLayout(1,2));
southPanel.add(new Button("Retrieve states"));
southPanel.add(new Button("Done"));

centerPanel.setLayout(new GridLayout(3,3,3,3));

Panel  panel7 =new Panel();
panel7.setLayout(new BorderLayout(3,3));
panel7.add(new Label("Plug no7"),"North");
panel70 = new Panel();
panel7.add(panel70,"Center");
panel70.setLayout(new FlowLayout());
centerPanel.add(panel7);

Panel  panel6 =new Panel();
panel6.setLayout(new BorderLayout(3,3));
panel6.add(new Label("Plug no6"),"North");
panel60 = new Panel();
panel6.add(panel60,"Center");
panel60.setLayout(new FlowLayout());
centerPanel.add(panel6);

Panel  panel5 =new Panel();
panel5.setLayout(new BorderLayout(3,3));
panel5.add(new Label("Plug no5"),"North");
panel50 = new Panel();
panel5.add(panel50,"Center");
panel50.setLayout(new FlowLayout());
centerPanel.add(panel5);

Panel  panel4 =new Panel();
panel4.setLayout(new BorderLayout(3,3));
panel4.add(new Label("Plug no4"),"North");
panel40 = new Panel();
panel4.add(panel40,"Center");
panel40.setLayout(new FlowLayout());
centerPanel.add(panel4);
```

```
Panel  panel3 =new Panel();
panel3.setLayout(new BorderLayout(3,3));
panel3.add(new Label("Plug no3"),"North");
panel30 = new Panel();
panel3.add(panel30,"Center");
panel30.setLayout(new FlowLayout());
centerPanel.add(panel3);

Panel  panel2 =new Panel();
panel2.setLayout(new BorderLayout(3,3));
panel2.add(new Label("Plug no2"),"North");
panel20 = new Panel();
panel2.add(panel20,"Center");
panel20.setLayout(new FlowLayout());
centerPanel.add(panel2);

Panel  panel1 =new Panel();
panel1.setLayout(new BorderLayout(3,3));
panel1.add(new Label("Plug no1"),"North");
panel10 = new Panel();
panel1.add(panel10,"Center");
panel10.setLayout(new FlowLayout());
centerPanel.add(panel1);

Panel  panel0 =new Panel();
panel0.setLayout(new BorderLayout(3,3));
panel0.add(new Label("Plug no0"),"North");
panel00 = new Panel();
panel0.add(panel00,"Center");
panel00.setLayout(new FlowLayout());
centerPanel.add(panel0);

Panel legendPanel = new Panel();
Panel temp = new Panel();

legendPanel.setLayout(new BorderLayout(2,2));
legendPanel.add("North",new Label("Legend",Label.CENTER));
legendPanel.add("Center",temp);
```

```
temp.setLayout(new GridLayout(2,2,2,2));
temp.add(new ImageCanvas(awakeImage,this,30,30));
temp.add(new Label("-ON",Label.LEFT));
temp.add(new ImageCanvas(sleepImage,this,30,30));
temp.add(new Label("-OFF",Label.LEFT));

centerPanel.add(legendPanel);



validate();
System.out.println("retrieve() returns "+ retrieve());
fillBoolean(statusBoolean,retrieve());
fillBoolean(selectionBoolean,(short)0);
screen(statusBoolean,selectionBoolean);

for (int i=7;i>-1;i--)
System.out.println("bit "+i+" is " +statusBoolean[i]);
System.out.println("portTextField.getText() returns "
        +portTextField.getText());
System.out.println("selectionString() returns "+selectionString());
}

public boolean action(Event evt, Object arg) {

  if(evt.target instanceof Button) {
    if((String) arg == "Retrieve states") {
       showStatus("Retrieving current states from server..");
       fillBoolean(statusBoolean,retrieve());
       screen(statusBoolean,selectionBoolean);
       showStatus("Current states displayed..");
       return true;
    }

    if((String) arg=="Done") {
       showStatus("Sending datagram to server..");
       try {
          datagramSend(selectionString());}
```

```
        catch (Exception e) {
          showStatus("Exception from sending datagram");
          return false;};
        screen(statusBoolean,selectionBoolean);
        return true;
      }
      return false;

    }
    return true;
}

public void datagramSend(String inputString) throws Exception {

    byte[] byte_array = new byte[inputString.length()];
    byte_array = inputString.getBytes();
    try {
      port = Integer.parseInt(portTextField.getText());}
    catch(NumberFormatException e){
      showStatus("NumberFormatException thrown by datagramSend()");};
    InetAddress server = InetAddress.getByName(hostName);

    DatagramPacket packet = new DatagramPacket(byte_array,
          inputString.length(),server,port);

    DatagramSocket socket = new DatagramSocket();

    socket.send(packet);
    socket.close();

}

public boolean mouseDown(Event evt, int x, int y) {

    if((10<=x && x <= 80) && (65<=y && y<=100)) {
      selectionBoolean[7] =!selectionBoolean[7];
      screen(statusBoolean,selectionBoolean);
      return true;}
```

