ปริญญานิพนธ์ปีการศึกษา 2536

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การส่งข้อมูลผ่านสาย AC

ผู้จัดทำ

1. นางสาว กมลวรรณ ด่านคอนสกุล 33100004
2. นาย กฤษณ์ เพียรปรัชญาพร 33100011
3. นางสาว นิภาภณธ์ ศิริพล 33100179

(ดร. รัตติกร วรากูลศิริพันธุ์)

อาจารย์ที่ปรึกษา

การส่งสัญญาณผ่านสาย AC

DATA TRANSMISSION THROUGH AC LINE

โดย นางสาว กมลวรรณ ด่านคอนสกุล 33100004
นาย กฤษณ์ เพียรปรัชญาพร 33100011
นางสาว นิภาภณช์ ศิริพล 33100179

อาจารย์ที่ปรึกษา ดร. รัตติกร วรากูลศิริพันธุ์

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอการนำสายไฟฟ้ากระแสสลับ ซึ่งมีติดตั้งอยู่แล้วภายในอาคาร มาใช้เป็นสื่อกลางในการส่งสัญญาณควบคุมจากคอมพิวเตอร์ไปยังอุปกรณ์เป้าหมาย

โครงงานนี้ประกอบด้วย 2 ส่วนสำคัญ คือ ส่วนซอร์ฟแวร์และส่วนฮาร์ดแวร์ ในส่วนซอร์ฟแวร์จะเป็นส่วนของคำสั่งในการควบคุมการทำงาน และกำหนดอุปกรณ์เป้าหมาย ส่วนฮาร์ดแวร์ จะประกอบด้วยส่วนของการเข้ารหัส, การมอดูเลชั่นแบบ FM, การถอดรหัส, การดีมอดูเลเตอร์แบบ FM, การควบคุมอุปกรณ์ไฟฟ้า ทำงานร่วมกันเพื่อตอบสนองคำสั่งจากส่วนซอร์ฟแวร์

ABSTRACT

THIS THESIS IS THE USED AC-LINE SUMMISSION THAT WAS ESTABLISHED IN THE BUILDING. BY THE MEDIA ; AC-LINE : IS TRANSMITTED THE CONTROL SIGNAL FROM COMPUTER TO THE TARGET DEVICE.

THIS PROJECT CONSISTS OF THE SOFTWARE PART AND THE HARDWARE PART. FOR CONTROL COMMAND THAT INVOLED THE OPERATION AND ASSIGN THE TARGET DEVICE IS THE SOFTWARE PART, AND FOR HARDWARE PART INCLUDES MANY SUBPARTS :ENCODE, FM MODULATION, DECODE, FM DEMODULATOR, DEVICE CONTROL:WHICH ARE CO-OPERATING TO RESPONSE THE COMMAND FROM SOFTWARE PART.

สารบัญ

บทนำ

การสื่อสารเป็นการส่งข่าวสารจากผู้ส่งไปยังผู้รับ โดยมีจุดมุ่งหมายให้ผู้รับ รับรู้และตอบสนองต่อข่าวสารนั้น ๆ ซึ่งปัจจุบันการสื่อสารก้าวหน้าไปพร้อมกับความก้าวหน้าทางเทคโนโลยี

สำหรับการส่งข้อมูลผ่านสายไฟบ้านนั้น เป็นการนำสายไฟบ้านมาใช้ให้เกิดประโยชน์ โดยมุ่งเน้นเพื่อศึกษาลักษณะการทำงานพื้นฐาน และหาแนวทางปรับปรุงเพื่อใช้งานจริงได้อย่างมีประสิทธิภาพในอนาคต โดยลักษณะการทำงานจะประกอบด้วย การนำข้อมูลจากแหล่งข้อมูล คือ คอมพิวเตอร์ แล้วนำมาเข้ารหัสแปลงสัญญาณให้เหมาะสม แล้วส่งผ่านสายไฟบ้าน โดยทางด้านรับ เมื่อรับสัญญาณแล้ว จะตอบสนองโดยส่งผลกลับไปให้ด้านส่งรับรู้ว่าสามารถตอบสนองข่าวสารของทางด้านส่ง แล้วทำการส่งตอบ(หรือไม่ก็ได้)กลับไปด้านส่งเพื่อบอกให้ด้านส่งแน่ใจว่าส่งข่าวสารสำเร็จ

ทฤษฎีทั่วไปที่ใช้ในโครงงานนี้

คลื่นพาห์ (carrier wave) คือคลื่นที่มีขนาดและความถี่คงที่ ใช้ในการมอดูเลตสัญญาณ(signal) เช่น สัญญาณเสียง โดยตัวคลื่นพาห์เองนั้นจะไม่สร้างสัญญาณเสียงที่ตัวรับสัญญาณ

การมอดูเลชั่น (modulation) คือ การทำให้ลักษณะของคลื่นพาห์เปลี่ยนไปตามสัญญาณที่เราต้องการจะส่ง (modulating signal) การรวมสัญญาณกับคลื่นพาห์ จะได้สัญญาณใหม่เป็น modulated wave

กำหนดให้คลื่นพาห์มีสมการเป็น

$$e = E_m \sin(2\pi ft+\theta)$$

การมอดูเลชั่น แบ่งเป็น 3 วิธีใหญ่ ๆ คือ

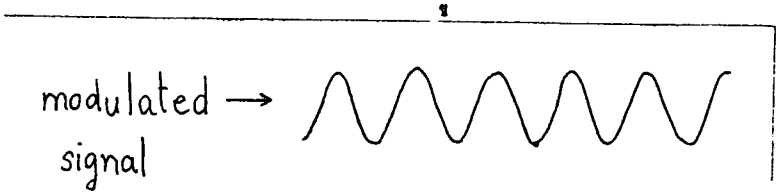
1. การมอดูเลชั่นเชิงขนาด (Amplitude modulation หรือ AM) เป็นการเปลี่ยนค่า E_m ตามสัญญาณ
2. การมอดูเลชั่นเชิงความถี่ (Frequency modulation หรือ FM) เป็นการเปลี่ยนค่า f ตามสัญญาณ
3. การมอดูเลชั่นเชิงเฟส (Phase modulation หรือ PM) มีลักษณะคล้ายกับ FM แต่จะเป็นการเปลี่ยนค่าเฟส ซึ่งมีความสัมพันธ์กับ θ

ในที่นี้จะพิจารณาเฉพาะ FM เท่านั้น

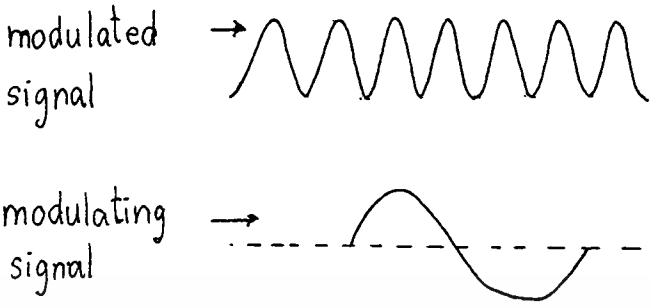
1. การมอดูเลชั่นเชิงความถี่ (FM)

1.1 ลักษณะทั่วไปของ FM

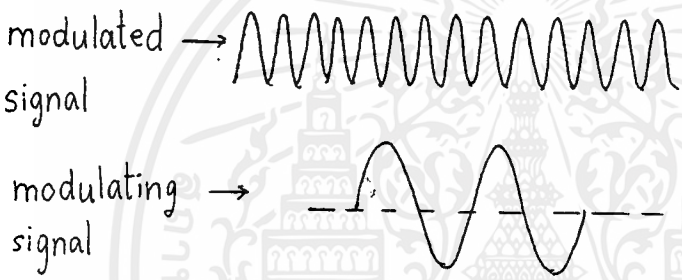
ความถี่ของคลื่นพาห์ที่ถูกมอดูเลตแล้ว จะแปรผันตามขนาดของสัญญาณที่นำมามอดูเลต (modulating signal) ดังรูป



a) คลื่นพาห์เมื่อยังไม่มีสัญญาณมามอดูเลต



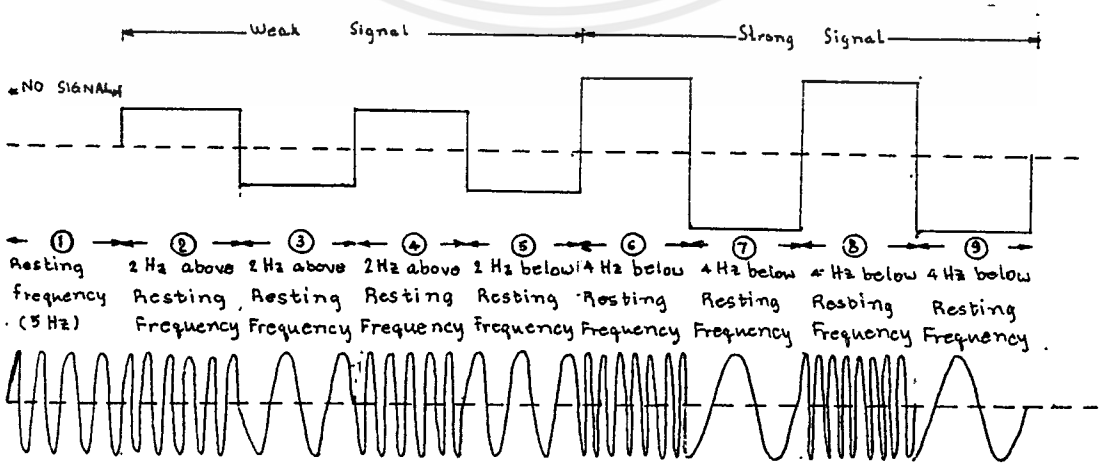
b) เมื่อมีสัญญาณความถี่ 500 Hz มามอดูเลต



c) เมื่อมีสัญญาณความถี่ 1000 Hz มามอดูเลต

จากรูป จะเห็นได้ว่าเมื่อขนาดแรงดันของสัญญาณที่นำมามอดูเลตเพิ่มขึ้น ความถี่ของสัญญาณที่ได้จากการมอดูเลตจะเพิ่มขึ้นตาม

ใน (b) และ (c) แสดงการเบี่ยงเบนของความถี่ตัวที่ขนาดของสัญญาณอัตราส่วนของการเบี่ยงเบนความถี่พิจารณาจากความถี่ของสัญญาณ



รูปที่ 1.1 แสดงคลื่นพาห์ซึ่งถูกมอดูเลตแบบความถี่โดยสัญญาณคลื่นสี่เหลี่ยม

รูปที่ 1.1 แสดงผลของการมอดูเลตทางความถี่ เมื่อขนาดของสัญญาณที่นำมามอดูเลต ต่างกัน คลื่นสัญญาณที่ได้จากการมอดูเลตจะมีความถี่ต่างกันด้วย

รูป 1.1 พิจารณาสัญญาณที่ได้จากการมอดดูเลต ในช่วงที่ 1 ไม่มีสัญญาณมามอดูเลต ดังนั้นสัญญาณที่ได้จากการมอดูเลตจึงมีความถี่เดียวกับคลื่นพาห์ เรียกความถี่นี้ว่า Resting Frequency ซึ่งในที่นี้เท่ากับ 5 Hz

ในช่วงที่ 6, 7, 8 ,9 ความถี่ของสัญญาณที่ผ่านการมอดูเลตที่ได้จะมากกว่า หรือน้อยกว่า Resting Frequency อยู่ 4 Hz

ในช่วง 2, 3, 4, 5 ความถี่ของคลื่นที่ได้จากการมอดูเลตที่ได้จะมากกว่า หรือน้อยกว่า Resting Frequency อยู่ 2 Hz

1.2 <u>ไซด์แบนด์ของ FM</u>

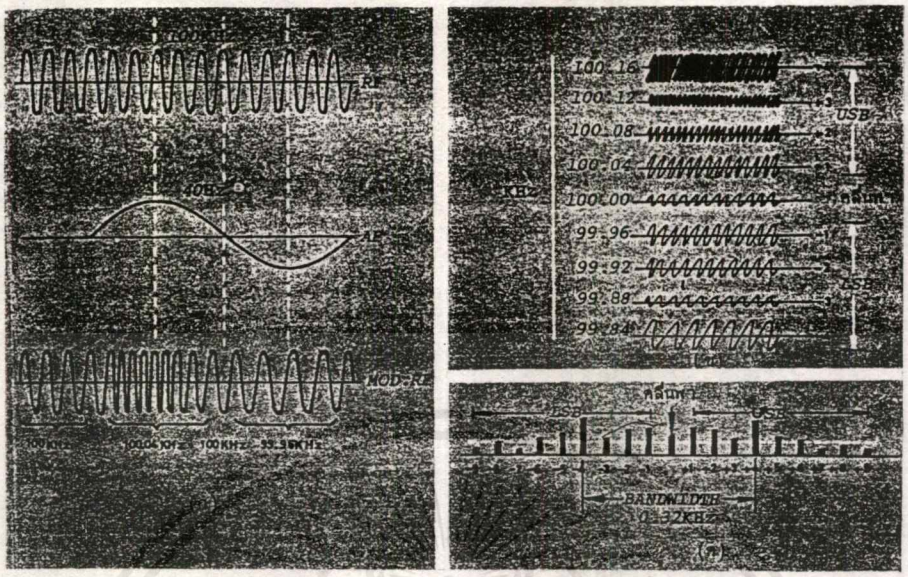
(ก) จำนวนไซด์แบนด์ของ FM มีไม่จำกัด ในการผสมแบบ FM จำนวนไซด์แบนด์ที่เกิดขึ้นมีจำนวนไม่จำกัด เช่นนำ 40 Hz ผสมกับ 100 Hz แบบ FM ไซด์แบนด์จะมี 1,2,3,4, ฯลฯ มากมายออกไปทั้งสองข้างของคลื่นพาห์

รูปที่ 1.2.1 เป็นส่วนหนึ่งของภาพการผสม FM

การผสมคลื่นแบบ FM ในรูป ก.นั้น ทำให้เกิดไซด์แบนด์ออกไปทั้งสองข้างเป็นคู่ ๆ คือ $\pm1, \pm2, \pm3, \pm4, \ldots$ และยังมีคู่อื่น ๆ อีกต่อไปไม่สิ้นสุด

เมื่อสังเกตค่าของไซด์แบนด์จะพบว่า

คู่ที่ 1 มีแถบกว้าง = 0.08 kHz อยู่ระหว่างความถี่ 99.96 - 100.04 kHz
คู่ที่ 2 มีแถบกว้าง = 0.16 kHz อยู่ระหว่างความถี่ 99.92 - 100.08 kHz
คู่ที่ 3 มีแถบกว้าง = 0.24 kHz อยู่ระหว่างความถี่ 99.88 - 100.12 kHz
คู่ที่ 4 มีแถบกว้าง = 0.32 kHz อยู่ระหว่างความถี่ 99.84 - 100.16 kHz



รูปที่ 1.2.1 แสดงไซด์แบนด์ของการผสมแบบ FM

จำนวนไซด์แบนด์มีจำนวนไม่จำกัด รูปที่ 1.2.1 ข.นั้น แสดงไซด์แบนด์ที่สำคัญเท่านั้น

(ข) เหตุที่มีจำนวนไซด์แบนด์มีไม่จำกัดนั้น เพราะว่าการผสมแบบ FM นั้น ขึ้นอยู่กับปัจจัย 2 ประการ คือ

(1) แอมปลิจูดของคลื่น AF ที่นำมาผสม ถ้าแอมปลิจูดของคลื่น AF สูง ระยะของความถี่ที่กระจายออกไปจากคลื่นพาห์ก็ยิ่งกว้าง คือมีแบนด์วิดธ์กว้างออกไป

(2) ความถี่ของคลื่น AF ที่มาผสม ถ้าความถี่ของคลื่น AF ยิ่งยาว แบนด์วิดธ์ก็ยาวออกไป ถ้าความถี่สั้น แบนด์วิดธ์ก็จะสั้นตาม

(ค) ขนาดของคลื่นพาห์ขณะผสม คลื่นพาห์ของ FM จะมีแอมปลิจูดต่ำลงในขณะผสมในการผสมแบบ AM นั้น คลื่นพาห์จะมีแอมปลิจูดสูงใหญ่กว่าแอมปลิจูดของไซด์แบนด์ ส่วนการผสมแบบ FM นั้นยอดของคลื่นพาห์จะต่ำกว่าคลื่นของไซด์แบนด์ ดังแสดงในรูป 1.2.1 ข.

(ง) ขนาดของไซด์แบนด์ ไซด์แบนด์ของการผสมแบบ FM ไม่ได้มีขนาดค่อยเล็กลงตามลำดับอย่างไซด์แบนด์ของคลื่น AM ขนาดของไซด์แบนด์ของ FM สูง ๆ ต่ำ ๆ บางอันสูงกว่าคลื่นพาห์ ดังแสดงเป็นไดอะแกรมในรูปที่ 1.2.1 ค.

จะอย่างไรก็ตาม แม้ว่าไซด์แบนด์ของ FM จะมีขนาดกำลัง(POWER)มากกว่าคลื่นพาห์ของมันก็ตาม แต่ไซด์แบนด์ที่ห่างจากคลื่นพาห์ออกไปมาก ๆ กำลังหรือขนาดของไซด์แบนด์ก็จะลดลง ในที่สุดก็เป็นศูนย์ จงสังเกตในรูป 1.2.1 ค. นั้น ไซด์แบนด์ที่ 7,8,9, และอื่น ๆ ที่ไม่ได้เขียนไว้จะมีขนาดเล็กลง จนหมดความสำคัญ

(จ) แบนด์วิดธ์(BANWIDTH) ของ FM เนื่องจากจำนวนไซด์แบนด์ของการผสมมีจำนวนไม่จำกัด ตามที่กล่าวมาแล้ว ฉะนั้นความกว้างของไซด์แบนด์ก็คงมีจำนวนไม่จำกัดด้วยที่กล่าวนี้โดยทฤษฎี

ในหลักปฏิบัติแล้วก็มีจำนวนจำกัด จำนวนไซด์แบนด์ปลาย ๆ เป็นไซด์แบนด์ที่ไม่มีความสำคัญเพราะกำลังต่ำ จึงตัดออกไป ในรูปที่ 1.2.1 ค.นั้น เราเอาแค่ไซด์แบนด์ ± 4 เท่านั้น ถ้าเลย ± 4 ไปแล้วเราไม่ใช้ ฉะนั้นไซด์แบนด์ของการผสม FM ณ.ที่นี้ จึงเท่ากับ 0.32 kHz (คือ 100.16 - 99.84 = 0.32 kHz)

1.3 ดัชนีการผสมคลื่นและแบนด์วิดธ์

ดัชนีการผสมคลื่น หรือ MODULATION INDEX

สูตร ดัชนีการผสมคลื่น FM $=$ $\dfrac{\text{ความถี่เบี่ยงเบน}}{\text{ความถี่ของสัญญาณที่นำมามอดูเลต}}$

$$= \frac{\text{deviation frequency}}{\text{AF frequency}}$$

$$\text{MOD. Index} = m_f = \frac{\Delta f}{f_m}$$

ความถี่เบี่ยงเบน (deviation frequency) Δw เรเดียนต่อวินาที ค่าของ m_f ซึ่งเท่ากับ $\Delta w/w_m$ ถ้าค่าของ $\Delta w << w_m$ คือประมาณว่า $m_f < 0.5$ แบนด์วิดธ์จะเท่ากับ $2w_m$ ซึ่งในกรณีนี้ถือว่าเป็น FM ช่วงความถี่แคบ (narrowband FM) ในทางตรงข้าม ถ้า m_f มีค่าสูงมาก $\Delta w >> w_m$ เป็นกรณีของ FM ช่วงความถี่กว้าง(wideband FM) แบนด์วิดธ์จะเท่ากับ $2\Delta w$ หรือถ้าใช้กฎของคาร์สัน(Carson's rule) จะได้แบนด์วิดธ์ $B = 2(\Delta f + f_m)$

ตัวอย่าง ถ้าคลื่นพาห์ความถี่ 10 MHz ถูกมอดูเลตโดยคลื่นไซน์ได้ความถี่เบี่ยงเบน 50 kHz หาแบนด์วิดธ์ของสัญญาณ FM ถ้าความถี่ของสัญญาณที่นำมามอดูเลตเป็น

(a) 500 kHz (b) 500 Hz (c) 10 kHz

(a) $m_f = \dfrac{\Delta f}{f_m} = \dfrac{50k}{500k} = 0.10$ <-- เป็นสัญญาณ FM ช่วงความถี่แคบ

$B = 2f_m = 1$ MHz

(b) $m_f = 100$ <-- เป็นสัญญาณ FM ช่วงความถี่กว้าง

$B = 2\Delta f = 100$ kHz

ถ้าใช้กฎของคาร์สัน จะได้ $B = 2(\Delta f + f_m) = 101$ kHz

(c) $m_f = 5$

ใช้กฎของคาร์สัน จะได้ $B = 2(\Delta f + f_m) = 120$ kHz

หรือถ้าหาแบนด์วิดธ์จากจำนวนไซด์แบนด์ก็จะได้

$B = 2nf_m = 2(8)(10\text{ kHz}) = 160$ kHz

B เป็นแบนด์วิดธ์ต่ำสุดที่จำเป็นในการสื่อสาร ถ้ามีแบนด์วิดธ์มากกว่านี้ สัญญาณรบกวนจะรบกวนการส่งข้อมูลได้มากขึ้น ดังนั้นจึงต้องให้แบนด์วิดธ์แต่ละช่องสัญญาณเล็กที่สุดเท่าที่จะทำได้

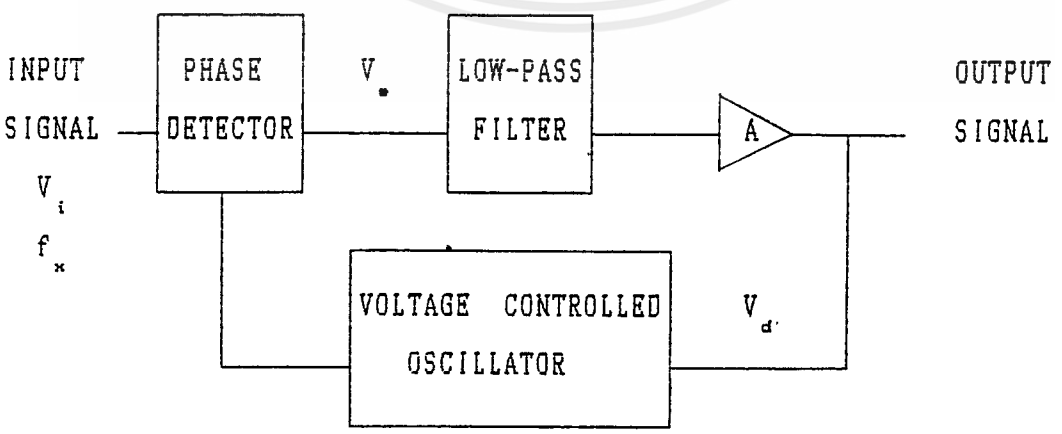
ไซด์แบนด์สำคัญนั้นกำหนดความสูง ต้องมีค่าตั้งแต่ 1% ของความสูงคลื่นพาห์เมื่อยังไม่ผสม(UNMODULATAED CARRIER) เช่น ดัชนี 5 มีจำนวนไซด์แบนด์ทั้งสองข้างไม่จำกัด แต่มีจำนวนไซด์แบนด์ที่สูงเกิน 1% ของความสูงคลื่นพาห์ที่ยังไม่ผสมเพียงข้างละ 8 อัน

2. เฟสล็อคลูป (PHASE LOCKED LOOPS หรือ PLL)

เป็นเทคนิคการควบคุมความถี่ของวงจรอิเลคทรอนิคส์ ซึ่งการใช้งานของเฟสล็อคลูปจำเป็นต้องกำหนดช่วงความถี่ใช้งาน

เฟสล็อคลูปเป็นระบบที่มีการป้อนสัญญาณกลับ (Feedback)

บล็อคไดอะแกรมของเฟสล็อคลูปเป็นดังรูปที่ 2.0



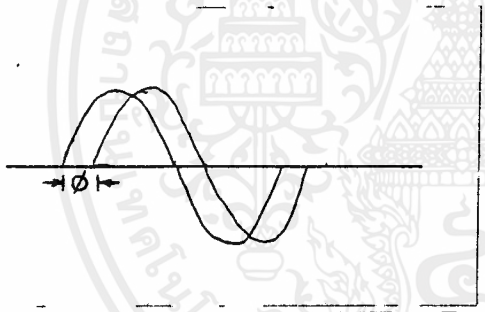
รูปที่ 2.0 บล็อคไดอะแกรมของเฟสล็อคลูป

วงจรเฟสล็อคลูป (PLL) มีประกอบด้วยส่วนสำคัญดังนี้

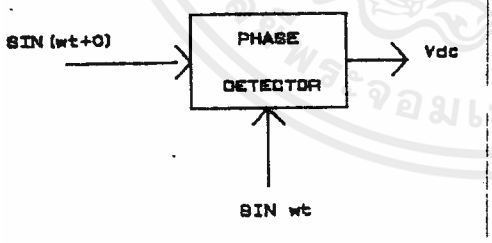
1. วงจรเปรียบเทียบเฟส (phase Detector หรือ Phase Comparator)
2. วงจรกรองผ่านความถี่ต่ำ (Low -Pass Filter)
3. วงจรแรงดันควบคุมความถี่(Voltage Controlled Oscillator)
4. วงจรขยาย (Amplifier) (จะมีหรือไม่ก็ได้)

2.1 วงจรเปรียบเทียบเฟส(Phase Detector หรือ Phase Comparator)

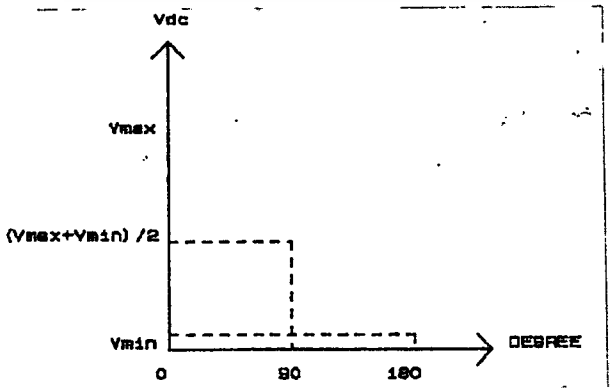
มีลักษณะเป็นมิกเซอร์ซึ่งมี 2 สัญญาณอินพุท และ 1 สัญญาณเอาท์พุท เป็นวงจรที่ใช้ในการเปรียบเทียบเฟสของสัญญาณอินพุททั้งสอง ถ้าสัญญาณอินพุททั้งสองมีความถี่เท่ากัน ความแตกต่างของความถี่จะเท่ากับ 0 Hz ทำให้ได้เอาท์พุทของวงจรเปรียบเทียบแรงดันเป็นแรงดันไฟตรงค่าหนึ่ง ซึ่งขนาดของแรงดันไฟตรงขึ้นอยู่กับมุมเฟส ø ระหว่างสัญญาณอินพุททั้งสอง



2.1.1a) มุมเฟสระหว่างสัญญาณ



2.1.1b) วงจรเปรียบเทียบเฟส



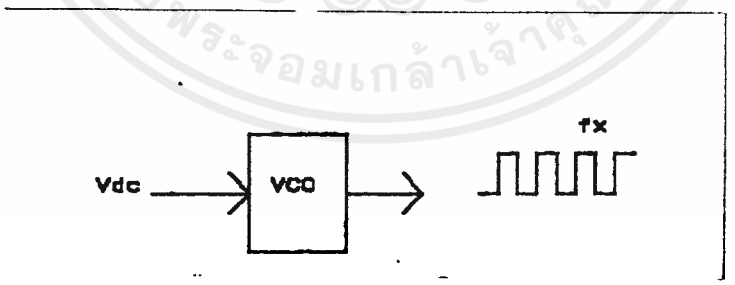
2.1.1c) เอาท์พุทของวงจรเปรียบเทียบเฟส

จากรูป 2.1.1c) เมื่อมุมเฟสเพิ่มจาก 0 - 180 องศา ค่าแรงดันไฟตรงจะมีค่าลดลงเรื่อย ๆ จากสูงสุด ถึง ต่ำสุด และที่ • = 90 องศา แรงดันไฟตรงที่เอาท์พุท จะเป็นค่าเฉลี่ยของค่าสูงสุดและค่าต่ำสุด เช่น สมมติว่าเอาท์พุทของวงจรเปรียบเทียบเฟสมีค่าสูงสุด 10 V และต่ำสุด 5 V เมื่ออินพุททั้งสองมีเฟสตรงกัน จะได้แรงดันไฟตรงที่เอาท์พุท เท่ากับ 10 V เมื่ออินพุททั้งสองมีเฟสต่างกัน 90 องศา แรงดันไฟตรงที่เอาท์พุทจะเท่ากับ 7.5V และเมื่ออินพุททั้งสองมีเฟสต่างกัน 180 องศา แรงดันไฟตรงที่เอาท์พุทจะเท่ากับ 5 V เป็นต้น

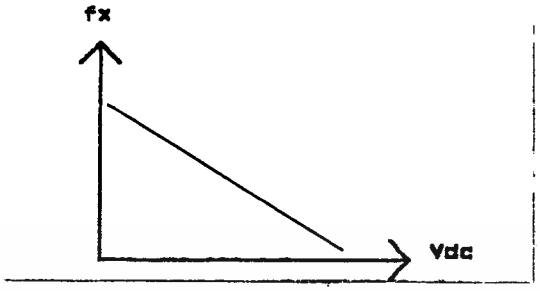
2.2 วงจรแรงดันควบคุมความถี่ (VOLTAGE CONTROLLED OSCILLATOR)

วงจรแรงดันควบคุมความถี่ หรือ VCO คือ วงจรสร้างความถี่ โดยความถี่ที่สร้างขึ้นนี้ถูกควบคุมโดยแรงดันไฟตรง การทำงานของ VCO จะมีความสัมพันธ์เป็นเชิงเส้นระหว่างความถี่ออสซิลเลตกับแรงดันควบคุม(control voltage) ซึ่งเป็นแรงดันไฟตรง

แรงดันไฟตรงใช้ในการควบคุมการออสซิลเลตความถี่เอาท์พุท ถ้าแรงดันไฟตรงเพิ่มขึ้น จะทำให้ VCO ออสซิลเลตความถี่ต่ำลง ในทางตรงกันข้าม ถ้าแรงดันไฟตรงลดลง VCO ก็จะออสซิลเลตความถี่สูงขึ้น จะเห็นว่าความถี่ที่ VCO ออสซิลเลตขึ้นจะลดลงแบบเชิงเส้นกับการเพิ่มของแรงดันไฟตรง ดังรูป 2.2.1 (b)

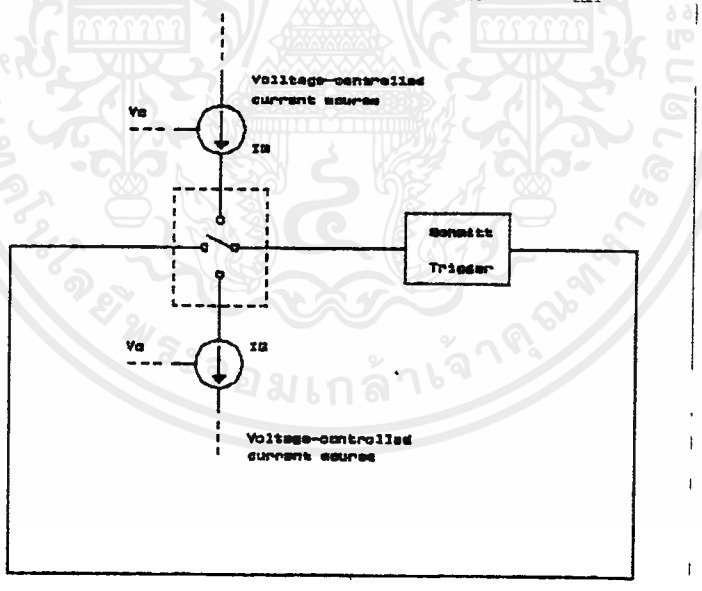


รูปที่ 2.2.1 (a) แสดงอินพุทที่เป็นแรงดันไฟตรงควบคุมเอาท์พุทของ VCO

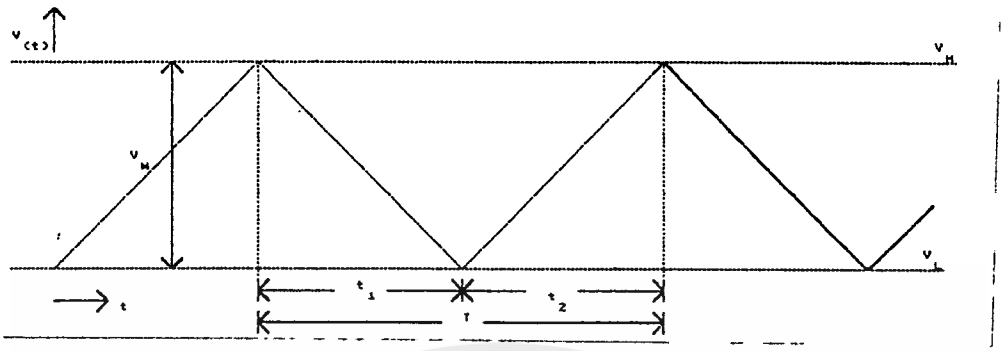


รูปที่ 2.2.1 (b) ความถี่ที่ VCO ออสซิลเลตขึ้น แปรผกผันกับแรงดันไฟตรงอินพุท

ลักษณะของ VCO มีความสัมพันธ์เชิงเส้นระหว่างความถี่ออสซิลเลตกับแรงดันควบคุม(control voltage)



รูปที่ 2.2.2 วงจรแรงดันควบคุมความถี่



รูปที่ 2.2.3 ลักษณะของรูปคลื่นที่คร่อมคาปาซิเตอร์

จากวงจรแรงดันควบคุมความถี่ ซึ่งมักใช้ในไอซี แรงดันไฟฟ้ากระแสตรงที่ออกจากฟิลเตอร์ จะควบคุมแหล่งจ่ายกระแสไฟฟ้า เพื่อใช้ในการชาร์จ และดิสชาร์จ ให้กับ timing capacitor (C_t) ช่วงเวลาของการชาร์จและดิสชาร์จจะไปควบคุมการทำงานของวงจรชมิททริกเกอร์ ซึ่งแรงดันตกคร่อม C_t ก็คือแรงดันอินพุทของวงจรชมิททริกเกอร์

