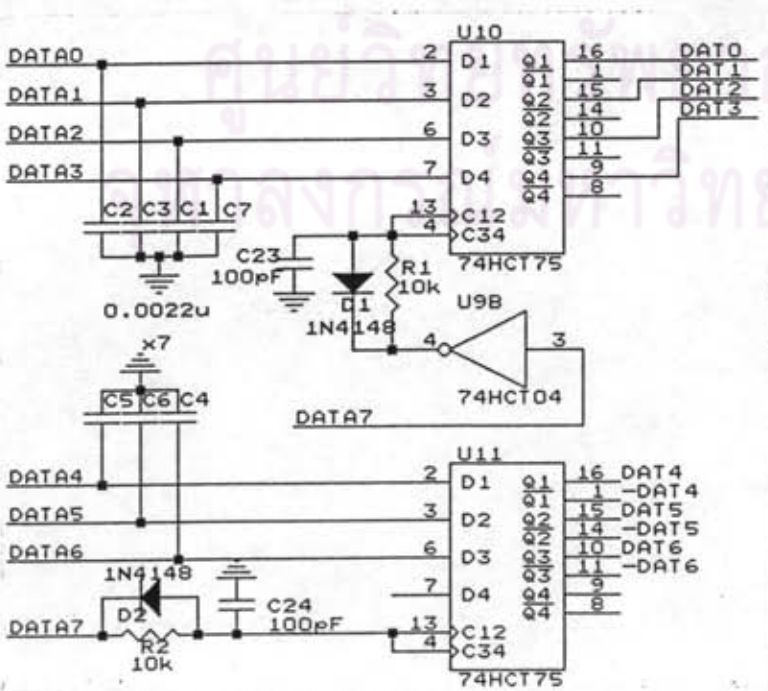
บทที่ 3
หลักการทำงานของวงจร

ในบทนี้จะได้กล่าวถึงฮาร์ดแวร์และซอฟต์แวร์ของชุดฝึกทดลอง

3.1 เมนบอร์ด (Main Board)

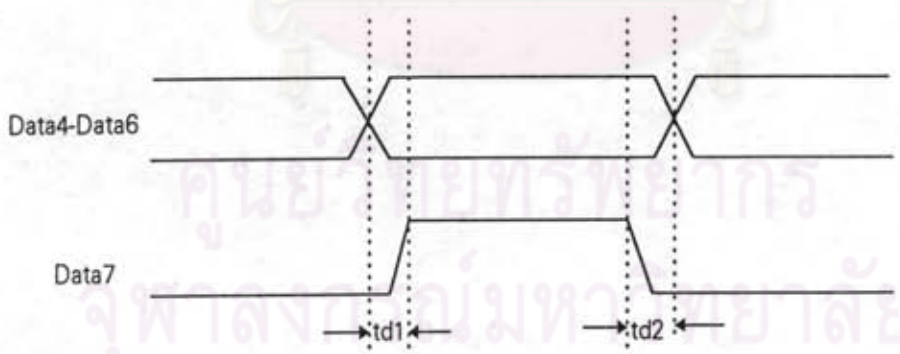
3.1.1 หลักการทำงาน

เมนบอร์ด ทำหน้าที่ในการสร้างสัญญาณของระบบบัส โดยส่วนสำคัญจะอยู่ที่การสร้างสัญญาณควบคุมตามแผนภาพการจัดจังหวะเวลา (ดูในรูปที่ 2.9) จากบล็อกไดอะแกรมของเมนบอร์ด (รูปที่ 2.8) สัญญาณออก ขนาด 8 บิต คือ Data0-Data7 จากพอร์ต A ของพอร์ตขนาน จะนำมาสร้างเป็นสัญญาณ ข้อมูลออกภายใน ขนาด 4 บิต และสัญญาณควบคุม ขนาด 3 บิต โดยใช้สัญญาณ Data7 ของพอร์ต A เป็นตัวกำหนดว่า สัญญาณที่ส่งออกมาจากพอร์ต A เป็นข้อมูล หรือ สัญญาณควบคุม ถ้า Data7 มีตรรก เป็น 1 สัญญาณที่ส่งออกมาคือสัญญาณควบคุม ข้อมูลจากพอร์ต A ในบิตของ Data4-Data6 จะถูกส่งออกไปยัง Dat4-Dat6 ผ่าน U11 โดยไม่มีการเปลี่ยนแปลงข้อมูลใน Dat0-Dat3 ถ้า Data7 มี ตรรก เป็น 0 สัญญาณที่ส่งออกมาจะเป็นสัญญาณ Internal data output ข้อมูลจากพอร์ต A ในบิตของ Data0-Data3 จะถูกส่งไปยัง Dat0-Dat3 ผ่าน U10 โดยไม่มีการเปลี่ยนแปลงข้อมูลใน Dat4-Dat6



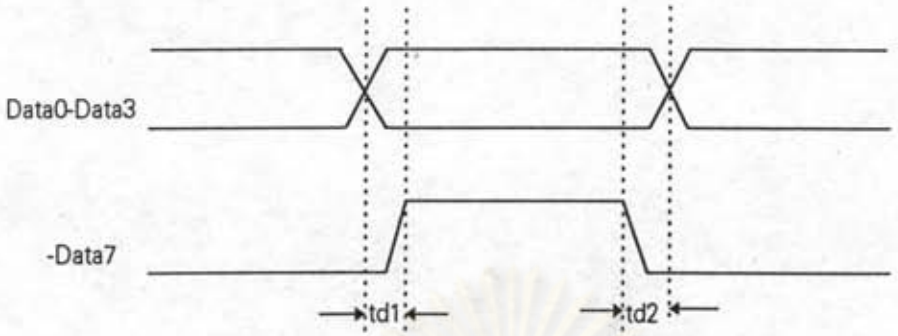
รูปที่ 3.1 วงจรแยกข้อมูล ใน Data0-Data7

U10, U11 ใช้ไอซีเบอร์ 74HCT75 ซึ่งเป็นแลตช์แบบทวิเสถียร (Bistable latch) ขนาด 4 บิต Bistable แลตช์ โดยจะผ่านข้อมูลจากขา D ไปขา Q เมื่อขา C มีสถานะเป็น High (Level Triggered) (จากรูปที่ 3.1) เมื่อ Data7 มีตรรกเป็น 1 จะทำให้ Data4-Data6 ถูกส่งผ่านไปยัง Dat4-Dat6 แต่เนื่องจาก Data4-Data6 และ Data7 ถูกส่งออกมาพร้อม ๆ กัน จากพอร์ตขนาน จึงอาจมีโอกาสที่ U11 เปิดให้ Data4-Data6 ซึ่งเป็นข้อมูลเดิม ออกไปที่ Dat4-Dat6 ได้ ซึ่งจะทำให้เกิด hazard อันจะทำให้วงจรทำงานผิดพลาดได้ จึงต้องทำการหน่วงเวลาสัญญาณ Data7 เพื่อให้เกิดความแน่ใจว่าข้อมูลใหม่ของ Data4-Data6 ได้ถูกส่งออกมาเรียบร้อยแล้ว ก่อนที่ Data7 จะมีสถานะเป็น High และเมื่อ Data7 เปลี่ยนสถานะจาก High เป็น Low จะต้องสามารถรักษาข้อมูลเดิมของ Dat4-Dat6 ไว้ให้ได้ ซึ่งในลักษณะเดียวกัน Data4-Data6 และ Data7 ถูกส่งออกมาพร้อม ๆ กัน จึงมีโอกาสทำให้ข้อมูลใหม่ของ Data4-Data6 ถูกส่งออกไปยัง Dat4-Dat6 ได้ อาจทำให้วงจรทำงานผิดพลาดได้ ฉะนั้น เหตุการณ์ของการเปลี่ยนสถานะของ Data7 เปลี่ยนสถานะจาก High เป็น Low จะต้องเกิดขึ้นก่อนการเปลี่ยนแปลงข้อมูลของ Data4-Data6 จึงต้องหน่วงเวลาของสัญญาณ Data4-Data6 จากลักษณะข้างต้นเราสามารถเขียนแผนภาพการจัดจังหวะเวลา Data4-Data7 ได้ดังนี้



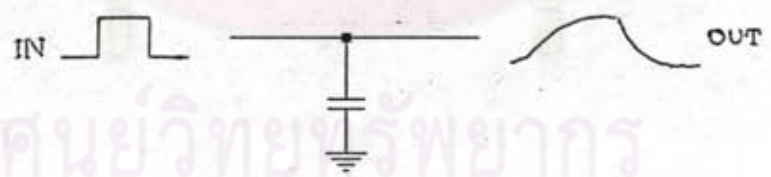
รูปที่ 3.2 สัญญาณของ Data4-Data7

ในทำนองเดียวกัน สำหรับสัญญาณ Data0-Data3 และ -Data7 จะต้องมีลักษณะดังเช่นแผนภาพการจัดจังหวะเวลา ในรูปที่ 3.2

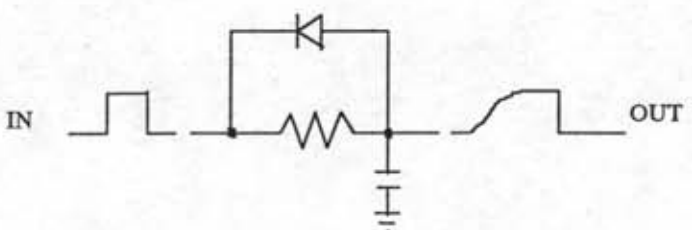


รูปที่ 3.3 สัญญาณของ Data0-Data3 และ Data7

การสร้างสัญญาณดังกล่าว ใช้ C1-C9, D1, D6-D7 และ R1-R2 โดย C1-C7 เป็นวงจรหน่วงเวลาของ Data4-Data6 และ Data0-Data3 และ D6, R1, C8 ต่อเป็นวงจรหน่วงเวลาของ Data7 ในช่วงการเปลี่ยนสถานะของ Data7 จาก Low เป็น High และเนื่องจาก Data4-Data6 ได้รับการหน่วงเวลาโดย C4-C6 ด้วย วงจรหน่วงเวลาดังกล่าว จึงต้องหน่วงเวลานานกว่า และจะต้องไม่มีการหน่วงเวลาในช่วงการเปลี่ยนสถานะของ Data7 จาก High เป็น Low วงจรดังกล่าวแสดงดังในรูปที่ 3.3 ส่วน D7, R2, C9 ต่อเป็นวงจรหน่วงเวลาของ -Data7 ในช่วงการเปลี่ยนสถานะ จาก Low เป็น High มีลักษณะเช่นเดียวกับวงจรหน่วงเวลาของ D6, R1, C8



รูปที่ 3.4 วงจรหน่วงเวลาสัญญาณ



รูปที่ 3.5 วงจรหน่วงเวลาสัญญาณขาขึ้น

3.1.2 การสร้างสัญญาณควบคุม

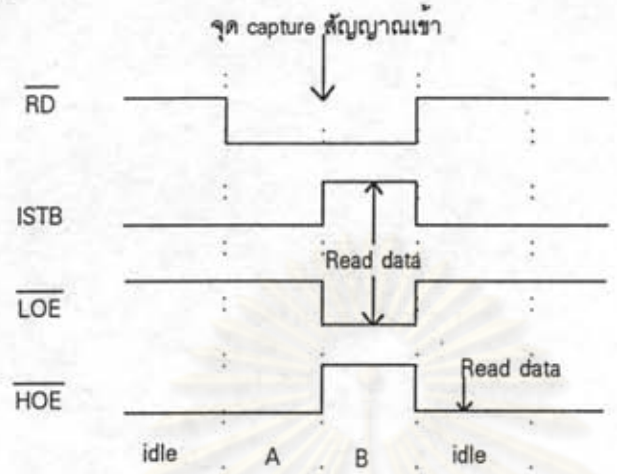
จากสัญญาณ Dat4-Dat6 3 บิต จะถูกนำมาสร้างเป็นสัญญาณควบคุมต่าง ๆ ซึ่งมีอยู่ด้วยกันทั้งหมด 12 สัญญาณ Dat4-Dat6 ใช้ในการกำหนดสถานะ และแต่ละสถานะจะกำหนดตรรกของสัญญาณควบคุม สัญญาณ Dat4-Dat6 3 บิต สามารถใช้กำหนดสถานะได้ทั้งหมด 2^3 = 8 สถานะ ฉะนั้น จะต้องพยายามลดจำนวนสถานะลงให้อยู่ในขอบเขตดังกล่าว จากรูปที่ 3.6 คือแผนภาพการจัดจังหวะเวลาของสัญญาณควบคุมที่ได้ทำการลดสถานะลงแล้ว เหลือ 8 สถานะ คือ idle state และ A-G state อนึ่ง ได้เขียนไว้ที่ใต้รูปสัญญาณ ดังรูปที่ 3.6

จากแผนภาพการจัดจังหวะเวลาของสัญญาณควบคุม เราสามารถเขียนเป็นตารางได้ดังตารางที่ 3.1 โดยในตาราง จะใช้สัญลักษณ์ D7-D4 แทน Data7-Data4

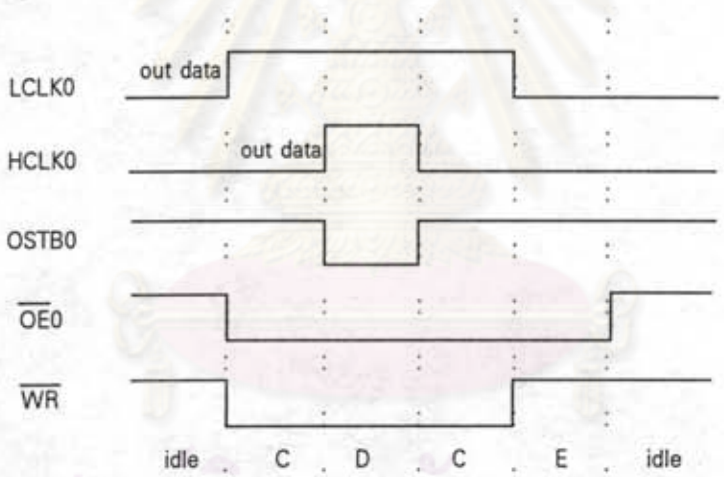
ตารางที่ 3.1 ตรรกของสัญญาณควบคุมที่สถานะต่าง ๆ ของเมนบอร์ด

State	No. Hex	D7	D6	D5	D4	$\overline{RD}$	ISTB	LCLK0	HCLK0	$\overline{OE0}$	LCLK1	HCLK1
idle	8_	1	0	0	0	1	0	0	0	1	0	0
A	9_	1	0	0	1	0	0	0	0	1	0	0
B	D_	1	1	0	1	0	1	0	0	1	0	0
C	E_	1	1	1	0	1	0	1	0	0	0	0
D	C_	1	1	0	0	1	0	1	1	0	0	0
E	F_	1	1	1	1	1	0	0	0	0	0	0
F	B_	1	0	1	1	1	0	0	0	1	1	0
G	A_	1	0	1	0	1	0	0	0	1	1	1