```
if((100<=x && x <= 170) && (65<=y && y<=100)) {
   selectionBoolean[6] =!selectionBoolean[6];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((190<=x && x <= 260) && (65<=y && y<=100)) {
   selectionBoolean[5] =!selectionBoolean[5];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((10<=x && x <= 80) && (175<=y && y<=205)) {
   selectionBoolean[4] =!selectionBoolean[4];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((100<=x && x <= 170) && (175<=y && y<=205)) {
   selectionBoolean[3] =!selectionBoolean[3];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((190<=x && x <= 260) && (175<=y && y<=205)) {
   selectionBoolean[2] =!selectionBoolean[2];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((10<=x && x <= 80) && (280<=y && y<=315)) {
   selectionBoolean[1] =!selectionBoolean[1];
   screen(statusBoolean,selectionBoolean);
   return true;}

if((100<=x && x <= 170) && (280<=y && y<=315)) {
   selectionBoolean[0] =!selectionBoolean[0];
   screen(statusBoolean,selectionBoolean);
   return true;}

  return false;
}

public String selectionString() {
```

```
    int selectionNumber=0;
    if(selectionBoolean[7])  selectionNumber+=128;
    if(selectionBoolean[6])  selectionNumber+=64;
    if(selectionBoolean[5])  selectionNumber+=32;
    if(selectionBoolean[4])  selectionNumber+=16;
    if(selectionBoolean[3])  selectionNumber+=8;
    if(selectionBoolean[2])  selectionNumber+=4;
    if(selectionBoolean[1])  selectionNumber+=2;
    if(selectionBoolean[0])  selectionNumber+=1;
    String selectionString= "a";
    selectionString =selectionString.concat(Integer.toString(selectionNumber));
    return selectionString;
}
public short retrieve() {
    try {
       URL statusURL = new URL(getCodeBase(),"Status.txt");
       DataInputStream aDI = new DataInputStream(
             new BufferedInputStream(statusURL.openStream()));
       line =aDI.readLine();
       aDI.close();
       line = line.trim();}
     catch (Exception e) {
       showStatus("Exception in retrieve()");};
     return((short) Integer.parseInt(line));

}
public void fillBoolean(boolean[] inputBoolean,short inputShort) {
     for (int i=0;i<inputBoolean.length;i++)
        inputBoolean[i]=false;
     int inputInt =(int) inputShort;

     if(inputInt>255)
        System.out.println("Error at fillBoolean()!");

     if(inputInt/128 ==1) {
        inputBoolean[7]=true;
        inputInt=inputInt-128;}

     if(inputInt/64 ==1) {
```

```
    inputBoolean[6]=true;
    inputInt=inputInt-64;}

  if(inputInt/32 ==1) {
    inputBoolean[5]=true;
    inputInt=inputInt-32;}

  if(inputInt/16 ==1) {
    inputBoolean[4]=true;
    inputInt=inputInt-16;}

  if(inputInt/8 ==1) {
    inputBoolean[3]=true;
    inputInt=inputInt-8;}

  if(inputInt/4 ==1) {
    inputBoolean[2]=true;
    inputInt=inputInt-4;}

  if(inputInt/2 ==1) {
    inputBoolean[1]=true;
    inputInt=inputInt-2;}

  if(inputInt ==1) {
    inputBoolean[0]=true;
  }

}
public void screen(boolean[] inputBoolean1,boolean[] inputBoolean2) {
  panel70.removeAll();
  panel60.removeAll();
  panel50.removeAll();
  panel40.removeAll();
  panel30.removeAll();
  panel20.removeAll();
  panel10.removeAll();
  panel00.removeAll();

  if(inputBoolean1[7]) {
```

```
    panel70.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel70.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[6]) {
    panel60.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel60.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[5]) {
    panel50.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel50.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[4]) {
    panel40.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel40.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[3]) {
    panel30.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel30.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[2]) {
    panel20.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel20.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[1]) {
    panel10.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel10.add(new ImageCanvas(sleepImage,this,30,30));}

  if(inputBoolean1[0]) {
    panel00.add(new ImageCanvas(awakeImage,this,30,30));}
  else{
    panel00.add(new ImageCanvas(sleepImage,this,30,30));}
```

```
if(inputBoolean2[7]) {
   panel70.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel70.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[6]) {
   panel60.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel60.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[5]) {
   panel50.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel50.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[4]) {
   panel40.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel40.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[3]) {
   panel30.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel30.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[2]) {
   panel20.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel20.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[1]) {
   panel10.add(new ImageCanvas(awakeImage,this,30,30));}
else{
   panel10.add(new ImageCanvas(sleepImage,this,30,30));}

if(inputBoolean2[0]) {
   panel00.add(new ImageCanvas(awakeImage,this,30,30));}
```

```
    else{
      panel00.add(new ImageCanvas(sleepImage,this,30,30));}

    validate();
  }

  public void paint(Graphics g) {
    g.setColor(MYBLUE);
  }

}
```

ImageCanvas.java

```
import java.awt.*;
import java.applet.*;
import java.net.*;

class ImageCanvas extends Canvas {
    Container pappy;
    Image image;
    boolean trueSizeKnown = false;
    Dimension minSize;
    int w, h;

    public ImageCanvas(Image image, Container parent,
                       int initialWidth, int initialHeight) {
        if (image == null) {
            System.err.println("Canvas got invalid image object!");
            return;
        }

        this.image = image;
        pappy = parent;

        w = initialWidth;
        h = initialHeight;

        minSize = new Dimension(w,h);
    }

    public Dimension preferredSize() {
        return minimumSize();
    }

    public synchronized Dimension minimumSize() {
        return minSize;
    }

    public void paint (Graphics g) {
        if (image != null) {
```

```
            if (!trueSizeKnown) {
                int imageWidth = image.getWidth(this);
                int imageHeight = image.getHeight(this);

                if ((imageWidth > 0) && (imageHeight > 0)) {
                    trueSizeKnown = true;

