การใช้คอมพิวเตอร์กราฟฟิกควบคุมอุปกรณ์

COMPUTER GRAPHIC CONTROLLER

โดย

นายบุญฤทธิ์ เข็มดี

นายสาวิตร์ สุวรรณรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

หัวข้อปริญญานิพนธ์	**การใช้คอมพิวเตอร์กราฟฟิกควบคุมอุปกรณ์**	
	(COMPUTER GRAPHIC CONTROLLER)	
ชื่อนักศึกษา	**นายบุญฤทธิ์ เข็มดี**	**เลขประจำตัว 40012018**
	นายสาวิตร์ สุวรรณรัตน์	**เลขประจำตัว 40012035**
ภาควิชา	**เทคนิคอุตสาหกรรม**	
อาจารย์ที่ปรึกษา	**ผศ.ชวลิต เบญจางคประเสริฐ**	
ปีการศึกษา	**2541**	

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการตรวจสอบปริญญานิพนธ์

..... ประธานกรรมการ
( )
..... กรรมการ
( )
..... กรรมการ
( )
..... กรรมการ
( )
..... กรรมการ
( )

การใช้คอมพิวเตอร์กราฟฟิกควบคุมอุปกรณ์

โดย

นายบุญฤทธิ์ เข็มดี

นายสาวิตร์ สุวรรณรัตน์

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ ชวลิต เบญจางคประเสริฐ

บทคัดย่อ

ในโครงงานนี้ได้เสนอวิธีการออกแบบ การประยุกต์ใช้ไมโครคอมพิวเตอร์ และไมโครคอนโทรลเลอร์ MCS-51 โดยการออกแบบ HARDWARE และ SOFTWARE เพื่อสร้างระบบการปิด-เปิดอุปกรณ์ไฟฟ้าที่อยู่ในที่ต่างๆในระยะไกล หรือในห้องพักของอาคารสูง โดยใช้ระบบสื่อสารอนุกรมมาตรฐาน RS422 ใช้งานบนระบบปฏิบัติการวินโดว์ด้วยโปรแกรมเดลไฟล์(delphi)

COMPUTER GRAPHIC CONTROLLER

By

Mr.Boonyarit Khemdee

Mr.Sawit Suwanrat

Adviser

Asst.Prof. Chawalit Benjangkaprasert

Abstract

This paper presents the simple method design utilizing of hardware and software for microcomputer and microcontroller MCS-51. In this system can controlling of electrical equipment on-off in other place for a long distance or in high building. Using standard serial communication RS-422 and display on window system by Delphi Program.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งจากท่านอาจารย์ที่ปรึกษา และท่าน คณาจารย์ต่างๆของภาควิชาที่ได้กรุณาให้ยืมเครื่องมือที่ใช้ประกอบในการพัฒนาโครงงานนี้ บิดา-มารดา ที่ได้ให้กำลังใจและกำลังทรัพย์ในการทำตลอดมา และเพื่อน ๆ ทั้งในภาควิชาเทคนิคอุตสาหกรรมและภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

นายบุญฤทธิ์ เข็มดี

นายสาวิตร์ สุวรรณรัตน์

สารบัญ

สารบัญรูป

สารบัญตาราง

บทนำ

ในปัจจุบันนี้ประเทศไทยมีการเจริญเติบโตทางเศรษฐกิจอย่างรวดเร็วประชากรมีรายได้เพิ่มขึ้น รายได้ประชาชาติเพิ่มขึ้น ก่อให้เกิดความเจริญด้านวัตถุ มีการนำเอาเทคโนโลยีใหม่ๆมาใช้ทำให้ประชากรส่วนใหญ่มีความสะดวกสบายขึ้น ตัวอย่างเช่น เราสามารถควบคุมอุปกรณ์ไฟฟ้าต่างๆจากสถานที่ที่อยู่ห่างจากอุปกรณ์นั้นๆ โดยใช้คอมพิวเตอร์ ผ่านทางสายอนุกรม RS232/RS422 ซึ่งสามารถประหยัดสายได้อย่างมากและใช้ไมโครคอนโทรลเลอร์ MCS-51 เป็นตัวสั่งงาน

โครงงานนี้ได้พยายามให้ผู้ที่ไม่คุ้นเคยกับการใช้คอมพิวเตอร์ ก็สามารถสั่งงานปิด-เปิดอุปกรณ์ไฟฟ้าได้โดยง่าย โดยใช้รูปที่มีอยู่จริงเป็นการสื่อสารระหว่างผู้ใช้คอมพิวเตอร์ ผู้ใช้เพียงแต่คลิกที่เม้าส์ (Mouse) ก็สามารถสั่งงานเปิด-ปิดอุปกรณ์ไฟฟ้าได้ตามต้องการ

โครงงานนี้ยังสามารถนำไปประยุกต์ได้กับงานหลายแขนงเช่น การปิด-เปิดไฟฟ้าส่องสว่างตามทางเดินของอาคารสูง ๆ หรือ ในโรงงานที่มีห้องควบคุมอยู่ห่างจากอุปกรณ์มาก ๆ เป็นต้น

บทที่ 1

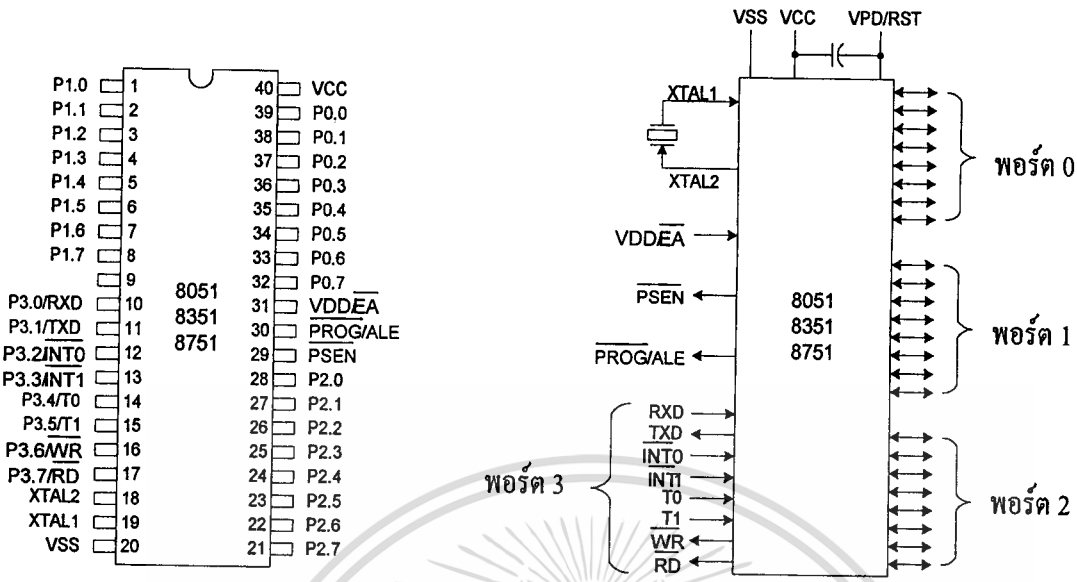
ทฤษฎีและหลักการ

1.1 คุณสมบัติของ MCS-51

- ต้องการแหล่งจ่ายไฟ + 5 V ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ 8031,8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์
- หน่วยความจำสำหรับโปรแกรมและดาต้า (Program Memory และ Data Memory แยกจากกันอย่างละ 64 ไบต์)
- คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 uS เมื่อทำงานที่ความถี่ 12 MHz
- มี Timer/Counter ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) ทั้ง รับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งได้ 4 โหมด
- มีคำสั่งในการทำ AND, OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิต และ 1 บิต

ตำแหน่งขาของMCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกันดังแสดงในรูปที่ 1.1



รูปที่ 1.1 แสดงตำแหน่งขาของ MCS-51

หน้าที่ในการใช้งานแต่ละขามีดังนี้

- ขา VSS (ขา 20) สำหรับต่อลงกราวด์
- ขา VCC(ขา 40) สำหรับต่อแหล่งจ่ายไฟกระแสตรง 5 โวลท์
- ขาพอร์ท 0 (ขา 32-39) มี 8 ขา ใช้เป็นขาสำหรับพอร์ท 0 ขนาด 8 บิต (P0.0-P0.7) แบบ Open Drain Bidirectional พอร์ทนี้สามารถใช้งานเป็นอินพุทเอาท์พุทพอร์ททั่วไปได้ โดยหากใช้งานเป็นอินพุทพอร์ท ต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ทนี้เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะเป็น high impedance) และพอร์ท 0 ยังใช้ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลภายนอกชิปด้วย โดยส่งค่าแอดเดรสบัสไบต์ต่ำ (A0-A7) และมัลติเพล็กซ์กับการรับส่งข้อมูล (D0-D7) จากหน่วยความจำภายนอกในระหว่างการเขียนหรืออ่านข้อมูลโดยมีวงจรพูลอัพภายใน
- ขาพอร์ท 1 (ขา1-8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ท 1 (P1.0-P1.7) สามารถใช้งานเป็นอินพุทหรือเอาท์พุทพอร์ททั่วไปได้ หากต้องการใช้งานเป็นอินพุทพอร์ท ต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ทนี้ เพื่อให้มีสถานะเป็น high impedance โดยมีวงจรพูลอัพภายใน
- ขาพอร์ท 2 (ขา 21- 28) มี 8 ขา ใช้เป็นขาสำหรับพอร์ท 2 (P2.0-P2.7) ขนาด 8 บิต แบบ open drain bidirectional พอร์ทนี้สามารถใช้งานเป็นอินพุทพอร์ทเอาท์พุทพอร์ททั่วไปได้ โดยหากใช้งานเป็นอินพุทพอร์ท ต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ทนี้ เพื่อบังคับ

ให้ขาอยู่ในสภาวะ high impedance นอกจากนี้ยังใช้ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมภายนอกด้วย โดยใช้สำหรับส่งค่าแอดเดรสไบต์สูง (A8-A15) และมีวงจรพูลอัพภายใน

- ขา พอร์ท 3 (ขา 10 – 17) มี 8 ขา ใช้เป็นขาสำหรับพอร์ท 3 (P3.0 - P3.7) สามารถใช้งานเป็นอินพุทเอาท์พุท ทั่วไปได้ หากต้องการเป็นอินพุทพอร์ทต้องโหลดค่า 1 ไปยัง แต่ละบิตของพอร์ทนี้ เพื่อให้มีสถานะเป็น high impedance โดยใช้วงจรพูลอัพภายใน นอกจากนี้ยังใช้งานในหน้าที่พิเศษต่างๆอีกหลายอย่างดังนี้

ขา P3.0 ใช้รับข้อมูลจากภายนอกแบบอนุกรม

ขา P3.1 ใช้ส่งข้อมูลออกไปภายนอกแบบอนุกรม

ขา P3.2 ใช้เป็นอินพุทเพื่อรับสัญญาณอินเตอร์รัพท์ชนิดที่ 0

ขา P3.3 ใช้เป็นอินพุทเพื่อรับสัญญาณอินเตอร์รัพท์ชนิดที่ 1

ขา P3.4 สัญญาณอินพุทให้เคาน์เตอร์ของไทม์เมอร์ 0

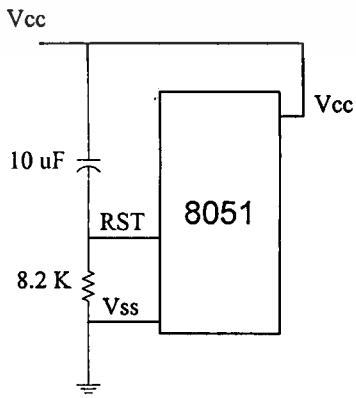
ขา P3.5 สัญญาณอินพุทให้เคาน์เตอร์ของไทม์เมอร์ 1

ขา P3.6 ใช้เป็นสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป

ขา P3.7 ใช้เป็นสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป

การใช้งานพอร์ท 3 ในหน้าที่พิเศษดังกล่าวนี้จะต้องโหลดค่า 1 ไปยังแต่ละบิตที่ต้องการใช้งานก่อนทุกครั้ง

ขา RST (ขา 9) ใช้สำหรับรีเซตวงจรทุกอย่างภายในชิป เพื่อเริ่มต้นการทำงานใหม่ดังรูป



รูปที่1.2 แสดงวงจรรีเซตของ MCS-51

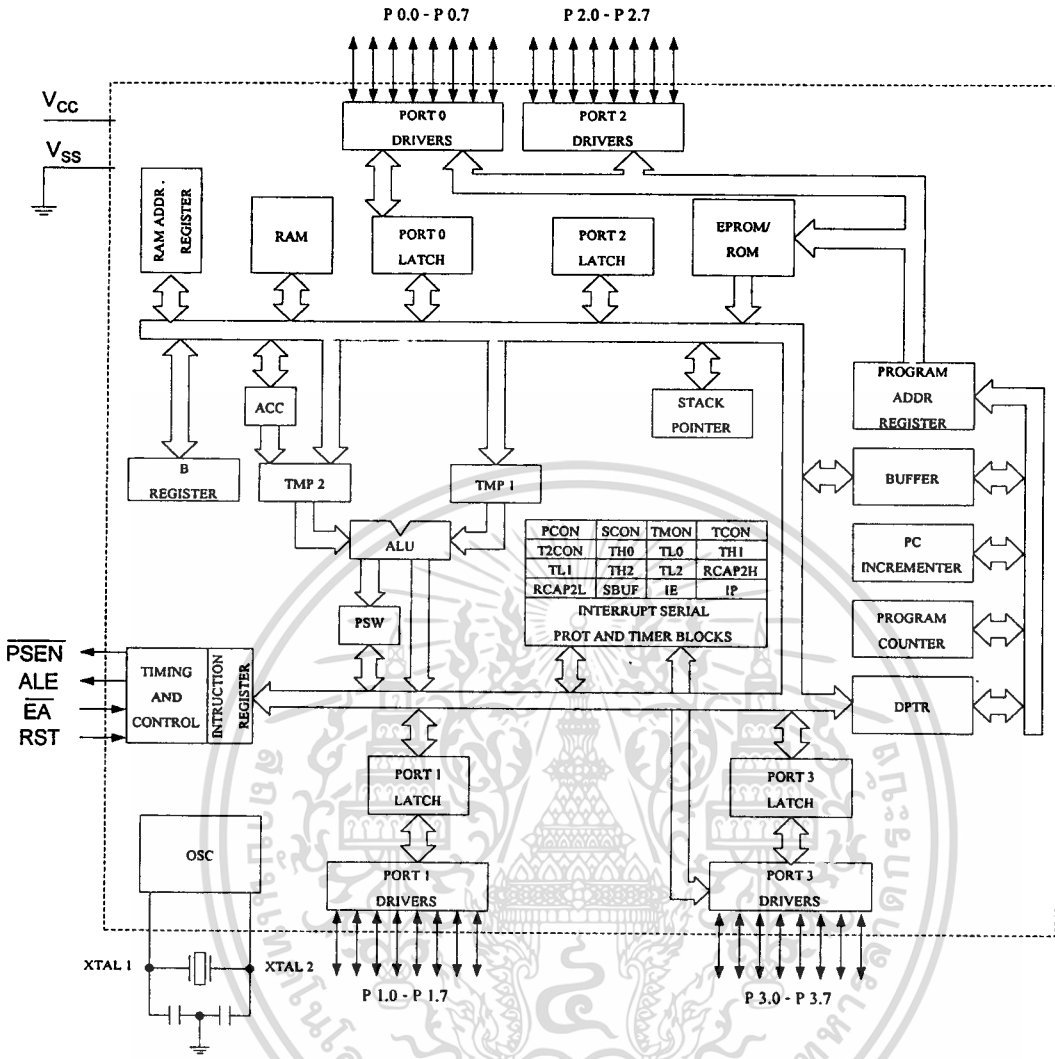
- ขา ALE/PROG (ขา 30) เป็นขาสำหรับใช้ส่งสัญญาณออกไปภายนอก เพื่อควบคุมการแลตซ์(latch) ค่าแอสเดรสไบต์ต่ำ จากพอร์ท 0 ในระหว่างการติดต่อสำหรับเก็บโปรแกรมหรือข้อมูลภายนอก ปกติเมื่อไม่มีการติดต่อหน่วยความจำภายนอกจะส่งสัญญาณพัลส์ออกมาด้วยความถี่ 1/8 ของความถี่ออสซิลเลเตอร์ ดังนั้นเราสามารถนำความถี่นี้ไปใช้งานอย่างอื่นได้
- ขา PSEN (ขา 29) ใช้ส่งสัญญาณสโตรปเพื่ออ่านคำสั่งจากโปรแกรมที่เก็บไว้ในหน่วยความจำภายนอกชิป (program strobe enable) เมื่อชิปทำงานด้วยโปรแกรมจากภายนอก ขานี้จะส่งสัญญาณสโตรป 2 ครั้งในแต่ละแมชชีนไซเคิล แต่ในช่วงการอ่านหรือเขียนจะไม่มีสัญญาณออกจากขานี้
- ขา EA/Vpp (ขา 31) เป็นขาที่ใช้เลือกให้ MCS-51 ทำงานจากโปรแกรมที่อยู่ภายในชิปหรืออยู่ภายนอกชิป โดยหากขานี้เป็น 0 ให้ใช้โปรแกรมจากภายนอกชิป แต่ถ้าเป็น 1 ให้ใช้โปรแกรมที่อยู่ภายในชิป สำหรับ MCS-51 ที่ไม่มีหน่วยความจำภายในชิปต้องต่อขานี้ลงกราวด์เสมอ

ขา XTAL 1 (ขา 19) ใช้ต่อกับคริสตอลภายนอก โดยเป็นอินพุทเข้าสู่วงจรออสซิลเลเตอร์

ขา XTAL 2 (ขา 18) ใช้ต่อคริสตอลภายนอก โดยเป็นเอาท์พุทจากวงจรออสซิลเลเตอร์

โครงสร้างภายในของ MCS-51

โครงสร้างภายในของชิปไมโครคอนโทรลเลอร์ MCS-51 ดังแสดงในรูปที่ 1.3

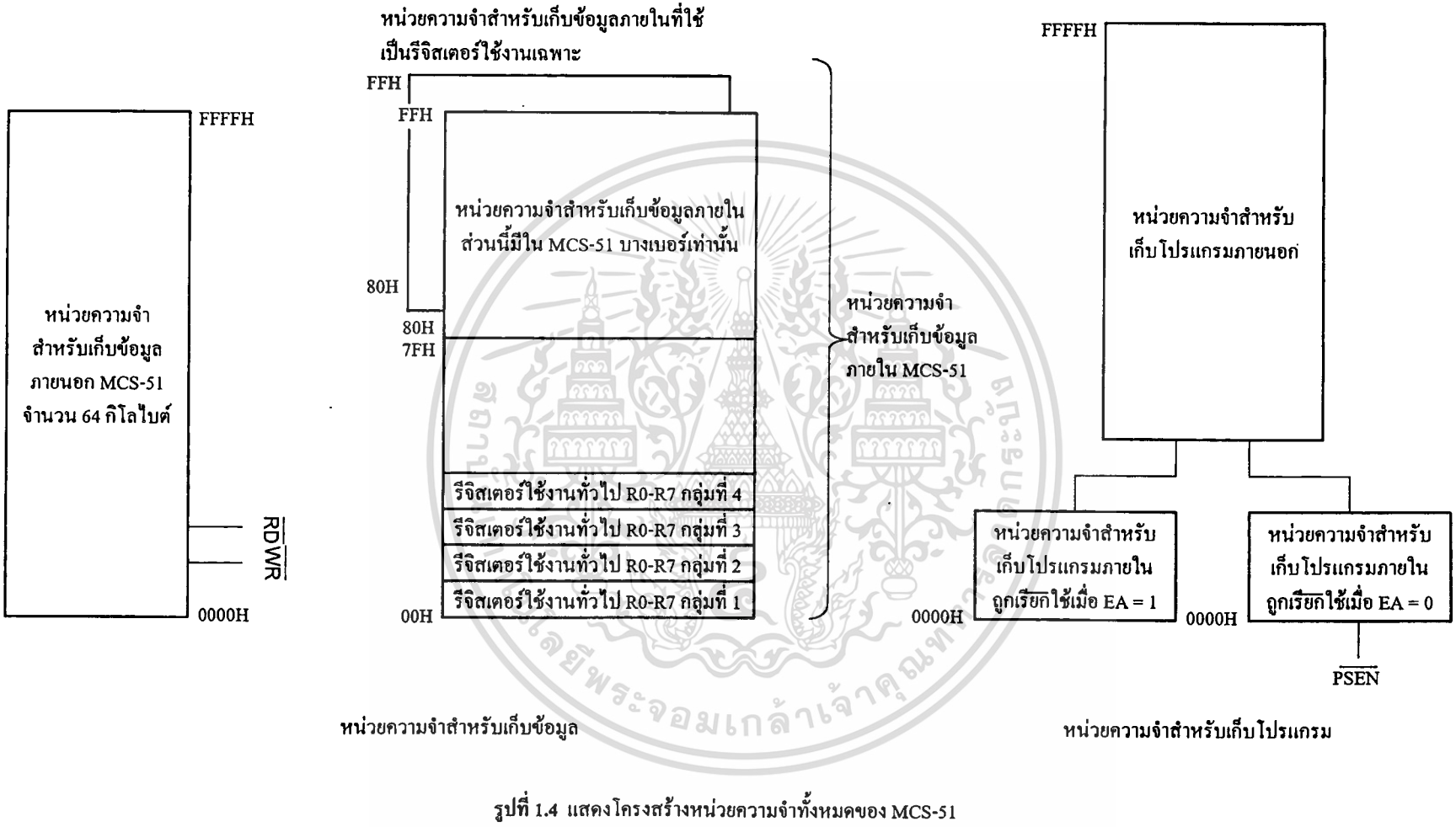


รูปที่1.3 แสดงโครงสร้างภายในชิปของ MCS-51

โครงสร้างภายในหน่วยความจำ MCS-51 ในไมโครคอนโทรลเลอร์ตระกูลนี้ทุกเบอร์แบ่งหน่วยความจำเป็น 2 ส่วน คือ

- หน่วยความจำสำหรับเก็บโปรแกรม (program memory)
- หน่วยความจำสำหรับเก็บข้อมูล (data memory)

สำหรับบางเบอร์จะไม่มีหน่วยความจำสำหรับเก็บภายในชิป ส่วนหน่วยความจำที่ 2 คือ หน่วยความจำเก็บข้อมูล ซึ่งใช้สำหรับเก็บเก็บข้อมูลระหว่างการทำงาน โครงสร้างภายในทั้งหมดของ MCS-51 มีดังแสดงในรูปที่1.4



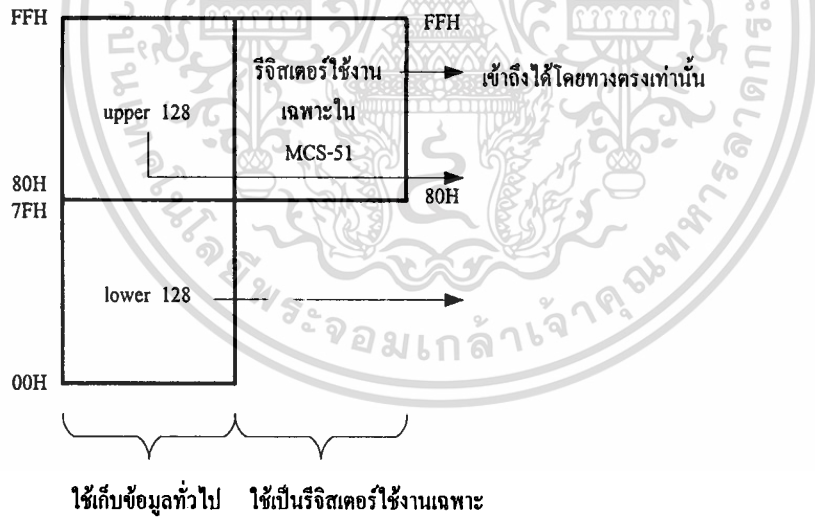
รูปที่ 1.4 แสดงโครงสร้างหน่วยความจำทั้งหมดของ MCS-51

หน่วยความจำสำหรับเก็บโปรแกรม หน่วยความจำสำหรับเก็บโปรแกรมใน MCS-51 จะแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำสำหรับเก็บไว้ในชิป (internal program memory) และหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป (external program memory) ซึ่งมีขนาดตั้งแต่ 0,4,8,16 กิโลไบต์

หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำของ MCS-51 จะแบ่งออกเป็น 2 ส่วนคือ หน่วยความจำสำหรับเก็บข้อมูลภายนอกชิป หน่วยความจำสำหรับเก็บข้อมูลภายในชิป ซึ่งยังแบ่งได้ย่อย ๆ ดังนี้

- ส่วนที่ใช้เก็บข้อมูลทั่วไป
- ส่วนที่ใช้เป็นรีจิสเตอร์เฉพาะงาน

หน่วยความจำส่วนที่เก็บข้อมูลทั่วไปภายในชิปเป็นหน่วยความจำสำหรับเก็บข้อมูลที่มีอยู่ภายใน MCS-51 หน่วยความจำส่วนนี้มีไว้สำหรับเก็บข้อมูลขณะทำงาน ส่วนหน่วยความจำสำหรับเก็บข้อมูลภายในชิปที่ใช้เป็นรีจิสเตอร์เฉพาะงานเพื่อควบคุมการทำงานและ บอกสถานะของซีพียู แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลดังแสดงในรูป1.5



รูปที่1.5 แผนภาพแสดงหน่วยความจำสำหรับเก็บข้อมูลภายในชิป

รีจิสเตอร์ใช้งานเฉพาะ เนื่องจาก MCS-51 ถูกออกแบบไว้ให้ใช้งานควบคุมระบบโดยเฉพาะจึงทำให้มีความสามารถหลายอย่าง ซึ่งจำเป็นต้องอาศัยวงจรภายในชิปเพิ่มขึ้น การควบคุมการทำงานของวงจรภายในไมโครคอนโทรลเลอร์จะกระทำผ่านรีจิสเตอร์ที่ถูกกำหนดหน้าที่ไว้แล้ว ดังนั้นหากต้องการใช้ MCS-51 ให้มีประสิทธิภาพ จำเป็นต้องทราบหน้าที่การทำงานของรีจิสเตอร์

ใช้งานเฉพาะแต่ละตัวให้ละเอียด รีจิสเตอร์ใช้งานเฉพาะทั้งหมดจะอยู่ในหน่วยความจำสำหรับเก็บข้อมูลภายในชิปบริเวณที่ใช้เป็นรีจิสเตอร์ใช้งานเฉพาะดังได้กล่าวมาแล้ว รีจิสเตอร์ทั้งหมดดังแสดงในตารางที่ 1

8 ไบต์

F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPL			PCON	87

ตารางที่ 1 แสดงโครงสร้างและตำแหน่งรีจิสเตอร์ใช้งานเฉพาะ

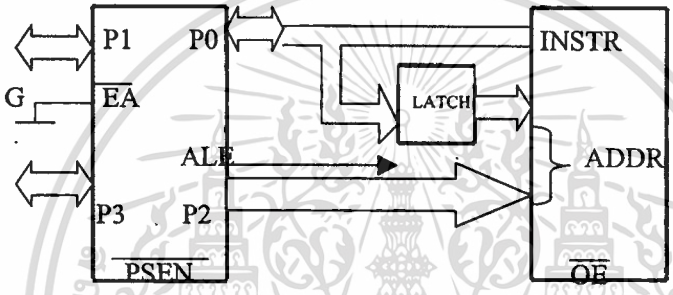
ในส่วนของหน่วยความจำสำหรับเก็บโปรแกรมและหน่วยความจำสำหรับเก็บข้อมูลที่อยู่ภายนอกชิป จะเป็นหน่วยความจำส่วนที่อยู่ภายนอกชิป MCS-51 ซึ่งผู้ใช้ต้องติดตั้งเพิ่มเอง การติดต่อระหว่าง MCS-51 กับหน่วยความจำทั้ง 2 ส่วนจะใช้ขา 32 ถึง 39 (พอร์ท 0) เป็นตัวส่งค่าแอสเดรสไบต์ต่ำ (A0-A7) และใช้รับส่งข้อมูลกับหน่วยความจำด้วย (ใช้เป็นดาต้าบัส) ส่วนค่าแอสเดรสไบต์สูง (A8-A15) จะใช้ขา 21 -28 (พอร์ท 2) ดังนั้นเมื่อพอร์ท 0 และพอร์ท 2 ถูกนำมาใช้ในการติดต่อกับหน่วยความจำภายนอก จะทำให้เหลือพอร์ทสำหรับใช้งานอื่นน้อยลง

การต่อหน่วยความจำเข้ากับ MCS-51

หากมีการเลือกใช้ชิปที่มีหน่วยความจำสำหรับเก็บโปรแกรมเป็น ROM หรือ EPROM ขนาด 16 กิโลไบต์อยู่ภายใน และขา $\overline{EA}$ ต่อกับ VCC ซีพียูจะเฟตช์คำสั่งในโปรแกรมตั้งแต่ตำแหน่ง 0000H–3FFFH จากหน่วยความจำที่เก็บอยู่ภายในชิป ส่วนคำสั่งในโปรแกรมตั้งแต่ตำแหน่ง 4000H-0FFFH จะถูกเฟตช์จากหน่วยความจำสำหรับเก็บภายนอกชิป

หากขา $\overline{EA}$ ต่อลงกราวด์ ซีพียูจะเฟตช์คำสั่งทั้งหมดจากหน่วยความจำสำหรับเก็บโปรแกรมที่อยู่ภายนอกชิป และสำหรับ MCS-51 ที่ไม่มีหน่วยความจำไว้สำหรับเก็บโปรแกรมภายในชิปจะต้องต่อขา $\overline{EA}$ ลงกราวด์เสมอ เพื่อให้ MCS-51 ทำงานได้อย่างถูกต้อง

สัญญาณควบคุมการเฟตช์คำสั่งกับโปรแกรมที่เก็บไว้ในหน่วยความจำสำหรับโปรแกรมภายนอกชิป(read strobe) คือ สัญญาณ $\overline{PSEN}$ ซึ่งจะต่อนำไปใช้งานกับขา $\overline{RD}$ ของหน่วยความจำสำหรับเก็บโปรแกรมภายนอก สัญญาณ $\overline{PSEN}$ จะไม่ถูกใช้งานเมื่อ MCS-51 ทำงานจากโปรแกรมซึ่งอยู่ในหน่วยความจำสำหรับเก็บโปรแกรมภายในชิป การใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมภายนอกชิป มีดังแสดงในรูปที่ 2.