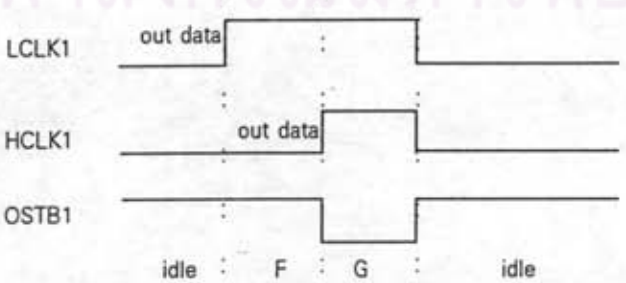
ขณะอ่านข้อมูล



ขณะเขียนข้อมูล



ขณะส่งแอดเดรส



รูปที่ 3.6 แผนภาพการจัดจังหวะเวลาของสัญญาณควบคุม และ ชื่อสถานะ

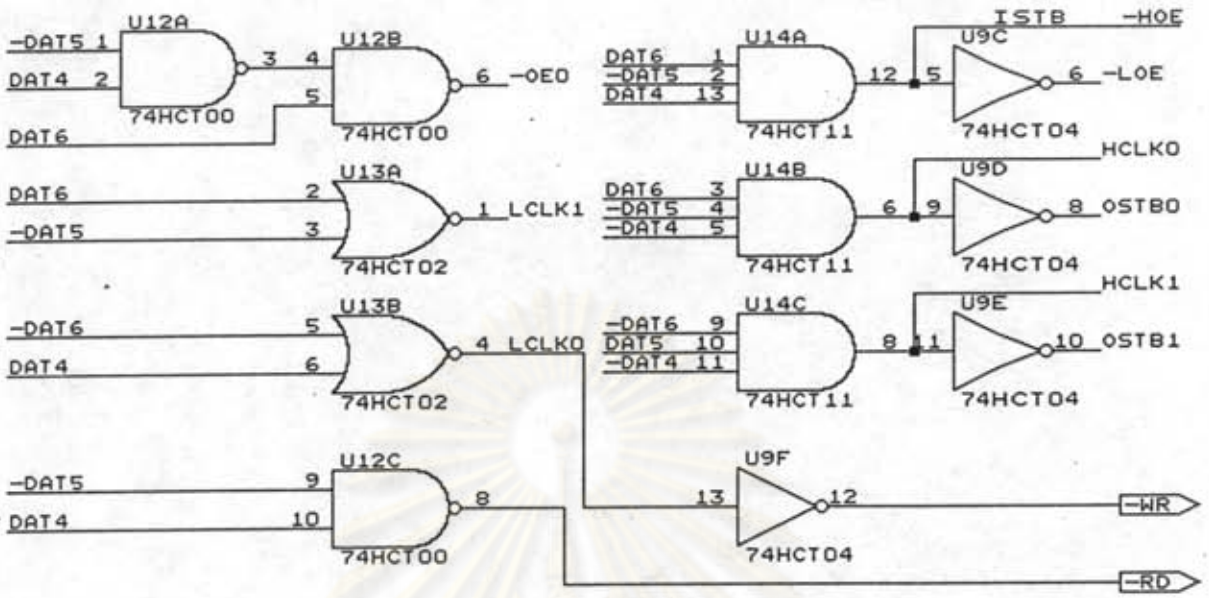
เมื่อผ่านการขจัด hazard ของสัญญาณแล้ว:

$\overline{OE0}$	=	$\overline{D6} + \overline{D5}D4$	=	$\overline{D6\overline{D5}D4}$
LCLK1	=	$\overline{D6}D5$	=	$\overline{D6 + \overline{D5}}$
LCLK0	=	$\overline{D6}D4$	=	$\overline{\overline{D6} + D4}$
$\overline{RD}$	=	$D5 + \overline{D4}$	=	$\overline{\overline{D5} \cdot D4}$
ISTB	=	$D6\overline{D5}D4$		
HCLK0	=	$D6\overline{D5D4}$		
HCLK1	=	$\overline{D6}D5\overline{D4}$		
$\overline{LOE}$	=	$\overline{ISTB}$		
$\overline{HOE}$	=	ISTB		
OSTB0	=	$\overline{HCLK0}$		
$\overline{WR}$	=	$\overline{LCLK0}$		
OSTB1	=	$\overline{HCLK1}$		

จากตารางจะเห็นว่า D4-D7 ไม่ได้เรียงตามลำดับตัวเลข ทั้งนี้ เนื่องจากได้ทำการสลับตัวเลข เพื่อให้สมการบูลลีนที่ได้ มีรูปแบบที่ง่ายที่สุดเท่าที่จะเป็นไปได้ เพื่อลดความซับซ้อนในการสร้างฮาร์ดแวร์ สมการบูลลีนดังกล่าวจะนำไปสร้างเป็นวงจรเชิงผสม (Combination) โดยมีข้อควรระวัง คือ จะต้องขจัด hazard ของสัญญาณให้หมดไป เนื่องจากอาจจะทำให้วงจรทำงานผิดพลาดได้ โดยใช้เทคนิควิธีการขจัด hazard [4]

จากตาราง เราสามารถเขียนเป็นสมการบูลลีน ได้ดังสมการที่แสดงไว้ใต้ตาราง

จากสมการบูลลีน จะนำมาสร้างเป็นวงจรเชิงผสม ดังแสดงในรูปที่ 3.7

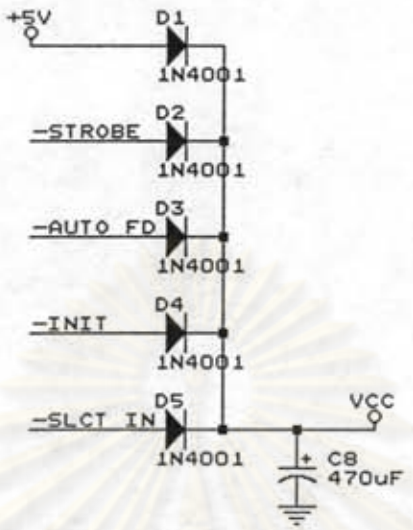


รูปที่ 3.7 วงจรเชิงผสมที่ใช้สร้างสัญญาณควบคุม

จากวงจร เราเลือกใช้ไอซีประเภท SSI แบบ HCT เนื่องจากเหตุผลที่ต้องการให้วงจรใช้กำลังไฟฟ้า เพื่อจะได้สามารถใช้ไฟเลี้ยงจากพอร์ตขนานของไมโครคอมพิวเตอร์ได้ กรณีใช้ PAL จะทำให้วงจรมีขนาดเล็ก แต่เนื่องจาก PAL ใช้กระแสมาก โดยจากการทดลองใช้กระแสประมาณ 50 mA จึงหลีกเลี่ยงไม่ใช้ในที่นี้

3.1.3 ไฟเลี้ยงของวงจร

เมนบอร์ด ถูกออกแบบให้ไม่ต้องใช้ไฟเลี้ยงจากภายนอก ใช้ไฟเลี้ยงจากพอร์ตขนาน แต่เนื่องจากพอร์ตขนานไม่มีไฟเลี้ยงส่งมา จึงใช้วิธีการใช้บิตที่เป็นสัญญาณออกที่ไม่ได้ใช้งาน มาขับให้เป็น High เพื่อป้อนเป็นไฟเลี้ยงแก่วงจร โดยผ่านไดโอด เพื่อป้องกันกระแสย้อนกลับ จากการทดลอง แต่ละบิตมีความสามารถจ่ายกระแสได้ประมาณ 15 mA โดยยังสามารถคงสภาพ High ของ TTL ได้ เราใช้งานทั้งหมด 4 บิต เพราะฉะนั้น จะจ่ายกระแสรวมได้สูงสุด 15x4 = 60 mA C8 ต่อไว้เพื่อเป็นแหล่งจ่ายกำลังสำรองในช่วงที่มีการเปลี่ยนแปลงสัญญาณ อย่างไรก็ตาม เราสามารถใช้ไฟเลี้ยงจากภายนอกได้ด้วย ในกรณีที่ต้องการให้บัสของระบบสามารถขับกระแสได้มาก ๆ (ดูรูปที่ 3.8)



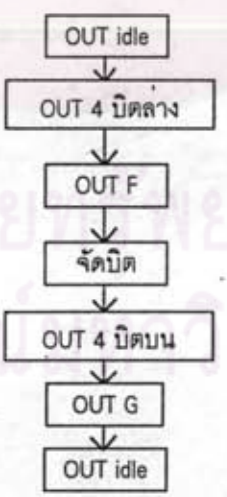
รูปที่ 3.8 วงจรจ่ายไฟของเมนบอร์ด

3.1.4 การเขียนซอฟต์แวร์ควบคุม

การสร้างสัญญาณควบคุม และสัญญาณบัสนั้น ทำได้โดยการเขียน-อ่านพอร์ตขนาน ในลำดับเวลาที่ถูกต้อง ดังแสดงในรูปที่ 3.9-3.11

รูปที่ 3.9 ขั้นตอนการรับข้อมูลเข้า

รูปที่ 3.10 ขั้นตอนการส่งข้อมูลออก



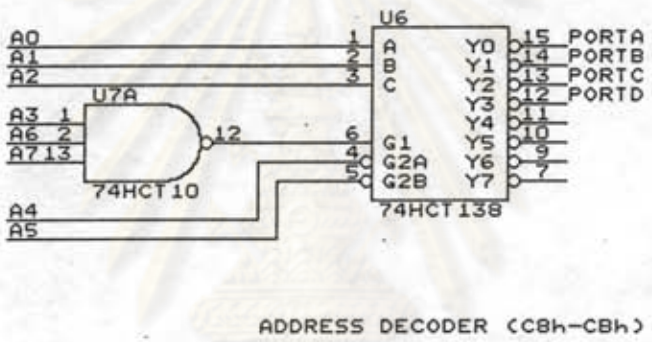
รูปที่ 3.11 ขั้นตอนการส่งแอดเดรสออก

3.2 มอดูลพอร์ต I/O

มอดูลพอร์ต I/O แบ่งออกเป็น 3 ส่วน คือ ส่วนของการถอดรหัสแอดเดรส ส่วนของวงจรสัญญาณเข้า และ ส่วนของวงจร สัญญาณออก มีรายละเอียดดังต่อไปนี้

3.2.1 ส่วนของวงจรการถอดรหัสแอดเดรส

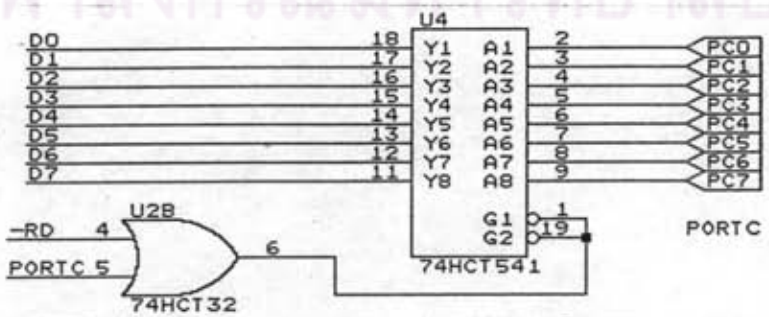
ใช้ไอซีเบอร์ 74HCT138 (3 to 8 line decoder) ร่วมกับ 74HCT11 (3 input AND gate) ทำการถอดรหัส เพื่อให้ได้แอดเดรสหมายเลข C8h-C13h เป็นสัญญาณเลือกพอร์ต A ถึงพอร์ตD ตามลำดับ



รูปที่ 3.12 วงจรถอดรหัสแอดเดรสของมอดูลพอร์ต I/O

3.2.2 ส่วนของวงจรพอร์ตสัญญาณขาเข้า

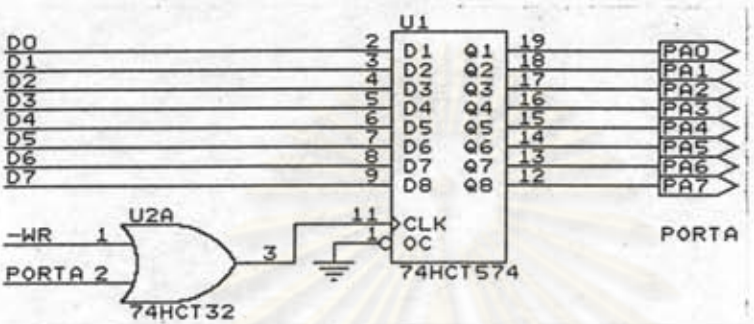
ใช้ไอซีเบอร์ 74HCT541 (3 state bus buffer) ทำหน้าที่ผ่านสัญญาณ จากอุปกรณ์ไปยังเครื่อง IBM PC



รูปที่ 3.13 วงจรพอร์ตสัญญาณขาเข้า

3.2.3 ส่วนของวงจรพอร์ตสัญญาณขาออก

ใช้ไอซีเบอร์ 74HCT574 (3 state D Type flipflop) ทำหน้าที่ในการ แลตช์ข้อมูลจาก บัสข้อมูลไปยังพอร์ตด้านออก



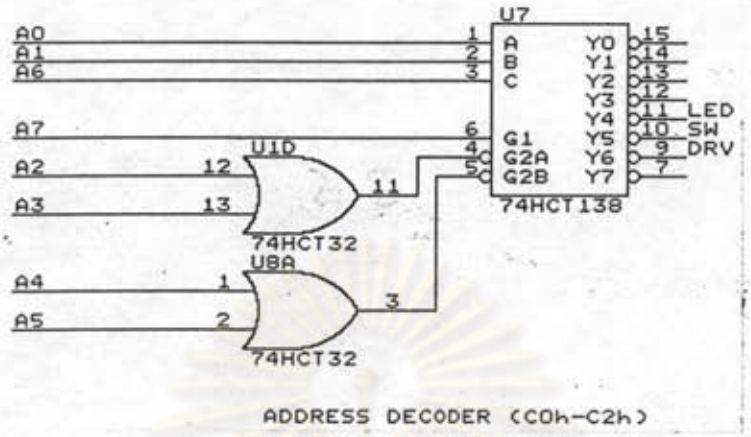
รูปที่ 3.14 วงจรพอร์ตสัญญาณขาออก

3.3 มอดูลสวิตช์

มอดูลสวิตช์ แบ่งออกเป็น 4 ส่วน คือ วงจรถอดรหัสแอดเดรส วงจรขับ LED วงจรสวิตช์ และ วงจรขับ รีเลย์ และ ทรานซิสเตอร์ มีรายละเอียดดังต่อไปนี้

3.3.1 วงจรถอดรหัสแอดเดรส

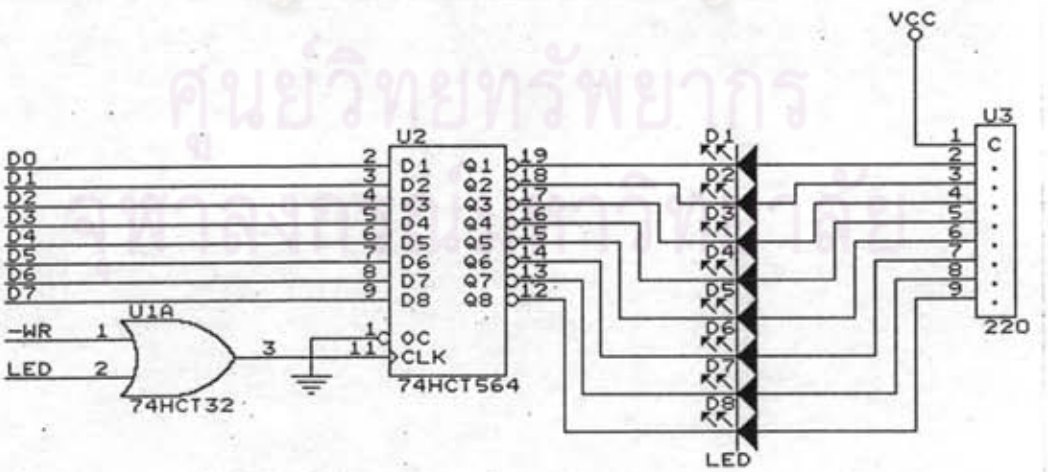
ใช้ไอซีเบอร์ 74HCT138 ร่วมกับ 74HCT32 (2 Input OR) ในการถอดรหัส หมายเลข 0h-C2h เพื่อใช้เป็นสัญญาณเลือกพอร์ต LED, สวิตช์, รีเลย์ และ ตัวขับทรานซิสเตอร์ โดย C0h เลือกพอร์ต LED C1h เลือกพอร์ตสวิตช์ และ C2h เลือกพอร์ต รีเลย์ และ ตัวขับทรานซิสเตอร์ โดยรีเลย์อยู่ในส่วนของ Low nibble และ ตัวขับทรานซิสเตอร์ อยู่ในส่วนของ High nibble