วงจรชมิททริกเกอร์จะมีแรงดันเทรสโฮลด์ 2 ระดับ คือ V_L และ V_H ดังนั้น Schmitt trigger hysteresis V_w มีค่าเป็น

$$V_w = V_H - V_L$$

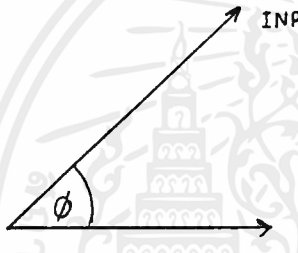
เนื่องจาก C_t ถูกชาร์จและดิสชาร์จ โดยแหล่งจ่ายกระแสไฟฟ้า และศักดาไฟฟ้าตกคร่อม C_t มีความสัมพันธ์เป็นเชิงเส้นกับเวลา ถ้ากระแสของการชาร์จและดิสชาร์จมีขนาดเท่ากัน รูปแบบของศักดาไฟฟ้าที่ตกคร่อม C_t จะเป็นคลื่นสามเหลี่ยมสมมาตร C_t จะถูกชาร์จจากแหล่งจ่ายกระแสระดับสูงจนได้ศักดาไฟฟ้าเท่ากับ V_H ระดับทริก(triggering level) ที่จุดนี้วงจรชมิททริกเกอร์จะทำงาน และเป็นเหตุให้แหล่งจ่ายกระแสถูกเปิดวงจร เพื่อแหล่งจ่ายกระแสระดับสูงไม่ต่อเชื่อมกับ C_t และ แหล่งจ่ายกระแสระดับต่ำจะเชื่อมต่ออยู่กับ C_t ซึ่งขณะนี้ C_t จะดิสชาร์จจนกระทั่งถึงระดับทริกล่าง(lower trigger) V_L หลังจากนั้นการทำงานก็จะวนสลับอย่างนี้ไปเรื่อย ๆ

2.3 การทำงานของ PHASE LOCKED LOOPS

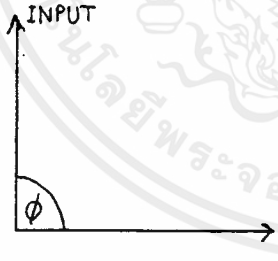
จากรูป 2.0 ซึ่งเป็น บล็อคไดอะแกรมของ PLL กำหนดให้สัญญาณอินพุท V_i มีความถี่เป็น f_x เป็นสัญญาณที่เข้าที่วงจรเปรียบเทียบเฟส และอินพุทที่เข้าที่วงจรเปรียบเทียบเฟส

อีกสัญญาณหนึ่งมาจาก VCO วงจรเปรียบเทียบเฟสทำการเปรียบเทียบเฟส และ ความถี่ของสัญญาณทั้งสองได้เอาท์พุทออกมาเป็นแรงดันผิดพลาด (error voltage) V_e ซึ่งมีความสัมพันธ์กับความแตกต่างของเฟสและความถี่ของอินพุททั้งสอง แรงดันผิดพลาดที่ได้จะผ่านวงจรกรองผ่านความถี่ต่ำและวงจรขยายเพื่อไปควบคุม VCO แรงดันที่ไปควบคุม VCO (V_d) จะไปเปลี่ยนความถี่เอาท์พุทของ VCO ในทิศทางที่ลดความแตกต่างของความถี่ระหว่าง f_x และ อินพุท

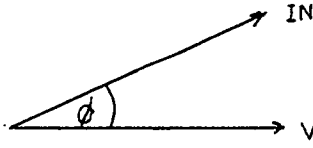
ถ้าความถี่ของ VCO มีค่าเท่ากับความถี่ของอินพุท แต่มุมเฟสระหว่างอินพุททั้งสองต่างกัน เมื่อพิจารณาค่าแตกต่างของเฟส จะพบว่า DC outputที่ได้จะมีค่าแตกต่างกัน



.3.1a) เฟสเซอร์ไดอะแกรม



2.3.1b) เมื่อ ความถี่อินพุทเพิ่ม มุมเฟสจะเพิ่ม



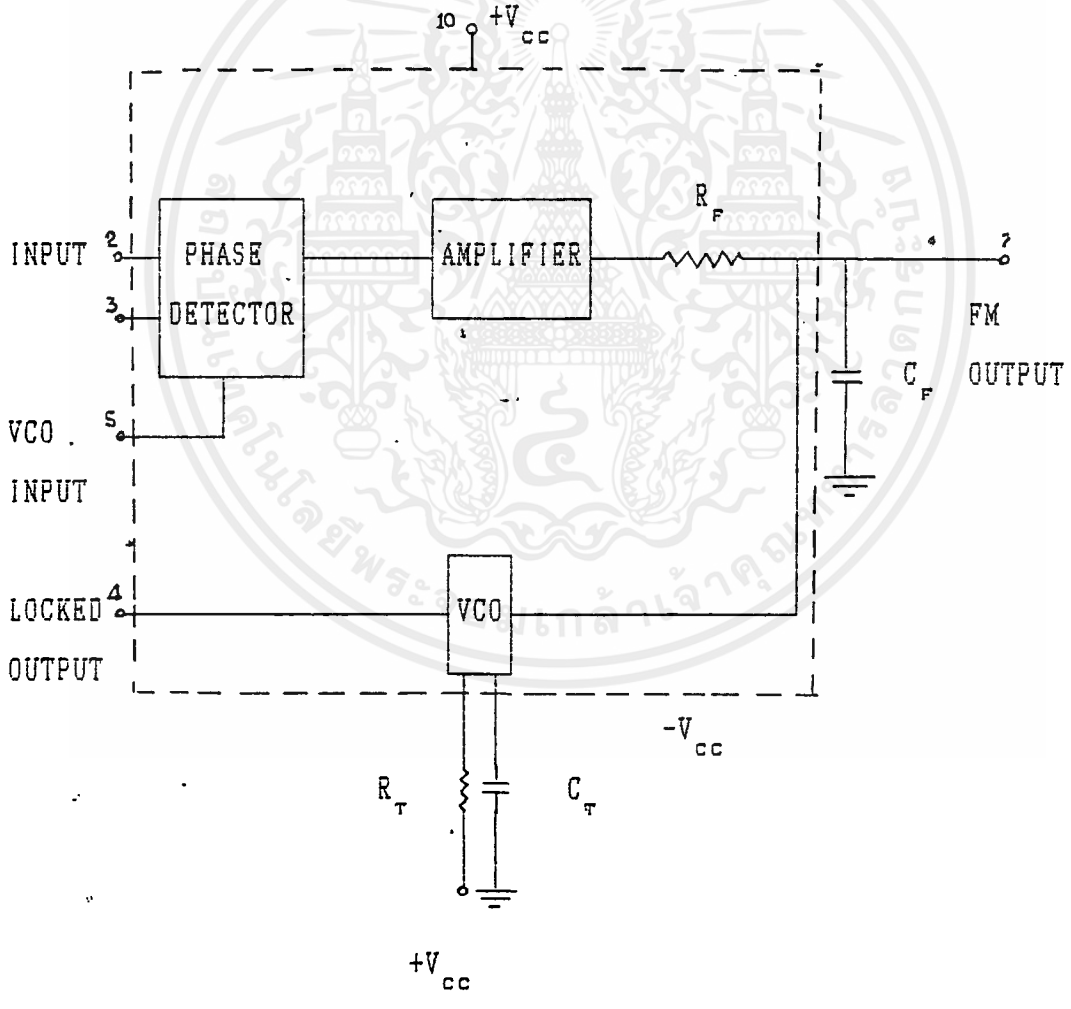
2.3.1c) เมื่อความถี่อินพุทลดลง มุมเฟส จะลดลง

ถ้ามีการเปลี่ยนแปลงความถี่อินพุทจะทำให้ ความถี่ VCO เปลี่ยนแปลงไป เช่น ถ้าความถี่อินพุทมีค่าเพิ่มขึ้น เฟสเซอร์ของมันจะหมุนตัวเร็วขึ้น ดังรูป 2.3.1b) ซึ่งหมาย

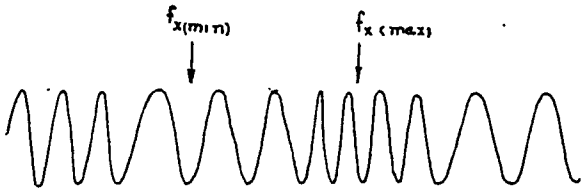
ความว่า ที่แรงดันไฟตรงมีค่าต่ำ จะทำให้ความถี่ VCO มีค่าเพิ่มขึ้นจนกระทั่งมีค่าเท่ากับความถี่อินพุท f_x

ในทางกลับกัน ถ้าความถี่อินพุทลดลง เฟสเชอร์จะแคบลง และมุมเฟส มีค่าลดลงดังรูป 2.3.1c) ซึ่งหมายถึงว่า เมื่อแรงดันไฟตรงมีค่ามากขึ้น จะทำให้ความถี่ VCO ลดลงจนกระทั่งมีค่าความถี่เท่ากับความถี่อินพุท f_x

3. ดีมอดเลชั่นของ FM (FREQUENCY DEMODULATION)



รูปที่ 3.1 บล็อคไดอะแกรมของ LM565



รูปที่ 3.2 สัญญาณอินพุท FM

ให้สัญญาณ FM เป็นอินพุทของ PLL และสัญญาณออกจาก VCO กับสัญญาณอินพุทมีความถี่เดียวกันแล้ว จะได้สัญญาณเอาท์พุทเป็นสัญญาณที่มอดูเลตมา หรือเอาท์พุทของการดีมอดูเลตแบบ FM สำหรับ LM 565 เป็นไอซีที่สามารถใช้งานในลักษณะ PLL ดังแสดงในบล็อคไดอะแกรม

ขา 2 และ ขา 3 เป็นอินพุทที่แตกต่างกัน(differential input) ให้กับ PLL โดยที่ขา 3 จะต่ออยู่กับกราวนด์ และให้สัญญาณอินพุทที่ขา 2

ขา 4 และขา 5 จะต่อถึงกัน เพื่อนำเอาท์พุทของ VCO มาป้อนเป็นอินพุทให้กับวงจรเปรียบเทียบเฟส

ขา 8 เป็น external timing resistor

ขา 9 เป็น external timing capacitor

C_T และ R_T เป็นตัวกำหนดความถี่ให้ตรงกับความถี่ของคลื่นพาห์ภาคส่ง

โดยที่ $f = 0.3/(R_T C_T)$

ขา 7 เป็น เอาท์พุท FM คือสัญญาณที่ผ่านการ demodulation แล้ว ซึ่งเป็นสัญญาณเดียวกับสัญญาณที่มอดูเลตมา

ที่ขา 7 จะมี C_F ต่อระหว่างขา 7 กับ กราวนด์ ซึ่ง C_F และ R_F มาจากวงจรกรองผ่านความถี่ต่ำ ซึ่งจะให้ค่าความถี่คัทออฟ

$$f = 1/(2\pi R_F C_F)$$

SIGNAL TO NOISE RATIO

ในระบบสื่อสารสัญญาณที่ออกจากระบบ หรือ สัญญาณเอาท์พุท ประกอบด้วยศักดาไฟฟ้าหรือกระแสไฟฟ้าที่เราต้องการและไม่ต้องการ สำหรับศักดาหรือกระแสไฟฟ้าที่เราไม่ต้องการเรียกว่า สัญญาณรบกวน (noise) สำหรับภาครับในระบบสื่อสาร สัญญาณเอาท์พุทที่นำมาใช้งานจำเป็นอย่างยิ่งที่กำลังของสัญญาณเอาท์พุทต้องมีค่ามากกว่ากำลังของสัญญาณรบกวน เพื่อง่ายต่อการพิจารณา กำหนดให้อัตราส่วนของ กำลังของสัญญาณ(signal power) ต่อ กำลังของสัญญาณรบกวน(noise power) เรียกว่า signal-to-noise ratio (S/N)

$$\text{signal-to-noise ratio} = \frac{\text{signal power}}{\text{noise power}}$$

$$= 10 \log_{10} \frac{\text{(signal power)}}{\text{(noise power)}} \quad \text{dB}$$

ความต้องการของ S/N ขึ้นอยู่กับการใช้งานของแต่ละระบบ เช่น วงจรโทรศัพท์ต้องการประมาณ 35-40 dB

สัญญาณรบกวนในสาย

สัญญาณรบกวนที่ได้ที่เอาท์พุทของระบบสายในการสื่อสาร ประกอบด้วยสัญญาณรบกวนที่เกิดจากสายส่งเอง และสัญญาณรบกวนที่ถูกสร้างขึ้นในแต่ละภาคของวงจร การเกิดสัญญาณรบกวนในสายส่งเป็นผลรวมของ thermal agitation noise ในความต้านทานในสาย , สัญญาณรบกวนจากจุดเชื่อมต่อ, สัญญาณรบกวนจากการอินเตอร์เฟสระหว่างสายส่งกำลัง และครอสทอล์ค(crosstalk) จากคู่สายอื่น

สำหรับในสายส่งที่เป็นสาย AC จะเกิดแรงดันของความถี่ที่ไม่ต้องการขึ้น เหนี่ยวนำขึ้นใน L,C คัปปลิ้งระหว่างสาย

สัญญาณรบกวนในทรานซิสเตอร์

Thermal agitation noise เป็นสัญญาณรบกวนที่เกิดขึ้นในตัวไบโพล่าร์ทรานซิสเตอร์เอง แต่จะเกิดจากเบสเป็นส่วนใหญ่ Shot noise เป็น noise ที่เกิดขึ้นในทรานซิสเตอร์ซึ่งเกิดจากการเคลื่อนที่แบบสุ่ม(random) ของ โฮล และอิเลคตรอนข้ามฝั่งของแต่ละรอยต่อ P-N ดังนั้นรอยต่อ P-N ทั้งสองจึงเป็นตัวกำหนดสัญญาณรบกวนชนิดนี้เอง

Partition noise เมื่อกระแสอินพุทที่ไหลเข้าทรานซิสเตอร์ผ่านไปยังรอยต่อเบส-อิมิตเตอร์ จะเกิดการแบ่งเป็นกระแสเบส และกระแสคอลเลคเตอร์ ($I_E = I_B + I_C$) ซึ่งกระแสเหล่านี้จะเกิดการแกว่งแบบสุ่ม ทำให้เกิดสัญญาณรบกวนขึ้น

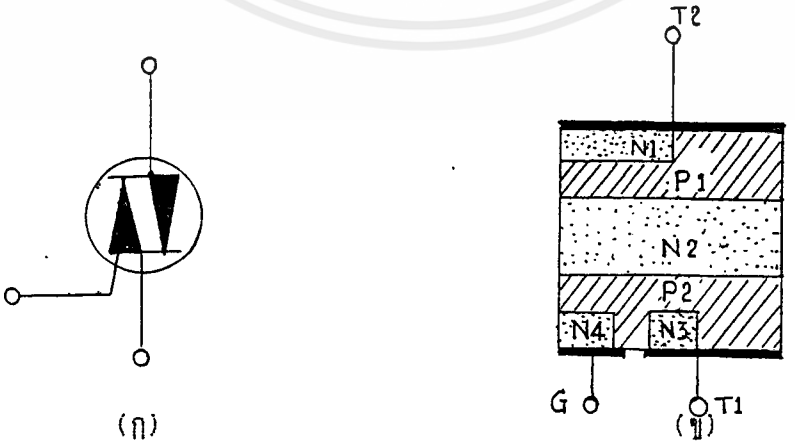
4. ไตรแอค(TRIAC หรือ BIDIRECTIONAL TRYRISTOR)

โครงสร้างพื้นฐานของไตรแอคแสดงได้ดังในรูปที่ 4.1ก ซึ่งจะเห็นว่ามีลักษณะคล้ายกับ SCR กล่าวคือ มีเกทหนึ่งเกท และอีกสองขั้วคือ T_1 และ T_2 ซึ่งกระแสหลักจะไหลผ่านไตรแอคแตกต่างจาก SCR ตรงที่ว่า ไตรแอคสามารถทำงานในภาวะ on-state ได้โดยการที่ขั้ว T_2 อาจมีศักย์เป็นบวกหรือลบก็ได้ และยิ่งไปกว่านี้ไบอัสที่เกทซึ่งจะกระตุ้นให้ไตรแอคทำงานนั้นก็อาจเป็นบวกหรือลบก็ได้เช่นกัน ดังนั้นเราจึงอาจเรียกไตรแอคนี้ว่า Bi-direction thyristor โดยทั่วไปไตรแอคจะทนกระแสและแรงดันได้น้อยกว่า SCR

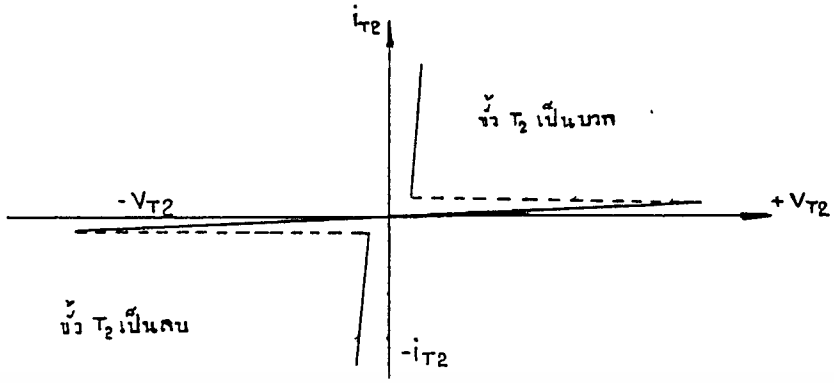
ลักษณะการทำงานของไตรแอคในขณะที่ได้รับไบอัสในแบบต่าง ๆ สรุปได้ดังแสดงในตารางที่ 1

ตารางที่ 1

Mode	ศักย์ของขั้ว T_2 เมื่อเทียบกับ T_1	ศักย์ของเกทเมื่อเทียบกับ T_1
I^+	บวก	บวก
I^-	บวก	ลบ
III^+	ลบ	บวก
III^-	ลบ	ลบ



รูป 4.1ก แสดงลักษณะโครงสร้างพื้นฐานของไตรแอคและสัญลักษณ์



รูป 4.2 แสดงคุณสมบัติความสัมพันธ์ระหว่างกระแสและแรงดันของไตรแอค

ลักษณะโครงสร้างของไตรแอคนี้เหมือนกับการนำเอา SCR 2 ตัว มาต่อขนานกันในลักษณะกลับขั้ว ส่วนขาเกทต่อร่วมเข้าด้วยกัน ดังนั้นไตรแอคจะทำหน้าที่เป็นตัวควบคุมระบบไฟได้ทั้งแบบไฟตรงและไฟสลับ นั่นคือความสามารถในการนำกระแสได้ทั้งสองทิศทาง โดยการทริกเกทนั้นก็สามารถกระทำได้ทั้งสองทิศทางเช่นกัน

<u>คุณสมบัติพื้นฐานของไตรแอคมีดังนี้</u>

1. โดยปกติถ้าไม่มีสัญญาณทริกที่ขาเกท ไตรแอคจะไม่ทำงาน โดยจะมีลักษณะเหมือนกับสวิตช์ที่ถูกเปิดวงจร

2. ถ้าในกรณีที่ MT_2 และ MT_1 ถูกป้อนด้วยแรงดันบวกและลบตามลำดับ ไตรแอคจะถูกกระตุ้นให้ทำงานได้โดยการป้อนสัญญาณพัลส์เพียงสั้น ๆ ที่ขาเกทของมัน ไตรแอคใช้เวลาเพียง $2 - 3*10^{-6}$ วินาทีเท่านั้นในการเริ่มทำงาน ในขณะที่ไตรแอคทำงานนั้นจะมีแรงดันคร่อมตัวมัน มีค่าประมาณ 1 หรือ 2 โวลท์เท่านั้น และก็เช่นกัน คือเมื่อไตรแอคเริ่มทำงานแล้ว ก็จะสามารถคงสภาพการทำงานอยู่เช่นนั้นต่อไปเรื่อย ๆ ตราบเท่าที่ยังมีกระแสไหลผ่านตัวมันอย่างต่อเนื่อง

3. หลังจากที่ไตรแอคคงสภาพการทำงานอยู่นั้น ทางเดียวที่จะหยุดการทำงานลงได้ก็โดยการลดปริมาณกระแสที่ไหลผ่านตัวมันลง ให้มีค่าต่ำกว่ากระแสโฮลดิ้งของมัน ในกรณีที่ใช้ไตรแอคในการจ่ายกระแส AC การหยุดทำงานจะเกิดขึ้นอย่างอัตโนมัติ เมื่อแรงดันของกระแสไฟสลับเข้าใกล้จุดตัดศูนย์ที่เกิดขึ้น ทุก ๆ ครึ่งคลื่น นั่นคือ กระแสจะลดลงเป็นศูนย์

4. ไตรแอคถูกกระตุ้นให้ทำงานได้ ทั้งสัญญาณแบบบวกและลบที่ป้อนให้แก่ขาเกท โดยไม่คำนึงถึงขั้วที่ต่ออยู่ที่ MT_1 และ MT_2 ดังนั้น การทำงานของไตรแอคนี้จะมีอยู่ 4 โหมด เมื่อเปรียบเทียบกับขั้วแรงดันที่ป้อนให้แก่ขาต่าง ๆ ของมัน ข้อแตกต่างกันเล็กน้อยของการทำงานในโหมดต่าง ๆ คือ ในกรณีของโหมดที่ขั้วแรงดันที่ให้แก่ขา MT_2 และเกทเหมือนกัน (ทั้งบวกและลบ) จะทำให้มีค่าความไวเกทสูงขึ้น

5. ไตรแอคสามารถทนการกระชากของกระแสได้สูง เช่น โดยปกติสำหรับไตรแอคที่ทน

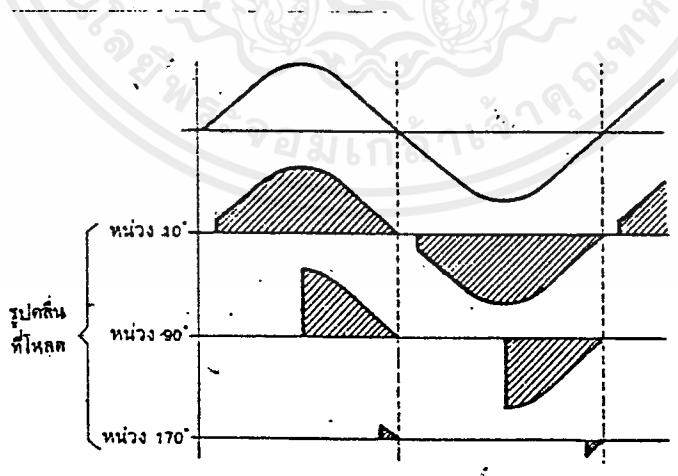
กระแสปกติได้ 10 แอมแปร์ (rms) สามารถทนการกระชากของกระแสในช่วงหนึ่งคาบเวลาของไฟ 60 Hz ได้สูงถึง 100 แอมแปร์ เป็นต้น

<u>การควบคุมกำลังไฟแบบเฟสทริกเกอร์</u>

การใช้ไตรแอคในวงจรหรี่ความสว่างของหลอดไฟ หรือวงจรควบคุมความเร็วของมอเตอร์ เป็นการใช้งานในลักษณะของการควบคุมกำลังไฟที่จ่ายให้แก่โหลดในระบบที่เรียกว่าเฟส-ทริกเกอร์

หลักการของวงจรที่มีลักษณะเป็นเฟส-ทริกเกอร์นั้นใช้ไตรแอคเป็นตัวควบคุมกำลังไฟที่จ่ายให้แก่โหลด โดยแทนที่จะทริกขาเกทด้วยสัญญาณไฟตรงนั้นตรง ๆ ก็ทริกโดยมีการหน่วงของเฟสด้วยวงจรอีกส่วนหนึ่ง

การหน่วงเฟสมีผลดังนี้ คือ ถ้าไตรแอคถูกทริกที่ตำแหน่งเฟส 10 องศา หลังจากที่ทุก ๆ ครึ่งรูปคลื่นเริ่มเข้ามา กำลังไฟเกือบทั้งหมดก็จะถูกป้อนให้แก่โหลด แต่ถ้าทำการทริกที่ตำแหน่งเฟส 90 องศา หลังจากทุก ๆ ครึ่งคลื่นเริ่มเข้ามา จะทำให้กำลังไฟที่ป้อนให้แก่โหลดนั้นลดลงเหลือเพียงครึ่งหนึ่งของกำลังทั้งหมด และถ้าไปทริกที่ตำแหน่งเฟส 170 องศา หลังจากที่ทุก ๆ ครึ่งรูปคลื่นเข้ามาแล้ว จะมีกำลังไฟส่วนน้อยเท่านั้นที่ป้อนให้แก่โหลด ดังแสดงในรูป 4.3



รูป 4.3 การเปลี่ยนแปลงค่าของกำลังไฟฟ้าที่ป้อนให้แก่โหลด โดยกำหนดได้จากตำแหน่งเวลาของการทริกที่ให้แก่ไตรแอค

5. การสื่อสารข้อมูล

หากพิจารณาตามทิศทางการส่งข้อมูลภายในสายส่ง สามารถแบ่งการส่งข้อมูลออกได้เป็น 3 ชนิดคือ

1. การส่งแบบทิศทางเดียว (one way transmission หรือ simplex transmission) คือ ข้อมูลจะไหลได้ทิศทางเดียว

2. การส่งแบบทิศทางใดทิศทางหนึ่ง (either-way transmission หรือ half duplex transmission) คือข้อมูลจะไหลได้สองทิศทาง แต่ต้องผลัดกันรับผลัดกันส่ง

3. การส่งแบบสองทิศทาง (both-way transmission หรือ full-duplex transmission) คือข้อมูลจะไหลได้สองทิศทางพร้อมกัน

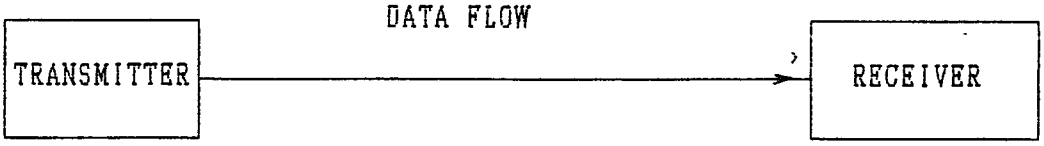
และถ้าพิจารณารูปแบบของข้อมูลที่ส่ง จะแยกได้เป็น 2 ชนิดคือ

1. แบบซิงโครนัส (synchronous trasmission)

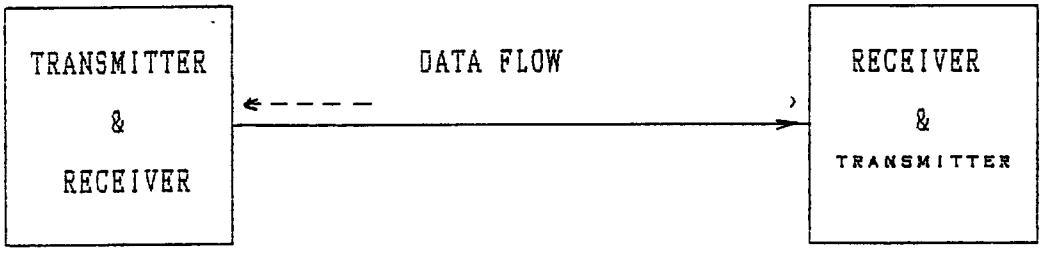
2. แบบอะซิงโครนัส (asynchronous transmission)

TRANSMISSION MODE

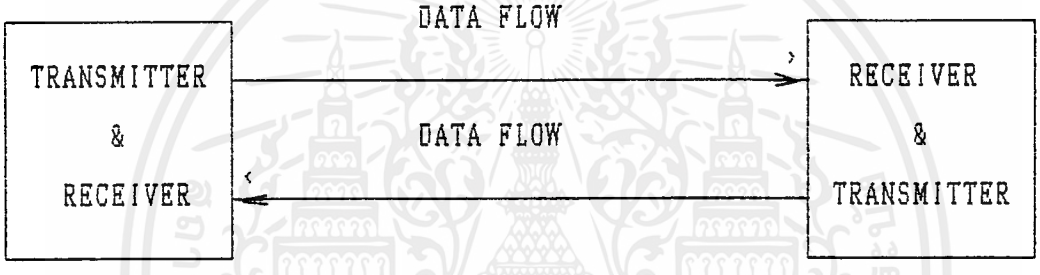
Mode	Number of Physical Wires	transmission Direction	simultaneous Transmission
Simplex	Two	One	No
Half Duplex	Two	Both	No
Four Wire HAlf Duplex	Four	Both	No
Full Duplex or	Four	Both	Yes



SIMPLEX CONFIGURATION



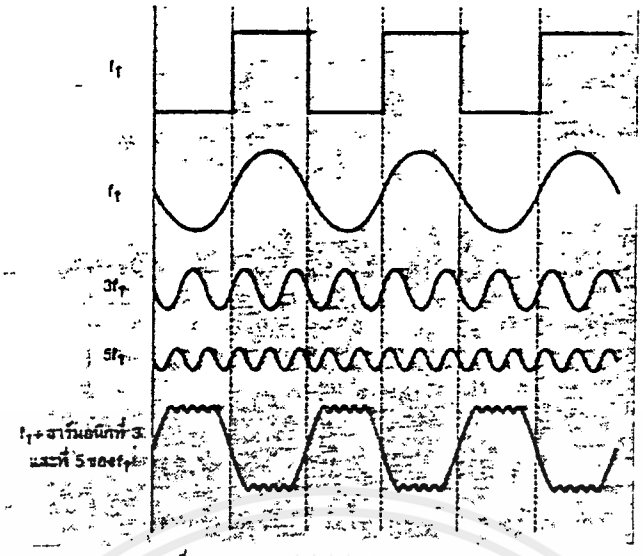
HALF-DUPLEX CONFIGURATION



FULL-DUPLEX CONFIGURATION

โปรโตคอล(Protocol) คือกฎข้อบังคับที่กำหนดรูปแบบและขั้นตอนการทำงานของฮาร์ดแวร์(hardware) และซอฟแวร์(software) เพื่อให้ทราบว่าจะเกิดข้อผิดพลาดในการรับส่งข้อมูลขึ้นเมื่อใด และในบางโปรโตคอลยังสามารถแก้ไขข้อผิดพลาดเหล่านั้นได้ด้วย

ในการรับส่งข้อมลแบบดิจิตอลผ่านไปตามสายส่งจนถึงปลายทางได้อย่างสะดวกและได้ระยะทางไกลนั้นอาจต้องใช้การมอดูเลชั่นเข้าช่วย ทั้งนี้เพราะข้อมูลแบบดิจิตอลที่ความถี่หนึ่งเป็นสัญญาณที่รวมผลของสัญญาณอนาลอกที่ความถี่นั้น ๆ รวมกับฮาร์โมนิกคี่ (odd harmonic) ของความถี่นั้น ๆ ซึ่งจะเป็นผลรวมของหลาย ๆ ความถี่ ทำให้เห็นเป็นรูปสัญญาณดิจิตอลที่เป็นเหลี่ยมมากขึ้น ดังรูปที่ 5.1



รูปที่ 5.1 สัญญาณดิจิตอลเป็นผลรวมของสัญญาณอนาลอกหลาย ๆ ความถี่

การส่งข้อมูลแบบดิจิตอลจะต้องมีแบนด์วิดธ์(bandwidth)กว้างมาก เช่น ถ้าจะส่งข้อมูลดิจิตอลความถี่ 6 kHz ความถี่นั้นจะต้องมีแบนด์วิดธ์อย่างน้อย (5*6kHz - 6kHz) เท่ากับ 24 kHz คือสามารถรองรับได้ถึงฮาร์โมนิกที่ 5 เป็นอย่างน้อย ในการใช้มอดูเลชั่นนั้น สัญญาณดิจิตอล "0" จะถูกเปลี่ยนเป็นความถี่อนาลอกค่าหนึ่ง และ "1" จะเปลี่ยนเป็นความถี่อนาลอกอีกค่าหนึ่ง เช่น 1,000/1,050 Hz แล้วทำการส่งไปตามสาย โดยให้ดีมอดูเลเตอร์ถอดรหัสที่ปลายทางกลับมาเป็นสัญญาดิจิตอลดังเดิม จะเห็นว่าความถี่ที่ใช้มีแบนด์วิดธ์เพียงแค่ 1,050-1,000 = 50 Hz ก็สามารถรับส่งกันได้ดีและได้ระยะทางไกลด้วย ซึ่งสัญญาณอนาลอกทั้งสองความถี่นี้จะไม่สูญหายง่าย ๆ เหมือนกับการส่งสัญญาณดิจิตอลโดยตรง เพราะยิ่งฮาร์โมนิกสูงมากขึ้นขนาดของสัญญาณอนาลอกจะยิ่งลดลง ทำให้ส่งได้ในระยะทางจำกัด

<u>วงจรสมมูลของสายส่ง</u>

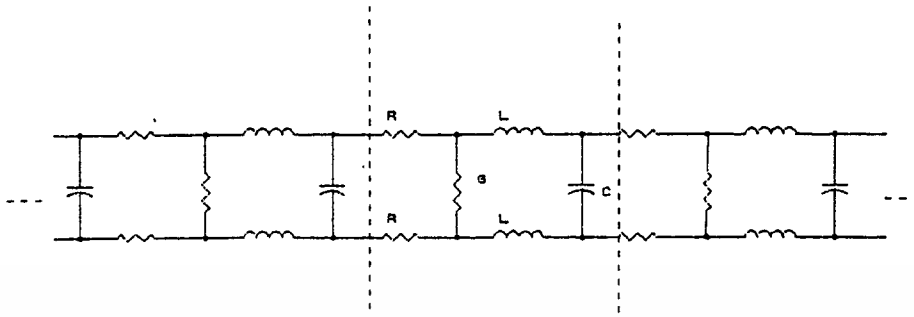
อุปกรณ์สายส่งทุกชนิดสามารถแทนด้วยวงจรในรูปที่ 7.2 ซึ่งใช้ในการวิเคราะห์คุณสมบัติของสายส่งแต่ละแบบ สำหรับค่าของแต่ละตัวจะหาได้โดยใช้สูตรเฉพาะของสายส่งแต่ละแบบ โดยมี

R เป็นความต้านทานของตัวนำของสายส่งต่อความยาวสาย 1 เมตร

G เป็นค่าความนำต่อความยาวสาย 1 เมตรของอุปกรณ์ที่เป็นฉนวน ซึ่งขึ้นอยู่กับชนิดของวัสดุที่ใช้เป็นฉนวน ซึ่งค่านี้มีผลต่อการสื่อสารโดยใช้แรงดันสูงเท่านั้น

L เป็นค่าอินดัคแตนซ์ต่อความยาว 1 เมตรของสายส่ง ขึ้นอยู่กับรูปร่างของสายส่ง

C เป็นค่าคาปาซิเตอร์ต่อความยาว 1 เมตรของสายส่ง ขึ้นกับรูปร่างของสายส่ง และคุณสมบัติของวัสดุที่ใช้ทำฉนวน



รูปที่ 5.2 วงจรสมมูลของสายส่งสัญญาณ

หน่วยการวัดการลดทอน

การผิดเพี้ยนของสัญญาณขณะเดินทางผ่านช่องสัญญาณมีอยู่ 2 อย่างคือ ความเพี้ยนเนื่องจากความแรงของสัญญาณลดลง (attenuation distortion) และความเพี้ยนเนื่องจากความล่าช้าของสัญญาณ (group-delay distortion) สัญญาณที่เดินทางจากด้านส่งไปยังด้านรับไปถึงช้าหรือเกิดการเลื่อนเฟสของสัญญาณขึ้น หน่วยที่ใช้วัดคือเรเดียน