รูปที่1.6 แสดงการใช้หน่วยความจำสำหรับเก็บข้อมูลภายนอก

ในรูปที่1.6 แสดงการใช้โปรแกรมจากหน่วยความจำสำหรับเก็บโปรแกรมที่อยู่ภายนอกชิป MCS-51 โดยใช้พอร์ท 0 สำหรับส่งค่าแอดเดรสบัสไบต์ต่ำ (ส่งค่าแอสเดรส A0-A7 หรือ PCL) และพอร์ต 2 สำหรับส่วนค่าแอดเดรสไบต์สูง (A8-A15 หรือ PCH) รวมทั้งสิ้น 16 เส้น

พอร์ท 0 จะทำหน้าที่ในการส่งแอดเดรสไบต์ต่ำ (A0-A7) และใช้เป็นดาต้าบัส (D0-D7) ด้วย โดยหน้าที่ทั้งสองถูกใช้คนละเวลา (time multiplex) ในตอนแรกพอร์ท 0 จะส่งค่าแอสเดรสไบต์ต่ำออกมา จากนั้นขาที่ส่งค่าแอดเดรสจะมีสถานะ high impedance เพื่อรอรับข้อมูลที่ส่งมาจากหน่วยความจำภายนอก

ขณะที่พอร์ท 0 ส่งค่าของแอสเดรสไบต์ต่ำออกมา ขา ALE (Address Latch Enable) จะแอกทีฟโดยการส่งสัญญาณไปยังชิปที่มีหน้าที่แลทช์ค่าแอสเดรสไบต์ต่ำ เพื่อให้เริ่มแลตช์ค่าแอสเดรสไบต์ต่ำไว้ ชิปที่ทำหน้าที่นี้คือ 74LS373 หรือเบอร์ที่มีคุณสมบัติเดียวกันก็ได้

ในขณะที่สัญญาณ ALE แอกทีฟ พอร์ท 2 จะเริ่มส่งค่าแอสเดรสไบต์สูงไปยังหน่วยความจำสำหรับเก็บโปรแกรมภายนอก (PCH) จากนั้นสัญญาณ $\overline{PSEN}$ จะเริ่มแอกทีฟโดยการส่งสัญญาณ

สโตรปไปให้หน่วยความจำที่เก็บโปรแกรมภายนอก เพื่อให้เริ่มส่งข้อมูลซึ่งเป็นคำสั่งมายัง MCS-51

หน่วยความจำสำหรับเก็บโปรแกรมจะใช้แอสเดรสในการติดต่อขนาด 16 บิตเสมอ ถึงแม้ขนาดของโปรแกรมจะยาวไม่ถึง 64 กิโลไบต์ โดยค่าแอสเดรสจะส่งผ่านพอร์ท 0 และพอร์ท 2 ของ MCS-51

รีจิสเตอร์สำหรับใช้งานทั่วไป มีรีจิสเตอร์ที่ใช้สำหรับงานทั่วไปที่ผู้เขียนโปรแกรมสามารถนำมาใช้ได้คือ รีจิสเตอร์ A,B และรีจิสเตอร์ใช้งานทั่วไป R0-R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิปบริเวณ 128 ไบต์แรก รีจิสเตอร์ใช้งานทั่วไป R0-R7 ในMCS-51 มีอยู่ด้วยกันทั้งหมด 4 กลุ่ม แต่ละกลุ่มประกอบด้วยรีจิสเตอร์จำนวน 8 ตัว (R0-R7) ซึ่งมีชื่อเรียกเหมือนกัน ดังนั้นจึงมีอยู่ด้วยกันทั้งหมด 32 ตัว ในการทำงานขณะใดๆนั้นจะถูกเลือกมาใช้งานเพียงกลุ่มเดียวเท่านั้น

โครงสร้างพอร์ท MCS-51 ทุกเบอร์จะมีพอร์ทขนาด 8 บิต จำนวน 4 พอร์ท (P0,P1,P2,P3) โดยสามารถกำหนดให้ทำงานแบบพอร์ทขนานขนาด 8 บิต 4 พอร์ท หรือจะใช้เป็นพอร์ทขนาด 1 บิต ได้ถึง 32 พอร์ท ทั้งนี้ผู้ใช้ยังสามารถกำหนดให้แต่ละพอร์ทใช้งานเป็นอินพุทหรือเอาต์พุทอย่างใดอย่างหนึ่งได้อย่างอิสระ

ในกรณีที่ผู้ออกแบบต้องการใช้หน่วยความจำภายนอก ไม่ว่าจะเป็นหน่วยความจำสำหรับเก็บข้อมูลหรือสำหรับโปรแกรม พอร์ท 0 จะถูกกำหนดการใช้งานเป็นดาต้าบัสและแอสเดรสบัสไบต์ต่ำ ส่วนพอร์ท 2 จะถูกกำหนดการใช้งานเป็นตัวส่งค่าแอสเดรสไบต์สูง และบางส่วนของพอร์ท 3 จะถูกส่งสัญญาณควบคุมบัส (สัญญาณที่ใช้ควบคุมการอ่านหรือเขียนข้อมูล) แต่หากหน่วยความจำที่ใช้ภายนอกต้องการไม่ถึง 64 กิโลไบต์ พอร์ท 2 ที่ใช้เป็นแอสเดรสบัสไบต์สูงจะไม่ถูกนำมาใช้ทั้งหมด แต่พอร์ท 0 จะถูกใช้ทั้งหมด 8 เส้น เพระต้องการใช้เป็นดาต้าบัส ส่วนพอร์ท 3 จะนำมาใช้ติดต่อหน่วยความจำด้วยหรือไม่นั้นขึ้นอยู่กับหน่วยความจำที่ใช้ภายนอกว่ามีหน่วยความจำส่วนที่ใช้เก็บข่อมูลด้วยหรือไม่ ดังนั้นในการออกแบบระบบหากต้องการใช้หน่วยความจำภายนอกขึ้นเพียงใดก็จะยิ่งทำให้เหลือพอร์ทที่จะนำมาใช้งานลดลง

ไทม์เมอร์/เคาน์เตอร์ ใน MCS-51 มีรีจิสเตอร์ใช้งานเฉพาะที่สามารถนับจำนวนสัญญาณนาฬิกาหรือแมชชีนไซเคิลของวงจรออสซิลเลเตอร์ภายใน (ทำงานเป็นไทม์เมอร์) หรือนับจำนวนครั้งของการเปลี่ยนแปลงสถานะของสัญญาณภายนอก (นับจำนวนพัลส์ภายนอก) ที่ขา T0,T1 ของ

พอร์ท 3 (ทำงานเป็นเคาน์เตอร์) รีจิสเตอร์ที่ใช้งานเป็นไทม์เมอร์ และเคาน์เตอร์มีขนาด 16 บิต จำนวน 2 ตัว คือ รีจิสเตอร์ไทม์เมอร์ 0 และรีจิสเตอร์ไทม์เมอร์ 1 ตามลำดับ เมื่อต้องการใช้งานไทม์เมอร์ 0 หรือ ไทม์เมอร์ 1 จะต้องโหลดค่าที่ต้องการนับไปไว้ในรีจิสเตอร์ไทม์เมอร์ 0 หรือไทม์เมอร์ 1 เมื่อนับได้จำนวนครบตามที่ตั้งไว้แล้วก็จะส่งสัญญาณอินเตอร์รัพท์เพื่อบอกให้ซีพียูทราบ

พอร์ทสื่อสารข้อมูลแบบอนุกรม MCS-51 สามารถรับส่งสัญญาณแบบอนุกรมได้โดยไม่ต้องพึ่งอุปกรณ์ภายนอกอื่น ๆ แต่อย่างใด ในด้านอัตราเร็วของการรับส่งข้อมูลก็สามารถกำหนดค่าได้ตามความต้องการของผู้ใช้ โดยสามารถ เลือกอัตราเร็วในการรับส่งข้อมูล (baurd rate) มาตรฐานได้ตั้งแต่ 110 ,1.2K, 4.8 K ,9.6K,19.2K,375K ตามมาตรฐานของ UART

การใช้งานพอร์ทอนุกรมของ MCS-51

โดยปกติแล้ว MCS-51 คือระบบคอมพิวเตอร์ ซึ่งมีความสามารถในการรับส่งข้อมูลจากภายนอกและนำมาประมวลผล พร้อมทั้งนำสัญญาณต่างๆออกไปควบคุมอุปกรณ์ภายนอกได้

รีจิสเตอร์ที่เกี่ยวข้องในการใช้งานพอร์ทอนุกรม

1. รีจิสเตอร์ควบคุมไทม์เมอร์ สิ่งที่ต้องคำนึงถึงในการใช้งานพอร์ทอนุกรมคือ อัตราการรับส่งข้อมูล หรืออัตราบอด (Baud Rate) คือจังหวะการเลื่อนข้อมูลเข้าหรือ ออกจาก MCS-51 โดยสามารถสร้างจากไทม์เมอร์แชลแนล 1 โดยทำงานในโหมด 2 รีจิสเตอร์ที่ต้องการทำโปรแกรม มีดังนี้

-TMOD ตำแหน่ง 89H ทำหน้าที่เลือกโหมดของไทม์เมอร์

-TCON ตำแหน่ง 88H ทำหน้าที่เริ่มต้นสร้างบอด

-TH1 ตำแหน่ง 8CH ทำหน้าที่ใส่ข้อมูลการนับของไทม์เมอร์ 1 เพื่อสร้างบอด

2. รีจิสเตอร์ควบคุมการลดกำลัง เนื่องจาก การสร้างบอดนั้นต้องนำบิตในรีจิสเตอร์ PCON มาใช้ในการคำนวณข้อมูลของ TH1 ดังนั้นรีจิสเตอร์ที่ใช้คือ

-PCON ตำแหน่ง 87H ทำหน้าที่ในการคำนวณข้อมูลที่จะใส่ในรีจิสเตอร์ TH1 ดังตารางที่ 2

7	6	5	4	3	2	1	0
SMOD	-	-	-	GF1	GF0	PD	IDL

ตารางที่ 2 แสดงตำแหน่งบิตในรีจิสเตอร์ PCON

บิต	สัญลักษณ์	รายละเอียด
7	SMOD	เป็นบิตที่ใช้แก้ไขอัตราการบอด
6-4	-	ไม่ใช้งาน
3	GF1	แฟล็กใช้งานทั่วไป
2	GF0	แฟล็กใช้งานทั่วไป
1	PD	บิตที่แสดงการลดลงของกำลังไฟ
0	IDL	บิตที่แสดงในโหมดไอเดิล

3. รีจิสเตอร์ควบคุมการอินเตอร์รัพท์ มีรีจิสเตอร์ที่เกี่ยวข้องดังนี้

-IE ตำแหน่ง A8H ทำหน้าที่ยอมให้เกิดการอินเตอร์รัพท์

-IP ตำแหน่ง B8H ทำหน้าที่จัดลำดับความสำคัญของการอินเตอร์รัพท์

4. รีจิสเตอร์ควบคุมพอร์ทอนุกรม ขึ้นอยู่กับรีจิสเตอร์โดยตรงคือ

-SBUF ตำแหน่ง 99H ทำหน้าที่เป็นบัฟเฟอร์การรับหรือส่งข้อมูล

-SCON ตำแหน่ง 98H ทำหน้าที่ควบคุมและกำหนดโหมดการใช้งานพอร์ทอนุกรมทั้งหมด ดังตารางที่ 3

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

ตารางที่ 3 แสดงตำแหน่งบิตภายในรีจิสเตอร์ SCON

บิต	สัญลักษณ์	รายละเอียด
7	SM0	โหมดของพอร์ทอนุกรมบิต 0 ทำการเซ็ต โดยใช้โปรแกรมสั่งงาน
6	SM1	โหมดของพอร์ทอนุกรมบิต 1 ทำการเซ็ต โดยใช้โปรแกรมสั่งงานเช่นกัน

SM0	SM1	โหมด	รายละเอียด
0	0	0	รีจิสเตอร์แบบเลื่อนบิต; อัตราการส่ง = f/12
0	1	1	UART ชนิด 8 บิต; อัตราการส่งเปลี่ยนแปลงได้
1	0	2	UART ชนิด 9 บิต; อัตราการส่ง= f/32 หรือ f/64
1	1	3	UART ชนิด 9 บิต; อัตราการส่งเปลี่ยนแปลงได้

บิต	สัญลักษณ์	รายละเอียด
5	SM2	ใช้เป็นบิตแสดงการติดต่อระหว่างไมโครโปรเซสเซอร์ ในกรณีนี้ใช้เฉพาะโหมด 2 และ 3 เมื่อบิตถูกเซ็ตเป็น 1การอินเตอร์รัพท์จะเกิดขึ้น
4	REN	บิตอีนาเบิ้ลการรับ บิตจะถูกเซ็ตเป็น1เมื่อต้องการสัญญาณอนุกรม
3	TB8	ใช้ว่าจะเลือกส่ง 8 บิตหรือไม่ใช้ สำหรับในโหมด 2 หรือ 3 บิตหยุดในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้บิตนี้
2	RB8	ใช้ว่าจะเลือกรับ 8 บิตหรือไม่ใช้ สำหรับในโหมด 2 หรือ 3 บิตหยุดในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้บิตนี้
1	TI	แฟลกอินเตอร์รัพท์เมื่อส่งข้อมูลในโหมด 0 จะถูกเซ็ตเป็น 1 หลังจากส่งบิต 7 ไปแล้ว
0	RI	แฟลกอินเตอร์รัพท์เมื่อรับข้อมูลในโหมด 0 จะถูกเซ็ตเป็น 1 หลังจากรับบิต 7 เข้ามาแล้ว

การกำหนดโหมดในการใช้งาน

การกำหนดโหมดการใช้งานพอร์ทสื่อสารอนุกรมในโหมดต่าง ๆ ดังตารางที่ 4

โหมด	SCON	SM2 VARIATION
0	10H	Single Processor Environment (SM2 = 0)
1	50H	
2	90H	
3	D0H	
0	NA	Multiprocessor Environment (SM2 = 1)
1	70H	
2	B0H	
3	F0H	

ตารางที่ 4 รูปตารางแสดงการใช้งานพอร์ทสื่อสารอนุกรมโหมดต่างๆ

วิธีการคำนวณและกำหนดค่า baud rate ในการใช้พอร์ทสื่อสารอนุกรมโหมดต่างๆ

โหมด 0 ค่า baud rate ถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ โดยไม่จำเป็นต้องมีการกำหนดหรือใช้รีจิสเตอร์ที่ใช้เป็นไทม์เมอร์หรือเคาน์เตอร์แต่อย่างใด ดังนั้น baud rate ของพอร์ทสื่อสารอนุกรมในโหมดนี้สามารถเขียนเป็นสมการได้ง่ายๆดังนี้

$$\text{baud rate ในโหมด 0} = \frac{\text{ความถี่ออสซิลเลเตอร์}}{12}$$

โหมด 1 ค่า baud rate สามารถแปรค่าได้โดยการกำหนดไทม์เมอร์ 1 หรือ ไทม์เมอร์ 2 (มีใน 8052) ดังนี้

การใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate เมื่อใช้ไทม์เมอร์ 1 เป็นตัวกำหนด baud rate จะใช้ไทม์เมอร์ 1 ในโหมด 2 (Auto-Reload) โดยมีสูตรในการคำนวณค่า baud rate ดังนี้

$$\text{baud rate ในโหมด 1} = \frac{2^{SMOD} \times \text{ความถี่ออสซิลเลเตอร์}}{32 \times 12 \times [256 - (TH1)]}$$

ในการใช้งานทั่วไป เรามักจะทราบว่าต้องการใช้ baud rate ค่าเท่าใด ดังนั้นค่าที่เราต้องการหาก็คือค่าที่ต้องโหลดไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 สมการในการคำนวณหาค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ TH1 เมื่อทราบค่า baud rate คือ

$$TH1 = 256 - \frac{SMOD \times \text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{384 \times \text{buad rate}}$$

ค่าในรีจิสเตอร์ใช้งานเฉพาะ TH1 จำเป็นต้องเป็นเลขจำนวนเต็ม การปัดเศษที่ได้จากการคำนวณทิ้งหรือปัดขึ้นอาจทำให้ไม่ได้ค่า baud rate ได้ตามต้องการ วิธีแก้ปัญหานี้คือ การเปลี่ยนค่าความถี่ของคริสตอลที่ใช้

ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL  PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCON ไม่สามารถเข้าถึงได้ในระดับบิต

การใช้ไทม์เมอร์ 2 เป็นตัวกำหนด baud rate ในไทม์เมอร์ 2 มีการทำงานที่ใช้สำหรับเป็นตัวกำหนด baud rate อยู่แล้ว

เมื่อใช้สัญญาณนาฬิกาจากภายนอกเป็นอินพุทเป็นอินพุทให้แก่ไทม์เมอร์ 2 ที่ขา T2(P1.0) ค่า baud rate สามารถคำนวณได้ดังนี้

$$\text{baud rate} = \frac{\text{อัตราการเกิด overflow ของไทม์เมอร์ 2}}{16}$$

เมื่อใช้สัญญาณนาฬิกาจากภายในชิป(กำหนดจากความถี่ออสซิลเลเตอร์) ค่า baud rate สามารถคำนวณได้ดังนี้

$$\text{baud rate} = \frac{\text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{32 \times [65535 - (\text{RCAP2H, RCAP2L})]}$$

หากต้องการหาค่าที่จะโหลดไปไว้ในรีจิสเตอร์ใช้งานเฉพาะ RCAP2H,RCAP2L เมื่อทราบค่า baud rate ที่ต้องการ จะหาได้จากสมการต่อไปนี้

$$\text{RCAP2H,RCAP2L} = 65535 - \frac{\text{ความถี่ออสซิลเลเตอร์ที่ใช้}}{32 \times \text{baud rate}}$$

โหมด 2 ค่า baud rate มีให้เลือกได้เพียง 2 ค่า ขึ้นอยู่กับการกำหนดค่าบิต SMOD ในรีจิสเตอร์ใช้งานเฉพาะ PCON ดังนี้

บิต SMOD = 0 : baud rate จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้

บิต SMOD = 1 : baud rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ที่ใช้

ในโหมดนี้ไม่จำเป็นต้องใช้รีจิสเตอร์สำหรับใช้งานเป็นไทมเมอร์หรือเคาน์เตอร์เพื่อกำหนดค่า baud rate แต่อย่างใด

ในการเซตบิต SMOD ให้ใช้คำสั่งต่อไปนี้

```
ORL  PCON,#10000000B
```

เนื่องจากรีจิสเตอร์ใช้งานเฉพาะ PCON ไม่สามารถเข้าถึงได้ในระดับบิต

สูตรการคำนวณค่า baud rate ในโหมด 2 มีดังนี้

$$\text{baud rate ในโหมด 2} = \frac{\left[2^{(\text{SMOD})} \times \text{ความถี่ออสซิลเลเตอร์ที่ใช้}\right]}{64}$$

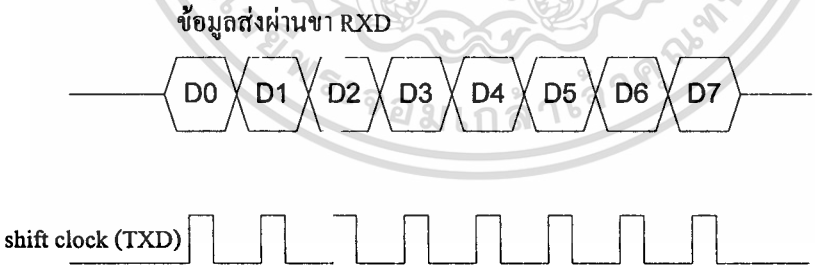
หากใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์ baud rate สูงสุดในการทำงานในโหมดนี้คือ 375K

โหมด 3 ค่า baud rate ในโหมดนี้คำนวณและกำหนดได้จากวิธีเดียวกันกับในโหมด 1

พอร์ทสื่อสารข้อมูลแบบอนุกรมใน MCS-51

การใช้งานพอร์ทสื่อสารข้อมูลแบบอนุกรมใน MCS-51 มีความสะดวกและคล่องตัวสูง ทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภท โดยสามารถกำหนดได้จากค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON ดังแสดงในรูปที่ 1.7 การใช้งานที่แตกต่างกัน 4 ประเภทนี้มีจุดประสงค์เพื่อความคล่องตัวในการรับส่งข้อมูลแบบอนุกรมแต่ละประเภทดังนี้

โหมด 0 การทำงานของพอร์ทสื่อสารอนุกรมในโหมด 0 ขา RXD จะใช้สำหรับรับและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ shift clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูล (ข้อมูลจะถูกรับหรือส่งตามจังหวะของสัญญาณ clock) ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิต โดยเริ่มรับและส่งบิตไบต์ต่ำก่อน (LSB first) อัตราการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ทสื่อสารข้อมูลแบบอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูล และบิตสิ้นสุดของข้อมูล เพราะจังหวะการรับส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock แล้ว

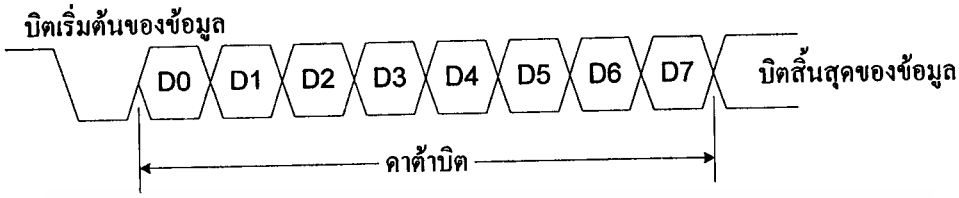


สัญญาณควบคุมจังหวะการรับ-ส่งข้อมูลในโหมด 0 ส่งผ่านขา TXD

รูปที่ 1.7 แสดงข้อมูลที่รับและส่งในการทำงานของพอร์ท

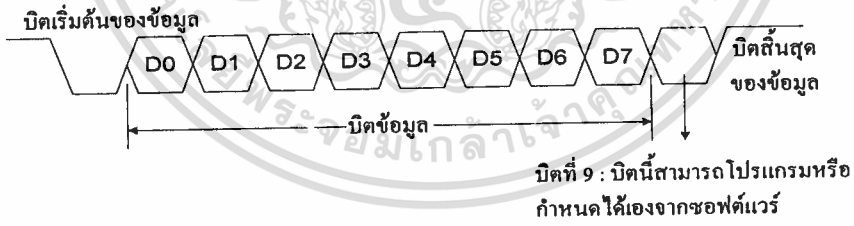
โหมด 1 การทำงานแบบที่สอง หรือการทำงานในโหมดที่ 1 มีการรับและส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งออกไปภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิตประกอบไปด้วยบิตเริ่มต้นข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับข้อมูล ค่าในบิตสิ้นสุด

ของข้อมูลที่รับได้จะไปอยู่ในบิต RB8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ทสื่อสารแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้



รูปที่ 1.8 การรับและส่งข้อมูลอนุกรมในโหมด 1

โหมด 2 การทำงานแบบที่ สาม หรือการทำงานในโหมด 2 จะมีการรับและส่งข้อมูลครั้งละ 11 บิต ข้อมูลจะถูกส่งออกภายนอกผ่านทางขา TXD และรับข้อมูลเข้ามาผ่านทางขา RXD ข้อมูลที่รับและส่งเข้ามาทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นข้อมูล 1 บิต บิตข้อมูล 8 บิต ตามด้วยบิตที่ 9 (ต่อจากบิตข้อมูลบิตสุดท้าย) ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็น 0 หรือ 1 ก็ได้ (programable 9th data bit) และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล ดังนั้นจำนวนบิตที่รับและส่งทั้งหมด 11 บิต ประกอบด้วยบิตต่าง ๆ ดังนี้



รูปที่ 1.9 แสดงข้อมูลรับและส่งข้อมูลอนุกรมโหมด 2 และ 3

โหมด 3 การทำงานของพอร์ทสื่อสารอนุกรมแบบสุดท้าย คือการทำงานในโหมดที่ 3 ในการทำงานโหมดนี้ข้อมูลจำนวน 11 บิต ถูกส่งผ่านขา TXD และถูกรับเข้ามาทางขา RXD ข้อมูลทั้ง 11 บิต ประกอบด้วยบิตเริ่มต้นข้อมูล 1 บิต บิตข้อมูล 8 บิต ตามด้วยบิตที่ 9 ซึ่งสามารถกำหนดค่าได้เหมือนในโหมด 2 และบิตสุดท้ายคือบิตสิ้นสุดของข้อมูล อัตราการส่งและรับข้อมูลสามารถเปลี่ยนแปลงได้ ดังนั้นจะเห็นได้ว่ารูปแบบการรับและส่งข้อมูลในโหมด 3 จะเหมือนกับในโหมด 2 ทุกอย่าง แต่ในโหมดนี้สามารถกำหนดอัตราเร็วในการรับส่งข้อมูลได้ตามความต้องการของผู้ใช้

การทำงานของพอร์ทสื่อสารอนุกรมจะเริ่มทำงานทันทีเมื่อมีคำสั่งใด ๆ ที่ใช้รีจิสเตอร์ใช้งานเฉพาะ SBUF เป็นรีจิสเตอร์ปลายทาง (destination register) เช่น

MOV SBUF,A

ส่วนในการรับข้อมูลจะเริ่มเมื่อมีเงื่อนไขดังนี้

- ในโหมด 0 เริ่มเมื่อค่าในบิต RI = 0 และบิต REN = 1
- ในโหมดอื่น ๆ การรับข้อมูลเริ่มเมื่อ MCS-51 ได้รับข้อมูลเข้ามา โดยที่บิต REN ในขณะนั้นต้องมีค่าเป็น 1

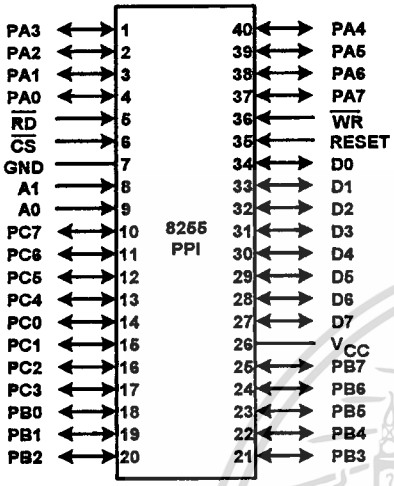
1.2 ทฤษฎีและการใช้งาน 8255 เบื้องต้น

ไมโครโปรเซสเซอร์นั้นนอกจากติดต่อกับหน่วยความจำโดยการนำข้อมูลไปเก็บไว้หรืออ่านข้อมูลใดๆ ออกจากหน่วยความจำแล้วตัว CPU เองอาจจะต้องติดต่อกับส่วนประกอบภายนอกอื่น ๆ อีก ด้วย เช่น การรับคีย์บอร์ดการแสดงผล หรือแม้แต่การนำ CPU ไปควบคุมอุปกรณ์ต่าง ๆ นั้น CPU ต้องติดต่อ (รับหรือส่งข้อมูล) โดยผ่านทางอินพุทหรือเอาท์พุทพอร์ทซึ่งอาจใช้ไอซี TTL บางเบอร์ มาใช้เป็นพอร์ทสำหรับ CPU ได้ แต่ทั้งนี้การใช้ไอซี TTL มีข้อจำกัดหลายอย่าง เช่น ในกรณีที่มีความจำเป็นจะต้องใช้พอร์ทหลาย ๆ พอร์ท เพราะต้องติดต่อกับอุปกรณ์ภายนอกหลายจุด จึงต้องใช้ไอซีเหล่านี้จำนวนหลายตัวและอาจทำให้ยากในการออกแบบวงจรอีกทั้งไม่สามารถจะเปลี่ยนแปลงลักษณะการทำงานมห้มีความแตกต่างไปจากเดิม ที่ได้ออกแบบไว้แล้ว ดังนั้นผู้ผลิต CPU ในตระกูลต่าง ๆ จึงมักจะผลิตไอซีประเภท LSI ที่ทำหน้าที่เป็น พอร์ทมาเพื่อใช้ CPU เบอร์นั้น ๆ ได้สะดวกซึ่งจะทำให้การรับส่งข้อมูลมีความเชื่อถือ และยังสามารถเปลี่ยนแปลงชนิดของพอร์ท (จากอินพุทเป็นเอาท์พุทหรือจากเอาท์พุทเป็นอินพุท) ได้ง่ายโดยการควบคุมของ CPU ไอซีที่ทำหน้าที่เป็นอินพุทและเอาท์พุทพอร์ท ซึ่งเป็นที่นิยมในการนำไปใช้งานมากที่สุดอีกทั้งยังมีราคาถูกหาซื้อได้ง่ายคือไอซีเบอร์ 8255 ของบริษัทอินเทล ซึ่งสามารถนำมาใช้งานกับ Z-80 และ CPU เบอร์อื่น ๆ ได้

ลักษณะเบื้องต้น

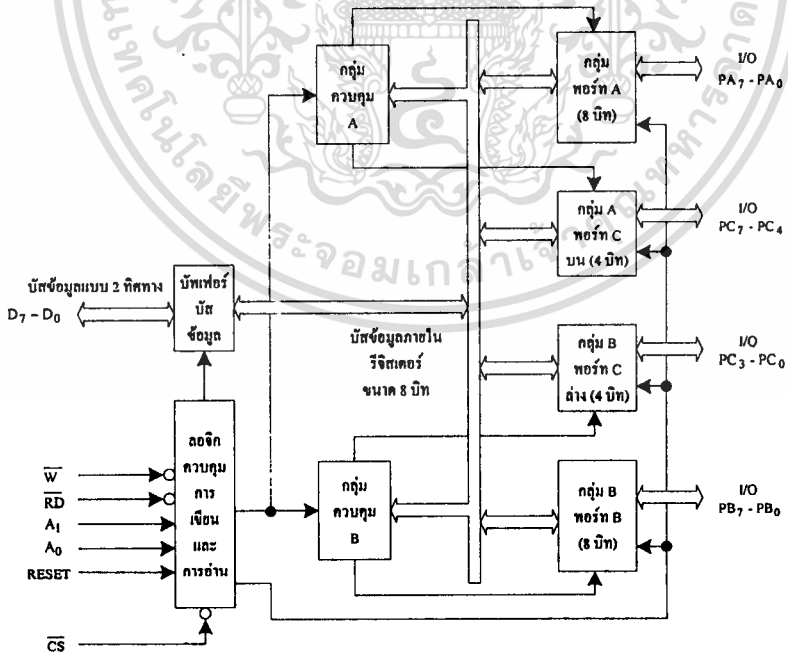
8255 นั้นเป็นไอซี LSI ขนาด 40 ขา จากรูปที่ 1.10a แสดงตำแหน่งของขาต่าง ๆ ทั้ง 40 ขา ส่วนรูป 1.10b แสดงแผนผังภายในของ 8255 ซึ่ง 8255 นี้พอร์ทสำหรับส่งข้อมูลอยู่ด้วยกัน 3 พอร์ท มีชื่อดังนี้คือ A , B , และ C โดยพอร์ท C นี้จะแบ่งออกเป็น 2 ส่วนคือ พอร์ท C ล่าง (CLO) กับ C บน (CHI) นอกจากนี้แล้วยังมีพอร์ทอีกพอร์ทหนึ่งซึ่งทำหน้าที่ควบคุมการทำงานของพอร์ท A , B และ C โดยการรับคำสั่งจาก CPU พอร์ทนี้เราเรียกว่าพอร์ทควบคุม (Control port) พอร์ทนี้จะใช้

งานก็ต่อเมื่อ CPU ต้องการกำหนดลักษณะการทำงานของพอร์ท A , B และ C หรือต้องการเปลี่ยนแปลงหลังจากที่กำหนดไว้เดิม CPU จะส่งรหัสควบคุมมาทางดาต้าบัส (Data Bus) ให้แก่พอร์ทควบคุมนี้



PIN NAME	DESCRIPTION	TYPE
D0-D7	Bidirectional Data Bus	Bidirectional
PA0-PA7	Eight I/O pins Gesignated as Port A	Bidirectional
PB0-PB7	Eight I/O pins Gesignated as Port B	Bidirectional
PC0-PC7	Eight I/O pins Gesignated as Port C	Bidirectional
	Upper and Port C lower	
$\overline{RD}$	Read from device control	Input
$\overline{WR}$	Write to device control	Input
RESET	System reset	Input
$\overline{CS}$	Device select	Input
A0,A1	I/O port select	Input
V_{cc},GND	Power and Ground	Input