รูปที่ 3.15 วงจรถอดรหัสแอดเดรสของมอดูลสวิตช์

3.3.2 วงจรขับ LED

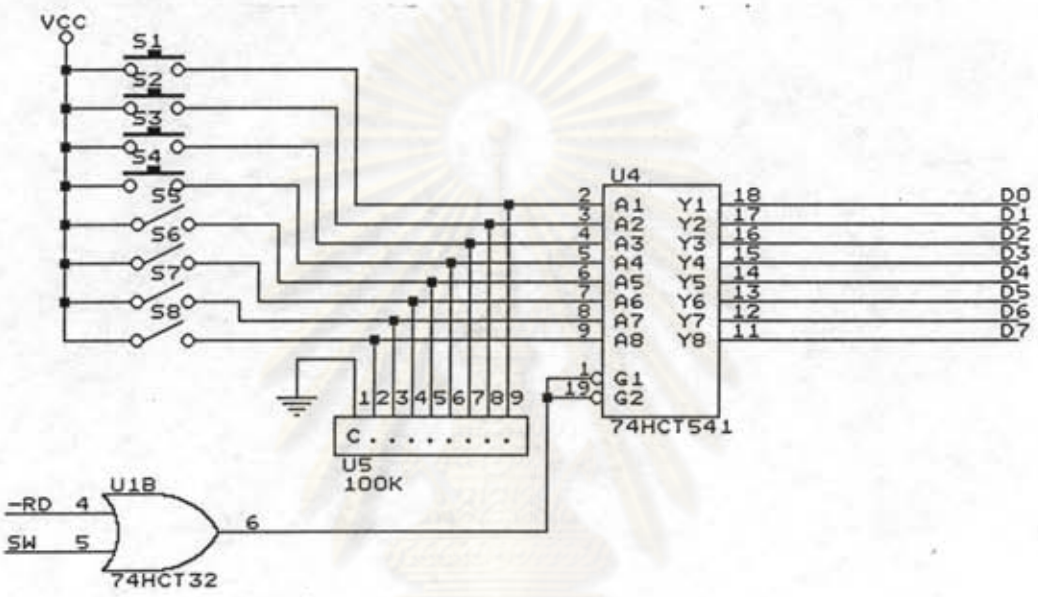
ใช้ไอซีเบอร์ 74HCT564 [3 state D type flipflop (inverted)] เพื่อ แลตช์ ข้อมูลจาก บัสข้อมูลเพื่อนำไปขับ LED และใช้ตัวต้านทานขนาด 220 โอห์ม จำกัดกระแส



รูปที่ 3.16 วงจรขับ LED

3.3.3 วงจรสวิตช์

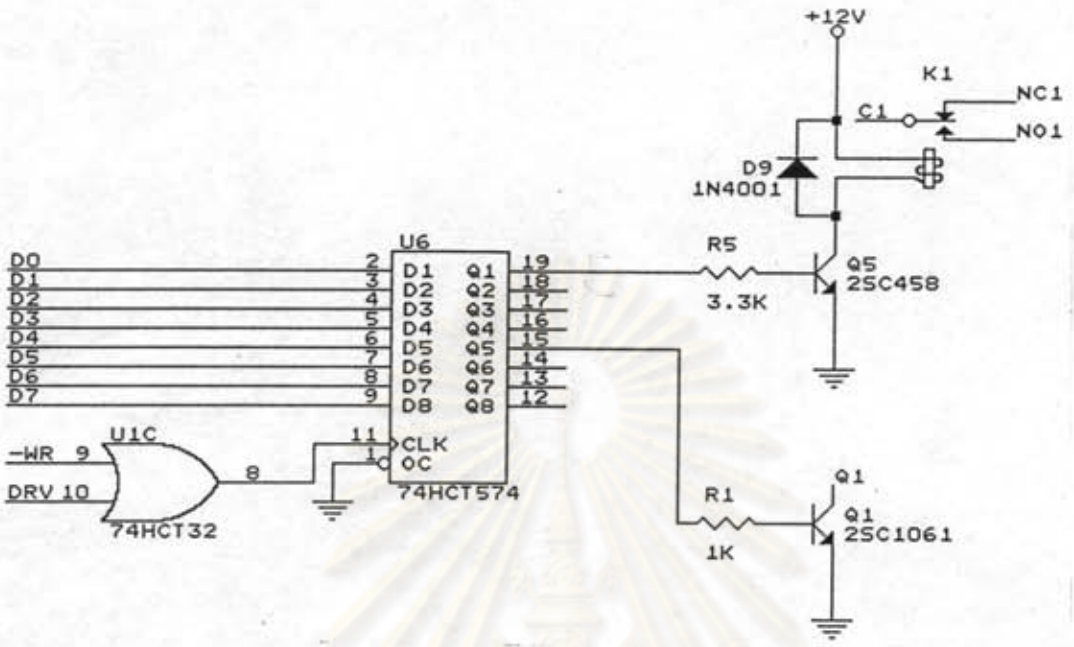
ประกอบด้วย สวิตช์กดและโยก อย่างละ 4 ตัว โดยใช้ไอซีเบอร์ 74HCT541 ในการผ่านข้อมูลจากสวิตช์ ไปยังบัสข้อมูล ตัวต้านทานขนาด 100K เป็นตัวดึงสัญญาณจากสวิตช์ให้เป็น low



รูปที่ 3.17 วงจรสวิตช์

3.3.4 วงจรขับรีเลย์และทรานซิสเตอร์

ใช้ไอซีเบอร์ 74HCT574 ในการ แลตช์ ข้อมูลจากบัสข้อมูล ส่วนวงจรขับ รีเลย์ ใช้ ทรานซิสเตอร์ เบอร์ 2SC458 และใช้ไดโอดต่อคร่อมรีเลย์ เพื่อป้องกันแรงเคลื่อนไฟฟ้ากลับทาง ที่อาจทำอันตรายต่อทรานซิสเตอร์ ส่วนวงจรขับจะใช้ ทรานซิสเตอร์ เบอร์ 2SC1061 ซึ่งทนต่อกระแสได้ 3 A[12]



รูปที่ 3.18 วงจรขับรีเลย์ และทรานซิสเตอร์

3.4 มอดูล ตัวจับเวลา/ตัวนับ

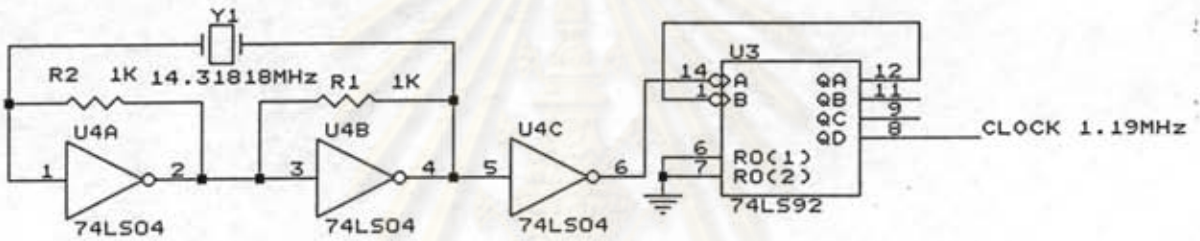
มอดูล ตัวจับเวลา/ตัวนับ ใช้ไอซีเบอร์ 8253 ซึ่งเป็น programmable interval timer ทำหน้าที่หลักของวงจร มีคุณสมบัติดังนี้[9],[13]

- เคาน์เตอร์ ขนาด 16 บิต 3 ช่องสัญญาณ
- ความถี่ใช้งาน DC-2.6 MHz
- สามารถเลือกการนับ เป็นแบบ ไบนารี หรือ BCD
- ใช้ไฟเลี้ยง ขนาด +5V
- โมด ของการนับ สามารถโปรแกรมได้ 6 โมด

นอกจากนี้ มอดูล ตัวจับเวลา/ตัวนับ ยังประกอบด้วยส่วนต่าง ๆ เพิ่มเติมคือ วงจรกำเนิดสัญญาณนาฬิกา วงจรถอดรหัสแอดเดรส วงจรควบคุม และ วงจรอ่านสถานะ ซึ่งมีรายละเอียดดังนี้

3.4.1 วงจรกำเนิดสัญญาณนาฬิกา

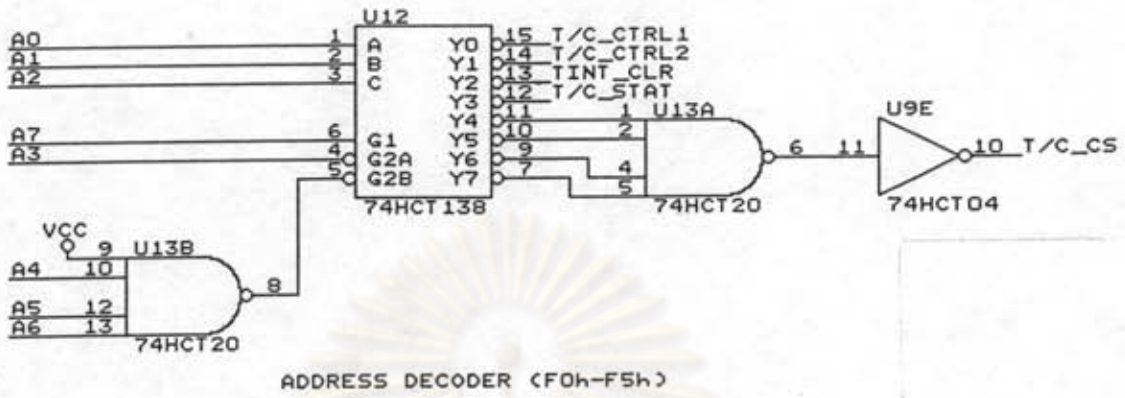
วงจรกำเนิดสัญญาณนาฬิกา จะสร้างสัญญาณนาฬิกาที่มีความถี่ 1.19 MHz ซึ่งเท่ากับความถี่ของสัญญาณนาฬิกาที่ป้อนให้แก่ 8253 ที่อยู่ภายในเครื่อง IBM PC ใช้ป้อนให้กับ 8253 เพื่อสร้างเป็นฐานเวลา ที่ใช้ในการนับ ใช้ไอซีเบอร์ 74LS04 ร่วมกับ ผลึก ประกอบเป็นวงจรออสซิลเลเตอร์ ความถี่ 14.31818 MHz แล้วนำไปผ่านวงจรหาร 12 โดยใช้ไอซีเบอร์ 74LS92 ซึ่งจะได้ความถี่เท่ากับ 1.19318 MHz หรือ มีช่วงเวลาใน 1 คาบ เท่ากับ 838.1 ns ส่วนสัญญาณออกของวงจร จะต่อเข้ากับขา CLK0 ของ 8253



รูปที่ 3.19 วงจรกำเนิดสัญญาณนาฬิกา

3.4.2 วงจรถอดรหัสแอดเดรส

ใช้ไอซีเบอร์ 74HCT138 ร่วมกับ 74HCT20, 74HCT04 ทำการถอดรหัสแอดเดรส หมายเลข F0h-F5h ขา A0-A1 ของระบบบัสต่อเข้ากับ 8253 โดยตรง เพื่อใช้เลือกรีจิสเตอร์ภายใน 8253 ที่ต้องการติดต่อด้วย สำหรับ แอดเดรส F0h ใช้ในการควบคุมการต่อสัญญาณต่าง ๆ เข้ากับขาของ 8253 แอดเดรส F1h ใช้ในการควบคุมสถานะของขาเกตของ 8253 แอดเดรส F2h ใช้ในการเคลียร์สัญญาณการอินเตอร์รัปต์ แอดเดรส F3h จะใช้ในการอ่านสถานะของขา OUT ทั้ง 3 ช่องสัญญาณ ของ 8253 แอดเดรส F4h-F7h นั้น เป็น แอดเดรสที่ใช้ในการเข้าถึงรีจิสเตอร์ ภายใน 8253 โดย F4h คือ รีจิสเตอร์ตัวนับช่องสัญญาณ 0 F5h คือ รีจิสเตอร์ตัวนับช่องสัญญาณ 1 F6h คือ รีจิสเตอร์ตัวนับช่องสัญญาณ 2 และ F7h คือ รีจิสเตอร์โมดควบคุม



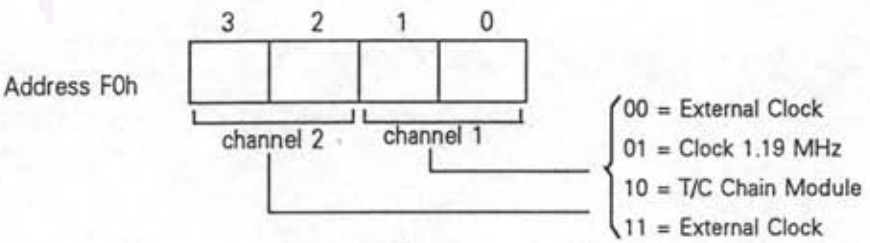
รูปที่ 3.20 วงจรถอดรหัสแอดเดรสของมอดูล T/C

3.4.3 วงจรควบคุม

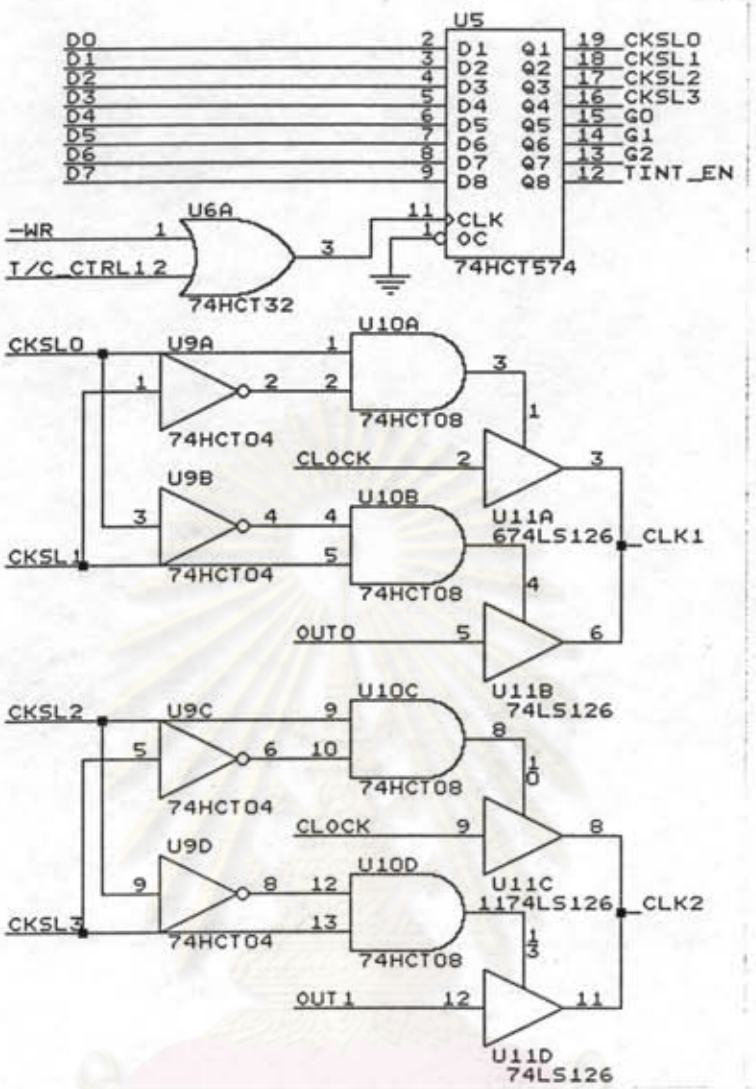
วงจรควบคุม ประกอบด้วย 3 ส่วน คือ

1) ส่วนควบคุมการป้อนสัญญาณ ให้กับขา CLK ของ 8253

CLK0 ของ 8253 จะต่อเข้ากับสัญญาณนาฬิกา 1.19 MHz ตลอดเวลา สำหรับ ขา CLK1 สามารถเลือกต่อเข้ากับสัญญาณนาฬิกาภายนอก สัญญาณนาฬิกา 1.19 MHz หรือ สัญญาณออก ของ เคาน์เตอร์ แชนเนล 0 (โมด chain) ส่วนขา CLK2 สามารถเลือกต่อเข้ากับสัญญาณนาฬิกาภายนอก สัญญาณ 1.19 MHz หรือสัญญาณออกของ วงจรนับแชนเนล 1 ส่วนควบคุมการป้อนสัญญาณนี้ จะอยู่ที่ บิต 0-3 ของพอร์ตหมายเลข F0h