การลดทอนขนาดของสัญญาณในสายส่งขึ้นอยู่กับชนิดและความยาวของสาย และความถี่ของสัญญาณ

ยิ่งสายมีขนาดใหญ่ขึ้นเท่าไร การลดทอนขนาดของสัญญาณก็จะยิ่งน้อยลงเท่านั้น

การลดทอนที่เกิดจากการลดทอนความแรงของสัญญาณ หน่วยที่นิยมใช้วัดการลดทอนนี้มีอยู่ 2 ลักษณะ คือ เดซิเบล และเนเปอร์ ซึ่งสามารถหาได้จากสูตร

$$\text{การลดทอนความแรงของสัญญาณ} = 10 \log_{10} (P_2 / P_1) \text{ dB}$$

$$= \frac{1}{2} \ln (P_2 / P_1) \text{ Np}$$

เมื่อ P_1 , P_2 เป็นกำลังสัญญาณที่จุดสองจุดที่ต้องการเปรียบเทียบ ซึ่งทั้งสองหน่วยมีความสัมพันธ์กันดังนี้ 1 Np = 8.686 dB การแก้ผลที่เกิดจากการลดทอนสัญญาณในสายส่งทำได้โดยการใช้วงจรทวนสัญญาณที่เรียกว่า รีพีทเตอร์ (repeater)

สาเหตุของการผิดเพี้ยนรูปทรงของสัญญาณในการส่งข้อมูล

สาเหตุของการผิดเพี้ยนรูปทรงของสัญญาณในสายส่งข้อมูล สามารถแยกเป็นข้อใหญ่ๆได้ดังนี้ คือ

1. สัญญาณรบกวนที่มาจากภายนอก เป็นปัญหาที่ไม่สามารถหลีกเลี่ยงได้ เนื่องจาก

ข้อมูลในรูปสัญญาณจะต้องเดินทางไปตามสายส่ง ซึ่งมีระยะทางยาว และอยู่ในสภาพแวดล้อมต่าง ๆ แต่การผิดเพี้ยนของสัญญาณไม่ได้เกิดขึ้นจากสัญญาณรบกวนที่มาจากภายนอกแต่เพียงอย่างเดียวเท่านั้น แต่อาจเกิดจากคุณสมบัติของตัวสายข้อมูลเองด้วย

2. **ผลตอบสนองความถี่ของสายส่งข้อมูล** สาเหตุของการผิดเพี้ยนรูปทรงของสัญญาณที่เกิดจากตัวสายส่งข้อมูลเองมีอยู่ 2 ประการ ประการแรก คือ จากการตอบสนองต่อความถี่ของสายส่ง กล่าวคือ ถ้าสายส่งมีความยาวมาก ก็จะทำให้เกิดการลดทอนขนาดของสัญญาณอันเนื่องมาจากตัวสาย การลดทอนขนาดของสัญญาณนี้มีค่าไม่คงที่ ขึ้นอยู่กับความถี่ของสัญญาณ ดังแสดงในรูปที่ 5.3

(ก) ผลตอบสนองทางขนาด (ข) ผลตอบสนองทางเฟส

รูปที่ 5.3 ผลตอบสนองต่อความถี่ของสายส่งข้อมูลโดยทั่วไป

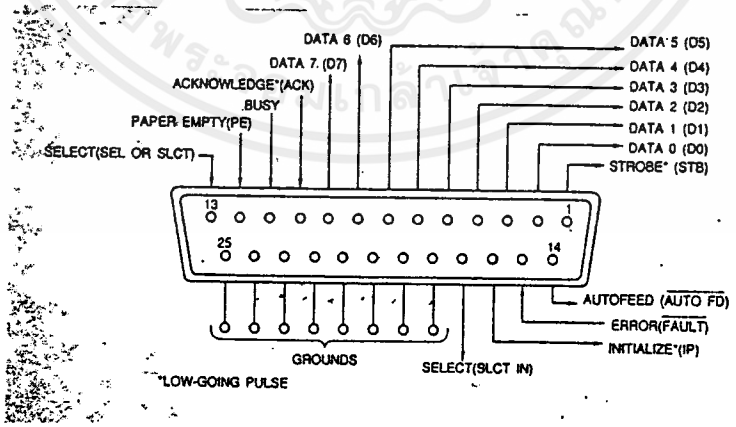
3. **การสะท้อนกลับของสัญญาณ** สัญญาณต่าง ๆ เดินทางไปในสายส่งข้อมูลในรูปของคลื่น ดังนั้นเมื่อคลื่นเดินทางไปถึงปลายสายหรือบริเวณที่มีรอยต่อของสาย ส่วนหนึ่งของคลื่นจะถูกสะท้อนกลับมา คลื่นของสัญญาณที่สะท้อนกลับมาจะรวมกับคลื่นที่ส่งไปตอนแรก เป็นสาเหตุทำให้เกิดการผิดเพี้ยนรูปทรงของสัญญาณ

6. พอร์ตขนาน (Parallel Port)

6.1 ลักษณะของการเชื่อมโยงแบบขนาน

ในระบบการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ โดยทั่วๆไปนั้น เราจะพบรูปแบบการการติดต่อระหว่างกันนั้น เป็น 2 แบบ คือ การติดต่อแบบอนุกรม และ การติดต่อแบบขนานซึ่งการติดต่อในแต่ละแบบนั้น ก็มีความแตกต่าง และข้อดี ข้อเสีย อยู่ในตัวเอง คือ ใน การติดต่อแบบขนาน สามารถส่งข้อมูล ได้ในจำนวนที่มากกว่า การติดต่อแบบอนุกรม แต่ว่าก็ถูกจำกัดทางด้านระยะทาง คือการติดต่อแบบขนานนั้น จะสามารถส่งข้อมูลได้ ในระยะทางที่ใกล้กว่า เนื่องจากมีการลดทอนมากกว่า และในที่นี้จะขอกล่าวในเรื่องการติดต่อแบบขนาน เพื่อจะสามารถเข้าใจการทำงาน ของโครงงานนี้ได้

ในการติดต่อสื่อสารข้อมูล หรือ การเชื่อมโยง(Interface) แบบขนานนั้น จะเป็นการสื่อสารข้อมูลผ่าน พอร์ตแบบขนาน ไปสู่อุปกรณ์ภายนอกเครื่องคอมพิวเตอร์ ซึ่งข้อมูลนั้นจะถูกส่งออกไปในเวลาที่พร้อมกัน โดยที่การเชื่อมโยงนั้นจะประกอบด้วย ตัวอุปกรณ์ที่สำคัญ นั่นก็ คือตัวคอนเน็กเตอร์ ซึ่งโดยทั่วไปในการเชื่อมโยงแบบขนาน จะใช้คอนเน็กเตอร์ ที่เป็นแบบ DB-25 ซึ่งคอนเน็กเตอร์ แบบนี้นั้น จะมีขาในการต่อเชื่อมโยงทั้งหมด 25 ขา โดยที่การวางตัวของขาและชื่อขาสัญญาณนั้น จะเป็นดังรูปที่ 6.1



รูปที่ 6.1 คอนเน็กเตอร์แบบ DB-25 และชื่อขาสัญญาณ

รูปที่ 6.1 นั้น แสดงถึงสัญญาณที่จะเดินทางเข้าสู่คอนเน็กเตอร์ ซึ่งสัญญาณเข้าจะแสดงด้วย ลูกศรที่ชี้เข้าหา ตัวคอนเน็กเตอร์ และสัญญาณออก จะแสดงด้วย ลูกศร ที่ชี้ ออกจาก ตัวคอนเน็กเตอร์ ซึ่งในทางเป็นจริงแล้วนั้น ตัวคอนเน็กเตอร์ ในรูปที่ 1 นั้น จะมี ลักษณะเหมือนที่อยู่ ข้างหลังตัวคอมพิวเตอร์ ดังที่กล่าวมา ข้างต้น ส่วนกราวนั้นนั้น จะแสดงด้วยหัววงกลมและอักษรย่อที่เป็นมาตราฐานของสัญญาณ โดยแสดงไว้ในวงเล็บหลังชื่อสัญญาณต่างๆ

6.2 สัญญาณต่างๆของ BD-25

โดยที่ระดับสัญญาณต่างๆ ที่ผ่านเข้าออก จากคอนเน็กเตอร์นั้น จะเป็นไปตาม ระดับของสัญญาณที่เป็นแบบ TTL ดังนั้น สัญญาณที่อยู่ระหว่าง 2.4 ถึง 5 โวลท์นั้น จะเป็น ในระดับของสัญญาณ High หรือ ลอจิก "1" และ สัญญาณที่อยู่ระหว่าง 0 ถึง 0.8 โวลท์นั้น จะเป็นระดับสัญญาณ Low หรือ ลอจิก "0" ส่วนสัญญาณที่อยู่ระหว่าง 0.8 ถึง 2.4 โวลท์นั้นจะถูกพิจารณาว่าเป็นโมฆะ หรือใช้การไม่ได้

สัญญาณแรกที่พิจารณาในที่นี้ คือ สัญญาณสโตรบ (Data strobe) คอมพิวเตอร์จะสร้างพัลส์ Low ที่สายเส้นนั้นเพื่อเป็นการบอกถึง สถานะแก่อุปกรณ์ที่ต่อเชื่อมโยงด้วยกัน ในระบบ ว่าข้อมูลที่ออกจากคอมพิวเตอร์นั้นพร้อมที่จะส่งแล้ว และทำการเตรียมตัว พร้อมที่จะรับข้อมูลเข้าสู่ตัวคอมพิวเตอร์แล้ว (Data 0 จนถึง Data 7)

สายข้อมูล หรือดาต้า (ที่ขา 2 จนถึงขา 9) มีขนาดของข้อมูลคือ 8 บิต เป็นตัวที่จะรับข้อมูล ได้พร้อมกันในเวลาเดียวกันคือ 1 ไบต์ ข้อมูลที่มา อาจเป็นตัวอักษรก็ได้ ซึ่งอุปกรณ์บางตัวจะรับข้อมูลได้ทีละ 7 บิตเท่านั้น ในกรณีนั้นขา ดาต้า 7 (Data 7) ไม่ได้ถูกนำมาใช้หรือในบางที อาจจะใช้ใน การตรวจสอบค่าพาริตี้ได้ ดังนั้นในบางครั้ง ควรที่จะทำ การดิสเอเบิล (Disable) การตรวจสอบพาริตี้ไว้เลย

วิธีนี้ สามารถสั่งให้คอมพิวเตอร์ รอจนกว่า จะพร้อมที่จะรับข้อมูล ในครั้งต่อไปได้ โดยอุปกรณ์ จะสามารถหยุดรับข้อมูลโดยการส่งสัญญาณ Busy เป็น High เมื่อ อุปกรณ์ ที่เชื่อมต่อร่วม ตรวจสอบพัลส์สโตรบ มันจะคงค่า Busy เป็น High จนกว่า จะทำการประมวลผลข้อมูลเดิมเสร็จ และเมื่อพร้อมที่จะรับข้อมูลแล้ว ก็จะทำให้สัญญาณ Busy เป็น Low ซึ่งทำให้ในส่วนคอมพิวเตอร์ สามารถส่งข้อมูลมายังอุปกรณ์ได้ สัญญาณ Busy นั้น บางครั้ง ก็อาจจะใช้สำหรับหยุดคอมพิวเตอร์ ในกรณีอื่นก็ได้ เช่น ในกรณีที่ Out of paper หรือ Off line ในการ ที่คอมพิวเตอร์เชื่อมโยงติดต่อกับเครื่องพิมพ์ เป็นต้น แต่ก็มี ข้อควรระวังอย่างหนึ่งนั่นคือ สัญญาณ Busy นี้ บางครั้งสามารถเปลี่ยนขั้วได้ โดยการกำหนด ดิพสวิตช์ ที่อุปกรณ์ร่วม ให้ เกิด การเปลี่ยนแปลง ได้เช่นกัน

สัญญาณ ที่สำคัญอีก สัญญาณหนึ่ง คือสัญญาณ Acknowledge และ Busy จะทำงานในลักษณะเดียวกัน แต่คนละหน้าที่ โดยที่ อุปกรณ์ที่มาเชื่อม โยงบางตัว อาจจะ สนับสนุน สัญญาณ Hand-shaking อื่นๆอีกที่ใช้สำหรับแสดง หรือ ควบคุมสถานะของอุปกรณ์ได้ เช่น ต้องการที่จะบอกกับคอมพิวเตอร์ว่า ไม่มีกระดาษ ก็จะสามารถกระทำได้โดยการทำให้สายPape empty เป็น High จนกว่า ได้รับกระดาษเข้าไป คุณสมบัตินี้มีอยู่ใน พอร์ตขนานของ IBM อยู่แล้ว แต่คอมพิวเตอร์อื่นๆ อาจไม่มีคุณสมบัตินี้ก็ได้

อุปกรณ์ร่วมสามารถที่จะบอก คอมพิวเตอร์ขณะที่ power up หรือ online ได้โดยการให้สาย select (ขา 13) เป็น High ซึ่งสัญญาณนี้ มีความจำเป็นในบางครั้งเท่านั้น เพราะอุปกรณ์ร่วมบางตัวสามารถทำ power up ได้เอง แต่จะถูก offline โดยการ ส่งสัญญาณในลักษณะพิเศษที่เรียกว่า Deselect (ซึ่งมีค่า ASCII เท่ากับ 19) เป็นต้น ดังนั้น อุปกรณ์ จึงสามารถที่จะ online ได้อีกครั้งโดยการส่งสัญญาณ Select (ซึ่งได้มีค่า ASCIIเท่ากับ 17)

ถ้าคอมพิวเตอร์ส่งพัลส์ Low-going เข้าไปในสาย Input initialize อุปกรณ์ร่วมจะสนับสนุนการกระทำสายนี้ โดยการรีเซตค่าพารามิเตอร์ ของมันไปยังค่า Defauit ยังเป็นโครงสร้างเริ่มแรก ของเครื่อง หรือ เป็น สถานะของ power up ที่อุปกรณ์ นั่นเอง อุปกรณ์สามารถแสดงค่าผิดพลาดทั่วไปได้ โดยการทำให้สาย Fault เป็น Low ซึ่งก็จะมีลักษณะคล้ายกับสาย Busy บางครั้ง อุปกรณ์ร่วมจะ ใช้สายนี้ในการแสดงว่า มัน offline หรือ out of paper ซึ่งก็เพียงพอแล้วสำหรับสัญญาณที่พอร์ตขนาน

6.3 พอร์ตต่างๆ บนพอร์ตขนาน

ในการติดต่อสื่อสารระหว่าง Parallel Port นั้น จะมีอินพุตและเอาท์พุตพอร์ตที่เกี่ยวข้องด้วย อยู่หลายพอร์ตด้วยกัน ซึ่งแต่ละพอร์ตนั้น ก็มีหน้าที่ที่ต่างๆ กันไปตาม ลักษณะของพอร์ตนั้นๆ ซึ่งลักษณะต่างๆ จะเป็นตามดังได้แสดงไว้ในรูปของ Parallel Port จะ สามารถสรุปการทำงาน และหน้าที่ของแต่ละพอร์ต ได้ดังนี้

-พอร์ตเบอร์ 0378H เป็นพอร์ตที่ใช้ในการอ่านหรือเขียนข้อมูลได้ ครั้งละ 8 บิต ซึ่งในการทำงาน จะทำงานเป็นโหมดเดียวเท่านั้น โดยการเลือกว่า จะต้องการให้เป็นอินพุตพอร์ต หรือเอาท์พุตพอร์ต แต่ในทางปฏิบัติแล้วเราสามารถเขียนข้อมูลออกได้อย่างเดียว เพราะ เอาท์พุตของพอร์ตนี้ ต่อกับภาคอินพุตในพอร์ตเดียวกัน ซึ่งทำให้ข้อมูลที่ส่งออกไปทางพอร์ตนี้นั้น จะไปปรากฏบนภาคอินพุทในพอร์ตเดียวกัน ดังนั้น เมื่อมีทำการส่งข้อมูลออกไปที่พอร์ตนี้ และทำการอ่าน ค่าข้อมูลเข้ามา ที่พอร์ตเดียวกัน ก็จะได้ค่าข้อมูลซึ่งทำการส่งออกไปเมื่อครั้งก่อน ซึ่งจะพบว่า ในการใช้แต่งานนั้น โดยทั่วไปแล้วนั้น จะ

กำหนดให้เป็น อินพต หรือ เอาท์พตพอร์ต อย่างเดียวเท่านั้น ซึ่งจริงๆ แล้ว เพราะว่าที่เอาท์พุต ของพอร์ตนี้ ได้ใช้ ไอซี 74LS374 ให้ทำการ Latch ข้อมูลค้างเอาไว้โดยต่อขา Output enable ไว้ ลง ground ดังนั้น ข้อมูล เดิมที่ส่งไปปรากฏที่ Data bus นี้คงไว้อยู่ตลอดเวลา และ เราก็ไม่สามารถทำให้ ไอซี 74LS374 นี้อยู่ในสภาวะ High impedance ได้ เพราะเหตุว่าขา Output enable อยู่บนบอร์ดภายในเครื่องคอมพิวเตอร์ ดังนั้น จึงไม่สามารถ อ่านข้อมูล ที่ได้จาก ภายนอกผ่านเข้ามาทาง Bus นี้ได้

ส่วนค่าที่อ่านได้นั้นจะเป็นข้อมูลเดิมที่ คอมพิวเตอร์ ส่งออกไป ซึ่ง สามารถใช้ได้เฉพาะในการสอบความถูกต้องเท่านั้น

-พอร์ตเบอร์ 0379H เป็นพอร์จที่ใช้ในการอ่านข้อมูลเพียงอย่างเดียวเท่านั้นโดยทั่วไป เป็นพอร์ตที่ใช้ในการอ่านสถานะของเครื่องพิมพ์ ซึ่งจะสามารถอ่านข้อมูล ได้ครั้งละ 5 บิตเท่านั้น โดยข้อมูลที่เข้ามาคือ บิต D3-D7 ส่วน บิต D0-D2 นั้น จะ ใช้ไม่ได้ และบิต D7 จะถูก กลับลอจิก ด้วยอินเวอร์เตอร์ โดยที่ข้อมูลที่อ่านเข้ามาจะมีรูปแบบดังนี้

D7	D6	D5	D4	D3	D2	D1	D0
o	*	*	*	*	X	X	X

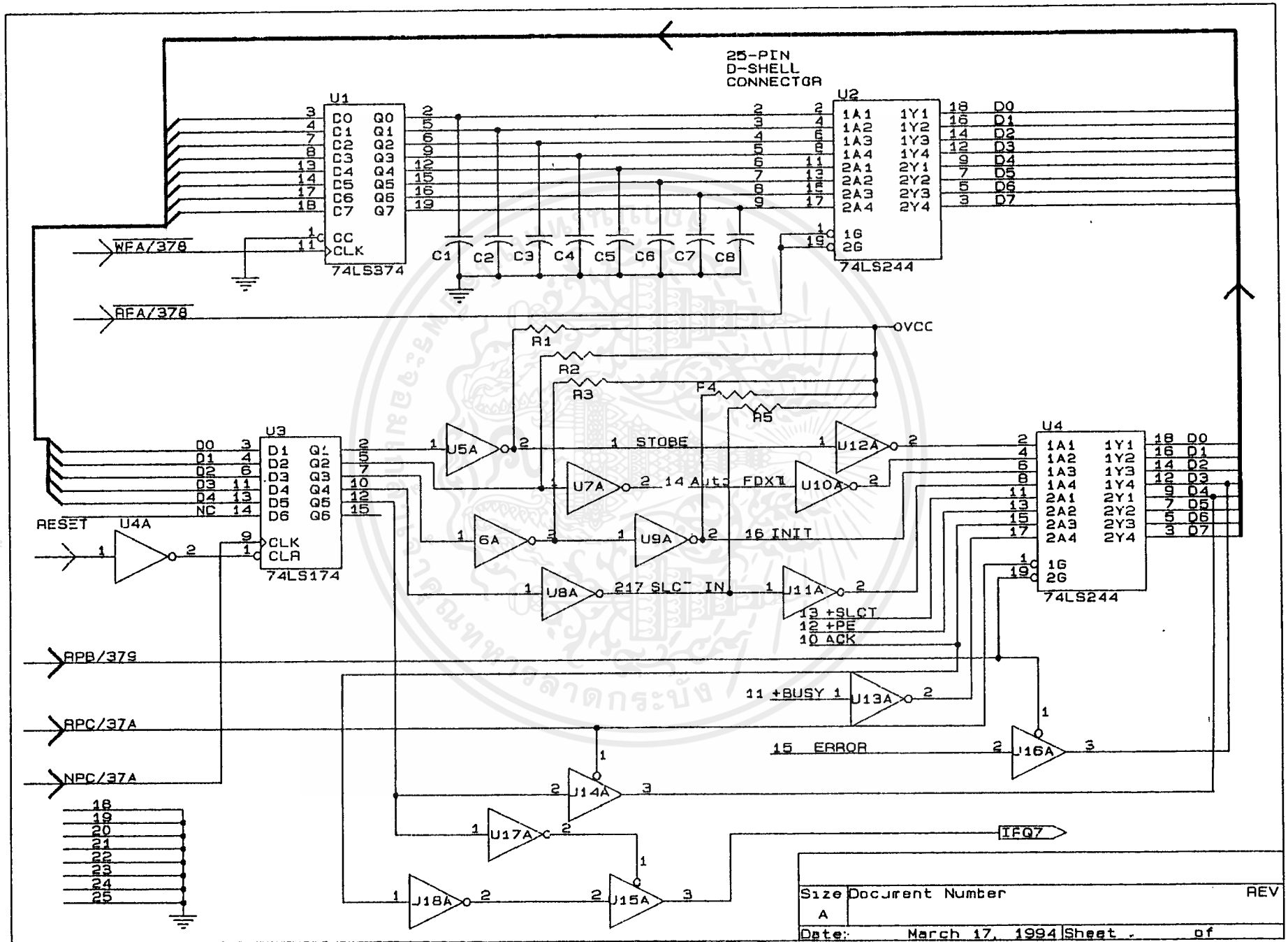
โดยที่ o = Data ที่ถูก inverse

* = Data ที่ไม่ inverse

X = ไม่ใช้

-พอร์ตเบอร์ 037AH เป็นพอร์ตที่ใช้ได้ทั้งการอ่านและเขียนข้อมูล โดยที่การอ่านหรือเขียนข้อมูลนั้นทำได้ครั้งละ 5 บิต โดยแบ่งการทำงานเป็น 2 โหมด ก็คือ การอ่านและการเขียนดังนี้

1) Write จะทำงานในโหมดส่ง คือจะทำการส่งข้อมูลไปยังพอร์ตโดยที่ค่าข้อมูลใน บิต D5-D7 ไม่ได้ใช้ ส่วนข้อมูลใน บิต D4 เป็น ขา +IRQEN และเมื่อทำการส่งข้อมูลออกไปจะถูก Latch ค่าไว้ด้วย 74LS174 ซึ่งเมื่อรวมกับสัญญาณ Acknowledge จะกลายเป็นสัญญาณ Interrupt ของเครื่องพิมพ์ (IRQ7) ซึ่งบิตนี้ ไม่สมควรจะนำมาใช้ เนื่องจากอาจเกิดการไม่ระวัง ในการใช้ขึ้นได้ ทำให้โดยรวมแล้ว ใน บิต D4-D7 จะไม่ถูกนำมาใช้ และจะต้องส่งค่าด้วย 0Xh (hex) ก็คือเลขในฐาน 16 โดยที่ X คือ


25-PIN
D-SHELL
CONNECTOR
U1
74LS374
U2
74LS244
U3
74LS174
U4
74LS244
WFA/378
RFA/378
RPB/379
RPC/37A
NPC/37A
RESET
VCC
R1
R2
R3
R4
R5
C1
C2
C3
C4
C5
C6
C7
C8
STOBE
AUTO FDXT
16 INIT
17 SLCT IN
+SLCT
+PE
ACK
+BUSY
ERROR
IFQ7
Size
A
Document Number
REV
Date: March 17, 1994
Sheet of

ค่า 4 บิตต่ำของข้อมูลใน ไบต์นั้น D0-D3 เป็น ส่วนที่ จะนำมาใช้งานอื่นได้ และ D0, D1 และ D3 จะถูก Inverse ในขณะส่งออกไป ดังนั้น รูปแบบในการส่งข้อมูลออก จะเป็นดังนี้

D7 D6 D5 D4 D3 D2 D1 D0

X X X 0 o * o o

ค่า บิต D4 นั้นควรเป็น ลอจิก '0'

2) Read จะทำงานในโหมดการอ่าน โดยที่ค่าบิต D4 ก็คือค่าที่ส่งออกไปในการ Write หรือ ส่งข้อมูลออก ไปส่วนบิต D0-D3 ค่า D0, D1, D3 จะถูกทำ การเปลี่ยนแปลง ลอจิก ด้วยการ Inverse ด้วย 74LS04 ในขณะส่งออกไป

7. การทำงานของ UM3750

ในการทำงานในด้าน การรับ และส่งข้อมูล ในปัจจุบันนั้น จะพบว่ามีไอซีเป็นจำนวนมากที่ได้ผลิตออกมา เพื่อทำให้การใช้งานด้าน การรับ และส่งข้อมูล มีความสามารถ ที่จะประยุกต์และนำมาใช้งาน เพื่อให้เกิดประโยชน์ อย่างกว้างขวางขึ้น และ UM3750 นั้นก็เป็นไอซีอีกตัวหนึ่งที่มีความสามารถทางด้าน การรับ และส่งข้อมูลที่ใช้งานทางด้านดิจิตอล ซึ่งจะได้กล่าวต่อไป

7.1 ลักษณะทั่วไปของ UM3750

UM3750 นั้น เป็นตัวไอซีชิฟเดี่ยว ที่มีความสามารถทำงานใน ทั้งภาครับ และภาคส่งนั่นคือเป็น ทั้งตัวเข้ารหัส (Encoder) และถอดรหัส (Decorder) ในตัว โดยใช้ไฟเลี้ยง +3 โวลท์ ถึง +11 โวลท์ สามารถปรับ ความถี่ ของข้อมูล ในการรับและส่งได้ด้วย การต่อค่าความต้านทาน และค่าตัวเก็บประจุ สามารถ ทำการติดต่อสื่อสาร ระหว่างกันได้ด้วยการส่งขบวนพัลส์ หรือที่เรียกว่า word ออกไปยังคู่สื่อสาร (ในที่นี้ก็คือ UM3750 แต่ละตัว) โดยการส่งขบวนพัลส์ ออกไป และภาครับจะต้องทำการรับ สัญญาณให้ได้ถูกต้อง เป็นจำนวน 4 ขบวน ในระยะเวลา 64 ms จึงจะถือว่าเป็น การรับสัญญาณได้ โดยทั่วไปแล้วจะใช้งานในด้าน ระบบควบคุมด้วยสัญญาณนาฬิกา ระบบรักษาความปลอดภัย หรือการควบคุมด้วยรีโมทต์ เป็นต้น

7.2 ลักษณะการทำงานของ UM3750

UM3750 จะเป็น ทั้งตัวเข้ารหัส และถอดรหัสในตัวเดียวกัน จะทำงานใน การรับ และส่ง ในระบบดิจิตอล เมื่อ UM3750 ทำงานในโหมดที่เป็น ภาคส่งนั้นจะทำการเข้ารหัสและส่งค่าข้อมูล ออกไปเป็น ขบวนพัลส์ จำนวน 12 บิต ซึ่งเป็นข้อมูลที่ขา A1 ถึง A12 ซึ่งการส่งแต่ละครั้งนั้น ที่ขาอินพุต ทั้ง 12 ขา จะถูกแปลงค่า เป็นขบวนพัลส์ ทั้งหมดรวมแล้วเป็นจำนวน 12 บิต. ในการทำงานเป็น ภาครับ ขบวนสัญญาณนั้นจะถูกรับ เข้ามาเปรียบเทียบกับ รหัสภายในซึ่งอาจจะถูก ตั้งไว้ในลักษณะที่เป็น ขบวนพัลส์ด้วยเช่นกัน ในทันทีที่เกิดการมีข้อผิดพลาด ระบบจะทำการ รีเซต ค่าต่างๆ ภายในตัวเอง และทำการเริ่มต้นเปรียบเทียบ ค่าของขบวนพัลส์ใหม่ ถ้า UM3750 ทำการ รับสัญญาณ และทำการเปรียบเทียบ เป็นจำนวน 12บิต แล้วถูกต้องตามรหัสของภาครับแล้ว ภายในตัวไอซี จะกำเนิด "Valid Signal" ซึ่งเป็นการแสดงว่าสัญญาณนั้นถูกต้อง สัญญาณนี้จะทำการเคลียร์วงจรนับซึ่งนับ 64 ms และตัวทริกจ์ที่เป็นตัวนับแบบ 3-stage และตัวนับแบบ 3-stage จะทำการนับ "Valid Signal" และเมื่อทำการนับสัญญาณนี้จนครบ 4 ครั้งแล้ว จะทำให้ขา Transmit/Receive output แปรเปลี่ยนลอจิกไปเป็น "0" หรือนั่นก็คือ ขานี้จะ Active ที่สถานะต่ำ หรือ Low นั่นเอง และหลังจากนั้นเมื่อขา Transmit/Receive output เปลี่ยนสถานะเป็น Low หรือ Active Low แล้ว เมื่อมี "Valid Signal" เข้ามาอีก ภายในเวลา 128 ms ก็จะทำให้ที่ขานี้ยังคงสถานะไปได้อีกระยะหนึ่ง ดังนั้น ในทางปฏิบัติแล้วนั้น ในการส่งสัญญาณไปยังภาครับนั้น ภาคส่งควรทำการส่ง "Valid Signal" ออกไป อย่างน้อย 6 พัลส์ เพื่อที่จะทำให้ ภาครับนั้นได้รับและ ทำการ Detect สัญญาณ ได้ ซึ่งจะทำให้ขา Transmit/Receive output ทำงานที่สถานะต่ำ หรือ Active Low

7.3 การสร้างสัญญาณนาฬิกาเปรียบเทียบ

ในการสร้างสัญญาณ ให้แก่ UM3750 เพื่อเป็นการกำหนด ความถี่ของสัญญาณ ที่จะนำมาเปรียบเทียบนั้น ทำได้โดยการต่อ ตัวความต้านทาน และตัวเก็บประจุ ที่มีขนาดที่ต้องการ โดยใช้สูตร $f = 2/RC$ ในการคำนวณ

ตารางแสดงขาสัญญาณ

ขา	ชื่อขา	การใช้งาน และหน้าที่
1-12	A1-A12	เป็นขาข้อมูลที่ใช้ในการกำหนด Address ในการเข้ารหัส/ถอดรหัส
13	R.C.INPUT	เป็นขาที่ใช้กำหนดความถี่ในการเป็นสัญญาณเปรียบเทียบ โดยที่ ตัวต้านทานจะต้องต่อจากขานี้ ไปยังไฟเลี้ยง และ ตัวเก็บประจุจะต้องต่อจากขานี้ ไปยังกราวด์ โดยที่ f = 2/RC
14	V_{ss}	เป็นขากราวด์ ของ UM3750
15	MODE SELECT	ขานี้เป็นการกำหนดว่าจะให้ ไอซีทำงานเป็น ภาครับ หรือ ภาคส่ง โดยที่ถ้าขานี้ต่อกราวด์ ไอซีก็จะทำงานในภาครับ และถ้าขานี้ต่อไฟเลี้ยง ไอซีก็จะทำงานเป็นภาคส่ง
16	RECEIVER INPUT	เป็นขารับสัญญาณที่เป็นดิจิตอล PCM ที่ มาจากภาคส่ง
17	TX/RX OUTPUT	ในโหมดส่ง จะเป็น ขาส่งสัญญาณดิจิตอล PCM ในโหมดรับจะเป็นตัวตรวจจับ Low เมื่อรับสัญญาณเข้า Decode ได้ถูกต้อง
18	V_{cc}	ไฟเลี้ยงของวงจร ควรอยู่ประมาณ 3-11 โวลท์

การเข้ารหัส

จากการเชื่อมต่อระหว่างคอมพิวเตอร์ กับ UM3750 นั้น จำเป็นที่จะต้องมีการกำหนดขาของ ส่วนที่เป็นขาคอนเน็กเตอร์ และส่วนที่เป็นขาของ UM3750 นั้นให้ตรงกัน อันทำให้ เป็นที่เข้าใจ ระหว่างด้านที่เป็นพอร์ทขนาน และด้านที่เป็น UM3750 ซึ่งจากหัวข้อของพอร์ตต่างๆ บนพอร์ทขนาน นั้น จะมีการเชื่อมต่อเป็นดังนี้

ขาด้าน Connector	บิตที่	พอร์ตที่ติดต่อ	ขาด้าน UM3750	หน้าที่ของขา UM3750
2 - 8	D0-D6	378h(output)	1 - 7	A1-A7 เป็น address ใช้ในการ เข้ารหัส/ถอดรหัส
1	D0	37Ah(output)	9	A9
14	D1	37Ah(output)	10	A10
16	D2	37Ah(output)	11	A11
17	D3	37Ah(output)	12	A12
9, 13	D7	378h(output)	15	MODE SELECT
12	D5	379h(input)	17	Tx/Rx OUTPUT
10	D6	379H(input)	10	RECEIVER INPUT
18 - 25	-	-	14	GROUND

ขา 8 ของ UM3750 นั้น ไม่ได้นำมาใช้ เพราะเหตุว่า ในการทำโครงงานเรื่องนี้เรา จะทำการพิจารณาว่า ขา A1-A8 นั้นเป็นระดับของสัญญาณ ที่จะส่งออกไป ในการควบคุมตัวอุปกรณ์ แต่ในโครงงานนี้ จะพิจารณาค่าของแต่ละบิต เป็นเหมือนตัวบอกระดับสัญญาณว่าเป็น ระดับที่บิตนั้น ดังนั้นเมื่อมี บิตใดบิตหนึ่งเป็น '1' บิตอื่นๆ ต้องเป็น '0' หมด และเนื่องจากการที่ไม่ใช้ ขา A8 และทำการต่อกับกราวด์ไว้ ดังนั้นขานี้จึงมีระดับลอจิกเป็น '0' ตลอด ระดับสัญญาณจึงมีทั้งหมด 7 ระดับด้วยกัน เช่น ข้อมูล 04h นั้น หมายถึงว่าใช้ระดับสัญญาณที่ 3 หรือ 08h นั้น หมายถึงว่าใช้ระดับสัญญาณที่ 4 เป็นต้น