                    //Component-initiated resizing.
                    w = imageWidth;
                    h = imageHeight;
                    minSize = new Dimension(w,h);
                    resize(w, h);
                    pappy.validate();
                }
            }

            g.drawImage(image, 0, 0, this);
            g.drawRect(0, 0, w - 1, h - 1);
        }
    }

}
```

ClientApplet.html

```
<APPLET CODE=ClientApplet.class WIDTH=275 HEIGHT=400>
</APPLET>
```

5404/7404 Hex Inverter

	Schottky TTL				High-Speed TTL				Low-Power Schottky TTL				Standard TTL				Low-Power TTL			
	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF
T.I.	SN54S04	J,W				SN54H04	J,W				SN54LS04	J,W				SN5404	J,W				SN54L04	J,W			
	SN74S04	J,N				SN74H04	J,N				SN74LS04	J,N				SN7404	J,N				SN74L04	J,N			
FAIRCHILD	FM54S04 FM9S04	D			F	FM54H04 FM9H04	D			F	FM54LS04/FM9LS04	D			F	FM5404 FM9N04	D			F					
	FC74S04 FC9S04	D	P			FC74H04/FC9H04	D	P			FC74LS04/FC9LS04	D	P			FM7404/FC9N04	D	P							
MOTOROLA						MC3705	L			F						MC5404	L			F					
						MC3008	L	P		F	SN74LS04		P			MC7404	L	P							
N.S.C.						DM54H04	J,D	N			DM54LS04					DM5404	J,D	N		W	DM54L04	J,D	N		F
	DM74S04		N			DM74H04	J,D	N			DM74LS04					DM7404	J	N			DM74L04	J,D	N		F
PHILIPS	N74S04		F			N74H04		①			N74LS04		①			FJH241,7404		F							
SIGNETICS	S54S04	F,D	A		W	S54H04	F,D	A,D		W						S5404	F	A		W					
	N74S04	F,D	A		W	N74H04	F,D	A,D			N74LS04		A			N7404	F,D	A,D							
SIEMENS																									
FUJITSU																FLH211		F							
HITACHI											74LS04		MP			MB418	D	MP							
	HD74S04		DP								HD74LS04		P			HD7404,HD2522	D	P,D							
MITSUBISHI	M5S004		P,D								M74LS04		P,D			M53204		P,D							
NEC																									
	74S04		C								74LS04		C			μPB235	D								
TOSHIBA																TD3404A		P,D							

Electrical Characteristics SN54LS04/SN74LS04

absolute maximum ratings over operating free-air temperature range

Supply voltage V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 150°C

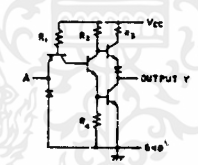
recommended operating conditions

	SN54LS04 Min	Nom	Max	SN74LS04 Min	Nom	Max	Unit
Supply voltage V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS †		MIN	TYP ‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IL} = V_{IL} max, I_{OH} = MAX		2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, I_{OL} = 4mA				0.4	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_{IH} = 2.7V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_{IL} = 0.4V				−0.4	mA
I_{OS}	Short-circuit output current *	V_{CC} = MAX	54LS Family	−20		−100	mA
			74LS Family	−20		−100	
I_{CCH}	Supply current	V_{CC} = MAX	Total, outputs high		1.2	2.4	mA
I_{CCL}	Supply current		Total, outputs low		3.6	6.6	mA
I_{CC}	Supply current	V_{CC} = 5V	Average per gate (50% duty cycle)		0.4		mA
t_{PLH}	Propagation delay time, low-to-high-level output	V_{CC} = 5V, C_L = 15PF,	T_A = 25°C, R_L = 2KΩ		9	15	ns
t_{PHL}	Propagation delay time, high-to-low-level output				10	15	ns

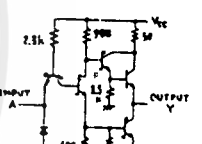
Schematics (each gate)



CIRCUIT	R1	R2	R3	R4
'04	4k	1 pk	130	1k
'L04	40k	20k	500	12k

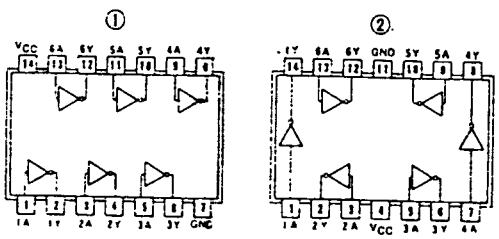
Input clamp diodes not on SN54L/SN74L* circuits.

'04, 'L04 CIRCUITS

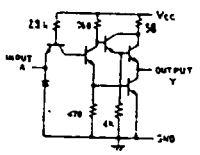


'S04 CIRCUIT

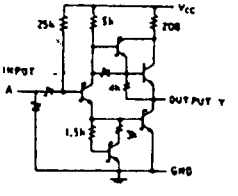
Pin Assignments (Top View)



positive logic:
$Y = \overline{A}$



'H04 CIRCUIT



'L304 CIRCUIT

Resistor values shown are nominal and in ohms.

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
* Not more than one output should be shorted at a time, and for SN54H', SN74H' and SN54S'/SN74S', duration of short-circuit should not exceed 1 second.