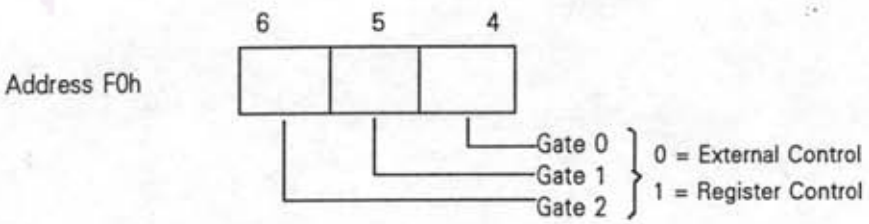
รูปที่ 3.21 บิตควบคุมการป้อนสัญญาณ ให้กับขา CLK ของ 8253



รูปที่ 3.22 วงจรควบคุมการป้อนสัญญาณให้กับขา CLK ของ 8253

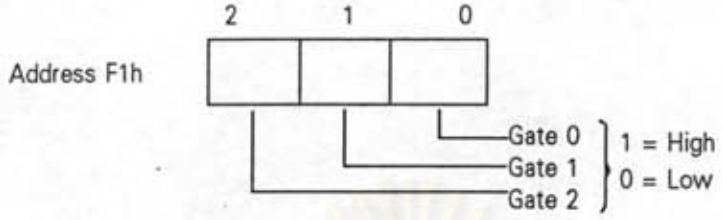
2) ส่วนควบคุมการป้อนสัญญาณ ให้กับขา gate ของ 8253

ขา gate ของ 8253 สามารถถูกควบคุมจากสัญญาณภายนอก หรือ ควบคุมได้ด้วยซอฟต์แวร์ โดยการเลือกที่บิต 4-6 ของพอร์ตหมายเลข F0h

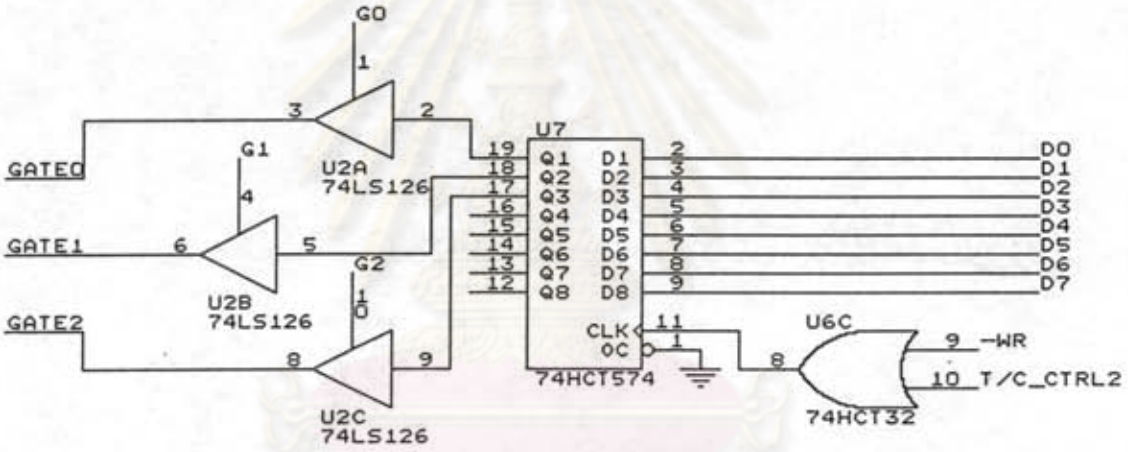


รูปที่ 3.23 บิตควบคุมการป้อนสัญญาณ ให้กับขา gate ของ 8253 โดยสัญญาณภายนอก

การควบคุมด้วยซอฟต์แวร์ ทำได้โดยการเขียนข้อมูลไปที่ บิต 0-2 ของพอร์ตหมายเลข F1h



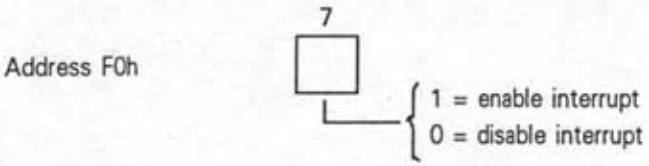
รูปที่ 3.24 บิตควบคุมการป้อนสัญญาณ ให้กับขา gate ของ 8253 โดยซอฟต์แวร์



รูปที่ 3.25 วงจรควบคุมการป้อนสัญญาณให้กับขา gate ของ 8253

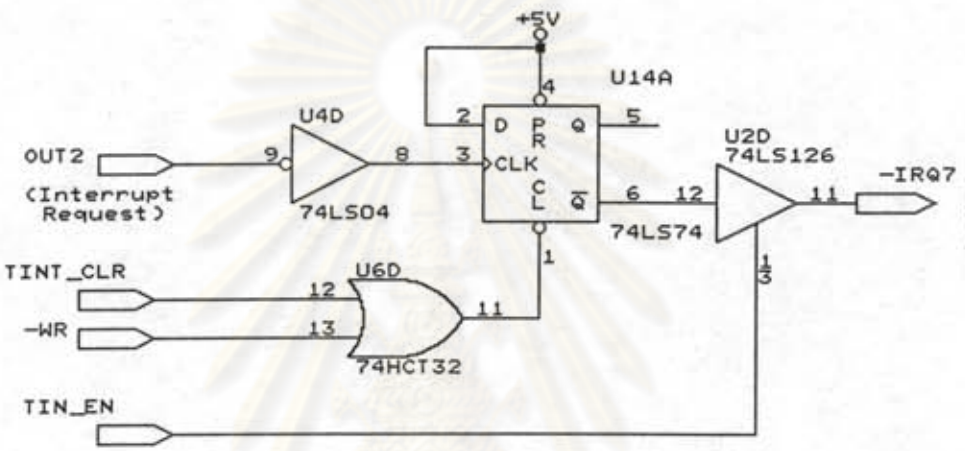
3) ส่วนควบคุมสัญญาณการอินเตอร์รัปต์

สัญญาณออกของ วงจรนับแชนเนล 2 จะสามารถเลือกได้ว่าจะต่อเข้ากับ $\overline{IRQ7}$ ของระบบหรือไม่ โดยทำการ โปรแกรม ที่บิต 7 ของพอร์ตหมายเลข F0h



รูปที่ 3.26 บิตควบคุมการอินเตอร์รัปต์

เนื่องจากสัญญาณที่ทำการร้องขออินเตอร์รัปต์จะต้องมีความยาวที่เหมาะสม เพื่อป้องกันการเกิดอินเตอร์รัปต์ซ้อน จึงต้องมีวงจรที่ทำหน้าที่เคลียร์สัญญาณการอินเตอร์รัปต์ ซึ่งมักจะทำการเคลียร์ในระหว่างการทำงานใน interrupt service routine[14,9] โดยการเขียนข้อมูลใด ๆ ไปที่พอร์ตแอดเดรส F2h



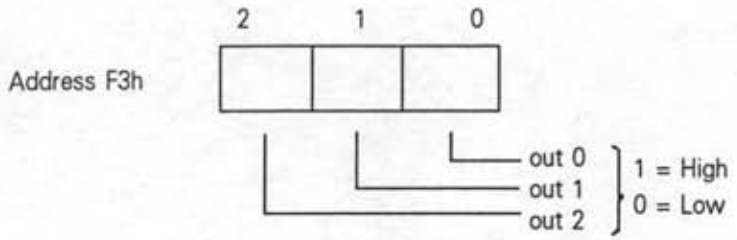
รูปที่ 3.27 วงจรการร้องขออินเตอร์รัปต์

write any data to address F2h = clear interupt request signal

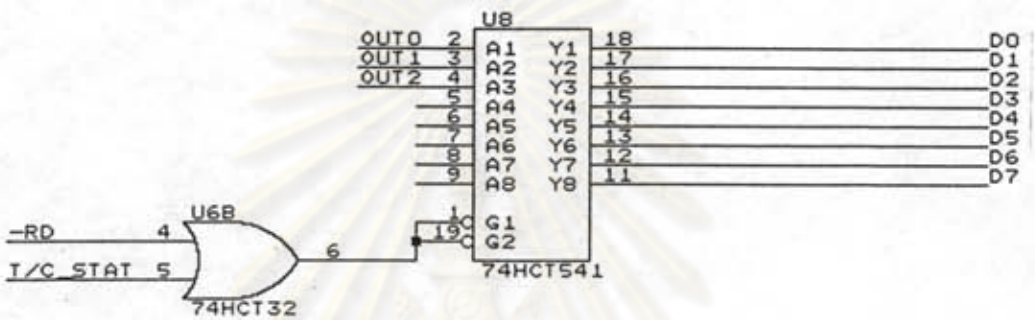
รูปที่ 3.28 การเคลียร์สัญญาณการร้องขออินเตอร์รัปต์

3.4.4 วงจรอ่านสถานะสัญญาณออกของ 8253

ใช้ไอซีเบอร์ 74HCT541 ในการผ่านสัญญาณออกของวงจรนับ ทั้ง 3 แชนเนล (OUT0-OUT2) ไปยังบัสข้อมูล การอ่านสถานะของสัญญาณออกนั้น จะทำโดยการอ่านข้อมูลจากพอร์ตหมายเลข F3h



รูปที่ 3.29 ส่วนอ่านสถานะสัญญาณออกของ 8253



รูปที่ 3.30 วงจรอ่านสถานะของขา Out ของ 8253

3.5 มอดูล ADAC

มอดูล ADAC สามารถแบ่งออกเป็น 3 ส่วน คือ วงจร A/D, วงจร D/A และ วงจรถอดรหัสแอดเดรส

3.5.1 วงจร A/D

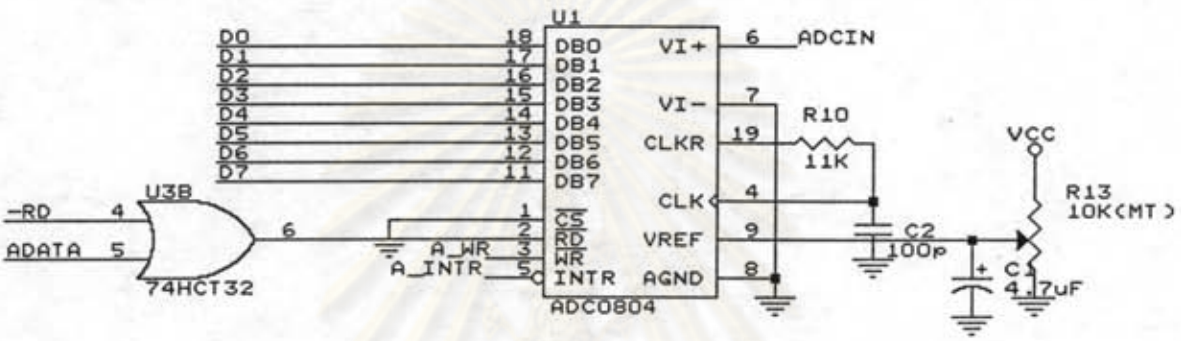
A/D คือ ส่วนที่ใช้ในการแปลงสัญญาณจากแอนะลอก เป็น ดิจิทัล โดยใช้ไอซีเบอร์ ADC0804 ซึ่งเป็น A/D ขนาด 8 บิต ที่ใช้งานร่วมกับไมโครโปรเซสเซอร์ มีความเร็วของการแปลงสัญญาณเท่ากับ 10 KHz (เวลาในการแปลงสัญญาณ 100 μS) จึงสามารถนำมาใช้กับการทดลองเรื่องการสุ่มสัญญาณเสียงได้ (โดยทั่วไปความถี่ที่ใช้ในการสุ่มสัญญาณเสียง จะต้องมีค่ามากกว่า 3.5 kHz)

ADC0804 ที่ใช้ข้างต้น จะมีวงจรสร้างสัญญาณนาฬิกาอยู่ภายใน โดยสามารถเลือกความถี่ได้ด้วยการเปลี่ยนค่า R, C ที่ต่อเข้ากับขา 4 และ 19 ตามสมการ

f = 1/(1.1*R*C) ; f = ความถี่ของสัญญาณนาฬิกา

ในที่นี้ ความถี่ที่ใช้ = 820 kHz โดยใช้ค่า R = 11K , C = 100pF

ที่ขา 9 ใช้เป็นจุดอ้างอิงแรงดันในการแปลงสัญญาณ โดยต่อวงจรให้สามารถปรับแรงดันอ้างอิงได้ โดยผ่าน Trimpot แรงดันอ้างอิงที่ได้ จะมีค่าเป็น 2 เท่าของแรงดันที่ขา 9 นี้



รูปที่ 3.31 วงจรของ ADC0804

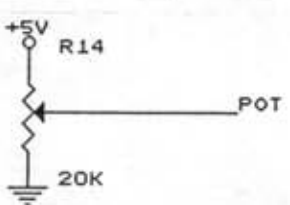
นอกจากนี้วงจร A/D ยังประกอบด้วยวงจรอีก 2 ส่วน คือ วงจรสัญญาณเข้าแบบแอนะลอก และ วงจรควบคุม มีรายละเอียดดังต่อไปนี้

1) วงจรสัญญาณเข้าแบบแอนะลอก

เป็นส่วนที่ทำหน้าที่ในการขยายสัญญาณจากเซนเซอร์ต่างๆให้มีขนาดเหมาะสม เพื่อป้อนให้กับ ไอซีA/D ต่อไป

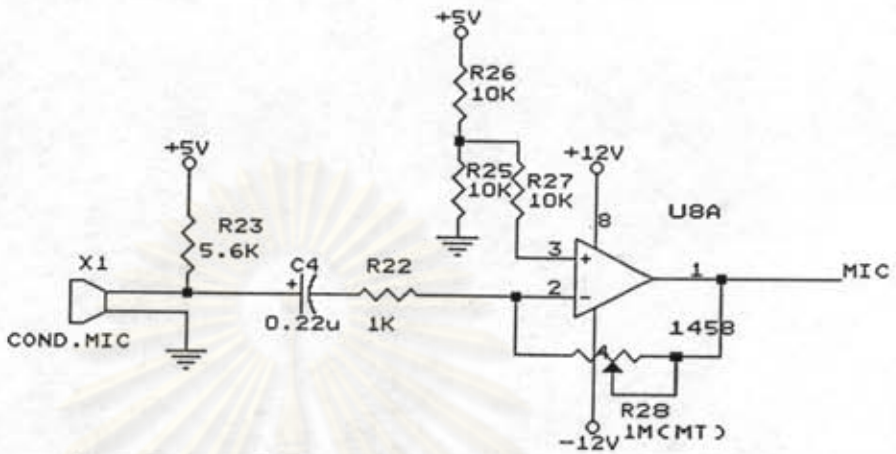
ส่วนสัญญาณเข้าแบบแอนะลอก ประกอบไปด้วย

1.1) พอต เป็นส่วนป้อนสัญญาณแรงดันที่ปรับค่าได้ ในช่วง 0-5 โวลต์



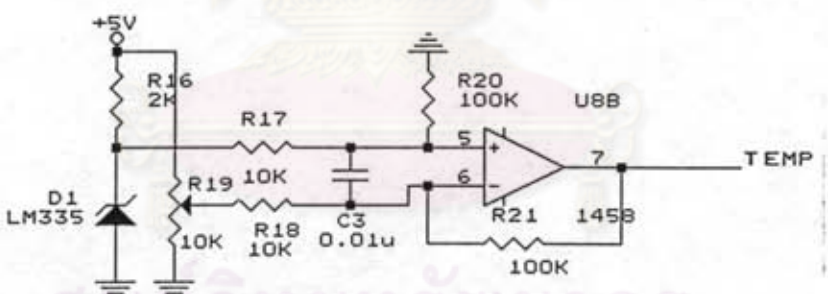
รูปที่ 3.32 วงจรพอต

1.2) ไมโครโฟน เป็นส่วนรับสัญญาณเสียงจากไมโครโฟน แล้วนำมาขยาย แล้วปรับให้สัญญาณอยู่ในช่วงแรงดัน 0-5 โวลต์ โดยสัญญาณดังกล่าว จะแกว่งขึ้นลงจากแรงดัน 2.5 โวลต์