ขา A9-A12 นั้นจะเชื่อมต่อกับ ขา 1,14,16,17 ตามลำดับ และนั่นก็คือเป็นการติดต่อกับพอร์ต 37Ah ซึ่งเป็น พอร์ตอินพุท และข้อมูลที่ออกมานั้น ก็คือ D0-D3 ของพอร์ตนั้นเองตามลำดับ โดยที่ D4-D7 ไม่ถูกนำมาใช้ แต่บิต D0,D1,D3 นั้นจะถูกอินเวอร์ส ด้วยอินเวอร์เตอร์ภายใน เครื่องคอมพิวเตอร์เอง ดังนั้นในทางซอร์ฟแวร์ ควรจะทำการอินเวอร์ส เพื่อ จะทำให้ ค่าที่ได้ออกมาจะตรงกับความเป็นจริง เมื่อตอนส่งข้อมูลออกมา

ขา Mode select นั้น จะเชื่อมต่อเป็นทั้ง อินพุท และเอาท์พุต คือจะสามารถ ควบคุมได้จากคอมพิวเตอร์ และสามารถอ่านสถานะ ของโหมดการทำงานของ UM3750 ได้ด้วย เพื่อมาทำการประมวลผลต่อไป

ขา 17 หรือ Transmit/Receive Input นั้น โดยตาม หน้าที่แล้วนั้น เมื่ออยู่ในการทำงานที่เป็น ภาคส่ง คอมพิวเตอร์ไม่ควรเข้าไปมีส่วนในการทำงาน เพราะ UM3750 จะส่งสัญญาณที่ทำการเข้ารหัสเป็นดิจิตอล PCM แล้วออกไป แต่ในทางตรงกันข้ามนั้น เมื่อเป็นภาครับ นั้นซอร์ฟแวร์จะทำการรับค่าที่ Detect ได้มาตรวจว่ารับสัญญาณ ได้ตรงหรือไม่ ถ้าภาครับ รับสัญญาณที่เป็น ดิจิตอล PCM ที่ขา 16 และทำการถอดรหัส ได้ตรงตามที่ได้ จากขา A1-A12 ที่ตั้งไว้ทั้ง ภาครับ และภาคส่งแล้ว ขา 17 จะ Active Low นั่นคือ การติดต่อสื่อสารเป็นผลสำเร็จ นั่นคือ ขา 16 และ ขา 17 มีไว้ เพื่อสนับสนุนการทำงาน แบบ Hand-Shaking นั่นเอง แต่ ในทางปฏิบัติแล้ว ไม่จำเป็นต้องเข้าไปเกี่ยวข้องกับการทำงานใน ส่วนของขาทั้ง 2 นี้ เนื่องจากเป็น หน้าที่การทำงานของ UM3750 อยู่แล้ว Software จึงทำการตั้งขาทั้ง 2 นี้ เป็น ขาอินพุททั้งคู่

ขา 14 ของ UM3750 นั้นเป็น กราวด์อยู่แล้ว และขา 18-25 ของคอนเน็กเตอร์ นั้นก็เป็นกราวด์เช่นกัน ดังนั้น จึงนำมาเชื่อมต่อกันเพื่อให้เป็น ระดับสัญญาณ อ้างอิงในระดับสัญญาณเดียวกัน

ส่วนขา 13 ของ UM3750 นั้นเป็นขา R.C.Input ซึ่งเป็นตัวกำเนิด สัญญาณความถี่ ที่ใช้ในการเปรียบเทียบ กับสัญญาณเข้าที่ขา 16 ดังนั้น จึงต่อ ตัวต้านทานและตัวเก็บประจุไว้

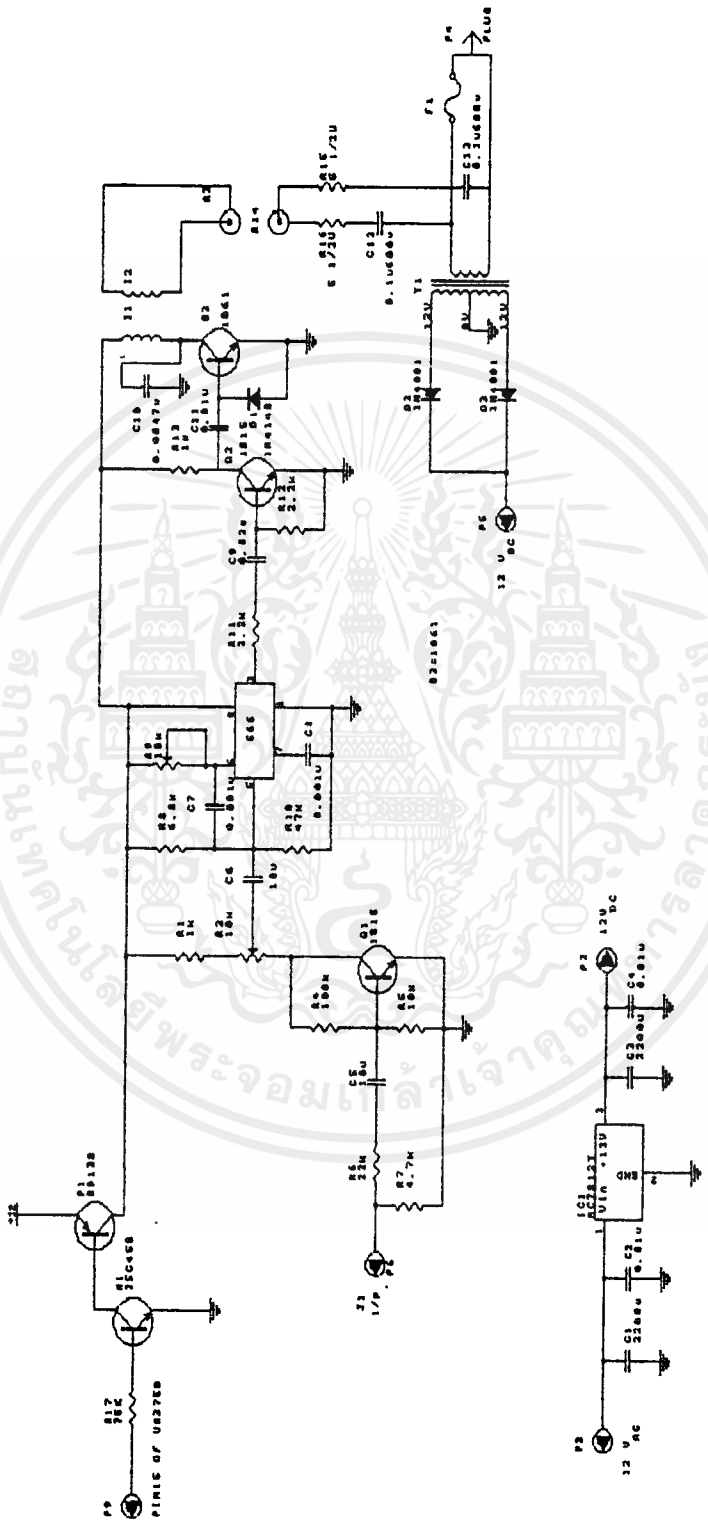
อธิบายการทำงานของวงจรมอดูเลชั่นเชิงความถี่

วงจรมอดูเลชั่นแบบ FM ที่ใช้ในการส่งสัญญาณไปนั้นใช้ไอซีเบอร์ LM 566 ซึ่งเป็นวงจรกำเนิดความถี่ที่ควบคุมด้วยแรงดัน (Voltage controlled oscillator หรือ VCO) ทำหน้าที่ในการมอดูเลตแบบ FM โดยสามารถกำหนดความถี่คลื่นพาห์ได้โดยกำหนด timing capacitor ที่ขา 7 และ timing resistor ที่ขา 6 นั่นคือกำหนดจาก C_8 และ R_9 ซึ่งเป็นความต้านทานปรับค่าได้

สัญญาณอินพุท ซึ่งในกรณีนี้เป็นสัญญาณเสียงที่มีขนาดประมาณ 0.5 V_{pp} ถูกส่งเข้ามาที่ J_1 สัญญาณนี้จะถูกขยายโดยทรานซิสเตอร์ Q_1 ซึ่งสามารถปรับอัตราขยายได้ด้วย R_2 ซึ่งเป็นความต้านทานปรับค่าได้ เพื่อให้ได้เอาท์พุทขนาดเหมาะสม และไม่เกิดการโอเวอร์มอดูเลชั่นที่เป็นสาเหตุหนึ่งของการเกิดเสียงเพี้ยน หลังจากนั้นสัญญาณจะถูกส่งเข้าไปที่ขา 5 ของ LM 566 ซึ่งเป็นวงจร VCO ทำหน้าที่ในการมอดูเลตแบบ FM และกำเนิดคลื่นพาห์ความถี่ประมาณ 100 - 350 kHz ความถี่คลื่นพาห์นี้สามารถเลือกได้โดยการปรับ R_9 สำหรับในโครงงานนี้ได้เลือกใช้ความถี่คลื่นพาห์ 200 kHz สัญญาณเอาท์พุทที่ขา 3 ของ LM 566 มีลักษณะเป็นคลื่นสี่เหลี่ยม คับปลิ้งผ่าน R_{11} และ C_9 เข้าขาเบสของ Q_2 โดยมี R_{12} เป็นตัวไบอัส

เอาท์พุทที่ขา C ของ Q_2 มีขนาดประมาณ 3 V_{pp} ถูกป้อนผ่าน C_{11} ไปยัง Q_3 ซึ่งเป็นวงจรขยายกำลังสูง ได้เอาท์พุทที่ขา C ของ Q_3 45 V_{pp} ที่ขา C ของ Q_3 มี L ต่ออยู่ ซึ่ง L ตัวนี้ทำหน้าที่คับปลิ้งสัญญาณจากด้านไพรมารี่ ไป เซคั่นดารี่ แล้วสัญญาณนี้ก็จะผ่านความต้านทาน R_{15} , R_{16} และ C_{12} , C_{13} ไปยังสายไฟ 220 โวลท์

ในส่วนของไฟเลี้ยงที่นำมาเลี้ยงวงจรนั้น ใช้ไฟเลี้ยง +12 โวลท์ โดยในการจ่ายไฟเลี้ยงให้วงจรมอดูเลตเตอร์ทำงานนั้น จะเกิดขึ้นต่อเมื่อมีสัญญาณจากขา 15 ของ UM3750 ซึ่งเป็นขาเลือกโหมดการส่งหรือรับมาทริกที่ขาของทรานซิสเตอร์ N_1 2SC458 ทำให้ P_1 BD138 ทำงาน จึงมีไฟเลี้ยงให้กับวงจรมอดูเลเตอร์



อธิบายการทำงานของวงจร FM ดีมอดูเลเตอร์

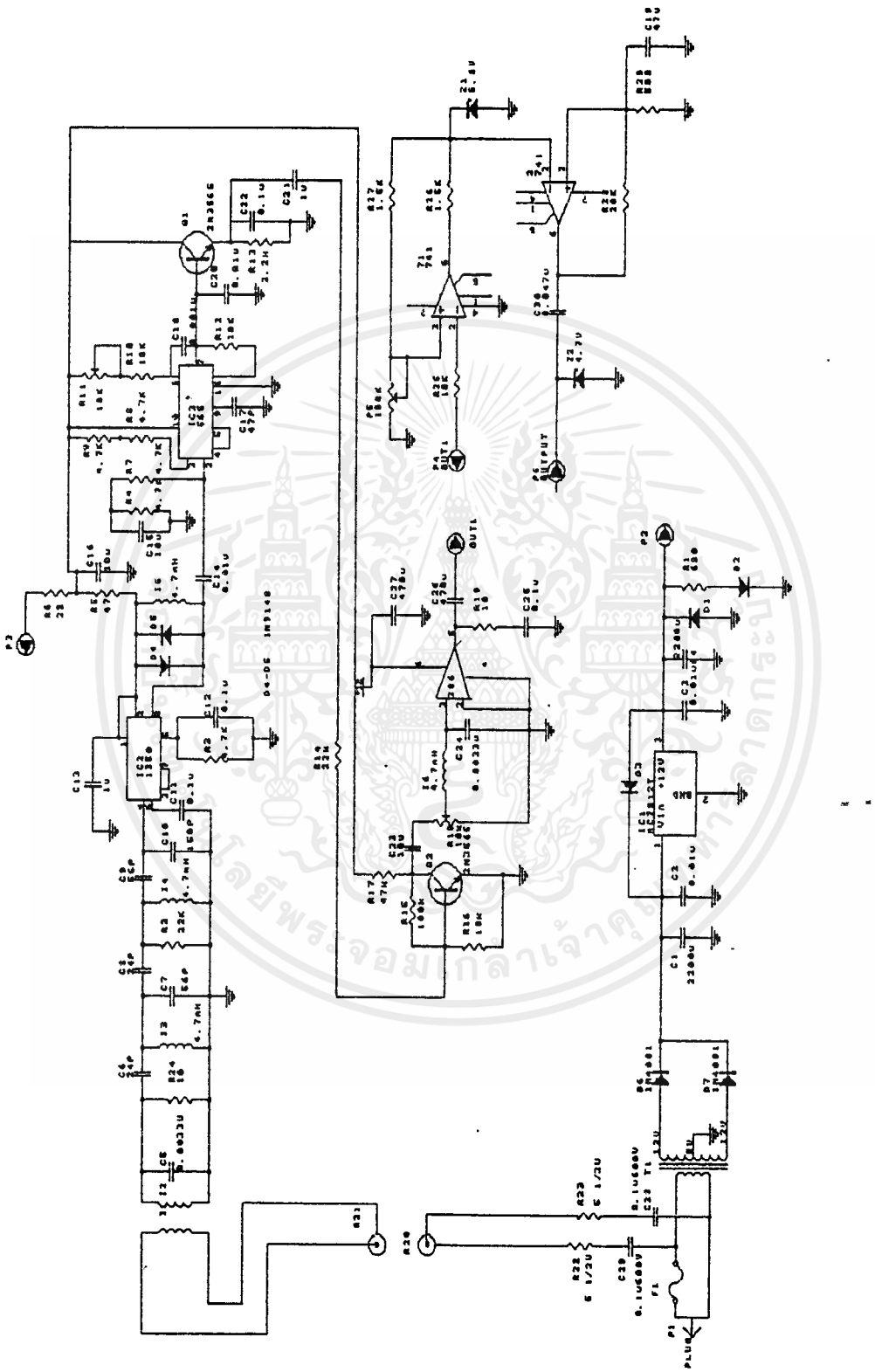
สัญญาณอินพุทที่มาจากสายเอซี 220 โวลท์ จะผ่าน C_{29},C_{28} และ R_{22}, R_{23} กรองความถี่ 50 Hz ออกไป ขดลวด L_1 จะทำการคัปปลิ้งสัญญาณที่กรองแล้วนี้ ซึ่งเป็นสัญญาณที่มาจากภาคส่ง ผ่านไปยังด้าน secondary ของ L_1 อุปกรณ์ตั้งแต่ C_5,R_{24},L_3, C_6,C_7 C_8, R_2, L_4, C_9, C_{10} เป็นวงจรแบนด์พาสฟิลเตอร์ ช่วงความถี่ผ่าน 200 ถึง 340 kHz สัญญาณที่ผ่านแบนด์พาสฟิลเตอร์นี้ จะไปเข้าที่ขา 4 ของ IC_2 เบอร์ MC 1350P ซึ่งเป็นไอซีขยายไอเอฟ(IF Amplifier)ที่มีแบนด์วิดธ์กว้าง และภายใน IC มีวงจร AGC ควบคุมอัตราขยายอัตโนมัติ IC_2 ทำหน้าที่ขยายสัญญาณให้แรงขึ้น ตัวเก็บประจุ C_{11},C_{12} เป็นตัวบายพาสสัญญาณภายใน IC_2 ตัวต้านทาน R_3 เป็นตัวให้ไบอัสแก่ IC_2 เพื่อให้ได้อัตราขยายสูงสุด สัญญาณที่ทำการขยายแล้วจะมีเอาท์พุทออกมาที่ขา 8 ขดลวด L_5 เป็นตัวให้ไบอัสแก่ IC_2 และเป็นโหลดอิมพีแดนซ์สูงกับขา 8 และขา 1 ส่วนไดโอด D_4 และ D_5 ทำหน้าที่ขลิบยอดสัญญาณให้เท่ากัน (limiter)ก่อนดีเทค

ตัวเก็บประจุ C_{14} จะคัปปลิ้งสัญญาณจาก IC_2 ไป IC_3 เบอร์ LM 565 ซึ่งเป็นเฟสล็อกลูปใช้ในการดีเทคสัญญาณแบบ FM (ดีมอดูเลเตอร์) ขา 8 และ ขา 9 ของ IC_3 ต่ออยู่กับวงจรกำเนิดความถี่ที่ควบคุมด้วยแรงดันหรือ VCO (voltage controlled oscillator) โดยมี R_{19} และ R_{14} ซึ่งเป็นความต้านทานปรับค่าได้ และ C_{17} เป็นตัวกำหนดความถี่ เอาท์พุทที่ได้จะปรากฏที่ขา 4 และป้อนกลับมาที่ขา 5 ซึ่งภายในจะทำการเปรียบเทียบเฟสของความถี่ที่รับเข้ามาที่ขา 2 กับความถี่ขา 5 สัญญาณ VCO และ สัญญาณอินพุทจะทำการเปรียบเทียบเฟสภายใน IC_3 และสัญญาณเอาท์พุท ซึ่งเป็นผลต่างของสัญญาณทั้งสองจะถูกขยายและปรากฏที่ขา 7 ได้เป็นสัญญาณความถี่เสียงที่มอดูเลตมากับสัญญาณ FM

สัญญาณที่ผ่านการดีมอดูเลตที่ขา 7 จะถูกส่งต่อไปให้ทรานซิสเตอร์ Q_1 สัญญาณไปออกที่ขา E ของ Q_1 แล้วผ่านตัวเก็บประจุ C_{25} และ R_{14} ซึ่งทำหน้าที่คัปปลิ้งสัญญาณเสียงไปที่ขาเบสของ Q_2 ทำการขยายสัญญาณเสียงขั้นต้น ให้เอาท์พุทประมาณ 1 V_{pp} แล้วนำสัญญาณนี้ไปขยายต่อโดยใช้ ICเบอร์ LM 386 เป็นตัวขยายภาคสุดท้ายก่อนส่งออกเป็นเอาท์พุทของภาคส่ง เราสามารถควบคุมความดังหรือขนาดของเอาท์พุทได้โดยการปรับ R_{18} ซึ่งเป็นความต้านทานปรับค่าได้ ส่วน L_6 และ C_{24} เป็นตัวกรองความถี่คลื่นพาห์ออกไปอีกครั้งหนึ่ง

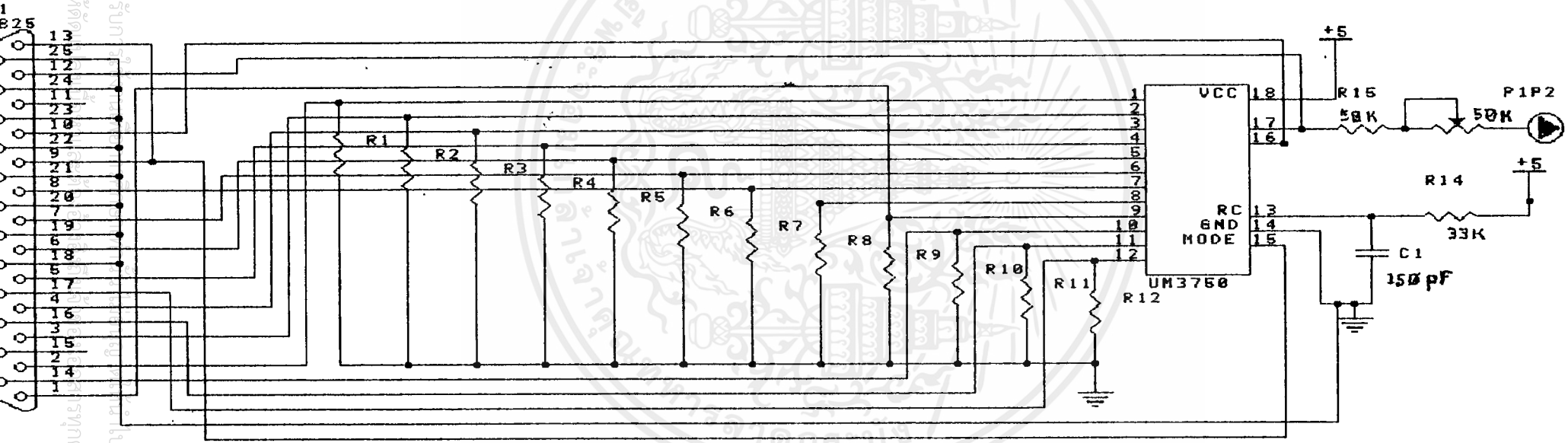
ในกรณีที่สัญญาณที่ใช้ส่งเป็นสัญญาณดิจิตอล เนื่องจากสัญญาณเอาท์พุทที่ได้จากขา 5

ของ IC LM386 มีความผิดเพี้ยนของรูปสัญญาณอยู่ ดังนั้นจึงต้องใช้ออปแอมป์ 741 และซีเนอร์ไดโอดช่วยทำให้รูปสัญญาณดีขึ้น เพื่อนำไปป้อนเป็นอินพุทให้กับ UM3750 ต่อไป



อธิบายวงจรเข้ารหัส

เมื่อข้อมูลส่งมาจากคอมพิวเตอร์ โดยข้อมูลส่งมาที่ขา 1 ถึงขา 8 และขา 14,16 17 ของ DB25 และขาเลือกโหมดที่ขา 9 และขา 13 ของ DB25 ดังรูปของการเชื่อมต่อวงจรนี้ เมื่อ UM3750 มีข้อมูลเข้ามา และข้อมูลเลือกโหมดเป็นลอจิก "1" แล้ว UM3750 จะส่งสัญญาณรูปพัลส์ ดังได้กล่าวไว้แล้วในส่วนทฤษฎีการทำงานของ UM3750 จากนั้นทำการปรับแต่งสัญญาณเอาท์พุทให้มีขนาดเหลือประมาณ 0.2 โวลท์ เพื่อเป็นสัญญาณอินพุทให้กับวงจรมอดูเลเตอร์ต่อไป


1
DB25
13
25
12
24
11
23
10
22
9
21
8
20
7
19
6
18
5
17
4
16
3
15
2
14
1
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
1
2
3
4
5
6
7
8
9
10
11
12
VCC
GND
MODE
RC
18
17
16
13
14
15
UM3750
+5
R15
50K
50K
P1P2
+5
R14
33K
C1
150pF

R1 - R12 = 10K

อธิบายวงจรถอดรหัสและส่งสัญญาณที่ถอดรหัสได้กลับไป

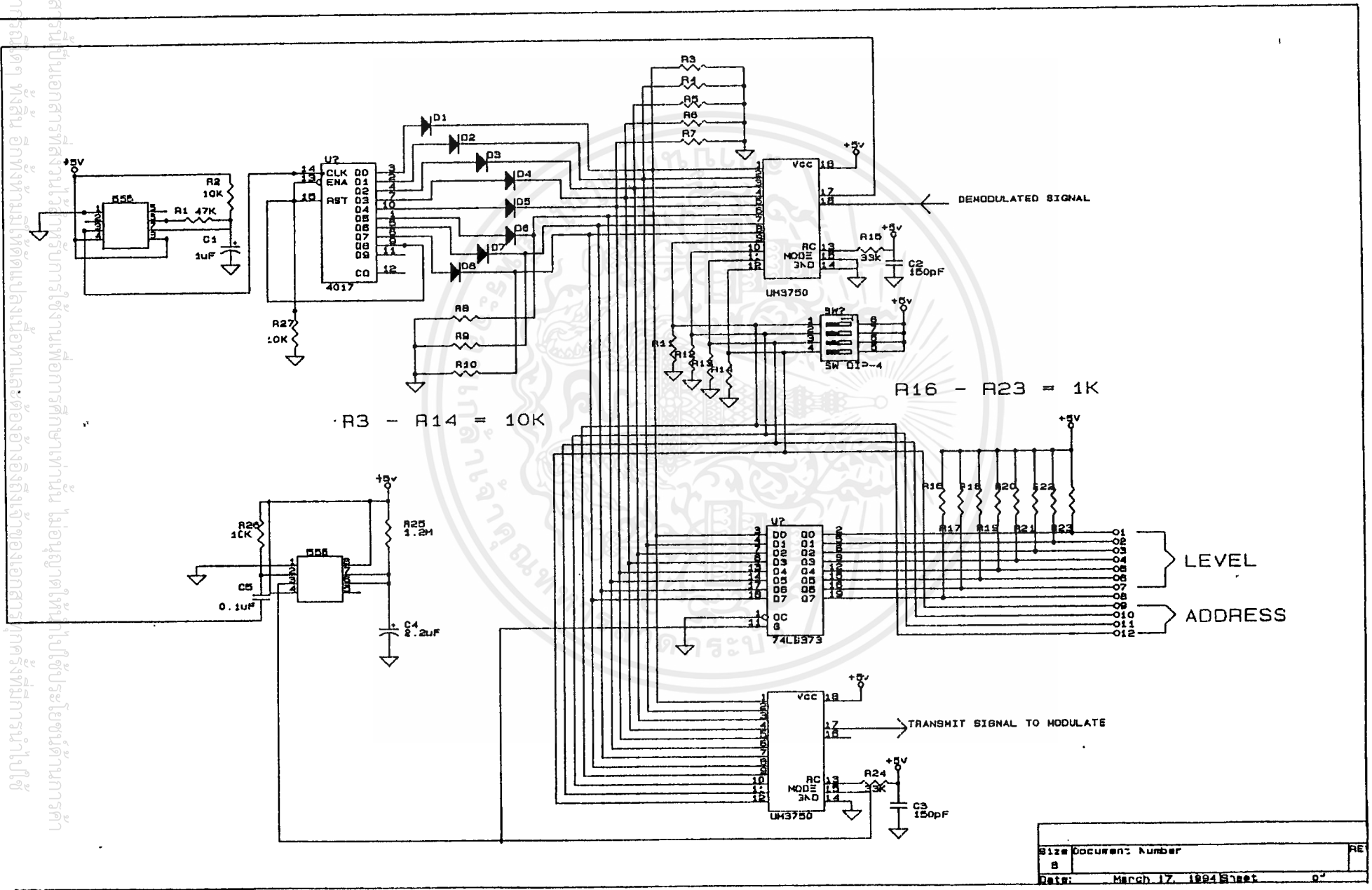
วงจรส่วนของการถอดรหัสประกอบด้วย MC 4017 , UM3750, 555 โดยในการทำงาน ได้กำหนดใช้ขาของ UM3750 8 ขา ในการเลือกระดับของความสว่างของหลอดไฟหรือกำลังที่จ่ายให้กับอุปกรณ์ MC 4017 ซึ่งเป็นตัวนับเชิงเลขฐานสิบ ข้อมูลตั้งแต่ขา 1 ถึง 8 ของ UM3750 จะเปลี่ยนไปตามการทำงานของ MC4017 โดยที่ความถี่ในการทำงานของ MC4017 จะขึ้นอยู่กับความถี่ของสัญญาณนาฬิกาจากวงจรสร้างสัญญาณนาฬิกาของ 555 ซึ่งความถี่นี้จะต้องน้อยกว่าความถี่ข้อมูลที่เข้ามาของ UM3750 อย่างน้อย 4 เท่า ทั้งนี้เนื่องจากว่า UM3750 จะให้สัญญาณที่ส่งมาทีละ 4 ชุด ใน 1 ครั้งของการส่ง

ความถี่ของข้อมูลที่ UM 3750 ส่งมา ; f = 2/RC Hz

ดังนั้นความถี่ของสัญญาณนาฬิกาต้องมีค่ามากกว่า 4*(2/RC) Hz จึงสามารถนำข้อมูลที่เปลี่ยนแปลงตามความถี่สัญญาณนาฬิกาจาก MC 4017 (เป็นข้อมูลเดียวกันกับข้อมูลที่ขา 1 ถึง 8 ของ UM3750) กับข้อมูลที่เป็นที่อยู่ (address)ซึ่งเซ็ทโดยดิปสวิทช์ มาเปรียบเทียบกับข้อมูลที่รับเข้ามาจากสายส่งที่ขา 16 ของ UM3750 ถ้าสัญญาณทั้งสองเหมือนกันทุกประการแล้ว (คือ ให้เป็นลอจิก "1" และ "0" ของแต่ละขา) ที่ขา 17 ของ UM3750 จะทำการรับรู้ว่าข้อมูลที่รับมาตรงกับข้อมูลทางภาครับแล้ว จะให้เปลี่ยนสถานะจากลอจิก "1" เป็นลอจิก "0" ซึ่งจะนำผลของการเปลี่ยนสถานะจากลอจิก "1" เป็น "0" ไปทำการทริกวงจรโมโนสเตเบิ้ล เพื่อให้ได้สัญญาณที่เป็นพัลส์เดี่ยวที่เปลี่ยนภาวะจากลอจิก "0" เป็น "1" ไปควบคุมขา clock enable ของ MC4017 ที่ขา 13 ให้หยุดทำงาน เพื่อคงค่าข้อมูลที่ถูกต้องนั้น แล้วนำข้อมูลดังกล่าวไปแลทช์ค่าไว้ที่ 74LS373 โดยใช้สัญญาณออกของวงจรโมโนสเตเบิ้ลเป็นสัญญาณทริกการทำงานของ 74LS373 ที่ขา 11 เพื่อนำข้อมูลไปใช้งานควบคุมอุปกรณ์ต่อไป

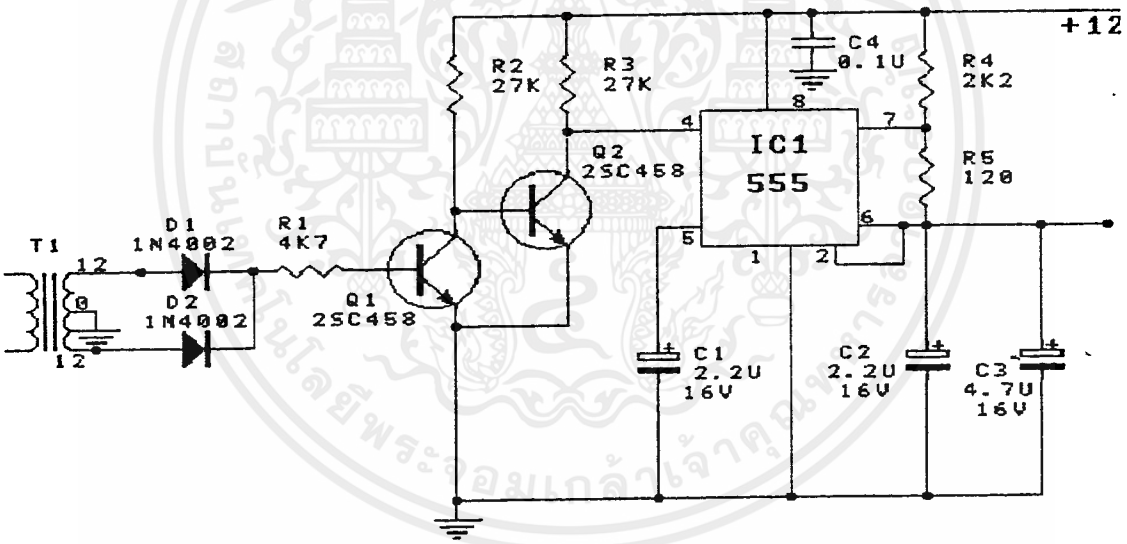
สำหรับวงจรส่วนที่ส่งข้อมูลกลับไปเพื่อให้ด้านส่งรับรู้ว่า ข้อมูลที่ส่งมานั้นถูกต้องทำโดยนำสัญญาณออกของวงจรโมโนสเตเบิ้ลไปใช้เลือกโหมดการทำงานที่ขา 15 ของ UM3750 จากเดิมซึ่งเป็น "0" ให้เป็น "1" ซึ่งจะทำงานเป็นตัวส่ง และนำข้อมูลจาก UM3750 ตัวที่ทำหน้าที่รับเป็นสัญญาณเข้า จากนั้นก็ส่งข้อมูลที่มีความถี่เดียวกันกับสัญญาณที่รับมาจากสายส่ง ทั้งนี้เพื่อให้ตรงตามความถี่เดียวกันกับความถี่ตัวส่ง

หมายเหตุ เวลาที่โมโนสเตเบิ้ลหน่วงไว้นานพอที่จะทำการแลทช์ข้อมูล และส่งข้อมูลกลับไปที่ภาคส่งเพื่อตรวจสอบว่าข้อมูลที่รับมาถูกต้อง


+5V
R2
10K
R1 47K
C1
1uF
U2
CLK
ENA
RST
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
CO
4017
D1
D2
D3
D4
D5
D6
D7
D8
R27
10K
R3
R4
R5
R6
R7
R8
R9
R10
R3 - R14 = 10K
VCC
RC
MODE
GND
UM3750
DEMODULATED SIGNAL
R15
33K
C2
150pF
R11
R12
R13
R14
5W DIP-4
R16 - R23 = 1K
R16
R17
R18
R19
R20
R21
R22
R23
U?
D0
D1
D2
D3
D4
D5
D6
D7
OC
G
74LS373
LEVEL
ADDRESS
R26
10K
R25
1.2M
C5
0.1uF
C4
2.2uF
TRANSMIT SIGNAL TO MODULATE
R24
33K
C3
150pF
Size
B
Document Number
Date:
March 17, 1994
Sheet
of

อธิบายการทำงานของวงจรหรี่ไฟ

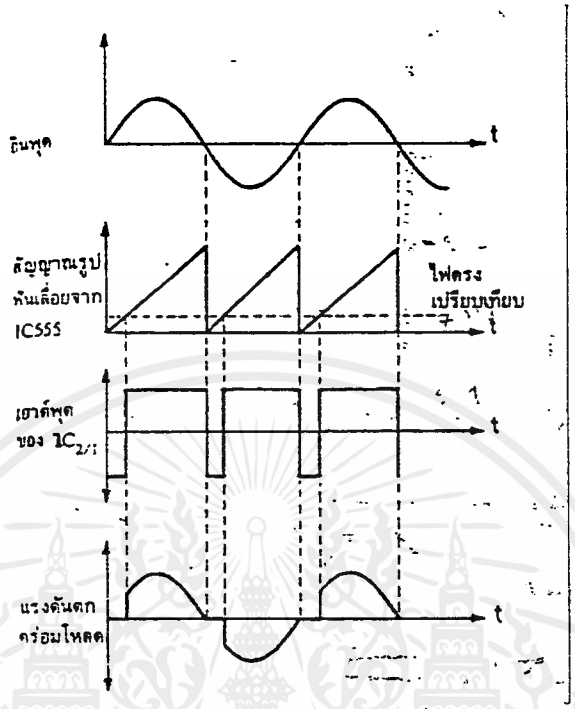
จากรูปวงจรหรี่ไฟ D_1 และ D_2 ทำหน้าที่แปลงไฟสลับให้เป็นไฟตรง โดยไม่ผ่านวงจรฟิลเตอร์ ดังนั้นสัญญาณที่ได้จึงมีความถี่ 100 Hz นำสัญญาณนี้ส่งไปยัง Q_1 และ Q_2 สัญญาณที่ได้ที่ขา C ของ Q_2 ยังคงมีความถี่ 100 Hz ป้อนเข้าขา 4 ของ IC_1 555 ซึ่งเป็นขารีเซต ทำให้เอาท์พุทเปลี่ยนจากสภาวะใด ๆ มาสู่สภาวะ "0" C_2 และ C_3 จึงมีโอกาสที่เก็บและคายประจุในช่วงเวลาที่แตกต่างกัน ขึ้นอยู่กับค่าความจุรวมของ C_2 และ C_3 ทำให้ได้สัญญาณที่มีความลาดชันแตกต่างกันด้วย ถ้ากำหนดให้ช่วงเวลาในการคายประจุสั้นมาก ๆ รูปคลื่นก็จะคล้ายฟันเลื่อยมากขึ้น ดังนั้น R_5 จึงควรมีค่าน้อย เพื่อให้ C_2 และ C_3 คายประจุได้อย่างรวดเร็ว