54164/74164 8-Parallel-Out Serial Shift Register

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	
T.I.											SN54LS164	J			W	SN54164	J	D		W	SN54L164	J			
											SN74LS164	J	D	N		SN74164	J	D	N		SN74L164	J	D	N	
FAIRCHILD											FM54LS164/FWRS164	D			F	FM54164, FW93164	D								
											FC74LS164/FCRLS164	D	P		F	FC74164/FC93164	D	P							
MOTOROLA											SN74LS164		P	D		MC73164		P	D						
N.S.C.											DM74LS164		J								DM54L164	J			
											DM54LS164		J			DM74164		N	J		DM74L164	J	D	N	
PHILIPS											N74LS164		D			N74164		D							
SIGNETICS																S54164	F	D	A	J	W				
											N74LS164		A	D		N74164	F	D	A	J					
SIEMENS																FLJ441			D						
FUJITSU											74LS164		M	D											
HITACHI											HD74LS164		P	D		HD74164		P	D						
MITSUBISHI											M74LS164		P	D		M53364			P	D					
NEC											AM74LS		C	T		μPB2164	D	D							
TOSHIBA																TD3503A		P	D						
AMD											Am74LS164														

Electrical Characteristics SN54LS164/SN74LS164

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7 V	Operating free-air temperature range	SN54*	−55°C	to 125°C
Input voltage	7 V		SN74*	0°C	to 70°C
		Storage temperature range		−65°C	to 150°C

recommended operating conditions

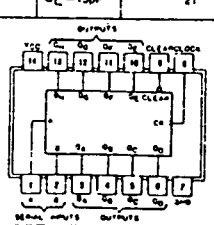
	SN54LS164			SN74LS164			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Clock frequency, f_{clock}	0		25	0		25	MHz
Width of clock or clear input pulse, t_w	20			20			ns
Data setup time, t_{setup}	15			15			ns
Data hold time, t_{hold}	5			5			ns
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −400 μA		2.7	3.5		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 8 mA			0.35	0.5	V
I_I	Input current maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4 V				0.4	mA
I_{OS}	Short-circuit output current*	V_{CC} = MAX	SN54LS	−20		100	mA
			SN74LS	−20		100	
I_{CC}	Supply current	V_{CC} = MAX, See Note 1			16	27	mA
f_{max}	Maximum clock frequency	V_{CC} = 5 V, T_A = 25°C, R_L = 2 kΩ	C_L = 15 pF	25	36		MHz
t_{PHL}	Propagation delay time, high-to-low level Q outputs from clear input		C_L = 15 pF		24	36	ns
t_{PLH}	Propagation delay time, low-to-high-level Q outputs from clock input		C_L = 15 pF		17	27	ns
t_{PHL}	Propagation delay time, high-to-low level Q outputs from clock input		C_L = 15 pF		21	32	ns

Pin Assignment (Top View)

①



positive logic: see function table

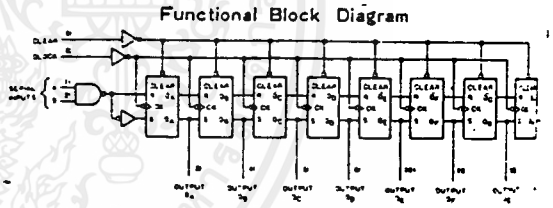
†For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type
‡All typical values are at V_{CC} = 5 V, T_A = 25°C.
*Not more than two outputs should be shorted at a time.

Function Table

'164, 'LS164, 'L164 (see Note 2)

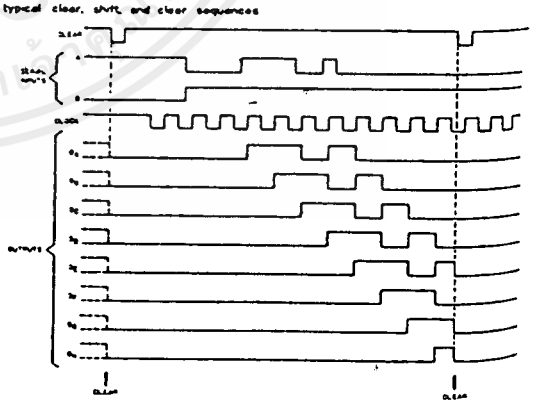
INPUTS				OUTPUTS			
CLEAR	CLOCK	A	B	QA	QB	···	QH
L	X	X	X	L	L		L
H	L	X	X	Q_{A0}	Q_{B0}		Q_{H0}
H	↑	H	H	H	Q_{An}		Q_{Gn}
H	↑	L	X	L	Q_{An}		Q_{Gn}
H	↑	X	L	L	Q_{An}		Q_{Gn}

Functional Block Diagram



'164, 'LS164, 'L164 8-BIT PARALLEL-OUT SHIFT REGISTER

typical clear, shift, and clear sequences



NOTES
1. I_{CC} is measured with outputs open, serial inputs grounded, and a momentary ground, then 4.5 V, applied to clear.
2. H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level.
Q_{A0}, Q_{B0}, Q_{H0} = the level of Q_A, Q_B, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Gn} = the level of Q_A or Q_G before the most recent ↑ transition of the clock; indicates a one-bit shift.

5432/7432 Quadruple 2-Input Positive-OR Gate

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S32	J①			W①						SN54LS32	J①			W①	SN5432	J①			W①					
	SN74S32	J①	N①								SN74LS32	J①	N①			SN7432	J①	N①							
FAIRCHILD	FM54S32/FM9S32	D①			F①						FM54LS32/FM9LS32	D①			F①	FM5432/FM9N32	D①			F①					
	FC74S32/FC9S32	D①	P①		F①						FC74LS32/FC9LS32	D①	P①		F①	FC7432/FC9N32	D①	P①							