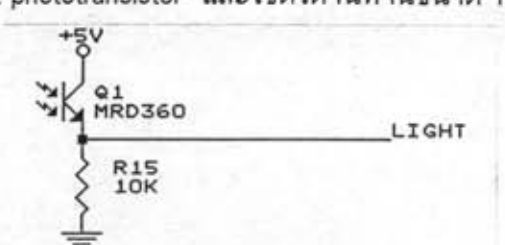
รูปที่ 3.33 วงจรไมโครโฟน

1.3) ตัววัดอุณหภูมิ ใช้ไอซีเบอร์ LM335 ต่อกับวงจรขยาย มีความไวในการวัดอุณหภูมิ 100 mV/°C



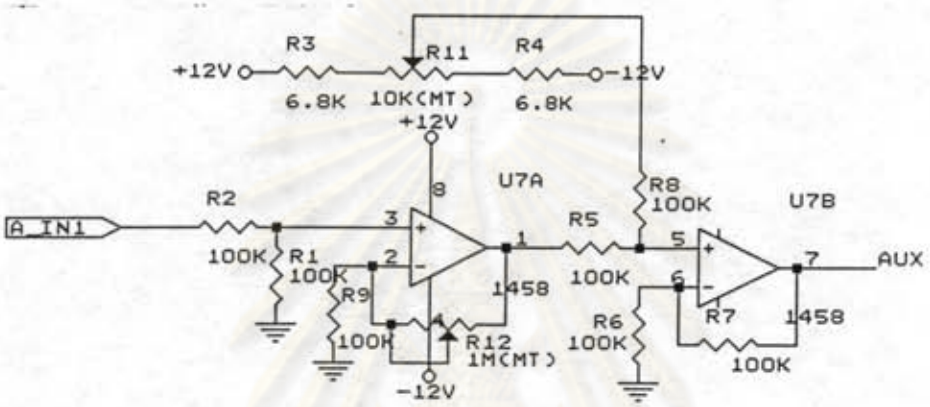
รูปที่ 3.34 วงจรตัวอุณหภูมิ

1.4) ตัววัดแสง ใช้ โฟโตทรานซิสเตอร์ เบอร์ MRD360 ซึ่งเป็น Darlington output phototransistor และใช้ตัวต้านทานขนาด 10K ในการไบแอส



รูปที่ 3.35 วงจรตัววัดแสง

1.5) วงจรปรับแต่งสัญญาณจากภายนอก(Signal Conditioning)[14],[7] กรณีสัญญาณภายนอกไม่อยู่ในช่วงที่ A/D สามารถรับได้ อาจต้องทำการปรับแต่ง โดยการขยาย และ/หรือ ปรับ offset โดยได้ออกแบบให้มีช่วงอัตราขยาย 0.5-5.5 และสามารถปรับค่า offset ได้ ตั้งแต่ -5V ถึง +5V อิมพีแดนซ์ด้านเข้า ของวงจรนี้มีค่าประมาณ 200 กิโลโอห์ม



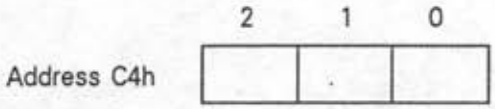
รูปที่ 3.36 วงจรปรับแต่งสัญญาณภายนอก

1.6) Analog Switch Multiplexer เป็นส่วนการเลือกสัญญาณที่ป้อนให้เข้ากับ A/D ไอซีที่ใช้ คือ เบอร์ 14051 ซึ่งเป็นตระกูล CMOS สามารถผ่านสัญญาณ แอนะลอกที่อยู่ในช่วง -0.5 ถึง 15 โวลต์ โดยยังอยู่ในช่วงเชิงเส้น [15] กราวด์สัญญาณแอนะลอก ของ 14051 จะต่อเข้ากับ กราวด์ ของระบบ

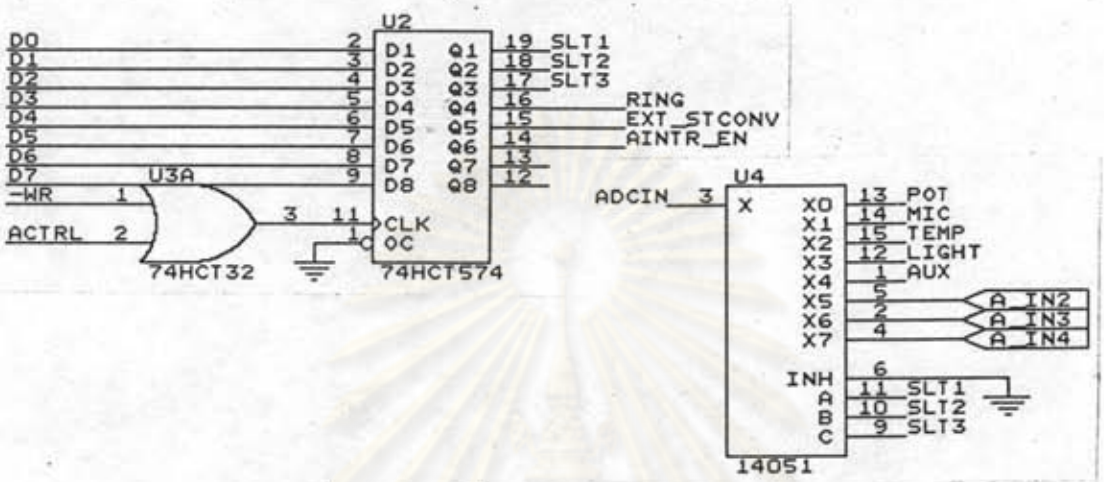
2) วงจรควบคุม

วงจรควบคุม ประกอบด้วย 3 ส่วน คือ ส่วนการเลือก แชนแนล ของสัญญาณ แอนะลอก ส่วนการเลือกวิธีเริ่มการแปลงผัน และ ส่วนควบคุม สัญญาณการอินเตอร์รัปต์

2.1) ส่วนการเลือก แชนแนล ของสัญญาณแอนะลอก ใช้ไอซีเบอร์ 74HCT574 ทำหน้าที่เป็น รีจิสเตอร์ ควบคุม โดยเอาต์พุตจะต่อเข้ากับไอซี 14051 รีจิสเตอร์นี้ อยู่ที่บิต 0-2 ของพอร์ตหมายเลข C4h



รูปที่ 3.37 บิตควบคุมการเลือกแชนแนลของสัญญาณแอนะลอก



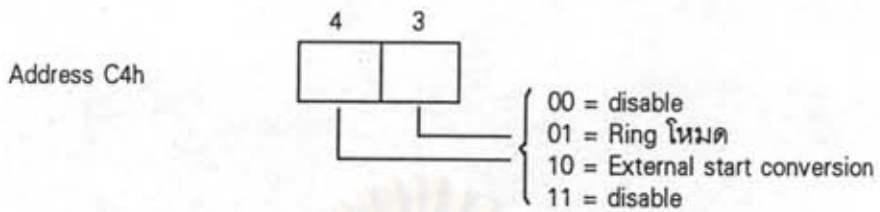
รูปที่ 3.38 วงจรเลือกแชนแนลของสัญญาณแอนะลอก

2.2) ส่วนการเลือกวิธีเริ่มการแปลงผัน ADC 0804 มีขา $\overline{WR}$ ที่ใช้ในการเริ่มต้นการแปลงสัญญาณ และขา INTR ที่ใช้บอกการสิ้นสุดการแปลงสัญญาณ การแปลงสัญญาณจะเริ่มต้นที่ขอบขาขึ้นของสัญญาณ $\overline{WR}$ และเมื่อแปลงสัญญาณเสร็จ ขา INTR จะเปลี่ยนสถานะจาก high เป็น Low และจะยังคงสถานะไปเช่นนั้นจนกว่าจะมีการอ่านข้อมูลไปจาก ADC0804 การเริ่มการแปลงสัญญาณสามารถทำได้ 2 วิธี คือ แบบ Ring และ แบบการกระตุ้นจากสัญญาณภายนอก

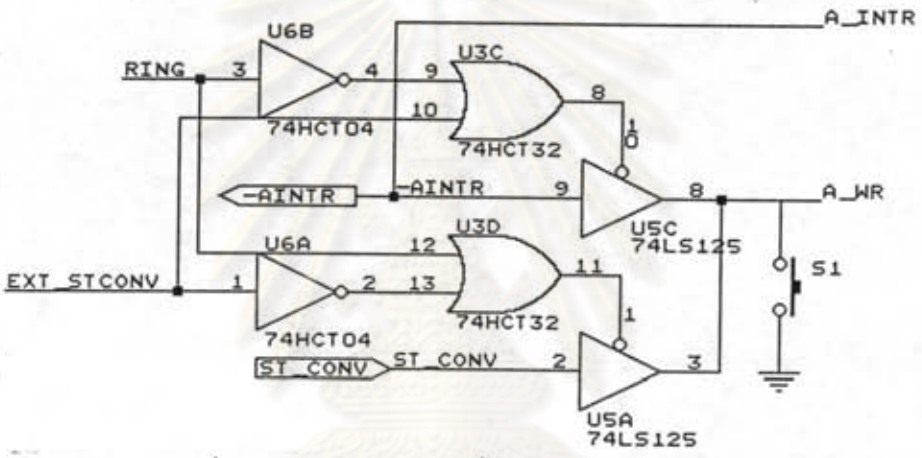
2.2.1) แบบ Ring - ใช้วิธีต่อขา INTR เข้ากับ $\overline{WR}$ วิธีนี้ เมื่อสิ้นสุดการแปลงสัญญาณแต่ละครั้ง และได้ทำการอ่านข้อมูลไปจาก ACD0804 แล้ว สัญญาณ INTR จะไปกระตุ้นให้เกิดการแปลงสัญญาณในรอบต่อ ๆ ไป ในรอบแรกของการแปลงสัญญาณอาจจะไม่เกิดการ กระตุ้น ฉะนั้นเพื่อเริ่มต้นการแปลงสัญญาณ จึงต้องต่อ S1 ไว้ เพื่อกระตุ้นให้เกิดการเริ่มต้นการแปลงสัญญาณ

2.2.2) แบบกระตุ้นจากสัญญาณภายนอก - โดยการป้อนสัญญาณภายนอกเข้าที่ขา $\overline{WR}$ การเริ่มต้นการแปลงจะเริ่มที่ขอบขาขึ้นของสัญญาณ

การพิจารณาเลือกใช้แบบ Ring หรือ แบบกระตุ้นจากสัญญาณภายนอก สามารถทำได้โดยการโปรแกรม บิต 3-4 ของพอร์ตหมายเลข C4h



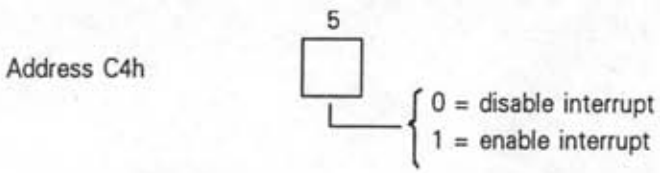
รูปที่ 3.39 บิตควบคุม ในส่วนการเลือกวิธีเริ่มการแปลงผัน



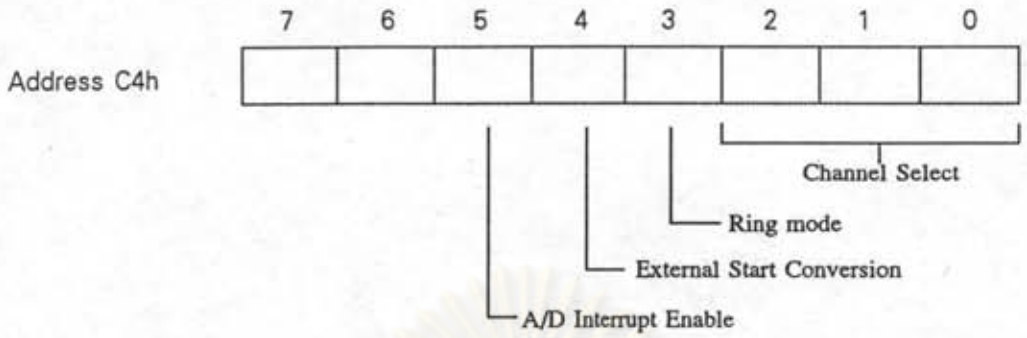
รูปที่ 3.40 วงจรเลือกวิธีการเริ่มการแปลงผัน ของ A/D

2.3) ส่วนควบคุมสัญญาณการอินเตอร์รัปต์ เราสามารถเลือกที่จะต่อสัญญาณ INTR จาก ADC 0804 เข้ากับ $\overline{IRQ7}$ ของระบบได้ เพื่อทำการอินเตอร์รัปต์ โดยการเลือกที่ บิต 5 ของพอร์ตหมายเลข C4h

สำหรับ A/D นี้จะไม่มีวงจรที่มำหน้าที่เคลียร์สัญญาณการร้องขออินเตอร์รัปต์ เนื่องจากการเคลียร์สัญญาณดังกล่าวจะเกิดขึ้นโดยอัตโนมัติในขณะที่มาทำการอ่านข้อมูลไปจาก A/D