รูปที่ 3.5.1 วงจรสร้างสัญญาณฟันเลื่อย

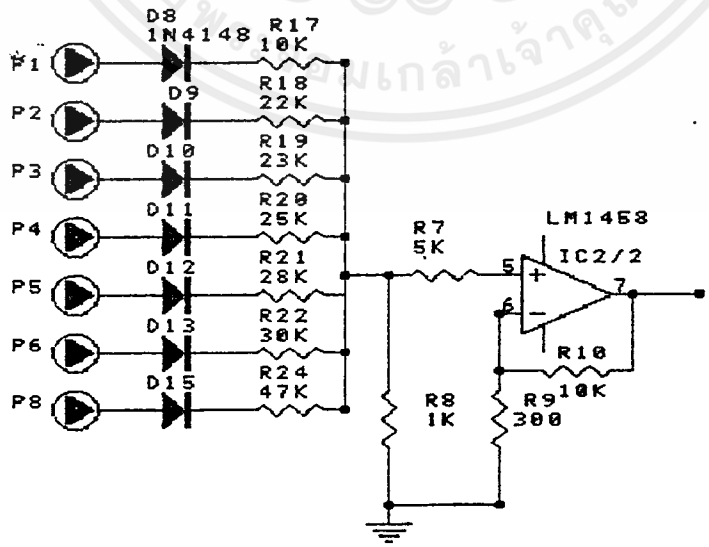
สัญญาณเอาท์พุทที่ขา 6 ของ IC_1 ซึ่งเป็นสัญญาณฟันเลื่อยจะถูกป้อนเข้าที่ขานอนอินเวอร์ติ้งของ $IC_{2/1}$ LM1458 เพื่อเปรียบเทียบกับแรงดันไฟตรงที่ได้มาจาก เอาท์พุทขา 7 ของ $IC_{2/2}$

เอาท์พุทที่ขา 1 ของ $IC_{2/1}$ มีลักษณะเป็นสัญญาณพัลส์ความถี่ 100 Hz โดยความกว้างของพัลส์นี้มีการเปลี่ยนแปลง โดยขึ้นอยู่กับระดับแรงดันไฟตรงที่นำมาเปรียบเทียบที่ขาอินเวอร์ติ้งของ $IC_{2/1}$ ความกว้างของพัลส์มีผลต่อช่วงเวลาในการทริกให้ไตรแอคนำกระแสดังรูป 3.5.2



รูปที่ 3.5.2 แสดงการเปรียบเทียบของสัญญาณฟันเลื่อยกับแรงดันไฟตรง

ในส่วนของวงจรในรูปที่ 3.5.3 เป็นวงจรที่สร้างระดับแรงดันไฟตรงเพื่อเปรียบเทียบกับสัญญาณฟันเลื่อย เพื่อให้ความกว้างของพัลส์ที่ไปควบคุมให้ไตรแอคทำงานมีความกว้างต่างกันไป โดยขึ้นอยู่กับแรงดันไฟตรงที่ได้จากวงจรในส่วนนี้นั่นเอง



รูปที่ 3.5.3 วงจรส่วนสร้างแรงดันไฟตรงเปรียบเทียบกับสัญญาณฟันเลื่อย

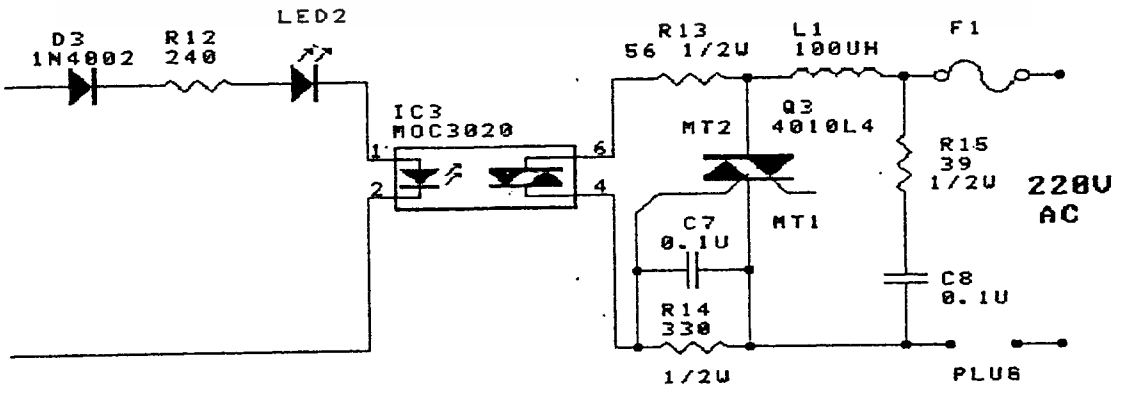
วงจรส่วนนี้จะรับสัญญาณควบคุมระดับความสว่างของหลอดไฟที่ออกมาจาก IC 74LS373 ซึ่งสามารถควบคุมความสว่างของหลอดไฟได้ทั้งหมด 8 ระดับ โดยถ้าต้องการความสว่างที่ระดับใด ก็ให้ที่ขานั้น ACTIVE ซึ่งจะมีระดับของแรงดัน +5 V ออกมาที่ขานั้น ในการทำให้เกิดแรงดันค่าต่าง ๆ เพื่อนำไปเปรียบเทียบกับสัญญาณฟันเลื่อยนั้น ทำได้โดยใช้หลักการของการแบ่งแรงดัน โดยการให้ค่า R_{17} - R_{24} มีความต้านทานแตกต่างกัน แรงดันที่ปรากฏที่ขา 5 ของ $IC_{2/2}$ จึงเกิดจากการแบ่งแรงดันระหว่าง R_8 กับ R_{17} - R_{24} ทั้งนี้ขึ้นอยู่กับว่าขาใดของ IC 4017 ทำงาน(ACTIVE)อยู่ หรือเลือกระดับความสว่างใดนั่นเอง ดังนั้นระดับแรงดันที่ขา 5 ของ $IC_{2/2}$ จึงแตกต่างกันไปตามระดับความสว่างที่เลือก

ระดับแรงดันที่ได้จากการแบ่งแรงดันจะเข้าที่ขานอนอินเวอร์ติ้ง หรือขา 5 ของ $IC_{2/2}$ ซึ่งเป็นออปแอมป์ การต่อออปแอมป์ในรูปนี้ เป็นการต่อแบบวงจรขยายไม่กลับขั้ว โดยมีอัตราขยายเป็น

$$\text{อัตราขยาย} = 1 + R_{10} / R_9$$
$$= 1 + 10k / 300$$
$$= 34.33$$

เอาท์พุทที่ได้ที่ขา 7 จะต่อเข้ากับขา 2 ซึ่งเป็นขาอินเวอร์ติ้งของ $IC_{2/1}$ เพื่อใช้ในการเปรียบเทียบกับสัญญาณฟันเลื่อย

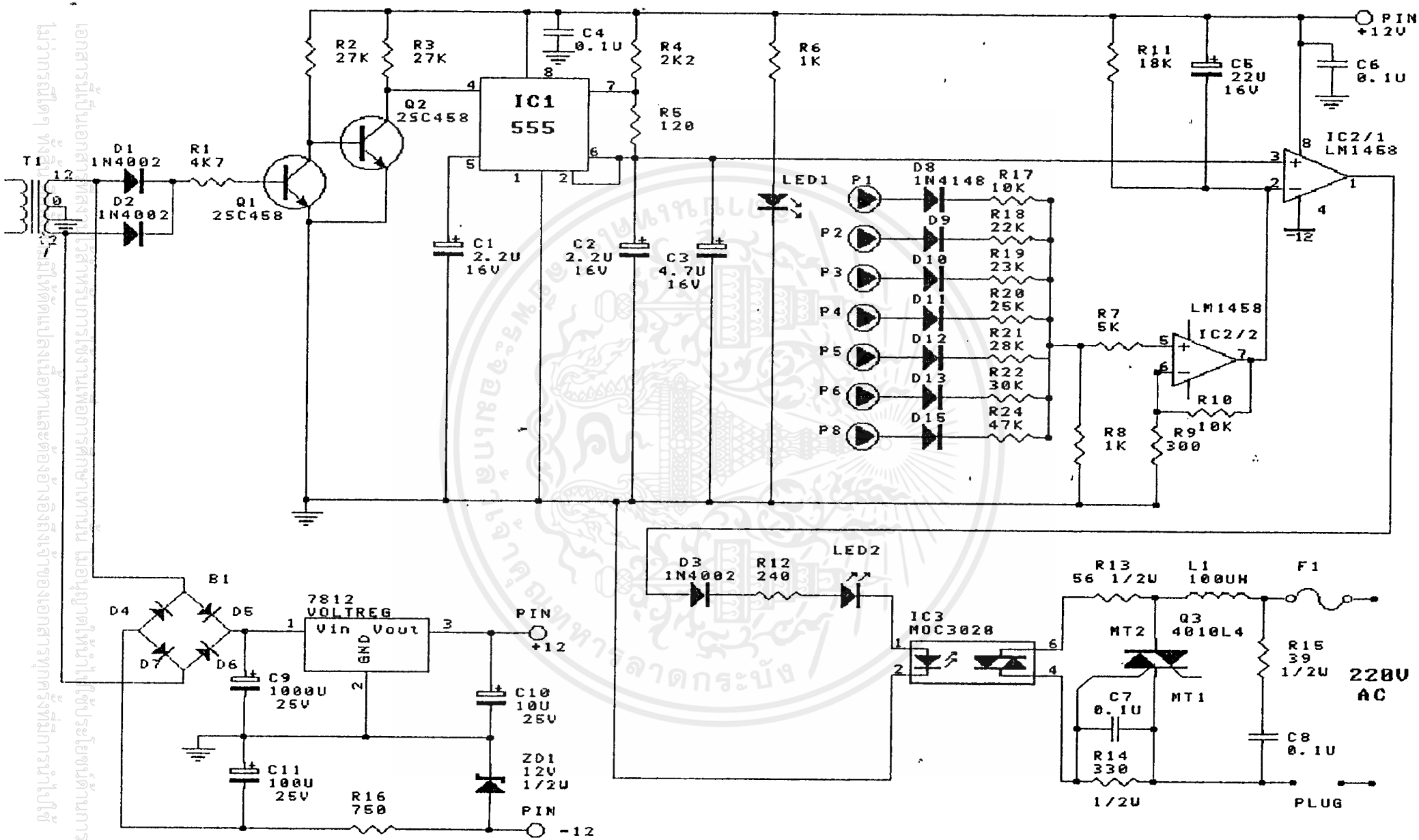
รูปที่ 3.5.4เป็นวงจรที่ใช้ออปโต้คัปเปลอร์(opto-coupler)ในการควบคุมวงจรกำลัง ซึ่งมีหลักการเช่นเดียวกับโซลิดสเตทรีเลย์



รูปที่ 3.5.4 วงจรที่ใช้ออปโตคัปเปลอร์ในการควบคุมวงจรกำลัง

IC_3 MOC 3020 เป็นออปโตไอโซเลเตอร์(opto-isolator) ซึ่งจะแยกวงจรควบคุมการทำงานด้านแรงดันต่ำออกจากวงจรกำลังโดยเด็ดขาด ภายในตัว MOC 3020 มี LED และ โฟโต้ไตรแอค

ด้าน LED มีตัวต้านทาน R_{12} ทำหน้าที่จำกัดกระแสที่ไหลผ่านไม่ให้เกินค่าสูงสุดที่ LED ในออปโต้ไอโซเลเตอร์ จะทนได้ คือ 60 mA D_3 เป็นตัวป้องกันความเสียหายของ LED และ $IC_{2/1}$ จากแรงดันกลับขั้ว เมื่อ LED ใน IC_3 ทำงาน ทำให้โฟโตไตรแอคภายในนำกระแส มีผลให้ขาเกทไตรแอค Q_3 ถูกทริก Q_3 จึงนำกระแส ทำให้โหลดทำงาน R_{13} และ R_{14} เป็นตัวแบ่งกระแสที่ไหลผ่านเกทของไตรแอคไม่ให้สูงเกินไป C_7 เป็นตัวป้องกันการทำงานผิดพลาดจากสัญญาณรบกวนจากภายนอก R_{15} และ C_8 เป็นตัวป้องกันทรานเซี้ยนต์จากการเพิ่มแรงดันของโหลดอย่างรวดเร็ว ซึ่งจะทำให้ไตรแอคเสียได้ นอกจากนี้ C_8 ยังเป็นตัวลดสัญญาณรบกวนและแรงดันกระชากในสายไฟด้วย R_4 เป็นตัวจำกัดกระแสที่ไหลผ่าน C_8


T1
D1 1N4002
D2 1N4002
R1 4K7
Q1 2SC458
R2 27K
R3 27K
Q2 2SC458
C4 0.1U
IC1 555
R4 2K2
R5 120
R6 1K
C1 2.2U 16V
C2 2.2U 16V
C3 4.7U 16V
LED1
P1
P2
P3
P4
P5
P6
P8
D8 1N4148
D9
D10
D11
D12
D13
D15
R17 10K
R18 22K
R19 23K
R20 25K
R21 28K
R22 30K
R24 47K
R11 18K
C5 22U 16V
C6 0.1U
PIN +12V
IC2/1 LM1458
R7 5K
IC2/2 LM1458
R10
R8 1K
R9 10K 300
B1
D4
D5
D6
D7
7812 VOLTREG
Vin Vout GND
PIN +12
C9 1000U 25V
C10 10U 25V
C11 100U 25V
ZD1 12V 1/2U
R16 750
PIN -12
D3 1N4002
R12 240
LED2
IC3 MOC3020
R13 56 1/2U
L1 100UH
F1
MT2
Q3 4010L4
MT1
R15 39 1/2U
C7 0.1U
C8 0.1U
R14 330
1/2U
220V AC
PLUG

วิเคราะห์และสรุปผล

จากการทำปริญญานิพนธ์นี้พบว่า การนำข้อมูลดิจิตอลจากคอมพิวเตอร์ไปควบคุมการทำงานของอุปกรณ์ไฟฟ้านั้น ต้องทำการกำหนดสัญญาณต่าง ๆ ที่ใช้ในการส่งข้อมูลดังนี้

1. ความถี่คลื่นพาห์ ให้มีค่าสูงกว่าความถี่ไฟบ้าน ซึ่งใช้ประมาณ 200 kHz

2. ความถี่ข้อมูล ที่ใช้มีค่าประมาณ 4.167 kHz

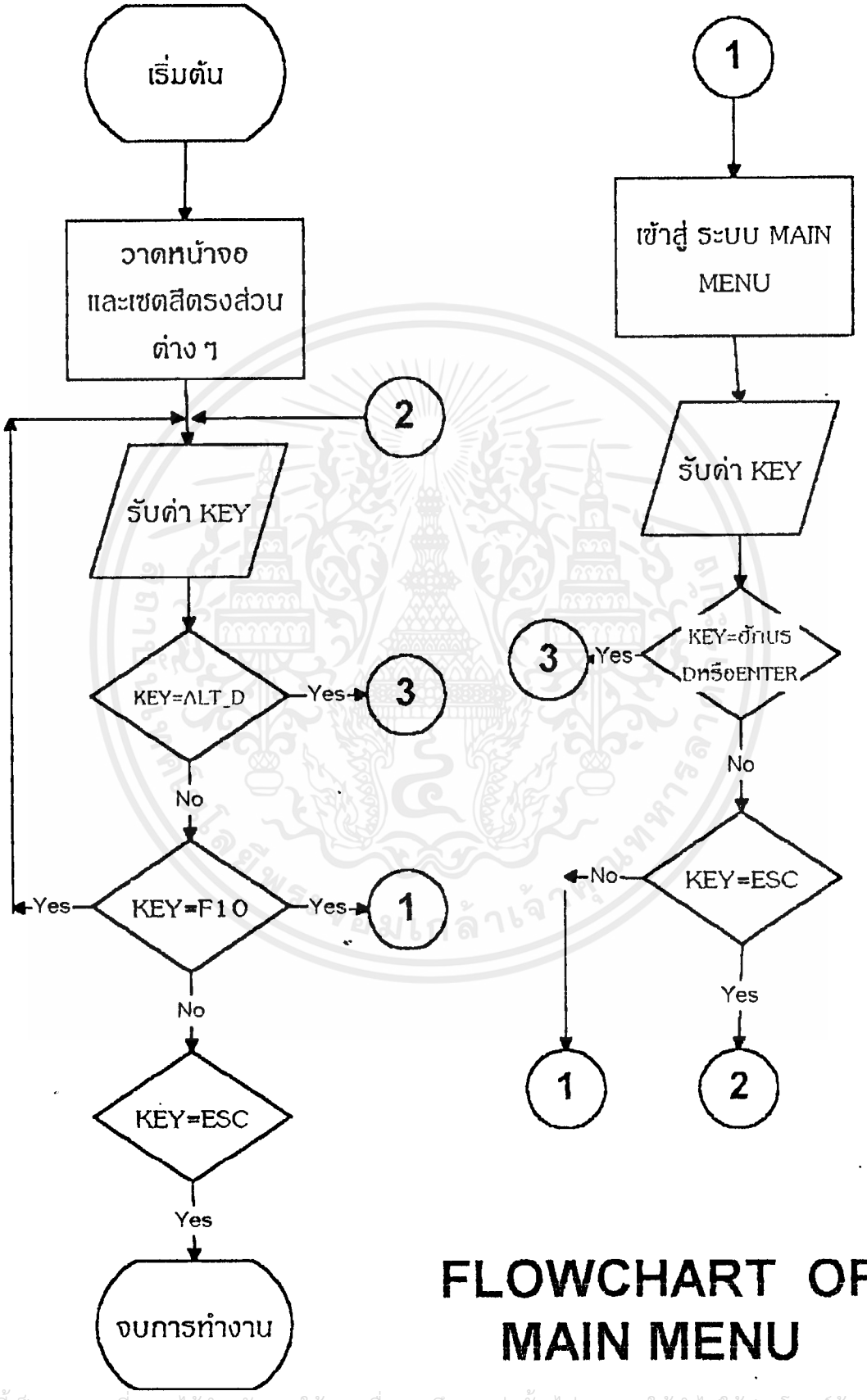
3. กำลังการส่งในกรณีที่เป็นความถี่สูง จะใช้หม้อแปลงไฟลง(step down transformer) เนื่องจากวงจรขยายความถี่สูง ความถี่ที่สามารถส่งได้ขึ้นอยู่กับค่าความต้านทานและค่าความจุของคาปาซิเตอร์ โดย f= 1/RC จึงจำเป็นต้องใช้หม้อแปลงไฟลง

ในส่วนของภาครับสัญญาณที่ได้จากวงจรมอดูเลเตอร์ยังมีรูปร่างที่ไม่เหมาะสมที่จะส่งไปเป็นอินพุทให้กับ UM3750 จึงต้องมีการปรับปรุงรูปร่างของสัญญาณทั้งทางขนาดและรูปร่าง โดยจะต้องทำให้ได้รูปสัญญาณเป็นดิจิตอล 5 โวลท์ ก่อนเข้าไปเป็นอินพุทให้กับ UM3750 ถ้าแอดเดรสที่ส่งมาตรงกับแอดเดรสของภาครับนั้น UM3750 ก็จะรับรู้และส่งสัญญาณไปยังวงจรส่วนอื่น ๆ ในภาครับ เพื่อควบคุมการทำงานของอุปกรณ์ปลายทาง ในขณะเดียวกันก็ส่งสัญญาณที่รับมาถูกต้องกลับไปให้ภาคส่งรับรู้ว่าข้อมูลที่ส่งมานั้นไปถึงปลายทางหรือไม่ ในส่วนของซอร์ฟแวร์จะต้องทำการดีเลย์เวลาช่วงหนึ่งหลังจากส่งข้อมูลไปแล้ว ถ้าไม่มีข้อมูลตอบกลับเมื่อถึงเวลาที่ดีเลย์นั้นแล้ว ให้เข้าใจว่าข้อมูลส่งไปไม่ถึง ก็ให้ส่งข้อมูลเก่าอีกครั้ง

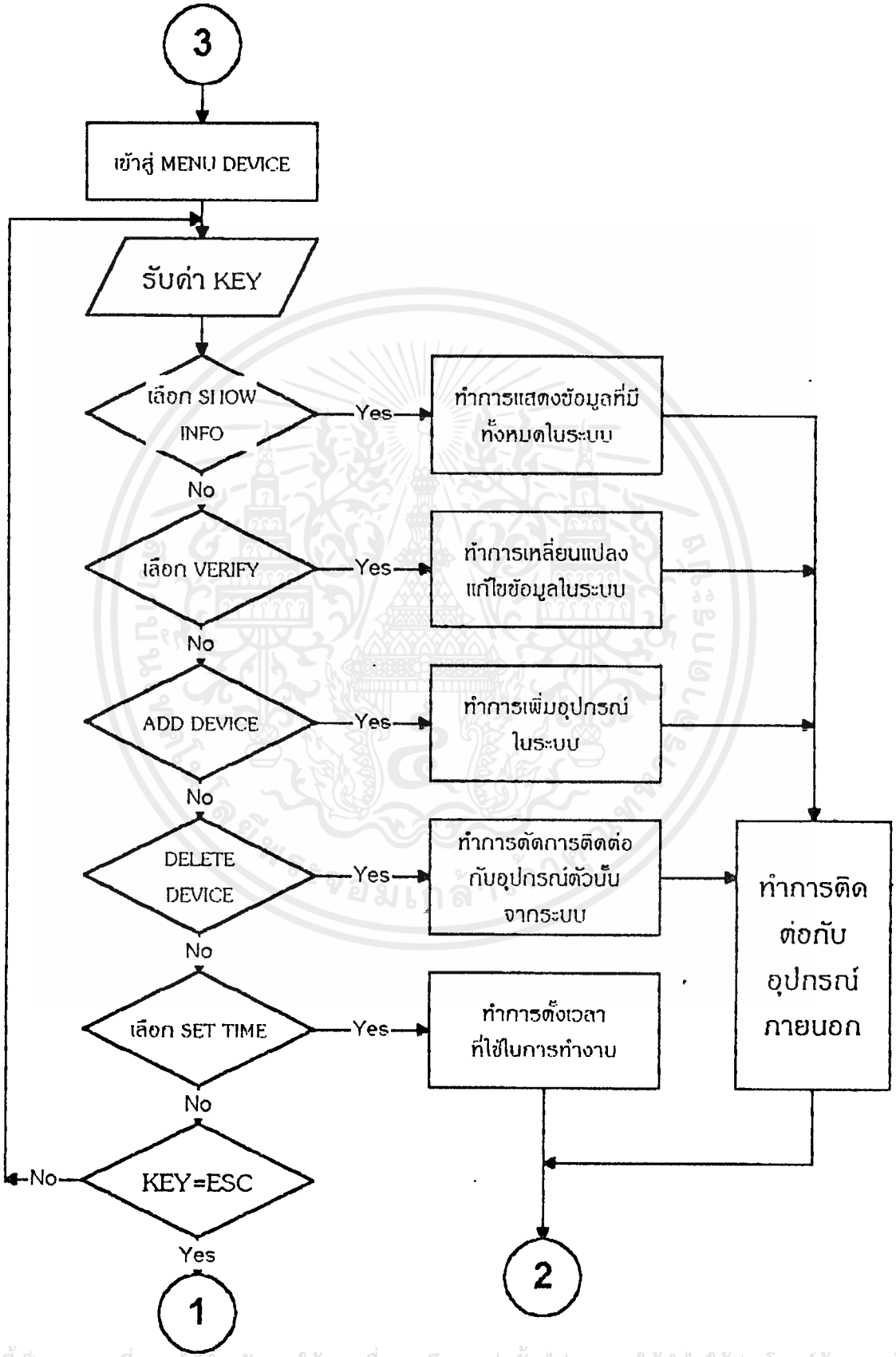
สำหรับผลการทำงานพบว่า ถ้าส่งข้อมูลที่เป็นอนาลอก (ไม่ใช่ส่งจากคอมพิวเตอร์ แต่อาจเป็นสัญญาณเสียง) พบว่าสามารถรับข้อมูลได้ดีกว่าข้อมูลดิจิตอล เนื่องจากว่าเมื่อความถี่ของสัญญาณอนาลอกคลาดเคลื่อนไป หรือสัญญาณที่รับมาไม่เสถียร ก็ยังคงรับสัญญาณได้ พอที่จะรับรู้ได้ แต่ถ้าเป็นสัญญาณดิจิตอลจำเป็นอย่างยิ่งที่จะต้องทำให้สัญญาณที่รับมานั้นนิ่ง เพื่อให้ UM3750 ตรวจรับสัญญาณได้ ดังนั้นจึงจำเป็นต้องตกแต่งสัญญาณก่อนเข้าภาครับ เพื่อให้สัญญาณนั้นเสถียร

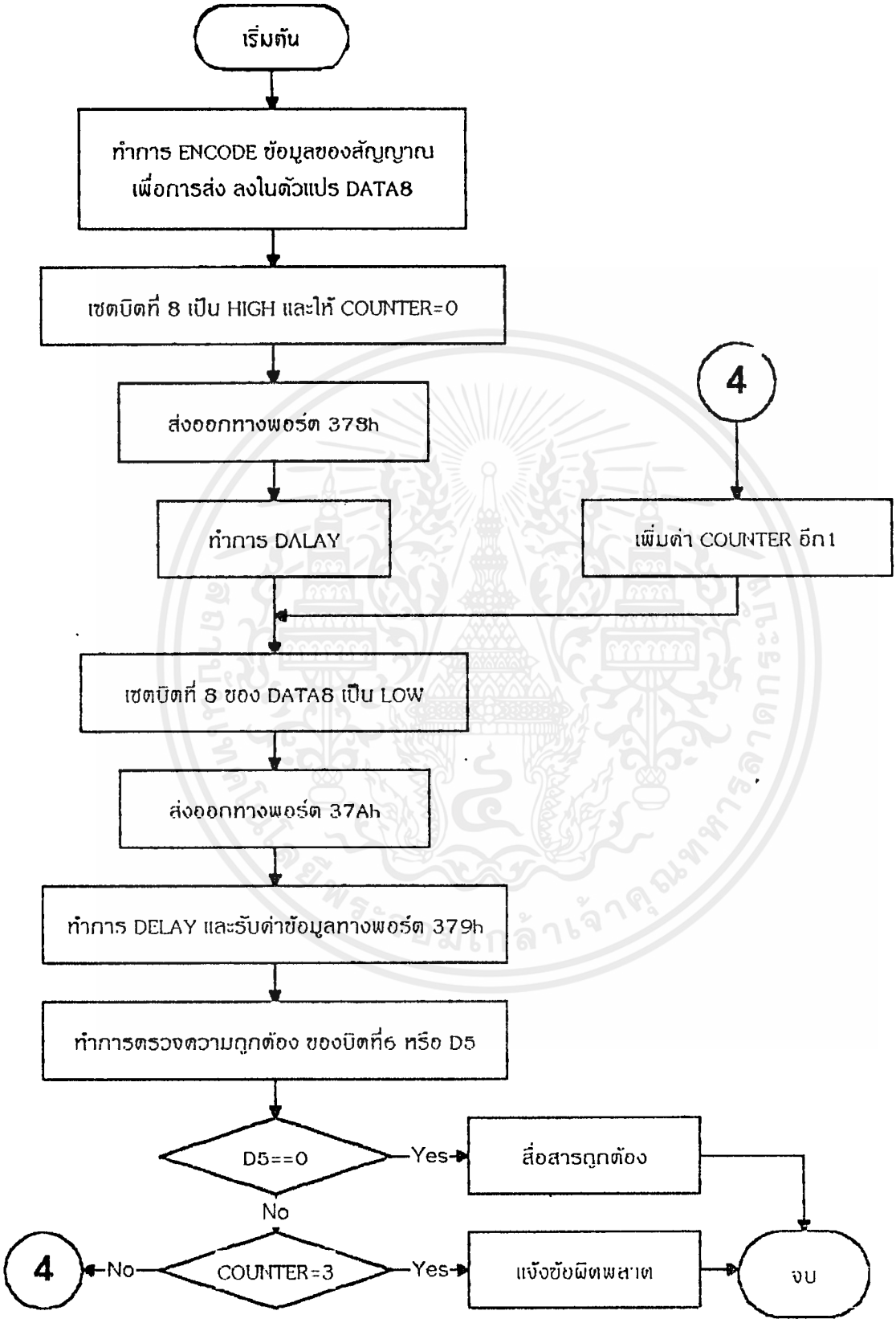
นอกจากนั้นแล้วในการรับ เนื่องจากสัญญาณที่เข้าที่ภาครับไม่เสถียร จึงจำเป็นต้องมีการจูนเพื่อให้รับสัญญาณได้ ซึ่งจากเหตุดังกล่าวจึงควรมีการปรับปรุงวงจรในส่วนนี้ให้ดีเพื่อให้มีประสิทธิภาพในการใช้งานได้เต็มที่

ดังนั้นสำหรับปริญญานิพนธ์นี้เป็นการศึกษาโครงสร้าง ลักษณะการทำงานที่เป็นไปได้ในทางปฏิบัติจริง เพื่อเป็นแนวทางในการปรับปรุงแก้ไข เพื่อสามารถนำไปใช้ได้จริงในชีวิตประจำวัน และให้ทันกับเทคโนโลยีที่เจริญก้าวหน้า และเพื่อเป็นพื้นฐานในการทำให้ดีขึ้นต่อไปในอนาคต