รูปที่ 1.10a ตำแหน่งขาต่าง ๆ ของ 8255



รูปที่ 1.10b แผนผังภายในของ 8255

การกำหนดรหัสที่ใช้ในการควบคุมพอร์ทต่าง ๆ นี้จะกล่าวในตอนต่อไป ในทางปฏิบัติบัติผู้ออกแบบระบบต้องนำรหัสควบคุมที่ได้มาตามข้อกำหนดของ 8255 นี้ไปใส่ในโปรแกรมเพื่อให้ CPU ทำการส่งรหัสควบคุมนี้มายังพอร์ทควบคุมเมื่อระบบนั้นเริ่มต้นทำงาน

หน้าที่ของขาต่าง ๆ

ก่อนที่จะกล่าวถึงการนำ 8255 ไปใช้งานควรทราบถึงหน้าที่ของขาต่าง ๆ ของ 8255 ทั้ง 40 ขาเสียก่อน จะทำให้เข้าใจถึงการใช้งานได้ดียิ่งขึ้น ขาต่าง ๆ ของ 8255 สามารถแบ่งออกได้ดังนี้

$\overline{CS}$ (Chip Select) ขานี้ใช้สำหรับรับสัญญาณจากภายนอกเพื่อใช้การเลือกว่าจะให้ 8255 ตัวนี้ทำงานหรือไม่ โดยแต่ถ้าขานี้ได้รับลอจิก " 0 " จะทำให้ 8255 เชื่อมต่อเข้ากับระบบบัสต่าง ๆ ของ CPU และพร้อมจะติดต่อกับ CPU ได้แต่เป็นลอจิก " 1 " มันก็จะปลดตัวเองออกจากระบบบัสของ CPU (โดยการเป็น Hi-Z)

$\overline{RD}$ (Read Enable) เป็นขาอินพุทที่รับสัญญาณจาก CPU ถ้าขานี้ได้รับลอจิก " 0 " และขณะนั้นขา $\overline{CS}$ ต้องเป็นลอจิก " 0 " ด้วย 8255 จะทำการส่งข้อมูลจากพอร์ทที่ CPU ต้องการติดต่อด้วยนั้นได้แก่ CPU ทางดาต้าบัส

$\overline{WR}$ (Write Enable) มีหน้าที่การทำงานตรงข้ามกับขา $\overline{RD}$ คือ ถ้าขา $\overline{WR}$ นี้ได้รับลอจิก " 0 " ($\overline{CS}$ ต้องเป็น " 0 " ด้วยเช่นกัน) 8255 จะรับข้อมูลจากดาต้าบัสของ CPU ส่งออกไปยังพอร์ทที่ CPU กำหนดไว้

RESET คือ ขาที่ทำหน้าที่ Reset 8255 เมื่อได้ที่ 8255 ได้รับสัญญาณ Reset มันจะกลับสู่โหมดอินพุทคือทุก ๆ พอร์ทจะเป็นอินพุทพอร์ทขา RESET นี้เมื่อต้องการเคลียร์สถานะต่าง ๆ ของ 8255

D_0-D_7 คือขาข้อมูลที่ใช้ในการติดต่อรับส่งข้อมูลกับ CPU โดยขา D_0-D_7 นี้จะต่อเข้ากับดาต้าบัสของ CPU เพื่อให้ CPU ส่งข้อมูลออกไปยังพอร์ทหรือรับข้อมูลจากพอร์ทส่งให้แก่ CPU ผ่านทาง D_0-D_7 นี้

A_0-A_1 คือขาแอดเดรสที่ใช้ในการเลือกพอร์ทที่ CPU ต้องการจะติดต่อด้วยซึ่งมีความเป็นไปได้ทั้งหมด 4 ค่า ดังนี้คือ

00 = พอร์ท A

01 = พอร์ท B

10 = พอร์ท C

11 = พอร์ท ควบคุม

$PA_0 - PA_7$ เป็นขาสัญญาณของพอร์ท A ใน8255ซึ่งถูกเลือกโดยค่าของ $A_0 - A_1$ และเมื่อพอร์ทนี้ถูกเลือกใช้ข้อมูลต่างๆก็จะถูกส่งผ่าน PA_0-PA_7 นี้ไปยัง D_0-D_7 (กรณีที่ให้พอร์ท A นี้ เป็นอินพุทพอร์ท) หรือจาก D_0- D_7 มายัง PA_0-PA_7 (กรณีที่เป็นเอาท์พุทพอร์ท)

PB_0-PB_7 เป็นขาสัญญาณของพอร์ท B ซึ่งจะถูกเลือกโดยลอจิกที่ A_0-A_1 เช่นกันกับพอร์ท A และพอร์ท B นี้มีข้อจำกัดการรับส่งข้อมูลที่ต่างจากพอร์ท A ในบางกรณี

PC_0-PC_7 เป็นสายสัญญาณของพอร์ท C ซึ่งแบ่งออกเป็น 2 กลุ่มคือ PC_0-PC_3 และ PC_4-PC_7 โดยแต่ละกลุ่มสามารถแยกกันทำงานได้โดยอิสระกลุ่มหนึ่งอาจเป็นอินพุทพอร์ทในขณะที่อีกกลุ่มหนึ่งเป็นเอาท์พุทพอร์ทได้แต่จะทำงานพร้อม ๆ กัน โดยการเลือกด้วยลอจิกที่ A_0-A_1

การใช้งาน 8255

8255 นั้นแบ่งลักษณะการทำงานออกเป็น 3 โหมด (Mode) ด้วยกันคือ

-โหมด 0 เป็นโหมดอินพุทหรือเอาท์พุทพอร์ทอย่างใดอย่างหนึ่ง ซึ่งทั้ง 3 พอร์ทคือ A,B และ C สามารถทำงานในโหมดนี้ได้

-โหมด 1 เป็นโหมดอินพุทหรือเอาท์พุทพอร์ทอย่างใดอย่างหนึ่งเช่นกันแต่จะมีลักษณะการทำงานเป็นลักษณะของ handshaking ซึ่งจะกล่าวรายละเอียดในภายหลังในโหมดนี้ทำงานได้เฉพาะพอร์ท A และ B

-โหมด 2 เป็นโหมด Bi-directional คือเป็นได้ทั้งอินพุทและเอาท์พุทพอร์ทในเวลาเดียวกันและทำงานแบบ handshaking เช่นเดียวกับโหมด 1ในโหมดนี้ใช้ได้เฉพาะพอร์ท A เท่านั้น

การกำหนดโหมดการทำงานของ 8255 นั้นทำได้โดยCPUทำการส่งรหัสควบคุมผ่านทางคาต้าบัสมายังพอร์ทควบคุม (Control port) ของ 8255 รหัสควบคุมนี้จะมีขนาด1ไบต์เรียกว่า Control Byte และในแต่ละบิตของ Control Byte (1Byte = 8bit) นั้นจะมีความหมายเฉพาะของตัวเองดังแสดงในรูปที่ 1.11 ซึ่งจะอธิบายได้ดังนี้

บิต D_7 เป็นบิตที่แสดงว่า Byte นี้เป็นรหัสควบคุม (Control Byte) ที่จะมีผลต่อการกำหนดโหมดการทำงานของ 8255

บิต D_6 และ D_5 มีความหมายในการเลือกโหมดของพอร์ท A ซึ่งสามารถทำงานได้ ทั้ง 3 โหมดโดยลอจิกที่ D_6 และ D_5 จะมีความหมายดังนี้

00 = โหมด 0

01 = โหมด 1

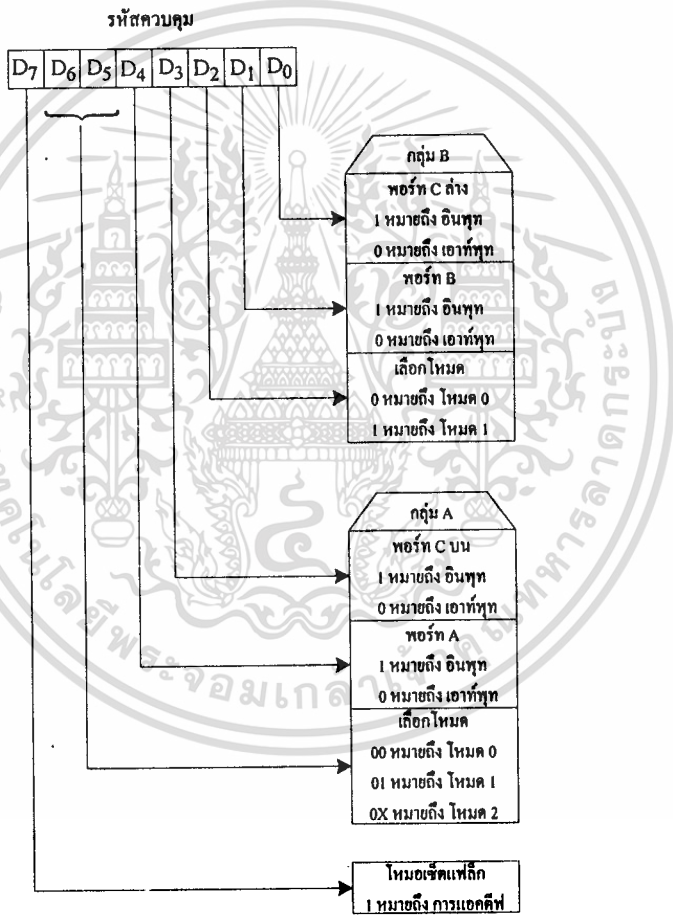
10 = โหมด 2

11 = โหมด 2

บิต D_4 ถ้าลอจิก " 0 " หมายถึงสั่งให้พอร์ท A ทำหน้าที่เป็นเอาท์พุทพอร์ท แต่ถ้าเป็นลอจิก " 1 " พอร์ท A จะเป็นอินพุทพอร์ท บิตนี้จะมีความหมายเมื่อเราให้ 8255 ทำงานในโหมด 0 หรือโหมด 1 เท่านั้น เพราะในโหมดที่ 2 พอร์ท A จะเป็นอินพุทและเอาท์พอร์ทในเวลาเดียวกัน

บิต D_3 เป็นที่กำหนดหน้าที่การทำงานของพอร์ท C บน (PC_4-PC_7) ถ้าบิตนี้เป็นลอจิก " 0 " พอร์ท C บน นี้จะเป็นเอาท์พุทพอร์ท ถ้าเป็น " 1 " จะเป็นอินพุทพอร์ท

บิต D_2 เป็นบิตที่ใช้สำหรับกำหนดโหมดการทำงานของพอร์ท B ถ้าเป็นลอจิก " 0 " หมายถึงให้พอร์ท B ทำงานในโหมด 0 ถ้าลอจิก " 1 " จะทำงานในโหมด 1



รูปที่ 1.11 แสดงความหมายของแต่ละบิตในรหัสควบคุม

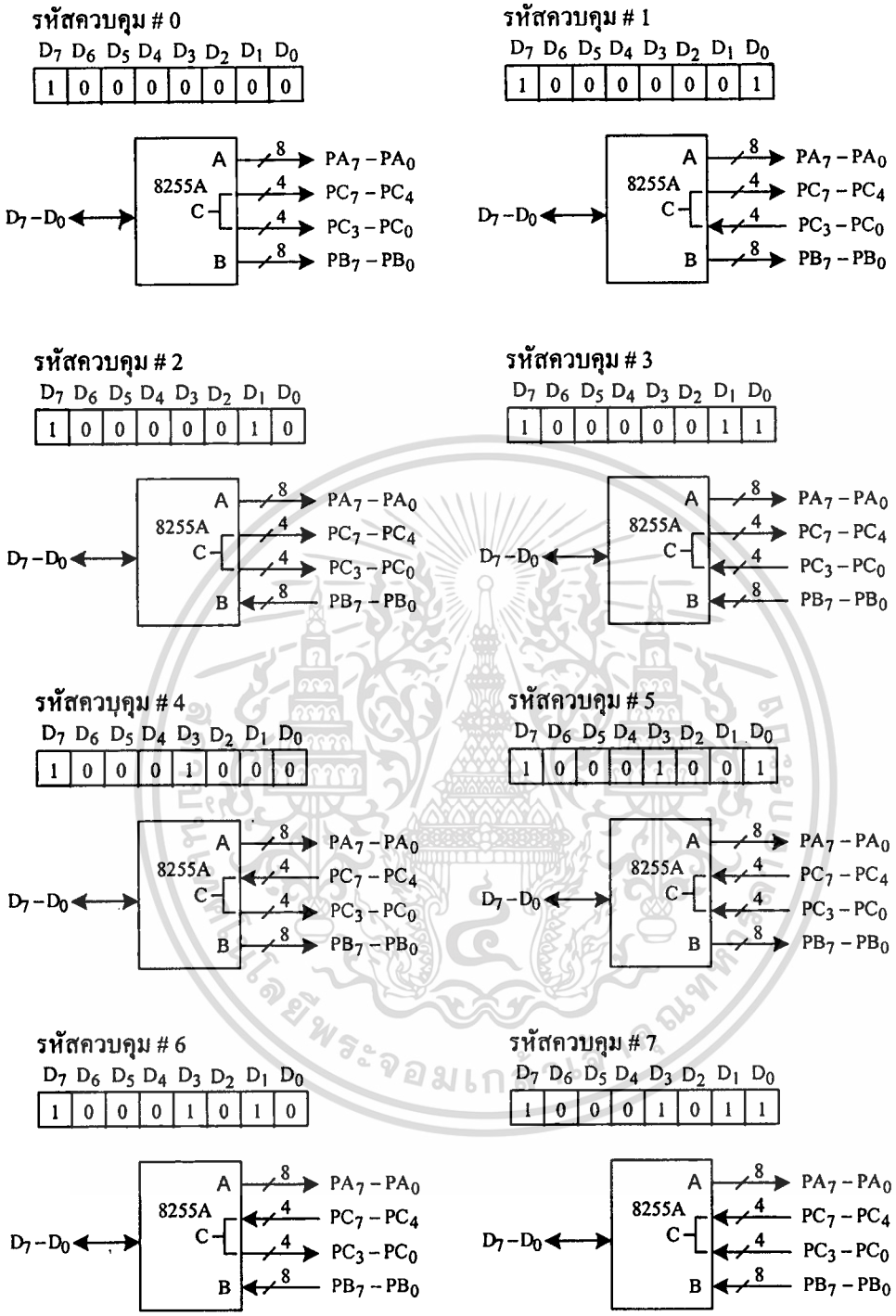
บิต D_1 เป็นการกำหนดให้พอร์ทเป็นอินพุทหรือเอาท์พุทพอร์ท ถ้า D1 เป็นลอจิก " 0 " จะเป็นเอาท์พอร์ทแต่ถ้าเป็นอินพุทพอร์ท

บิต D_0 เป็นอินพุทหรือเอาท์พอร์ท C ล่าง (PC_0 –PC_3) ถ้าบิตนี้เป็น " 0" จะเป็นเอาท์พุท ถ้าเป็น " 1" จะเป็นอินพุท

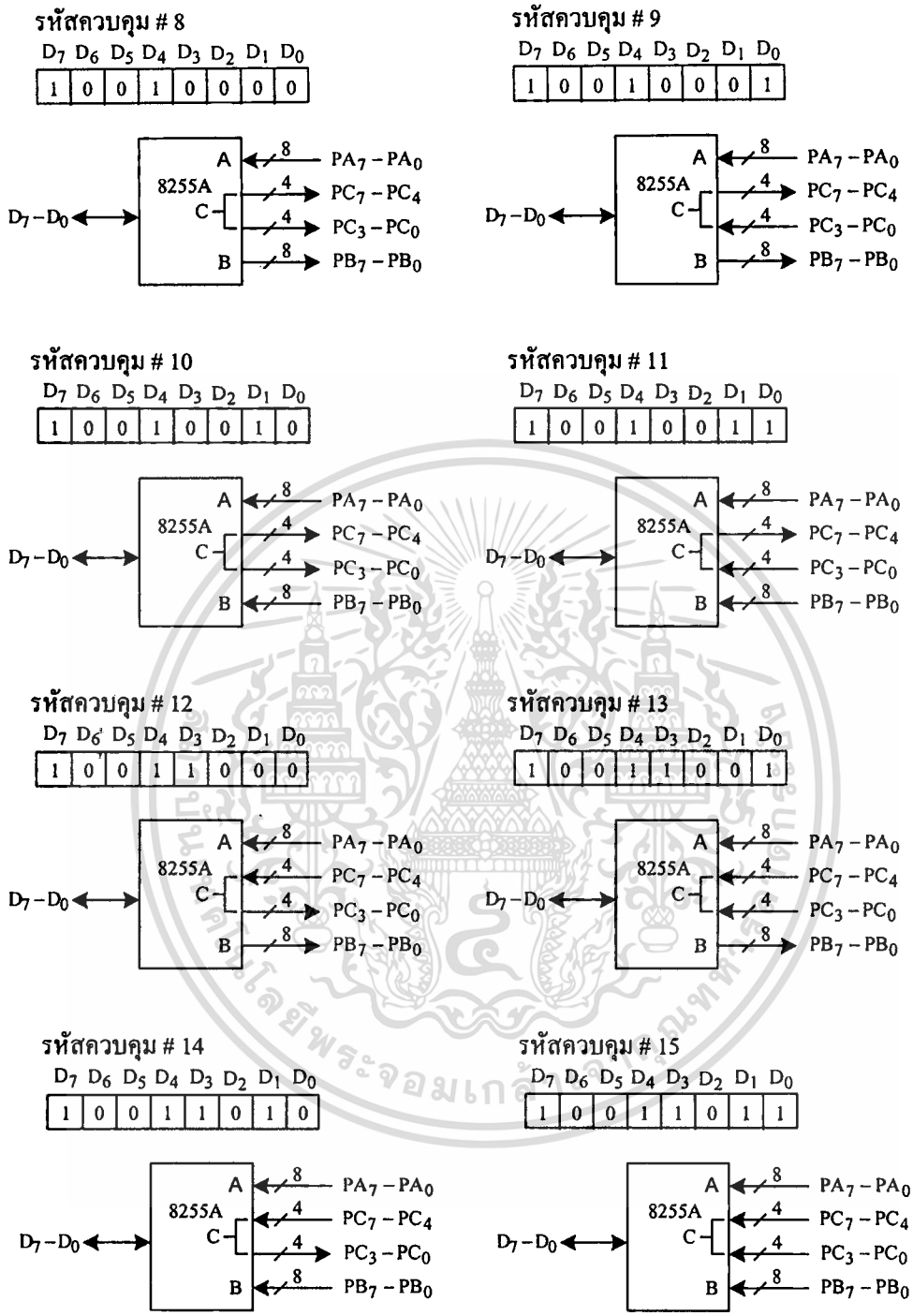
รายละเอียดการทำงานในแต่ละโหมดของ8255

-<u>โหมด 0</u>

ในโหมด 0 นี้เป็นการกำหนดให้พอร์ททุกพอร์ทของ 8255 นี้เป็นอินพุทเอาท์พุทพอร์ทแบบพื้นฐานหรือที่เรียกว่า Simple I /O Port ซึ่งเป็นที่นิยมใช้กันมาก เมื่อนำมาเป็นพอร์ทของ MCS-51 มีรูปแบบความเป็นไปได้ในการโปรแกรมให้พอร์ท A , B และCเป็นอินพุทหรือเอาท์พุทพอร์ทได้ทั้งหมด 16 รูปแบบดังรูปที่1.12



รูปที่1.12 รหัสควบคุมของการทำงานในโหมด 0



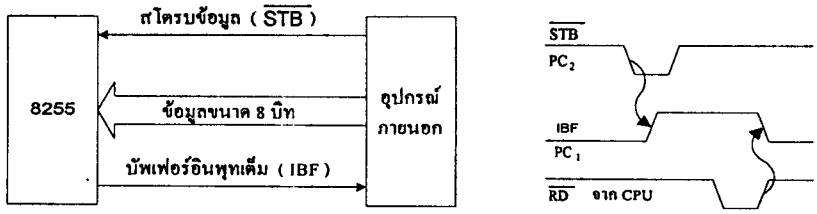
รูปที่1.12 รหัสควบคุมของการทำงานในโหมด 0 (ต่อ)

จากรูปในแต่ละช่องจะแสดงรูปแบบการทำงานในโหมด 0 ของ 8255 หนึ่งในรูปแบบ โดยจะแสดงถึงสถานะการทำงานของพอร์ท A,B และ C ด้วยลูกศร ถ้าลูกศรชี้ขึ้นออกจากตัว 8255 หมายถึงพอร์ทนั้นเป็นเอาท์พุทพอร์ท แต่ถ้าลูกศรชี้ขึ้นหมายถึงเป็นอินพุทพอร์ท ส่วนพอร์ท C นั้น จะแบ่งเป็น 2 ส่วน คือ พอร์ท C บน และพอร์ท C ล่าง พอร์ทละ 4 เส้น ส่วนด้านบนของแต่ละช่องแสดงค่าของรหัสควบคุม (Control word) ที่จะต้องส่งให้แก่พอร์ทควบคุมในรูปเลขฐาน 16 เพื่อโปรแกรมให้พอร์ทต่าง ๆ ทำงานเป็นอินพุท-เอาท์พุทตามที่แสดงเอาไว้ในช่องนั้น ๆ เช่น ถ้าเราต้องการให้พอร์ท A และพอร์ท C บนเป็นอินพุทพอร์ทส่วนพอร์ท B และพอร์ท C ล่างเป็นเอาท์พุทพอร์ท ถ้าดูจากรูปที่ 1.12 ก็จะตรงกับช่องของ Control word # 14 มีลอจิกที่ D_0-D_7 ดังนี้คือ 10011010 หรือ 9H ในฐาน 16 เช่นเดียวกับที่เคยกล่าวมาแล้วนั่น

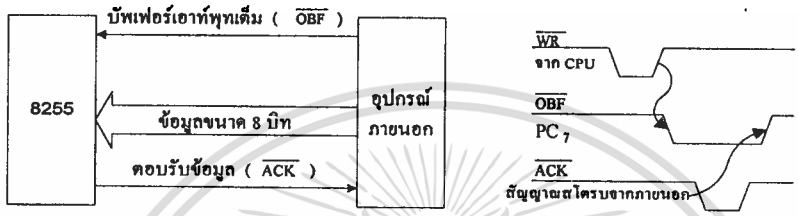
<u>โหมด 1</u>

สำหรับโหมด 1 นี้เป็นการรับส่งข้อมูลในแบบ Handshaking ความหมายของ ก็คือระหว่าง CPU พอร์ท และอุปกรณ์ภายนอกนั้น ขณะที่รับส่งข้อมูลกันแล้ว ยังต้องมีสัญญาณในการตอบรับในแต่ละครั้งของการรับส่งข้อมูล โดยผู้รับกับผู้ส่งนั้นจะต้องทำงานสัมพันธ์กันตลอดเวลาซึ่งเป็นประโยชน์ในกรณีที่อุปกรณ์ภายนอกนั้นมีการทำงานที่ช้ากว่า CPU จึงต้องใช้วิธีรับส่งข้อมูลแบบ Handshaking โดยอุปกรณ์ภายนอกจะเป็นตัวกำหนดจังหวะในการรับส่งข้อมูลเอง เช่นการส่งข้อมูลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์นั้นทำงานได้ช้ากว่าคอมพิวเตอร์มากเมื่อคอมพิวเตอร์ส่งข้อมูลตัวอักษรตัวแรกให้แก่เครื่องพิมพ์ทำการพิมพ์เครื่องพิมพ์ก็จะทำการประมวลผลต่าง ๆ และเมื่อพร้อมที่จะพิมพ์ตัวอักษรตัวนี้นแล้วก็จะส่งสัญญาณบอกคอมพิวเตอร์ให้ทำการส่งตัวอักษรตัวต่อมาได้ การติดต่อระหว่างเครื่องพิมพ์กับคอมพิวเตอร์จึงเป็นไปโดยไม่ผิดพลาด ส่วนสัญญาณที่ใช้ในการควบคุม การรับส่งข้อมูลนี้จะได้มาจาก พอร์ท C เพราะในโหมดนี่พอร์ทที่ใช้ในการรับส่งข้อมูลได้คือ พอร์ท A และ B เท่านั้น ส่วน พอร์ท C จะเป็นตัวส่งและรับสัญญาณควบคุมพับอุปกรณ์ภายนอก และสัญญาณควบคุม ในแต่ละบิตของ พอร์ท C จะเป็นตามตารางที่ 5

ลักษณะของการรับส่งข้อมูลแบบ Handshaking นั้นแสดงในรูปที่ 1.13 ในกรณีที่เป็นอินพุทพอร์ท (A หรือ B ก็ตาม) เมื่ออุปกรณ์ภายนอกต้องการส่งข้อมูลให้แก่ CPU มันก็จะส่งข้อมูลเข้ามายังพอร์ทพร้อมกับส่งสัญญาณ $\overline{\text{STB}}$ (Strobe) มาบอกแก่ 8255 เมื่อ 8255 ได้รับสัญญาณ $\overline{\text{STB}}$ ก็จะรับข้อมูลนั้นไปเก็บไว้ในรีจิสเตอร์ภายในก่อนแล้ว 8255 ก็จะส่งสัญญาณ IBF (Input Buffer Full) เป็นลอจิก " 1 " ไปบอกแก่อุปกรณ์ภายนอกว่าพอร์ทได้รับข้อมูลมาเก็บไว้ในบัฟเฟอร์ (รีจิสเตอร์) แล้วและอย่างเพิ่มส่งมาอีกจนกว่า CPU จะรับข้อมูลนั้นไปจากพอร์ทแล้ว " 0 " (โดยดูจากการที่



กรณีอินพุทพอร์ท



กรณีเอาท์พุทพอร์ท

รูปที่ 1.13 การจัดสัญญาณในแบบ Handshaking

ในทางปฏิบัติแล้วการ CPU จะทราบได้อย่างไรว่าเมื่อไรที่อุปกรณ์ภายนอกได้ส่งข้อมูลเข้า CPU มาที่บัฟเฟอร์ของพอร์ทแล้วในกรณีที่เป็นอินพุทพอร์ท หรือเมื่อไรที่อุปกรณ์ภายนอกได้รับข้อมูลจากพอร์ทไปแล้วพร้อมที่จะให้ CPU ส่งข้อมูลต่อไปได้ในกรณีที่เป็นอินพุทพอร์ทนั้น มีวิธีจะให้ CPU ทราบได้ 2 ลักษณะ คือ

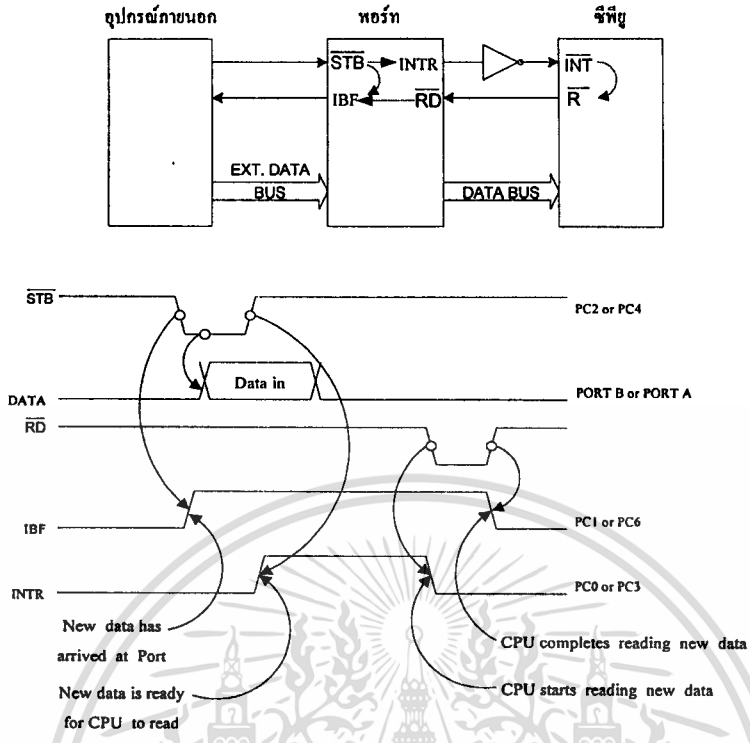
การเขียนโปรแกรมให้ CPU ทำการตรวจสอบสัญญาณควบคุมจาก 8255 เช่นในกรณีเอาต์พุทพอร์ทก็จะให้ CPU คอยตรวจสอบบิตที่ 7 ของ พอร์ท C ($\overline{OBF}$) หลังจากที่ส่งข้อมูลให้แก่พอร์ทไปแล้ว ถ้าบิต 7 นี้ยังเป็นลอจิก " 0 " อยู่แสดงว่าอุปกรณ์ภายนอกยังไม่ได้รับข้อมูลไปจากพอร์ท แต่ถ้าบิตที่ 7 เป็น " 1 " แสดงว่าอุปกรณ์ภายนอกได้รับข้อมูลไปแล้วจึงให้ CPU ทำการส่งข้อมูลชุดใหม่ต่อไป เช่นเดียวกันในกรณีของอินพุทพอร์ทก็ให้ CPU ตรวจสอบบิตที่ 1 ของพอร์ท C ได้ในทำนองเดียวกัน แต่วิธีนี้ CPU จะต้องเสียเวลาตรวจสอบลอจิกของบิตดังกล่าวอยู่ตลอดเวลา ซึ่ง จะทำให้เสียเวลาของ CPU ไปโดยเปล่าประโยชน์ จึงควรเลือกใช้วิธีนี้ต่อเมื่อพิจารณาแล้วว่าการเสียเวลาของ CPU นั้น ไม่มีผลเสียต่อระบบ

- อีกวิธีหนึ่งนี้คือการให้อุปกรณ์ภายนอกนั้นทำการขออินเตอร์รัพท์จากที่แสดงใน ตารางที่ 2 นั้นจะเห็นว่า บิต 0 และบิต 3 ของพอร์ท C นั้นเป็นสัญญาณอินเตอร์รัพท์ (INTR) ของ พอร์ท B และ A ตามลำดับ ซึ่งสามารถใช้สัญญาณอินเตอร์รัพท์นี้ให้เป็นประโยชน์ได้ ในการต่องาน

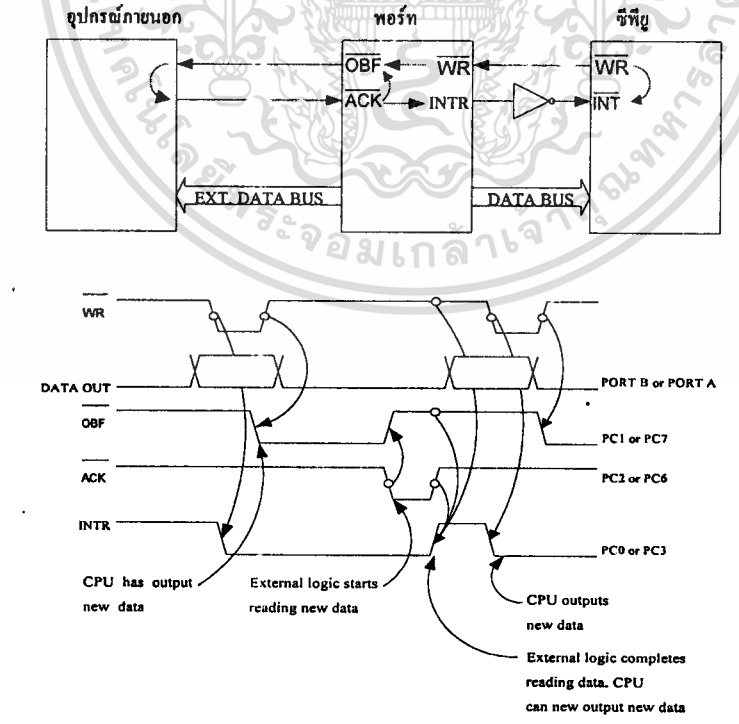
สัญญาณ INTR ไปที่ขา $\overline{INT}$ ของ MCS-51 Not gate (Inverter) เสียก่อน เนื่องจากการขออินเตอร์รัพท์ของ MCS-51 นั้นเป็นแอคทีฟ "0" การรับส่งข้อมูลในแบบนี้แสดงในรูปที่ 1.14 และรูปที่ 1.15