MOTOROLA																									
											SN74LS32		P①												
N.S.C.											DM74LS32		①			DM5432	J①	N①		W①	DM54L32				
											DM54LS32		①			DM74232	J①	N①			DM74L32				
PHILIPS	N74S32		①								N74LS32		①			N7432		①							
SIGNETICS																5432	F①	A①		W①					
											N74LS32		A①			7432	F①	A①							
SIEMENS																									
																FLH631		①							
FUJITSU																									
											74LS32		M①												
HITACHI																									
											HD74LS32		P①			HD7432	①	P①							
MITSUBISHI																									
											M53LS32		P①												
NEC																									
											74LS32		C①												
TOSHIBA																									

Electrical Characteristics SN54LS32/SN74LS32

absolute maximum ratings over operating free-air temperature range

Supply voltage V_{CC}	7V	Operating free-air temperature range	SN54_S	−55°C to 125°C
Input voltage	7V		SN74_S	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommended operating conditions

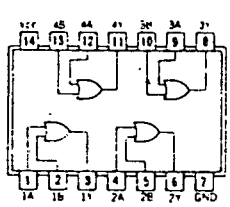
	SN54LS32			SN74LS32			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS †		MIN	TYP ‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, I_{OH} = MAX, V_{IH} = 2V		2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, I_{OL} = 4mA, V_{IL} = V_{IL} max.			0.25	0.4	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7V				0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_{IH} = 2.7V				20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_{IL} = 0.4V				−0.4	mA
I_{OS}	Short-circuit output current §	V_{CC} = MAX	54LS Family	−20		−100	mA
			74LS Family	−20		−100	
I_{CCH}	Supply current	V_{CC} = MAX	Total outputs high		3.1	6.2	mA
I_{CCL}	Supply current		Total outputs low		4.9	9.8	mA
I_{CC}	Supply current	V_{CC} = 5V	Average per gate (50% duty cycle)		1.0		mA
t_{PLH}	Propagation delay time, low-to-high-level output	V_{CC} = 5V, C_L = 15pF,	T_A = 25°C, R_L = 2kΩ		14	22	ns
t_{PHL}	Propagation delay time, high-to-low-level output				14	22	ns

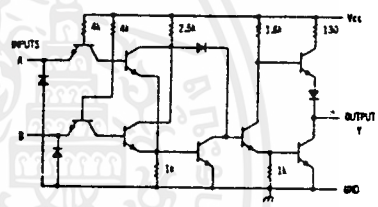
Pin Assignment (Top View)

①

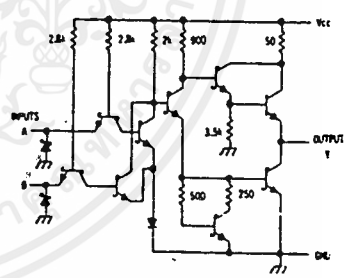


positive logic:
Y = A + B

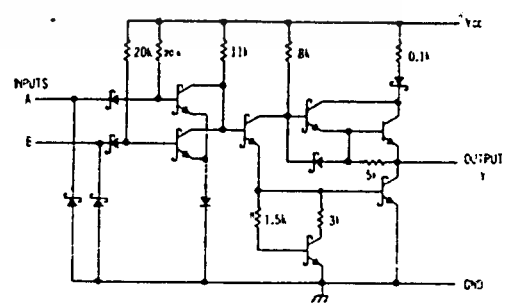
Schematics(each gate)



'32 CIRCUIT



'S32 CIRCUIT



'LS32 CIRCUIT

Resistor values shown are nominal and in ohms

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions
‡ All typical values are at V_{CC} = 5V, T_A = 25°C
§ Not more than one output should be shorted at a time

54393/74393 Dual 4-Bit Binary Counter

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T. I.											SN54LS393	J①			W①	SN54393	J①			W①					
FAIRCHILD											SN74LS393	J①	N①			SN74393	J①	N①							
MOTOROLA																									
N. S. C.																									
PHILIPS																									
SIGNETICS																									
SIEMENS																									
FUJITSU																									
HITACHI																									
MITSUBISHI											M74LS393		P①												
NEC																									
TOSHIBA																									

Electrical Characteristics SN54LS393/SN74LS393

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 125°C

recommended operating conditions

		SN54LS390			SN74LS390			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				400			−400	μA
Low-level output current, I_{OL}				4			8	mA
Clock frequency, f_{count}	A input	0		25	0		25	MHz
	B input	0		20	0		20	
Pulse width, t_w	A input high or low	20			20			ns
	B input high or low	25			25			
	Clear high	20			20			
Inactive-state setup time, t_{su}		25 ↓			25 ↓			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

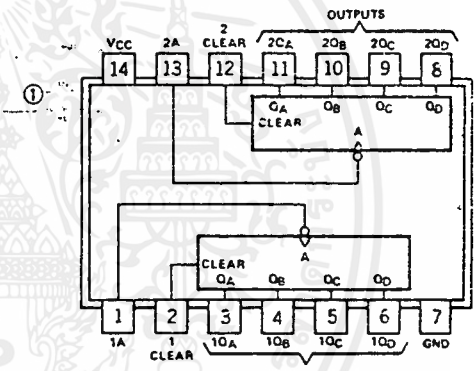
PARAMETER		TEST CONDITIONS †		MIN	TYP‡	MAX	UNIT