FLOWCHART OF MAIN MENU





โปรแกรม Shoot.c ซึ่งเป็น โปรแกรมหลัก

```
#include <conio.h>
#include <mem.h>
#include <bios.h>
#include <dos.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <ctype.h>
#include "popup.h"
#include "mouse.h"
#define  screen (*screen_ptr)
typedef texel screen_array[25][80];
screen_array far *screen_ptr = (screen_array far *)0xb8000000L;

#define SHADOW  0x07
#define MAXLEN  40  /* maximum of text of record in link-list */
#define LVLDIGIT  2
#define   MAX(a,b)   ((a) > (b) ? (a) : (b))
#define   MIN(a,b)   ((a) < (b) ? (a) : (b))
#define INTR 0X1C    /* The clock tick interrupt */
#define ALT_PRESS -777
 /************** USE IN LINK HARDWARE Tx|Rx MODE ****************/
#define  DELAY_TIME  300   /* milisecond */
#define  O_P_PORT8  0x0378    /* output port 8 bit (level and mode) */
#define  O_P_PORT4  0x037A    /* output port 4 bit (device)  */
#define  I_PORT379  0x0379    /* input port 3bd */
#define  I_PORT378  0x0378    /* input port 3bc */
#define  RECEIV_OK  888       /* O.K. when RECIEPTION is already */
#define  BAD_TRANS  777       /* BAD if not-already */
#define  show_cur()  cursor(1)
#define  hide_cur()  cursor(0)

enum typemode { Rx, Tx };
char *tmode[] = { "Reception", "Transmition" };
  /*********************************************************************/
    /**************** Save old interrupt handler ************/
void interrupt (*oldhandler)();
```

```
int counter=0;
char hour_char[3];
char min_char[3];
   /*********************************************************/
/****************** DEFINE FOR LINK LIST OF RECORD *****************/
struct device {
   int number;
   char *name;
   char *comment;
   int level;
   struct device *pre_p;
   struct device *next_p;
};
int record_no = 0;
int curr_page = 0;
enum act_posn { top=0, bottom };
enum act_posn active_flag = top;
int top_have = 0;
int bottom_have = 0;  /***** check for record have *****/
struct device *head;
struct device *current;
struct device *last;
#define MAXNAME  15
#define COMMENT  20
#define FAULTLVL  0
#define GET_VALUE  -7   /*** USE in function get_put_int to get_value **/
/****************** USE IN TRANS-RECIEV ************************
char nm[MAXNAME];
char cmmnt[COMMENT];
struct device compare_device;  ******************/
struct order {         /**** choice of menu selected **********/
   int first;
   int second;
};
struct order result;
windesc *state2_w;
/*************************************************************************/
char *title = " KRIT ";
wincolors slctcolor = {1, 0x30, 0x30, 0x30, 0x03 };
/*******      The 3rd of popcolor is border when It's ACTIVE    ********/
wincolors popcolor  = {2, 0x17, 0x1f, 0x17, 0x4f };
wincolors state2_color = {0, 0x00, 0x71, 0x00, 0x17 };
int choice_attr=0x03, hot=0x04, old=0x30;       /* ALL ATTRIBUTE */
```

```
int schoice_flag=0;
int trigmenu_flag=0;
int num_my_menu;
int start;
   /********* USE IN FUNCTION REPORT OF DEVICE *****************/
enum report_flag { GETVALUE , SHOWVALUE, VERIFY };
typedef enum report_flag report_type;

struct time t;
windesc *active_top;
windesc *active_bottom;
char *my_menu[] = {
   "Devices" ,
   "Orange" ,
   "Grape" ,
   "Mango" ,
   ""
};
char letter[] = { 'd','o','g','A' };
char *test1[] = {
   "System Information" ,
   "Verify Device" ,
   "Add Device" ,
   "Delete Device" ,
   "-" ,
   "Set Time" ,
   "Witun watunou" ,
   ""
};
char lettest1[] = { 'I','V','A','D','-','S','U' };  /* Care */
char *test2[] = {
   "Microsoft" , "Borland" ,
   "-" , "Turbo Vision" ,
   "-" , "IBM PC" ,
   "Quick Basic" , "Haili" ,
   "Maxell" , ""
};
char lettest2[] = { 'M','b','-','t','-','p','q','h','A' };
char *test3[] = {
   "Violet" , "Blue" ,
   "Orange" , "Sunset Yellow" ,
   "-" , "Green" ,
   "Pink" , "Black" ,
```

```
    ""
};
char lettest3[] = { 'Y','b','R','y','-','G','p','K' };
char *test4[] = {
    "Black Lebel"   , "Chivas Regal"    ,
    "SangTip"       , "-"               ,
    "SingHa"        , "Red Lebel"       ,
    "-"             , "Passport"        ,
    "Regensy"       , "Coca-Cola"       ,
    "Blue-Fanta"    , ""
};
char lettest4[] = { 'b','c','s','-','l','d','-','p','r','l','u' };
  /****************** USE IN DISPLAY MENU-GET ******************/
void yee(void);
float menu_disp(void);
int display(char *menu[]);
void first_h_menu(char *menu[], char letter[], int k);
int sub_first_h(int choice, char *menu[]);
void trigmenu(char *menu[], char letter[], int trig_hot_flag);
void schoice(char *menu[], char letter[], int choice, int flag);
int get_slct(char *select[], char let_ter[],int offset);
void disp_entry(windesc *w, char *entry[], int entryno, int hilite, int x, int y, unsigned char
ch);
void alt_service(int key);
  /*************************************************************/
  /****************** USE IN ATRIBUTE ******************************/
windesc *showselect(char *select[], char let_ter[], int offset);
void change_attr(windesc *w, int entryno, int hilite, int x, int y, unsigned char ch);
void showhot(windesc *w,int x,int y, unsigned char ch);
void show_status1(int line, unsigned char attr);
void hideselect(windesc *w);
void background(unsigned char attr);
void slct_active(windesc *w, int flag);
void make_shadow(windesc *w);
  /*************************************************************/
  /****************** USE IN TOOL OF DEVICE ********************/
void init_list(void);
struct device *add_device(void);
void sub_delete(int rec_num);
struct device *show_record_list(void);
void device_report(char *p, struct device *dev, int *gname, int *gcom, report_type kind);
void write(windesc *w, struct device *p, int hilite);
void func_switcher(void);
```

```
int sub_alt(int alt_key);
  /*************************************************************/
  /*************** USE IN SHOW STATUS IN PAGE ***************/
int chinstr(char *p, char c);
int maxoflen(char *menu[]);
int get_put_str(int i, int limit, char *string);
int get_put_int(int value, int limit_digit);
void prt_cur_page(void);
int inpage(int rec_no, int pagenum);
void show_page(int pagenum);
void pup_or_pdn(void);
void status2(void);
int maxpage(void);
int top_bot_active(void);
void free_record(struct device *dv);
  /*************************************************************/
  /****************** USE IN SHOW TIME ************************/
int count(char *menu[]);
int done1(void);
int done2(void);
void interrupt handler(void);
void show_time(void);
void setting_time(void);
 /*************************************************************/
 /********************** USE IN TRANS-RECEIV MODE *****************/
void setmode(enum typemode mode, unsigned char *data);
unsigned char l_encode(int level);
unsigned char d_encode(int device);
int decode_5( unsigned char data, enum typemode *mode);
unsigned char transmit(struct device *instru);
void cursor(int flag);
void bin_disp(unsigned char byte);
void set_m_receiv(unsigned char *data_8);
int receive(unsigned char *data, enum typemode *mode);
void connect(struct device *instru);
 /*************************************************************/

/*********************************************************************
**********          MAIN FUNCTION IS START HERE          ************
*********************************************************************/
main()
{
   init_win();
```

```
    init_list();
    menu_disp();
    getch();
 }
 float menu_disp(void)
 {
    int i, j, k, maxpg;
    float c;
    struct device *dv;

    textattr(0x17); clrscr();
    for (i=0; i<79; i++) cprintf("Thisisthetestofscreen images");
    gotoxy(1,2); getch(); cprintf(" %f ",(c=clock()/CLK_TCK)); gotoxy(1,1);
    textattr(0x47);  cprintf(title);

    show_status1(2, 0x51);
    show_status1(3, 0x71);
    show_status1(23,0x71);
    show_status1(24,0x31);
    show_status1(25,0x51);
    background(0x67);textattr(0x70);
    gotoxy(5, 17); cprintf("  P A G E  ");
    gotoxy(5, 18); cprintf(" NO.<=> 0  ");
    ต- ~ +@    ~ ~ ~ ~ ~ _ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ฮ~ ~ .~ Wบ
 cprintf(" Time =>        .m. ");
   state2_w = draw_win(5, 22, 60, 1,"", popup, &state2_color);
   status2();
   active_top = draw_win(20, 5, 60, 9, " Active_TOP ",popup,&popcolor);
   active_bottom = draw_win(20,14,60, 9, " Active_BOTTOM ",popup,&popcolor);
   slct_active(active_top, 1);
   show_time();
   start = strlen(title);
   result.first = -1; result.second = -1;
        /****************************************************/
   /* save the old interrupt vector */
   oldhandler = getvect(INTR);
   /* install the new interrupt handler */
   setvect(INTR, handler);
       /****************************************************/
   slct_win(base_win); gotoxy(strlen(title)+1,1);
   num_my_menu = display(my_menu);
   yee();
   do {
```

```
switch (k) {
    case F10KEY:
      trigmenu(my_menu, letter, 1);
      schoice(my_menu, letter, 0, 1);
      first_h_menu( my_menu, letter, 0);
    break;
    case ALT_PRESS:
      trigmenu(my_menu, letter, 1); k=0;
      while ( (bioskey(2)&0x08) )
        if ( !(sub_alt( (k=bioskey(1)) )) && k!=0 ) {
            k=bioskey(0);
            sound(200); delay(100); nosound();
        } else if (k!=0) {
            k=bioskey(0);
            break;
        }
      if (k!=0) {
        alt_service(k);
      } else {
       trigmenu(my_menu, letter, 1);
        schoice(my_menu, letter, 0, 1);
        first_h_menu( my_menu, letter, 0);
      }
    break;
    case PGDNKEY:
      if ( ++curr_page > (maxpg=maxpage())-1 )
        curr_page = maxpg-1;
      else
        show_page(curr_page);
    break;
    case PGUPKEY:
      if ( --curr_page < 0 )
        curr_page = 0;
      else
        show_page(curr_page);
    break;
    case SHFT_F3:
      show_page(0);
    break;
    case TABKEY:
      if ( active_flag == top ) {
        if ( bottom_have == 1 ) {
            slct_active(active_top, 0);
```

```
                slct_active(active_bottom, 1);
                active_flag = bottom;
            }
          } else {
            if ( top_have == 1 ) {
                slct_active(active_bottom, 0);
                slct_active(active_top, 1);
                active_flag = top;
            }
          }
        break;
        default: ;
      }
    } while (k!=ESCKEY);
      /*************************************************/
    /* reset the old interrupt handler */
    setvect(INTR, oldhandler);
      /*************************************************/
    return (c);
}
void first_h_menu(char *menu[], char letter[], int k)
{
    int i, offset, choice=0, key=0;
    windesc *w;
    char **slct_main, *slct_sub;
    char ch;
    /************ install compare_buffer in TRANS-RECEIV mode ******
    compare_device->name = nm;
    compare_device->comment = cmmnt;
    ***************************************************************/
    for (i=0,offset=0; i<choice; i++)  offset += strlen(menu[i])+2;
    i = num_my_menu;
    do {
      do {
        if (k==0 && (key==ESCKEY || key==0))   k = done2();
        else if ( key==RIGHTKEY || key==LEFTKEY ) {
          if (key==RIGHTKEY)  k=RIGHTKEY;
            else if  (key==LEFTKEY)  k=LEFTKEY;
        } else if ( k==RIGHTKEY || k==LEFTKEY )  k=0;
        else if ( sub_alt(k) && key==0)  k = done2();

        ch = (char)lo(k);
        if (  (ch>='a' && ch<='z') || (ch>='A' && ch<='Z')  ) {
```

```
        for (i=0; i<num_my_menu && letter[i]!=toupper(ch) &&
letter[i]!=tolower(ch); i++);
        if (i==num_my_menu)
          { putch('\a'); k=0; break; }    /* not found */
        else                              /* found */
          { schoice(menu,letter,choice,0);
            choice = i;  schoice_flag = choice+1; }

      } else if (k!=RIGHTKEY && k!=LEFTKEY && k!=ESCKEY && k!=ENTER
                 && k!=0) {
        putch('\a'); k=0; break;
      }
      schoice(menu,letter,choice,0);
      switch (k) {
        case LEFTKEY:
          choice = (choice-1<0) ? num_my_menu-1 : --choice;
          schoice_flag = choice+1;
          if (key == LEFTKEY) {
            trigmenu(menu,letter,0);
            schoice(menu,letter,choice,1);
            key = sub_first_h( choice, menu );
            trigmenu(menu,letter,1);
           if (key==ESCKEY)  k=0;
          }
        break;
        case RIGHTKEY:
          choice = (choice+1>=num_my_menu) ? 0 : ++choice;
          schoice_flag = choice+1;
          if (key == RIGHTKEY) {
            trigmenu(menu,letter,0);
            schoice(menu,letter,choice,1);
            key = sub_first_h( choice, menu );
           trigmenu(menu,letter,1);
           if (key==ESCKEY)  k=0;
          }
        break;
        case ESCKEY:
          trigmenu(menu,letter,0);
        break;
        case ENTER:
          trigmenu(menu,letter,0);
          schoice(menu,letter,choice,1);
          key = sub_first_h( choice, menu );
```

```
                trigmenu(menu,letter,1);
                 if (key==ESCKEY)  k=0;
              break;
              default:
                 if (i<num_my_menu)  {
                     trigmenu(menu,letter,0);
                     schoice(menu,letter,choice,1);
                     key = sub_first_h(choice, menu);
                     trigmenu(menu,letter,1);
                  i = num_my_menu;
                     if (key==ESCKEY)  k=0;
                 } else k=0;
              break;
           }   /* end of switch loop */
           schoice(menu,letter,choice,1);
      } while ( k!= ESCKEY && key!=ENTER );
   } while ( k!=ESCKEY && key!=ENTER);
   schoice(menu, letter, choice, 0);
   trigmenu(menu,letter,0);
   if (key==ENTER) {
      result.first = choice;
      func_switcher();
      slct_win(base_win);  textattr(0x24);
      cprintf("\r RETURN ORDER %d %d",result.first,result.second);
   }
}
int sub_first_h(int choice, char *menu[])
{
   int j, offset, key;
   char **slct_main, *slct_sub;
   switch (choice) {
      case 0:
           slct_main = test1;
           slct_sub  = lettest1;
      break;
      case 1:
           slct_main = test2;
           slct_sub  = lettest2;
      break;
      case 2:
           slct_main = test3;
           slct_sub  = lettest3;
      break;
```

```
        case 3:
            slct_main = test4;
            slct_sub  = lettest4;
        break;
        default: ;
    }
    for (j=0,offset=0; j<choice; j++)  offset += strlen(menu[j])+2;
    key = get_slct(slct_main, slct_sub, offset+2);
    return (key);
}
int display(char *menu[])
{
   int i, num;
   textattr(old);
   for (num=0; *menu[num] != '\0'; num++)
     cprintf("  %s", menu[num]);
   for (i=wherex()-1; i<80; i++) {
     screen[0][i].ch = ' ';
     screen[0][i].attr = old;
   }
   gotoxy(1,2);
   return (num);
}
int count(char *menu[])
{
   int num;
   for (num=0; *menu[num] != '\0'; num++);
   return num;
}
void trigmenu(char *menu[], char letter[], int trig_hot_flag)
{
   int i, j, offset, charoff, len2;
   char *whereup=NULL, *wherelo=NULL, *p;
   windesc *w;
   j = start;
   i = 0;
   while ( (whereup=strchr(menu[i],toupper(letter[i]))) != NULL ||
           (wherelo=strchr(menu[i],tolower(letter[i]))) != NULL ) {
     if (whereup == NULL)
         offset = (int)(wherelo-menu[i]);
     else if (wherelo == NULL)
         offset = (int)(whereup-menu[i]);
     else if (whereup > wherelo)     /* both is true */
```

```
            offset = (int)(wherelo-menu[i]);
        else
            offset = (int)(whereup-menu[i]);
        j = j+offset+2;
        if (schoice_flag==i+1)
            screen[0][j].attr = (trig_hot_flag) ? (choice_attr&0xf0)|hot :        choice_attr
;
        else
            screen[0][j].attr = (trig_hot_flag) ? (old&0xf0)|hot : old;
        j += strlen(menu[i]);
        i++;
    }    /* end of while loop */
    trigmenu_flag = trig_hot_flag;
    if (i != num_my_menu ) { gotoxy(35,12); printf("Have ERROR"); exit(1); }
}
void schoice(char *menu[], char letter[], int choice, int flag)
{
    /*************************************************************
    ::::::::::::::: if flag is '1' then it's show a choice ::::::::::
    ::::::::::::::: else ('0') then hide that choice ::::::::::::::::
     *************************************************************/
    int i, j, len2, offset, charoff;
    for (i=0,offset=0; i<choice; i++)
        offset += strlen(menu[i])+2;
    len2 = strlen(menu[choice])+2;
    j = start+offset;
    while (len2--)
        screen[0][++j].attr =  (flag) ? choice_attr : old;
    if (trigmenu_flag) {
        charoff = chinstr(menu[choice], letter[choice]);
        j = start+offset+charoff+1;
        screen[0][j].attr =  (flag) ? (choice_attr&0xf0)|hot : (old&0xf0)|hot;
    }
    schoice_flag = (flag) ? choice+1 : 0 ;
}
int chinstr(char *p, char c)
{
    /*  find char-position in string and return position of character in string
        IF FIND_NOT_FOUND it's return -1    IF it's '-' character it's return (0)
*******/
    int i=0, slen, charoff = 1;
    char ch1, ch2;
    slen = strlen(p);
```

```
    do {
      ch1 = (char)toupper(c);
      ch2 = (char)tolower(c);
      if (*p!=ch1 && *p!=ch2)  { charoff++; p++; }
    } while ( *p!=ch1 && *p!=ch2 && i<slen );
    if (i==slen)
      return (-1);
    else if (*p == '-')
      return (0);
    else   return (charoff);
}
int get_slct(char *select[], char let_ter[], int offset)
{
    int i, num, key, charoff, entryno=1;
    char ch;
    windesc *w;

    w = showselect(select, let_ter, offset);
    num = i = count(select);
    do {  /* loop until ECSKEY is pressed */
      do {  /* make selection */
          charoff = chinstr(select[entryno-1], let_ter[entryno-1]);
          if (charoff > 0) {
            change_attr(w, entryno, 1,  charoff+1, entryno, let_ter[entryno-1]);
            key = done2();
           ch = (char)lo(key);
            if (  (ch>='a' && ch<='z') || (ch>='A' && ch<='Z')  ) {
              for (i=0; i<num && let_ter[i]!=toupper(ch) &&  let_ter[i]!=tolower(ch);
i++);
              if (i==num) { putch('\a'); break; }
            }
            change_attr(w, entryno, 0, charoff+1, entryno, let_ter[entryno-1]);
          }
          switch (key) {
            case UPKEY:
              if (--entryno < 1) entryno = num;
            break;
            case DOWNKEY:
              if (++entryno > num) entryno = 1;
            break;
            case RIGHTKEY:
              hideselect(w);
              return (RIGHTKEY);
```

```
            case LEFTKEY:
               hideselect(w);
               return (LEFTKEY);
            case ESCKEY:
            case ENTER:
            break;
            default:
               if (i<num)  { entryno = i+1; i = num; key = ENTER; }
               else { putch('\a'); entryno = 1; }
            break;
         }
      } while ((key != CRKEY) && (key != ESCKEY));
      if (key == CRKEY) {
         hideselect(w);
         result.second = entryno-1;
         return (ENTER);
      }
   } while (key != ESCKEY);
   hideselect(w);
   return (ESCKEY);
}
int maxoflen(char *menu[])
{
   int i, num, len;
   num = count(menu);
   for (i=0,len=0; i<num; i++)
      len = MAX(len, strlen(menu[i]));
   return (len);
}
windesc *showselect(char *select[], char let_ter[], int offset)
{
   int i, j, num, y=2, entryno=1;
   int charoff,ip, temp1, temp2, wd, ht;
   windesc *w;
   j = start + offset - 1;
   num = count(select);
   ht = num+2;
   wd = maxoflen(select)+5;
   w = draw_win(j, y, wd, ht, "", popup, &slctcolor);
   for (i=0; i<num; i++) {
      if ( (charoff=chinstr(select[i],let_ter[i])) != -1) {
         if (*select[i] == '-') {
            screen[y+i][j-1].ch = 0xc3;  /* 195   */
```

```
            screen[y+i][j+wd-2].ch = 0xb4;  /*  196 */
            prtfstr(1, y+i-1, "\0xb4",w->wc.text_color,0-wd);
            entryno++;
          } else {
            disp_entry(w, select, entryno, 0, charoff+1, i+1, let_ter[i]);
            entryno++;
          }
        } else
          printf("ERROR in module _showselect");
    }  /* end of for loop */
    make_shadow(w);
    return(w);
}
void make_shadow(windesc *w)
{
    int temp1, temp2, ip, i, j;
    w->shadow = (char *)malloc( (w->ht*2) + (w->wd-1) );
    if (w->shadow == NULL)
      printf("ERROR in module _make_shadow");
    ip = 0; temp1 = w->xlr+2; temp2 = w->yul+w->ht;
    for (i=w->xlr; i<temp1; i++)
      for (j=w->yul; j<temp2; j++) {
          w->shadow[ip] = screen[j][i].attr;
          screen[j][i].attr = SHADOW;
          ip++;
      }
    j = w->ylr;  temp2 = w->xul+w->wd-1;
    for (i=w->xul; i<temp2; i++) {
      w->shadow[ip] = screen[j][i].attr;
      screen[j][i].attr = SHADOW;
      ip++;
    }
}
void hideselect(windesc *w)
{
    int i, j, ip, temp1, temp2;
    ip = 0; temp1 = w->xlr+2; temp2 = w->yul+w->ht;
    for (i=w->xlr; i<temp1; i++)
      for (j=w->yul; j<temp2; j++) {
          screen[j][i].attr = w->shadow[ip];
          ip++;
      }
    j = w->ylr;  temp2 = w->xul+w->wd-1;
```

```
    for (i=w->xul; i<temp2; i++) {
      screen[j][i].attr = w->shadow[ip];
      ip++;
    }
    free(w->shadow);
    rmv_win(w);
}
void showhot(windesc *w,int x,int y, unsigned char ch)
{
    int i, j;
    unsigned char old_at;
    i = w->yul+y-1;
    j = w->xul+x-1;
    if ( screen[i][j].ch != (char)toupper(ch) &&
         screen[i][j].ch != (char)tolower(ch) ) {
      printf("Have ERROR in module _showhot");
      return;
    }
    old_at = screen[i][j].attr;
    screen[i][j].attr = (old_at&0xf0)|hot;
}
void change_attr(windesc *w, int entryno, int hilite, int x, int y, unsigned char ch)
{
    int i, xi, yi, bar_color;
    if (hilite) bar_color = slctcolor.hilite_color;
      else      bar_color = slctcolor.text_color;
    yi = w->yul+entryno-1;
    xi = w->xul;
    for (i=0; i<w->wd-2; i++)
      screen[yi][xi+i].attr = bar_color;

    showhot(w, x, y, ch);
}
void disp_entry(windesc *w, char *entry[], int entryno, int hilite, int x, int y, unsigned char
ch)
{
    int i, xi, yi, bar_color;
    if (hilite) bar_color = w->wc.hilite_color;
      else      bar_color = w->wc.text_color;   /* slctcolor. */
    prtfstr(1, entryno, " %-*.*s", bar_color, 80,
            w->wd-2,w->wd-2,entry[entryno-1]);
    if (x!=0 && y!=0) showhot(w, x, y, ch);
}
```

```
void interrupt handler(void)
{
   /* increase the global counter */
   counter++;
   /* call the old routine */
   oldhandler();
}
int done1(void)
{
   int key;
   while ( !(bioskey(2)&0x08) && bioskey(1)==0 );
   if ( bioskey(2)&0x08 ) key = ALT_PRESS;
   else key = bioskey(0);
   if ( counter%1000 == 0 )
      show_time();
   return (key);
}
int done2(void)
{
   int key;
   while ( bioskey(1)==0 );
   key = bioskey(0);
   if ( counter%1000 == 0 )
      show_time();
   return (key);
}
void show_time(void)
{
   int i;
   gettime(&t);
   itoa(t.ti_hour, hour_char, 10);
   itoa(t.ti_min , min_char , 10);
   if ( hour_char[1] == '\x0' ) {
      hour_char[2] = '\x0';
      hour_char[1] = hour_char[0];
      hour_char[0] = '0';
   }
   if ( min_char[1] == '\x0' ) {
      min_char[2] = '\x0';
      min_char[1] = min_char[0];
      min_char[0] = '0';
   }
```

```
    screen[1][67].ch = hour_char[0];
    screen[1][68].ch = hour_char[1];
    screen[1][69].ch = ':';
    screen[1][70].ch =  min_char[0];
    screen[1][71].ch =  min_char[1];
    if (t.ti_hour>=0 && t.ti_hour<12) {
       screen[1][73].ch = 'a';
    } else {
       screen[1][73].ch = 'p';
    }
}
void setting_time(void)
{
    windesc *dialog, *old;
    char hour, min;
    old = curr_win;
    dialog = draw_win(CTRWIN,CTRWIN,40,5," SET_TIME ",popup, &slctcolor);
    make_shadow(dialog);
    cprintf("  Input number of time \r\n  hour => [..]");
    cprintf("\r\n  min  => [..]");
    gotoxy(12,2);  hour = get_put_int(GET_VALUE, 3);
    gotoxy(12,3);  min  = get_put_int(GET_VALUE, 3);
    if (hour<0 || hour>23 || min<0 || min>58) {
       clrscr();
       cprintf("\r\n       ERROR in SETTING TIME");
       done2();
       hideselect(dialog);
       slct_win(old);
       return;
    }
    t.ti_hour = hour; t.ti_min = ++min;
    t.ti_hund = 0; t.ti_sec = 0;
    settime(&t);
    show_time();
    hideselect(dialog);
    slct_win(old);
}
void show_status1(int line, unsigned char attr)
{
    int i;
    for (i=0; i<80; i++) {
       screen[line-1][i].ch = ' ';
       screen[line-1][i].attr = attr;
```

```
    }
  }
  void background(unsigned char attr)
  {
    int i, j;
    for (i=0; i<80; i++)
      for (j=3; j<22; j++) {
          screen[j][i].ch = '\xb0';    /* ALT-176  or \xb0H */
          screen[j][i].attr = attr;
      }
  }
  void slct_active(windesc *w, int flag)
  {
    int i, color;
    /********** IF flag = 1 then slct_active ******************
     *********  ELSE is unslct_active ************************/
    color = (flag) ? w->wc.text_color : w->wc.title_color;
    screen[w->yul-1][w->xul-1].attr = color;
    screen[w->yul-1][w->xlr-1].attr = color;
    screen[w->ylr-1][w->xul-1].attr = color;
    screen[w->ylr-1][w->xlr-1].attr = color;
    for (i=0; i<w->wd-2; i++) {
      screen[w->yul-1][w->xul+i].attr = color;
      screen[w->ylr-1][w->xul+i].attr = color;
    }
    for (i=0; i<w->ht-2; i++) {
      screen[w->yul+i][w->xul-1].attr = color;
      screen[w->yul+i][w->xlr-1].attr = color;
    }
    slct_win(w);
  }
  void init_list(void)
  {
    last = current = head;
  }
  struct device *add_device(void)
  {
    struct device *temp;
    int i, gname, gcom;
    temp = (struct device *)malloc(sizeof(struct device));
    temp->number = record_no;
    temp->name = (char *)malloc(MAXNAME);
    temp->comment = (char *)malloc(COMMENT);
```

```
    temp->pre_p = temp->next_p = NULL;
    if (temp == NULL)  { cprintf("CAN'T allocate memory"); return(NULL); }
    device_report(" ADD_DEVICE ", temp, &gname, &gcom, GETVALUE);
    if ( gname==-1 || gname==-2 ) {
       free_record(temp);
       return (NULL);
    }
    if ( gcom == -1 ) {
       free_record(temp);
       return (NULL);
    }
    if (temp->level == -1) {
       free_record(temp);
       return (NULL);
    } else {
          if (record_no==0)
             head = current = last = temp;
          else {
             last->next_p = temp;
             temp->pre_p  = last;
             last = temp;
          }
          record_no++;
          connect(last);
    }
     show_page(curr_page);
     status2();
     return (temp);
   }
}
void free_record(struct device *dv)
{
   free(dv->name);
   free(dv->comment);
   free(dv);
}
void sub_delete(int rec_num)
{
   windesc *dialog, *old_w;
   struct device *mark, *start_p;
   int c, i;
   if (rec_num > record_no || rec_num < 0) {
      cprintf("\r\nERROR in module DELETE_DEVICE."); return;
```

```
    }
    old_w = curr_win;
    dialog = draw_win(CTRWIN,CTRWIN,40,5,"",popup,&slctcolor);
    make_shadow(dialog);
    hide_cur();
    gotoxy(4,2); cprintf(" Do you sure to delete ? (Y/N) ");
    while ( (c=getch())!='y' && c!='Y' && c!='n' && c!='N' && c!='\1b' );
    if ( c=='y' || c=='Y' ) {
       gotoxy(4,2); cprintf("       Please waiting ...          ");
       i = 0; mark = head;
       if (rec_num > 0) {
           while (i<rec_num) {
              mark = mark->next_p;
              i++;
           }
           start_p = mark->next_p;
           mark->pre_p->next_p = mark->next_p;
           if (mark->next_p != NULL)  mark->next_p->pre_p = mark->pre_p;
           else last = mark->pre_p;
       } else if (rec_num == 0) {
           if (head->next_p != NULL) {
              head = head->next_p;
              head->pre_p = NULL;
              start_p = head;
           } else head = NULL;  /*** Have only one record to delete ***/
       }
       free_record(mark);
       start_p->number = i;
       record_no--;
       if ( record_no==curr_page*2 )
           if (--curr_page < 0)  curr_page=0;
       while ( i < record_no ) {
           start_p->number = i;
           start_p = start_p->next_p;
           i++;
       }
       show_page(curr_page);
    }
    show_cur();
    hideselect(dialog);
    status2();
    slct_win(old_w);
}
```

```
struct device *show_record_list(void)
{
   windesc *dialog, *old_w;
   struct text_info r;
   int i, x, y, key;
   int num_show=5;      /* how much to display number of record */
   int shift=5;    /* shift_right to make new window */
   old_w = curr_win;
   dialog = draw_win(CTRWIN,CTRWIN, MAXNAME+30,num_show+3, " DEVICE_LIST ",
                     popup, &sictcolor);
   make_shadow(dialog);
   hide_cur();
   if (head != NULL) {
      cprintf("  NUMBER  LEVEL     NAME  ");
      gettextinfo(&r);
      window(r.winleft +shift , r.wintop+1, r.winright, r.winbottom);
      current = head;
      for (i=1; i<num_show && current!=NULL; i++) {
         write(dialog, current, 0);
          current = current->next_p;
         putch('\n');
      }
      if (current!=NULL)  write(dialog, current, 0);
      x=1; y=1; gotoxy(x,y);
      current = head;
      do {
          write(dialog, current, 1);
         key = done2();
          write(dialog, current, 0);
          switch (key) {
             case UPKEY:
                --y;
                if (current!=head) {
                    current = current->pre_p;
                    if (y<1) insline();
                }
               if (y<1) y=1;
             break;
             case DOWNKEY:
                if (++y>num_show) {
                    y=num_show;
                }
                if (current->next_p != NULL) {
```

```
                current = current->next_p;
                  if (current->number >= num_show) {
                    if (current->number<record_no) {
                        cprintf("\n");
                    } else {
                        y=num_show;
                    }
                  }
                } else if (record_no<num_show)
                    y=record_no;
              break;
              default: ;
            }
            gotoxy(x,y);
        } while (key!=ESCKEY && key!=ENTER);
        gettextinfo(&r);
        window(r.winleft -shift, r.wintop-1, r.winright, r.winbottom);
    } else {
        gotoxy(5,num_show/2);
        cprintf(" NO record in record_list ");
        done2();
    }
    show_cur();
    hideselect(dialog);
    slct_win(old_w);
    if (key==ESCKEY)  return(NULL);
    else    return (current);
}
void write(windesc *w, struct device *p, int hilite)
{
    int bar_color;
    if (hilite) bar_color = w->wc.hilite_color;
      else     bar_color = w->wc.text_color;
    cprintf("\r %*s ", MAXNAME+15, " ");
    textattr(bar_color);
    cprintf("\r %d ", p->number);
    textattr(w->wc.text_color);
    cprintf("     %d      %s ", p->level, p->name);
}
void device_report(char *p, struct device *dev, int *gname, int *gcom, report_type kind)
{
    windesc *dialog, *old_w;
    int i, lname, lcom, llevel;
```

```
 old_w = curr_win;
 dialog = draw_win(CTRWIN,CTRWIN,40,8, p ,popup, &slctcolor);
 make_shadow(dialog);
 gotoxy(1,2);
 cprintf("\r    Device Name: ");
 putch('['); for (i=MAXNAME-1; i>0; i--) putch('.'); putch(']');
 cprintf("\r\n    Comment: ");
 putch('['); for (i=COMMENT-1; i>0; i--) putch('.'); putch(']');
 cprintf("\r\n    Level: ");
 putch('['); for (i=LVLDIGIT-1; i>0; i--) putch('.'); putch(']');
 if ( kind==GETVALUE ) {
    lname = 0; lcom = 0; llevel = GET_VALUE;
 } else {
    gotoxy(19,2); cprintf("%s", dev->name);
    gotoxy(15,3); cprintf("%s", dev->comment);
    gotoxy(13,4); cprintf("%d", dev->level);
    if ( kind==VERIFY ) {
        lname = strlen(dev->name);
        lcom =  strlen(dev->comment);
        llevel = dev->level;
    } else {      /** kind==SHOWVALUE ***/
        done2();
       hideselect(dialog);
        slct_win(old_w);
        return;
    }
 }
 gotoxy(19,2);
 *gname = get_put_str(lname, MAXNAME, dev->name);
 if ( *gname != -1 && *gname != -2 ) {
    gotoxy(15,3);
    *gcom = get_put_str(lcom, COMMENT, dev->comment);
    if ( *gcom != -1 ) {
        gotoxy(13,4);
        do {
          dev->level = get_put_int(llevel, LVLDIGIT);
         if (kind==VERIFY)  putch('\b');
          if ( (dev->level<1 || dev->level>7) &&
                dev->level!=-1 && dev->level!=-2 ) {
             putch('\a');
             putch('\b');putch('.');putch('\b');
          }
        } while ( (dev->level<1 || dev->level>7) && dev->level!=-1) ;
```

```
        }  /*** end of if COMMENT ****/
      }  /*** end of if NAME ***/
      hideselect(dialog);
      slct_win(old_w);
      show_page(curr_page);
    }
    int get_put_str(int i, int limit, char *string)
    {
      char ch;
      int first;
      if (i>limit) return(0);
      for (first=0; first<i; first++)  putch(string[first]);
      while ( (ch=getch())!='\r' && ch!='\x1b' )
        if ( i<limit-1 && ( isalnum(ch) || ch==' ' ) || ch=='\b' ) {
            if ( ch != '\b' ) {
              string[i] = ch;
              putch(ch);
              i++;
            } else {
              if (i>0) {
                putch(ch); putch('.'); putch(ch);
                i--;
              }
            }
        } else
            putch('\a');
      string[i] = '\x0';
      if ( ch=='\x1b' )
        return (-1);          /****** return -1 if ESC **********/
      else if ( ch=='\r' && i==0 )
        return (-2);          /***** return -2 if ENTER *********/
      else    return (i);
    }
    int get_put_int(int value, int limit_digit)
    {
      char digit[10], ch;
      int i=0;
      if (value != GET_VALUE) {
        itoa(value, digit, 10);
        while ( digit[i]!='\x0' && i<limit_digit ) { putch(digit[i]); i++; }
      }
      while ( (ch=getch())!='\r' && ch!='\x1b' )
        if ( (i<limit_digit-1) && (isdigit(ch)) || ch=='\b' ) {
```

```
            if ( ch != '\b' ) {
               digit[i] = ch;
               putch(ch);
               i++;
            } else {
               if (i>0) {
                  putch(ch); putch('.'); putch(ch);
                  i--;
               }
            }
         } else
            putch('\a');
         digit[i] = '\x0';
         if ( ch=='\x1b' )
            return (-1);            /****** return -1 if ESC **********/
         else if ( ch=='\r' && i==0 )
            return (-2);            /****** return -2 if ENTER *******/
         else
            return (atoi(digit));
      }
      void prt_cur_page(void)
      {
         gotoxy(1,2);
         cprintf("\r  Device No. => "); cprintf("%d", current->number);
         cprintf("\r\n  Name : "); cprintf("%s", current->name);
         cprintf("\r\n  Comment : "); cprintf("%s", current->comment);
         cprintf("\r\n  Level : "); cprintf("%d", current->level);
      }
      int inpage(int rec_no, int pagenum)
      {
        if ( pagenum == rec_no/2  )
            return (1);
         else
            return (0);
      }
      void pup_or_pdn(void)
      {
         int max;
         windesc *old_w;

         old_w = curr_win;  slct_win(base_win);
         textattr(0x30);  gotoxy(5, 20);
         max = maxpage();
```

```
    if (curr_page == 0)
      if (max == 1)    cprintf("          ");
      else  cprintf("   PgDn   ");
    else if (curr_page > 0 && curr_page < max-1 )
      cprintf(" PgUp/PgDn ");
    else                          /*** curr_page == max-1 ***/
      cprintf("   PgUp   ");
    gotoxy(1,2);
    slct_win(old_w);
}
int top_bot_active(void)
{
    return (curr_page*2+active_flag);
}
void show_page(int pagenum)
{
    int i=0;
    char pgchar[5];
    windesc *old_w;
    top_have = bottom_have = 0;  /***** CLEAR object in BOTH PAGES *****/
    slct_active(active_bottom, 0);
    slct_active(active_top, 1);
    active_flag = top;
    if ( pagenum >= 0 ) {
          old_w = curr_win;
          current = head;
          curr_page = pagenum;
          itoa(pagenum, pgchar, 10);
          while (pgchar[i]!='\x0') {
            screen[17][12+i].ch = pgchar[i];
            i++;
          }
          for(i=0; i<pagenum*2; i++)
            current = current->next_p;
            /***** NOW i==pagenum*2 *****/
          pup_or_pdn();                    /** PgUp or PgDn **/
          if (current!=NULL) {
            slct_win(active_top);
            clrscr();
            prt_cur_page();
            top_have = 1;          /***** Have record in TOP *****/
            slct_win(old_w);
            current = current->next_p;
```

```
            slct_win(active_bottom);
             clrscr();
             if (current!=NULL)  {
                prt_cur_page();
                bottom_have = 1;  /***** Have record in BOTTOM *****/
             }
            slct_win(old_w);
          } else if (head==NULL) {
            clrscr();
          }
    }
}
void status2(void)
{
   windesc *old_w;
   old_w = curr_win;
   slct_win(state2_w);
   cprintf("\r SYSTEM  have  %d  devices in Connection", record_no);
   slct_win(old_w);
}
int maxpage(void)    /* return max number of page */
{
   int i;
   for (i=0; !(inpage(record_no-1,i)) ; i++);
   return (++i);
}
void func_switcher(void)
{
   int gname, gcom;
   struct device *select_one;   /* A SELECT ONE IN LIST */
   switch (result.first) {
      case 0:
         switch (result.second) {
            case 0:
               show_record_list();
            break;
            case 1:
               select_one = show_record_list();
               if (select_one != NULL ) {
                  device_report(" VERIFY_DEVICE ", select_one, &gname,
                                 &gcom, VERIFY);
                  connect(select_one);
               }
```

```
            break;
            case 2:
               add_device();
            break;
            case 3:
               select_one = show_record_list();
               if (select_one != NULL ) {
                  device_report(" DELETE_DEVICE ", select_one, &gname,     &gcom,
0WVALUE);
                  sub_delete(select_one->number);
               }
            break;
            case 5:
               setting_time();
            break;
            default: ;
         }
      break;
      default: ;
   }
}
void alt_service(int key)
{
   int sub_key;
   sub_key = sub_alt(key);
   switch (key) {
      case ALT_D:
      case ALT_0:
      case ALT_G:
      case ALT_A:
         first_h_menu( my_menu, letter, sub_key);
      break;
      default :
         sound(300); delay(100); nosound();
         trigmenu(my_menu, letter, 0);
      break;
   }
}
int sub_alt(int alt_key)
{
   int i=0;
   int alt_array[] = {
      ALT_D, ALT_0, ALT_G, ALT_A
```

```
    };
    int alt_char[] = {
        D_KEY, 0_KEY, G_KEY, A_KEY
    };
    while (i<num_my_menu && alt_array[i]!=alt_key ) i++;
    if (i==num_my_menu)
        return 0;
    else
        return (alt_char[i]);
}
void setmode(enum typemode mode, unsigned char *data)
{
    if (mode == Tx)
        if (!(*data & 0x80))  *data |= 0x80;
    if (mode == Rx)
        if (*data & 0x80)  { *data <<= 1; *data >>= 1; }
}
unsigned char l_encode(int level)
{
    unsigned char data = 0x01;
    if (level>7 && level<1) {
        printf("\nHave ERROR in input LEVEL ");
        return(0);
    }
    data = data << (level-1);
    setmode(Tx, &data);
    return data;
}
unsigned char d_encode(int device)
{
    unsigned char data, temp;
    data = ~(unsigned char)device;
    data = data & 0x0b;
    temp = (unsigned char)device & 0x04;
    data = data | temp;
    return (data);
}
int decode_5( unsigned char data, enum typemode *mode)
{
    unsigned char flag;
    data = data & 0x30;
    if (data & 0x10)  *mode = Tx;
    else *mode = Rx;
```

```
    flag = (data >> 1) | data;
    if ( flag==0 )  return RECEIV_OK;
    else  return BAD_TRANS;
}
void bin_disp(unsigned char byte)
{
    int i;
    for (i=0; i<8; i++) {
      if (i==4)  putchar('-');
      if ( byte & (0x80>>i) )  putchar('1');
      else  putchar('0');
    }
}
unsigned char transmit(struct device *instru)
{
    unsigned char data8, data4;
    hide_cur();
    data8 = l_encode( instru->level );
    outportb(0_P_PORT8, data8);
    data4 = d_encode( instru->number );
    outportb(0_P_PORT4, data4);
    show_cur();
    return data8;
}
void cursor(int flag)
{
    union REGS inregs;
    inregs.h.ah = 0x01;
    inregs.h.ch = 0x12;
    if (flag) inregs.h.cl = 0x13;
    else inregs.h.cl = 0x00;
    int86(0x10, &inregs, &inregs);
}
void set_m_receiv(unsigned char *data_8)
{
    setmode(Rx, data_8);
    outportb(0_P_PORT8, *data_8);
}
int receive(unsigned char *data, enum typemode *mode)
{
    int flag;
    hide_cur();
    *data = inport(I_PORT379);
```

```
    flag = decode_5( *data ,mode);
    show_cur();
    return (flag);
}
void connect(struct device *instru)
{
    windesc *dialog, *old_w;
    enum typemode mode;
    int flag;     /*** flag of transmit or receive ***/
    unsigned char data5, data8, data4 /**********/ ;
    int active;

    old_w = curr_win;
    dialog = draw_win(CTRWIN,CTRWIN,35,5,"",popup, &slctcolor);
    make_shadow(dialog);
    gotoxy(2,2); cprintf(" Please waiting ... ");
    data8 = transmit( instru );
    data4 = d_encode( instru->number );
    delay(500);
    set_m_receiv( &data8 );
    delay(DELAY_TIME);
    flag = receive( &data5, &mode );
    active = (flag==RECEIV_OK) ? 1 : 0;
    hideselect(dialog);
    dialog = draw_win(CTRWIN,CTRWIN,40,15,"",popup, &slctcolor);
    make_shadow(dialog);
    mprintf("\r\n (Port_%x) Data_out_is   ",0_P_PORT8);bin_disp(data8);
    mprintf("\r\n (Port_%x) Data_out_is   ",0_P_PORT4);bin_disp(data4);
    mprintf("\r\n");
    mprintf("\r\n   Device number  %d" , instru->number);
    mprintf("\r\n   Level of device is  %d", instru->level);
    mprintf("\r\n   In mode ->  %s", tmode[mode]);
    mprintf("\r\n");
    mprintf("\r\n (Port_%x) Data_in_is   ",I_PORT379);bin_disp(data5);
    mprintf("\r\n");
    mprintf("\r\n   Mode status is  %s", tmode[mode]);
    mprintf("\r\n   Device is %s active", (active) ? "" : "NOT" );
    done2();
    hideselect(dialog);
    slct_win(old_w);
}
void yee(void)
{
```

```
    windesc *dialog;
    dialog = draw_win(CTRWIN,CTRWIN, 55, 12, "",popup, &slctcolor);
    make_shadow(dialog);
    hide_cur();
    cprintf("\r\n");
    cprintf("   PROJECT: DATA - TRANSMISSION THROUGH AC LINE\r\n");
    cprintf(" \r\n");
    cprintf("     ADVISOR: DR. RUTTIKORN VARAKULSIRIPUNTH \r\n\r\n");
    cprintf("                PRESENT BY \r\n");
    cprintf("     KAMOLWUN  DANKHONSAKUL    33100004 \r\n");
    cprintf("     KRIT  PIENPRAJYAPORN      33100011 \r\n");
    cprintf("     NIPAPON  SIRIPON          33100179 \r\n");
    done2();
    show_cur();
    hideselect(dialog);
}
```