ในรูปที่ 1.13 กรณีที่เป็นอินพุทพอร์ทเมื่ออุปกรณ์ภายนอกส่งข้อมูลในแก่พอร์ทพร้อมทั้งส่งสัญญาณ $\overline{STB}$ มาให้พอร์ทรับข้อมูลนี้ไปเก็บในรีจิสเตอร์ เมื่อพอร์ทรับข้อมูลนี้ไปเก็บแล้วจะส่งสัญญาณ IBF ตอบรับไปยังอุปกรณ์ภายนอก เพื่อให้หยุดการส่งข้อมูลซ้อนแล้วจึงต้องการขออินเตอร์รัพท์โดยการส่งสัญญาณ INTR ผ่าน Inverter ไปที่ขา $\overline{INT}$ ของ CPU เพื่อขอให้ CPU มารับข้อมูลนี้ไป หลังจาก CPU รับข้อมูลนี้ไปแล้ว (8255 ได้รับสัญญาณ $\overline{RD}$) พอร์ทก็จะขอถอนอินเตอร์รัพท์ทำให้สัญญาณ IBF กลับเป็น " 0 " เป็นการบอกให้อุปกรณ์ภายนอกได้ทราบว่าพร้อมที่จะรับข้อมูลไบต์ต่อไปได้แล้ว

ส่วนในรูปที่ 1.15 เป็นกรณีของเอาท์พุทพอร์ทเมื่อ CPU ต้องการส่งข้อมูลให้อุปกรณ์ภายนอก โดยการส่งสัญญาณ $\overline{WR}$ แก่ 8255 ให้รับข้อมูลไปไว้ที่รีจิสเตอร์บัฟเฟอร์โดยพอร์ทจะรับข้อมูลไว้แล้วส่งสัญญาณ $\overline{OBF}$ ไปบอกอุปกรณ์ภายนอกให้มารับข้อมูลไปและหลังจากที่อุปกรณ์ภายนอกรับข้อมูลไปและหลังจากที่อุปกรณ์ภายนอกรับข้อมูลไปแล้วก็จะตอบรับโดยการส่งสัญญาณ $\overline{ACK}$ แก่ 8255 ทำให้สัญญาณ $\overline{OBF}$ กลับไปเป็น " 1 " พร้อมกับทำให้สัญญาณ INTR เป็น " 1 " ด้วย ซึ่งจะเป็นการขออินเตอร์รัพท์ให้ CPU ส่งข้อมูลใหม่ต่อไปในกรณีที่ต้องการส่ง ข้อมูลต่อเนื่องตามกรรมวิธีเดิม แต่ถ้า CPU ยังไม่ต้องการส่งข้อมูลต่อไปหรือยังไม่มีข้อมูลก็สามารถสั่งให้พอร์ทขอถอนอินเตอร์รัพท์ได้ (ทำให้ INTR เป็น " 0 ") ด้วยการส่งแอดเดรสไบต์ไปยังพอร์ทควบคุมเป็นการกำหนดให้บิตใด ๆ ของพอร์ท C เป็นลอจิก "0" หรือ "1" ตามต้องการ มีวิธีการกำหนดดังนี้



รูปที่ 1.14 การขออินเตอร์รัพท์กรณีเป็นอินพุทพอร์ท



รูปที่ 1.15 การขออินเตอร์รัพท์กรณีเป็นเอาท์พุทพอร์ท

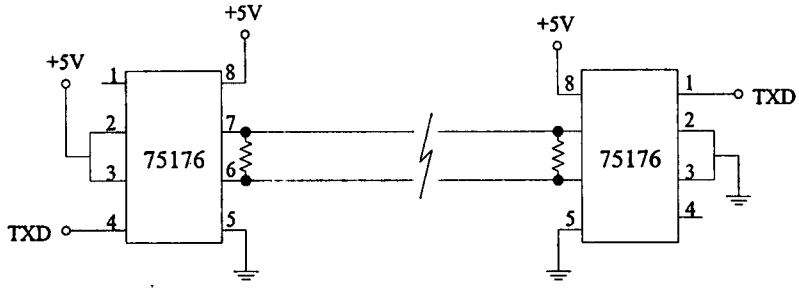
พอร์ท C	ความหมาย
PC0	I/O
PC1	I/O
PC2	I/O
PC3	$INTR_A$
PC4	STB_A
PC5	IBF_A
PC6	ACK_A
PC7	OBF_A

ตารางที่ 6 แสดงตำแหน่งขาของพอร์ท C

ขณะที่โปรแกรมให้พอร์ท A ทำงานในโหมด 2 นี้ เรายังสามารถโปรดแกรมให้พอร์ท นั้นทำงานในโหมด 0 หรือ โหมด 1 ก็ได้ ซึ่งจะทำให้สามารถทำพอร์ท B ไปใช้งานอื่น ๆ ได้อีกโดยที่เป็นอิสระจากกัน

1.3 การเชื่อมต่อพอร์ตสื่อสารอนุกรมด้วยมาตรฐาน RS-232/422

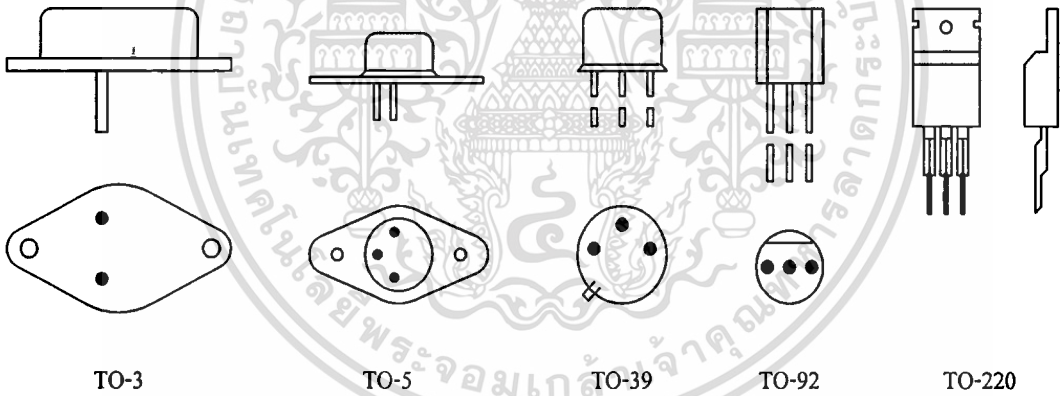
โดยปกติแล้ว 8032 จะสามารถสื่อสารแบบอนุกรม 2 ทางได้ แต่ระบบนี้เป็นการส่งข้อมูลช่วงสั้นๆโดยจะผลัดกันส่ง เพื่อความประหยัดคู่สายที่ใช้ส่งจึงใช้การส่งแบบกึ่ง 2 ทาง (half duplex) โดยการเลือกใช้ไอซีเบอร์ DS75176 ซึ่งจะเป็นการรับ-ส่ง ตามมาตรฐาน RS-422A หรือ RS-485 ก็ได้ ซึ่งการส่งจะใช้สัญญาณแบบดิฟเฟอเรนเชียล หลักการคือสัญญาณที่จะรับและส่งเป็นการเปรียบเทียบระหว่างสัญญาณ 2 เส้น ทำให้เกิดการลดทอนและสัญญาณรบกวนในระหว่างการส่งข้อมูลจะเกิดขึ้นในสายทั้ง 2 เส้น ด้วยค่าที่เท่ากันหรือใกล้เคียงกัน ความแตกต่างของสัญญาณในคู่สายจึงยังคงเหมือนเดิม หรือเปลี่ยนแปลงเล็กน้อย ทำให้ระบบนี้สามารถส่งได้ไกลกว่ามาตรฐาน RS232 และเนื่องจากไอซี เบอร์ DS75176 นี้มีความไวในการรับได้เมื่อมีความแตกต่างในคู่สาย $\pm$ 200 mV หรือมากกว่า



รูปที่ 1.17 แสดงการเชื่อมต่อRS422

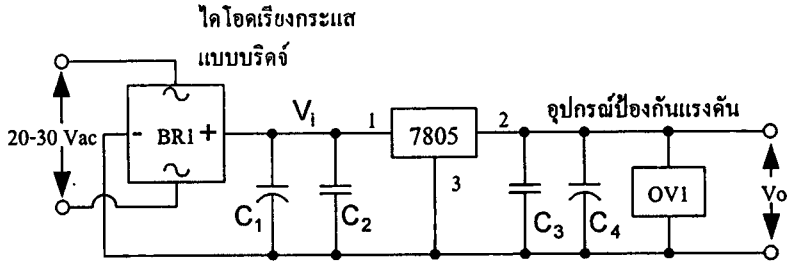
1.4 ระบบแหล่งจ่ายไฟ

วงจรรักษาระดับแรงดันที่ใช้ไอซีแบบ 3 ขานี้ เป็นกลุ่มของอุปกรณ์ซึ่งรักษาระดับแรงดันเอาท์พุทคงที่ บรรจุอยู่ในตัวถังมาตรฐานซึ่งสามารถให้กระแสได้แตกต่างกัน ซึ่งแสดงดังรูปที่ 1.18



รูปที่1. 18 แสดงตัวถังไอซีแบบต่างๆ

วงจรมาตรฐานทั่วไปที่นิยมใช้ไอซีรักษาระดับแรงดันแบบ 3 ขาแสดงให้เห็นดังรูปที่ 1.19



รูปที่1.19 แสดงวงจรรักษาระดับแรงดันทั่วไป

ไดโอดเรียงกระแสแบบบริดจ์ BR1 และตัวเก็บประจุ C1 นั้นเหมือนกันหมดสำหรับแหล่งจ่ายไฟใดๆ ในการเลือกใช้อุปกรณ์เหล่านี้มีหลักง่ายๆคือ ตัวเก็บประจุ C1 มีค่าไม่ต่ำกว่า 1000 uF/A ของกระแสโหลดสูงสุด

ตัวเก็บประจุ C2 และ C3 ใช้สำหรับปรับค่าความต้านทานต่อสัญญาณให้ดีขึ้น และมีค่าอยู่ระหว่าง 0.1 – 0.47 uF ค่าที่แท้จริงขึ้นอยู่กับการเลือกใช้ ยกเว้นว่าต้องการจ่ายกระแสให้โหลดสูงขึ้นก็จะเลือกค่าความจุสูงขึ้น เราสามารถใช้ค่าความจุ 0.1 uF สำหรับแหล่งจ่ายไฟขนาด 1 แอมแปร์ หรือ 0.33 uF หรือ 0.47 uF สำหรับกระแส 3 และ 5 แอมแปร์ตามลำดับ

ตำแหน่งของ C2 และ C3 นั้นเป็นสิ่งที่สำคัญมาก ตัวเก็บประจุเหล่านี้ใช้สำหรับกดสัญญาณรบกวนต่างๆ ซึ่งประกอบด้วยพัลส์ความถี่สูง ดังนั้นตัวเก็บประจุ C2 และ C3 จะต้องติดตั้งให้ใกล้กับตัวถังของไอซีรักษาระดับแรงดันเท่าที่จะทำได้ ถ้าตัวเก็บประจุติดตั้งห่างจากไอซีรักษาระดับมากเกินไปก็จะทำให้การทำงานของตัวเก็บประจุลดประสิทธิภาพลง

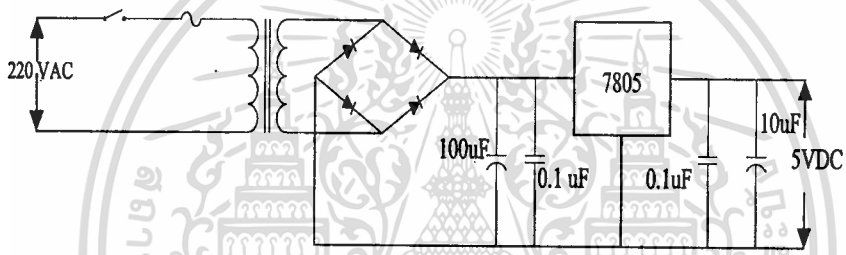
สำหรับตัวเก็บประจุ C4 เป็นตัวที่เพิ่มเติมเข้ามา แนะนำให้ใช้เฉพาะในวงจรที่มีการเปลี่ยนแปลงของกระแสโหลดอย่างมากในเวลาสั้น ๆ เช่น ในวงจรดิจิตอล ซึ่งในกรณีนี้เป็นหน้าที่ของตัวเก็บประจุ C4

หน้าที่ของตัวเก็บประจุ C4 ก็คือการปรับค่าตอบสนองทรานเซี้ยนต์ (transient response) ของวงจรรักษาระดับแรงดันตัวเก็บประจุจะทำหน้าที่เป็นแหล่งสะสมประจุ เพื่อที่จะป้อนกระแสเข้าสู่โหลดในช่วงเวลาสั้น ๆ ในขณะที่วงจรรักษาระดับแรงดันกำลังปรับตัวเอง เพื่อรับกับความต้องการกระแสที่สูงขึ้น

อุปกรณ์ระบุว่า OV1 นั้นเป็นวงจรป้องกันแรงดันเกิน บางครั้งเรียกว่า วงจรเอสซีอาร์โควลบาร์ (SCR crowbar) หน้าที่ของ OV1 นั้นก็เพื่อป้องกันวงจรภายนอกที่รับกระแสจากวงจรรักษาระดับแรงดันนี้ ไม่ให้เกิดความเสียหายเมื่อวงจรรักษาระดับแรงดัน U1 เกิดความเสียหายขึ้น

แผ่นระบายความร้อนสำหรับ U1 นั้น บางครั้งก็มีเพิ่มเติมมาโดยผู้ผลิตแต่ละราย แต่จริงๆ ควรติดตั้งเข้าไปกับวงจรรักษาระดับแรงดันทุกชนิด เนื่องจากวงจรรักษาระดับแรงดันเป็นอุปกรณ์สารกึ่งตัวนำที่กระจายพลังงานออกมา ดังนั้นความเชื่อถือได้จะแปรผันโดยตรงกับค่าอุณหภูมิ สำหรับวงจรที่จ่ายกระแสได้ขนาด 1 แอมแปร์หรือน้อยกว่า ตัวถังที่เป็นโลหะของตัวมันเองก็อาจเพียงพอ

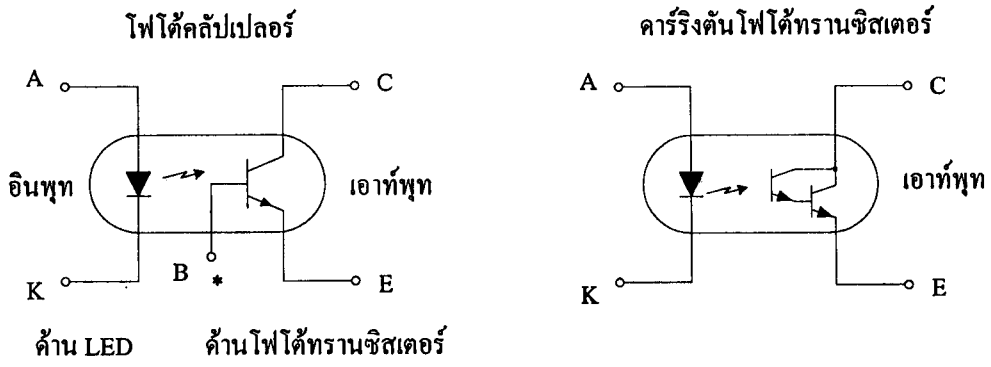
เราอาศัย ไอซี สำเร็จรูปเพื่อสร้างแหล่งไฟตรงเพื่อจ่าย กระแสให้กับวงจร โดยใช้ไอซี เบอร์ 7805 สำหรับสร้างแรงดันไฟบวก 5 โวลท์ โดยตัวเก็บประจุที่ต่อในวงจรเพื่อเป็นการกรองกระแสให้เรียบยิ่งขึ้น และ ไอซี เบอร์ 7812 สร้างแรงดันไฟตรง +12 โวลท์ โดยเราเพียงแค่ต่อตัวเก็บประจุภายนอกร่วมไม่กี่ตัว เพื่อกรองกระแสให้เรียบยิ่งขึ้นเท่านั้น



รูปที่ 1.20 แสดงวงจรภาคจ่ายไฟ

1.5 การอินเตอร์เฟสโดยใช้โฟโต้คัปเปลอร์ (photo coupler)

โฟโต้คัปเปลอร์เป็นอุปกรณ์ที่ใช้ในการอินเตอร์เฟสกันมาก ภายในโฟโต้คัปเปลอร์จะมี LED กับโฟโต้ทรานซิสเตอร์วางคู่กันอยู่ LED จะอยู่ทางด้านอินพุทส่วนโฟโต้ทรานซิสเตอร์จะอยู่ทางด้านเอาท์พุท เมื่อเราจ่ายกระแสเข้าไปที่ LED LED จะเปล่งแสงออกมา แสงนี้เรามองไม่เห็นเพราะอยู่ภายในตัวถัง โฟโต้ทรานซิสเตอร์เมื่อได้รับแสงนี้จะอยู่ในสภาวะอิ่มตัวหรือ ON โฟโต้คัปเปลอร์ที่มีขายทั่วไปในปัจจุบัน มีรูปร่างเหมือนไอซีทั่วไป อยู่ในตัวถังแบบดิพมีทั้งแบบพลาสติกและเซรามิก โฟโต้ทรานซิสเตอร์ภายในบางครั้งเป็นแบบดาร์ลิงตันมีอัตราการขยายสูงมาก อัตราการขยายกระแสของโฟโต้คัปเปลอร์คิดจากอัตรสส่วนของกระแสอินพุทที่ป้อนให้ LED กับกระแสขาออกที่ทรานซิสเตอร์ขับได้ บางครั้งเรานำโฟโต้คัปเปลอร์มาใช้เป็นวงจรขยายในวงจรอนาล็อกก็ได้

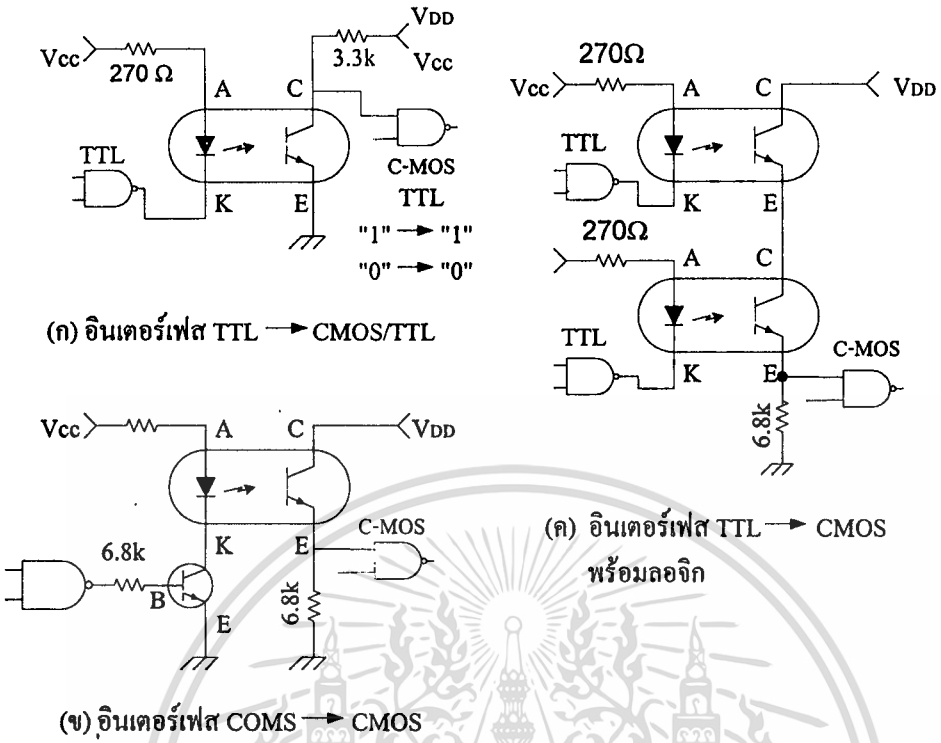


* บางตัวก็มีขาเบสออกมาให้

รูปที่ 1.21 แสดงโครงสร้างของโฟโต้คัปเปลอร์

คุณสมบัติที่เด่นที่สุดของโฟโต้คัปเปลอร์คือ การที่อินพุทและเอาต์พุทแยกทางกันทางไฟฟ้าอย่างเด็ดขาดนั่นเอง คุณสมบัติอันนี้ทำให้วงจรที่เชื่อมต่อกันด้วยโฟโต้คัปเปลอร์ไม่ต่อถึงกันทางไฟฟ้า ใช้แหล่งจ่ายไฟแยกกันได้ ในวงจรที่เกี่ยวข้องกับความปลอดภัยและวงจรที่ต้องการกำจัดสัญญาณรบกวนมักจะใช้โฟโต้คัปเปลอร์นี้

รูปที่ 1.22 แสดงวิธีการใช้โฟโต้คัปเปลอร์เป็นวงจรอินเตอร์เฟสระหว่างสองวงจร อาจใช้เชื่อมต่อกันระหว่าง TTL กับ CMOS ก็ได้หรือระหว่าง CMOS กับ TTL กันเองก็ได้ สังเกตว่าการต่อวงจรทั้งแบบ (ก) และ(ข) ในรูปที่ 1.22 นั้น โฟโต้คัปเปลอร์ทำหน้าที่คล้ายบัฟเฟอร์ คือสัญญาณ " 1 " และ " 0 " จะถูกส่งถ่ายกันโดยไม่มีการกลับสัญญาณ



รูปที่1.22 ตัวอย่างการอินเตอร์เฟสใช้โฟโต้คัปเปลอร์

ในรูปที่ 1.22 (ค) เป็นตัวอย่างการใช้งานที่ประยุกต์ขึ้นมา โดยใช้โฟโต้คัปเปลอร์สองตัวมาต่อเป็นเงื่อนไข AND กัน จะเห็นว่าถ้า LED ทั้งสองสว่างจึงจะทำให้มีกระแสไหลผ่านโฟโต้คัปเปลอร์ทั้งสองตัวไปที่ตัวต้านทาน 6.8 กิโลโอห์ม ทำให้สัญญาณ " 1 " เป็นเอาท์พุท ถ้ากรณีอื่นจะได้สัญญาณ " 0 " เป็นเอาท์พุท เราอาจใช้โฟโต้คัปเปลอร์มาประกอบเป็นวงจร AND,OR,NOR,NAND ได้เช่นกัน

บทที่ 2

ทฤษฎีการทำงานของโครงงาน

ในการพัฒนาโครงงานนี้ได้ใช้บอร์ดคอนโทรลเลอร์สำเร็จรูปรุ่น V-3155 ของบริษัทศิลารีเสิจน์ จำกัด ซึ่งมีรายละเอียดต่างๆดังนี้

2.1 รายละเอียดต่าง ๆ บนบอร์ดไมโครคอนโทรลเลอร์

บอร์ดไมโครคอนโทรลเลอร์ V-3155 จะใช้ไมโครคอนโทรลเลอร์เบอร์ 80C31 เป็นหลัก แต่อย่างไรก็ตาม ผู้ใช้สามารถใช้กับไมโครคอนโทรลเลอร์ต่าง ๆ ในตระกูล MCS-51 ที่เป็นแบบ 40-PIN DIP ได้ทั้งหมด เช่น 8032 8751 8752 ซึ่งจะทำให้ได้คุณสมบัติเป็นไปตามโครงสร้างบอร์ดนั้นๆ การเลือก JUMPER $\overline{EA}$ จะใช้เพื่อการเลือกให้ทำงานจาก ROM หรือ EPROM ภายในตัวไมโคร (INT) หรือเลือก EPROM จากภายนอก (EXT) ทั้งนี้การเลือก $\overline{EA}$ ในตำแหน่ง INT จะใช้กับไมโครที่มีโปรแกรมภายในเท่านั้น ซึ่งปกติจะเป็น 8751 หรือ 8752 (กรณี 8051 หรือ 8052 มี ROM อยู่ภายในก็จริง แต่ในทางปฏิบัติ การโปรแกรม ROM ภายในของ 8051 หรือ 8052 จะทำได้จากโรงงานผู้ผลิตเท่านั้น)

หน่วยความจำและการเลือก JUMPER

หน่วยความจำบนบอร์ดสามารถใช้ได้เป็นแบบ ROM(EPROM) สามารถเลือกเบอร์ต่างๆได้ตามต้องการ โดยใช้ JUMPER ที่อยู่ใต้ 74LS373 ซึ่งสามารถเลือกหน่วยความจำได้ตั้งแต่ 8K-32K จะกระทำได้โดยปรับตัว JUMPER ให้ตรงกับช่องที่ระบายที่ขาวทึบตรงตามเบอร์ที่ต้องการ

SERIAL PORT

เป็นพอร์ตการสื่อสารอนุกรมของบอร์ด V-3155 ซึ่งสามารถที่จะเลือกใช้หรือไม่ใช้ก็ได้ ถ้าต้องการใช้ก็ให้เสียบชิปเบอร์ MAX232 และใช้งานที่ขั้วต่อแบบ 3 PIN

SYSTEM BUS และ PORT 8255 BUS

SYSTEM BUS ของบอร์ด V-3155 จะใช้ขาสัญญาณจากตัวไมโคร เป็นขนาด 40 PIN ซึ่งใช้สำหรับขยายระบบตามต้องการ หรือจะใช้กับบอร์ดขยายของบริษัท ศิลา จำกัด ก็ได้ ทั้งนี้ยังสามารถดึง PORT 1 และขาสัญญาณ $\overline{INT0}$, $\overline{INT1}$, $\overline{T0}$, $\overline{T1}$ ออกมาเพื่องานประยุกต์ใดๆก็ได้ ส่วนพอร์ท 8255 BUS ซึ่งใช้ชิปซัพพอร์ทเบอร์ 8255 ซึ่งทำหน้าที่เป็นพอร์ทอินพุท/เอาท์พุทมีตำแหน่ง แอดเดรสดังนี้

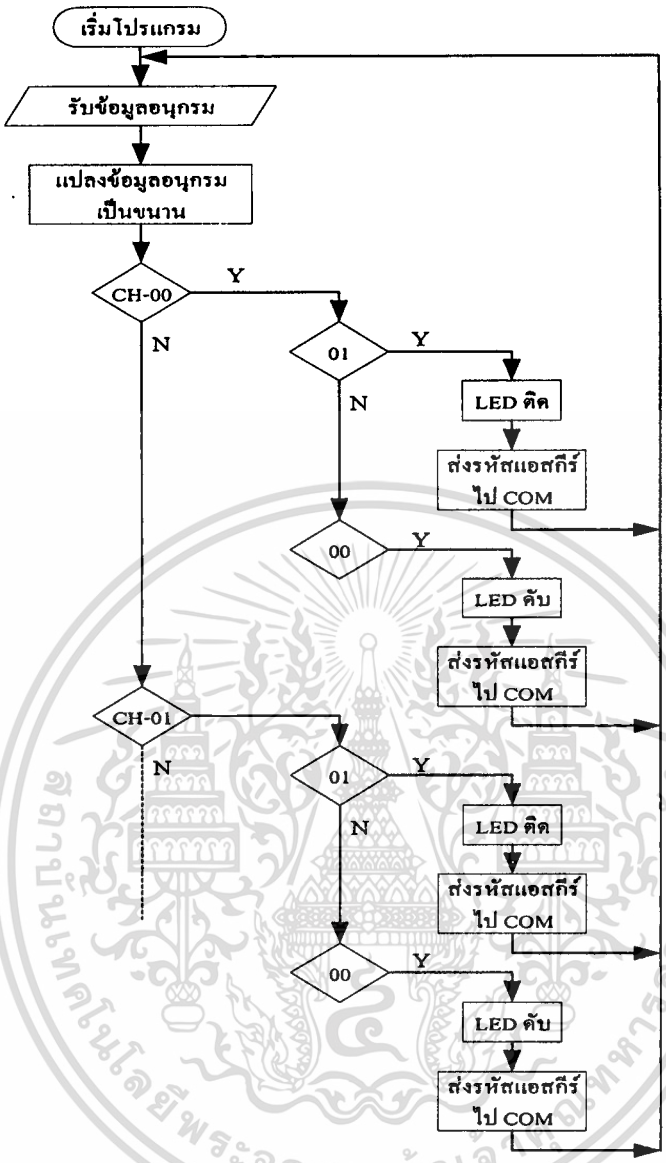
USER PORT1	แอดเดรส 000H+8255 offset addr = actual
PORT A	ตำแหน่งแอดเดรส 0000H+00H = 0000H
PORT B	ตำแหน่งแอสเดรส 0000H+01H = 0001H
PORT C	ตำแหน่งแอสเดรส 0000H+02H = 0002H
PORT D	ตำแหน่งแอสเดรส 0000H+03H = 0003H

ก่อนที่ใช้งานพอร์ท 8255 ผู้ใช้ต้องทำการกำหนดโหมดการทำงาน (configulation) ของพอร์ท A,B และ C ให้เป็นพอร์ทอินพุทหรือเอาท์พุท โดยทำการเขียนค่า control code ไปที่ Mode Port ซึ่ง Mode Port นี้สามารถเขียนได้เท่านั้นไม่สามารถอ่านได้ซึ่งในที่นี้จะกล่าวเฉพาะการทำงานในโหมด 0 ซึ่งเป็นโหมดที่ใช้งานได้สะดวกและง่ายต่อการทำความเข้าใจ ดังแสดงค่า control code ในตารางที่ 7

Port A (PA0-PA7)	Port C บน (PC4-PC7)	Port B (PB0-PB7)	Port C ล่าง (PC0-PC3)	Control code (HEX)
Output	Output	Output	Output	80H
Output	Output	Output	Input	81H
Output	Output	Input	Output	82H
Output	Output	Input	Input	83H
Output	Input	Output	Output	88H
Output	Input	Output	Input	89H
Output	Input	Input	Output	8AH
Output	Input	Input	Input	8BH
Input	Output	Output	Output	90H
Input	Output	Output	Input	91H
Input	Outout	Input	Output	92H
Input	Output	Input	Input	93H
Input	Input	Output	Output	98H
Input	Input	Output	Input	99H
Input	Input	Input	Output	9AH
Input	Input	Input	Input	9BH

ตารางที่ 7 8255 MODE 0 CONFIGULATION

การเขียนโปรแกรมภาษา แอสแซมบลี้ เพื่อให้ ไมโครคอนโทรลเลอร์ #8052 ทำงานตามต้องการ ซึ่งเขียนเป็นโฟว์ชาร์ทได้ดังนี้