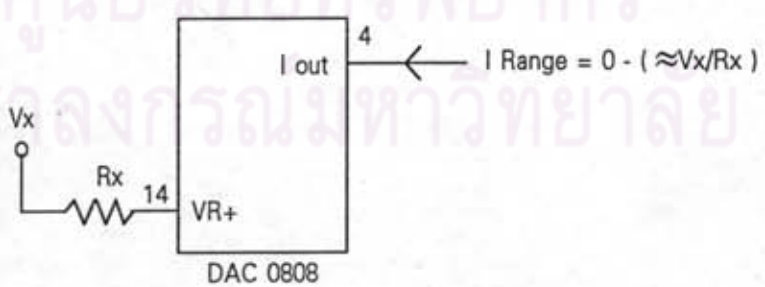
รูปที่ 3.41 บิตควบคุมการอินเตอร์รัปต์



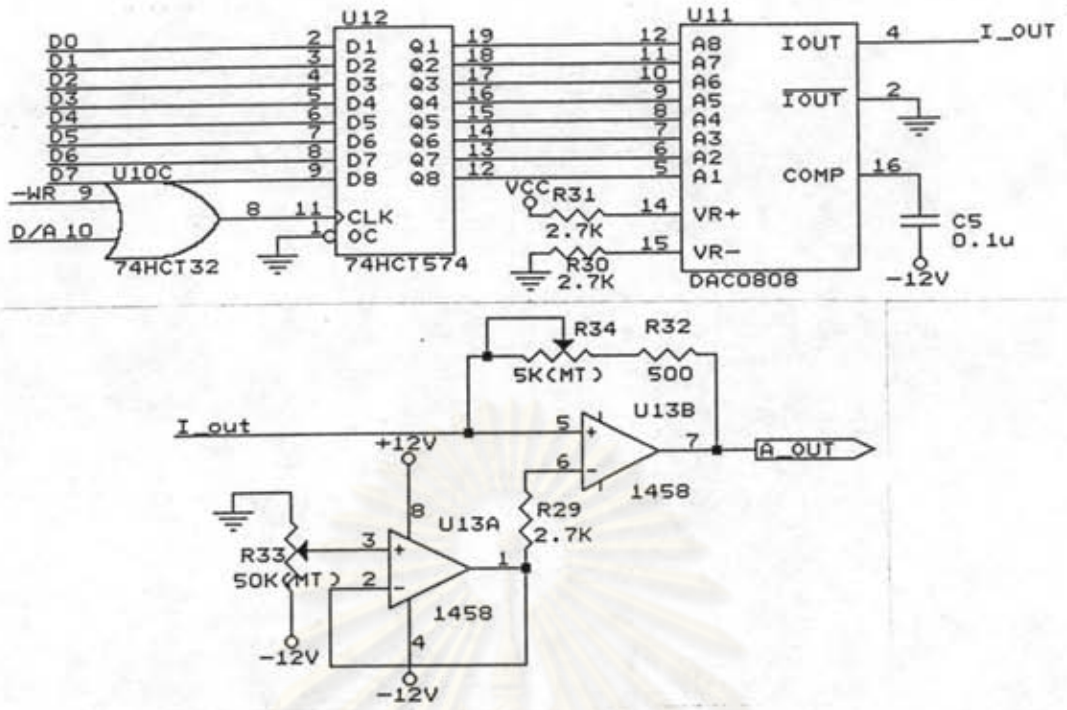
รูปที่ 3.42 รีจิสเตอร์ควบคุมของ A/D

3.5.2 วงจร D/A

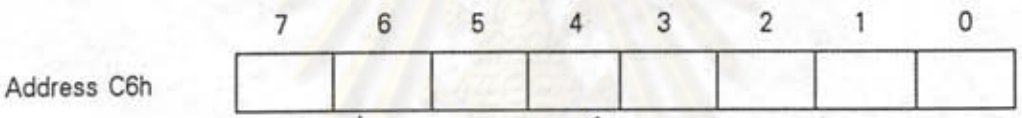
วงจร D/A คือ ส่วนที่ใช้ในการแปลงสัญญาณ จาก ดิจิทัลไปเป็นแอนะลอก ไอซีที่ใช้คือเบอร์ DAC0808 ซึ่งเป็น D/A ขนาด 8 บิต ใช้เวลาในการแปลงสัญญาณ ประมาณ 150 ns (settling time) ให้สัญญาณออกเป็นกระแส ในช่วง 0-2 mA โดยสามารถปรับช่วงได้ที่ขา 14 (ดูรูปที่ 3.43)[16] สัญญาณออกกระแสที่ได้ จะนำไปผ่านวงจรแปลงกระแสเป็นแรงดัน โดยใช้ ออปแอมป์เบอร์ 1458 ต่อเป็นวงจรขยาย ซึ่งสามารถปรับช่วงแรงดันขาออกได้ โดยมีช่วงตั้งแต่ -11V ถึง +11V และแรงดันขาออกสามารถปรับ offset ได้ในช่วง 0V ถึง -12V ส่วนที่ป้อนข้อมูล ดิจิทัลให้แก่ DAC0808 ได้ใช้ไอซีเบอร์ 74HCT574 ทำการแลตช์ ข้อมูลจาก บัสข้อมูลของระบบเพื่อป้อนให้แก่ DAC0808 ในส่วนของ D/A นี้ จะอยู่ที่พอร์ตหมายเลข C6h (รูปที่ 3.45)



รูปที่ 3.43 ช่วงของกระแสขาออกของ DAC 0808



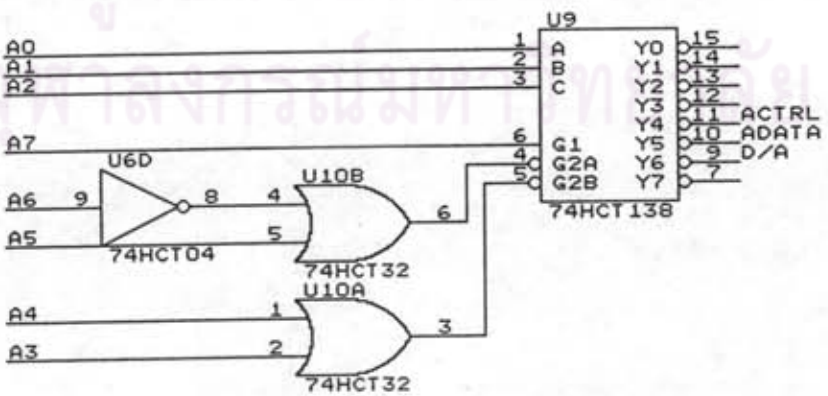
รูปที่ 3.44 วงจรของ DAC0808



รูปที่ 3.45 แอดเดรสพอร์ตของวงจร D/A

3.5.3 วงจรถอดรหัสแอดเดรส

วงจรถอดรหัสแอดเดรส ใช้ไอซีเบอร์ 74HCT138, 74HCT32 และ 74HCT04 ทำการถอดรหัสแอดเดรส หมายเลข C4h-C6h โดย แอดเดรส C4h คือ รีจิสเตอร์ควบคุม A/D C5h คือ A/D Data และ C6h คือ ข้อมูล D/A



รูปที่ 3.46 วงจรถอดรหัสแอดเดรสของมอดูล ADAC

3.6 มอดูลการสื่อสารแบบอนุกรม

มอดูลการสื่อสารแบบอนุกรม ทำหน้าที่ในการเชื่อมต่อ DTE (data terminal equipment) เข้ากับ DCE (data communication equipment) [17] หรือ เชื่อมต่อ DTE เข้ากับ DTE หรือ DCE เข้ากับ DCE ลักษณะการทำงานเหมือนกับ Null Modem คือ ใช้ในการไขว้สาย jump สายของสายเคเบิล ตามมาตรฐาน RS-232 ทั้งนี้ เพื่อให้ตรงกับความต้องการของระบบ นอกจากนี้ ในด้านหนึ่งของพอร์ต มี LED แสดงสถานะของสัญญาณต่าง ๆ เพื่อใช้บอกสถานะการทำงานของระบบ LED ได้ต่อไว้ เพื่อแสดงสถานะของสัญญาณต่าง ๆ ดังนี้

Tx	- Transmitted Data
Rx	- Received Data
RTS	- Request to Send
CTS	- Clear to Send
DTR	- Data Terminal Ready
DSR	- Data Set Ready
DCD	- Data Carrier Detect
RI	- Ring Indicator

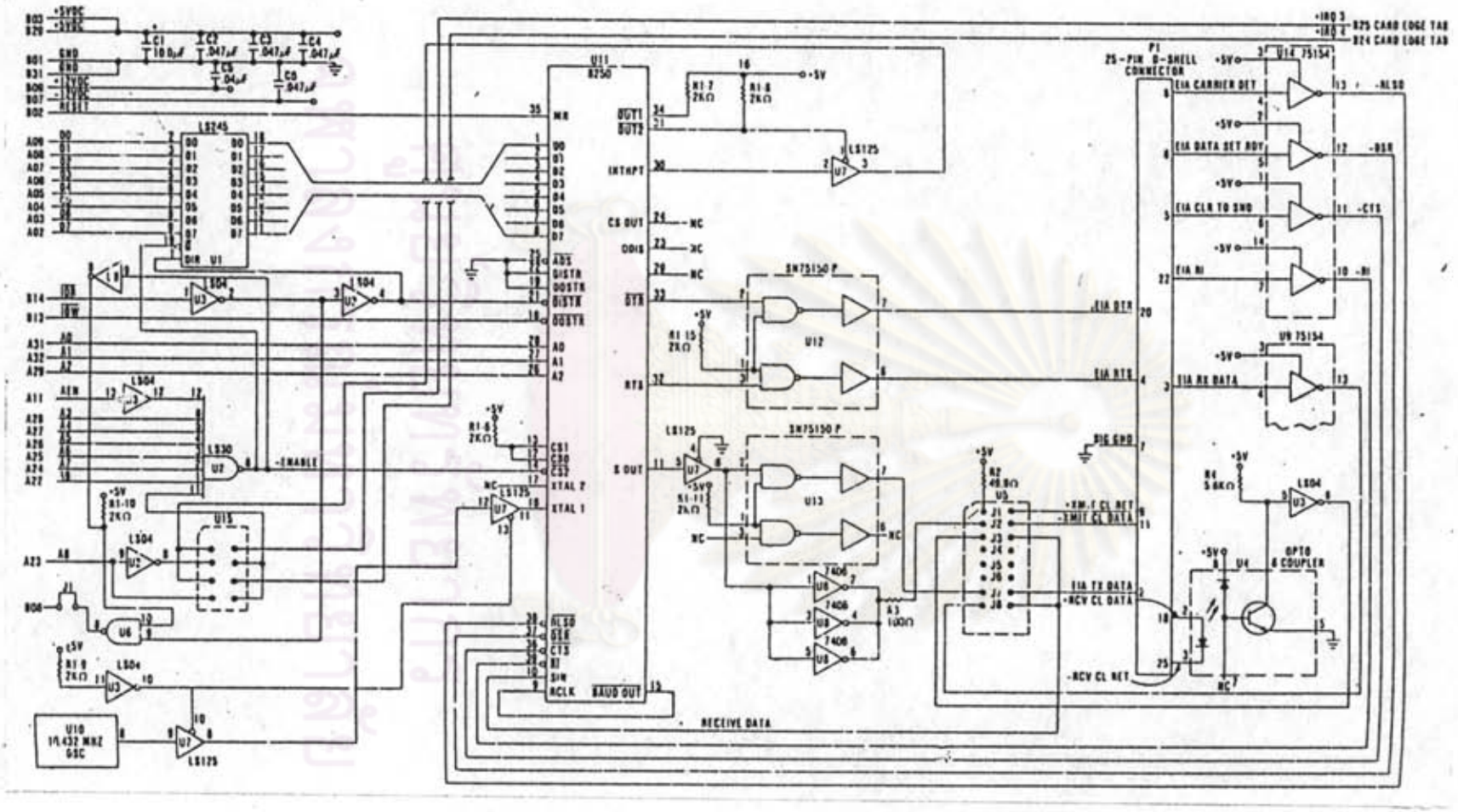
การประยุกต์มอดูลใช้งาน เราสามารถนำมาใช้เชื่อมต่อ IBM PC เข้ากับ โมเด็ม (DTE กับ DCE) เชื่อมต่อ IBM PC 2 เครื่อง เข้าด้วยกัน (DTE กับ DTE) เพื่อสื่อสารข้อมูลถึงกัน หรือใช้ทำ loop back เพื่อป้อนสัญญาณกลับเข้าไปยังเครื่องเดิม เพื่อทดสอบการทำงานของพอร์ตอนุกรม หรือ เพื่อการศึกษาเรื่องการสื่อสารแบบอนุกรม

สัญญาณต่าง ๆ จะต่อเข้ากับโปรโตบอร์ด (protoboard) บนมอดูล โดยสัญญาณเหล่านี้ได้มาจาก พอร์ตอนุกรมของเครื่อง IBM PC ซึ่งใช้คอนเนกเตอร์ แบบ DB25[18],[10] อธิบายดังตารางที่ 3.2

ตารางที่ 3.2 สัญญาณที่ต่อออกจากพอร์ตอนุกรมของเครื่อง IBM PC

ขาคอนเนกเตอร์	ชื่อสัญญาณ	รายละเอียด
2	Tx	Transmitted Data
3	Rx	Received Data
4	RTS	Request to Send
5	CTS	Clear to Send
6	DSR	Data Set Ready
7	DTR	Data Terminal Ready
8	RI	Ring Indicator
20	DCD	Data Carrier Detect
22	GND	Signal Ground

ในการทดลอง จะใช้มอดูลเชื่อมต่อเครื่อง IBM PC 2 เครื่อง โดยใช้พอร์ตอนุกรมดังแสดงในรูปที่ 3.47 เพื่อทำการสื่อสารข้อมูลระหว่างกัน ในกรณีที่มีเครื่อง IBM PC เพียง 1 เครื่อง ก็สามารถทำ loop back เพื่อส่งสัญญาณป้อนกลับเข้าเครื่อง โดยใช้สายเคเบิลแบบที่ใช้เชื่อมต่อ DTE กับ DCE (ไม่มีการไขว้สายภายใน) สายดังกล่าว สามารถหาซื้อได้ทั่วไป โดยเรียกสายนี้ว่า สายโมเด็ม แต่สายโมเด็มบางชนิด อาจจะเชื่อมต่อสัญญาณไม่ครบ โดยเฉพาะอย่างยิ่งสัญญาณ RI ซึ่งเป็นข้อที่ต้องนำมาพิจารณาประกอบการใช้งาน



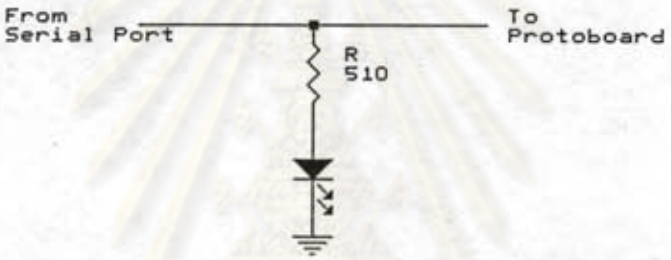
รูปที่ 3.47 วงจรของพอร์ตอนุกรมบนเครื่อง IBM PC

วงจร LED จะต่อเข้ากับสัญญาณจากพอร์ตอนุกรม โดยผ่าน ตัวต้านทานขนาด 510 โอห์ม ค่าดังกล่าวคำนวณได้จาก สัญญาณตามมาตรฐาน RS 232 ซึ่งมีระดับแรงดัน ระหว่าง +5V ถึง +15V แทน ตรรก 1 และระดับระหว่าง -5V ถึง -15V แทน ตรรก 0 ที่ ตรรก 1 จะต้องการให้ LED มีสถานะติดระดับแรงดันเฉลี่ยของ ตรรก 1 มีค่าประมาณ 8V แรงดันคร่อม LED มีค่าประมาณ 2V จึงได้ว่า