หนังสืออ้างอิง

1. Morris slurzburg, William Osterheld, "Essentials of Communication Electronics", Mc Graw-Hill, Inc., Singapore,1973
2. Pual B. Zbar,Albert P. Malvino, Michael A. Miller, " Basic Electronics", Mc Graw-Hill, USA, 1983
3. Sidney Soclof, "Design and applications of analog integrated circuits", Pretice-Hall, USA, 19911
4. พลผดุง ผดุงกุล, "ไตรแอคและเอสซีอาร์", เซมิคอนดัคเตอร์อิเลคทรอนิคส์ 91,92 บริษัทซีเอ็ดยูเคชั่น
5. ชาญยุทธ พฤกษ์ค้มวงศ์,"เรียนรู้การสื่อสารข้อมูล",เซมิคอนดัคเตอร์อิเลคทรอนิคส์ 128 130 ,บริษัทซีเอ็ดยูเคชั่น

National Semiconductor Corporation

LM566C Voltage Controlled Oscillator

General Description

The LM566CN is a general purpose voltage controlled oscillator which may be used to generate square and triangular waves, the frequency of which is a very linear function of a control voltage. The frequency is also a function of an external resistor and capacitor.

The LM566CN is specified for operation over the 0°C to +70°C temperature range.

Features

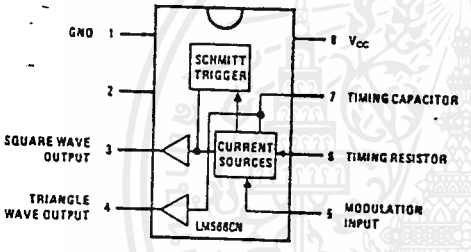
- Wide supply voltage range: 10V to 24V
- Very linear modulation characteristics
- High temperature stability
- Excellent supply voltage rejection
- 10 to 1 frequency range with fixed capacitor
- Frequency programmable by means of current, voltage, resistor or capacitor

Applications

- FM modulation
- Signal generation
- Function generation
- Frequency shift keying
- Tone generation

Connection Diagram

Dual-In-Line Package

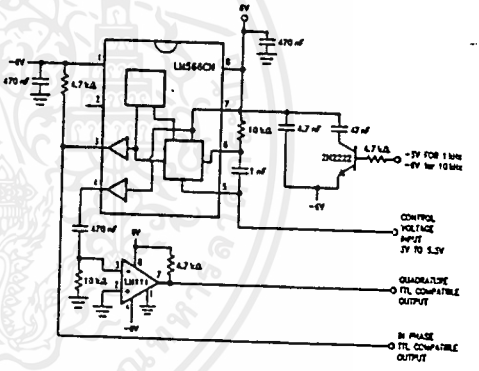


TL/H/7854–2

Order Number LM566CN
See NS Package Number N08E

Typical Application

1 kHz and 10 kHz TTL Compatible Voltage Controlled Oscillator



TL/H/7854–3

Absolute Maximum Ratings

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

Power Supply Voltage	26V
Power Dissipation (Note 1)	1000 mW
Operating Temperature Range, LM566CN	0°C to +70°C
Lead Temperature (Soldering, 10 sec.)	+260°C

Electrical Characteristics V_{CC} = 12V, T_A = 25°C, AC Test Circuit

Parameter	Conditions	LM566C			Units
		Min	Typ	Max	
Maximum Operating Frequency	R0 = 2k C0 = 2.7 pF	0.5	1		MHz
VCO Free-Running Frequency	C_O = 1.5 nF R_O = 20k f_O = 10 kHz	−30	0	+30	%
Input Voltage Range Pin 5		¾ V_{CC}		V_{CC}	
Average Temperature Coefficient of Operating Frequency			200		ppm/°C
Supply Voltage Rejection	10–20V		0.1	2	%/V
Input Impedance Pin 5		0.5	1		MΩ
VCO Sensitivity	For Pin 5, From 8–10V, f_O = 10 kHz	6.0	6.6	7.2	kHz/V
FM Distortion	±10% Deviation		0.2	1.5	%
Maximum Sweep Rate			1		MHz
Sweep Range			10:1		
Output Impedance Pin 3			50		Ω
Pin 4			50		Ω
Square Wave Output Level	R_{L1} = 10k	5.0	5.4		Vp-p
Triangle Wave Output Level	R_{L2} = 10k	2.0	2.4		Vp-p
Square Wave Duty Cycle		40	50	60	%
Square Wave Rise Time			20		ns
Square Wave Fall Time			50		ns
Triangle Wave Linearity	+1V Segment at ½ V_{CC}		0.5		%

Note 1: The maximum junction temperature of the LM566CN is 150°C. For operation at elevated junction temperatures, maximum power dissipation must be derated based on a thermal resistance of 115°C/W, junction to ambient.

Applications Information

The LM566CN may be operated from either a single supply as shown in this test circuit, or from a split (±) power supply. When operating from a split supply, the square wave output (pin 3) is TTL compatible (2 mA current sink) with the addition of a 4.7 kΩ resistor from pin 3 to ground.

A 0.001 μF capacitor is connected between pins 5 and 6 to prevent parasitic oscillations that may occur during VCO switching.

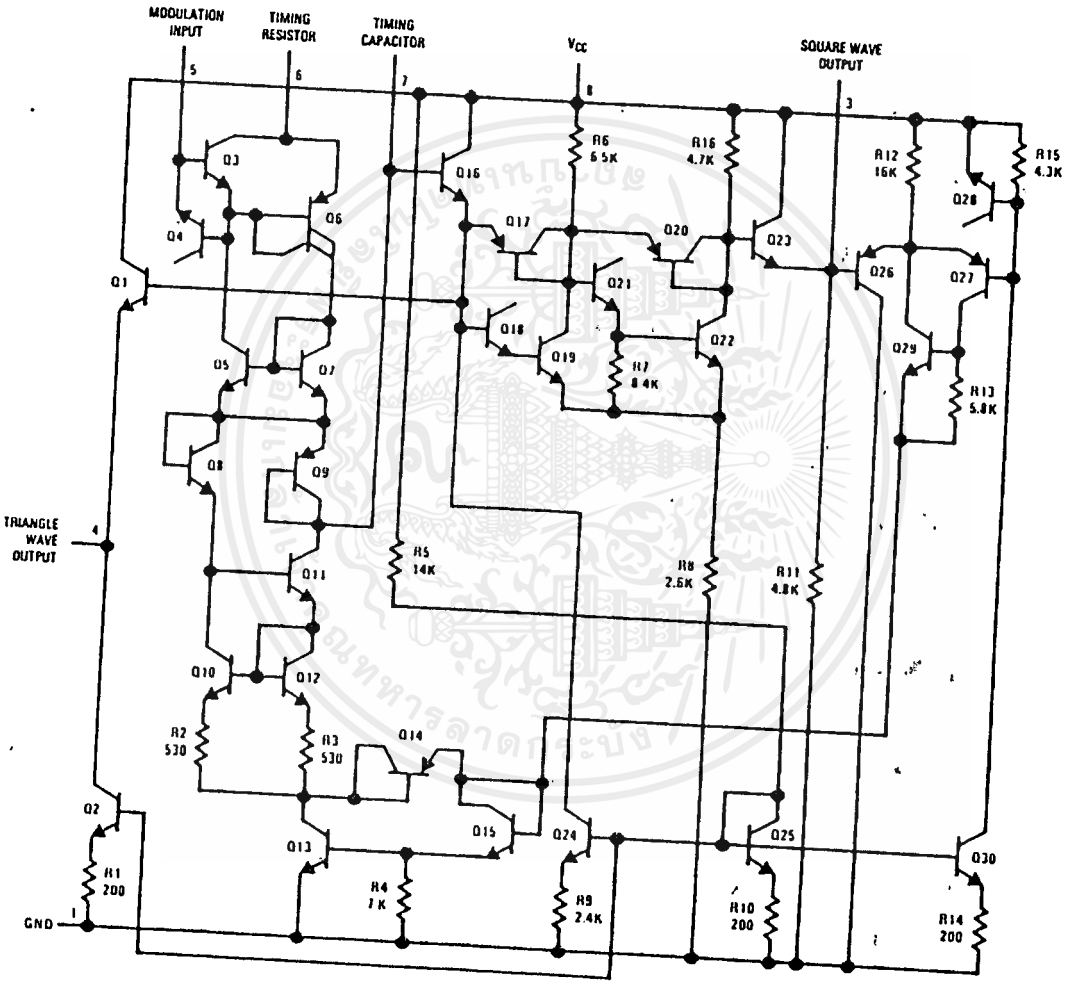
$$f_O = \frac{2.4(V^+ - V_5)}{R_O C_O V^+}$$

where

$2K < R_O < 20K$

and V_5 is voltage between pin 5 and pin 1.

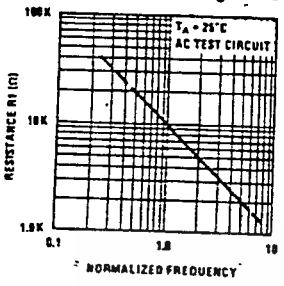
Schematic Diagram



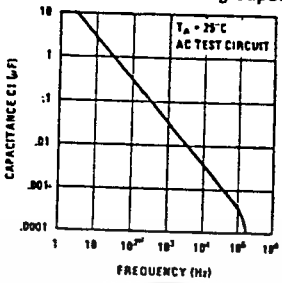
TL/H/7854–1

Typical Performance Characteristics

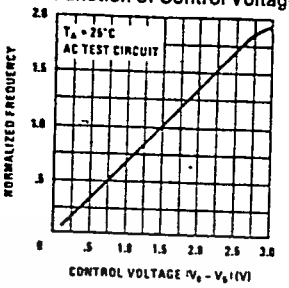
Operating Frequency as a Function of Timing Resistor



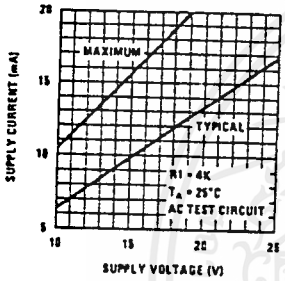
Operating Frequency as a Function of Timing Capacitor



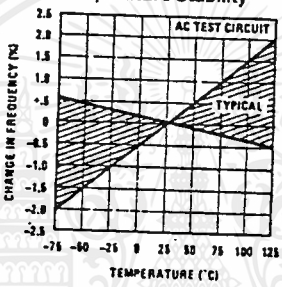
Normalized Frequency as a Function of Control Voltage



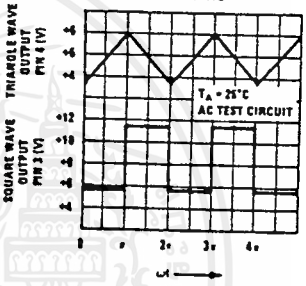
Power Supply Current



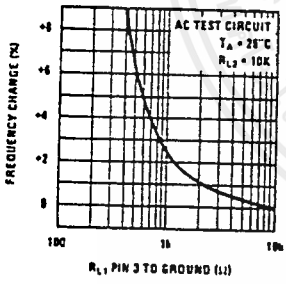
Temperature Stability



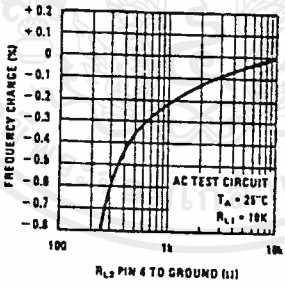
VCO Waveforms



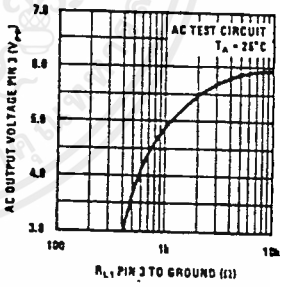
Frequency Stability vs Load Resistance (Square Wave Output)



Frequency Stability vs Load Impedance (Triangle Output)

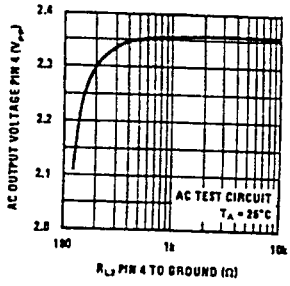


Square Wave Output Characteristics



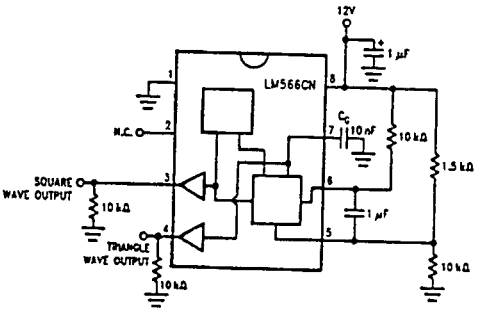
TL/H/7854–4

Triangle Wave Output Characteristics



TL/H/7854–5

AC Test Circuit



TL/H/7854–6

National Semiconductor Corporation

LM565/LM565C Phase Locked Loop

General Description

The LM565 and LM565C are general purpose phase locked loops containing a stable, highly linear voltage controlled oscillator for low distortion FM demodulation, and a double balanced phase detector with good carrier suppression. The VCO frequency is set with an external resistor and capacitor, and a tuning range of 10:1 can be obtained with the same capacitor. The characteristics of the closed loop system—bandwidth, response speed, capture and pull in range—may be adjusted over a wide range with an external resistor and capacitor. The loop may be broken between the VCO and the phase detector for insertion of a digital frequency divider to obtain frequency multiplication.

The LM565H is specified for operation over the −55°C to +125°C military temperature range. The LM565CH and LM565CN are specified for operation over the 0°C to +70°C temperature range.

Features

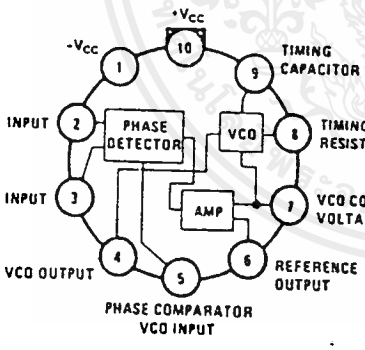
- 200 ppm/°C frequency stability of the VCO
- Power supply range of ±5 to ±12 volts with 100 ppm/% typical
- 0.2% linearity of demodulated output
- Linear triangle wave with in phase zero crossings available
- TTL and DTL compatible phase detector input and square wave output.
- Adjustable hold in range from ±1% to > ±60%

Applications

- Data and tape synchronization
- Modems
- FSK demodulation
- FM demodulation
- Frequency synthesizer
- Tone decoding
- Frequency multiplication and division
- SCA demodulators
- Telemetry receivers
- Signal regeneration
- Coherent demodulators

Connection Diagrams

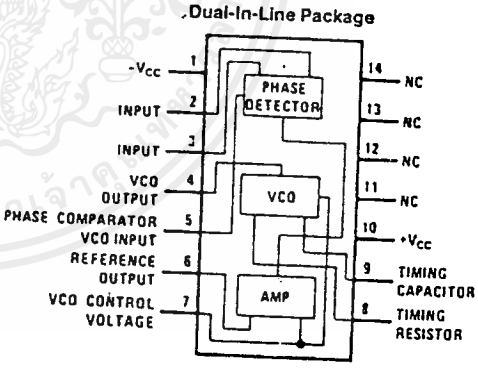
Metal Can Package



TL/H/7853-2

Order Number LM565H or LM565CH
See NS Package Number H10C

Dual-In-Line Package



TL/H/7853-3

Order Number LM565CN
See NS Package Number N14A

Absolute Maximum Ratings

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

Supply Voltage	±12V
Power Dissipation (Note 1)	1400 mW
Differential Input Voltage	±1V
Operating Temperature Range	
LM565H	−55°C to +125°C
LM565CH, LM565CN	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	260°C

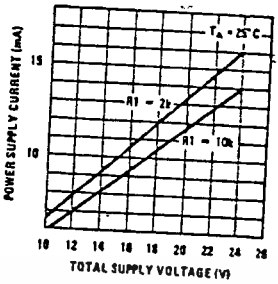
Electrical Characteristics AC Test Circuit, $T_A = 25°C$, $V_{CC} = \pm 6V$

Parameter	Conditions	LM565			LM565C			Units
		Min	Typ	Max	Min	Typ	Max	
Power Supply Current			8.0	12.5		8.0	12.5	mA
Input Impedance (Pins 2, 3)	$-4V < V_2, V_3 < 0V$	7	10			5		kΩ
VCO Maximum Operating Frequency	$C_o = 2.7$ pF	300	500		250	500		kHz
VCO Free-Running Frequency	$C_o = 1.5$ nF $R_o = 20$ kΩ $f_o = 10$ kHz	−10	0	+10	−30	0	+30	%
Operating Frequency Temperature Coefficient			−100			−200		ppm/°C
Frequency Drift with Supply Voltage			0.1	1.0		0.2	1.5	%/V
Triangle Wave Output Voltage		2	2.4	3	2	2.4	3	V_{p-p}
Triangle Wave Output Linearity			0.2			0.5		%
Square Wave Output Level		4.7	5.4		4.7	5.4		V_{p-p}
Output Impedance (Pin 4)			5			5		kΩ
Square Wave Duty Cycle		45	50	55	40	50	60	%
Square Wave Rise Time			20			20		ns
Square Wave Fall Time			50			50		ns
Output Current Sink (Pin 4)		0.6	1		0.6	1		mA
VCO Sensitivity	$f_o = 10$ kHz		6600			6600		Hz/V
Demodulated Output Voltage (Pin 7)	±10% Frequency Deviation	250	300	400	200	300	450	mV_{p-p}
Total Harmonic Distortion	±10% Frequency Deviation		0.2	0.75		0.2	1.5	%
Output Impedance (Pin 7)			3.5			3.5		kΩ
DC Level (Pin 7)		4.25	4.5	4.75	4.0	4.5	5.0	V
Output Offset Voltage $\lvert V_7 - V_6 \rvert$			30	100		50	200	mV
Temperature Drift of $\lvert V_7 - V_6 \rvert$			500			500		μV/°C
AM Rejection		30	40			40		dB
Phase Detector Sensitivity K_D			.68			.68		V/radian

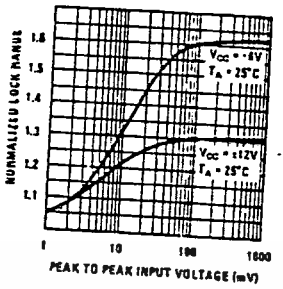
Note 1: The maximum junction temperature of the LM565 and LM565C is +150°C. For operation at elevated temperatures, devices in the TO-5 package must be derated based on a thermal resistance of +150°C/W junction to ambient or +45°C/W junction to case. Thermal resistance of the dual-in-line package is +85°C/W.

Typical Performance Characteristics

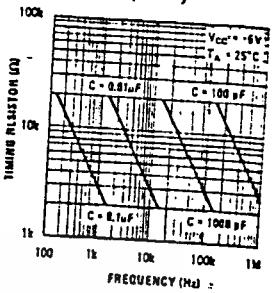
Power Supply Current as a Function of Supply Voltage



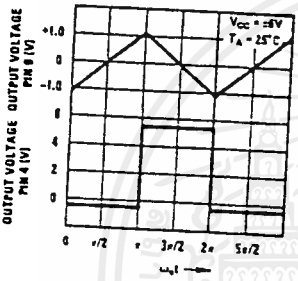
Lock Range as a Function of Input Voltage



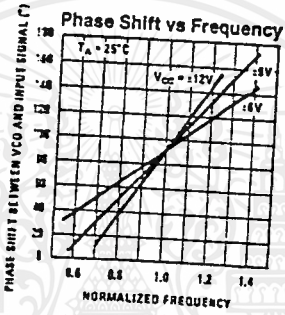
VCO Frequency



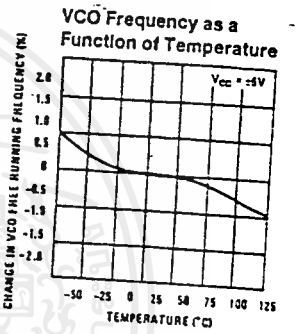
Oscillator Output Waveforms



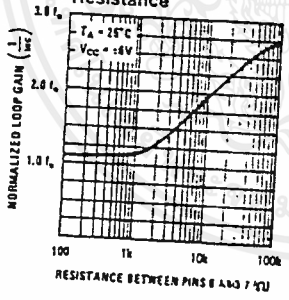
Phase Shift vs Frequency



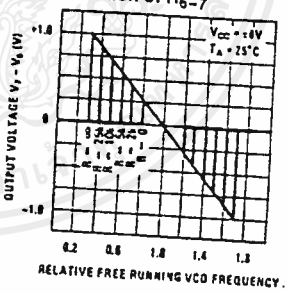
VCO Frequency as a Function of Temperature



Loop Gain vs Load Resistance

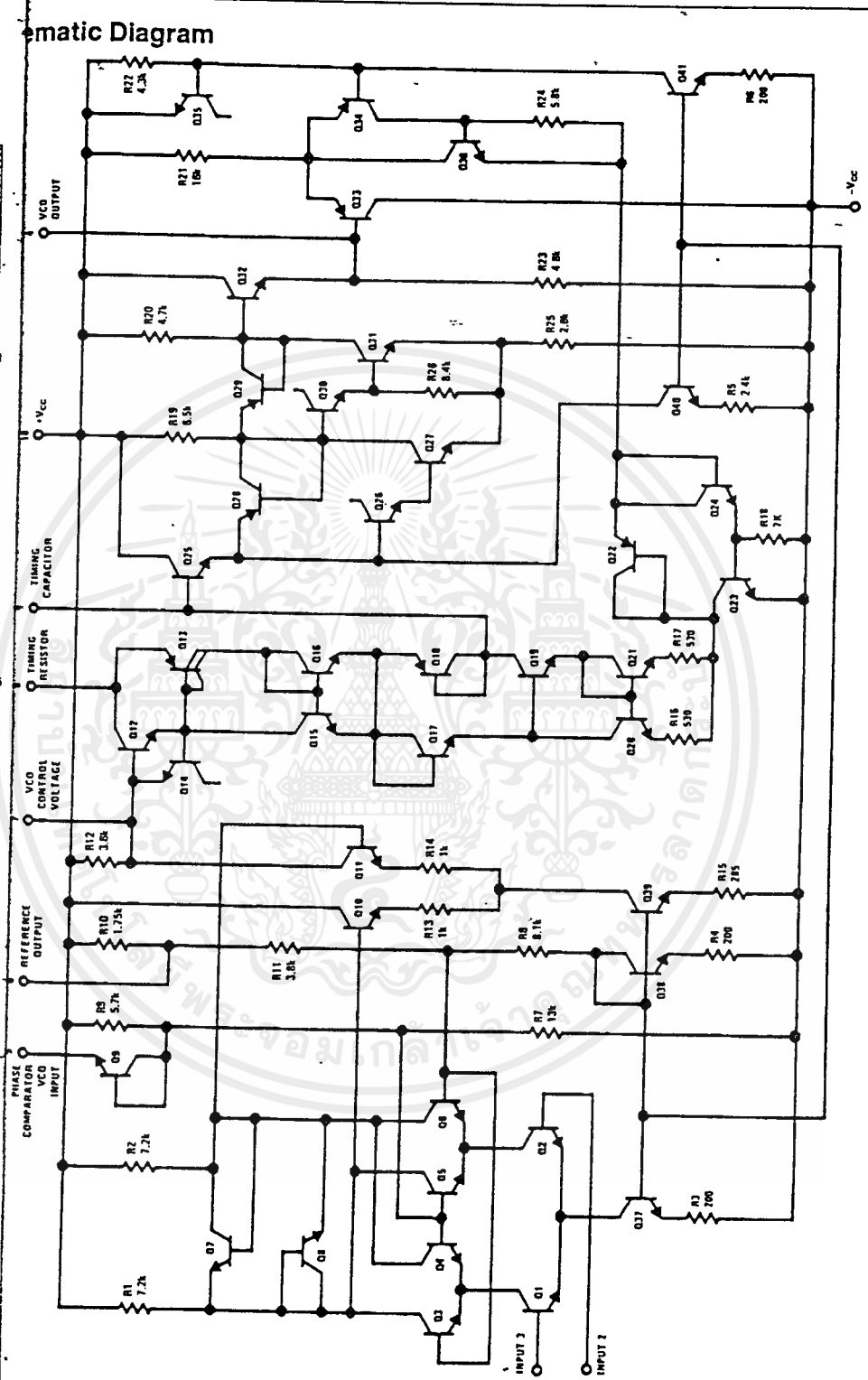


Hold In Range as a Function of R_{6-7}

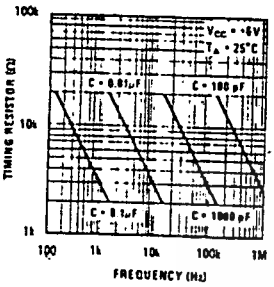


TL/H/7953-4

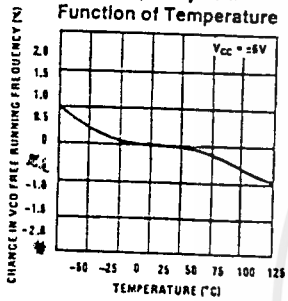
ematic Diagram



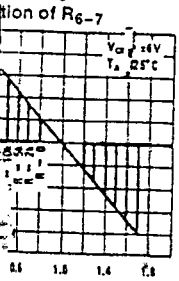
VCO Frequency



VCO Frequency as a Function of Temperature



In Range as a tion of R_{6-7}



—REE RUNNING VCO FREQUENCY.

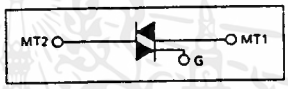
MAC210FP Series
MAC210AFP Series

Triacs
Silicon Bidirectional Thyristors

ISOLATED TRIACs
THYRISTORS
10 AMPERES RMS
200 thru 800 VOLTS

. . . designed primarily for full-wave ac control applications, such as light dimmers, motor controls, heating controls and power supplies; or wherever full-wave silicon gate controlled solid-state devices are needed. Triac type thyristors switch from a blocking to a conducting state for either polarity of applied anode voltage with positive or negative gate triggering.

- Blocking Voltage to 800 Volts
- All Diffused and Glass Passivated Junctions for Greater Parameter Uniformity and Stability
- Small, Rugged, Thermowatt Construction for Low Thermal Resistance, High Heat Dissipation and Durability
- Gate Triggering Guaranteed in Three Modes (MAC210FP Series) or Four Modes (MAC210AFP Series)



CASE 221C-02
STYLE 3

3

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Repetitive Peak Off-State Voltage, Note 1 (T_J = −40 to +125°C) 1/2 Sine Wave 50 to 60 Hz, Gate Open MAC210-4FP, MAC210A4FP MAC210-6FP, MAC210A6FP MAC210-8FP, MAC210A8FP MAC210-10FP, MAC210A10FP	V_{DRM}	 200 400 600 800	Volts
On-State RMS Current (T_C = +70°C) Full Cycle Sine Wave 50 to 60 Hz, Note 2	$I_{T(RMS)}$	10	Amps
Peak Nonrepetitive Surge Current (One Full Cycle, 60 Hz, T_C = +70°C) preceded and followed by rated current	I_{TSM}	100	Amps
Circuit Fusing (t = 8.3 ms)	I^2t	40	A^2s
Peak Gate Power (T_C = +70°C, Pulse Width = 10 μs)	P_{GM}	20	Watts
Average Gate Power (T_C = +70°C, t = 8.3 ms)	$P_{G(AV)}$	0.35	Watt
Peak Gate Current (T_C = +70°C, Pulse Width = 10 μs)	I_{GM}	2	Amps
RMS Isolation Voltage (T_A = 25°C, Relative Humidity ≤ 20%)	$V_{(ISO)}$	1500	Volts
Operating Junction Temperature	T_J	−40 to +125	°C
Storage Temperature Range	T_{stg}	−40 to +125	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	$R_{\theta JC}$	2.2	°C/W
Thermal Resistance, Case to Sink	$R_{\theta CS}$	2.2 (typ)	°C/W
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	60	°C/W

Notes: 1. Ratings apply for open gate conditions. Thyristor devices shall not be tested with a constant current source for blocking capability such that the voltage applied exceeds the rated blocking voltage.
2. The case temperature reference point for all T_C measurements is a point on the center lead of the package as close as possible to the plastic body.