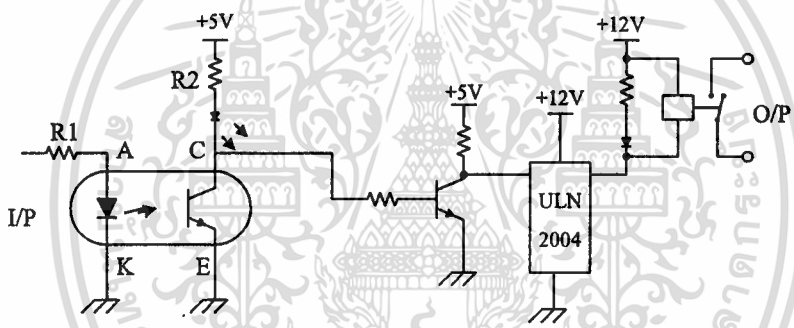
รูปที่ 2.1 แสดงโฟว์ชาร์ทของโปรแกรม

เมื่อนำโปรแกรมไปใส่ลงใน LITHIEM RAM (ซึ่งมีคุณสมบัติ คือยังสามารถรักษาข้อมูลไว้ได้แม้จะทำการปลดแหล่งจ่ายไฟออกแล้วก็ตาม) บนบอร์ดคอนโทรลแล้ว จึงป้อนสัญญาณอนุกรมจาก COMPUTER เข้ามา โดยใช้โปรแกรมเคลไฟน์ซึ่งสามารถสั่งพัลส์อนุกรมออกมาทางพอร์ทอนุกรมได้ ซึ่งพัลส์อนุกรมนี้ก็คือ คำสั่งที่ทำให้ ไมโครคอนโทรลเลอร์ 8051 ทำงาน และ8051 ก็จะทำการควบคุมชุดอุปกรณ์ไฟฟ้าผ่านทาง พอร์ทของ 8255 นั่นเอง ก็จะสามารถควบคุมการปิด-เปิดไฟได้ตามต้องการ

2.2 การทำงานของชุดควบคุมอุปกรณ์ไฟฟ้า

สัญญาณที่ออกจากพอร์ทเอาท์พุท เป็นสัญญาณดิจิตอลมีขนาดแรงดันดังนี้ ลอจิก "1" ประมาณ 5 V และ ลอจิก "0" ประมาณ 0 V ซึ่งขนาดของกระแสที่จะไหลผ่านโหลดได้น้อยและโหลดที่เป็นรีเลย์ ต้องการแรงดัน 12 V ดังนั้น การที่จะนำสัญญาณไปใช้งานจำเป็นต้องมีอุปกรณ์ที่จะเปลี่ยนแรงดันจาก 5 V เป็น 12 V และทำให้กระแสไหลผ่านโหลดได้มากขึ้นด้วย ซึ่งอุปกรณ์นั้นเรียกว่าไดร์เวอร์ (DRIVER)

อุปกรณ์ที่ใช้เป็นไอซี ULN2004 ด้านอินพุทต่อมาจากตัวไอโซเลเตอร์ซึ่งมาจากพอร์ทของ 8255 และมีทรานซิสเตอร์ซึ่งทำหน้าที่เป็นสวิตซ์กลับสัญญาณอินพุทก่อนที่จะเข้าอินพุทของไอซี ด้วยคุณสมบัติของตัวไอซีเองมันจะกลับเฟสของสัญญาณที่ได้รับ คือ ถ้าอินพุทเป็นลอจิก " 1 " เอาท์พุทจะเป็น 0 V และถ้าอินพุทเป็น " 0 " เอาท์พุทจะเป็น 12 V แสดงวงจรดังรูปที่ 2.2

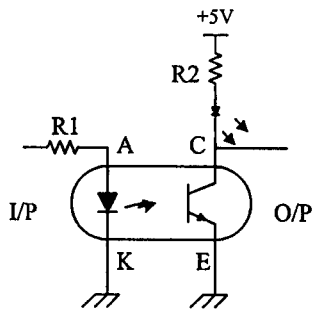


รูปที่ 2.2 แสดงวงจรการควบคุมชุดอุปกรณ์

ดังนั้นถ้า 8255 เป็นลอจิก "0" ที่เอาท์พุทของ ULN2004 จึงเป็น 12V ตัวรีเลย์จึงไม่ทำงาน แต่ถ้าที่เอาท์พุทของ 8255 เป็นลอจิก "1" ที่เอาท์พุทของ ULN2004 เป็น 0V ดังนั้น รีเลย์จึงทำงานเปิดอุปกรณ์ไฟฟ้าได้

2.3 การทำงานของตัวไอโซเลเตอร์

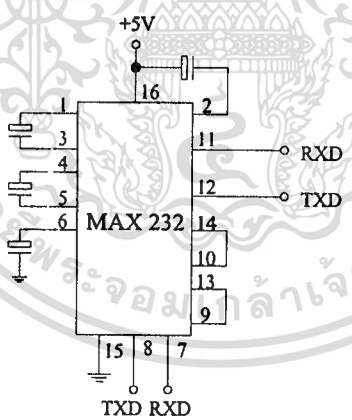
ในโครงงานจะใช้ออปโต้คัปเปลอร์ (opto coupler) มาทำการแยกกราวด์ เพื่อลดสัญญาณรบกวนที่เกิดขึ้นในภาคไดรเวอร์ซึ่งอาจทำให้วงจรทางด้านดิจิตอลทำงานผิดพลาดได้



รูปที่ 2.3 ลักษณะการต่อวงจรออปโต้คัปเปลอร์

2.4 การทำงานของชุดเปลี่ยนมาตรฐาน RS-232/RS-422

ต่ามปกติแล้วพอร์ทอนุกรมที่ส่งมาจาก COMPUTER จะเป็นมาตรฐาน RS-232 ซึ่งมีขีดจำกัดในเรื่องระยะทางที่จะส่งไป ดังนั้นในโครงงานนี้จึงดัดแปลงให้เป็นมาตรฐาน RS-422 ซึ่งส่งไปได้ไกลกว่า โดยใช้ ไอซี #75176 ซึ่งได้กล่าวรายละเอียดไปแล้ว และต้องใช้ร่วมกับตัวเปลี่ยนแรงดัน RS-232 เป็นมาตรฐาน TTL ซึ่งก็คือ ไอซี #MAX232 ดังแสดงได้ดังนี้



รูปที่ 2.4 แสดงการแปลงมาตรฐาน RS-232/TTL

2.5 การเขียนซอฟแวร์(software)

ในโครงงานนี้ได้ใช้โปรแกรมเดลไฟล์ (DELPHI) ซึ่งทำงานบนระบบปฏิบัติการวินโดว์ เป็นตัวส่งสัญญาณควบคุมด้วยพัลส์อนุกรมจากคอมพิวเตอร์ ไปสั่งการให้ตัวไมโครคอนโทรลเลอร์ ซึ่งอยู่บนแผงคอนโทรลทำงาน ปิด-เปิด อุปกรณ์ไฟฟ้า

รายละเอียดของโปรแกรมต่าง ๆ แสดงไว้ในภาคผนวก

บทที่ 3

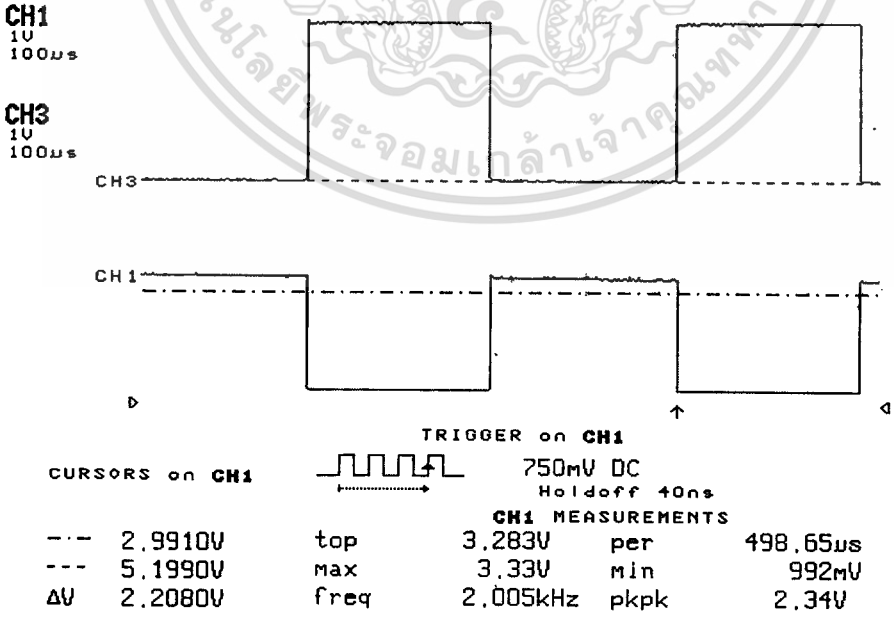
การทดลองและผลการทดลอง

เมื่อนำส่วนต่าง ๆ ของโครงงานมาทดสอบซึ่งได้ผลดังนี้

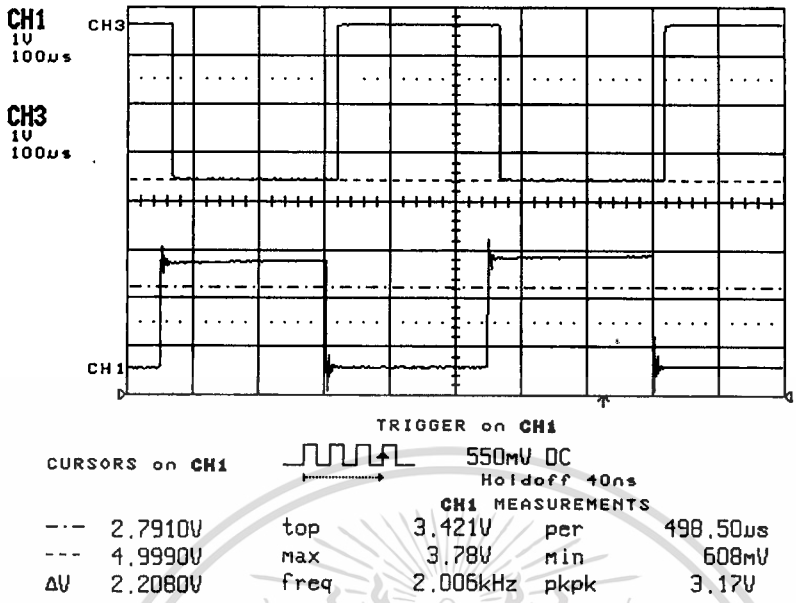
- ทดลองการเขียนโปรแกรมภาษาแอสแซมบลี้ (assembly) เพื่อรับสัญญาณอนุกรมและสั่งงานควบคุมการปิด-เปิดชุดควบคุมอุปกรณ์ไฟฟ้าผ่านทางพอร์ทของ 8255 โดยใช้โปรแกรม Procomplus เป็นตัวสร้างสัญญาณอนุกรมขึ้นมา และบนบอร์ดมี LED เป็นตัวแสดงผล และได้ทำการเขียนโปรแกรมบน computer และทำการ complie ด้วย complier A51 จากนั้นจึง ลิงค์(link) ลงบนบอร์ด

- ทดลองการเขียนโปรแกรมเดลไฟล์ ให้สามารถส่งสัญญาณออกทางพอร์ทอนุกรม เพื่อส่งสัญญาณคำสั่งนี้ไปยังคอนโทรลบอร์ดอีกที ซึ่งโปรแกรมก็สามารถส่งสัญญาณอนุกรมออกทางพอร์ทอนุรมได้ตามต้องการซึ่งที่พอร์ทนี้เป็นไปตามมาตรฐาน RS-232

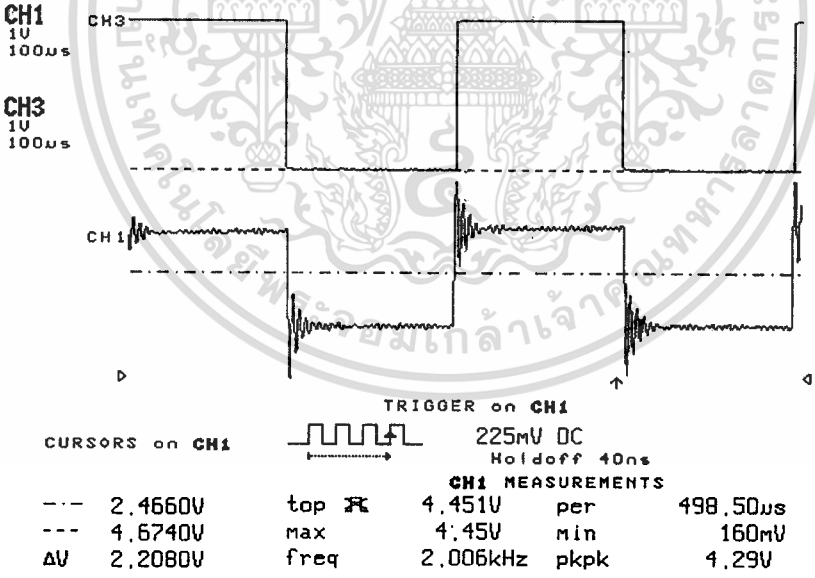
- ทดลองชุดการเปลี่ยนมาตรฐานจาก RS-232 ไปเป็น RS-422 ซึ่งสามารถส่งเป็นระยะทางที่ไกลกว่า มาตรฐาน RS-232 ซึ่งระยะทางสูงสุดที่ RS-422 จะส่งไปได้ประมาณ 4,000 ฟุต



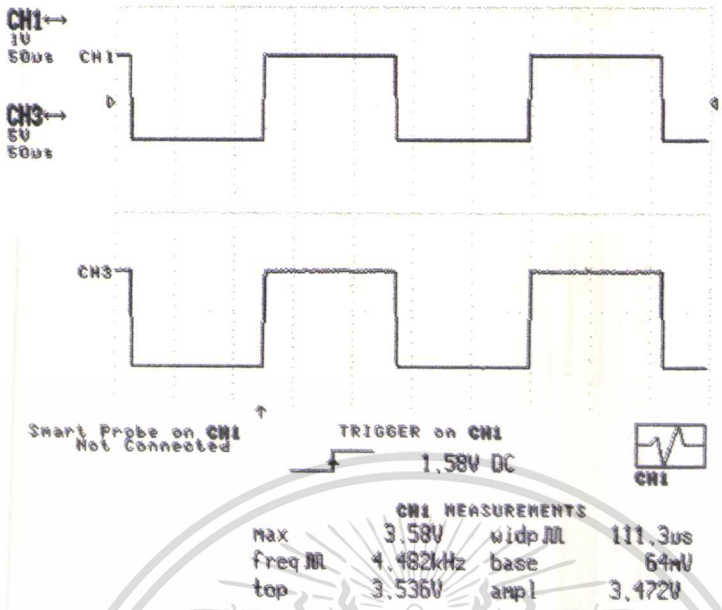
รูปที่3.1 แสดงสัญญาณอนุกรม RS-422 ที่ระยะ 0 เมตร



รูปที่3.2 แสดงสัญญาณอนุกรม RS422 ที่ระยะ 45 เมตร



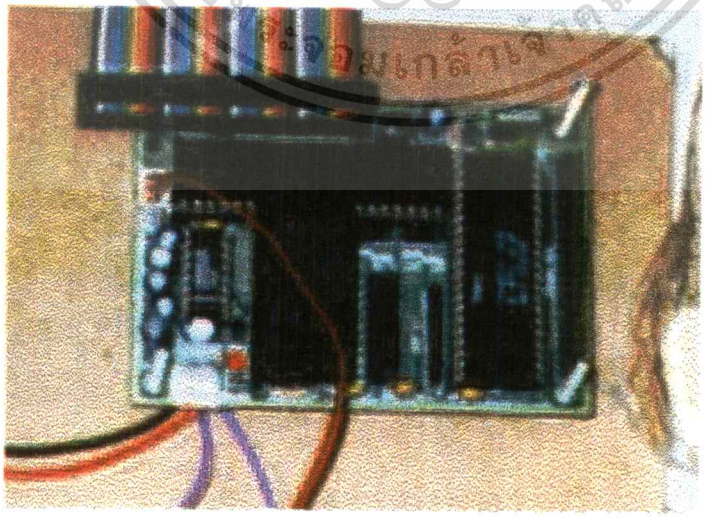
รูปที่ 3.3 แสดงสัญญาณอนุกรม RS422 ที่ระยะ 150



รูปที่ 3.4 แสดง สัญญาณอนุกรม RS-232 ที่ทำการส่งด้วย RS-422

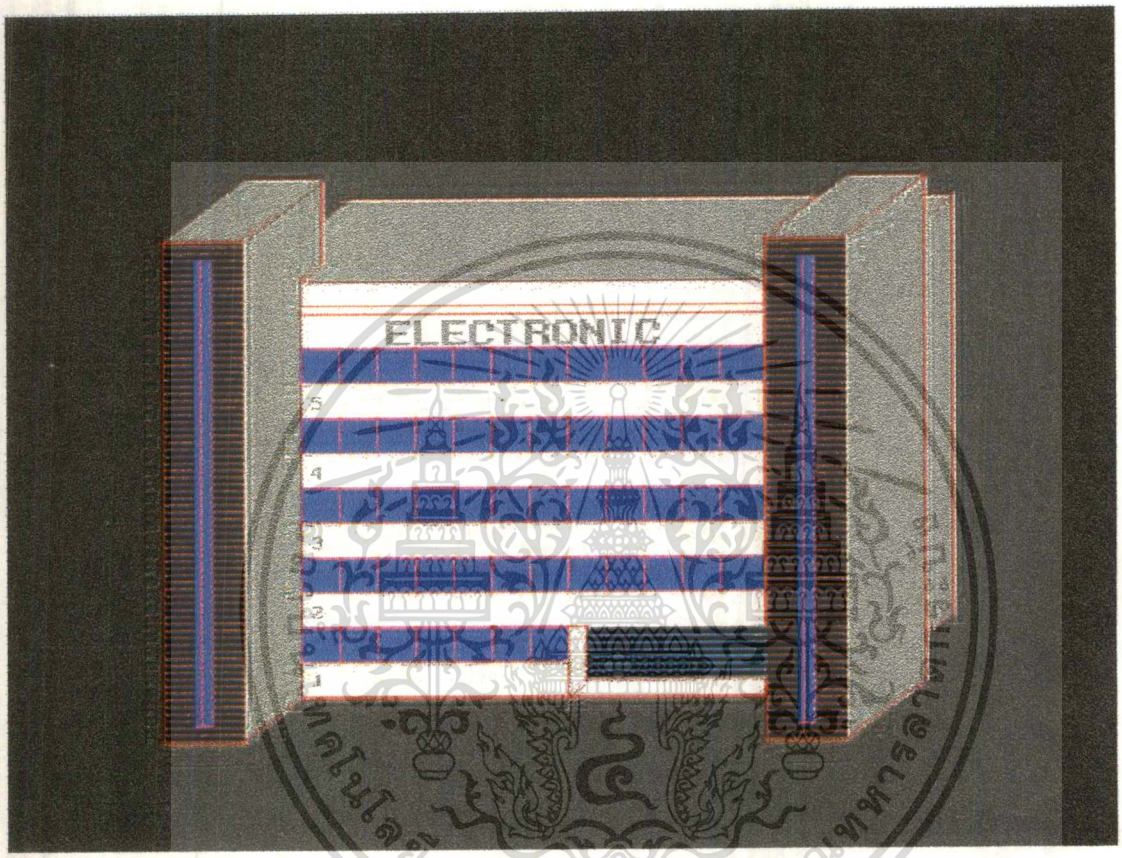
จากรูปที่3.4 จะเห็นได้ว่าสัญญาณที่คืนกลับนั้นมีลักษณะใกล้เคียงกับด้านส่ง

- ทดลองวงจรของชุดควบคุมอุปกรณ์ไฟฟ้า โดยใช้รีเลย์เป็นตัวควบคุมการปิด-เปิด และมีออปโต้ไอโซเลเตอร์เป็นตัวแยกกราวด์ของสองวงจรออกจากกันเพื่อป้องกันสัญญาณรบกวนโดยวงจรนี้ต่อมาจากตัวคอนโทรลบอร์ดซึ่งสามารถทำงานได้ตามต้องการ

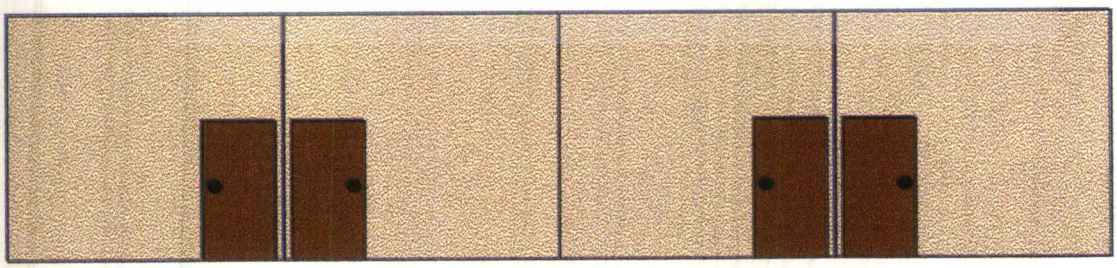


รูปที่ 3.5 ลักษณะของชุดควบคุมอุปกรณ์ไฟฟ้า

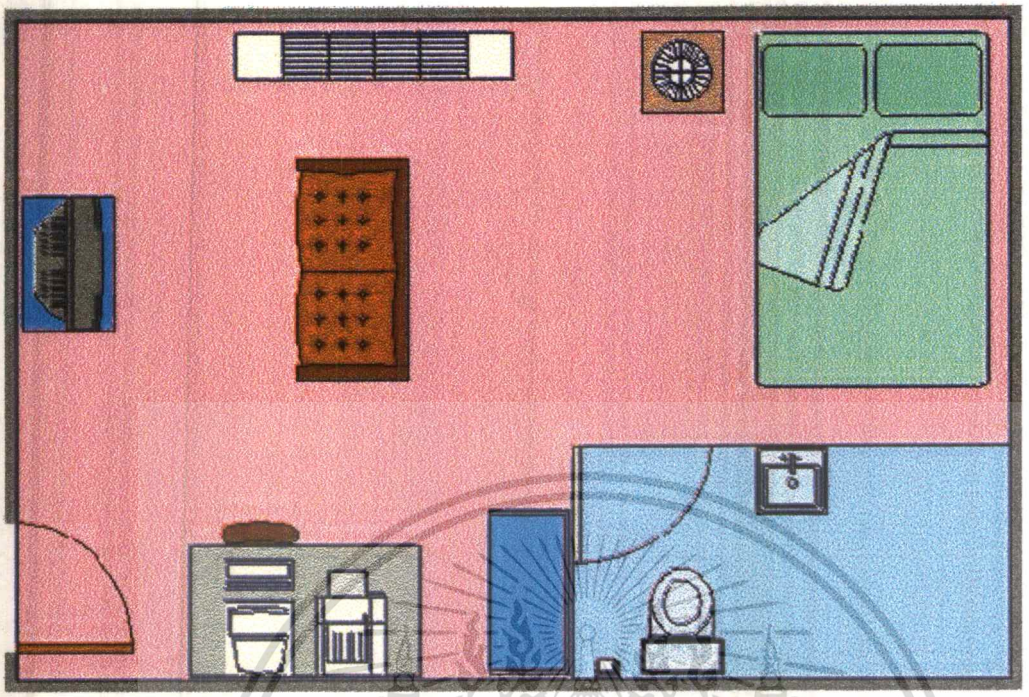
- ทดลองการควบคุมอุปกรณ์ไฟฟ้าซึ่งสั่งงานโดยโปรแกรมเคลไฟล์ ซึ่งจะใช้ภาพเสมือนจริง (real picture) ซึ่งจำลองมาไว้บน computer เพื่อให้ผู้ที่ไม่คุ้นเคยกับการใช้ computer ก็สามารถใช้งานได้โดยใช้ภาพเป็นสื่อ



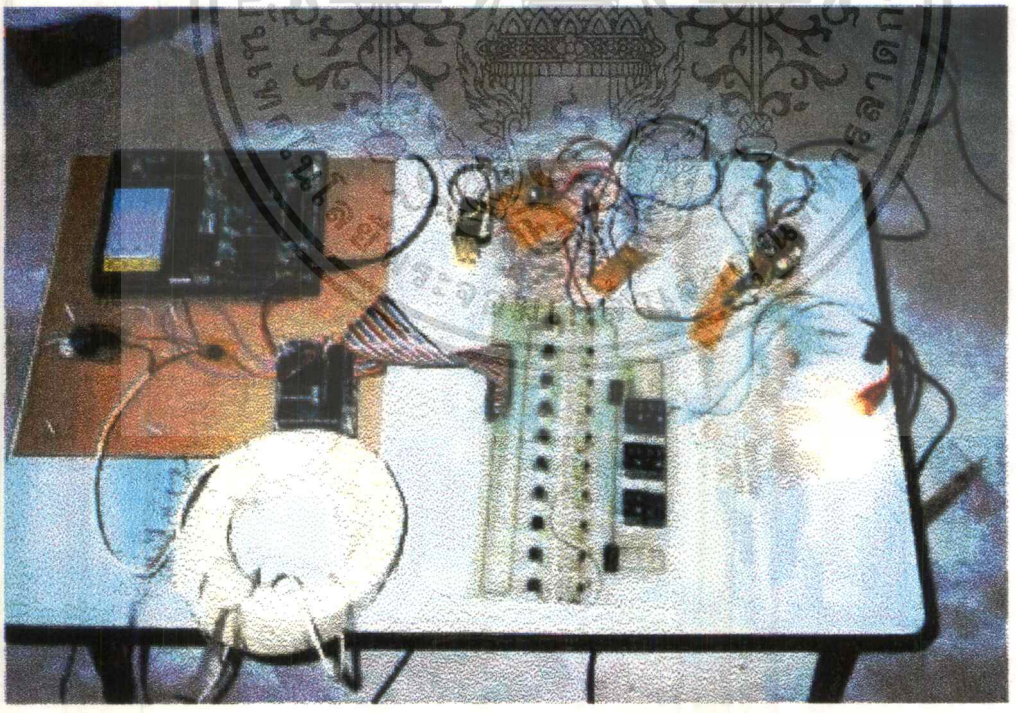
รูปที่ 3.6 แสดงภาพมุมกว้างของตัวอาคาร



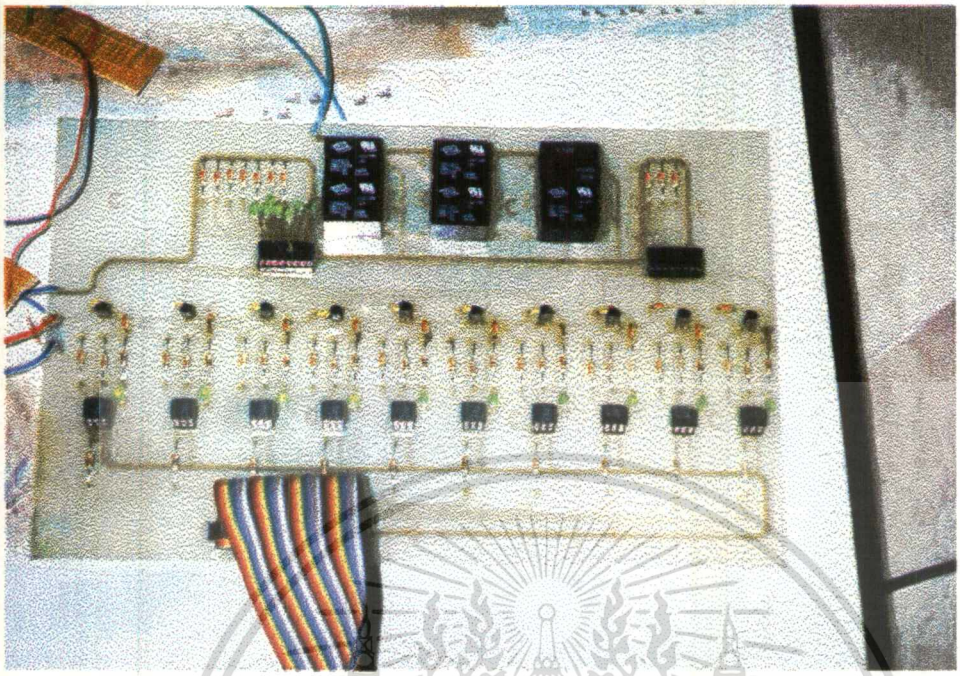
รูปที่ 3.7 แสดงชั้นของอาคาร



รูปที่ 3.8 แสดงห้องต่าง ๆ ที่ใช้สั่งงาน



รูปที่ 3.9 แสดงส่วนประกอบต่าง ๆ ของโครงงาน



รูปที่ 3.10 ชุดควบคุมการปิด-เปิดอุปกรณ์

บทที่ 4

สรุปผลการทดลอง

จากการทดลองทั้งหมดที่ผ่านมา สรุปผลได้ว่าตัวเครื่องสามารถทำงานได้อย่างดี บรรลุจุดประสงค์ที่ได้ตั้งขอบเขตไว้ก่อนนี้ โดยสามารถเรียงลำดับขั้นตอนการทำงานได้ดังนี้

1.เมื่อเปิดโปรแกรมขึ้นเราสามารถที่จะเลือกชั้นต่างๆจากจำนวน 4 ชั้นดังรูปที่ 3.7

2.เมื่อทำการเลือกชั้นแล้วก็ทำการเลือกห้องที่เราต้องการควบคุมดังรูปที่ 3.8

3.เมื่อได้ดังนี้แล้วก็ทำการเลือกอุปกรณ์ที่เราต้องการควบคุมโดยที่ใช้เม้าส์ ไปแตะที่ตัวอุปกรณ์ POINTER ที่ปรากฏอยู่บนจอคอมพิวเตอร์จะเปลี่ยนเป็นรูปมือ พร้อมทั้งขึ้นข้อความว่าอุปกรณ์ตัวนั้นๆคืออะไร โดยที่สถานะของอุปกรณ์ได้แสดงไว้ที่ขอบจอด้านล่าง การออกคำสั่งให้อุปกรณ์ ปิด-เปิด ก็เพียงแค่ใช้ปลายนิ้วคลิ๊กที่เม้าส์ปุ่มซ้ายมือ สถานะของอุปกรณ์ก็จะเปลี่ยนไปตามที่เราต้องการ ซึ่งเราสามารถเลือกทำงานในชั้นอื่นๆได้เช่นเดียวกัน โดยโปรแกรมจะยังคงสถานะเดิมไว้อยู่และเราก็สามารถใช้โปรแกรมอื่นๆได้ในขณะเดียวกันกับใช้โปรแกรมนี้

ตัวโปรแกรมเดลไฟน์จะส่งสัญญาณเป็นพัลส์อนุกรมออกมาผ่านทางพอร์ทอนุกรมและถูกแปลงมาตรฐานเป็น RS-422 โดยตัวส่ง และส่งไปทางสายนำสัญญาณซึ่งในโครงงานได้ใช้สายโทรศัพท์ (AWG#22) ผ่านไปยังตัวรับก็จะแปลงมาตรฐานกลับอีกครั้งหนึ่ง จากนั้นพัลส์เหล่านี้จะไปเป็นคำสั่งให้ตัว CONTROLLER ที่อยู่บนบอร์ดทำการควบคุมอุปกรณ์อีกทีหนึ่ง โดยที่ผลการทดลองและสัญญาณต่างๆได้แสดงไว้แล้วในบทที่ 3