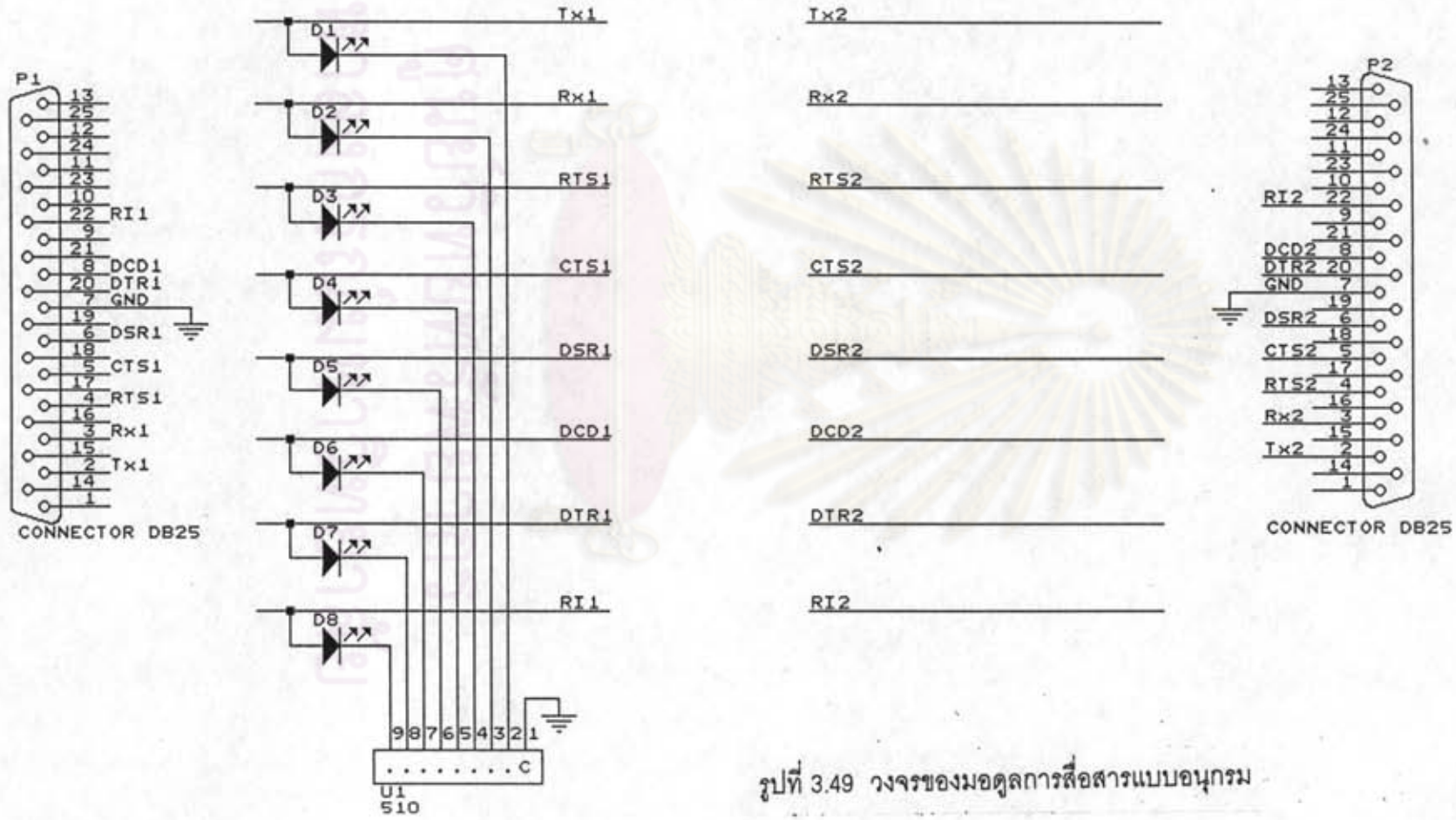
$$I = (8-2)/510 \quad A$$

$$= 12 \quad mA$$

ซึ่งอยู่ในช่วงที่ทำให้ LED สว่าง



รูปที่ 3.48 วงจร LED แสดงสถานะของสัญญาณ



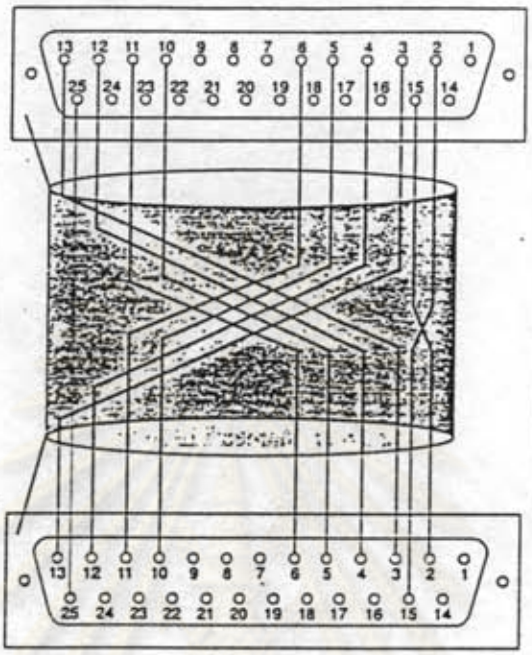
รูปที่ 3.49 วงจรของมอดูลการสื่อสารแบบอนุกรม

3.7 มอดูลการสื่อสารแบบขนาน[8]

มอดูลการสื่อสารแบบขนาน มีลักษณะเป็นกล่องที่นำสัญญาณจากพอร์ตขนานของ IBM PC 2 เครื่อง มาไว้บนโปรโตบอร์ดเพื่อนำมา jump เข้าด้วยกัน เพื่อสื่อสารข้อมูลระหว่างกัน ในกรณีที่มีเครื่อง IBM PC 1 เครื่อง สามารถทำ loop back เพื่อส่งสัญญาณย้อนกลับเข้าเครื่อง โดย jump สายบนบอร์ดต้นแบบ สัญญาณที่นำมาจากพอร์ตขนาน ประกอบด้วย สัญญาณออก และสัญญาณเข้าจำนวนอย่างละ 5 บิต ในการสื่อสารข้อมูล สัญญาณออกของพอร์ตขนานด้านหนึ่ง ๆ จะต่อเข้ากับสัญญาณเข้าของพอร์ตขนานอีกด้านหนึ่ง และมี LED เป็นตัวบอกสถานะทางตรรกของสัญญาณ โดยจะต่อเข้ากับสัญญาณของพอร์ตขนานด้านหนึ่ง สัญญาณที่นำออกจากพอร์ตขนาน มีดังนี้

- บิต D0-D4 ของพอร์ต A (แอดเดรสหมายเลข 378h หรือ 278h) ขนาด 5 บิต เป็นสัญญาณออก
- บิต 3-7 ของพอร์ต B (แอดเดรสหมายเลข 379h หรือ 279h) สัญญาณต่าง ๆ เหล่านี้ต่อเข้ากับขาต่าง ๆ ของคอนเนกเตอร์ DB 25 ของพอร์ตขนาน ดังตารางที่ 2.2

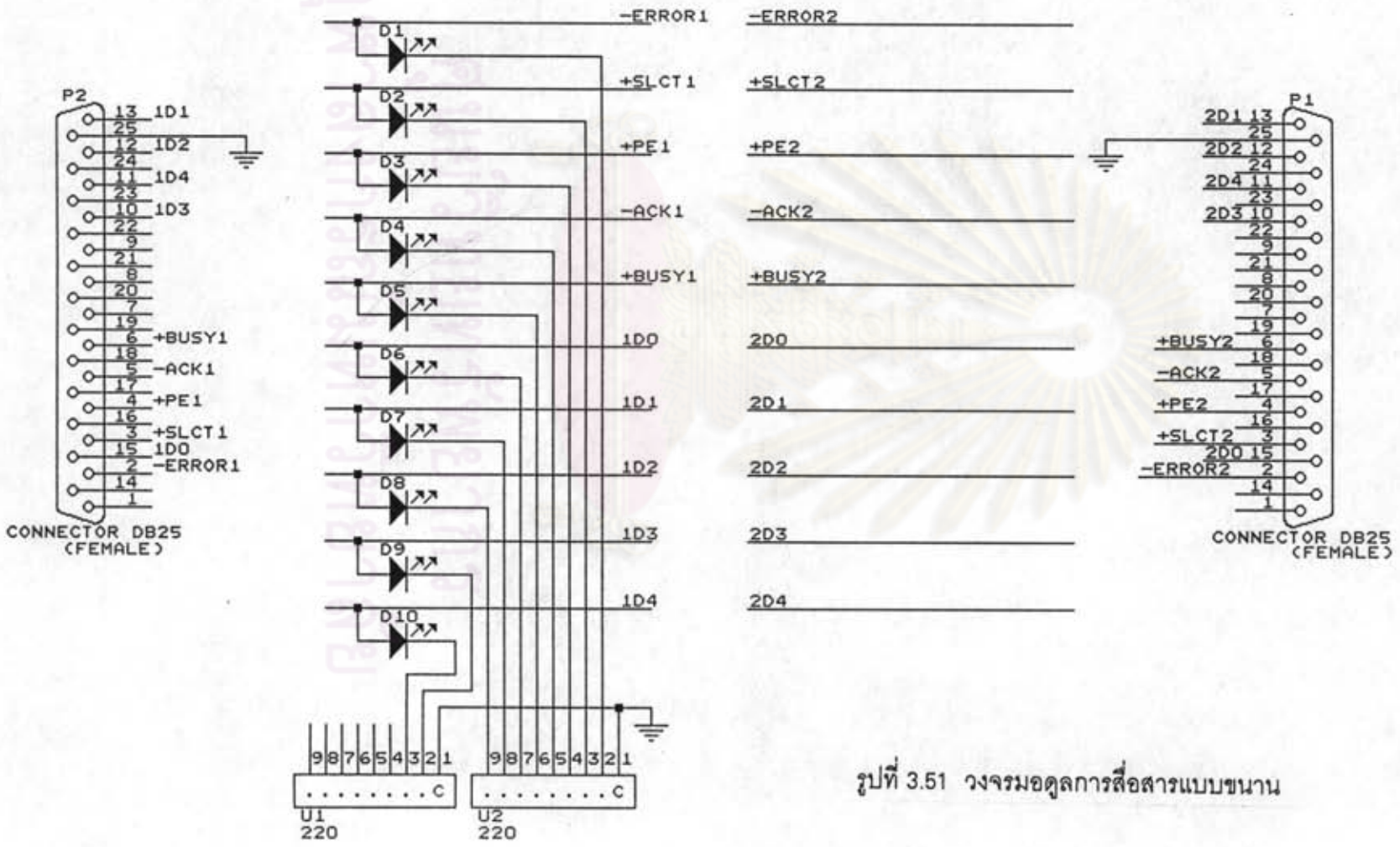
สายเคเบิลที่ใช้ นำสัญญาณจากพอร์ตขนานมายังโปรโตบอร์ด จะใช้สาย LapLink® หรือ สายที่มีความคล้ายคลึงกัน โครงสร้างภายในของสาย LapLink แสดงในรูปที่ 3.50 และตารางที่ 3.3 จะเห็นว่า สาย LapLink มีการไขว้สายภายใน ฉะนั้นในวงจรของมอดูลจึงมีการไขว้สัญญาณกัน เพื่อกลับสัญญาณให้อยู่ในตำแหน่งที่ถูกต้องดังเดิม ดังแสดงในรูปที่ 3.51 สำหรับวงจร LED ที่แสดงสถานะของสัญญาณ จะต่อเข้ากับสัญญาณโดยผ่านตัวต้านทานขนาด 220 โอห์ม



รูปที่ 3.50 การไขว้สายภายในของสาย LapLink

ตารางที่ 3.3 การไขว้สายภายในของสาย LapLink

ขา (Pin)	ขา (Pin)
2	15
3	13
4	12
5	10
6	11
15	2
13	3
12	4
11	6
10	5

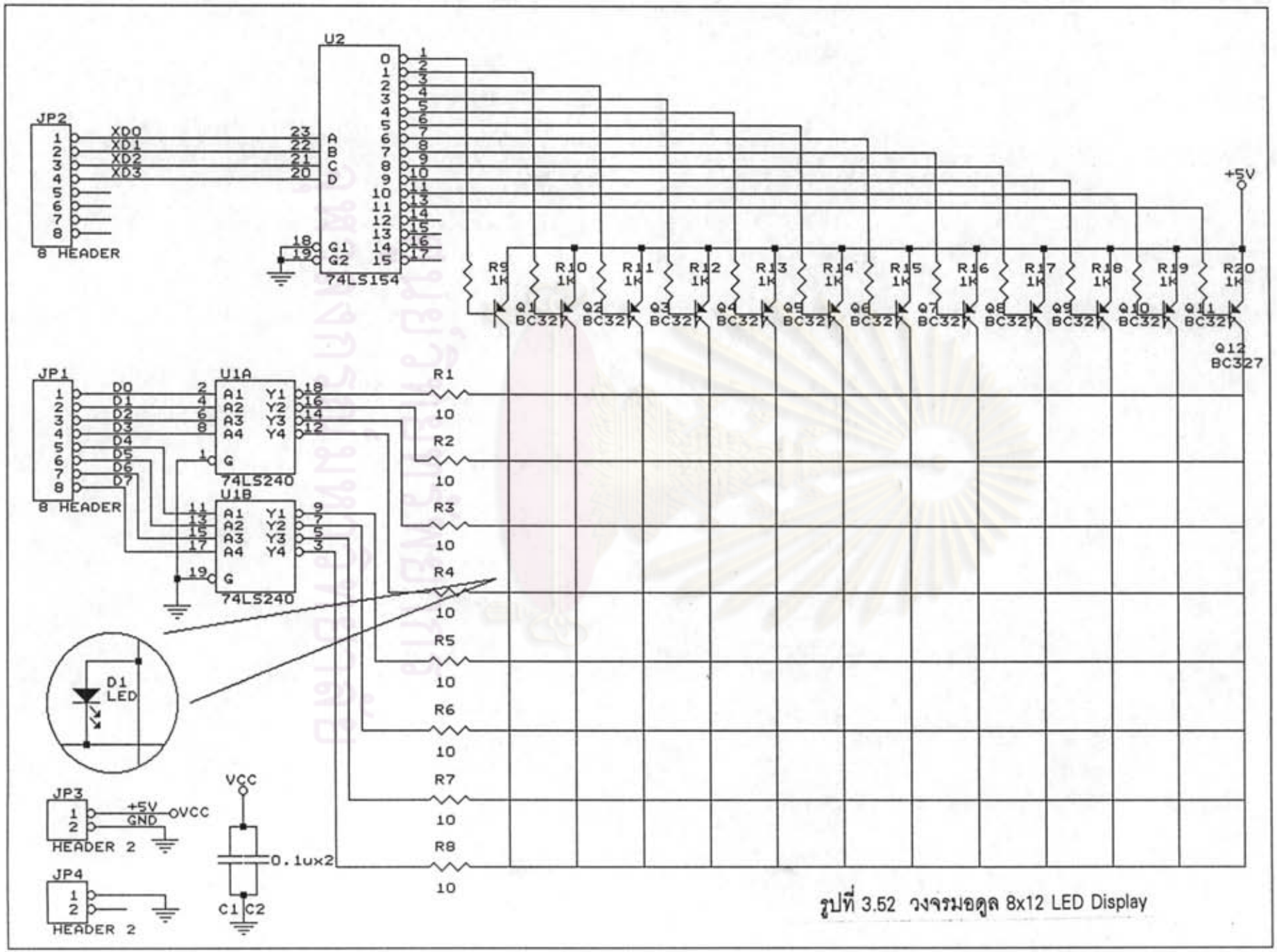


รูปที่ 3.51 วงจรมอดูลการสื่อสารแบบขนาน

3.8 มอดูลแสดงผลแบบแอลอีดีขนาด 8x12 จุด

มอดูลแสดงผลแบบแอลอีดีขนาด 8x12 จุด ใช้หลักการแสดงผลแบบ มัลติเพลกซ์ (จากรูปที่ 3.52) โดยข้อมูลจะถูกป้อนตามแนวนอน และสัญญาณมัลติเพลกซ์จะอยู่ในแนวดิ่ง มอดูลนี้ใช้ไอซีเบอร์ 74HCT244 [3 state Bus Buffer (inverted)] ในการรับสัญญาณในแนวนอนจากพอร์ต I/O และ ไอซีเบอร์ 74LS154 (4 to 16 line decoder) ในการมัลติเพลกซ์สัญญาณในแนวตั้ง เนื่องจากสัญญาณในแนวตั้ง อาจต้องขับ LED พร้อม ๆ กัน ถึง 8 ดวง จึงต้องอาศัยทรานซิสเตอร์ เบอร์ BC327 ซึ่งทนกระแสได้ 800 mA เป็นตัวขับกระแส ส่วนสัญญาณในแนวนอน ณ ช่วงเวลาหนึ่ง ๆ จะขับ LED อย่างมากเพียง 1 ตัว ทำให้ไม่ต้องใช้ทรานซิสเตอร์ช่วยในการขับกระแส