V_{IH} High-level input voltage				2			V
V_{IL} Low-level input voltage						0.8	V
V_{IK} Input clamp voltage		V_{CC}=MIN, I_I= −18mA				−1.5	V
V_{OH} High-level output voltage		V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OH}= −400μA		2.7	3.4		V
V_{OL} Low-level output voltage		V_{CC}=MIN, V_{IH}=2V, V_{IL}=0.8V, I_{OL}=8mA			0.35	0.5	mA
I_I Input current at maximum input voltage	Clear	V_{CC}=MAX, V_I=7V				0.1	mA
	Input A	V_{CC}=MAX, V_I=5.5V				0.2	
	Input B					0.4	
I_{IH} High-level input current	Clear	V_{CC}=MAX, V_I=2.4V				20	μA
	A					100	
	B					100	
I_{IL} Low-level input current	Clear	V_{CC}=MAX, V_I=0.4V				−0.4	mA
	A					−1.6	
	B					−2.4	
I_{OS} Short-circuit output current ¶		V_{CC}=MAX	SN54	−20		−100	mA
			SN74	−20		−100	
I_{CC} Supply current		V_{CC}=MAX, See Note 1			15	26	mA

Clock A to output Q_A	V_{CC} = 5 V, T_A = 25°C, C_L = 15pF, R_L = 2kΩ	25	35		MHz
Clear A to output Q_A			13	20	ns
Clock A to output Q_D			40	60	ns
Clear to Q_D output			40	60	
Clear to B low output			24	36	ns

I_{CC} is measured with all outputs open, both clear inputs grounded following momentary connection to 4.5V, and all other inputs grounded.

† Arrow indicates that the falling edge of the clock pulse is used for reference.
‡ f_{max} = maximum count frequency, t_{PHL} = propagation delay time, high-to-low-level output, t_{PLH} = propagation delay time, low-to-high-level output.
¶ Q_A outputs of the 390 are tested at I_{OL} = 16 mA plus the limit value for I_{IL} for the B input. This permits driving the B input while maintaining full fan-out capability.
NOTE 1: With all outputs open, apply the appropriate value specified under recommended operating conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ Typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time.

Pin Assignment (Top View)



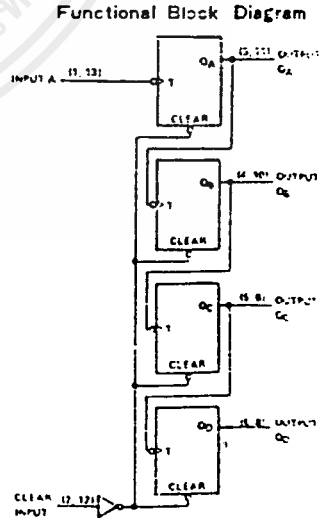
positive logic: High input to clear resets all four outputs low

Function Table

COUNT SEQUENCE (EACH COUNTER)

COUNT	OUTPUT			
	QD	QC	QB	QA
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

Functional Block Diagram



'393 DUAL 4-BIT BINARY COUNTER

5474/7474 Dual D-Type Positive-Edge-Triggered Flip-Flop with Preset and Clear

	Schottky TTL					High-Speed TTL *					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package				Device Type	Package			
		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF		C	P	M	CF
T.I.	SN54S74	J①			W①	SN54H74	J①			W②	SN54LS74	J①			W①	SN5474	J①			W②	SN54L74	J①	N①	T①	
	SN74S74	J①	N①			SN74H74	J①	N①			SN74LS74	J①	N①			SN7474	J①	N①			SN74L74	J①	N①	T①	
FAIRCHILD	FM54S74/FM9S74	D①				FM54H74/FM9H74	D①			F②	FM54LS74/FM9LS74	D①			F①	FM5474/FM9N74	D①			F②					
	FC74S74/FC9S74	D①	P①			FC74H74/FC9H74	D①	P①		F②	FC74LS74/FC9LS74	D①	P①		F①	FC7474/FC9N74	D①	P①							
MOTOROLA											SN74LS74		P①												
N.S.C.	DM74S74		N①			DM54H74	J①	N①			DM54LS74		P①			DM5474	J①	N①		W②	DM54L74	J①	N①		F②
						DM74H74	J①	N①			DM74LS74					DM7474	J①	N①			DM74L74	J①	N①		F②
PHILIPS	N74S74		①			GJJI31/74H74		①			N74LS74		①			FJJI31/7474		①							
SIGNETICS	8S4874					S54H74	F①	A①		W①						S5474	F①	A①		W②					
	N74S74					N74H74	F①	A①			N74LS74		A①			N7474	F①	A①							
SIEMENS																FLJ141		①							
FUJITSU											74LS74		MA①			MB420	①	M①							
HITACHI	HD74S74	①	P①								HD74LS74		P①			HD7474/HD2S10	①	P①							
MITSUBISHI	M74S74		P①								M74LS74		F①			M5327/M5374		P①							
NEC	74S74		C①								74LS74		C①			μPB214	D①	C①							
TOSHIBA																TD3474A		P①							

Electrical Characteristics SN54LS74/SN74LS74

absolute maximum ratings over operating free-air temperature range

Supply voltage V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	5.5V		SN74LS	0°C to 70°C