โครงงานนี้ได้มุ่งหวังให้ผู้ที่ไม่คุ้นเคยกับการใช้คอมพิวเตอร์ ก็สามารถใช้งานได้อย่างไม่ยากเย็นนักเนื่องจากมีภาพเป็นสื่อในการทำความเข้าใจอยู่แล้ว อย่างไรก็ตามการใช้งานโปรแกรมเดลไฟน์สามารถยืดหยุ่นได้ตามจุดประสงค์ของผู้ใช้ซึ่งสามารถดัดแปลงพัฒนาโปรแกรมนี้ใช้งานในรูปแบบอื่นๆได้อีกหลากหลาย

โปรแกรมควบคุมการปิด-เปิดอุปกรณ์ไฟฟ้าที่เขียนด้วยภาษาแอสแซมบลี้

```
;*********************************
;*   ORIGIN 0000H                *
;*   PROT   0000H-0003H          *
;*   WEE MOVE LIGHT PROGRAM      *
;*   USE TWE DELAY TIME          *
;*********************************
            ORG         0000H

    V_DLYTIME11:  EQU       01H
    V_DLYTIME12:  EQU       50H
    V_DLYTIME13:  EQU       00H
    V_DLYTIME21:  EQU       01H
    V_DLYTIME22:  EQU       20H
    V_DLYTIME23:  EQU       00H

    V_INIPORT:    EQU       80H
    V_PORTA:      EQU       0000H
    V_PORTB:      EQU       0001H
    V_PORTC:      EQU       0002H
    V_CONPORT:    EQU       0003H
    V_RAM_PORTA:  EQU       20H
    V_RAM_PORTB:  EQU       21H
    V_RAM_OFF:    EQU       22H
    V_RAM_ON:     EQU       23H
    V_RAM_CH:     EQU       24H
    V_RAM_STATE:  EQU       25H
    V_START1:     EQU       'V'
    V_START2:     EQU       'v'
    V_CANCEL:     EQU       '#'
    V_TABLE:      EQU       1000H
```

```
CH00_ON:    EQU    00000001B
CH01_ON:    EQU    00000010B
CH02_ON:    EQU    00000100B
CH03_ON:    EQU    00001000B
CH04_ON:    EQU    00010000B
CH05_ON:    EQU    00100000B
CH06_ON:    EQU    01000000B
CH07_ON:    EQU    10000000B
PORTA_ON:   EQU    11111111B
PORTB_ON:   EQU    11111111B
CH00_OFF:   EQU    11111110B
CH01_OFF:   EQU    11111101B
CH02_OFF:   EQU    11111011B
CH03_OFF:   EQU    11110111B
CH04_OFF:   EQU    11101111B
CH05_OFF:   EQU    11011111B
CH06_OFF:   EQU    10111111B
CH07_OFF:   EQU    01111111B
PORTA_OFF:  EQU    00000000B
PORTB_OFF:  EQU    00000000B


END_TABLE:  EQU    00H


            MOV    SP,#5FH
WAIT:       MOV    R1,#40H
WAIT1:      MOV    R2,#00H
            DJNZ   R2,$
            DJNZ   R1,WAIT1
```

```
V_INITIAL:  MOV     A,#V_INIPORT
            MOV     DPTR,#V_CONPORT
            MOVX    @DPTR,A
            MOV     V_RAM_PORTA,#00H
            MOV     A,V_RAM_PORTA
            MOV     DPTR,#V_PORTA
            MOVX    @DPTR,A

            MOV     V_RAM_PORTB,#00H
            MOV     A,V_RAM_PORTB
            MOV     DPTR,#V_PORTB
            MOVX    @DPTR,A

            LJMP    V_MAIN

V_INIT_SP:  PUSH    A
            MOV     TMOD,#00100000B
            MOV     SCON,#01010010B
            MOV     PCON,#10000000B
            MOV     A,#0FDH
            MOV     TH1,A
            SETB    TR1         ;TR1 IS TCON.6
            POP     A
            RET

            ORG     0100H

V_MAIN:     LCALL   V_INIT_SP
            LCALL   V_RX
            LCALL   V_TX
            CJNE    A,#V_START1,V_CHKSTART
```

```
                AJMP        V_START
V_CHKSTART:     CJNE        A,#V_START2,V_MAIN
V_START:        LCALL       V_RX
                LCALL       V_TX
                CJNE        A,#V_CANCEL,V_CHK_CH
                AJMP        V_MAIN
V_CHK_CH:       MOV         V_RAM_CH,A
V_CHK_STATE:    LCALL       V_RX
                LCALL       V_TX      ;ADD TO CHECK DATA

                CJNE        A,#V_CANCEL,V_CHK_OFF
                AJMP        V_MAIN


V_CHK_OFF:      CJNE        A,#'0',V_CHK_ON
                AJMP        V_CHK_PASS
V_CHK_ON:       CJNE        A,#'1',V_CHK_STATE

V_CHK_PASS:     MOV         V_RAM_STATE,A

                MOV         A,V_RAM_CH

NEXT00:         CJNE        A,#'0',NEXT01
                LJMP        V_CH00
NEXT01:         CJNE        A,#'1',NEXT02
                LJMP        V_CH01
NEXT02:         CJNE        A,#'2',NEXT03
                LJMP        V_CH02
NEXT03:         CJNE        A,#'3',NEXT04
                LJMP        V_CH03
NEXT04:         CJNE        A,#'4',NEXT05
                LJMP        V_CH04
```

```
NEXT05:   CJNE    A,#'5',NEXT06
          LJMP    V_CH05
NEXT06:   CJNE    A,#'6',NEXT07
          LJMP    V_CH06
NEXT07:   CJNE    A,#'7',NEXT08
          LJMP    V_CH07
NEXT08:   CJNE    A,#'8',NEXT09
          LJMP    V_CH08
NEXT09:   CJNE    A,#'9',NEXT10
          LJMP    V_CH09
NEXT10:   CJNE    A,#'A',NEXT10_1
          LJMP    V_CH10
NEXT10_1: CJNE    A,#'a',NEXT11
          LJMP    V_CH10
NEXT11:   CJNE    A,#'B',NEXT11_1
          LJMP    V_CH11
NEXT11_1: CJNE    A,#'b',NEXT12
          LJMP    V_CH11
NEXT12:   CJNE    A,#'C',NEXT12_1
          LJMP    V_CH12
NEXT12_1: CJNE    A,#'c',NEXT13
          LJMP    V_CH12
NEXT13:   CJNE    A,#'D',NEXT13_1
          LJMP    V_CH13
NEXT13_1: CJNE    A,#'d',NEXT14
          LJMP    V_CH13
NEXT14:   CJNE    A,#'E',NEXT14_1
          LJMP    V_CH14
NEXT14_1: CJNE    A,#'e',NEXT15
          LJMP    V_CH14
NEXT15:'  CJNE    A,#'F',NEXT15_1
          LJMP    V_CH15
```

```
NEXT15_1:   CJNE        A,#'f',NEXT_PORTA
            LJMP        V_CH15
NEXT_PORTA:     CJNE    A,#'@',NEXT_PORTB
            LJMP        V_PORTA_ON
NEXT_PORTB:     CJNE    A,#'*',NEXT_READAB
            LJMP        V_PORTB_ON
NEXT_READAB:    CJNE    A,#'R',NEXT_READAB_1
            LJMP        V_READ_PORTAB
NEXT_READAB_1: CJNE     A,#'r',V_END
            LJMP        V_READ_PORTAB

V_END:      AJMP        V_MAIN

            ORG         0D00H

V_TX:       PUSH  IE
            CLR   ES        ;ES IS IE.4
            CLR   TI        ;TI IS SCON.1
            MOV   SBUF,A
            JNB   TI,$
            CLR   TI        ;CLEAR TI OR NOT TEST IT
            POP   IE
            RET



            ORG         0E00H

V_RX:       PUSH  IE
            CLR   ES
GETCHR1:    JNB   RI,$
            CLR   RI        ;RI IS SCON.0
```

```
        MOV   A,SBUF
        POP   IE
        RET

        .

        ORG         0F00H

V_CH00: MOV         A,V_RAM_STATE
        CJNE        A,#'1',V_CH00_OFF
        MOV         V_RAM_ON,#CH00_ON
        LCALL       V_ON_PORTA
        MOV         A,#00H
        LCALL       V_SEND_PC
        LJMP        V_MAIN

V_CH00_OFF: MOV     V_RAM_OFF,#CH00_OFF
        LCALL       V_OFF_PORTA
        MOV         A,#01H
        LCALL       V_SEND_PC
        LJMP        V_MAIN

V_CH01: MOV         A,V_RAM_STATE
        CJNE        A,#'1',V_CH01_OFF
        MOV         V_RAM_ON,#CH01_ON
        LCALL       V_ON_PORTA
        MOV         A,#02H
        LCALL       V_SEND_PC
        LJMP        V_MAIN

V_CH01_OFF: MOV     V_RAM_OFF,#CH01_OFF
        LCALL       V_OFF_PORTA
```

```
                MOV     A,#03H
                LCALL   V_SEND_PC
                LJMP    V_MAIN

V_CH02:         MOV     A,V_RAM_STATE
                CJNE    A,#'1',V_CH02_OFF
                MOV     V_RAM_ON,#CH02_ON
                LCALL   V_ON_PORTA
                MOV     A,#04H
                LCALL   V_SEND_PC
                LJMP    V_MAIN

V_CH02_OFF:     MOV     V_RAM_OFF,#CH02_OFF
                LCALL   V_OFF_PORTA
                MOV     A,#05H
                LCALL   V_SEND_PC
                LJMP    V_MAIN

V_CH03:         MOV     A,V_RAM_STATE
                CJNE    A,#'1',V_CH03_OFF
                MOV     V_RAM_ON,#CH03_ON
                LCALL   V_ON_PORTA
                MOV     A,#06H
                LCALL   V_SEND_PC
                LJMP    V_MAIN

V_CH03_OFF:     MOV     V_RAM_OFF,#CH03_OFF
                LCALL   V_OFF_PORTA
                MOV     A,#07H
                LCALL   V_SEND_PC
                LJMP    V_MAIN
```

```
V_CH04:     MOV         A,V_RAM_STATE
            CJNE        A,#'1',V_CH04_OFF
            MOV         V_RAM_ON,#CH04_ON
            LCALL       V_ON_PORTA
            MOV         A,#08H
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH04_OFF: MOV         V_RAM_OFF,#CH04_OFF
            LCALL       V_OFF_PORTA
            MOV         A,#09H
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH05:     MOV         A,V_RAM_STATE
            CJNE        A,#'1',V_CH05_OFF
            MOV         V_RAM_ON,#CH05_ON
            LCALL       V_ON_PORTA
            MOV         A,#0AH
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH05_OFF: MOV         V_RAM_OFF,#CH05_OFF
            LCALL       V_OFF_PORTA
            MOV         A,#0BH
            LCALL       V_SEND_PC
            LJMP        V_MAIN
```

```
V_CH06:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH06_OFF
            MOV     V_RAM_ON,#CH06_ON
            LCALL   V_ON_PORTA
            MOV     A,#0CH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH06_OFF: MOV     V_RAM_OFF,#CH06_OFF
            LCALL   V_OFF_PORTA
            MOV     A,#0DH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH07:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH07_OFF
            MOV     V_RAM_ON,#CH07_ON
            LCALL   V_ON_PORTA
            MOV     A,#0EH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH07_OFF: MOV     V_RAM_OFF,#CH07_OFF
            LCALL   V_OFF_PORTA
            MOV     A,#0FH
            LCALL   V_SEND_PC
            LJMP    V_MAIN
```

```
V_CH08:     MOV         A,V_RAM_STATE
            CJNE        A,#'1',V_CH08_OFF
            MOV         V_RAM_ON,#CH00_ON
            LCALL       V_ON_PORTB
            MOV         A,#10H
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH08_OFF: MOV         V_RAM_OFF,#CH00_OFF
            LCALL       V_OFF_PORTB
            MOV         A,#11H
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH09:     MOV         A,V_RAM_STATE
            CJNE        A,#'1',V_CH09_OFF
            MOV         V_RAM_ON,#CH01_ON
            LCALL       V_ON_PORTB
            MOV         A,#12H
            LCALL       V_SEND_PC
            LJMP        V_MAIN

V_CH09_OFF: MOV         V_RAM_OFF,#CH01_OFF
            LCALL       V_OFF_PORTB
            MOV         A,#13H
            LCALL       V_SEND_PC
            LJMP        V_MAIN
```

```
V_CH10:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH10_OFF
            MOV     V_RAM_ON,#CH02_ON
            LCALL   V_ON_PORTB
            MOV     A,#14H
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH10_OFF: MOV     V_RAM_OFF,#CH02_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#15H
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH11:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH11_OFF
            MOV     V_RAM_ON,#CH03_ON
            LCALL   V_ON_PORTB
            MOV     A,#16H
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH11_OFF: MOV     V_RAM_OFF,#CH03_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#17H
            LCALL   V_SEND_PC
            LJMP    V_MAIN
```

```
V_CH12:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH12_OFF
            MOV     V_RAM_ON,#CH04_ON
            LCALL   V_ON_PORTB
            MOV     A,#18H
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH12_OFF: MOV     V_RAM_OFF,#CH04_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#19H
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH13:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH13_OFF
            MOV     V_RAM_ON,#CH05_ON
            LCALL   V_ON_PORTB
            MOV     A,#1AH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH13_OFF: MOV     V_RAM_OFF,#CH05_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#1BH
            LCALL   V_SEND_PC
            LJMP    V_MAIN
```

```
V_CH14:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH14_OFF
            MOV     V_RAM_ON,#CH06_ON
            LCALL   V_ON_PORTB
            MOV     A,#1CH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH14_OFF: MOV     V_RAM_OFF,#CH06_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#1DH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH15:     MOV     A,V_RAM_STATE
            CJNE    A,#'1',V_CH15_OFF
            MOV     V_RAM_ON,#CH07_ON
            LCALL   V_ON_PORTB
            MOV     A,#1EH
            LCALL   V_SEND_PC
            LJMP    V_MAIN

V_CH15_OFF: MOV     V_RAM_OFF,#CH07_OFF
            LCALL   V_OFF_PORTB
            MOV     A,#1FH
            LCALL   V_SEND_PC
            LJMP    V_MAIN
```

```
V_PORTA_ON:   MOV     A,V_RAM_STATE
              CJNE    A,#'1',V_PORTA_OFF
              MOV     V_RAM_ON,#PORTA_ON
              LCALL   V_ON_PORTA
              MOV     A,#20H
              LCALL   V_SEND_PC
              LJMP    V_MAIN

V_PORTA_OFF:  MOV     _RAM_OFF,#PORTA_OFF
              LCALL   V_OFF_PORTA
              MOV     A,#21H
              LCALL   V_SEND_PC
              LJMP    V_MAIN

V_PORTB_ON:   MOV     A,V_RAM_STATE
              CJNE    A,#'1',V_PORTB_OFF
              MOV     V_RAM_ON,#PORTB_ON
              LCALL   V_ON_PORTB
              MOV     A,#22H
              LCALL   V_SEND_PC
              LJMP    V_MAIN

V_PORTB_OFF:  MOV     V_RAM_OFF,#PORTB_OFF
              LCALL   V_OFF_PORTB
              MOV     A,#23H
              LCALL   V_SEND_PC
              LJMP    V_MAIN

V_READ_PORTAB: MOV    A,V_RAM_STATE
              CJNE    A,#'0',V_READ_PORTB
              MOV     A,V_RAM_PORTA
```

```
                LCALL   V_INIT_SP
                LCALL   V_TX
                LJMP    V_MAIN

V_READ_PORTB:   MOV     A,V_RAM_PORTB
                LCALL   V_INIT_SP
                LCALL   V_TX
                LJMP    V_MAIN

V_ON_PORTA:     MOV     DPTR,#V_PORTA
                MOV     A,V_RAM_PORTA
                ORL     A,V_RAM_ON
                MOV     V_RAM_PORTA,A
                MOVX    @DPTR,A
                RET

V_OFF_PORTA:    MOV     DPTR,#V_PORTA
                MOV     A,V_RAM_PORTA
                ANL     A,V_RAM_OFF
                MOV     V_RAM_PORTA,A
                MOVX    @DPTR,A
                RET

V_ON_PORTB:     MOV     DPTR,#V_PORTB
                MOV     A,V_RAM_PORTB
                ORL     A,V_RAM_ON
                MOV     V_RAM_PORTB,A
                MOVX    @DPTR,A
                RET

V_OFF_PORTB:    MOV     DPTR,#V_PORTB
```

```
                MOV         A,V_RAM_PORTB
                ANL         A,V_RAM_OFF
                MOV         V_RAM_PORTB,A
                MOVX        @DPTR,A
                RET

V_SEND_PC:      LCALL       V_INIT_SP
                LCALL       V_TX

                RET

                END
```

โปรแกรม build.pas

```
unit build;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComDrv32, ExtCtrls;

type
  TStructure = class(TForm)
    CommPortDriver: TCommPortDriver;
    Panel1: TPanel;
    Image1: TImage;
    bottom1: TShape;
    bottom4: TShape;
    bottom2: TShape;
    Bottom3: TShape;
    procedure buttom1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure buttom2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Bottom3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
 procedure ApplyCommSettings;
 end;

var
 Structure: TStructure;

implementation

uses Floor_1, Floor_2, Floor_3, Floor_4, Title;

{$R *.DFM}
procedure TStructure.ApplyCommSettings;
var wasConnected: boolean;
begin
wasConnected := CommPortDriver.Connect;
if wasConnected then
end;

procedure TStructure.buttom1MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
begin
   Floor1.Showmodal ;
end;

procedure TStructure.buttom2MouseDown(Sender: TObject;
 Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    Floor2.Showmodal;
end;
```

```
procedure TStructure.Bottom3MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    Floor3.Showmodal;
end;

procedure TStructure.bottom4MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    Floor4.Showmodal;
end;

procedure TStructure.FormCreate(Sender: TObject);
begin
      Sleep(600);
      Intro.Hide;
end;
end.
```

โปรแกรม Floor_1.pas

```
unit Floor_1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TFloor1 = class(TForm)
    Panel1: TPanel;
    Image: TImage;
    bottom1: TShape;
  procedure bottom1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Floor1: TFloor1;

implementation

uses Room_1;

{$R *.DFM}
```

```
procedure TFloor1.bottom1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
   room1.ShowModal;
end;


end.
```

โปรแกรม Floor_2.pas

```
unit Floor_2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TFloor2 = class(TForm)
    Panel1: TPanel;
    Image: TImage;
    bottom: TShape;
    procedure bottomMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Floor2: TFloor2;

implementation

uses Room_2;

{$R *.DFM}
```

```
procedure TFloor2.bottomMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    Room2.ShowModal;
end;


end.
```

โปรแกรม Floor_3.pas

```
unit Floor_3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TFloor3 = class(TForm)
    Panel1: TPanel;
    Image: TImage;
    bottom: TShape;
    procedure bottomMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Floor3: TFloor3;

implementation

uses Room_3;

{$R *.DFM}
```

```
procedure TFloor3.bottomMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    Room3.Showmodal;
end;


end.
```

โปรแกรม Floor_4.pas

```
unit Floor_4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TFloor4 = class(TForm)
    Panel1: TPanel;
    Image: TImage;
    bottom: TShape;
    procedure bottomMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Floor4: TFloor4;

implementation

uses Room_4;

{$R *.DFM}
```

```
procedure TFloor4.bottomMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    Room4.Showmodal;
end;


end.
```

โปรแกรม Room_1.pas

```
unit Room_1;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, ComCtrls, ExtCtrls;

type
 TRoom1 = class(TForm)
  Image: TImage;
  bottom1: TShape;
  StatusBar: TStatusBar;
  position1: TShape;
  Font1: TLabel;
  bottom2: TShape;
  bottom3: TShape;
  position2: TShape;
  font2: TLabel;
  position3: TShape;
  font3: TLabel;
  bottom4: TShape;
  bottom5: TShape;
  position4: TShape;
  font4: TLabel;
  position5: TShape;
  font5: TLabel;
  Television: TLabel;
  Label1: TLabel;
  Label2: TLabel;
```

```
    Label3: TLabel;
    Label4: TLabel;
    procedure bottom1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom5MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Room1: TRoom1;

implementation

uses build;

{$R *.DFM}

procedure TRoom1.bottom1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
```

```
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom1.Tag = 0 then
begin
s:='V01';
Structure.CommPortDriver.SendString(s);
bottom1.Tag := 1;
position1.Brush.Color := clgreen ;
font1.Caption := 'ON ';
end
else
begin
s := 'V00';
Structure.CommPortDriver.SendString(s);
bottom1.tag := 0;
position1.Brush.Color := clred ;
font1.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;
```

```
procedure TRoom1.bottom2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom2.Tag = 0 then
begin
s:='V11';
Structure.CommPortDriver.SendString(s);
bottom2.Tag := 1;
position2.Brush.Color := clgreen ;
font2.Caption := 'ON ';
end
else
begin
s := 'V10';
Structure.CommPortDriver.SendString(s);
bottom2.tag := 0;
position2.Brush.Color := clred ;
font2.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;
```

```
end;

procedure TRoom1.bottom3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom3.Tag = 0 then
begin
s:='V21';
Structure.CommPortDriver.SendString(s);
bottom3.Tag := 1;
position3.Brush.Color := clgreen ;
font3.Caption := 'ON ';
end
else
begin
s := 'V20';
Structure.CommPortDriver.SendString(s);
bottom3.tag := 0;
position3.Brush.Color := clred ;
font3.Caption := 'OFF ';
end

end
else
begin
```

```
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;

procedure TRoom1.bottom4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom4.Tag = 0 then
begin
s:='V31';
Structure.CommPortDriver.SendString(s);
bottom4.Tag := 1;
position4.Brush.Color := clgreen ;
font4.Caption := 'ON ';
end
else
begin
s := 'V30';
Structure.CommPortDriver.SendString(s);
bottom4.tag := 0;
position4.Brush.Color := clred ;
font4.Caption := 'OFF ';
end
```

```
end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;

procedure TRoom1.bottom5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom5.Tag = 0 then
begin
s:='V41';
Structure.CommPortDriver.SendString(s);
bottom5.Tag := 1;
position5.Brush.Color := clgreen ;
font5.Caption := 'ON ';
end
else
begin
s := 'V40';
Structure.CommPortDriver.SendString(s);
bottom5.tag := 0;
position5.Brush.Color := clred ;
```

```
font5.Caption := 'OFF ';
end


end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;


end;


end.
```

โปรแกรม Room_2.pas

```
unit Room_2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls;

type
  TRoom2 = class(TForm)
    Image: TImage;
    StatusBar: TStatusBar;
    bottom1: TShape;
    bottom2: TShape;
    bottom3: TShape;
    position1: TShape;
    font1: TLabel;
    position2: TShape;
    font2: TLabel;
    position3: TShape;
    font3: TLabel;
    Label2: TLabel;
    Label1: TLabel;
    Label3: TLabel;
    procedure bottom1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Room2: TRoom2;

implementation

uses build;

{$R *.DFM}

procedure TRoom2.bottom1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom1.Tag = 0 then
begin
s:='V51';
Structure.CommPortDriver.SendString(s);
bottom1.Tag := 1;
position1.Brush.Color := clgreen ;
font1.Caption := 'ON ';
end
```

```
else
begin
s := 'V50';
Structure.CommPortDriver.SendString(s);
bottom1.tag := 0;
position1.Brush.Color := clred ;
font1.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;
```

```
procedure TRoom2.bottom2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom2.Tag = 0 then
begin
```

```
s:='V61';
Structure.CommPortDriver.SendString(s);
bottom2.Tag := 1;
position2.Brush.Color := clgreen ;
font2.Caption := 'ON ';
end
else
begin
s := 'V60';
Structure.CommPortDriver.SendString(s);
bottom2.tag := 0;
position2.Brush.Color := clred ;
font2.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;


procedure TRoom2.bottom3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then
```

```
begin
StatusBar.SimpleText := 'Connected';
if bottom3.Tag = 0 then
begin
s:='V71';
Structure.CommPortDriver.SendString(s);
bottom3.Tag := 1;
position3.Brush.Color := clgreen ;
font3.Caption := 'ON ';
end
else
begin
s := 'V70';
Structure.CommPortDriver.SendString(s);
bottom3.tag := 0;
position3.Brush.Color := clred ;
font3.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;


end.
```

โปรแกรม Room_3.pas

```
unit Room_3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls;

type
  TRoom3 = class(TForm)
    StatusBar: TStatusBar;
    position1: TShape;
    font1: TLabel;
    position2: TShape;
    font2: TLabel;
    position3: TShape;
    font3: TLabel;
    position4: TShape;
    font4: TLabel;
    Image1: TImage;
    bottom3: TShape;
    bottom4: TShape;
    bottom1: TShape;
    bottom2: TShape;
    Label2: TLabel;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    procedure bottom1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
    procedure bottom2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure bottom4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Room3: TRoom3;

implementation

uses build;

{$R *.DFM}



procedure TRoom3.bottom1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then
```

```
begin
StatusBar.SimpleText := 'Connected';
if bottom1.Tag = 0 then
begin
s:='V81';
Structure.CommPortDriver.SendString(s);
bottom1.Tag := 1;
position1.Brush.Color := clgreen ;
font1.Caption := 'ON ';
end
else
begin
s := 'V80';
Structure.CommPortDriver.SendString(s);
bottom1.tag := 0;
position1.Brush.Color := clred ;
font1.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;

procedure TRoom3.bottom2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
```

```
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom2.Tag = 0 then
begin
s:='V91';
Structure.CommPortDriver.SendString(s);
bottom2.Tag := 1;
position2.Brush.Color := clgreen ;
font2.Caption := 'ON ';
end
else
begin
s := 'V90';
Structure.CommPortDriver.SendString(s);
bottom2.tag := 0;
position2.Brush.Color := clred ;
font2.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;
end;
```

```
procedure TRoom3.bottom3MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom3.Tag = 0 then
begin
s:='Va1';
Structure.CommPortDriver.SendString(s);
bottom3.Tag := 1;
position3.Brush.Color := clgreen ;
font3.Caption := 'ON ';
end
else
begin
s := 'Va0';
Structure.CommPortDriver.SendString(s);
bottom3.tag := 0;
position3.Brush.Color := clred ;
font3.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
```

```
end;


end;


procedure TRoom3.bottom4MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then

begin
StatusBar.SimpleText := 'Connected';
if bottom4.Tag = 0 then
begin
s:='Vb1';
Structure.CommPortDriver.SendString(s);
bottom4.Tag := 1;
position4.Brush.Color := clgreen ;
font4.Caption := 'ON ';
end
else
begin
s := 'Vb0';
Structure.CommPortDriver.SendString(s);
bottom4.tag := 0;
position4.Brush.Color := clred ;
font4.Caption := 'OFF ';
end
end
```

```
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;

end.
```

โปรแกรม Room_4.pas

```
unit Room_4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, StdCtrls;

type
  TRoom4 = class(TForm)
    Image1: TImage;
    position1: TShape;
    font1: TLabel;
    position2: TShape;
    font2: TLabel;
    position3: TShape;
    font3: TLabel;
    position4: TShape;
    font4: TLabel;
    StatusBar: TStatusBar;
    bottom2: TShape;
    bottom4: TShape;
    bottom3: TShape;
    bottom1: TShape;
    Label2: TLabel;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    procedure bottom3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
  procedure bottom1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure bottom2MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure bottom4MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Room4: TRoom4;

implementation

uses build;

{$R *.DFM}



procedure TRoom4.bottom1MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
var s:string;
begin
Structure.ApplyCommSettings;
if Structure.CommPortDriver.Connect then
```

```
begin
StatusBar.SimpleText := 'Connected';
if bottom1.Tag = 0 then
begin
s:='Vc1';
Structure.CommPortDriver.SendString(s);
bottom1.Tag := 1;
position1.Brush.Color := clgreen ;
font1.Caption := 'ON ';
end
else
begin
s := 'Vc0';
Structure.CommPortDriver.SendString(s);
bottom1.tag := 0;
position1.Brush.Color := clred ;
font1.Caption := 'OFF ';
end

end
else
begin
StatusBar.SimpleText :='ERROR';
MessageBeep (0);
end;

end;


procedure TRoom4.bottom2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var s:string;
```