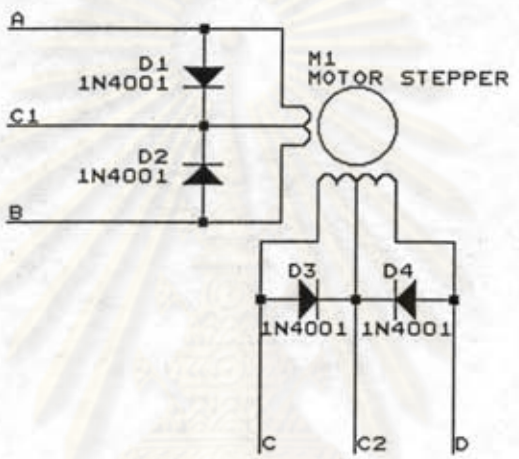
สำหรับตัวต้านทานขนาด 10 โอห์ม ทำหน้าที่ในการจำกัดกระแสที่ไหลผ่าน LED ตัวต้านทานดังกล่าว ควรมีค่าต่ำ เนื่องจากวัฏจักรงาน (duty cycle) ของการทำงานของ LED ไม่เท่ากับ 100% ซึ่งอาจทำให้ LED เปล่งแสงได้ไม่สว่างเพียงพอ จึงต้องทำการชดเชย โดยให้กระแสที่ไหลผ่าน LED มีค่ามากกว่าค่าใช้งานปกติ(ประมาณ 10 mA) มอดูลนี้จะใช้งานโดยต่อเข้ากับพอร์ต A และ B ซึ่งเป็นพอร์ตสัญญาณออกของมอดูลพอร์ต I/O



รูปที่ 3.52 วงจรมอดูล 8x12 LED Display

3.9 มอดูลสเตปปิงมอเตอร์

มอดูลสเตปปิงมอเตอร์ประกอบด้วยสเตปปิงมอเตอร์ 4 เฟส ขนาด 12 โวลต์ มีค่าความต้านทานกระแสตรงของแต่ละเฟส เท่ากับ 33 โอห์ม มีจำนวนสเตป 200 สเตป (1.8°/สเตป) ไดโอด D1-D4 ต่อเพื่อป้องกันแรงดันย้อนกลับจากมอเตอร์ ที่อาจทำอันตรายต่อวงจรขับได้ มอดูลนี้ จะใช้งาน โดยต่อเข้ากับมอดูลสวิตช์ในส่วนของวงจรขับทรานซิสเตอร์



รูปที่ 3.53 วงจรของมอดูลสเตปปิงมอเตอร์

3.10 มอดูลการควบคุมมอเตอร์[1]

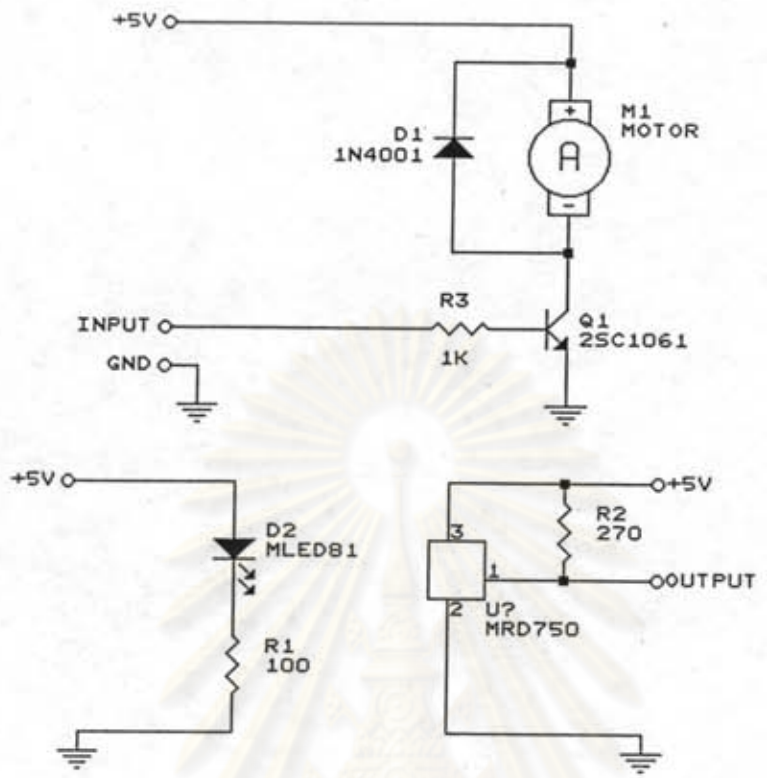
มอดูลการควบคุมมอเตอร์นี้สามารถนำมาใช้ในการศึกษาเรื่องการใช้ไมโครคอมพิวเตอร์ในการควบคุมความเร็วของดีซีมอเตอร์ ส่วนประกอบของมอดูล แบ่งได้เป็น 2 ส่วน คือ ส่วนของวงจรขับมอเตอร์ ส่วนของการวัดความเร็ว ทั้งสองส่วนมีรายละเอียดดังต่อไปนี้

3.10.1 ส่วนของวงจรขับมอเตอร์

ส่วนของวงจรขับมอเตอร์ ใช้ ทรานซิสเตอร์ เบอร์ 2SC1061 ซึ่งทนกระแสได้ 3 A มอเตอร์ที่ใช้ในขณะการทำงานที่ความเร็วปกติจะใช้กระแสประมาณ 100 mA และใช้แรงดันขนาด 6 V แต่ในวงจรนี้จะป้อนแรงดัน 5 V ให้แก่มอเตอร์ ซึ่งมอเตอร์ยังสามารถทำงานได้ การควบคุมความเร็วของมอเตอร์จะใช้วิธีควบคุมแบบป้อนสัญญาณ PWM (pulse-width modulation) ซึ่งนำมาจากพอร์ตสัญญาณออกของมอดูลพอร์ต I/O

3.10.2 ส่วนของการวัดความเร็ว

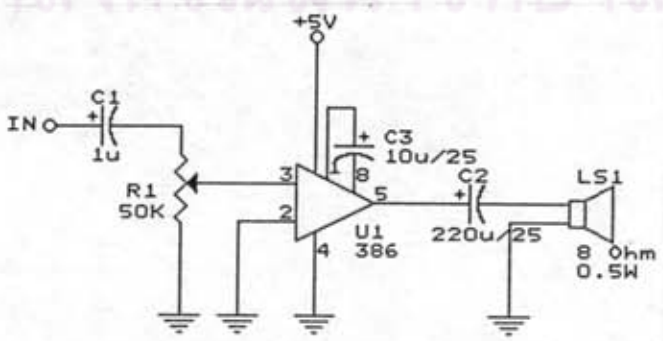
ส่วนของการวัดความเร็ว ใช้ slotted wheel ต่อเข้ากับแกนของมอเตอร์ โดยมีหลักการทำงาน คือ ใช้ slotted wheel เปิด-ปิดทางเดินของแสง และใช้อุปกรณ์ตรวจจับแสง ซึ่งจะทำให้ได้สัญญาณพัลส์ ความถี่ของสัญญาณพัลส์ที่ได้สามารถนำมาคำนวณหาความเร็วของมอเตอร์ ส่วนอุปกรณ์ที่เกี่ยวกับแสง ใช้สารกึ่งตัวนำ MOC75 ซึ่งประกอบด้วย ตัวส่ง และตัวรับแสงอินฟราเรด ประกอบเป็น photo-interrupter ให้สัญญาณออกเป็นสัญญาณ TTL จึงสามารถต่อกับวงจรไมโครโปรเซสเซอร์ได้โดยตรง สัญญาณออกเมื่อนำไปเข้าวงจรตัวจับเวลา/ตัวนับ จะทำให้สามารถทราบความถี่ของสัญญาณพัลส์ได้ ซึ่งสามารถนำไปคำนวณหาความเร็วของมอเตอร์ต่อไป



รูปที่ 3.54 วงจรของมอดูลการควบคุมมอเตอร์

3.11 มอดูลขยายเสียง (Amplifier Module)

มอดูลขยายเสียง ใช้ไอซีเบอร์ LM386 ซึ่งเป็นไอซีที่ใช้ขยายสัญญาณเสียงโดยเฉพาะ มีกำลังขับ 0.5 วัตต์ ซึ่งเพียงพอในการขับลำโพงให้เปล่งเสียงได้ VR 50 K ใช้ปรับระดับสัญญาณขาเข้าซึ่งเป็นการปรับระดับความดังของเสียง



รูปที่ 3.55 วงจรมอดูลขยายเสียง

3.12 ส่วนของโปรแกรมควบคุม

โปรแกรมควบคุมเขียนด้วยภาษา C โดยใช้โปรแกรม Turbo C ของบริษัท Borland Inc. ซึ่งเป็น C Compiler ที่มีองค์ประกอบต่าง ๆ เช่น Editor ที่ใช้ในการเขียนโปรแกรม โปรแกรม TCC ที่ใช้ในการ compile โปรแกรมให้เป็น object code และ โปรแกรม TLIB ที่ใช้ในการสร้าง Library และมีฟังก์ชันต่างๆที่สามารถนำไปใช้ได้ตามต้องการ[19],[20] โปรแกรมควบคุม แบ่งออกเป็น 2 ส่วน คือ

1) Header File (EDL.H) เป็นส่วน header ของโปรแกรม EDL.C ที่มีรายละเอียดดังนี้

```
#include <dos.h>
#define porta 0x378          /* 0x278 for LPT2 */
#define portb 0x379          /* 0x279 for LPT2 */
#define portc 0x37a          /* 0x27a for LPT2 */

void outaddr(char addr);
void out(char addr,char data);
char in(char addr);
```

ในส่วนนี้จะถูกบรรจุอยู่ในไดเรกทอรี INCLUDE ของ Turbo C โดยมีวิธีการสร้าง ดังนี้

1.1) ใช้โปรแกรม Editor ช่วยในการเขียนโปรแกรม EDL.H

1.2) บันทึกแฟ้มข้อมูล ลงในไดเรกทอรี INCLUDE

2) ส่วนของฟังก์ชัน IN,OUT (โปรแกรม EDL.C) เป็นส่วนที่ทำงานตามขั้นตอนการนำข้อมูลเข้า และ นำข้อมูลออก ตามรูปที่ 3.9 ถึง 3.11 คำสั่งสำคัญที่ใช้ในโปรแกรม คือ คำสั่ง outportb( ) และคำสั่ง inportb( )[20] คำสั่ง outportb( ) เป็นคำสั่งที่ทำการส่งข้อมูลขนาด 1 ไบต์ ไปยังพอร์ต I/O ซึ่งมีรูปแบบการใช้งานดังนี้

outportb(address No.,data 1 byte)

คำสั่ง inportb( ) เป็นคำสั่งการอ่านข้อมูลขนาด 1 ไบต์ จากพอร์ต I/O ซึ่งมีรูปแบบการใช้งานดังนี้

X = inportb(address No.);

ซึ่งจะส่งค่ากลับเป็นข้อมูลขนาด 1 ไบต์ ในที่นี้ข้อมูลจะถูกเก็บไว้ในตัวแปร X

รายละเอียดของโปรแกรม EDL.C มีดังนี้

```
#include <edl.h>

void outaddr(char addr)                    /* function for address output */
{
        outportb(porta,0x80);              /* out idle state */
        outportb(porta,0x0f&addr);         /* out low nibble */
        outportb(porta,0xb0);              /* out F state */
        outportb(porta,0x0f&(addr>>4)); /* out high nibble */
        outportb(porta,0xa0);              /* out G state */
        outportb(porta,0x80);              /* out idle state */
}

void out(char addr,char data)              /* function for out data to address port */
{
/*      outportb(portc,0x04);      */      /* turn on power of mainboard */
        outaddr(addr);                     /* out address */
        outportb(porta,0x0f&data);         /* out low nibble */
        outportb(porta,0xe0);              /* out C state */
        outportb(porta,0x0f&(data>>4));    /* out high nibble */
        outportb(porta,0xc0);              /* out D state */
        outportb(porta,0xe0);              /* out C state */
        outportb(porta,0xf0);              /* out E state */
```

```
        outportb(porta,0x80);                   /* out idle state */
}

char in(char addr)                      /* function for input data from address port */
{
        char low,high,temp;

        outaddr(addr);                  /* out address */
        outportb(porta,0x90);           /* out A state */
        outportb(porta,0xd0);           /* out B state */
        low = inportb(portb);           /* read low nibble */
        outportb(porta,0x80);           /* out idle state */
        high = inportb(portb);          /* read high nibble */
/*---------- bit manipulation ---------*/
        temp = (high)&0x80;
        high = ((high<<1)&0x70)|temp;
        temp = (low>>4)&0x08;
        low  = ((low>>3)&0x07)|temp;
        temp = high|low;
        return temp;
}
```


ในส่วนนี้จะถูกผนวกเข้าในไลบรารีของ Turbo C โดยมีขั้นตอนการสร้าง ดังนี้

2.1) ใช้โปรแกรม Editor ของ Turbo-C ช่วยในการเขียนโปรแกรม EDL.C

2.2) บันทึกแฟ้มข้อมูล ลงในไดเรกทอรี LIB

2.3) ทำการคอมไพล์โปรแกรม EDL.C โดยกดปุ่ม F9 จะได้ไฟล์ EDL.OBJ โดยก่อนทำการคอมไพล์ จะต้องตั้งตัวเลือกให้ Turbo C ส่งไฟล์ผลลัพธ์ ไปยังไดเรกทอรีชื่อ LIB โดยตั้งที่เมนู Options / Directories

2.4) ออกไปยัง DOS และเข้าสู่ไดเรกทอรี LIB ซึ่งถ้าตรวจดูไฟล์ข้อมูล จะพบไฟล์ดังนี้

```
Volume in drive C has no label
Volume Serial Number is 166D-14F5
Directory of C:\TC\LIB

.            <DIR>          12-29-95   1:14a
..           <DIR>          12-29-95   1:14a
C0C      OBJ         2,026 08-29-88   2:00a
C0M      OBJ         2,217 08-29-88   2:00a
MATHC    LIB        21,509 08-29-88   2:00a
MATHM    LIB        21,470 08-29-88   2:00a
CC       LIB       107,895 08-29-88   2:00a
CM       LIB       107,959 08-29-88   2:00a
C0S      OBJ         2,200 08-29-88   2:00a
C0T      OBJ         2,107 08-29-88   2:00a
C0L      OBJ         2,043 08-29-88   2:00a
MATHS    LIB        20,957 08-29-88   2:00a
MATHL    LIB        21,510 08-29-88   2:00a
CL       LIB       110,973 08-29-88   2:00a
EMU      LIB        15,384 08-29-88   2:00a
GRAPHICS LIB        29,247 08-29-88   2:00a
FP87     LIB         3,540 08-29-88   2:00a
C0H      OBJ         1,977 08-29-88   2:00a
MATHH    LIB        20,747 08-29-88   2:00a
CH       LIB       107,556 08-29-88   2:00a
INIT     OBJ         1,642 08-29-88   2:00a
EDL      OBJ         1,246 03-23-96   6:25p
CS       BAK        97,792 04-27-96  10:34p
CS       LIB        96,768 04-27-96  10:34p
      24 file(s)        798,765 bytes
                    206,012,416 bytes free
```

2.5) พิมพ์คำสั่ง

TLIB CS +EDL.OBJ

2.6) ทำซ้ำขั้นตอนในข้อ 2.5 แต่เปลี่ยน CS เป็น CL , CC , CM , CH [19]ในแต่ละครั้งจนครบทั้งหมด

สำหรับการใช้งานฟังก์ชัน IN, OUT จะมีรูปแบบ ดังนี้

IN(address) ทำการอ่านข้อมูลขนาด 1 ไบต์ จากพอร์ตหมายเลข address ของชุดฝึกทดลอง โดย address มีค่าตั้งแต่ 0-255 (ฐานสิบ) คือ สามารถอ้างอิงพอร์ตได้จำนวน 256 พอร์ต

ตัวอย่าง

```
char x ;
x = IN(0XC1) ;
```

OUT(address, value) ทำการส่งค่า value ขนาด 1 ไบต์ไปยังพอร์ตหมายเลข address ของชุดฝึกทดลอง address มีค่าตั้งแต่ 0-255

ตัวอย่าง

```
OUT(0XC0, 0XFF) ;
```

โดยในส่วนของ C preprocessor ของโปรแกรมจะต้องเพิ่มบรรทัด

```
#include <EDL.H>
```