		Storage temperature range		−65°C to 150°C

recommended operating conditions

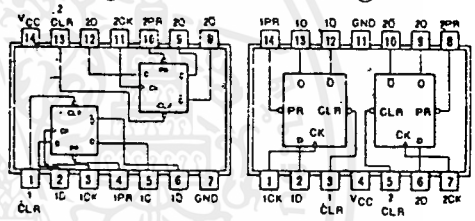
		SN54LS74			SN74LS74			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−400			−400	μA
Low-level output current, I_{OL}				4			8	mA
Pulse width, t_w	Clock high	25			25			ns
	Preset or clear low	25			25			
Input setup time, t_{setup}	High-level data	25↑			25↑			ns
	Low-level data	20↑			20↑			
Input hold time, t_{hold}		5↑			5↑			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range

PARAMETER¶		TEST CONDITIONS†	MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage		2			V
V_{IL}	Low-level input voltage				0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18mA			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8 V, I_{OH} = MAX	2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8 V, I_{OL} = 4mA		0.25	0.4	V
I_I	Input current at maximum input voltage: D, J, K	V_{CC} = MAX, V_I = 7V			0.1	mA
	Clear				0.2	
	Preset				0.2	
	Clock				0.1	
I_{IH}	High-level input current: D, J, K	V_{CC} = MAX, V_I = 2.7V			20	μA
	Clear				40	
	Preset				40	
	Clock				20	
I_{IL}	Low-level input current: D, J, K	V_{CC} = MAX, V_I = 0.4 V			−0.4	mA
	Clear				−0.8	
	preset				−0.8	
	Clock				−0.4	
I_{OS}	Short-circuit output current* Series 54LS	V_{CC} = MAX	−20		−100	mA
	Series 74LS		−20		−100	
I_{CC}	Supply current (Average per flip-flop)	V_{CC} = MAX, See Note 1		4	8	mA
	clock frequency	V_{CC} = 5V, T_A = 25°C, C_L = 15pF, R_L = 2kΩ	25	33		MHz
t_{PLH}	from clear, preset or clock (as appropriate) to Q or $\bar{Q}$			13	25	ns
t_{PHL}				25	40	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
* Not more than one output should be shorted at a time.
t_{PLH} = propagation delay time, low-to-high-level output.
t_{PHL} = propagation delay time, high-to-low-level output.
¶ The arrow indicates the edge of the clock pulse used for reference: ↑ for the rising edge.

Pin Assignments (Top View)

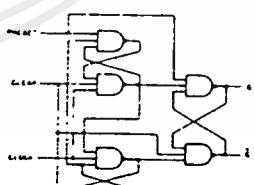


Functional Table

'74, 'H74, 'LS74, 'S74 (See Note 2)

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

Functional Block Diagram



'74, S74, H74, LS74, S74 DUAL D-FLIP-FLOP WITH CLEAR AND PRESET

NOTES: 1. With all outputs open, I_{CC} is measured with the Q and $\bar{Q}$ outputs high in turn. At the time of measurement, the clock input is grounded.
2. H = high level (steady state), L = low level (steady state), X = irrelevant, ↑ = transition from low to high level.
Q_0 = the level of Q before the indicated input conditions were established.
* This configuration is nonstable; that is, it will not persist when preset and clear inputs return to their inactive (high) level.

5493/7493 4-Bit Binary Counter

	Schottky TTL					High-Speed TTL					Low-Power Schottky TTL					Standard TTL					Low-Power TTL				
	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF	Device Type	Package C	P	M	CF
T I.											SN54LS93	J 1			W 1	SN5493A	J 1			W 1	SN54L93	J 2	P 2		T 2
FAIRCHILD											SN74LS93	J 1	N 1			SN7493A	J 1	N 1			SN74L93	J 2	N 2		T 2
MOTOROLA											FJH54LS93 FJH9LS93	D 1			F 1	FJH5493 FJH9393	D 1			F 1					
N. S. C											FC74LS93 FC9LS93	D 1	P 1			FJH7493 FC9393	D 1	P 1							
PHILIPS																MC5493	L 1								
SIGNETICS											SN74LS93		P 1			MC7493	L 1	P 1							
SIEMENS											DM54LS93		J			DM5493	J 1	N 1		W 1	DM54L93	J 2	P 2		F 2
FUJITSU											DM74LS93		J			DM7493	J 1	N 1			DM74L93	J 2	N 2		F 2
HITACHI											N74LS93		J			FJJ221 7493		J							
MITSUBISHI																S5493	F 1	A 1		W 1					
NEC											N74LS93		A 1			N7493	F 1	A 1							
TOSHIBA																FLJ181		J							
											HD74LS93		P 1			HD7493A/HD2520	J	P 1							
											M74LS93		P 1			M53293		P 1							