โปรแกรม ComDrv32.pas

```
unit ComDrv32;

interface

uses
  Windows, Messages, SysUtils, Classes, Forms;

type
  // COM Port Baud Rates
  TComPortBaudRate = ( br110, br300, br600, br1200, br2400, br4800,
                       br9600, br14400, br19200, br38400, br56000,
                       br57600, br115200{v1.02: removed ->, br128000, br256000} );
  // COM Port Numbers
  TComPortNumber = ( pnCOM1, pnCOM2, pnCOM3, pnCOM4 );
  // COM Port Data bits
  TComPortDataBits = ( db5BITS, db6BITS, db7BITS, db8BITS );
  // COM Port Stop bits
  TComPortStopBits = ( sb1BITS, sb1HALFBITS, sb2BITS );
  // COM Port Parity
  TComPortParity = ( ptNONE, ptODD, ptEVEN, ptMARK, ptSPACE );
  // COM Port Hardware Handshaking
  TComPortHwHandshaking = ( hhNONE, hhRTSCTS );
  // COM Port Software Handshaing
  TComPortSwHandshaking = ( shNONE, shXONXOFF );

  TComPortReceiveDataEvent = procedure( Sender: TObject; DataPtr: pointer;
DataSize: integer ) of object;



  TCommPortDriver = class(TComponent)
  protected
```

```
  FComPortHandle          : THANDLE; // COM Port Device Handle

  FComPort              : TComPortNumber; // COM Port to use (1..4)
  FComPortBaudRate        : TComPortBaudRate; // COM Port speed (brXXXX)
  FComPortDataBits        : TComPortDataBits; // Data bits size (5..8)
  FComPortStopBits        : TComPortStopBits; // How many stop bits to use
(1,1.5,2)
  FComPortParity          : TComPortParity; // Type of parity to use
(none,odd,even,mark,space)
  FComPortHwHandshaking     : TComPortHwHandshaking; // Type of hw
handshaking to use
  FComPortSwHandshaking     : TComPortSwHandshaking; // Type of sw
handshaking to use
  FComPortInBufSize       : word; // Size of the input buffer
  FComPortOutBufSize      : word; // Size of the output buffer
  FComPortReceiveData     : TComPortReceiveDataEvent; // Event to raise on data
reception
  FComPortPollingDelay    : word; // ms of delay between COM port pollings
  FEnableDTROnOpen        : boolean; { enable/disable DTR line on connect }
  FOutputTimeout          : word; { output timeout - milliseconds }
  FNotifyWnd              : HWND; // This is used for the timer
  FTempInBuffer           : pointer;

  procedure SetComHandle( Value: THANDLE );
  procedure SetComPort( Value: TComPortNumber );
  procedure SetComPortBaudRate( Value: TComPortBaudRate );
  procedure SetComPortDataBits( Value: TComPortDataBits );
  procedure SetComPortStopBits( Value: TComPortStopBits );
  procedure SetComPortParity( Value: TComPortParity );
  procedure SetComPortHwHandshaking( Value: TComPortHwHandshaking );
  procedure SetComPortSwHandshaking( Value: TComPortSwHandshaking );
  procedure SetComPortInBufSize( Value: word );
```

```
  property ComPortSwHandshaking: TComPortSwHandshaking
      read FComPortSwHandshaking write SetComPortSwHandshaking default
shNONE;
  // Input Buffer size
  property ComPortInBufSize: word read FComPortInBufSize write
SetComPortInBufSize default 2048;
  // Output Buffer size
  property ComPortOutBufSize: word read FComPortOutBufSize write
SetComPortOutBufSize default 2048;
  // ms of delay between COM port pollings
  property ComPortPollingDelay: word read FComPortPollingDelay write
SetComPortPollingDelay default 50;
  // v1.02: Set to TRUE to enable DTR line on connect and to leave it on until
disconnect.
  //      Set to FALSE to disable DTR line on connect.
  property EnableDTROnOpen: boolean read FEnableDTROnOpen write
FEnableDTROnOpen default true;
  // v1.02: Output timeout (milliseconds)
  property OutputTimeout: word read FOutputTimeOut write FOutputTimeout
default 4000;
  // Event to raise when there is data available (input buffer has data)
  property OnReceiveData: TComPortReceiveDataEvent read FComPortReceiveData
write FComPortReceiveData;
 end;

procedure Register;

implementation

constructor TCommPortDriver.Create( AOwner: TComponent );
begin
 inherited Create( AOwner );
```

```
  // Initialize to default values
  FComPortHandle          := 0;      // Not connected
  FComPort                := pnCOM2;  // COM 2
  FComPortBaudRate        := br9600;  // 9600 bauds
  FComPortDataBits        := db8BITS; // 8 data bits
  FComPortStopBits        := sb1BITS; // 1 stop bit
  FComPortParity          := ptNONE;  // no parity
  FComPortHwHandshaking   := hhNONE;  // no hardware handshaking
  FComPortSwHandshaking   := shNONE;  // no software handshaking
  FComPortInBufSize       := 2048;    // input buffer of 2048 bytes
  FComPortOutBufSize      := 2048;    // output buffer of 2048 bytes
  FComPortReceiveData     := nil;     // no data handler
  FComPortPollingDelay    := 50;      // poll COM port every 50ms
  FOutputTimeout          := 4000;    // output timeout - 4000ms
  FEnableDTROnOpen        := true;    // DTR high on connect
  // Temporary buffer for received data
  GetMem( FTempInBuffer, FComPortInBufSize );
  // Allocate a window handle to catch timer's notification messages
  if not (csDesigning in ComponentState) then
    FNotifyWnd := AllocateHWnd( TimerWndProc );
end;

destructor TCommPortDriver.Destroy;
begin
  // Be sure to release the COM device
  Disconnect;
  // Free the temporary buffer
  FreeMem( FTempInBuffer, FComPortInBufSize );
  // Destroy the timer's window
  DeallocateHWnd( FNotifyWnd );
  inherited Destroy;
end;
```

```
// v1.02: The COM port handle made public and writeable.
// This lets you connect to external opened com port.
// Setting ComPortHandle to 0 acts as Disconnect.
procedure TCommPortDriver.SetComHandle( Value: THANDLE );
begin
  // If same COM port then do nothing
  if FComPortHandle = Value then
    exit;
  { If value is $FFFFFFFF then stop controlling the COM port
    without closing in }
  if Value = $FFFFFFFF then
  begin
    if Connected then
      { Stop the timer }
      if Connected then
        KillTimer( FNotifyWnd, 1 );
    { No more connected }
    FComPortHandle := 0;
  end
  else
  begin
    { Disconnect }
    Disconnect;
    { If Value is = 0 then exit now }
    { (ComPortHandle := 0 acts as Disconnect) }
    if Value = 0  then
      exit;

    { Set COM port handle }
    FComPortHandle := Value;
```

```
  { Start the timer (used for polling) }
  SetTimer( FNotifyWnd, 1, FComPortPollingDelay, nil );
 end;
end;

procedure TCommPortDriver.SetComPort( Value: TComPortNumber );
begin
 // Be sure we are not using any COM port
 if Connected then
  exit;
 // Change COM port
 FComPort := Value;
end;

procedure TCommPortDriver.SetComPortBaudRate( Value: TComPortBaudRate );
begin
 // Set new COM speed
 FComPortBaudRate := Value;
 // Apply changes
 if Connected then
  ApplyCOMSettings;
end;

procedure TCommPortDriver.SetComPortDataBits( Value: TComPortDataBits );
begin
 // Set new data bits
 FComPortDataBits := Value;
 // Apply changes
 if Connected then
  ApplyCOMSettings;
 end;
```

```
procedure TCommPortDriver.SetComPortStopBits( Value: TComPortStopBits );
begin
 // Set new stop bits
 FComPortStopBits := Value;
 // Apply changes
 if Connected then
   ApplyCOMSettings;
end;

procedure TCommPortDriver.SetComPortParity( Value: TComPortParity );
begin
 // Set new parity
 FComPortParity := Value;
 // Apply changes
 if Connected then
   ApplyCOMSettings;
end;

procedure TCommPortDriver.SetComPortHwHandshaking( Value:
TComPortHwHandshaking );
begin
 // Set new hardware handshaking
 FComPortHwHandshaking := Value;
 // Apply changes
 if Connected then
   ApplyCOMSettings;
end;

procedure TCommPortDriver.SetComPortSwHandshaking( Value:
TComPortSwHandshaking );
begin
 // Set new software handshaking
```

```
 FComPortSwHandshaking := Value;
 // Apply changes
 if Connected then
   ApplyCOMSettings;
end;

procedure TCommPortDriver.SetComPortInBufSize( Value: word );
begin
 { Do nothing if connected }
 if Connected then
   exit;
 // Free the temporary input buffer
 FreeMem( FTempInBuffer, FComPortInBufSize );
 // Set new input buffer size
 FComPortInBufSize := Value;
 // Allocate the temporary input buffer
 GetMem( FTempInBuffer, FComPortInBufSize );
end;

procedure TCommPortDriver.SetComPortOutBufSize( Value: word );
begin
 { Do nothing if connected }
 if not Connected then
   exit;
 // Set new output buffer size
 FComPortOutBufSize := Value;
end;

procedure TCommPortDriver.SetComPortPollingDelay( Value: word );
begin
 // If new delay is not equal to previous value...
 if Value <> FComPortPollingDelay then
```

```
  begin
    // Stop the timer
    if Connected then
      KillTimer( FNotifyWnd, 1 );
    // Store new delay value
    FComPortPollingDelay := Value;
    // Restart the timer
    if Connected then
      SetTimer( FNotifyWnd, 1, FComPortPollingDelay, nil );
  end;
end;


const
  Win32BaudRates: array[br110..br115200] of DWORD =
    ( CBR_110, CBR_300, CBR_600, CBR_1200, CBR_2400, CBR_4800,
CBR_9600,
      CBR_14400, CBR_19200, CBR_38400, CBR_56000, CBR_57600, CBR_115200
{v1.02 removed: CRB_128000, CBR_256000} );


const
  dcb_Binary            = $00000001;
  dcb_ParityCheck       = $00000002;
  dcb_OutxCtsFlow       = $00000004;
  dcb_OutxDsrFlow       = $00000008;
  dcb_DtrControlMask    = $00000030;
    dcb_DtrControlDisable  = $00000000;
    dcb_DtrControlEnable   = $00000010;
    dcb_DtrControlHandshake = $00000020;
  dcb_DsrSensivity      = $00000040;
  dcb_TXContinueOnXoff   = $00000080;
  dcb_OutX              = $00000100;
  dcb_InX               = $00000200;
```

```
  dcb_ErrorChar          = $00000400;
  dcb_NullStrip          = $00000800;
  dcb_RtsControlMask     = $00003000;
   dcb_RtsControlDisable  = $00000000;
   dcb_RtsControlEnable   = $00001000;
   dcb_RtsControlHandshake = $00002000;
   dcb_RtsControlToggle   = $00003000;
  dcb_AbortOnError       = $00004000;
  dcb_Reserveds          = $FFFF8000;

// Apply COM settings.
procedure TCommPortDriver.ApplyCOMSettings;
var dcb: TDCB;
begin
  // Do nothing if not connected
  if not Connected then
    exit;

  // Clear all
  fillchar( dcb, sizeof(dcb), 0 );
  // Setup dcb (Device Control Block) fields
  dcb.DCBLength := sizeof(dcb); // dcb structure size
  dcb.BaudRate := Win32BaudRates[ FComPortBaudRate ]; // baud rate to use
  // Set fBinary: Win32 does not support non binary mode transfers
  // (also disable EOF check)
  dcb.Flags := dcb_Binary;
  if EnableDTROnOpen then
    { Enabled the DTR line when the device is opened and leaves it on }
    dcb.Flags := dcb.Flags or dcb_DtrControlEnable;

  case FComPortHwHandshaking of // Type of hw handshaking to use
    hhNONE:; // No hardware handshaking
```

```
  hhRTSCTS: // RTS/CTS (request-to-send/clear-to-send) hardware handshaking
    dcb.Flags := dcb.Flags or dcb_OutxCtsFlow or dcb_RtsControlHandshake;
 end;
 case FComPortSwHandshaking of // Type of sw handshaking to use
  shNONE:; // No software handshaking
  shXONXOFF: // XON/XOFF handshaking
    dcb.Flags := dcb.Flags or dcb_OutX or dcb_InX;
 end;
 dcb.XONLim := FComPortInBufSize div 4; // Specifies the minimum number of
bytes allowed
                          // in the input buffer before the XON character is sent
                          // (or CTS is set)
 dcb.XOFFLim := 1; // Specifies the maximum number of bytes allowed in the input
buffer
            // before the XOFF character is sent. The maximum number of bytes
            // allowed is calculated by subtracting this value from the size,
            // in bytes, of the input buffer
 dcb.ByteSize := 5 + ord(FComPortDataBits); // how many data bits to use
 dcb.Parity := ord(FComPortParity); // type of parity to use
 dcb.StopBits := ord(FComPortStopbits); // how many stop bits to use
 dcb.XONChar := #17; // XON ASCII char
 dcb.XOFFChar := #19; // XOFF ASCII char
 SetCommState( FComPortHandle, dcb );
 { Flush buffers }
 FlushBuffers( true, true );
 // Setup buffers size
 SetupComm( FComPortHandle, FComPortInBufSize, FComPortOutBufSize );
end;

 function TCommPortDriver.Connect: boolean;
 var comName: array[0..4] of char;
   tms: TCOMMTIMEOUTS;
```

```
begin
 // Do nothing if already connected
 Result := Connected;
 if Result then
  exit;
 // Open the COM port
 StrPCopy( comName, 'COM' );
 comName[3] := chr( ord('1') + ord(FComPort) );
 comName[4] := #0;
 FComPortHandle := CreateFile(
                    comName,
                    GENERIC_READ or GENERIC_WRITE,
                    0, // Not shared
                    nil, // No security attributes
                    OPEN_EXISTING,
                    FILE_ATTRIBUTE_NORMAL,
                    0 // No template
                   );
 Result := Connected;
 if not Result then
  exit;
 // Apply settings
 ApplyCOMSettings;
 // Setup timeouts: we disable timeouts because we are polling the com port!
 tms.ReadIntervalTimeout := 1; // Specifies the maximum time, in milliseconds,
                    // allowed to elapse between the arrival of two
                    // characters on the communications line
 tms.ReadTotalTimeoutMultiplier := 0; // Specifies the multiplier, in milliseconds,
                    // used to calculate the total time-out period
                    // for read operations.
 tms.ReadTotalTimeoutConstant := 1; // Specifies the constant, in milliseconds,
                    // used to calculate the total time-out period
```

```
                        // for read operations.
  tms.WriteTotalTimeoutMultiplier := 0; // Specifies the multiplier, in milliseconds,
                        // used to calculate the total time-out period
                        // for write operations.
  tms.WriteTotalTimeoutConstant := 0; // Specifies the constant, in milliseconds,
                        // used to calculate the total time-out period
                        // for write operations.
  SetCommTimeOuts( FComPortHandle, tms );
  // Start the timer (used for polling)
  SetTimer( FNotifyWnd, 1, FComPortPollingDelay, nil );
end;

procedure TCommPortDriver.Disconnect;
begin
  if Connected then
  begin
    // Stop the timer (used for polling)
    KillTimer( FNotifyWnd, 1 );
    // Release the COM port
    CloseHandle( FComPortHandle );
    // No more connected
    FComPortHandle := 0;
  end;
end;

function TCommPortDriver.Connected: boolean;
begin
  Result := FComPortHandle > 0;
end;

// v1.02: flish rx/rx buffers
procedure TCommPortDriver.FlushBuffers( inBuf, outBuf: boolean );
```

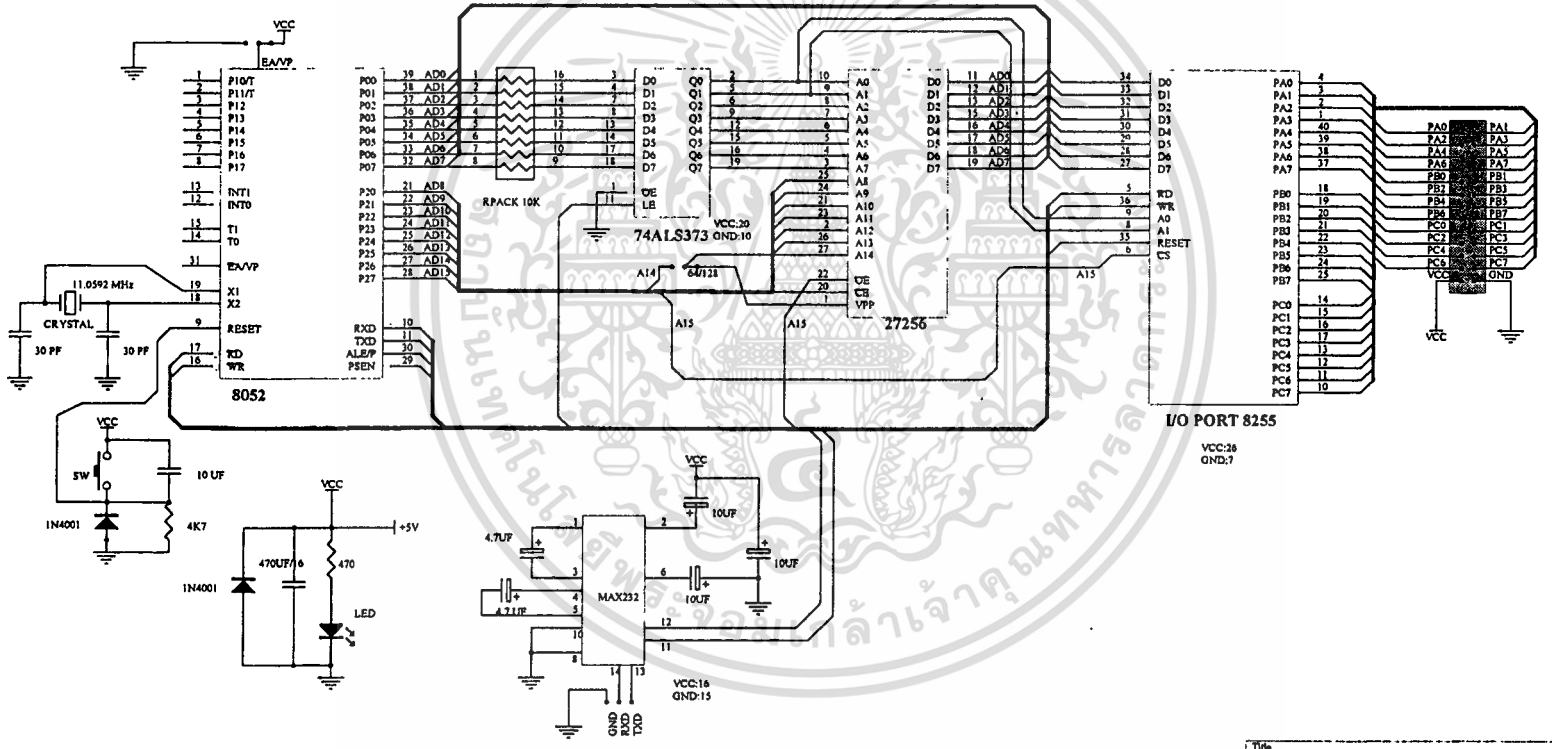
```
begin
  Result := WriteFile( FComPortHandle, DataPtr^, DataSize, nsent, nil );
  Result := Result and (nsent=DataSize);
end;}

{ Send data (breaks the data in small packets if it doesn't fit in the output
  buffer) }
function TCommPortDriver.SendData( DataPtr: pointer; DataSize: integer ): integer;
var nToSend, nsent: integer;
    t1: longint;
begin
  { 0 bytes sent }
  Result := 0;
  { Do nothing if not connected }
  if not Connected then
    exit;
  { Current time }
  t1 := GetTickCount;
  { Loop until all data sent or timeout occurred }
  while DataSize > 0 do
  begin
    { Get output buffer free space }
    nToSend := OutFreeSpace;
    { If output buffer has some free space... }
    if nToSend > 0 then
    begin
      { Don't send more bytes than we actually have to send }
      if nToSend > DataSize then
        nToSend := DataSize;
      { Send }
      WriteFile( FComPortHandle, DataPtr^, DataSize, nsent, nil );
      { Update number of bytes sent }
```

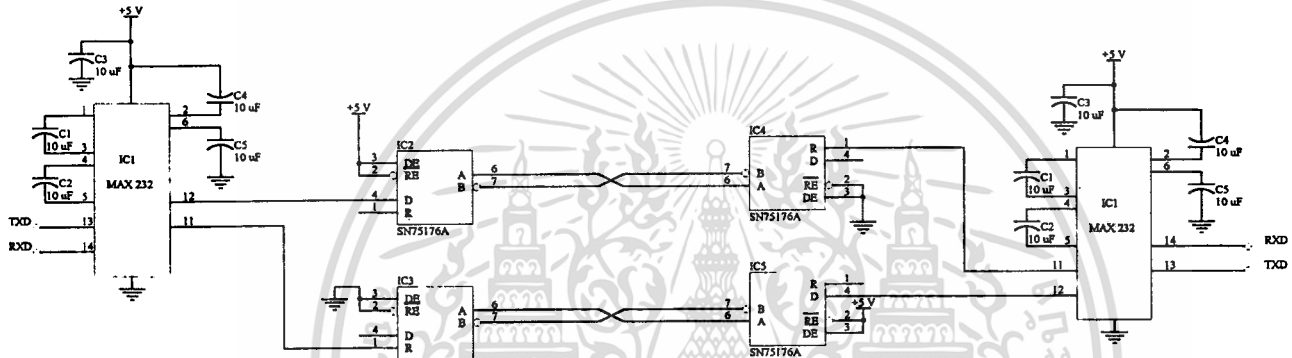
```
    Result := Result + abs(nsent);
    { Decrease the count of bytes to send }
    DataSize := DataSize - abs(nsent);
    { Get current time }
    t1 := GetTickCount;
    { Continue. This skips the time check below (don't stop
      trasmitting if the FOutputTimeout is set too low) }
    continue;
   end;
   { Buffer is full. If we are waiting too long then
     invert the number of bytes sent and exit }
   if (GetTickCount-t1) > FOutputTimeout then
   begin
    Result := -Result;
    exit;
   end;
  end;
end;

// Send a pascal string (NULL terminated if $H+ (default))
function TCommPortDriver.SendString( s: string ): boolean;
var len: integer;
begin
  len := length( s );
  {$IFOPT H+}
  // New syle pascal string (NULL terminated)
  Result := SendData( pchar(s), len ) = len;
  {$ELSE}
  // Old style pascal string (s[0] = length)
  Result := SendData( pchar(@s[1]), len ) = len;
  {$ENDIF}
end;
```

```
// v1.02: send a C-style strings (NULL terminated)
function TCommPortDriver.SendZString( s: pchar ): boolean;
var len: integer;
begin
  len := strlen( s );
  Result := SendData( s, len ) = len;
end;

// v1.02: set DTR line high (onOff=TRUE) or low (onOff=FALSE).
//       You must not use HW handshaking.
procedure TCommPortDriver.ToggleDTR( onOff: boolean );
const funcs: array[boolean] of integer = (CLRDTR,SETDTR);
begin
  if Connected then
    EscapeCommFunction( FComPortHandle, funcs[onOff] );
end;

// v1.02: set RTS line high (onOff=TRUE) or low (onOff=FALSE).
//       You must not use HW handshaking.
procedure TCommPortDriver.ToggleRTS( onOff: boolean );
const funcs: array[boolean] of integer = (CLRRTS,SETRTS);
begin
  if Connected then
    EscapeCommFunction( FComPortHandle, funcs[onOff] );
end;

// COM port polling proc
procedure TCommPortDriver.TimerWndProc( var msg: TMessage );
var nRead: dword;
begin
  if (msg.Msg = WM_TIMER) and Connected then
```

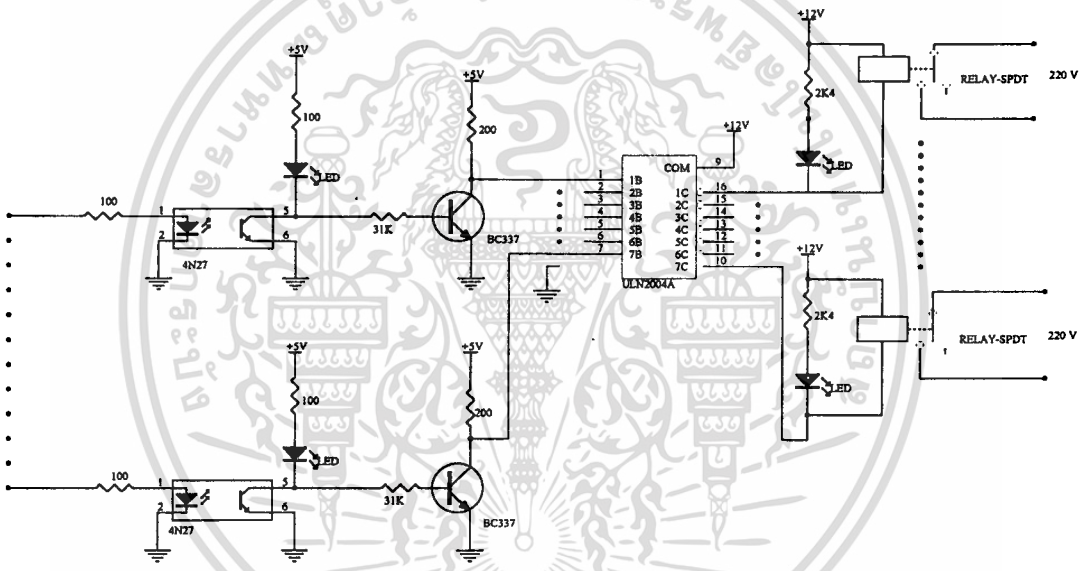
```
 begin
   nRead := 0;
   if ReadFile( FComPortHandle, FTempInBuffer^, FComPortInBufSize, nRead, nil )
then
     if (nRead <> 0) and Assigned(FComPortReceiveData) then
       FComPortReceiveData( Self, FTempInBuffer, nRead );
  end;
end;

procedure Register;
begin
  { Register this component and show it in the 'System' tab
    of the component palette }
  RegisterComponents('System', [TCommPortDriver]);
end;

 end.
```

ภาคผนวก ข


8052
74ALS373
27256
I/O PORT 8255
MAX232
RPACK 10K
CRYSTAL
11.0592 MHz
30 PF
30 PF
5W
10 UF
1N4001
4K7
470UF/16
470
LED
+5V
4.7UF
10UF
VCC:20
GND:10
VCC:26
GND:7
VCC:16
GND:15
CONTROL BOARD
Size
B
Number
Revision
Date: 23-Apr-1999
File: A:\SCH_1.SCH
Sheet of
Drawn By:



Title	Convert 232 to 422	
Size: B	Number	Revision
Date:	12-May-1999	Sheet of
File:	D:\MMM422.SCH	Drawn By:



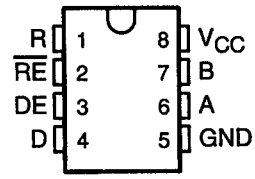
Title	Control Board		
Size: B	Number		Revision
Date: 12-May-1999		Sheet of	
File: C:\MYDOCU~1\SCH_2.SCH		Drawn By:	

ภาคผนวก ค

SLLS101A – JULY 1985 – REVISED MAY 1995

- Bidirectional Transceivers
- Meet or Exceed the Requirements of ANSI Standards EIA/TIA-422-B and RS-485 and ITU Recommendations V.11 and X.27
- Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments
- 3-State Driver and Receiver Outputs
- Individual Driver and Receiver Enables
- Wide Positive and Negative Input/Output Bus Voltage Ranges
- Driver Output Capability . . . ±60 mA Max
- Thermal Shutdown Protection
- Driver Positive and Negative Current Limiting
- Receiver Input Impedance . . . 12 kΩ Min
- Receiver Input Sensitivity . . . ±200 mV
- Receiver Input Hysteresis . . . 50 mV Typ
- Operate From Single 5-V Supply

D OR P PACKAGE
(TOP VIEW)



description

The SN65176B and SN75176B differential bus transceivers are monolithic integrated circuits designed for bidirectional data communication on multipoint bus transmission lines. They are designed for balanced transmission lines and meet ANSI Standards EIA/TIA-422-B and RS-485 and ITU Recommendations V.11 and X.27.

The SN65176B and SN75176B combine a 3-state differential line driver and a differential input line receiver, both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be externally connected together to function as a direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus whenever the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges making the device suitable for party-line applications.

The driver is designed for up to 60 mA of sink or source current. The driver features positive- and negative-current limiting and thermal shutdown for protection from line-fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The SN65176B and SN75176B can be used in transmission line applications employing the SN75172 and SN75174 quadruple differential line drivers and SN75173 and SN75175 quadruple differential line receivers.

The SN65176B is characterized for operation from –40°C to 105°C and the SN75176B is characterized for operation from 0°C to 70°C.

Function Tables

DRIVER

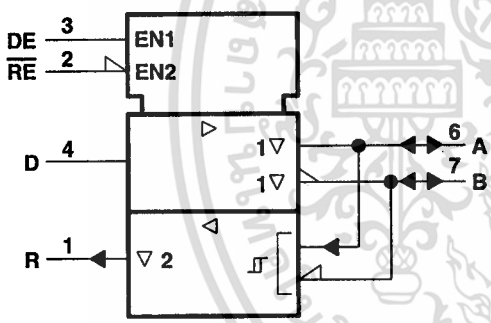
INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A – B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	H

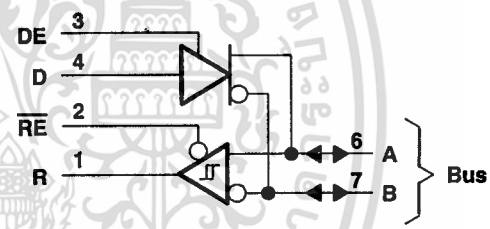
H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

logic symbol†

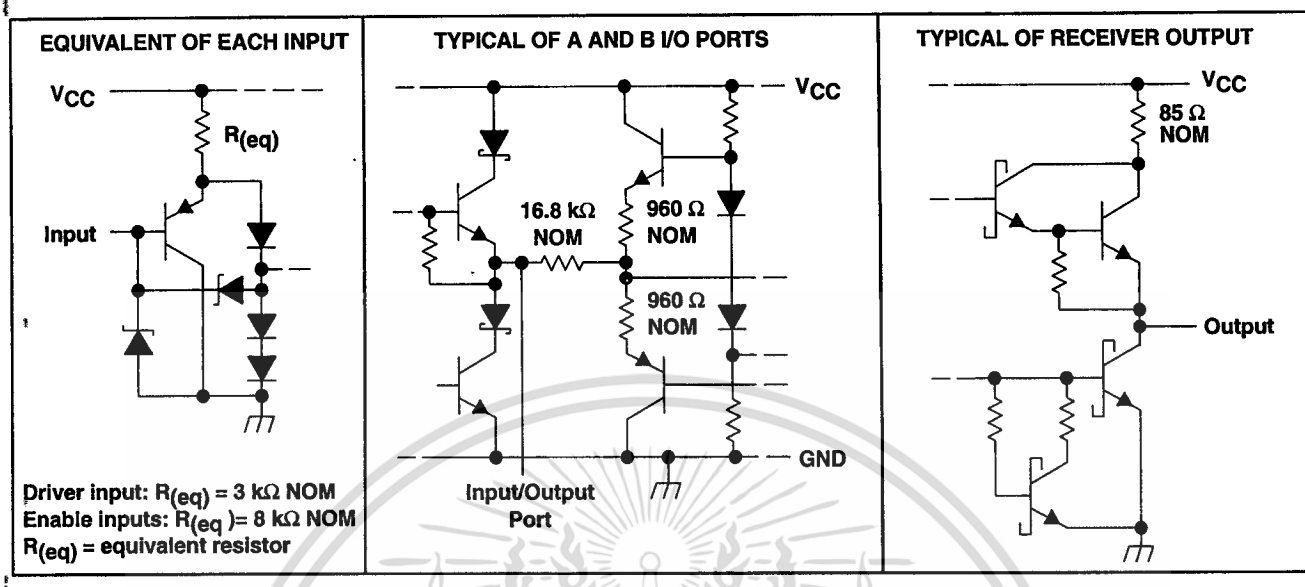


† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



schematics of inputs and outputs



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) ..... 7 V
Voltage range at any bus terminal ..... −10 V to 15 V
Enable input voltage, V_I ..... 5.5 V
Continuous total power dissipation ..... See Dissipation Rating Table
Operating free-air temperature range, T_A: SN65176B ..... −40°C to 105°C
SN75176B ..... 0°C to 70°C
Storage temperature range, T_{stg} ..... −65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C

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

NOTE 1: All voltage values, except differential input/output bus voltage, are with respect to network ground terminal.

DISSIPATION RATING TABLE

PACKAGE	$T_A \le 25°C$ POWER RATING	DERATING FACTOR ABOVE $T_A = 25°C$	$T_A = 70°C$ POWER RATING	$T_A = 105°C$ POWER RATING
D	725 mW	5.8 mW/°C	464 mW	261 mW
P	1100 mW	8.8 mW/°C	704 mW	396 mW

recommended operating conditions

		MIN	TYP	MAX	UNIT
Supply voltage, V_{CC}		4.75	5	5.25	V
Voltage at any bus terminal (separately or common mode), V_I or V_{IC}				12	V
				−7	
High-level input voltage, V_{IH}	D, DE, and $\overline{RE}$	2			V
Low-level input voltage, V_{IL}	D, DE, and $\overline{RE}$			0.8	V
Differential input voltage, V_{ID} (see Note 2)				±12	V
High-level output current, I_{OH}	Driver			−60	mA
	Receiver			−400	µA
Low-level output current, I_{OL}	Driver			60	mA
	Receiver			8	
Operating free-air temperature, T_A	SN65176B	−40		105	°C
	SN75176B	0		70	

NOTE 2: Differential-input/output bus voltage is measured at the noninverting terminal A with respect to the inverting terminal B.