ELECTRICAL CHARACTERISTICS (T_C = +25°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Peak Blocking Current (Either Direction) Rated V_{DRM}, Gate Open T_J = 25°C T_J = +125°C	I_{DRM}	 — —	 — —	 10 2	 μA mA
Peak On-State Voltage (Either Direction) I_{TM} = 14 A Peak; Pulse Width = 1 to 2 ms, Duty Cycle ≤ 2%	V_{TM}	—	1.2	1.65	Volts
Gate Trigger Current (Continuous dc) Main Terminal Voltage = 12 Vdc, R_L = 100 Ohms Minimum Gate Pulse Width = 2 μs MT2(+), G(+) MT2(+), G(−) MT2(−), G(−) MT2(−), G(+) "A" SUFFIX ONLY	I_{GT}	 — — — —	 12 12 20 35	 50 50 50 75	mA
Gate Trigger Voltage (Continuous dc) Main Terminal Voltage = 12 Vdc, R_L = 100 Ohms Minimum Gate Pulse Width = 2 μs MT2(+), G(+) MT2(+), G(−) MT2(−), G(−) MT2(−), G(+) "A" SUFFIX ONLY Main Terminal Voltage = Rated V_{DRM}, R_L = 10 kΩ, T_J = +125°C MT2(+), G(+); MT2(+), G(-); MT2(-), G(-) MT2(-), G(+) "A" SUFFIX ONLY	V_{GT}	 — — — — 0.2 0.2	 0.9 0.9 1.1 1.4 — —	 2 2 2 2.5 — —	Volts
Holding Current (Either Direction) Main Terminal Voltage = 12 Vdc, Gate Open, Initiating Current = 500 mA, T_C = +25°C	I_H	—	6	50	mA
Turn-On Time Rated V_{DRM}, I_{TM} = 14 A, I_{GT} = 120 mA, Rise Time = 0.1 μs, Pulse Width = 2 μs	t_{gt}	—	1.5	—	μs
Critical Rate of Rise of Commutation Voltage Rated V_{DRM}, I_{TM} = 14 A, Commutating di/dt = 4.3 A/ms, Gate Unenergized, T_C = +70°C	$dv/dt_{(c)}$	—	5	—	V/μs
Critical Rate of Rise of Off-State Voltage (V_D = V_{DROM}, Exponential Voltage Rise, Gate Open, T_C = +70°C)	dv/dt	—	100	—	V/μs

3

TYPICAL CHARACTERISTICS

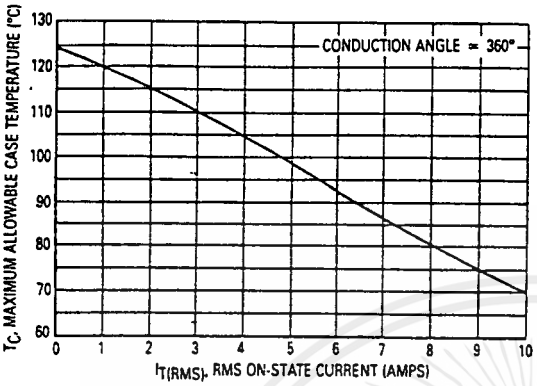


Figure 1. Current Derating

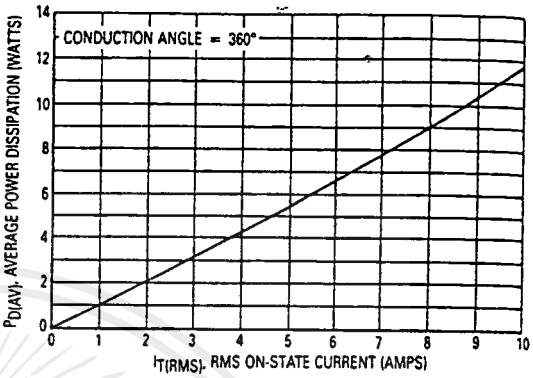


Figure 2. Power Dissipation

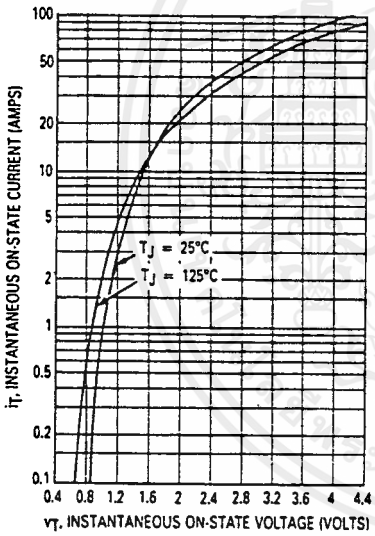


Figure 3. Maximum On-State Characteristics

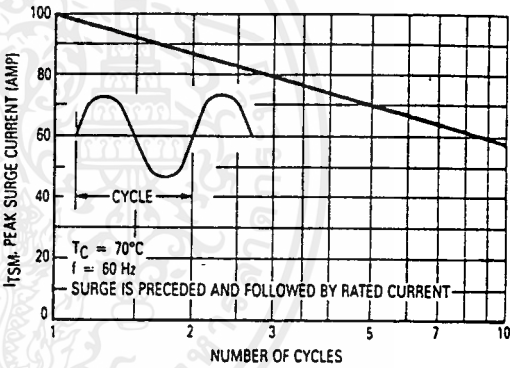


Figure 4. Maximum Nonrepetitive Surge Current

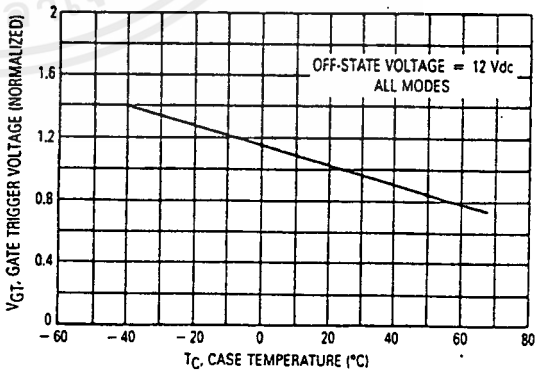


Figure 5. Typical Gate Trigger Voltage

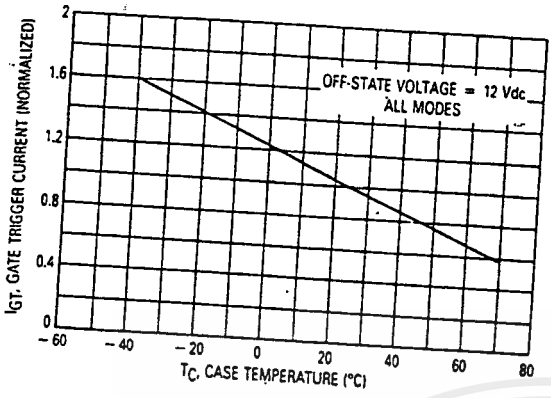


Figure 6. Typical Gate Trigger Current

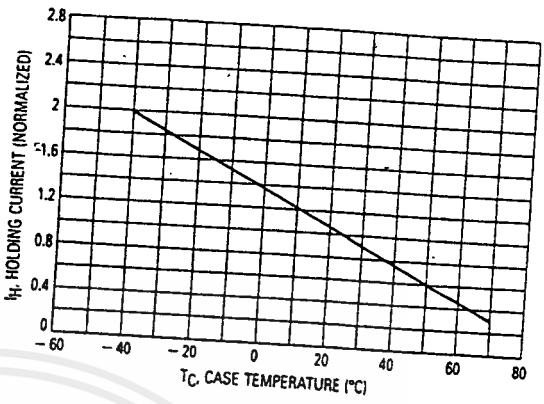


Figure 7. Typical Holding Current

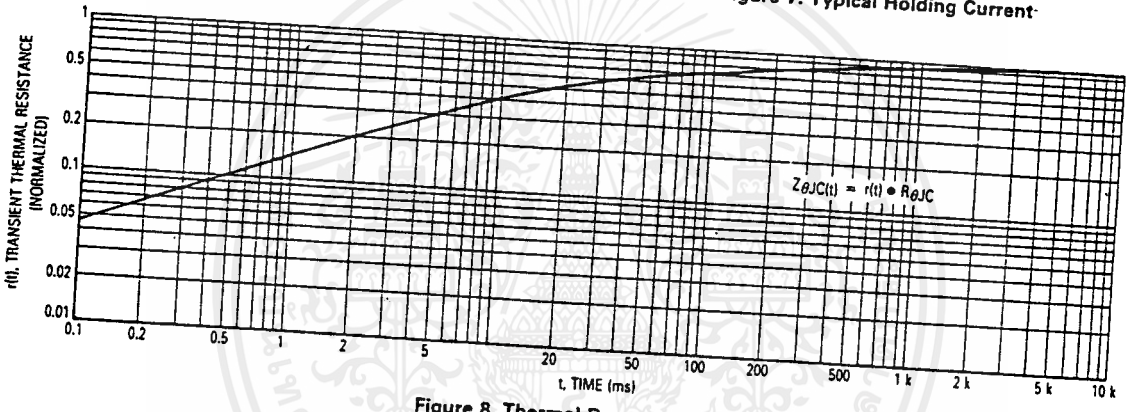


Figure 8. Thermal Response

IC รีโมท ตัวนี้ดีมาก

UMC

UM3750

Encoder/Decoder

Features

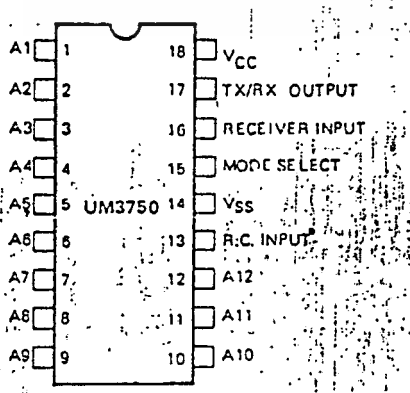
- Single chip contains both encoder and decoder.
- 3V to 11V operation.
- RC Oscillator
- Cross interference of receivers is virtually eliminated by circuitry which requires 4 valid words to be received each within 64ms of the other.
- Applications: alarm control system, security system, cordless telephone, remote control.
- Interfaces with RF, ultrasonic, or infrared modulators and demodulators.

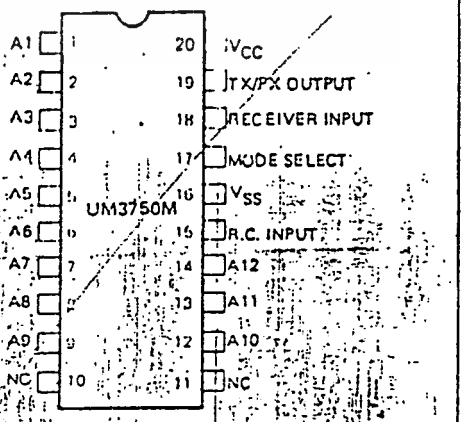
General Description

The UM3750 Encoder/Decoder is an MOS/LSI digital code Transmitter-Receiver system. Working in the transmission mode, the UM3750 will sequentially encode and transmit 12 bits of inputs. Each of the 12 inputs may be 1 or 0 to allow 4096 different codes.

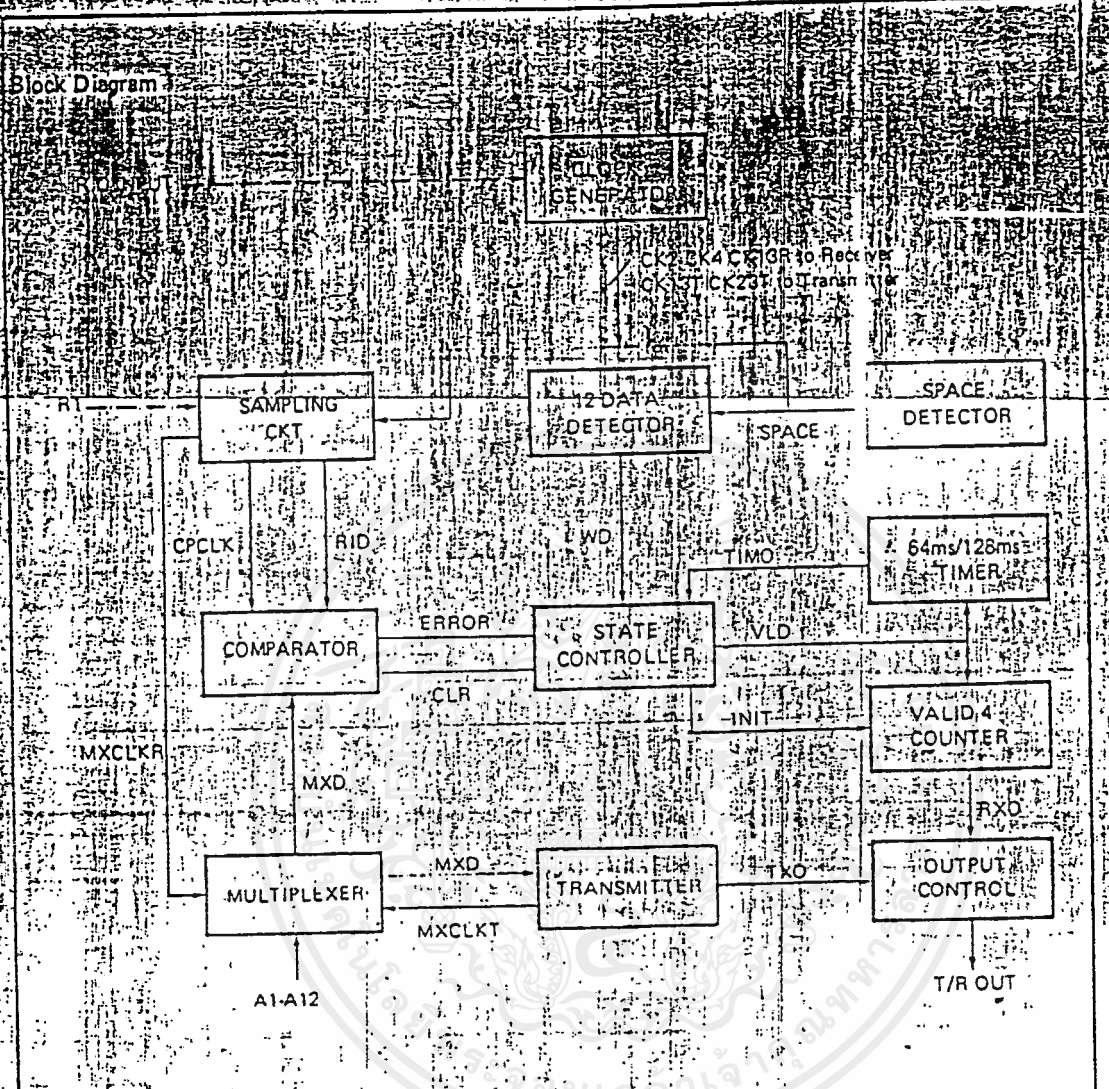
In the receiver mode, the incoming signal is compared to the local code in a sequential manner. Once an error is detected the system will reset and begin its comparison on the next word. If all 12 bits are received correctly, a "valid" signal is generated. This signal clears a 64ms counter and triggers a 3-stage counter. The 3-stage counter counts three "valid" pulses and when 4 pulses have been detected, the Transmit/Receive output goes low. After the Transmit/Receive output goes low, the next "valid" must be received within 128ms, giving a one valid in 6 requirement to keep the Transmit/Receive low.

Pin Configuration





Block Diagram



Block Diagram Description

CPCLK: CLK of Comparator
MXCLKR: CLK of Multiplexer when in Receiver mode
MXCLKT: CLK of Multiplexer when in Transmitter mode
MXD: Output data of Multiplexer (one of A1, A2 ..... A12)
RID: Sampled data by Sampling CKT
VLD: "Valid" signal. It is used to trigger Valid 4 Counter and reset 64ms/128ms Timer

CLR: Clear signal of Comparator
ERROR: Error signal from Comparator
TIMO: TIMER time-out signal (64ms or 128ms)
T/R OUT: Transmit/Receiver output pin
INIT: Reset signal of Valid 4 Counter
WD: Word detected signal
TXO: Transmitter output
PXO: Receiver output

Absolute Maximum Ratings*

Power Supply Voltage ..... −0.3V to 11V

Operating Temperature ..... −20 Deg to 70 Deg C

Storage Temperature (Tstg) ..... −55 Deg to 150 Deg C

Apply Voltage on any Pin

$$V_{SS} - 0.3 < V_{IN} < V_{DD} + 0.3$$

*Comments

Stress above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those indicated in the operational sections of this specification is not implied and exposure to absolute maximum rating conditions for extended periods may affect device reliability.

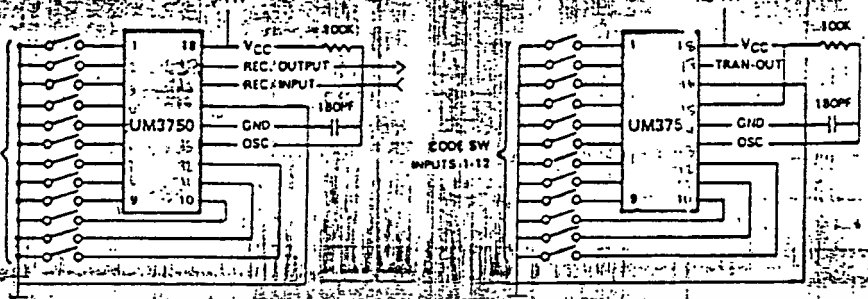
Electrical Characteristics (T_A = 25°C V_{DD} = 9V unless otherwise secified)

Parameter	Symbol	Min.	Max.	Unit	Condition
Operating Voltage	V_{DD}	3.0	11	V	
Schmitt Trigger Input Level		V_{SS}+4 V_{SS}+7		V V	Level 1 Level 0
Other Pins Input Level		V_{DD}−0.5 V_{SS}	V_{DD} V_{SS}+0.5	V	Level 1 Level 0
Output Pin Logic Level	V_{OH} V_{OL}	V_{DD}−0.5 V_{SS}	V_{DD} V_{SS}+1	V V	Isource = 5μA Isink = 2mA
Input Resistor to V_{CC}		200K	1.2M	Ohm	

Pin Designation

Pin No.	Designation	Description
1 – 12	A1 – A12	These data select lines are used to set the address of the encoder/decoder pair. They have on-chip pull-up resistors.
13	R.C. INPUT	R.C. input pin for single pin oscillator. A resistor is hooked from this pin to V_{CC} and a capacitor from this pin to GND. The frequency = 2/RC.
14	V_{SS}	The ground pin of the UM3750.
15	MODE SELECT	This pin changes the IC from Receiver mode to Transmitter mode. By grounding this pin the IC is put in to the Receiver mode. By connection to V_{CC} the IC is put in to the Transmitter mode.
16	RECEIVER INPUT	The receiver input receive the digital PCM waveform from the detect circuit.
17	TX/RX OUTPUT	The receiver input receives the digital PCM waveform from the detect low in the Receive mode.
18	V_{CC}	The positive supply pin.

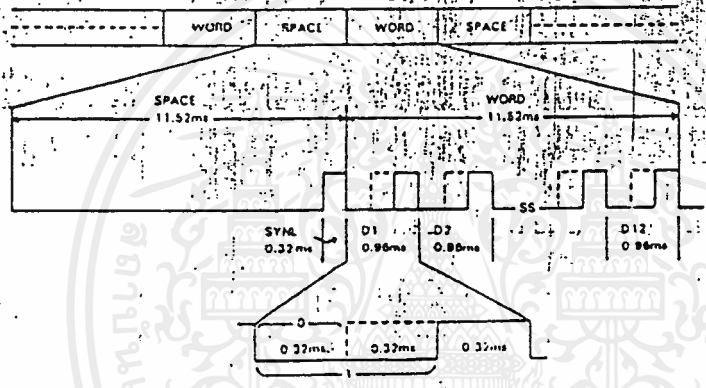
Connection Diagram for Transmitter/Receiver



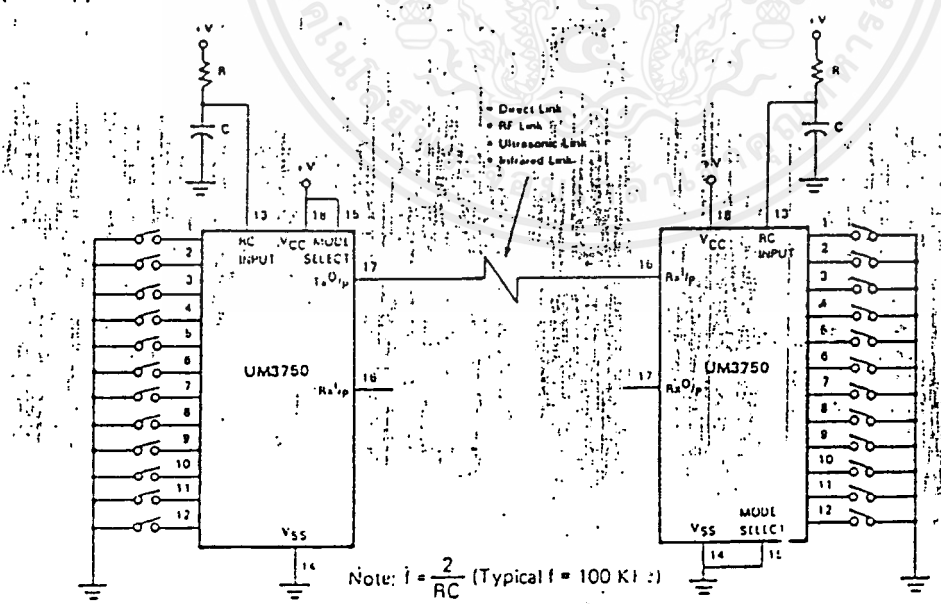
Pin Connections for Receiver Mode

Pin Connections for Transmitter Mode

Output Waveform (base on 100 KHz)

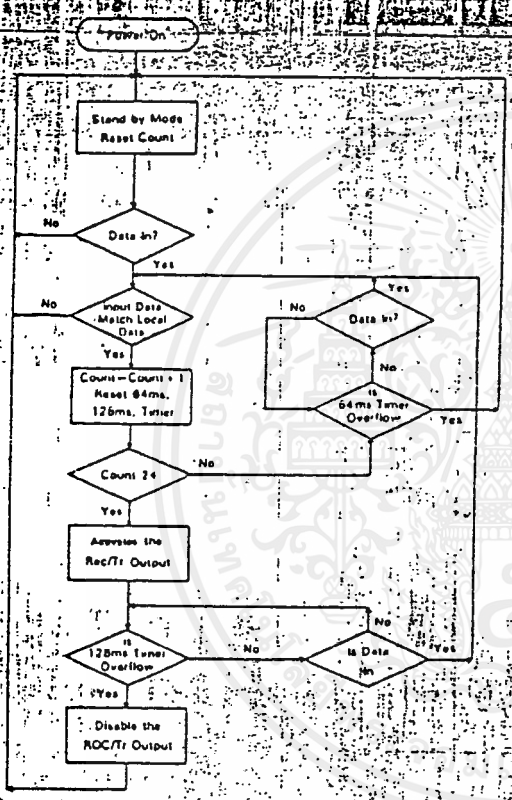


Typical Application CKT



Note: $f = \frac{2}{RC}$ (Typical f = 100 KHz)

Decoder Flowchart



Ordering Information

Part No.	Package
UM3750	18 Pin plastic Dual-In-Line
UM3750M	20 Pin Small Outline

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V_{DD} Vdc	T_{low}* Min	T_{low}* Max	25°C Min	25°C Typ	25°C Max	T_{high}* Min	T_{high}* Max	Unit
Output Voltage "0" Level V_{in} = V_{DD} or 0	V_{OL}	5.0	—	0.05	—	0	0.05	—	0.05	Vdc
		10	—	0.05	—	0	0.05	—	0.05	
		15	—	0.05	—	0	0.05	—	0.05	
"1" Level V_{in} = 0 or V_{DD}	V_{OH}	5.0	4.95	—	4.95	5.0	—	4.95	—	Vdc
		10	9.95	—	9.95	10	—	9.95	—	
		15	14.95	—	14.95	15	—	14.95	—	
Input Voltage# "0" Level	V_{IL}									Vdc
(V_O = 4.5 or 0.5 Vdc)		5.0	—	1.5	—	2.25	1.5	—	1.5	
(V_O = 9.0 or 1.0 Vdc)		10	—	3.0	—	4.50	3.0	—	3.0	
(V_O = 13.5 or 1.5 Vdc)		15	—	4.0	—	6.75	4.0	—	4.0	
"1" Level	V_{IH}									
(V_O = 0.5 or 4.5 Vdc)		5.0	3.5	—	3.5	2.75	—	3.5	—	Vdc
(V_O = 1.0 or 9.0 Vdc)		10	7.0	—	7.0	5.50	—	7.0	—	
(V_O = 1.5 or 13.5 Vdc)		15	11.0	—	11.0	8.25	—	11.0	—	
Output Drive Current (AL Device)	I_{OH}									mAdc
(V_{OH} = 2.5 Vdc) Source		5.0	−1.2	—	−1.0	−1.7	—	−0.7	—	
(V_{OH} = 4.6 Vdc)		5.0	−0.25	—	−0.2	−0.36	—	−0.14	—	
(V_{OH} = 9.5 Vdc)		10	−0.62	—	−0.5	−0.9	—	−0.35	—	
(V_{OH} = 13.5 Vdc)		15	−1.8	—	−1.5	−3.5	—	−1.1	—	
(V_{OL} = 0.4 Vdc) Sink	I_{OL}	5.0	0.64	—	0.51	0.88	—	0.36	—	mAdc
(V_{OL} = 0.5 Vdc)		10	1.6	—	1.3	2.25	—	0.9	—	
(V_{OL} = 1.5 Vdc)		15	4.2	—	3.4	8.8	—	2.4	—	
Output Drive Current (CL/CP Device)	I_{OH}									mAdc
(V_{OH} = 2.5 Vdc) Source		5.0	−1.0	—	−0.8	−1.7	—	−0.6	—	
(V_{OH} = 4.6 Vdc)		5.0	−0.2	—	−0.16	−0.36	—	−0.12	—	
(V_{OH} = 9.5 Vdc)		10	−0.5	—	−0.4	−0.9	—	−0.3	—	
(V_{OH} = 13.5 Vdc)		15	−1.4	—	−1.2	−3.5	—	−1.0	—	
(V_{OL} = 0.4 Vdc) Sink	I_{OL}	5.0	0.52	—	0.44	0.88	—	0.36	—	mAdc
(V_{OL} = 0.5 Vdc)		10	1.3	—	1.1	2.25	—	0.9	—	
(V_{OL} = 1.5 Vdc)		15	3.6	—	3.0	8.8	—	2.4	—	
Input Current (AL Device)	I_{in}	15	—	±0.1	—	±0.00001	±0.1	—	±1.0	μAdc
Input Current (CL/CP Device)	I_{in}	15	—	±0.3	—	±0.00001	±0.3	—	±1.0	μAdc
Input Capacitance (V_{in} = 0)	C_{in}	—	—	—	—	5.0	7.5	—	—	pF
Quiescent Current (AL Device) (Per Package)	I_{DD}	5.0	—	5.0	—	0.005	5.0	—	150	μAdc
		10	—	10	—	0.010	10	—	300	
		15	—	20	—	0.015	20	—	600	
Quiescent Current (CL/CP Device) (Per Package)	I_{DD}	5.0	—	20	—	0.005	20	—	150	μAdc
		10	—	40	—	0.010	40	—	300	
		15	—	80	—	0.015	80	—	600	
Total Supply Current**† (Dynamic plus Quiescent, Per Package) (C_L = 50 pF on all outputs, all buffers switching)	I_T	5.0	I_T = (0.27 μA/kHz) f + I_{DD}							μAdc
		10	I_T = (0.55 μA/kHz) f + I_{DD}							
		15	I_T = (0.83 μA/kHz) f + I_{DD}							

*T_{low} = −55°C for AL Device, −40°C for CL/CP Device.
T_{high} = +125°C for AL Device, +85°C for CL/CP Device.
#Noise immunity specified for worst-case input combination.
Noise Margin for both "1" and "0" level = 1.0 Vdc min @ V_{DD} = 5.0 Vdc
2.0 Vdc min @ V_{DD} = 10 Vdc
2.5 Vdc min @ V_{DD} = 15 Vdc
†To calculate total supply current at loads other than 50 pF:
$I_T(C_L) = I_T(50\ pF) + 1.1 \times 10^{-3} (C_L - 50) V_{DD} f$
where: I_T is in μA (per package), C_L in pF, V_{DD} in Vdc, and f in kHz is input frequency.
**The formulas given are for the typical characteristics only at 25°C.

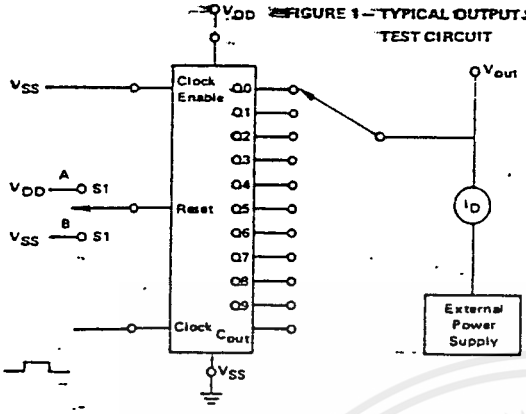
This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$.
Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

SWITCHING CHARACTERISTICS* (C_L = 50 pF, T_A = 25°C)

Characteristic	Symbol	V_{DD} Vdc	Min	Typ	Max	Unit
Output Rise Time t_{TLH} = (3.0 ns/pF) C_L + 30 ns t_{TLH} = (1.5 ns/pF) C_L + 15 ns t_{TLH} = (1.1 ns/pF) C_L + 10 ns	t_{TLH}	5.0 10 15	— — —	180 90 65	360 180 130	ns
Output Fall Time t_{THL} = (1.5 ns/pF) C_L + 25 ns t_{THL} = (0.75 ns/pF) C_L + 12.5 ns t_{THL} = (0.55 ns/pF) C_L + 12.5 ns	t_{THL}	5.0 10 15	— — —	100 50 40	200 100 80	ns
Propagation Delay Time Reset to Decade Output t_{PLH}, t_{PHL} = (1.7 ns/pF) C_L + 415 ns t_{PLH}, t_{PHL} = (0.66 ns/pF) C_L + 197 ns t_{PLH}, t_{PHL} = (0.5 ns/pF) C_L + 150 ns	t_{PLH}, t_{PHL}	5.0 10 15	— — —	500 230 175	1000 460 350	ns
Propagation Delay Time Clock to C_{out} t_{PLH}, t_{PHL} = (1.7 ns/pF) C_L + 315 ns t_{PLH}, t_{PHL} = (0.66 ns/pF) C_L + 142 ns t_{PLH}, t_{PHL} = (0.5 ns/pF) C_L + 100 ns	t_{PLH}, t_{PHL}	5.0 10 15	— — —	400 175 125	800 350 250	ns
Propagation Delay Time Clock to Decode Output t_{PLH}, t_{PHL} = (1.7 ns/pF) C_L + 415 ns t_{PLH}, t_{PHL} = (0.66 ns/pF) C_L + 197 ns t_{PLH}, t_{PHL} = (0.5 ns/pF) C_L + 150 ns	t_{PLH}, t_{PHL}	5.0 10 15	— — —	500 230 175	1000 460 350	ns
Turn-Off Delay Time Reset to C_{out} t_{PLH} = (1.7 ns/pF) C_L + 315 ns t_{PLH} = (0.66 ns/pF) C_L + 142 ns t_{PLH} = (0.5 ns/pF) C_L + 100 ns	t_{PLH}	5.0 10 15	— — —	400 175 125	800 350 250	ns
Clock Pulse Width	t_{WH}	5.0 10 15	250 100 75	125 50 35	— — —	ns
Clock Frequency	f_{cl}	5.0 10 15	— — —	5.0 12 16	2.0 5.0 6.7	MHz
Reset Pulse Width	t_{WH}	5.0 10 15	500 250 190	250 125 95	— — —	ns
Reset Removal Time	t_{rem}	5.0 10 15	750 275 210	375 135 105	— — —	ns
Clock Input Rise and Fall Time	t_{TLH}, t_{THL}	5.0 10 15	No Limit			—
Clock Enable Setup Time	t_{su}	5.0 10 15	350 150 115	175 75 52	— — —	ns
Clock Enable Removal Time	t_{rem}	5.0 10 15	420 200 140	260 100 70	— — —	ns

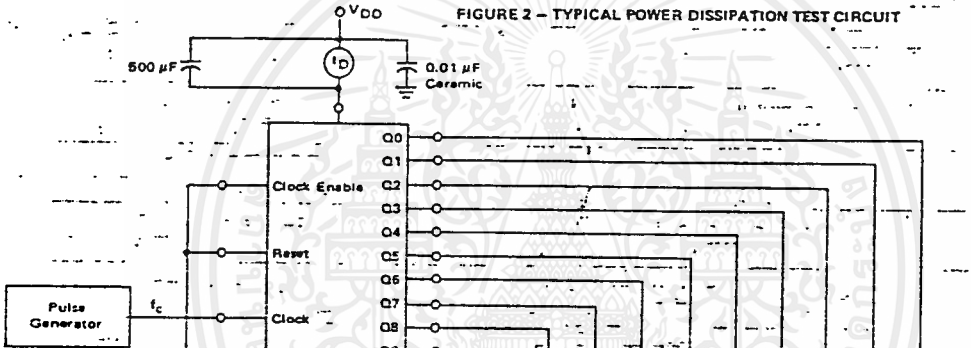
*The formula given is for the typical characteristics only.

FIGURE 1 – TYPICAL OUTPUT SOURCE AND OUTPUT SINK CHARACTERISTICS TEST CIRCUIT



	OUTPUT SINK DRIVE	OUTPUT SOURCE DRIVE
DECODE OUTPUTS	(S1 to A)	Clock to desired outputs (S1 to B)
Carry	Clock to 5 thru 9 (S1 to B)	S1 to A
V_{GS} =	V_{DD}	$-V_{DD}$
V_{DS} =	V_{out}	$V_{out} - V_{DD}$

FIGURE 2 – TYPICAL POWER DISSIPATION TEST CIRCUIT



APPLICATIONS INFORMATION

Figure 3 shows a technique for extending the number of decoded output states for the MC14017B. Decoded outputs are sequential within each stage and from stage to stage, with no dead time (except propagation delay).

FIGURE 3 – COUNTER EXPANSION

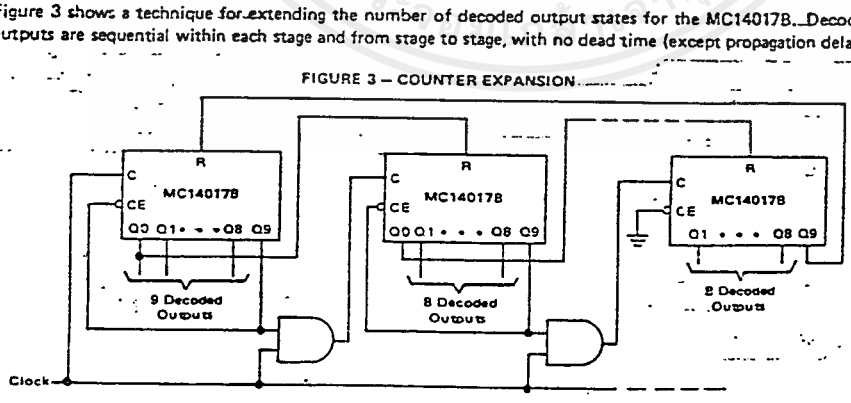


FIGURE 4 – AC MEASUREMENT DEFINITION AND FUNCTIONAL WAVEFORMS