																μPB223	D 1	C 1							
																TD3493A		P 1							

Electrical Characteristics SN54LS93A SN74LS93A

absolute maximum ratings over operating free-air temperature range

Supply voltage, V_{CC}	7V	Operating free-air temperature range	SN54LS	−55°C to 125°C
Input voltage	7V		SN74LS	0°C to 70°C
Emitter-base voltage (see Note 1)	5.5V	Storage temperature range		−65°C to 150°C

recommended operating conditions

		SN54LS93A MIN	NOM	MAX	SN74LS93A MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}				−400			−400	μA
Low-level output current, I_{OL}				4			8	mA
Count frequency, f_{count}	A input	0		32	0		32	MHz
	B input	0		16	0		16	
Pulse width, t_w	A input	15			15			ns
	B input	30			30			
	Reset inputs	15			15			
Reset inactive-state setup time, t_{setup}		25			25			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

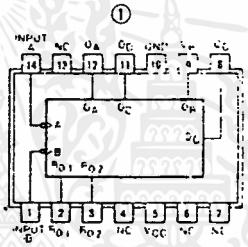
electrical characteristics over recommended operating free-air temperature range

PARAMETER		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_I	Input clamp voltage	V_{CC} = MIN, I_I = −18mA				−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = −400μA		2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = 8mA			0.35	0.5	V
I_I	Input current at maximum input voltage	Any reset	V_{CC} = MAX, V_I = 7V			0.1	mA
		A input	V_{CC} = MAX, V_I = 5.5V			0.2	
		B input				0.2	
I_{IH}	High-level input current	Any reset	V_{CC} = MAX, V_I = 2.7V			20	μA
		A input				80	
		B input				80	
I_{IL}	Low-level input current	Any reset	V_{CC} = MAX, V_I = 0.4V			0.4	mA
		A input				2.4	
		B input				1.6	
I_{OS}	Short-circuit output current *	V_{CC} = MAX	SN54LS	−20		−100	mA
			SN74LS	−20		−100	
I_{CC}	Supply current	V_{CC} = MAX, See Note 2			5	15	mA
f_{count}	From A to output Q_A	V_{CC} = 5V, T_A = 25°C, C_L = 15pF, R_L = 2kΩ		32	42		MHz
	From B to output Q_B			16			
t_{PLH}	From A to output Q_A				10	16	ns
t_{PHL}					12	18	
t_{PLH}	From A to output Q_D				46	70	ns
t_{PHL}					46	70	
t_{PLH}	From B to output Q_B				10	16	ns
t_{PHL}					14	21	
t_{PLH}	From B to output Q_C				21	32	ns
t_{PHL}					23	35	
t_{PLH}	From B to output Q_D				34	51	ns
t_{PHL}					34	51	
t_{PHL}	From Set-to-0 to Any output ¶				26	40	ns

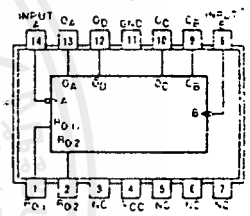
† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable type.
‡ All typical values are at V_{CC} = 5V, T_A = 25°C.
* Not more than one output should be shorted at a time.
¶ Outputs are tested at I_{OL} = 16mA plus the limit value for I_{IL} for the B input. This permits driving the B input while maintaining full fan-out capability.
f_{count} = maximum count frequency; t_{PLH} = propagation delay time, low-to-high-level output; t_{PHL} = propagation delay time, high-to-low-level output.

Pin Assignments (Top View)

①



②



positive logic: see function tables
NC No internal connection

Function Table

COUNT SEQUENCE
'93A,'LS93,'L93 (See Note 3,4)

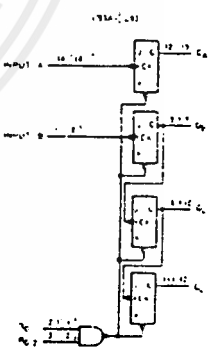
COUNT	OUTPUT Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

RESET/COUNT
'93A,'LS93,'L93 (See Note 4)

RESET INPUTS $R_{0(1)}$	$R_{0(2)}$	OUTPUT Q_D	Q_C	Q_B	Q_A
H	H	L	L	L	L
L	X	COUNT			
X	L	COUNT			

Functional Block Diagram

(see Note 5)



'93A,'LS93,'L93
4-BIT BINARY COUNTER

NOTES

1. This is the voltage between two emitters of a multiple-emitter transistor. For this circuit, this rating applies between the two $R_{0(1)}$ inputs.
2. I_{CC} is measured with all outputs open, both $R_{0(1)}$ inputs grounded following momentary connection to 4.5V, and all other inputs grounded.
3. Output Q_A is connected to input B.
4. H = high level, L = low level, X = irrelevant.
5. The J and K inputs shown without connection are for reference only and are functionally at a high level.