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IK}	Input clamp voltage	$I_I = -18$ mA				−1.5	V
V_O	Output voltage	$I_O = 0$		0		6	V
$\lvert V_{OD1}\rvert$	Differential output voltage	$I_O = 0$		1.5	3.6	6	V
$\lvert V_{OD2}\rvert$	Differential output voltage	R_L = 100 Ω,	See Figure 1	1/2 V_{OD1} or 2¶			V
		R_L = 54 Ω,	See Figure 1	1.5	2.5	5	V
V_{OD3}	Differential output voltage	See Note 4		1.5		5	V
$\Delta\lvert V_{OD}\rvert$	Change in magnitude of differential output voltage§	R_L = 54 Ω or 100 Ω,	See Figure 1			±0.2	V
V_{OC}	Common-mode output voltage					+3 −1	V
$\Delta\lvert V_{OC}\rvert$	Change in magnitude of common-mode output voltage§					±0.2	V
I_O	Output current	Output disabled, See Note 3	V_O = 12 V			1	mA
			V_O = −7 V			−0.8	
I_{IH}	High-level input current	V_I = 2.4 V				20	µA
I_{IL}	Low-level input current	V_I = 0.4 V				−400	µA
I_{OS}	Short-circuit output current	V_O = −7 V				−250	mA
		V_O = 0				150	
		V_O = V_{CC}				250	
		V_O = 12 V				250	
I_{CC}	Supply current (total package)	No load	Outputs enabled		42	70	mA
			Outputs disabled		26	35	

† The power-off measurement in ANSI Standard EIA/TIA-422-B applies to disabled outputs only and is not applied to combined inputs and outputs.
‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.
§ $\Delta\lvert V_{OD}\rvert$ and $\Delta\lvert V_{OC}\rvert$ are the changes in magnitude of V_{OD} and V_{OC}, respectively, that occur when the input is changed from a high level to a low level.
¶ The minimum V_{OD2} with a 100-Ω load is either 1/2 V_{OD1} or 2 V, whichever is greater.
NOTES: 3. See ANSI Standard RS-485 Figure 3.5, Test Termination Measurement 2.
4. This applies for both power on and off; refer to ANSI Standard RS-485 for exact conditions. The EIA/TIA-422-B limit does not apply for a combined driver and receiver terminal.

switching characteristics, V_{CC} = 5 V, R_L = 110 kΩ, T_A = 25°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
$t_{d(OD)}$	Differential-output delay time	R_L = 54 Ω,	See Figure 3		15	22	ns
$t_{t(OD)}$	Differential-output transition time				20	30	ns
t_{PZH}	Output enable time to high level	See Figure 4			85	120	ns
t_{PZL}	Output enable time to low level	See Figure 5			40	60	ns
t_{PHZ}	Output disable time from high level	See Figure 4			150	250	ns
t_{PLZ}	Output disable time from low level	See Figure 5			20	30	ns

SYMBOL EQUIVALENTS

DATA-SHEET PARAMETER	EIA/TIA-422-B	RS-485
V_O	V_{oa}, V_{ob}	V_{oa}, V_{ob}
$\lvert V_{OD1}\rvert$	V_o	V_o
$\lvert V_{OD2}\rvert$	V_t (R_L = 100 Ω)	V_t (R_L = 54 Ω)
$\lvert V_{OD3}\rvert$		V_t (Test Termination Measurement 2)
$\Delta\lvert V_{OD}\rvert$	$\lvert\, \lvert V_t\rvert - \lvert\overline{V}_t\rvert \,\rvert$	$\lvert\, \lvert V_t - \overline{V}_t\rvert \,\rvert$
V_{OC}	$\lvert V_{os}\rvert$	$\lvert V_{os}\rvert$
$\Delta\lvert V_{OC}\rvert$	$\lvert V_{os} - \overline{V}_{os}\rvert$	$\lvert V_{os} - \overline{V}_{os}\rvert$
I_{OS}	$\lvert I_{sa}\rvert$, $\lvert I_{sb}\rvert$	
I_O	$\lvert I_{xa}\rvert$, $\lvert I_{xb}\rvert$	I_{ia}, I_{ib}

RECEIVER SECTION

electrical characteristics over recommended ranges of common-mode input voltage, supply voltage, and operating free-air temperature (unless otherwise noted)

	PARAMETER	TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{IT+}	Positive-going input threshold voltage	V_O = 2.7 V,	I_O = −0.4 mA			0.2	V
V_{IT-}	Negative-going input threshold voltage	V_O = 0.5 V,	I_O = 8 mA	−0.2‡			V
V_{hys}	Input hysteresis voltage ($V_{IT+} - V_{IT-}$)				50		mV
V_{IK}	Enable Input clamp voltage	I_I = −18 mA				−1.5	V
V_{OH}	High-level output voltage	V_{ID} = 200 mV, See Figure 2	I_{OH} = −400 µA,	2.7			V
V_{OL}	Low-level output voltage	V_{ID} = −200 mV, See Figure 2	I_{OL} = 8 mA,			0.45	V
I_{OZ}	High-impedance-state output current	V_O = 0.4 V to 2.4 V				±20	µA
I_I	Line input current	Other input = 0 V, See Note 5	V_I = 12 V			1	mA
			V_I = −7 V			−0.8	
I_{IH}	High-level enable input current	V_{IH} = 2.7 V				20	µA
I_{IL}	Low-level enable input current	V_{IL} = 0.4 V				−100	µA
r_I	Input resistance	V_I = 12 V		12			kΩ
I_{OS}	Short-circuit output current			−15		−85	mA
I_{CC}	Supply current (total package)	No load	Outputs enabled		42	55	mA
			Outputs disabled		26	35	

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the less positive (more negative) limit is designated minimum, is used in this data sheet for common-mode input voltage and threshold voltage levels only.
NOTE 5: This applies for both power on and power off. Refer to EIA Standard RS-485 for exact conditions.

switching characteristics, V_{CC} = 5 V, C_L = 15 pF, T_A = 25°C

	PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low- to high-level output	V_{ID} = 0 to 3 V, See Figure 6		21	35	ns
t_{PHL}	Propagation delay time, high- to low-level output			23	35	ns
t_{PZH}	Output enable time to high level	See Figure 7		10	20	ns
t_{PZL}	Output enable time to low level			12	20	ns
t_{PHZ}	Output disable time from high level	See Figure 7		20	35	ns
t_{PLZ}	Output disable time from low level			17	25	ns

PARAMETER MEASUREMENT INFORMATION

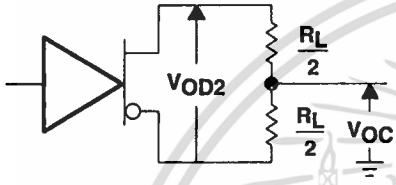


Figure 1. Driver V_{OD} and V_{OC}

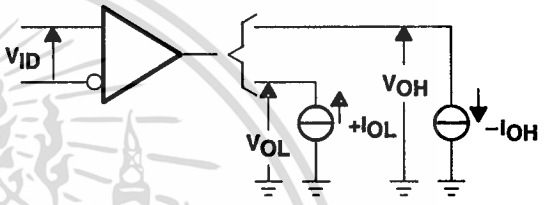
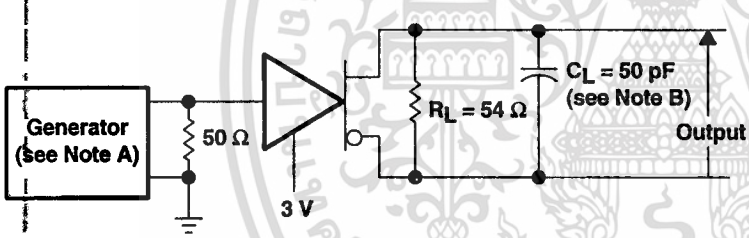
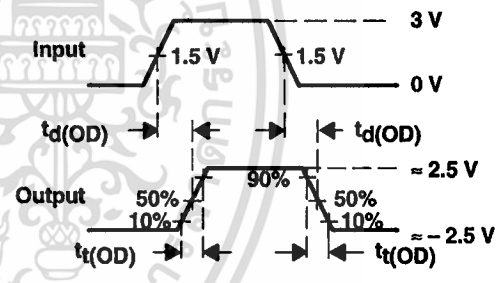


Figure 2. Receiver V_{OH} and V_{OL}



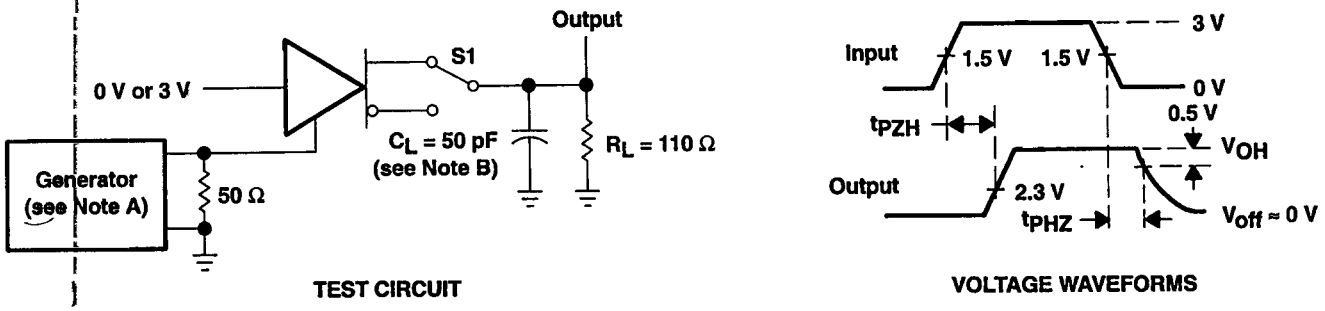
TEST CIRCUIT



VOLTAGE WAVEFORMS

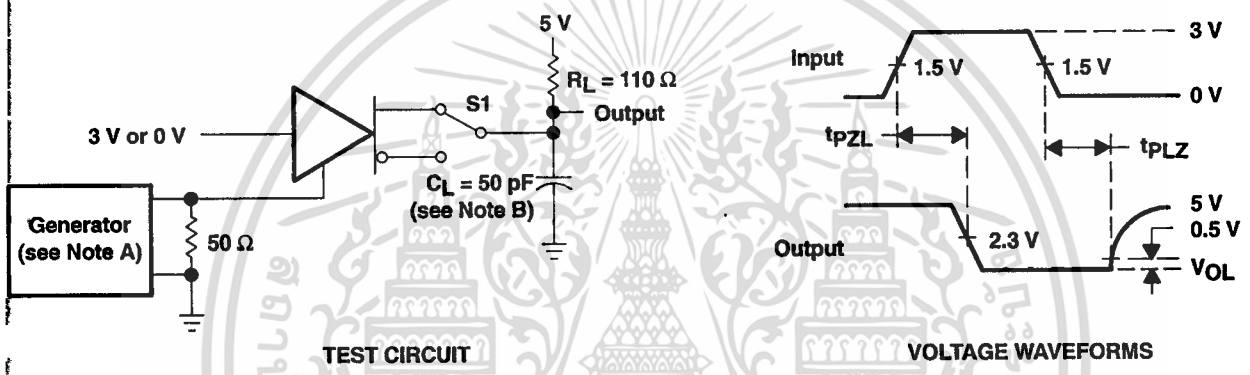
NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR ≤ 1 MHz, 50% duty cycle, t_r ≤ 6 ns, t_f ≤ 6 ns, Z_O = 50 Ω.
B. C_L includes probe and jig capacitance.

Figure 3. Driver Test Circuit and Voltage Waveforms



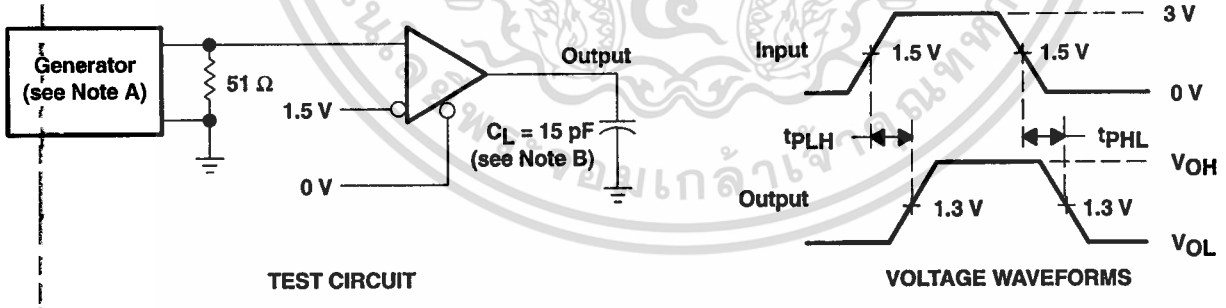
NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR ≤ 1 MHz, 50% duty cycle, t_r ≤ 6 ns, t_f ≤ 6 ns, Z_O = 50 Ω.
B. C_L includes probe and jig capacitance.

Figure 4. Driver Test Circuit and Voltage Waveforms



NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR ≤ 1 MHz, 50% duty cycle, t_r ≤ 6 ns, t_f ≤ 6 ns, Z_O = 50 Ω.
B. C_L includes probe and jig capacitance.

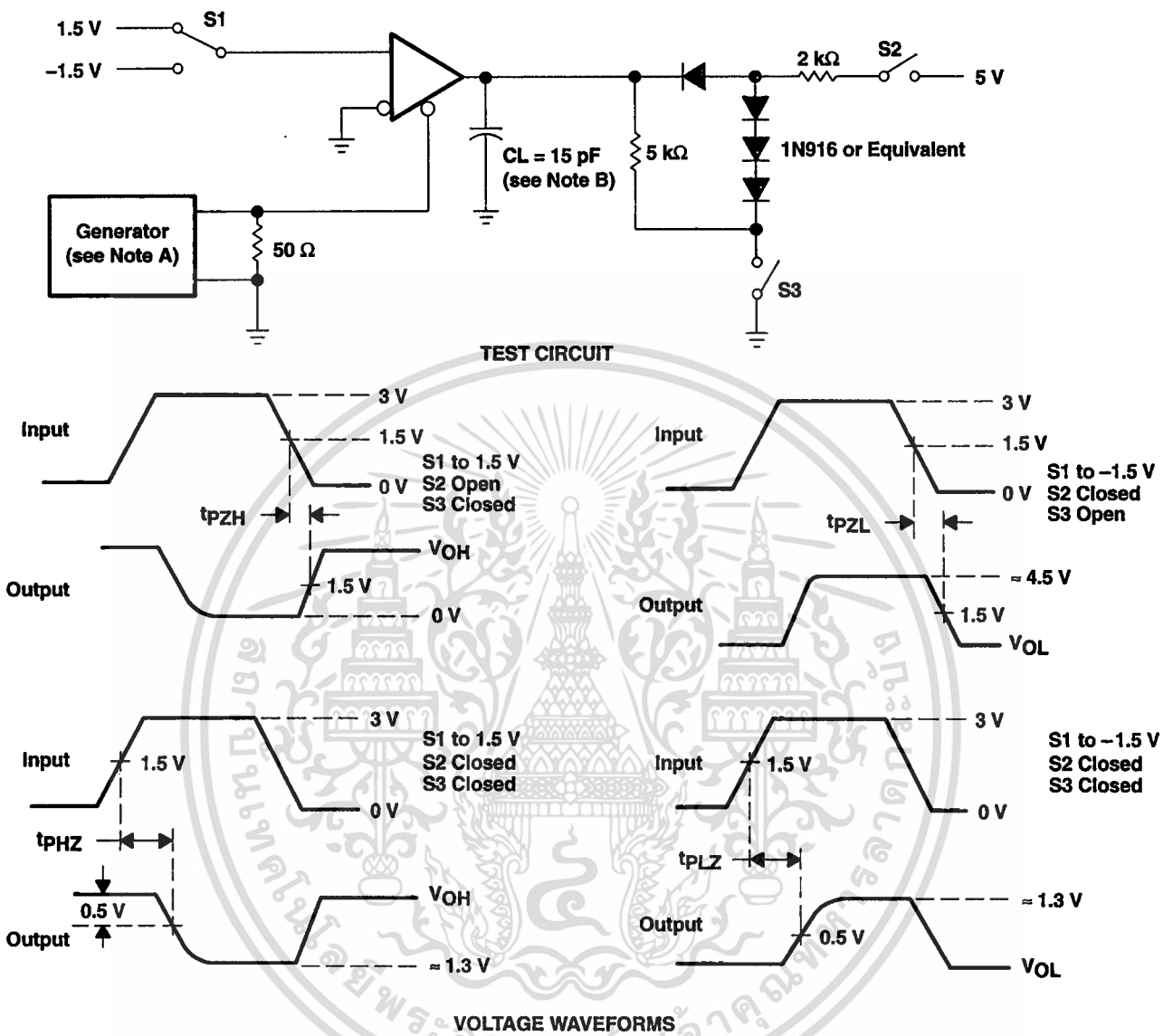
Figure 5. Driver Test Circuit and Voltage Waveforms



NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR ≤ 1 MHz, 50% duty cycle, t_r ≤ 6 ns, t_f ≤ 6 ns, Z_O = 50 Ω.
B. C_L includes probe and jig capacitance.

Figure 6. Receiver Test Circuit and Voltage Waveforms

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The input pulse is supplied by a generator having the following characteristics: PRR ≤ 1 MHz, 50% duty cycle, t_r ≤ 6 ns, t_f ≤ 6 ns, Z_O = 50 Ω.
B. C_L includes probe and jig capacitance.

Figure 7. Receiver Test Circuit and Voltage Waveforms

TYPICAL CHARACTERISTICS

DRIVER
HIGH-LEVEL OUTPUT VOLTAGE
vs
HIGH-LEVEL OUTPUT CURRENT

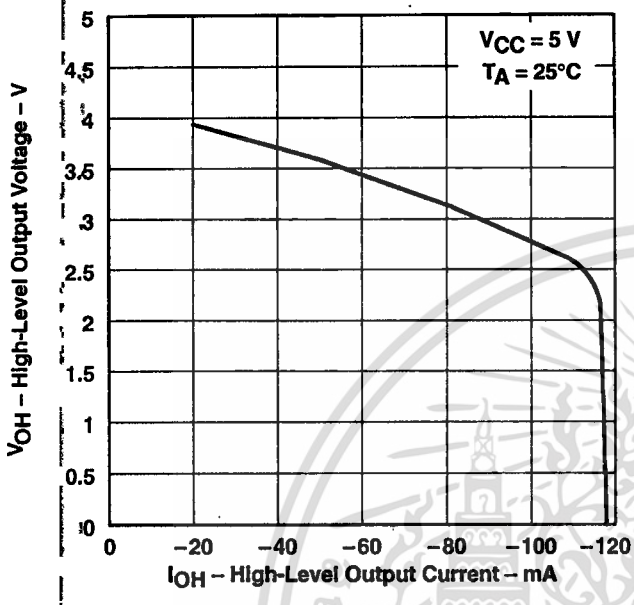


Figure 8

DRIVER
LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

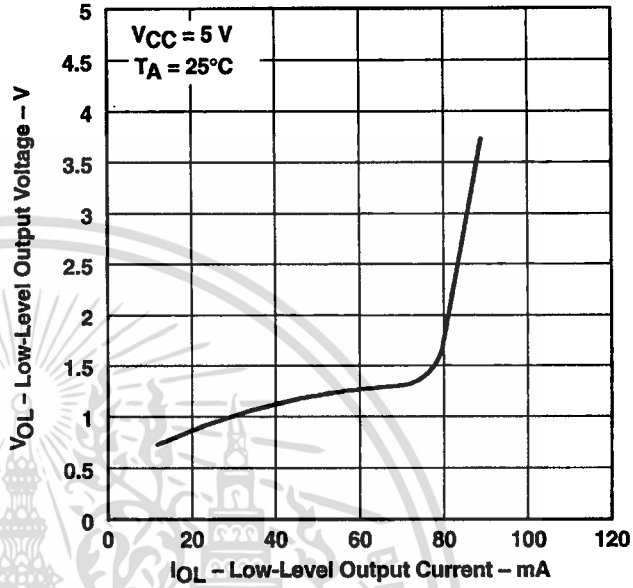


Figure 9

DRIVER
DIFFERENTIAL OUTPUT VOLTAGE
vs
OUTPUT CURRENT

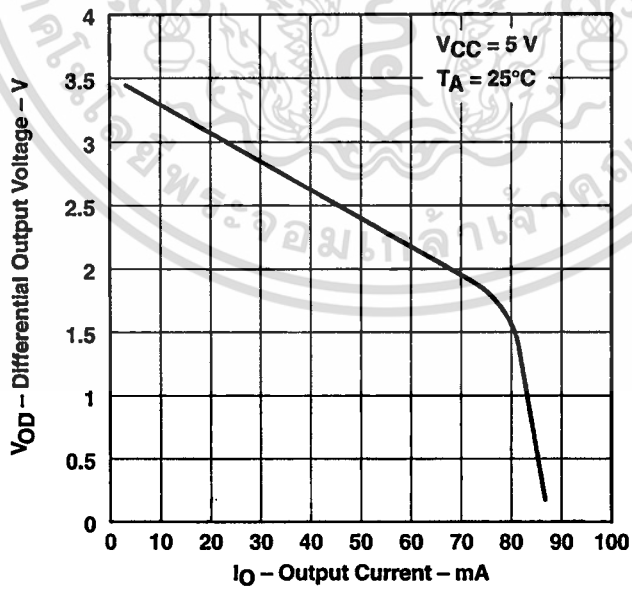


Figure 10

TYPICAL CHARACTERISTICS

RECEIVER
HIGH-LEVEL OUTPUT VOLTAGE
vs
HIGH-LEVEL OUTPUT CURRENT

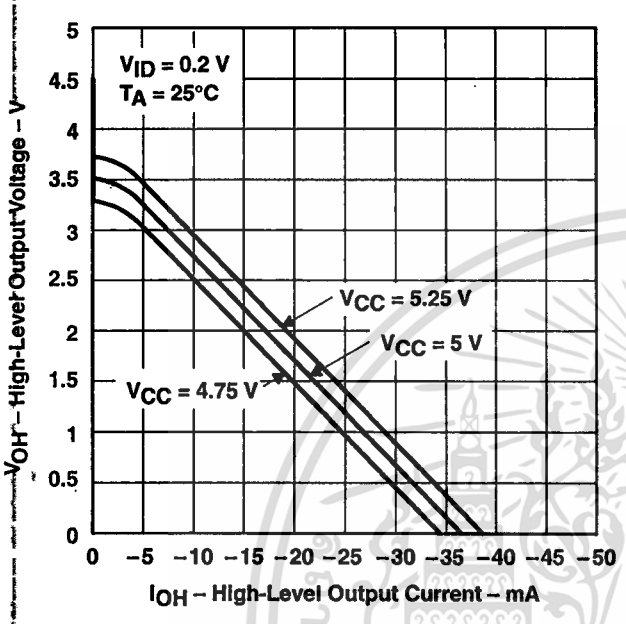
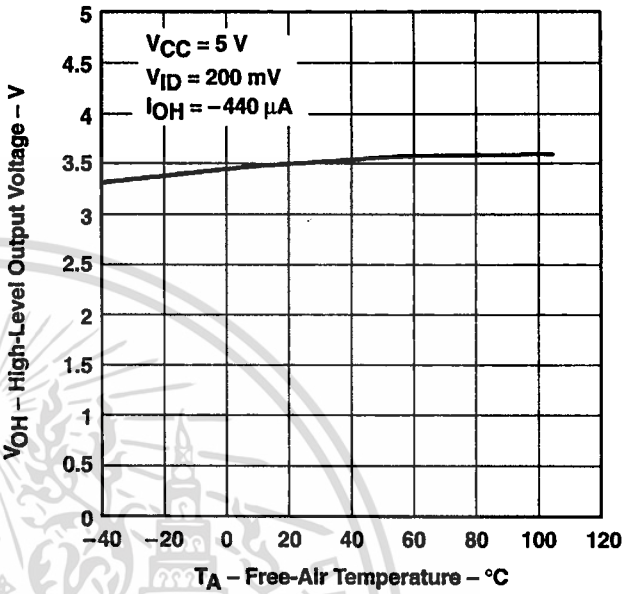


Figure 11

RECEIVER
HIGH-LEVEL OUTPUT VOLTAGE
vs
FREE-AIR TEMPERATURE†



† Only the 0°C to 70°C portion of the curve applies to the SN75176B.

Figure 12

RECEIVER
LOW-LEVEL OUTPUT VOLTAGE
vs
LOW-LEVEL OUTPUT CURRENT

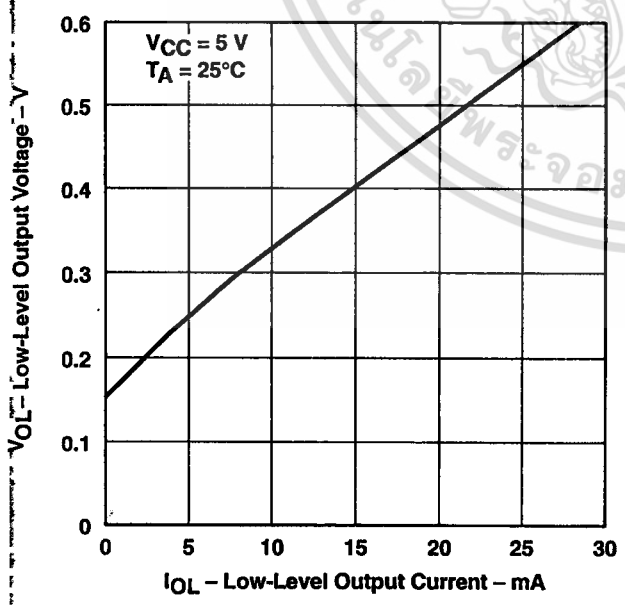


Figure 13

RECEIVER
LOW-LEVEL OUTPUT VOLTAGE
vs
FREE-AIR TEMPERATURE

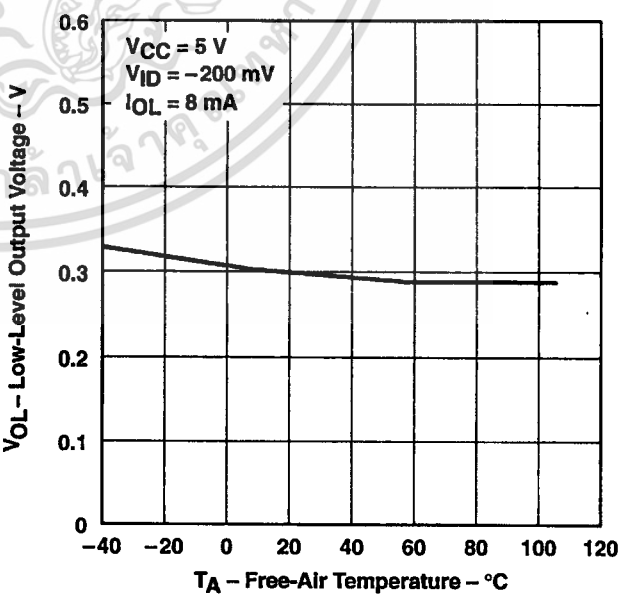


Figure 14

TYPICAL CHARACTERISTICS

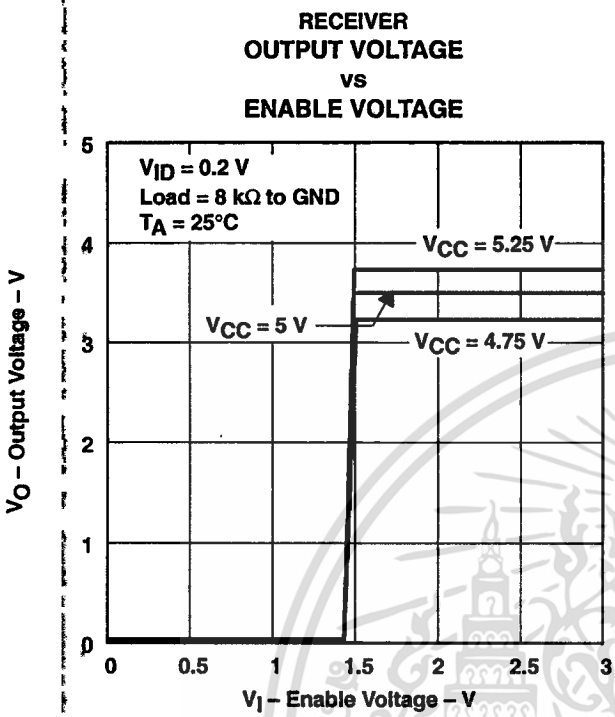


Figure 15

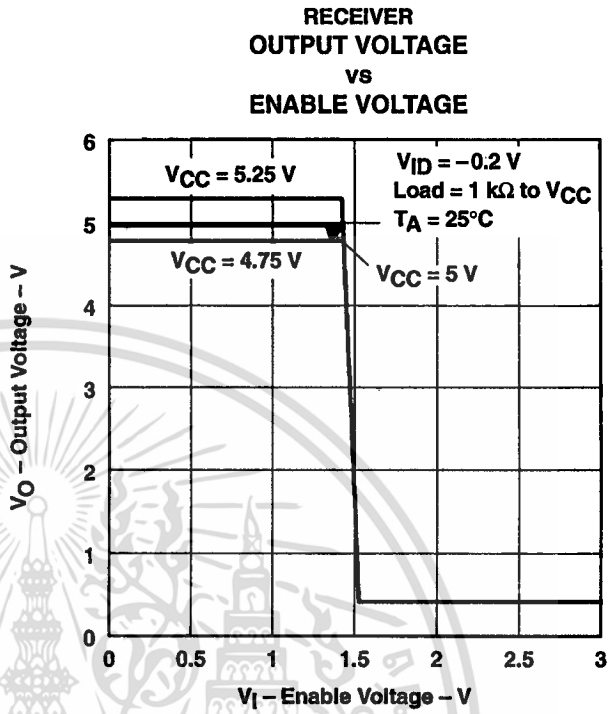
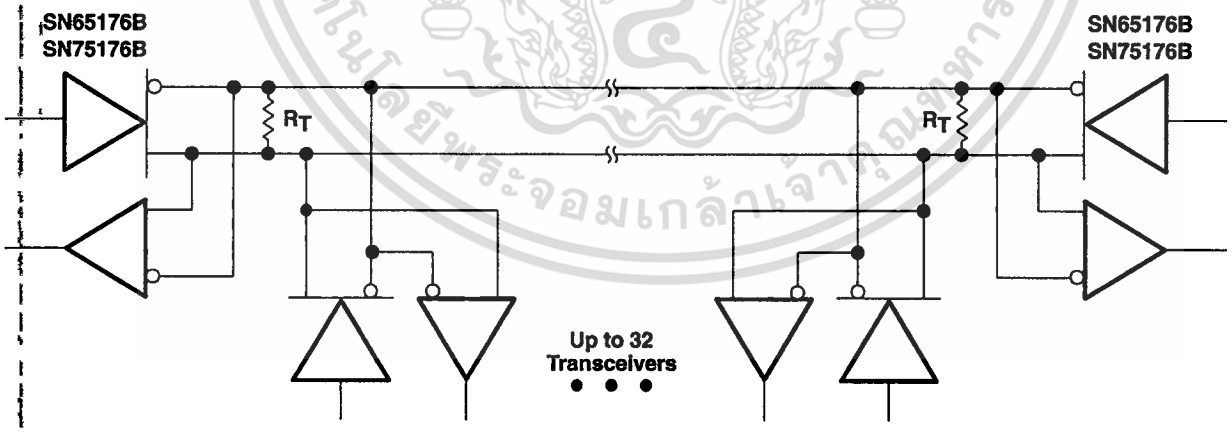


Figure 16

APPLICATION INFORMATION



NOTES: A. The line should be terminated at both ends in its characteristic impedance ($R_T = Z_O$). Stub lengths off the main line should be kept as short as possible.

Figure 17. Typical Application Circuit

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.